Review - Image segmentation

- Goal: identify groups of pixels that go together

Review - The goals of segmentation

- Separate image into coherent “objects”

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

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

Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation


http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html
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.
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
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. Center of mass of pixels within both image and range domain windows

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 \((h_x, h_y) = (8, 6)\). (d) Segmentation result.

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

Mean shift segmentation results

More results

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

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

http://www.caip.rutgers.edu/~comanicii/MSPAMI/msPamiResults.html
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
• Boundary detection

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
Graphs Representations

Adjacency Matrix

A Weighted Graph and its Representation

Affinity Matrix

Segmentation by graph partitioning

Affinity between pixels

Similarities among pixel descriptors

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

- 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
**Affinity between pixels**

Similarities among pixel descriptors

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

Interleaving edges

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

With \( P_b \) = probability of boundary

\( \sigma \) = Scale factor...

it will hunt us later

**Scale affects affinity**

- Small \( \sigma \): group only nearby points
- Large \( \sigma \): group far-away points

**Example eigenvector**

The eigenvectors of \( W \) are:

The first 2 eigenvectors group the points as desired...

**Feature grouping by “relocalisation” of eigenvectors of the proximity matrix**


Guy L. Scott
Robotics Research Group
Department of Engineering Science
University of Oxford

H. Christopher Longuet-Higgins
University of Sussex
Brighton

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

With an appropriate \( \sigma \)

\[
\begin{array}{ccc}
A & B & C \\
-1.00 & 0.63 & 0.03 \\
0.63 & 1.00 & 0.0 \\
0.03 & 0.0 & 1.00 \\
\end{array}
\]

\[
W= \\
\begin{array}{c}
A \\
B \\
C \\
\end{array}
\]

The eigenvectors of \( W \) are:

The first 2 eigenvectors group the points as desired...
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
- Boundary detection

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:

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

where $A \cap B = \emptyset$. 

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 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
- Boundary detection
Normalized cuts

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

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

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

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

with $A \cap B = \emptyset$

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

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

$A$ and $B$ not necessarily disjoint


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 $D(i,i) = \Sigma_j W(i,j)$
- 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 $i$th position if the $i$th feature point belongs to $A$ and a negative constant otherwise


Slide credit: S. Lazebnik

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 $(D - W) x = \lambda D x$ 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

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.
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

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

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
- Boundary detection
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

Path Search (basic idea)

- Graph Search Algorithm
  - Computes minimum cost path from seed to all other pixels

Questions

- How to define costs?
- How to find the path?

Intelligent Scissors

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

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 to the link
- $c = \text{difference of intensity to link}$

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 $c = (\text{max}|\text{filter response}|)$
    - where 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 $c = (\text{max}|\text{filter response}|)$
    - where max = maximum |filter response| over all pixels in the image

Slide credit: S. Seitz

---

Dijkstra’s shortest path algorithm

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

Slide credit: S. Seitz
Algorithm
1. init node costs to $\infty$, set $p = \text{seed point, } \text{cost}(p) = 0$
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)
3. set $r = \text{node with minimum cost on the ACTIVE list}$
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

---

**Top-down segmentation**


---

**Do we need recognition to take the next step in performance?**

---

**Top-down segmentation**

Motion segmentation

Input sequence
Image Segmentation
Motion Segmentation

Input sequence
Image Segmentation
Motion Segmentation


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
- Boundary detection

Berkeley Segmentation Data Set
David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik

Slide credit: K. Grauman

Slide credit: J. Hays
Protocol

You will be presented a photographic image. Divide the image into some number of segments, where the segments represent “things” or “parts of things” in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance.

- Custom segmentation tool
- Subjects obtained from work-study program (UC Berkeley undergraduates)
**Segmentations are Consistent**

A, C are refinements of B
A, C are mutual refinements
A, B, C represent the same percept
- Attention accounts for differences

**Perceptual organization forms a tree:**

- Two segmentations are consistent when they can be explained by the same segmentation tree (i.e. they could be derived from a single perceptual organization).

**Dataset Summary**

- 30 subjects, age 19-23
  - 17 men, 13 women
  - 9 with artistic training
- 8 months
- 1,458 person hours
- 1,020 Corel images
- 11,595 Segmentations
  - 5,555 color, 5,554 gray, 486 inverted/negated

**Pb Detector**

- Image
- Boundary Cues: Brightness, Color, Texture
- Cue Combination
- Model
- \( P_b \)

**Challenges:** texture cue, cue combination
**Goal:** learn the posterior probability of a boundary \( P_b \) from local information only
Brightness and Color Features

- 1976 CIE $L^a*b^*$ colorspace
- Brightness Gradient (B)
  - Chi$^2$ difference in $L^*$ distribution
- Color Gradient (C)
  - Chi$^2$ difference in $a^*$ and $b^*$ distributions

$$\chi^2(g,h) = \frac{1}{2} \sum_i \frac{(g_i - h_i)^2}{g_i + h_i}$$

Texture Feature

- Texture Gradient (T)
- Chi$^2$ difference of texton histograms
  - Textons are vector-quantized filter outputs

Cue Combination Models

- Classification Trees
  - Top-down splits to maximize entropy, error bounded
- Density Estimation
  - Adaptive bins using k-means
- Logistic Regression, 3 variants
  - Linear and quadratic terms
  - Confidence-rated generalization of AdaBoost (Schapire&Singer)
- Hierarchical Mixtures of Experts (Jordan&Jacobs)
  - Up to 8 experts, initialized top-down, fit with EM
- Support Vector Machines (libsvm, Chang&Lin)

- Range over bias, complexity, parametric/non-parametric
**Computing Precision/Recall**

Recall = Pr(signal|truth) = fraction of ground truth found by the signal  
Precision = Pr(truth|signal) = fraction of signal that is correct  
- Always a trade-off between the two  
- Standard measures in information retrieval (van Rijsbergen XX)  
- ROC from standard signal detection the wrong approach

**Strategy**

- Detector output (Pb) is a soft boundary map  
- Compute precision/recall curve:  
  - Threshold Pb at many points t in [0,1]  
  - Recall = Pr(Pb>t|seg=1)  
  - Precision = Pr(seg=1|Pb>t)

**Cue Calibration**

- All free parameters optimized on training data  
- All algorithmic alternatives evaluated by experiment

- Brightness Gradient  
  - Scale, bin/kernel sizes for KDE  
- Color Gradient  
  - Scale, bin/kernel sizes for KDE, joint vs. marginals  
- Texture Gradient  
  - Filter bank: scale, multiscale?  
  - Histogram comparison  
  - Number of textons, Image-specific vs. universal textons  
- Localization parameters for each cue

**Dataflow**

- Image  
- Optimized Cues  
  - Brightness  
  - Color  
  - Texture  
- Cue Combination  
- Model  
- Pb

**Pb Images**

- Canny  
- 2MM  
- Us  
- Human

**Benchmark**

- Human Segmentations  

Slide credits: J. Hays
**Findings**

1. A simple linear model is sufficient for cue combination
   - All cues weighted approximately equally in logistic
2. Proper texture edge model is not optional for complex natural images
   - Texture suppression is not sufficient!
3. Significant improvement over state-of-the-art in boundary detection
4. Empirical approach critical for both cue calibration and cue combination

*Slide credit: J. Hays*
**Image Features – 21350 dimensions!**

- 35x35 patches centered at every pixel
- 35x35 “channels” of many types:
  - Color (3 channels)
  - Gradients (3 unoriented + 8 oriented channels)
    - $\Sigma = 0, \Theta = 0, \pi/2, \pi, 3\pi/2$
    - $\Sigma = 1.5, \Theta = 0, \pi/2, \pi, 3\pi/2$
    - $\Sigma = 5$
  - Self Similarity
    - 5x5 maps of self similarity within the above channels for a particular anchor point.

**Self-similarity features**

Self-similarity features: The L1 distance from the anchor cell (yellow box) to the other 5 x 5 cells are shown for color and gradient magnitude channels. The original patch is shown to the left.

**Learning**

- Random Forest Classifiers, one for each sketch token + background, trained 1-vs-all
- Advantages:
  - Fast at test time, especially for a non-linear classifier.
  - Don’t have to explicitly compute independent descriptors for every patch. Just look up what the decision tree wants to know at each branch.
Frequency of example features being selected by the random forest: (first row) color channels, (second row) gradient magnitude channels, (third row) selected orientation channels.

Detections of individual sketch tokens

• Simply add the probability of all non-background sketch tokens

• Free parameter: number of sketch tokens
  – k = 1 works poorly, k = 16 and above work OK.
Summary

- Distinct from previous work, cluster the human annotations to discover the mid-level structures that you want to detect.
- Train a classifier for every sketch token.
- Is as accurate as any other method while being 200 times faster and using no global information.

Evaluation on BSDS

<table>
<thead>
<tr>
<th>Method</th>
<th>ODS</th>
<th>OIS</th>
<th>AP</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>.80</td>
<td>.80</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Canny</td>
<td>.60</td>
<td>.64</td>
<td>.58</td>
<td>1/15 s</td>
</tr>
<tr>
<td>Felz-Hutt [12]</td>
<td>.61</td>
<td>.64</td>
<td>.56</td>
<td>1/10 s</td>
</tr>
<tr>
<td>gPb (local) [1]</td>
<td>.71</td>
<td>.74</td>
<td>.65</td>
<td>60 s</td>
</tr>
<tr>
<td>SCG (local) [24]</td>
<td>.72</td>
<td>.74</td>
<td>.75</td>
<td>100 s</td>
</tr>
<tr>
<td>Sketch tokens</td>
<td>.73</td>
<td>.75</td>
<td>.78</td>
<td>1 s</td>
</tr>
<tr>
<td>gPb (global) [1]</td>
<td>.73</td>
<td>.76</td>
<td>.73</td>
<td>240 s</td>
</tr>
<tr>
<td>SCG (global) [24]</td>
<td>.74</td>
<td>.76</td>
<td>.77</td>
<td>280 s</td>
</tr>
</tbody>
</table>