BSB 663
Image Processing
May. 28, 2013

Segmentation – Part 2
Review - Image segmentation

• Goal: identify groups of pixels that go together

Slide credit: S. Seitz, K. Grauman
Review- The goals of segmentation

• Separate image into coherent “objects”

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

Slide credit: S. Lazebnik
Review- What is segmentation?

• Clustering image elements that “belong together”

  – **Partitioning**
    • Divide into regions/sequences with coherent internal properties
  
  – **Grouping**
    • Identify sets of coherent tokens in image
Review- 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:
    $\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} ||p - c_i||^2$
Review- K-means clustering

K-means

1. Ask user how many clusters they’d like. (e.g. k=5)
Review- K-means clustering

K-means

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations

Slide credit: K Grauman, A. Moore
K-means clustering

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to. (Thus each Center “owns” a set of datapoints)
Review- K-means clustering

K-means

1. Ask user how many clusters they’d like. *(e.g. k=5)*

2. Randomly guess k cluster Center locations

3. Each datapoint finds out which Center it’s closest to.

4. Each Center finds the centroid of the points it owns
Review- K-means clustering

K-means

1. Ask user how many clusters they’d like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it’s closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!
Review - 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

Feature space: intensity value (I-d)
quantization of the feature space; segmentation label map
Review - Segmentation as clustering

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

Grouping pixels based on color similarity

Feature space: color value (3-d)
Review - 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.
Review - 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.

Slide credit: K Grauman
Review - K-means: pros and cons

Pros

• Simple, fast to compute
• Converges to local minimum of within-cluster squared error

Cons/issues

• Setting k?
• Sensitive to initial centers
• Sensitive to outliers
• Detects spherical clusters
• Assuming means can be computed
Segmentation methods

- Segment foreground from background
- Histogram-based segmentation
- Segmentation as clustering
  - K-means clustering
  - Mean-shift segmentation
- Graph-theoretic segmentation
  - Min cut
  - Normalized cuts
- Interactive segmentation
Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation


[Image: Segmented "landscape 1" and Segmented "landscape 2"]

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

Slide credit: S. Lazebnik
Finding Modes in a Histogram

- How Many Modes Are There?
  - Easy to see, hard to compute
Mean shift algorithm

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

Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the “mode” or point of highest density of a data distribution:

Two issues:
(1) Kernel to interpolate density based on sample positions.
(2) Gradient ascent to mode.

Slide credit: B. Freeman and A. Torralba
Mean shift
Mean shift

- Search window
- Center of mass
- Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass
Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift
Mean shift

Center of mass

Search window

Mean Shift vector

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Slide credit: Y. Ukrainitz & B. Sarel
Mean shift

Search window
Center of mass

Slide credit: Y. Ukrainitz & B. Sarel
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
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

Slide credit: S. Lazebnik
Apply mean shift jointly in the image (left col.) and range (right col.) domains.

1. Window in image domain
2. Intensities of pixels within image domain window
3. Window in range domain
4. Center of mass of pixels within both image and range domain windows
5. Center of mass of pixels within both image and range domain windows
6. Window in range domain
7. Window in range domain

Slide credit: B. Freeman and A. Torralba
Fig. 4. Visualization of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. The black dots are the points of convergence. (c) Filtering result \((h_x, h_y) = (8, 4)\). (d) Segmentation result.

Comaniciu and Meer, IEEE PAMI vol. 24, no. 5, 2002

Slide credit: B. Freeman and A. Torralba
Mean shift segmentation results

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

Slide credit: S. Lazebnik
More results
More results
Mean shift pros and cons

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

• Cons
  – Output depends on window size
  – Computationally expensive
  – Does not scale well with dimension of feature space
Segmentation methods

• Segment foreground from background
• Histogram-based segmentation
• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation
• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts
• Interactive Segmentation
Graph-Theoretic Image Segmentation

Build a weighted graph $G=(V,E)$ from image

$V$: image pixels

$E$: connections between pairs of nearby pixels

$W_{ij}$: probability that $i$ & $j$ belong to the same region

Segmentation = graph partition

Slide credit: B. Freeman and A. Torralba
Graphs Representations

Adjacency Matrix

Slide credit: B. Freeman and A. Torralba

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
A Weighted Graph and its Representation

\[
W = \begin{bmatrix}
1 & .1 & .3 & 0 & 0 \\
.1 & 1 & .4 & 0 & .2 \\
.3 & .4 & 1 & .6 & .7 \\
0 & 0 & .6 & 1 & 1 \\
0 & .2 & .7 & 1 & 1
\end{bmatrix}
\]

\[
W_{ij} : \text{probability that } i \& j \text{ belong to the same region}
\]

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
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

Slide credit: S. Seitz
Affinity between pixels

Similarities among pixel descriptors

$$W_{ij} = \exp\left(-\|z_i - z_j\|^2 / \sigma^2\right)$$

$\sigma = $ Scale factor…

it will hunt us later

Slide credit: B. Freeman and A. Torralba
Affinity between pixels

Similarities among pixel descriptors

\[ W_{ij} = \exp\left(-\frac{|| z_i - z_j ||^2}{\sigma^2}\right) \]

Interleaving edges

\[ W_{ij} = 1 - \max Pb \]

Line between i and j

With Pb = probability of boundary

\[ \sigma = \text{Scale factor} \ldots \]

it will hunt us later

Slide credit: B. Freeman and A. Torralba
Scale affects affinity

- Small $\sigma$: group only nearby points
- Large $\sigma$: group far-away points
Feature grouping by “relocalisation” of eigenvectors of the proximity matrix

Three points in feature space

\[ W_{ij} = \exp\left(-\frac{||z_i - z_j||^2}{s^2}\right) \]

With an appropriate \( s \)

\[
\begin{array}{ccc}
A & B & C \\
A & 1.00 & 0.63 & 0.03 \\
B & 0.63 & 1.00 & 0.00 \\
C & 0.03 & 0.00 & 1.00
\end{array}
\]

The eigenvectors of \( W \) are:

\[
\begin{array}{ccc}
\text{Eigenvalues} & E_1 & E_2 & E_3 \\
A & 1.63 & 0.01 & 0.71 \\
B & 0.01 & 0.05 & 0.71 \\
C & -0.03 & 1.00 & 0.00
\end{array}
\]

The first 2 eigenvectors group the points as desired…


Slide credit: B. Freeman and A. Torralba
Example eigenvector

points

Affinity matrix

eigenvector

Slide credit: B. Freeman and A. Torralba
Example eigenvector
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?
Segmentation methods

• Segment foreground from background

• Histogram-based segmentation

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts

• Interactive segmentation
Minimum cut

A cut of a graph $G$ is the set of edges $S$ such that removal of $S$ from $G$ disconnects $G$.

**Cut**: sum of the weight of the cut edges:

$$cut(A,B) = \sum_{u \in A, v \in B} W(u,v),$$

with $A \cap B = \emptyset$.

Slide credit: B. Freeman and A. Torralba
Minimum cut

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

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

Minimum cut example
Drawbacks of Minimum cut

- Weight of cut is directly proportional to the number of edges in the cut.
Segmentation methods

• Segment foreground from background
• Histogram-based segmentation
• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation
• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts
• Interactive segmentation
Normalized cuts

Write graph as $V$, one cluster as $A$ and the other as $B$

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

$cut(A,B)$ is sum of weights with one end in $A$ and one end in $B$

$$cut(A,B) = \sum_{u \in A, v \in B} W(u,v),$$

with $A \cap B = \emptyset$

assoc($A,V$) is sum of all edges with one end in $A$.

$$assoc(A,B) = \sum_{u \in A, v \in B} W(u,v)$$

$A$ and $B$ not necessarily disjoint

J. Shi and J. Malik. [Normalized cuts and image segmentation.](http://dx.doi.org/10.1109/34.868688) PAMI 2000

Slide credit: B. Freeman and A. Torralba
Normalized cut

- Let $W$ be the adjacency matrix of the graph
- Let $D$ be the diagonal matrix with diagonal entries $D(i, i) = \sum_j W(i, j)$
- Then the normalized cut cost can be written as

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

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

J. Shi and J. Malik. [Normalized cuts and image segmentation.](https://doi.org/10.1109/34.868688) PAMI 2000
Normalized cut

• Finding the exact minimum of the normalized cut cost is NP-complete, but if we relax \( y \) to take on arbitrary values, then we can minimize the relaxed cost by solving the \textit{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 \( i \)th entry of \( y \) can be viewed as a “soft” indication of the component membership of the \( i \)th feature
  – Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

Normalized cut algorithm

1. Given an image or image sequence, set up a weighted graph $G = (V, E)$, and set the weight on the edge connecting two nodes being a measure of the similarity between the two nodes.

2. Solve $(D - W)\mathbf{x} = \lambda D\mathbf{x}$ for eigenvectors with the smallest eigenvalues.

3. Use the eigenvector with second smallest eigenvalue to bipartition the graph.

4. Decide if the current partition should be sub-divided, and recursively repartition the segmented parts if necessary.

Slide credit: B. Freeman and A. Torralba
Global optimization

• In this formulation, the segmentation becomes a global process.
• Decisions about what is a boundary are not local (as in Canny edge detector)
Boundaries of image regions defined by a number of attributes

- Brightness/color
- Texture
- Motion
- Stereoscopic depth
- Familiar configuration
Example

Affinity:

\[ w_{ij} = e^{-\frac{\|F(i) - F(j)\|^2}{\sigma_f}} \]

\[
\begin{cases}
\frac{-\|X(i) - X(j)\|^2}{\sigma_X} & \text{if } \|X(i) - X(j)\|_2 < r \\
0 & \text{otherwise}
\end{cases}
\]

\[ \text{brightness} \]

\[ \text{Location} \]

\[ N \text{ pixels} = \text{ncols} \times \text{nrows} \]

Slide credit: B. Freeman and A. Torralba
Figure 12: Subplot (1) plots the smallest eigenvectors of the generalized eigenvalue system (11). Subplot (2) - (9) shows the eigenvectors corresponding the 2nd smallest to the 9th smallest eigenvalues of the system. The eigenvectors are reshaped to be the size of the image.

Slide credit: B. Freeman and A. Torralba
Brightness Image Segmentation

A multiresolution implementation can be used to reduce this running time further on larger images. In our current experiments, with this implementation, the running time on a 300 × 400 image can be reduced to about 20 seconds on Intel Pentium 300MHz machines. Furthermore, the bottleneck of the computation, a sparse matrix-vector
Brightness Image Segmentation


Slide credit: B. Freeman and A. Torralba
Results on color segmentation


Slide credit: B. Freeman and A. Torralba
Example results
Results: Berkeley Segmentation Engine

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

Slide credit: S. Lazebnik
Normalized cuts: Pro and con

• Pros
  – Generic framework, can be used with many different features and affinity formulations

• Cons
  – High storage requirement and time complexity
  – Bias towards partitioning into equal segments

Slide credit: S. Lazebnik
Segmentation methods

• Segment foreground from background

• Histogram-based segmentation

• Segmentation as clustering
  – K-means clustering
  – Mean-shift segmentation

• Graph-theoretic segmentation
  • Min cut
  • Normalized cuts

• Interactive segmentation
Intelligent Scissors [Mortensen 95]

- Approach answers a basic question
  - Q: how to find a path from seed to mouse that follows object boundary as closely as possible?

Mortensen and Barrett, Intelligent Scissors for Image Composition, Proc. 22nd annual conference on Computer graphics and interactive techniques, 1995

Figure 2: Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t₀, t₁, and t₂) are shown in green.
Intelligent Scissors

• Basic Idea
  – Define edge score for each pixel
    • edge pixels have low cost
  – Find lowest cost path from seed to mouse

Questions
• How to define costs?
• How to find the path?
Path Search (basic idea)

- Graph Search Algorithm
  - Computes minimum cost path from seed to \textit{all other pixels}

Slide credit: S. Seitz
How does this really work?

- Treat the image as a graph

Graph
- node for every pixel $p$
- link between every adjacent pair of pixels, $p, q$
- cost $c$ for each link

Note: each link has a cost
- this is a little different than the figure before where each pixel had a cost
Defining the costs

• Treat the image as a graph

Want to hug image edges: how to define cost of a link?
  • the link should follow the intensity edge
    – want intensity to change rapidly \( \partial \) to the link
  • \( c \begin{array}{c} \text{difference of intensity} \\ \partial \end{array} \text{ to link} \)
Defining the costs

- \( c \) can be computed using a cross-correlation filter
  - assume it is centered at \( p \)

- Also typically scale \( c \) by its length
  - set \( c = (\text{max} - |\text{filter response}|) \)
    - where \( \text{max} \) = maximum |filter response| over all pixels in the image

Slide credit: S. Seitz
Defining the costs

- $c$ can be computed using a cross-correlation filter
  - assume it is centered at $p$

- Also typically scale $c$ by its length
  - set $c = (\max - |\text{filter response}|)$
    - where $\max$ = maximum $|\text{filter response}|$ over all pixels in the image
Dijkstra’s shortest path algorithm

Algorithm
1. init node costs to \( \infty \), set \( p = \) seed point, \( \text{cost}(p) = 0 \)
2. expand \( p \) as follows:
   for each of \( p \)'s neighbors \( q \) that are not expanded
   » set \( \text{cost}(q) = \min( \text{cost}(p) + c_{pq}, \text{cost}(q) ) \)
Dijkstra’s shortest path algorithm

Algorithm

1. init node costs to \( \infty \), set \( p \) = seed point, \( \text{cost}(p) = 0 \)

2. expand \( p \) as follows:
   for each of \( p \)'s neighbors \( q \) that are not expanded
   » set \( \text{cost}(q) = \min( \text{cost}(p) + c_{pq}, \text{cost}(q) ) \)
     » if \( q \)'s cost changed, make \( q \) point back to \( p \)
   » put \( q \) on the ACTIVE list (if not already there)

Slide credit: S. Seitz
Dijkstra’s shortest path algorithm

Algorithm

1. init node costs to $\infty$, set $p$ = seed point, cost$(p) = 0$
2. expand $p$ as follows:
   for each of $p$’s neighbors $q$ that are not expanded
      » set cost$(q) = \min( \text{cost}(p) + c_{pq}, \text{cost}(q) )$
      » if $q$’s cost changed, make $q$ point back to $p$
      » put $q$ on the ACTIVE list (if not already there)
3. set $r$ = node with minimum cost on the ACTIVE list
4. repeat Step 2 for $p = r$

Slide credit: S. Seitz
Dijkstra’s shortest path algorithm

1. init node costs to $\infty$, set $p =$ seed point, cost($p$) = 0
2. expand $p$ as follows:
   for each of $p$’s neighbors $q$ that are not expanded
   » set cost($q$) = min( cost($p$) + $c_{pq}$, cost($q$) )
   » if $q$’s cost changed, make $q$ point back to $p$
   » put $q$ on the ACTIVE list (if not already there)
3. set $r =$ node with minimum cost on the ACTIVE list
4. repeat Step 2 for $p = r$
Segmentation by min (s-t) cut

- **Graph**
  - node for each pixel, link between pixels
  - specify a few pixels as foreground and background
    - create an infinite cost link from each bg pixel to the “t” node
    - create an infinite cost link from each fg pixel to the “s” node
  - compute min cut that separates s from t
  - how to define link cost between neighboring pixels?

Y. Boykov and M-P Jolly, Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images, ICCV, 2001.
Random Walker

- Compute probability that a random walker arrives at seed


http://cns.bu.edu/~lgrady/Random_Walker_Image_Segmentation.html
Do we need recognition to take the next step in performance?

Slide credit: B. Freeman and A. Torralba
Top-down segmentation


Slide credit: S. Lazebnik
Top-down segmentation

- E. Borenstein and S. Ullman, Class-specific, top-down segmentation, ECCV 2002
Motion segmentation


Slide credit: K. Grauman