Image segmentation
Segmentation challenges

Kittens are distinguishable by color (sort of), but not texture.

Chameleons are distinguishable by texture, but not color.

Wheels are part of the car, but not similar in color or texture.

How do we recognize that the head and body/sweater are the same “person”?
Segmentation challenges

http://optical-illusions.wikia.com/wiki/Emergence
Segmentation issues

- “Bottom-up” or “top-down” process?
- Supervised or unsupervised?
- What is the application?
Outline

• Bottom-up segmentation
  • Superpixel segmentation
  • Normalized cuts

• Interactive segmentation
  • CRF energy functions, graph cut optimization

• Supervised or top-down segmentation
  • CRFs
  • Deep networks (next time)
Superpixel segmentation

- Group together similar-looking pixels as an intermediate stage of processing
  - “Bottom-up” process
  - Typically unsupervised
  - Should be fast
  - Typically aims to produce an over-segmentation

Graph-based segmentation

- **Node** = pixel
- **Edge** = pair of neighboring pixels
- **Edge weight** = similarity or dissimilarity of the respective nodes

Source: S. Seitz
Efficient graph-based segmentation

- Runs in time nearly linear in the number of edges
- Easy to control coarseness of segmentations
- Results can be unstable

P. Felzenszwalb and D. Huttenlocher,
Efficient Graph-Based Image Segmentation, IJCV 2004
Felzenszwalb & Huttenlocher algorithm

- **Graph definition:**
  - Vertices are pixels, edges connect neighboring pixels, weights correspond to dissimilarity in \((x,y,r,g,b)\) space

- **The algorithm:**
  - Start with each vertex in its own component
  - For each edge in increasing order of weight:
    - If the edge is between vertices in two different components \(A\) and \(B\), merge if the edge weight is lower than the internal dissimilarity within either of the components
    - Threshold is the minimum of the following values, computed on \(A\) and \(B\):
      - \((\text{Highest-weight edge in minimum spanning tree of the component}) + (k / \text{size of component})\)
Example results

http://www.cs.brown.edu/~pff/segment/
Other superpixel algorithms

- **Watershed segmentation**
- **Simple linear iterative clustering (SLIC)**
Segmentation by graph cuts

• 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 cuts

• A graph cut is a set of edges whose removal disconnects the graph
• Cost of a cut: sum of weights of cut edges
• Two-way minimum cuts can be found efficiently
Segmentation by graph cuts

• A graph cut is a set of edges whose removal disconnects the graph
• Cost of a cut: sum of weights of cut edges
• Two-way minimum cuts can be found efficiently

Affinity matrix
Normalized cut

- Minimum cut tends to cut off very small, isolated components

Ideal Cut

Cuts with lesser weight than the ideal cut
Normalized cut

• To encourage larger segments, normalize the cut by the total weight of edges incident to the segment
• The *normalized cut* cost is:

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

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

• Intuition: big segments will have a large \( w(A, V) \), thus decreasing \( ncut(A, B) \)

• Finding the globally optimal cut is NP-complete, but a relaxed version can be solved using a generalized eigenvalue problem

Normalized cut: Algorithm

- Let $W$ be the affinity matrix of the graph ($n \times n$ for $n$ pixels)
- Let $D$ be the diagonal matrix with entries $D(i, i) = \sum_j W(i, j)$
- Solve *generalized eigenvalue problem* $(D - W)y = \lambda Dy$ for the eigenvector with the second smallest eigenvalue
  - The $i$th entry of $y$ can be viewed as a “soft” indicator of the component membership of the $i$th pixel
    - Use 0 or median value of the entries of $y$ to split the graph into two components
  - To find more than two components:
    - Recursively bipartition the graph
    - Run k-means clustering on values of several eigenvectors
Example result

Original image

Eigenvectors for 2\textsuperscript{nd} and 3\textsuperscript{rd} smallest eigenvalues

More eigenvectors
Normalized cuts: Pro and con

- **Pro**
  - Generic framework, can be used with many different features and affinity formulations

- **Con**
  - High storage requirement and time complexity: involves solving a generalized eigenvalue problem of size $n \times n$, where $n$ is the number of pixels
Segmentation as labeling

- Suppose we want to segment an image into foreground and background
- Binary pixel labeling problem
Segmentation as labeling

• Suppose we want to segment an image into foreground and background
  • Binary pixel labeling problem
  • Naturally arises in interactive settings

User scribbles
Labeling by energy minimization

- Define a labeling $c$ as an assignment of each pixel to a class (foreground or background)

- Find the labeling that minimizes a global energy function:

$$E(c \mid x) = \sum_i f_i(c_i, x) + \sum_{i,j \in \mathcal{E}} g_{ij}(c_i, c_j, x)$$

  - **Unary potential** (local data term): score for pixel $i$ and label $c_i$
  - **Pairwise potential** (context or smoothing term): neighboring pixels

- These are known as Markov Random Field (MRF) or Conditional Random Field (CRF) functions
Segmentation by energy minimization

\[ E(c \mid x) = \sum_i f_i(c_i, x) + \sum_{i,j \in \mathcal{E}} g_{ij}(c_i, c_j, x) \]

- **Unary potentials:**
  \[ f_i(c, x) = -\log P(c \mid x_i) \]
  - Cost is infinity if label does not match the user scribble
  - Otherwise, it is computed based on a color model of user-labeled pixels
Segmentation by energy minimization

\[ E(c \mid x) = \sum_i f_i(c_i, x) + \sum_{i,j \in \mathcal{E}} g_{ij}(c_i, c_j, x) \]

- **Unary potentials:**  \[ f_i(c, x) = -\log P(c \mid x_i) \]
- **Pairwise potentials:**  \[ g_{ij}(c, c', x) = w_{ij} \left| c - c' \right| \]
- Neighboring pixels should have the same label unless they look very different

[Image of a flower with high and low affinity points]
Segmentation by energy minimization

\[ E(c | x) = \sum_{i} f_i(c_i, x) + \sum_{i,j \in \varepsilon} g_{ij}(c_i, c_j, x) \]

- Can be optimized efficiently by finding the minimum cut in the following graph:

---

Y. Boykov and M. Jolly, Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images, ICCV 2001
Recall: Stereo as energy minimization

\[ E(D) = \sum_{i} \left( W_1(i) - W_2(i + D(i)) \right)^2 + \lambda \sum_{\text{neighbors } i,j} \rho(D(i) - D(j)) \]

- **data term**
- **smoothness term**
GrabCut

C. Rother, V. Kolmogorov, and A. Blake,
“GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts, SIGGRAPH 2004
Semantic segmentation

- Problem: label each pixel by one of C classes
- Define an energy function where unaries correspond to local classifier responses and smoothing potentials correspond to contextual terms
- Solve a *multi-class* graph cut problem
Example: TextonBoost

\[
\log P(c | x, \theta) =
\]

\[
\sum_i \psi_i(c_i, x; \theta_\psi) + \pi(c_i, x_i; \theta_\pi) + \lambda(c_i, i; \theta_\lambda)
\]

\[
+ \sum_{(i,j) \in \mathcal{E}} \phi(c_i, c_j; g_{ij}(x); \theta_\phi)
\]

Example: SuperParsing

- CRF energy function is defined on superpixels
  - Unaries are based on nearest neighbor retrieval
  - Pairwise potentials capture class co-occurrence statistics

J. Tighe and S. Lazebnik, SuperParsing: Scalable Nonparametric Image Parsing with Superpixels, ECCV 2010
Example: SuperParsing

- CRF energy function is defined on superpixels
  - Unaries are based on nearest neighbor retrieval
  - Pairwise potentials capture class co-occurrence statistics

J. Tighe and S. Lazebnik, SuperParsing: Scalable Nonparametric Image Parsing with Superpixels, ECCV 2010
Review: segmentation

- **Unsupervised segmentation**
  - Superpixel segmentation
  - Normalized cuts

- **Interactive segmentation**
  - CRF energy functions, graph cut optimization

- **How to evaluate segmentation?**