Segmentation

CSE P 576

Larry Zitnick (larryz@microsoft.com)

Many slides courtesy of Steve Seitz

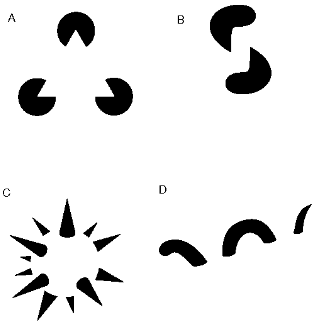
Human segmentations

Berkeley segmentation dataset, 5 subjects

Occlusion is an important cue in grouping.

Courtesy of Bill Freeman, MIT.

Consequence: Groupings by Invisible Completions



* Images from Steve Lehar's Gestalt papers: http://cns-alumni.bu.edu/pub/slehar/Lehar.html

And the famous…

Gestalt psychology

"The whole is greater than the sum of the parts"

The Law of Proximity: Stimulus elements that are closed together tend to be perceived as a group

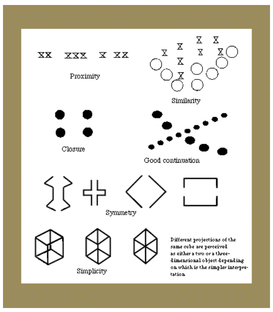
The Law of Similarity: Similar stimuli tend to be grouped, this tendency can even dominate grouping due to proximity

The Law of Closure: Stimuli tend to be grouped into complete figures

The Law of Good Continuation: Stimuli tend to be grouped as to minimize change or discontinuity

The Law of Symmetry: Regions bound by symmetrical boarders tend to be perceived as coherent figures

The Law Simplicity: Ambiguous stimuli tend to be resolved in favor of the simplest



Fischler and Firshein, 1987

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

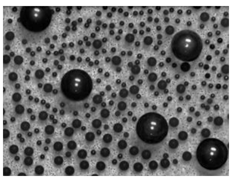
$$F[x, y] \rightarrow RGB$$

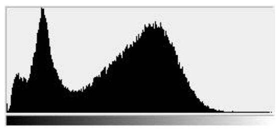
  - The histogram is defined to be

$$H_F[c] = |\{(x, y) \mid F[x, y] = c\}|$$

  - What is the dimension of the histogram of an NxN RGB image?

What do histograms look like?





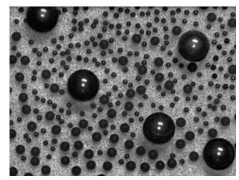
How Many Modes Are There?

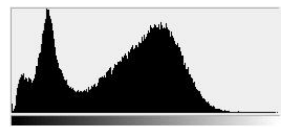
- Easy to see, hard to compute

Histogram-based segmentation

Goal

- Break the image into K regions (segments)
- Solve this by reducing the number of colors to K and mapping each pixel to the closest color

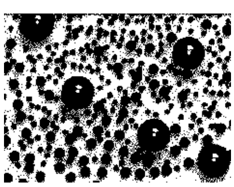


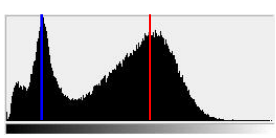


Histogram-based segmentation

Goal

- Break the image into K regions (segments)
- Solve this by reducing the number of colors to K and mapping each pixel to the closest color





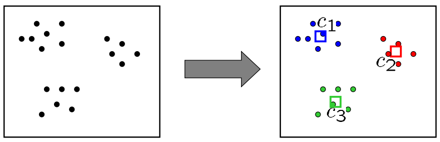
Here's what it looks like if we use two colors

K-means

Clustering

How to choose the representative colors?

- This is a clustering problem!



Objective

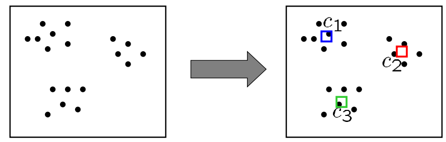
- Each point should be as close as possible to a cluster center
  - Minimize sum squared distance of each point to closest center

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

Break it down into subproblems

Suppose I tell you the cluster centers c_i

- Q: how to determine which points to associate with each c_i?
- A: for each point p, choose closest c_i



Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

K-means clustering

K-means clustering algorithm

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

Java demo: http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletKM.html

Properties

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

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

Probabilistic clustering

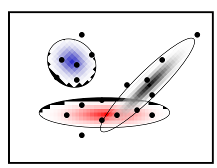
Basic questions

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

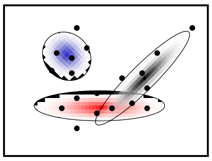
K-means doesn't answer these questions

Probabilistic clustering (basic idea)

- Treat each cluster as a Gaussian density function



Expectation Maximization (EM)



A probabilistic variant of K-means:

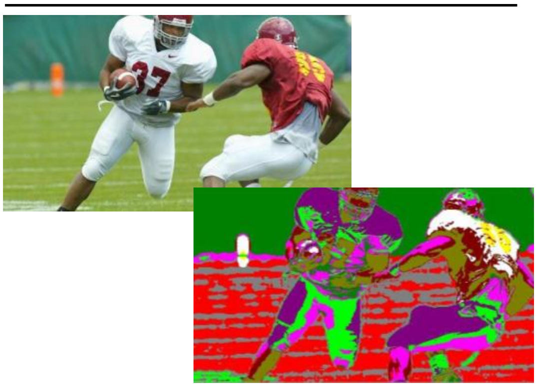
- E step: "soft assignment" of points to clusters
  - estimate probability that a point is in a cluster
- M step: update cluster parameters
  - mean and variance info (covariance matrix)
- maximizes the likelihood of the points given the clusters

Applications of EM

Turns out this is useful for all sorts of problems

- any clustering problem
- model estimation with missing/hidden data
- finding outliers
- segmentation problems
  - segmentation based on color
  - segmentation based on motion
  - foreground/background separation
- ...

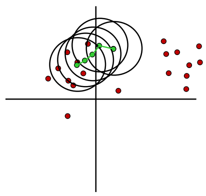
Example



Mean-shift

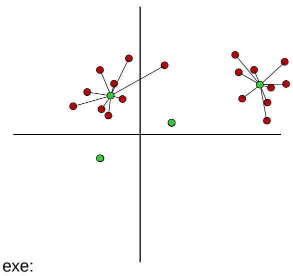
Mean-shift

Good if you don’t know the number of clusters, but you do know the variance between points in the same cluster.



Mean-shift

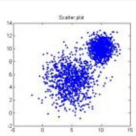
All points that end up in the same location are clustered together.

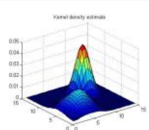


Example exe:

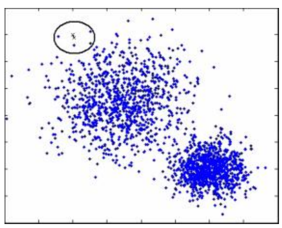
http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

More examples





Climbing up the hill…



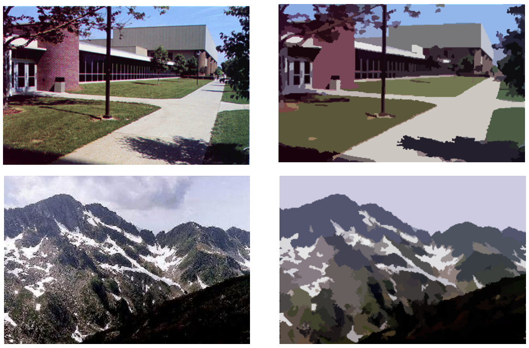
Clustering spatially

What if you want to include spatial relationships?

Cluster in 5 dimensions instead of 3:

$$\begin{bmatrix} r \\ g \\ b \\ x \\ y \end{bmatrix} \qquad \begin{bmatrix} r \\ g \\ b \end{bmatrix}$$

Mean Shift color&spatial Segmentation Results:



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

Fast method

Fast and dirty

What if you just want to group similar neighboring pixels?

1	1	1	1	1	1	1
1	9	1	1	1	13	1
1	9	1	1	1	13	1
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42

Step 1:

	B
C	A

A = min(S), where S is the set of similar pixels in {A, B, C}. A is always in S.

Fast and dirty

What if you just want to group similar neighboring pixels?

1	1	1	1	1	1	1
1	9	1	1	1	9	1
1	9	1	1	1	9	1
1	9	1	1	1	9	1
1	9	9	9	9	9	1
1	1	1	1	1	1	1

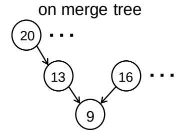
Step 1:

	B
C	A

A = min(S), where S is the set of similar pixels in {A, B, C}. A is always in S.

Step 2:

Reassign clusters based on merge tree



Dilation operator: $G = H \oplus F$

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

$F[x, y]$

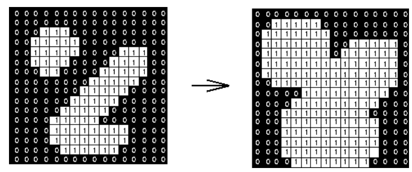
1	1	1
1	1	1
1	1	1

$H[u, v]$

Dilation: does H “overlap” F around [x,y]?

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

Dilation operator



Erosion operator: $G = H \ominus F$

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

$F[x, y]$

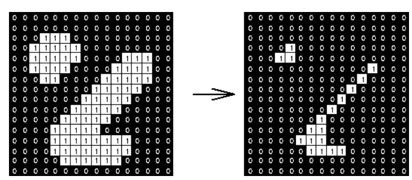
1	1	1
1	1	1
1	1	1

$H[u, v]$

Erosion: is H “contained in” F around [x,y]

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

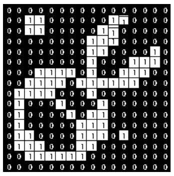
Erosion operator



Nested dilations and erosions

What does this operation do?

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

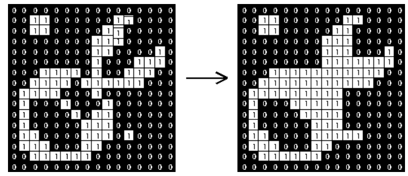


- this is called a **closing** operation

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

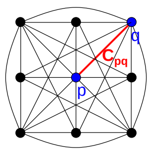
You can clean up binary pictures by applying combinations of dilations and erosions

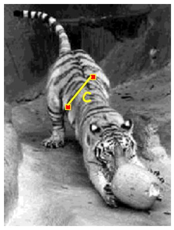
Dilations, erosions, opening, and closing operations are known as **morphological operations**

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

Normalized Cuts

Images as graphs

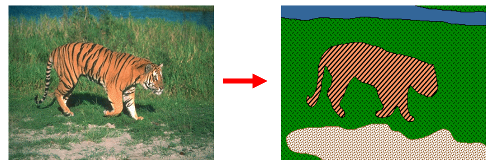




Fully-connected graph

- node for every pixel
- link between *every* pair of pixels, **p,q**
- cost c_{pq} for each link
  - c_{pq} measures *similarity* of pixels

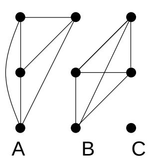
Graph-based segmentation?



Similarity can be measured in many ways:

- Color
- Position
- Texture
- Motion
- Depth
- Etc.

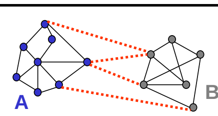
Segmentation by Normalized Cuts



Break Graph into Segments

- Delete links that cross between segments
- Easiest to break links that have low cost (not similar)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Cuts in a graph



Link Cut

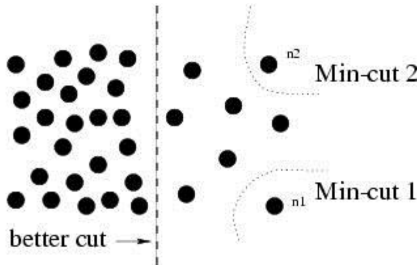
- set of links whose removal makes a graph disconnected
- cost of a cut:

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

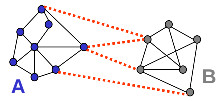
Find minimum cut

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

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



Cuts in a graph



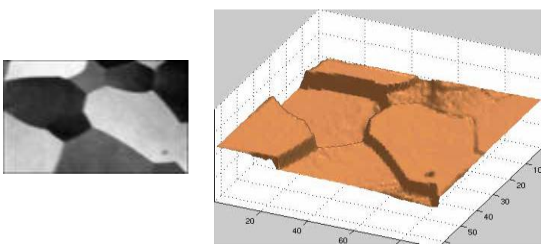
Normalized Cut

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

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

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

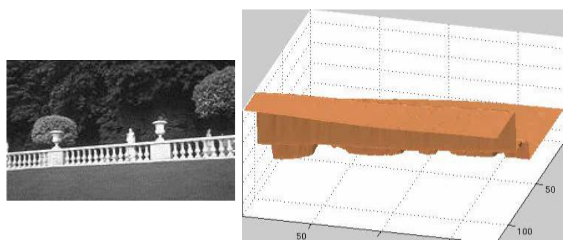
Interpretation as a Dynamical System



Treat the links as springs and shake the system

- elasticity proportional to cost
- vibration “modes” correspond to segments
  - can compute these by solving an eigenvector problem

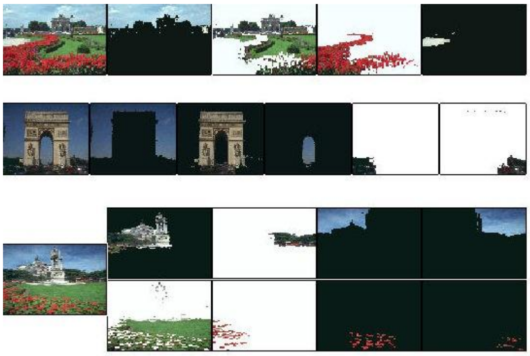
Interpretation as a Dynamical System



Treat the links as springs and shake the system

- elasticity proportional to cost
- vibration “modes” correspond to segments
  - can compute these by solving an eigenvector problem

Color Image Segmentation



Which is best?

Both mean-shift and normalized cuts are popular.

Normalized cuts can handle complex similarly measures.

Both do a better job than K-means at segmenting areas of smoothly varying intensities.