Computer Vision - Lecture 7

Segmentation as Energy Minimization

19.11.2015

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de
Announcements

• Please don’t forget to register for the exam!
  ➢ On the Campus system
Course Outline

- Image Processing Basics
- Segmentation
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Recognition
  - Global Representations
  - Subspace representations
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking
Recap: Image Segmentation

- Goal: identify groups of pixels that go together
Recap: 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:
      \[
      \sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2
      \]
Recap: Expectation Maximization (EM)

- **Goal**
  - Find blob parameters $\theta$ that maximize the likelihood function:
  
  \[ p(data|\theta) = \prod_{n=1}^{N} p(x_n|\theta) \]

- **Approach:**
  1. **E-step:** given current guess of blobs, compute ownership of each point
  2. **M-step:** given ownership probabilities, update blobs to maximize likelihood function
  3. Repeat until convergence
Recap: EM Algorithm

- **Expectation-Maximization (EM) Algorithm**
  - **E-Step**: softly assign samples to mixture components
    \[
    \gamma_j(x_n) \leftarrow \frac{\pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \ n = 1, \ldots, N
    \]
  - **M-Step**: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \hat{N}_j \leftarrow \sum_{n=1}^{N} \gamma_j(x_n) = \text{soft number of samples labeled } j
    \]
    \[
    \hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}
    \]
    \[
    \hat{\mu}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)x_n
    \]
    \[
    \hat{\Sigma}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)(x_n - \hat{\mu}_j^{\text{new}})(x_n - \hat{\mu}_j^{\text{new}})^T
    \]
MoG Color Models for Image Segmentation

- User assisted image segmentation
  - User marks two regions for foreground and background.
  - Learn a MoG model for the color values in each region.
  - Use those models to classify all other pixels.
  ⇒ Simple segmentation procedure
     (building block for more complex applications)
Recap: Mean-Shift Algorithm

- **Iterative Mode Search**
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the “mean”) of W: \( \sum_{x \in W} x H(x) \)
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence
Recap: 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
Recap: Mean-Shift Segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode
Back to the Image Segmentation Problem...

• Goal: identify groups of pixels that go together

• Up to now, we have focused on ways to group pixels into image segments based on their appearance...
  - Segmentation as clustering.

• We also want to enforce region constraints.
  - Spatial consistency
  - Smooth borders

B. Leibe
Topics of This Lecture

• Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

• Graph cuts for image segmentation
  - Basic idea
  - s-t Min-cut algorithm
  - Extension to non-binary case

• Applications
  - Interactive segmentation
**Markov Random Fields**

- Allow rich probabilistic models for images
- But built in a local, modular way
  - Learn local effects, get global effects out

Slide credit: William Freeman
MRF Nodes as Pixels

Original image

Degraded image

Reconstruction from MRF modeling pixel neighborhood statistics

\[ \Phi(x_i, y_i) \]

\[ \Psi(x_i, x_j) \]

B. Leibe
Network Joint Probability

\[ p(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]

Scene

Image

Image-scene compatibility function

Scene-scene compatibility function

Local observations

Neighboring scene nodes

Slide credit: William Freeman

B. Leibe
Energy Formulation

- Joint probability
  \[ p(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]

- Maximizing the joint probability is the same as minimizing the negative log
  \[ -\log p(x, y) = -\sum_i \log \Phi(x_i, y_i) - \sum_{i,j} \log \Psi(x_i, x_j) \]

  \[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]

- This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call \( E \) an energy function.

- \( \phi \) and \( \psi \) are called potentials.
Energy Formulation

- **Energy function**
  \[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]
  - **Single-node potentials** \( \phi \) ("unary potentials")
    - Encode local information about the given pixel/patch
    - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
  - **Pairwise potentials** \( \psi \)
    - Encode neighborhood information
    - How different is a pixel/patch’s label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

\[ \phi(x_i, y_i) \]
\[ \psi(x_i, x_j) \]
Energy Minimization

• Goal:
  - Infer the optimal labeling of the MRF.

• Many inference algorithms are available, e.g.
  - Gibbs sampling, simulated annealing
  - Iterated conditional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts

• Recently, Graph Cuts have become a popular tool
  - Only suitable for a certain class of energy functions
  - But the solution can be obtained very fast for typical vision problems (~1MPixel/sec).
Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case

- Applications
  - Interactive segmentation
Graph Cuts for Optimal Boundary Detection

- Idea: convert MRF into source-sink graph

Minimum cost cut can be computed in polynomial time
(max-flow/min-cut algorithms)

\[ w_{ij} = \exp \left\{ -\frac{\Delta I_{ij}}{2\sigma^2} \right\} \]

Slide credit: Yuri Boykov
Simple Example of Energy

\[ E(x, y) = \sum_i \phi_i(x_i) + \sum_{i,j} w_{i,j} \cdot \delta(x_i \neq x_j) \]

- **Unary terms**
- **Pairwise terms**

A cut

\[ w_{ij} = \exp \left\{ -\frac{\Delta I_{ij}}{2\sigma^2} \right\} \]

\[ x \in \{ s, t \} \]

(binary object segmentation)
Adding Regional Properties

Regional bias example

Suppose $I^s$ and $I^t$ are given “expected” intensities of object and background

\[
\phi_i(s) \propto \exp\left(-\frac{\| I_i - I^s \|^2}{2\sigma^2}\right)
\]

\[
\phi_i(t) \propto \exp\left(-\frac{\| I_i - I^t \|^2}{2\sigma^2}\right)
\]

NOTE: hard constrains are not required, in general.
Adding Regional Properties

“expected” intensities of object and background can be re-estimated

\[
\phi_i(s) \propto \exp \left( - \| I_i - I^s \|^2 / 2\sigma^2 \right)
\]

\[
\phi_i(t) \propto \exp \left( - \| I_i - I^t \|^2 / 2\sigma^2 \right)
\]

EM-style optimization

[Boykov & Jolly, ICCV’01]
Adding Regional Properties

- More generally, regional bias can be based on any intensity models of object and background

\[ \phi_i(L_i) = -\log p(I_i | L_i) \]

Given object and background intensity histograms

\[ p(I_i | t) \]
\[ p(I_i | s) \]
How to Set the Potentials? Some Examples

- **Color potentials**
  - e.g., modeled with a Mixture of Gaussians
    \[
    \phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) N(y_i; \mu_k, \Sigma_k)
    \]

- **Edge potentials**
  - E.g., a “contrast sensitive Potts model”
    \[
    \psi(x_i, x_j, g_{ij}(y); \theta_\psi) = -\theta_\psi g_{ij}(y) \delta(x_i \neq x_j)
    \]
    where
    \[
    g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \quad \beta = \frac{1}{2} \left( \text{avg} \left( \|y_i - y_j\|^2 \right) \right)^{-1}
    \]

- **Parameters** \( \theta_\phi, \theta_\psi \) need to be learned, too!

[Shotton & Winn, ECCV’06]
Example: MRF for Image Segmentation

- MRF structure

Unary potentials

Pairwise potentials

Data (D)  Unary likelihood  Pair-wise Terms  MAP Solution

Slide adapted from Phil Torr
Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case

- Applications
  - Interactive segmentation
How Does it Work? The s-t-Mincut Problem

Graph \((V, E, C)\)

Vertices \(V = \{v_1, v_2, ..., v_n\}\)

Edges \(E = \{(v_1, v_2), ..., \}\}\)

Costs \(C = \{c_{(1,2)}, ..., \}\)
The s-t-Mincut Problem

What is an st-cut?

An st-cut $(S, T)$ divides the nodes between source and sink.

What is the cost of an st-cut?

Sum of cost of all edges going from $S$ to $T$

$5 + 2 + 9 = 16$
The s-t-Mincut Problem

What is an st-cut?
An st-cut \((S, T)\) divides the nodes between source and sink.

What is the cost of an st-cut?
Sum of cost of all edges going from \(S\) to \(T\)

What is the st-mincut?
st-cut with the minimum cost

\[2 + 1 + 4 = 7\]
How to Compute the s-t-Mincut?

Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink

Constraints
Edges: Flow < Capacity
Nodes: Flow in = Flow out

Min-cut/Max-flow Theorem
In every network, the maximum flow equals the cost of the st-mincut

Slide credit: Pushmeet Kohli
History of Maxflow Algorithms

Augmenting Path and Push-Relabel

<table>
<thead>
<tr>
<th>year</th>
<th>discoverer(s)</th>
<th>bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>Dantzig</td>
<td>$O(n^2 m U)$</td>
</tr>
<tr>
<td>1955</td>
<td>Ford &amp; Fulkerson</td>
<td>$O(m^2 U)$</td>
</tr>
<tr>
<td>1970</td>
<td>Dinitz</td>
<td>$O(n^2 m)$</td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>$O(m^2 \log U)$</td>
</tr>
<tr>
<td>1973</td>
<td>Dinitz</td>
<td>$O(nm \log U)$</td>
</tr>
<tr>
<td>1974</td>
<td>Karzanov</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>1977</td>
<td>Cherkassky</td>
<td>$O(n^2 m^{1/2})$</td>
</tr>
<tr>
<td>1980</td>
<td>Galil &amp; Naamad</td>
<td>$O(nm \log^2 n)$</td>
</tr>
<tr>
<td>1983</td>
<td>Sleator &amp; Tarjan</td>
<td>$O(nm \log n)$</td>
</tr>
<tr>
<td>1986</td>
<td>Goldberg &amp; Tarjan</td>
<td>$O(nm \log(n^2/m))$</td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>$O(nm + n^2 \log U)$</td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja et al.</td>
<td>$O(nm \log(n \sqrt{\log U/m}))$</td>
</tr>
<tr>
<td>1989</td>
<td>Cheriyan &amp; Hagerup</td>
<td>$O(nm + n^2 \log^2 n)$</td>
</tr>
<tr>
<td>1990</td>
<td>Cheriyan et al.</td>
<td>$O(n^3/\log n)$</td>
</tr>
<tr>
<td>1990</td>
<td>Alon</td>
<td>$O(nm + n^{8/3} \log n)$</td>
</tr>
<tr>
<td>1992</td>
<td>King et al.</td>
<td>$O(nm + n^{2+\epsilon})$</td>
</tr>
<tr>
<td>1993</td>
<td>Phillips &amp; Westbrook</td>
<td>$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$</td>
</tr>
<tr>
<td>1994</td>
<td>King et al.</td>
<td>$O(nm \log_m(n \log n) n)$</td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Rao</td>
<td>$O(m^{3/2} \log(n^2/m) \log U)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(n^{2/3} m \log(n^2/m) \log U)$</td>
</tr>
</tbody>
</table>

$n$: #nodes
$m$: #edges
$U$: maximum edge weight

Algorithms assume non-negative edge weights
**Maxflow Algorithms**

Flow = 0

Source

\[ v_1 \]
2
1
5

\[ v_2 \]
9
2
4

Sink

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Flow = 0 + 2

Source

v1

v2

Sink

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity

2. Push maximum possible flow through this path

3. Repeat until no path can be found

Algorithms assume non-negative capacity

Source

Sink

v_1

v_2

Given capacities:

- Source to v_1: 0
- v_1 to v_2: 1
- v_2 to Sink: 2
- Source to Sink: 9

Flow through the path found: 2

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 2 + 4

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity
Maxflow Algorithms

Flow = 6

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

Flow = 6 + 1

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity
Maxflow Algorithms

Flow = 7

Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Maxflow Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli
Applications: Maxflow in Computer Vision

- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity \( m \sim O(n) \)

- Dual search tree augmenting path algorithm
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently.
  - High worst-case time complexity.
  - Empirically outperforms other algorithms on vision problems.
  - Efficient code available on the web
    [http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html](http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html)
When Can s-t Graph Cuts Be Applied?

\[ E(L) = \sum_{p} E_p(L_p) + \sum_{pq \in N} E(L_p, L_q) \]

- s-t graph cuts can only globally minimize binary energies that are submodular. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

\[ E(L) \text{ can be minimized by s-t graph cuts} \quad \iff \quad E(s,s) + E(t,t) \leq E(s,t) + E(t,s) \]

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - \(\Rightarrow\) Solution will be globally optimal.
Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case

- Applications
  - Interactive segmentation
Dealing with Non-Binary Cases

- Limitation to binary energies is often a nuisance.
  ⇒ E.g. binary segmentation only...
- We would like to solve also multi-label problems.
  - The bad news: Problem is NP-hard with 3 or more labels!
- There exist some approximation algorithms which extend graph cuts to the multi-label case:
  - $\alpha$-Expansion
  - $\alpha\beta$-Swap
- They are no longer guaranteed to return the globally optimal result.
  - But $\alpha$-Expansion has a guaranteed approximation quality (2-approx) and converges in a few iterations.
**α-Expansion Move**

- **Basic idea:**
  - Break multi-way cut computation into a sequence of binary s-t cuts.
$\alpha$-Expansion Algorithm

1. Start with any initial solution
2. For each label “$\alpha$” in any (e.g. random) order:
   1. Compute optimal $\alpha$-expansion move (s-t graph cuts).
   2. Decline the move if there is no energy decrease.
3. Stop when no expansion move would decrease energy.

Slide credit: Yuri Boykov
Example: Stereo Vision

Original pair of “stereo” images

Depth map

ground truth

Slide credit: Yuri Boykov
\(\alpha\)-Expansion Moves

- In each \(\alpha\)-expansion a given label “\(\alpha\)” grabs space from other labels

For each move, we choose the expansion that gives the largest decrease in the energy: \(\Rightarrow\) binary optimization problem

Slide credit: Yuri Boykov
Topics of This Lecture

• Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

• Graph cuts for image segmentation
  - Basic idea
  - s-t MinCut algorithm
  - Extension to non-binary case

• Applications
  - Interactive segmentation
GraphCut Applications: “GrabCut”

- **Interactive Image Segmentation** [Boykov & Jolly, ICCV’01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges

- **Procedure**
  - User marks foreground and background regions with a brush.
  - This is used to create an initial segmentation which can then be corrected by additional brush strokes.

Slide credit: Matthieu Bray
GrabCut: Data Model

- Obtained from interactive user input
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

Global optimum of the energy

Slide credit: Carsten Rother
GrabCut: Coherence Model

• An object is a coherent set of pixels:

\[ \psi(x, y) = \gamma \sum_{(m,n) \in C} \delta[x_n \neq x_m] e^{-\beta \|y_m - y_n\|^2} \]

How to choose \( \gamma \)?
Iterated Graph Cuts

Result

Color model (Mixture of Gaussians)

Energy after each iteration
GrabCut: Example Results

- This is included in the newest version of MS Office!

Image source: Carsten Rother
Applications: Interactive 3D Segmentation

Slide credit: Yuri Boykov

B. Leibe

[Y. Boykov, V. Kolmogorov, ICCV’03]
Summary: Graph Cuts Segmentation

• **Pros**
  - Powerful technique, based on probabilistic model (MRF).
  - Applicable for a wide range of problems.
  - Very efficient algorithms available for vision problems.
  - Becoming a de-facto standard for many segmentation tasks.

• **Cons/Issues**
  - Graph cuts can only solve a limited class of models
    - Submodular energy functions
    - Can capture only part of the expressiveness of MRFs
  - Only approximate algorithms available for multi-label case
References and Further Reading

• A gentle introduction to Graph Cuts can be found in the following paper:

• Read how the interactive segmentation is realized in MS Office 2010

• Try the GraphCut implementation at
  [http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html](http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html)