Segmentation I
Goal

Separate image into coherent “regions”

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

"Intelligent scissors"

Foreground / background segmentation

Finding skin-colored regions

Finding the moving objects

Finding the cars in a video sequence

Semantics
Questions

What is coherent?
Questions

What is coherent?

• Spatial proximity?
• Similar color?
• Similar texture?
Questions

What is coherent?
- Spatial proximity?
- Similar color?
- Similar texture?

What kinds of regions?
Questions

What is coherent?

• Spatial proximity?
• Similar color?
• Similar texture?

What kinds of regions?

• Nearly convex?
• Smooth boundaries?
• Nearly equal sizes?
• What granularity?
Grouping Cues

Gestalt factors:

- Not grouped
- Proximity
- Similarity
- Similarity
- Common Fate
- Common Region
- Parallelism
- Symmetry
- Continuity
- Closure
A Typical Segmentation Problem

Partition an image into arbitrarily shaped regions containing pixels with similar colors and positions.
Segmentation Algorithms?

What kinds of algorithm(s) can solve this problem?
Some Segmentation Algorithms

- Divisive clustering
- Hierarchical clustering
- K-means clustering
- Mean shift clustering
- Graph cuts

More next time …
Segmentation as Clustering

Segmentation can be treated as a clustering problem
Divisive Clustering

Start with whole image in one cluster

Iterate:
- Find cluster with largest intra-cluster variation
- Split into two pieces that yield largest inter-cluster distance

Stopping criteria?
Divisive Clustering

Start with whole image in one cluster

Iterate:
• Find cluster with largest intra-cluster variation
• Split into two pieces that yield largest inter-cluster distance

Stopping criteria
Divisive Clustering

Start with whole image in one cluster

Iterate:
• Find cluster with largest intra-cluster variation
• Split into two pieces that yield largest inter-cluster distance

Stopping criteria
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:
- Find pair of clusters with smallest inter-cluster distance
- Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:
  • Find pair of clusters with smallest inter-cluster distance
  • Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:
- Find pair of clusters with smallest inter-cluster distance
- Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:

- Find pair of clusters with smallest inter-cluster distance
- Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:
- Find pair of clusters with smallest inter-cluster distance
- Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:

• Find pair of clusters with smallest inter-cluster distance
• Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:
- Find pair of clusters with smallest inter-cluster distance
- Merge

Stopping criteria?
Hierarchical Clustering

Start with each pixel in its own cluster

Iterate:

• Find pair of clusters with smallest inter-cluster distance
• Merge

Stopping criteria?
Hierarchical Clustering

Conservative stopping criteria yields “superpixels”, which can be used as starting point for more complex algorithms
Problems with These Algorithms?
Problems with These Algorithms

Greedy

- Decisions made early in process dictate final result

Making “good” early decisions is hard/expensive

- Many possibilities at each iteration
- Computing “good” merge or split is expensive

Heuristics to speed things up:

- For agglomerative clustering, approximate each cluster by average for distance computations
- For divisive clustering, use summary (histogram) of a region to compute split
Some Segmentation Algorithms

Divisive clustering
Hierarchical clustering
k-means clustering
Mean shift clustering
Graph cuts
More next time …
\textit{k}-Means Clustering

Instead of merging or splitting, start out with the clusters and move them around

1. Pick number of clusters $k$
2. Randomly scatter $k$ “cluster centers” in color space
3. Repeat:
   a. Assign each data point to its closest cluster center
   b. Move each cluster center to the mean of the points assigned to it
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
$k$-Means Clustering
Results of $k$-Means Clustering

Original Image  $k$-means, $k=5$  $k$-means, $k=11$
Results of $k$-Means Clustering

Sample clusters with $k$-means clustering based on color
k-Means Pros and Cons?
k-Means Pros and Cons

Pros

• Very simple method

Cons

• Need to pick K
• Converges to a local minimum
• Sensitive to initialization
• Sensitive to outliers
• Only finds “spherical” clusters

Sensitive to outliers

Spherical clusters

(A): Two natural clusters
(B): k-means clusters

Slide by K. Grauman
Some Segmentation Algorithms

Divisive clustering
Hierarchical clustering
k-means clustering
Mean shift clustering
Graph cuts
More next time …
Mean Shift Clustering

Seek *modes* (peaks) of density in feature space

Image

Feature space (color values)

Slide by Y. Ukrainitz & B. Sarel
Mean Shift Clustering

Algorithm:
- 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
Mean Shift Clustering

Search window
Center of mass
Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean Shift Clustering

Center of mass

Search window

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarek
Mean Shift Clustering

- Search window
- Center of mass
- Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel
Mean Shift Clustering
Mean Shift Clustering

Search window
Center of mass
Mean Shift vector
Mean Shift Clustering
Mean Shift Clustering

Cluster all data points in the attraction basin of a mode

• Separate segment for each mode
• Assign points to segments based on which mode is at the end of their mean shift trajectories
Mean Shift Clustering
Mean Shift Results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html
Mean Shift Results
Mean Shift Pros and Cons

Pros

• Finds variable number of modes
• Does not assume spherical clusters
• Just a single parameter (window size)
• Robust to outliers

Cons

• Output depends on window size
• Computationally expensive
• Does not scale well with dimension of feature space
Some Segmentation Algorithms

- Divisive clustering
- Hierarchical clustering
- k-means clustering
- Mean shift clustering
- Graph cuts
- More next time
Graph Cuts

Create weighted graph:

- Nodes = pixels in image
- Edge between pairs of pixels
- Edge weight = similarity (intensity, color, texture, etc.)
Intuition: partition graph into disconnected segments by removing edges that have low cost (low similarity)

- Similar pixels should be in the same segments
- Dissimilar pixels should be in different segments
Graph Cuts

Graph cut
- Set of links whose removal makes a graph disconnected
- Partitions the graph (defines a segmentation)
- Cut cost = sum of costs of all edges in set

Example by S. Seitz
Graph Cuts

Can define arbitrary similarity function between pixels

Example by L. Lazebnik
Graph Cuts

Simple similarity function:

• 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:

$$\exp \left( -\frac{1}{2\sigma^2} \text{dist}(x_i, x_j)^2 \right)$$
More sophisticated similarity functions:

- Difference of histograms built from properties of pixels in optimizally oriented hemi-circles

\[
\chi^2(g,h) = \frac{1}{2} \sum_i \left( \frac{(g_i - h_i)^2}{g_i + h_i} \right)
\]
More sophisticated similarity functions:
• Similarity based on intervening contours
Now, need to find best cut. How?

• Want to partition nodes based on similarities

Example by L. Lazebnik
Graph Cuts

Min-cut

- Find cut with minimum cost
- Fast (polynomial-time) algorithm

Example by L. Lazebnik
Graph Cuts

Min-cut

• Find cut with minimum cost
• Fast (polynomial-time) algorithm
• Not always the best choice

Cuts with lesser weight than the ideal cut

Example by L. Lazebnik
Graph Cuts

Normalized Cut

- Find minimum cut “normalized by segment size”

\[
\frac{w(A, B)}{w(A, V)} + \frac{w(A, B)}{w(B, V)}
\]

\(w(A, B) = \text{sum of weights of all edges between A and B}\)
Graph Cuts

Normalized Cut

- No polynomial-time algorithm to find optimal cut
- Can use an approximation based on eigen-analysis of the graph adjacency matrix
Graph Cuts Pros and Cons

Pros

• Generic framework, can be used with many different features and affinity formulations
• Fast and practical for interactive foreground-background segmentation

Cons

• High storage requirement and time complexity for automatic segmentation
Summary

Segmentation:

- Partitioning image into coherent regions

Algorithms:

- Divisive and hierarchical clustering
- k-means clustering
- Mean shift clustering
- Graph cuts
- More next time …

Applications

- Image processing, object recognition, interactive image editing, etc.