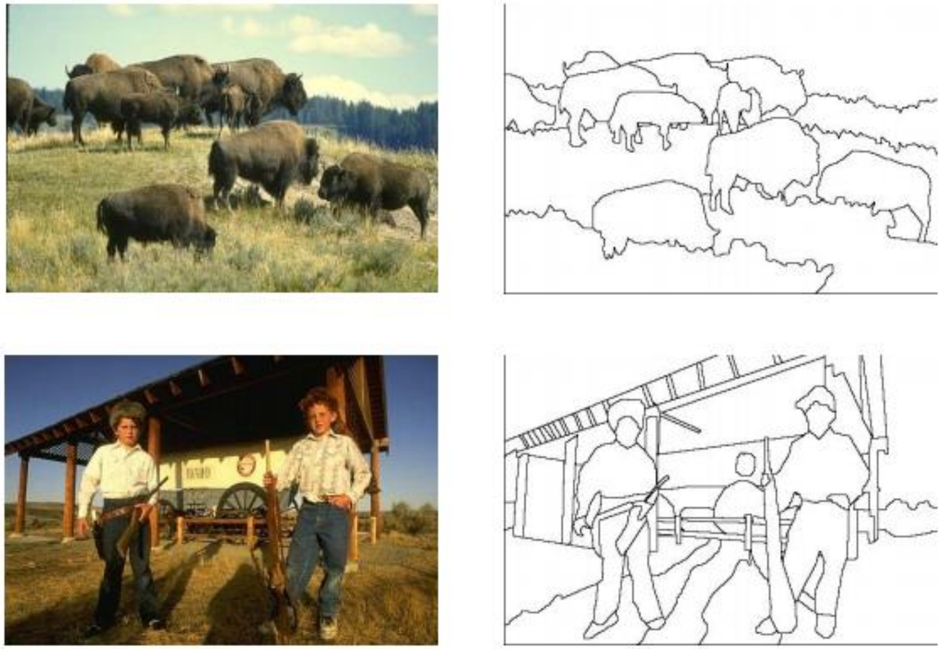
Image Segmentation

CSE473/573

Segmentation of Coherent Regions



Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

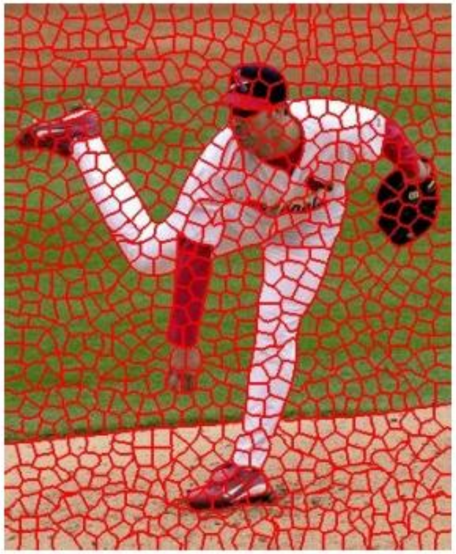
Figure-Ground Segmentation

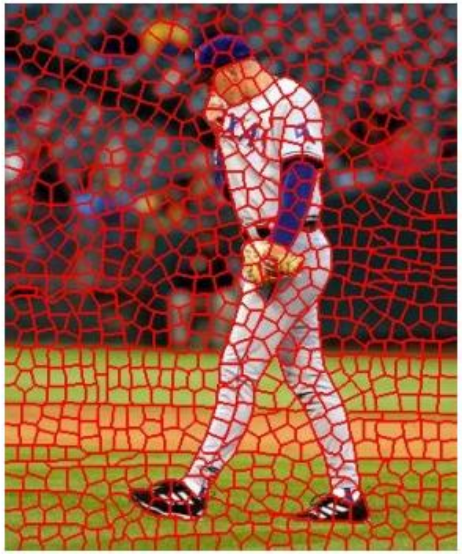
- Separate the foreground object (figure) from the background (ground)



D. Tsai, M. Flagg, and J. M. Rehg. "Motion Coherent Tracking with Multi-label MRF optimization." BMVC 2010.

Grouping of Similar Neighbors





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

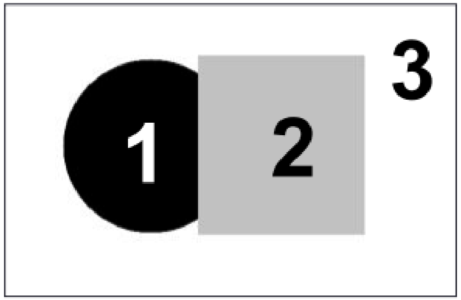
Slide by Svetlana Lazebnik

Extension beyond Single Images

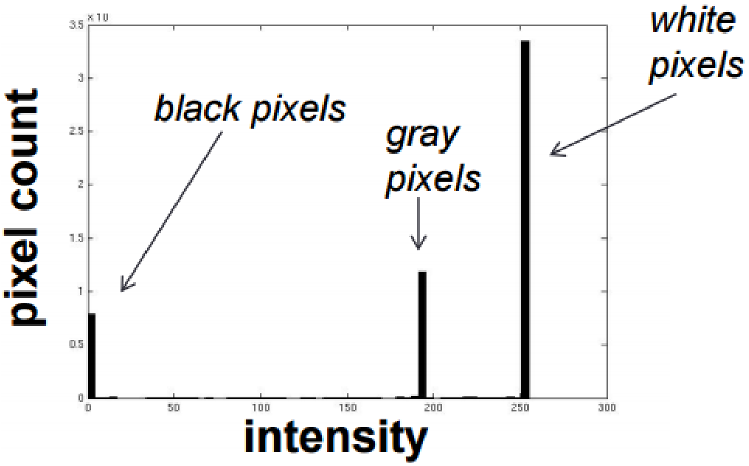
J. Strom, A. Richardson, E. Olson. "Graph-based Segmentation for Colored 3D Laser Point Clouds." IROS 2010.

M. Grundmann, V. Kwatra, M. Han, I. Essa. "Efficient Hierarchical Graph-Based Video Segmentation." CVPR 2010.

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?

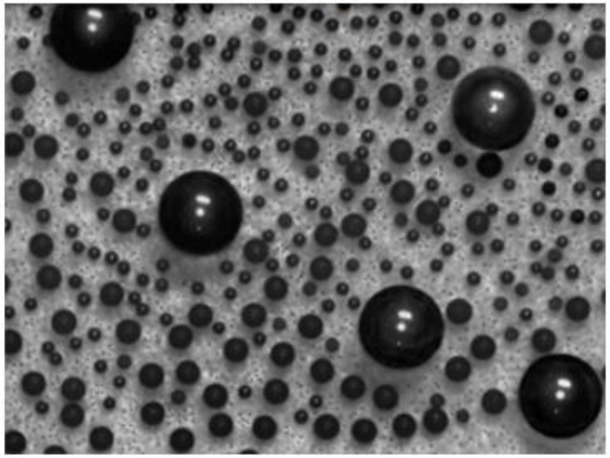
Image Histograms

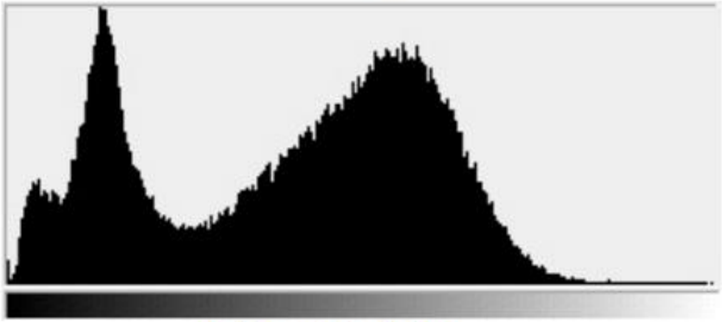
How many "orange" pixels are in this image?

- This type of question answered by looking at the *histogram*
- A histogram counts the number of occurrences of each color
  - Given an image $F[x, y] \rightarrow RGB$
  - The histogram is $H_F[c] = |\{(x, y) \mid F[x, y] = c\}|$

    i.e., for each color value c (x-axis), count # of pixels with that color (y-axis)

Histograms of Grayscale intensities





How Many Modes Are There?

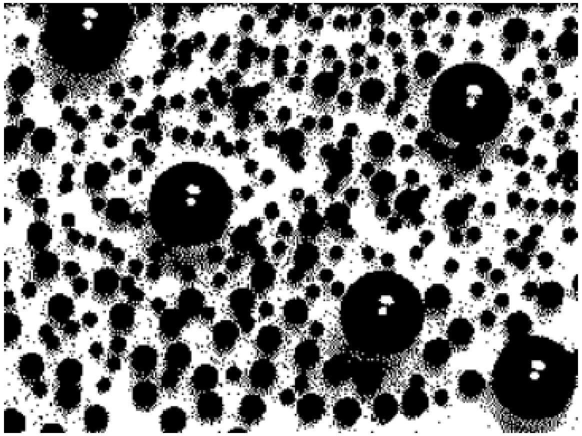
- Easy to see, hard to compute

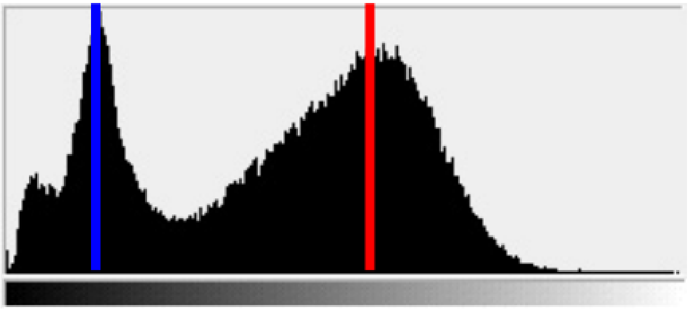
Histograms of Grayscale intensities

Idea: Break the image into K regions (segments) by

- reducing the number of colors to K and
- assigning each pixel to the closest color

Here's what our image looks like if we use two colors (intensities)





Segmentation as Clustering - Issues

Is broccoli same as a pepper?

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes

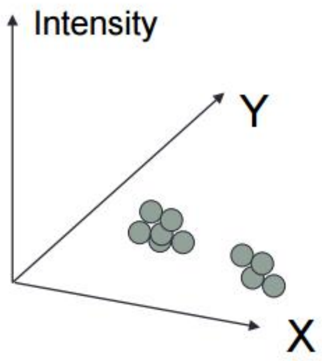
Image | Intensity-based clusters | Color-based clusters

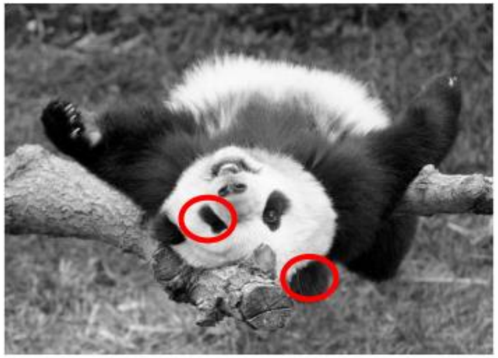
Slide by Svetlana Lazebnik

Segmentation as Clustering - Improvement

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

Grouping pixels based on **intensity+position** similarity



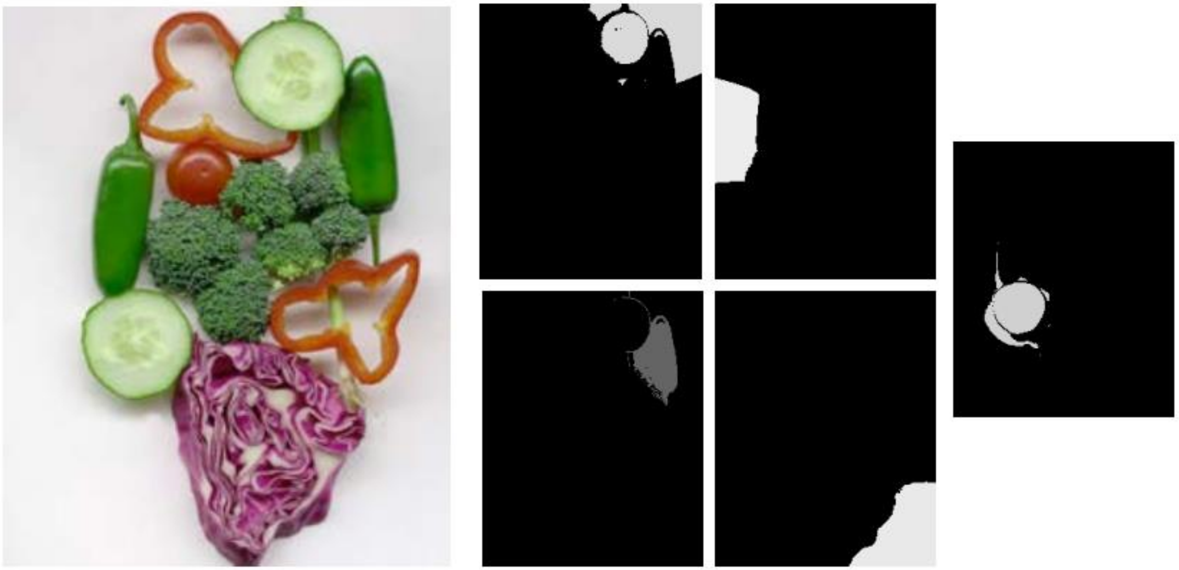


Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Kristen Grauman

Segmentation as Clustering - Improvement

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



Slide by Svetlana Lazebnik

Segmentation as Clustering - Challenges

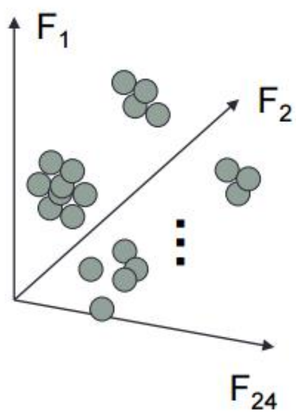
- Color, brightness, position alone are not enough to distinguish all regions…

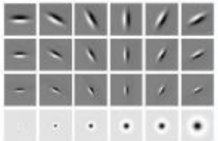


Segmentation as Clustering

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

Grouping pixels based on **texture** similarity





Filter bank of 24 filters

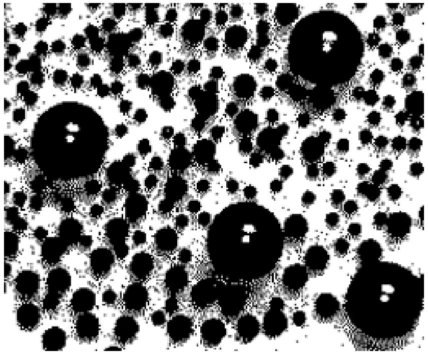
Feature space: filter bank responses (e.g., 24-d)

Kristen Grauman

Cleaning up after segmentation

Problem:

- Histogram-based segmentation can produce messy regions
  - segments do not have to be connected
  - may contain holes



How can these be fixed?

Use Morphological operators!

Image Dilation

Assume: binary image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	0	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

1	1	1
1	1	1
1	1	1

$H[u, v]$

Dilation: Move mask H over image F, turning F's pixels to 1 if F and H both have 1s *anywhere* in the region of overlap

- G[x,y] = 1 if H[u,v] and F[x+u-1,y+v-1] are both 1 **somewhere**
  0 otherwise
- Written as $G = H \oplus F$

Image Dilation

Assume: binary image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

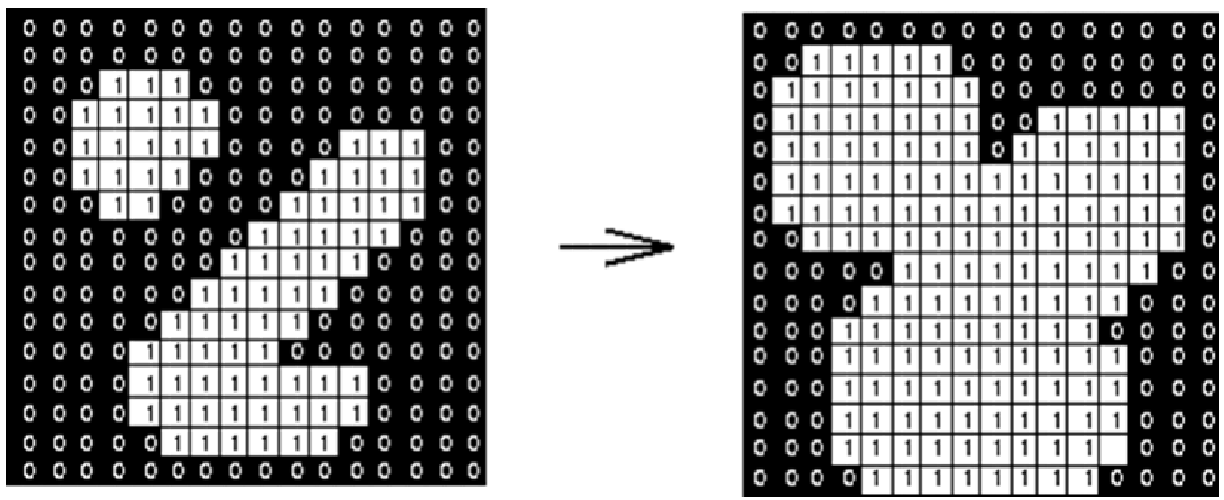
1	1	1
1	1	1
1	1	1

$H[u, v]$

Dilation: Move mask H over image F, turning F's pixels to 1 if F and H have 1s *anywhere* in the region of overlap

- G[x,y] = 1 if H[u,v] and F[x+u-1,y+v-1] are both 1 **somewhere**
  0 otherwise
- Written as $G = H \oplus F$

Image Dilation - Example



Demo: http://www.cs.bris.ac.uk/~majid/mengine/morph.html

Image Erosion

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	0	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

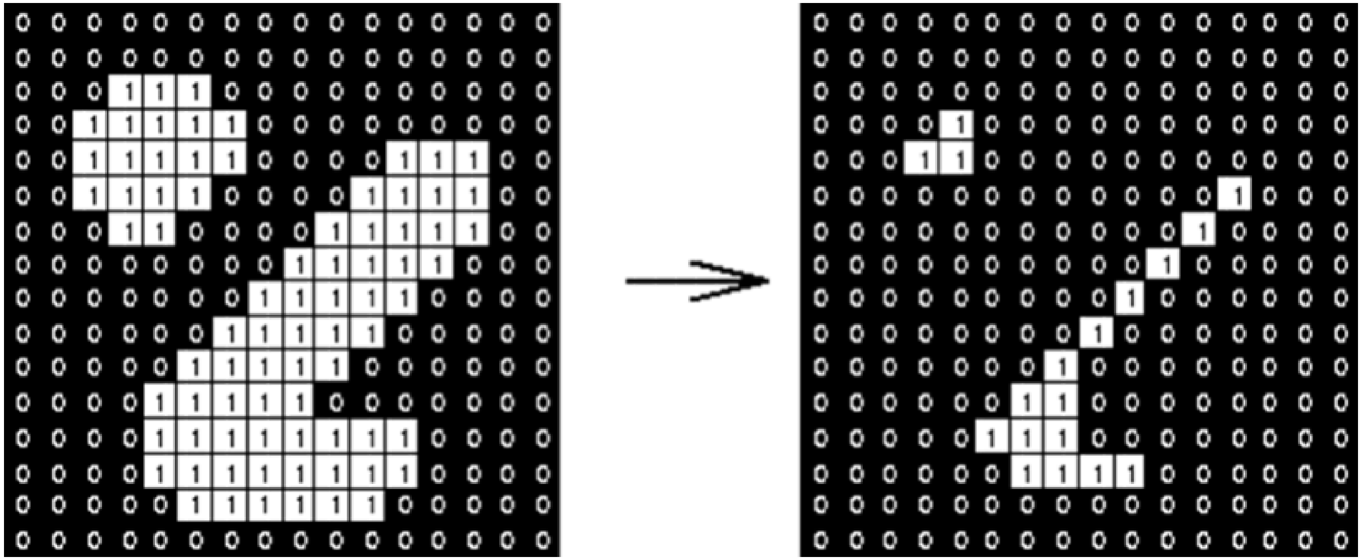
1	1	1
1	1	1
1	1	1

$H[u, v]$

Erosion: Move mask H over image F, turning F's pixels to 1 if F and H both have 1s *everywhere* in the region of overlap

- G[x,y] = 1 if F[x+u-1,y+v-1] is 1 **everywhere** that H[u,v] is 1
  0 otherwise
- Written $G = H \ominus F$

Image Erosion

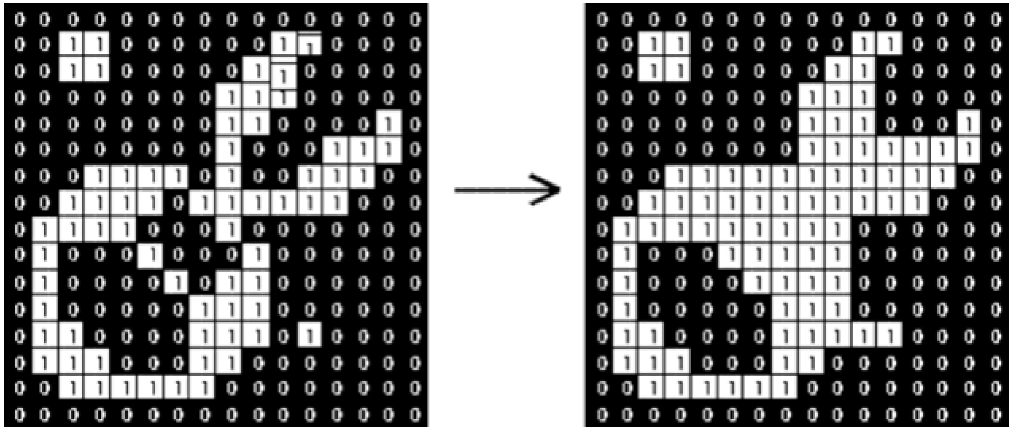


Demo: http://www.cs.bris.ac.uk/~majid/mengine/morph.html

Nested Dilations and Erosions

What does this operation do?

$$G = H \ominus (H \oplus F)$$



- This is called a **closing** operation

Is this the same thing as the following?

$$G = H \oplus (H \ominus F)$$

Nested Dilations and Erosions

What does this operation do?

$$G = H \oplus (H \ominus F)$$

- This is called an **opening** operation

http://www.dai.ed.ac.uk/HIPR2/open.htm

You can clean up binary images (e.g., results of segmentation) by applying combinations of dilations and erosions Dilations, erosions, opening, and closing operations are known as morphological operations

see http://www.dai.ed.ac.uk/HIPR2/morops.htm

Mean-shift Segmentation

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

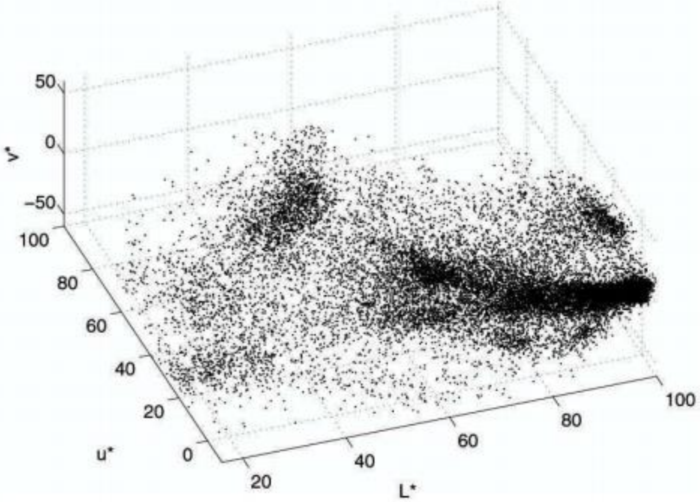
image

Mean-shift Segmentation

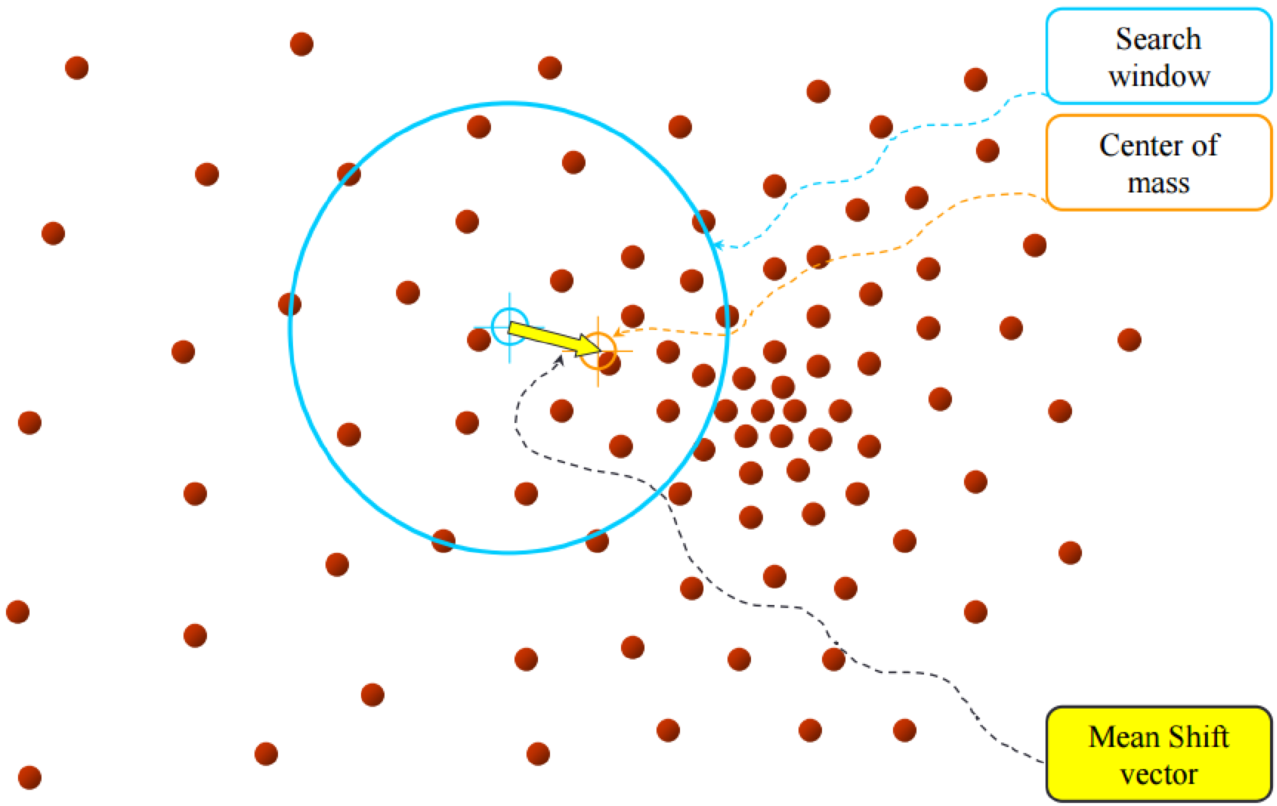
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

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

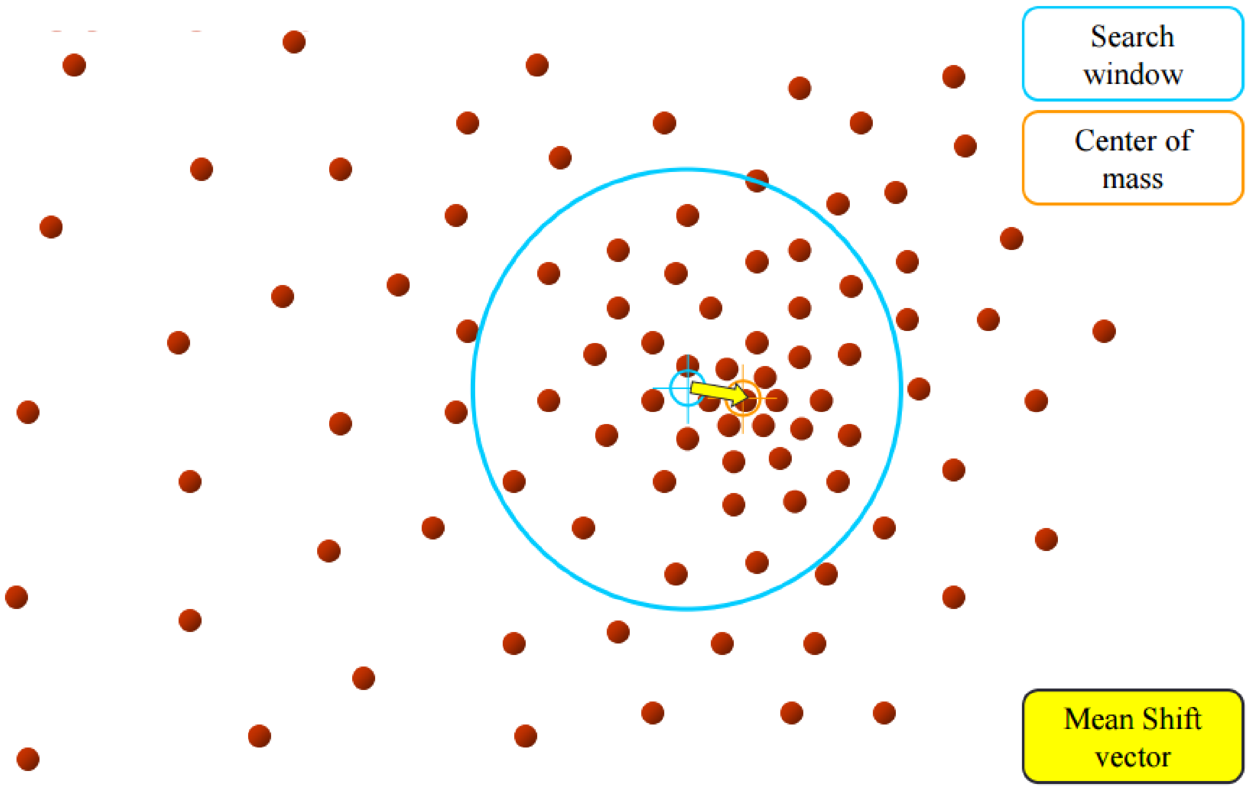


Mean-Shift



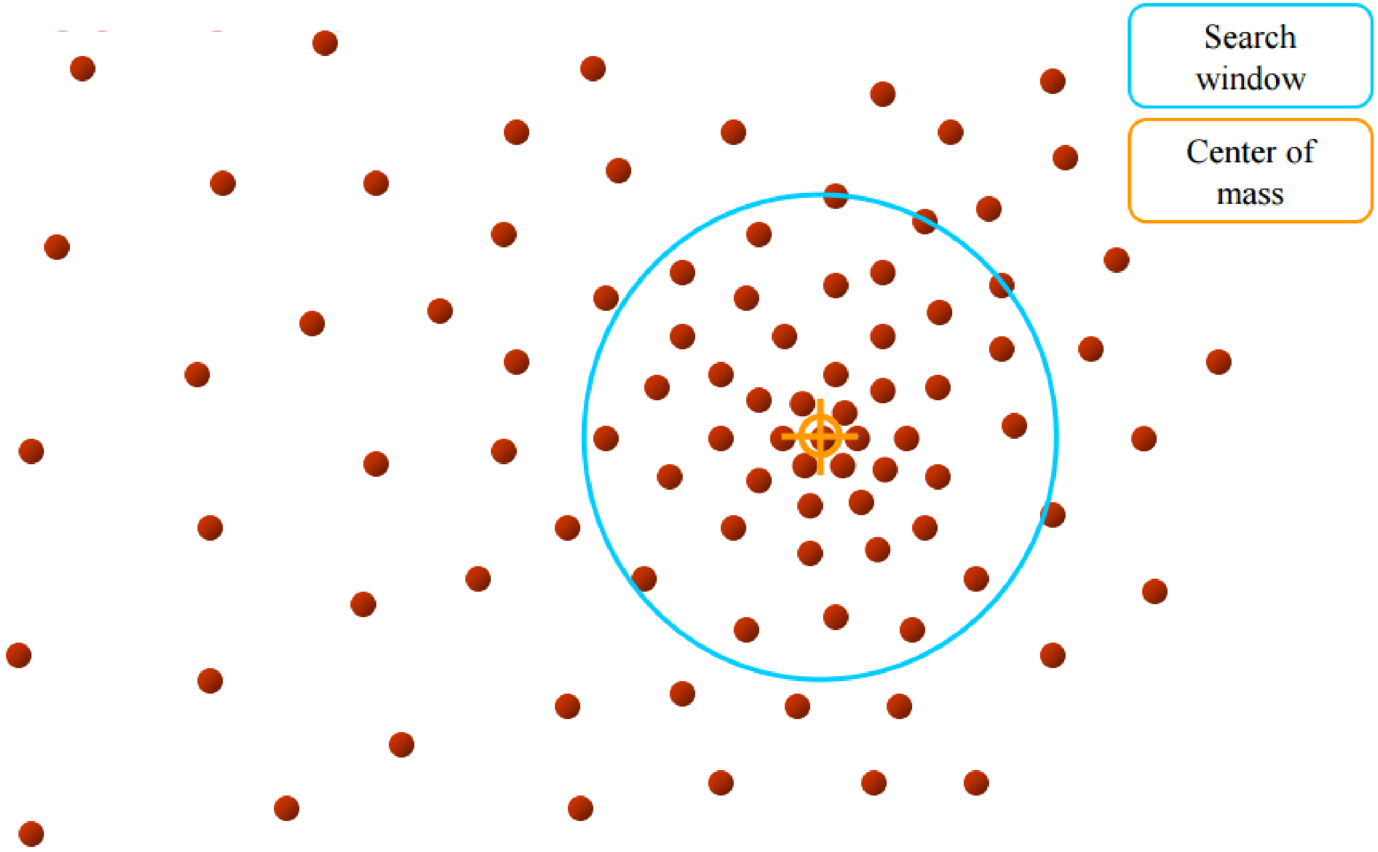
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



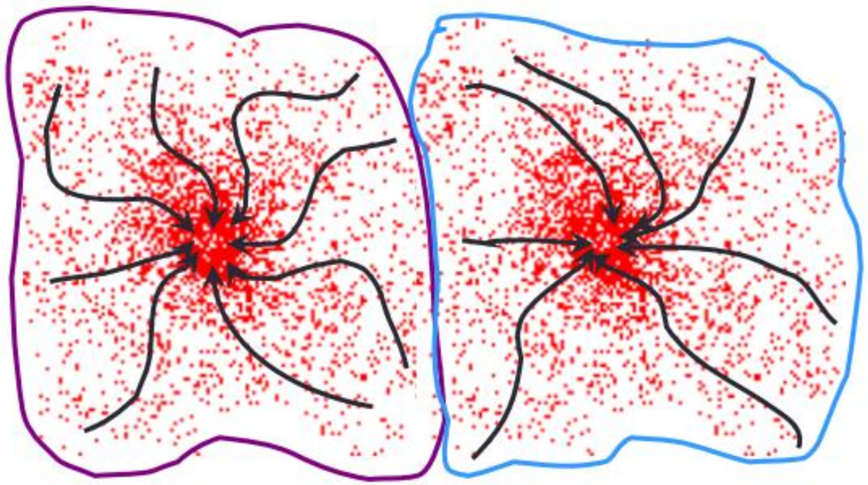
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



Mean-Shift Clustering

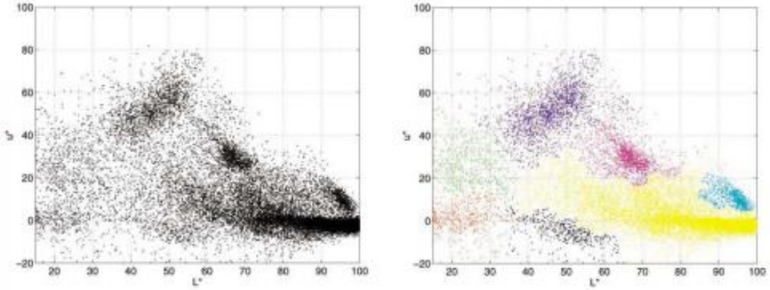
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



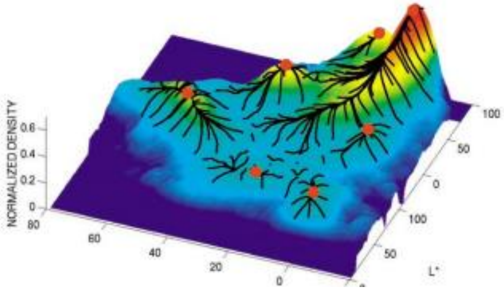
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Segmentation

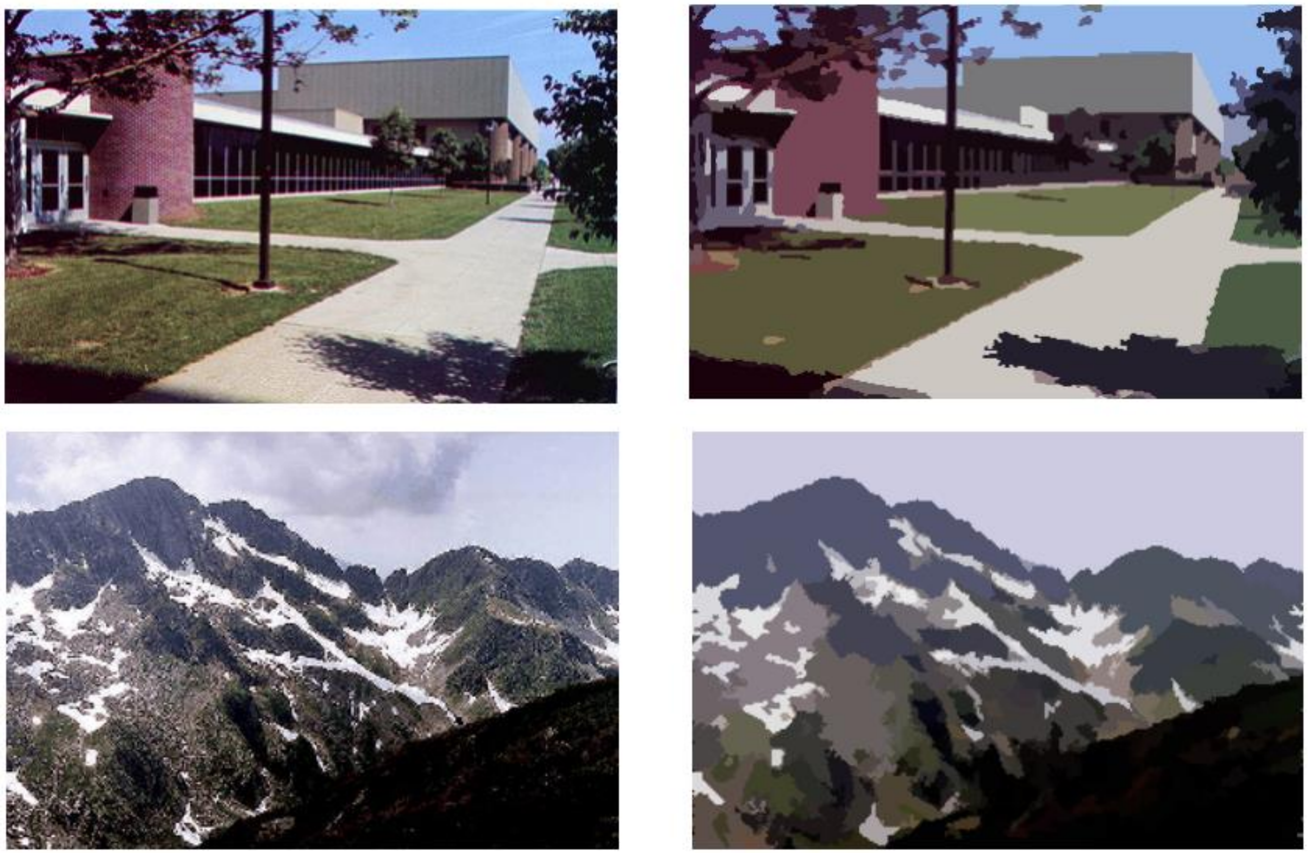
- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points (pixels)
- Perform mean shift for each window (pixel) until convergence
- Merge windows (pixels) that end up near the same “peak” or mode



(a) (b)

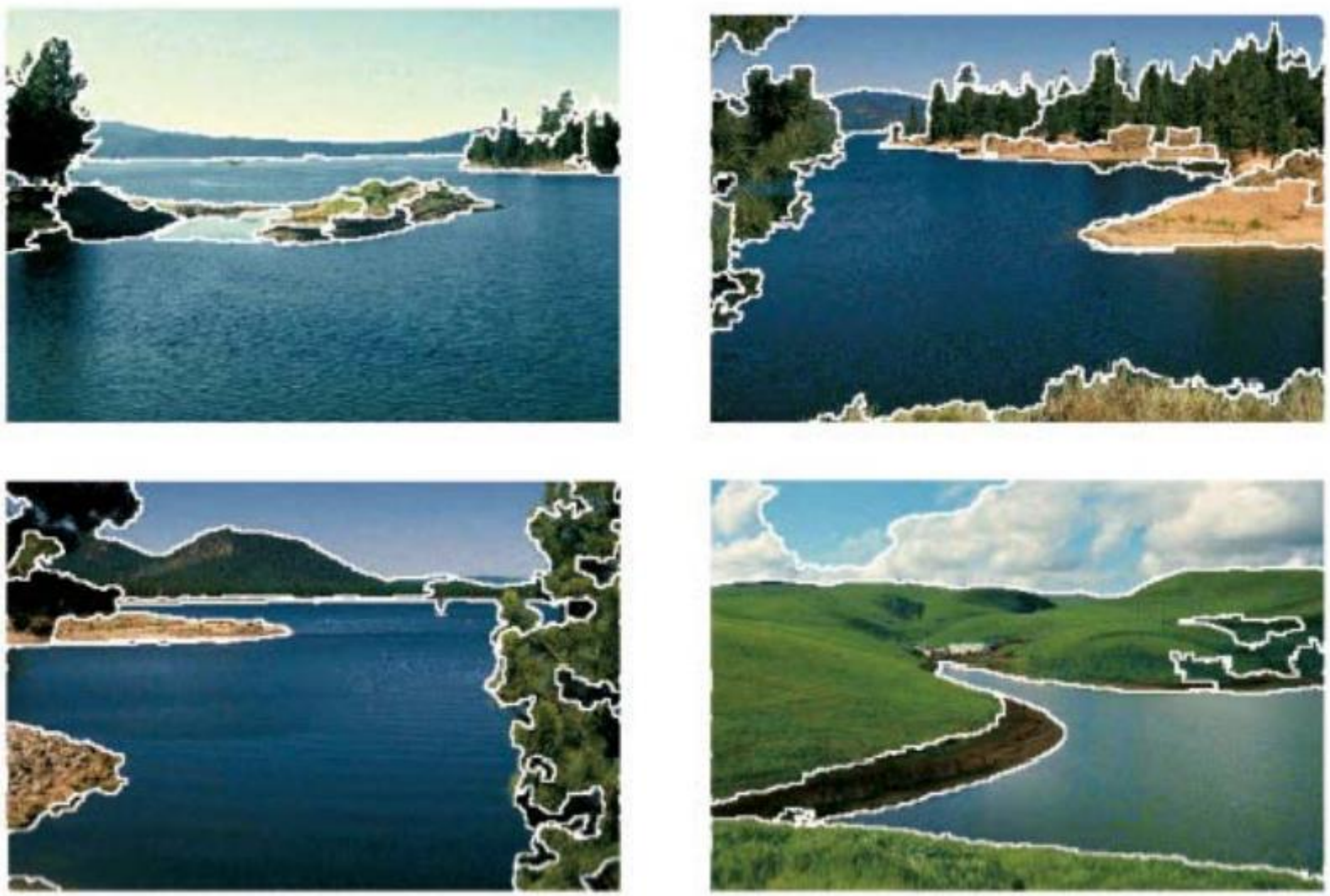


Mean-Shift Segmentation Results



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

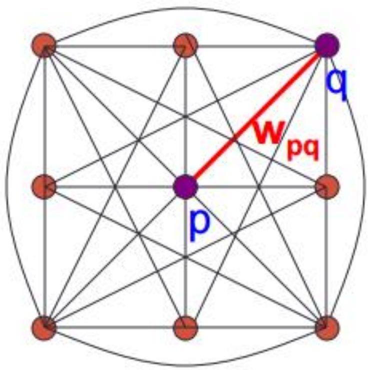
Mean-Shift Segmentation Results

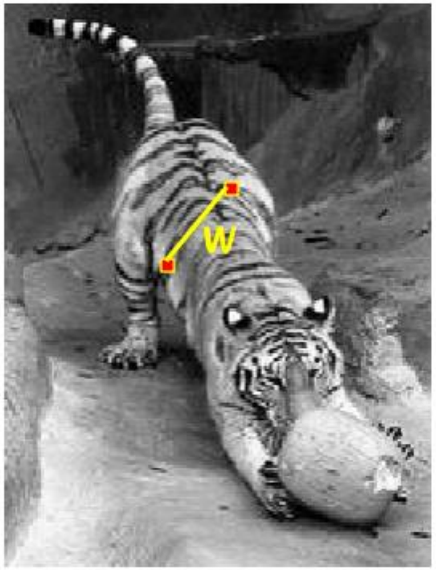


Mean-Shift Segmentation

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size)
  - Generic technique
  - Find multiple modes
- Cons:
  - Selection of window size
  - Does not scale well with dimension of feature space

Images as Graphs





- *Fully-connected* graph
  - node (vertex) for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - affinity weight $\mathbf{w_{pq}}$ for each link (edge)
    - $\mathbf{w_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference (in color and position…)

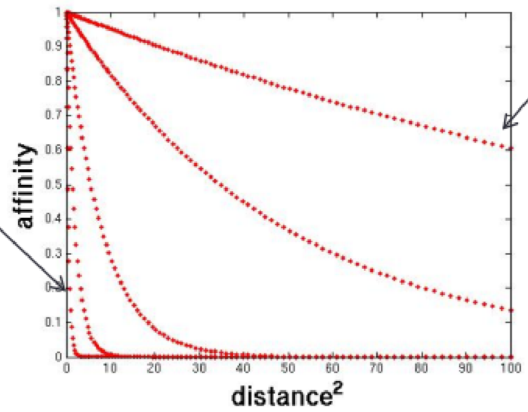
Source: Steve Seitz

Measuring Affinity

- One possibility:

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

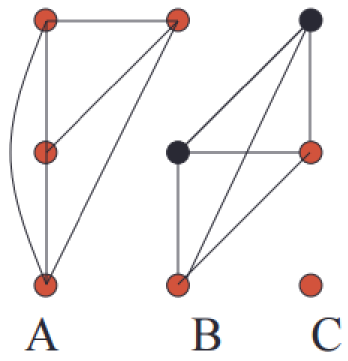
Small sigma: group only nearby points



Large sigma: group distant points

Kristen Grauman

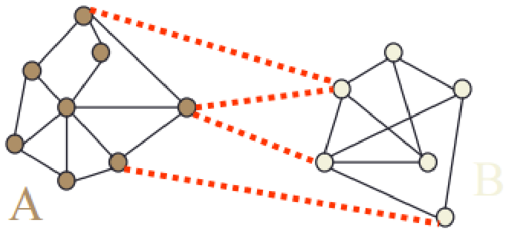
Segmentation by Graph Partitioning



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

Source: S. Seitz

Segmentation by Graph Partitioning



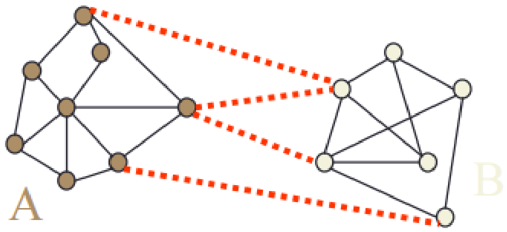
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges

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

- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?

Source: S. Seitz

Segmentation by Graph Partitioning



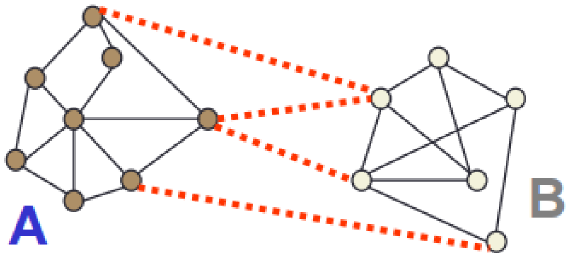
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges

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

- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

Source: S. Seitz

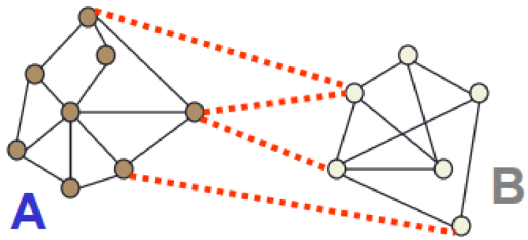
Min Cut



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

Find minimum cut
- gives you a segmentation

Normalized Cut



Normalized Cut

- fix bias of Min Cut by **normalizing** for size of segments:

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

assoc(A,V) = sum of weights of all edges that touch A

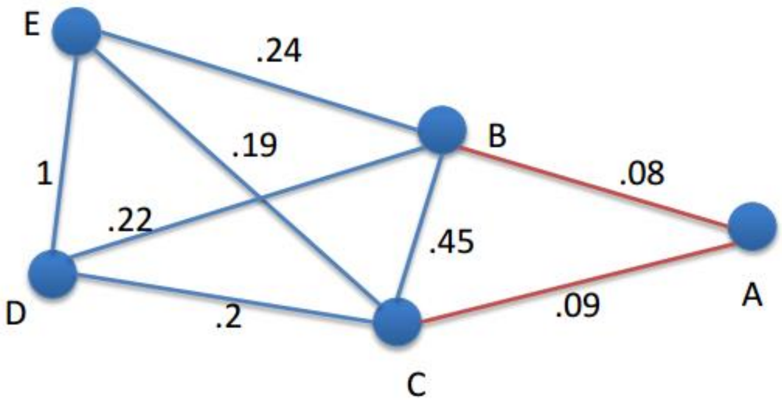
- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Source: Steve Seitz

Min Cut - Example

- Minimum Cut

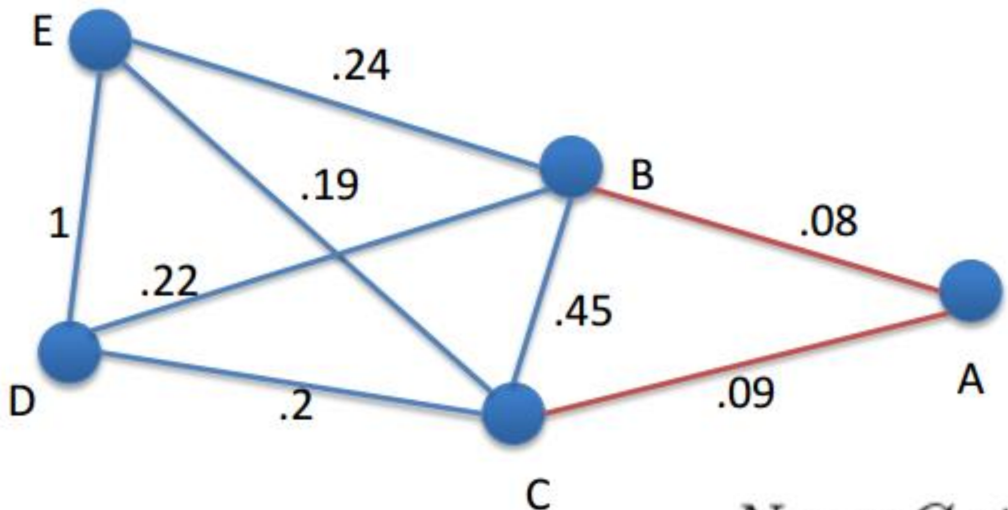


$$Cut(BCDE, A) = 0.17$$

Normalized Cut - Example

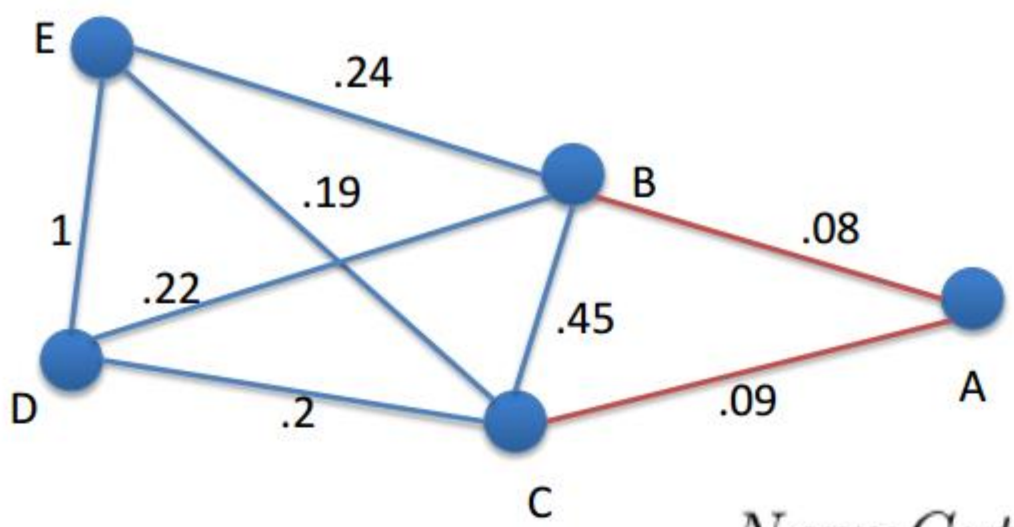
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

assoc(A,V) = sum of weights of all edges that touch A



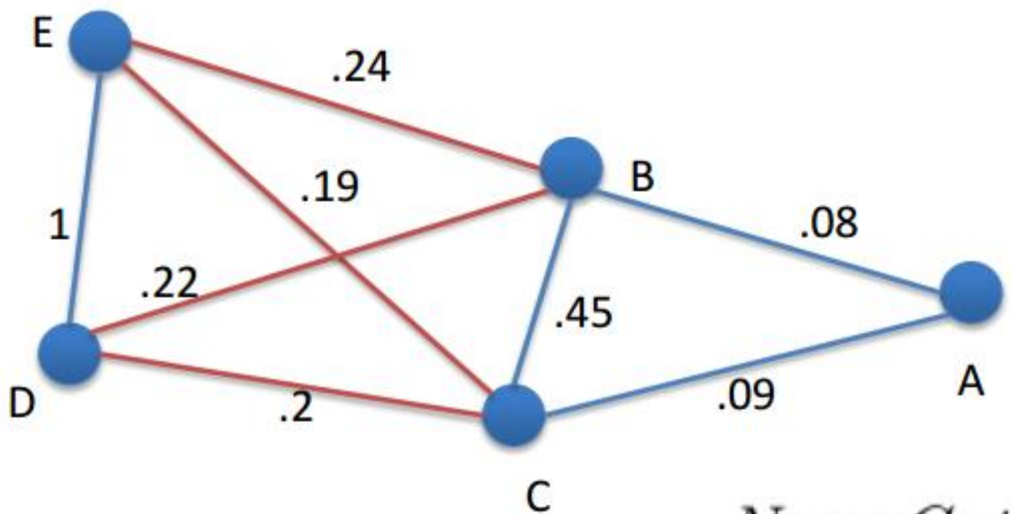
$$NormCut(BCDE, A) =$$

Normalized Cut - Example



$$NormCut(BCDE, A) = 1.06$$

Normalized Cut – Generalizes better



$$NormCut(BCDE, A) = 1.06$$

$$NormCut(ABC, DE) = 1.03$$

Graph Cut - Results

http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized Cuts – Pros and Cons

Pros:

- Generic framework, flexible to choice of function that computes weights ("affinities") between nodes
- Does not require model of the data distribution

Cons:

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

Normalized Cuts – Pros and Cons

<u>Pros:</u>

- Generic framework, flexible to choice of function that computes weights (“affinities”) between nodes
- Does not require model of the data distribution

<u>Cons:</u>

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions