Computer Vision – Lecture 6
Segmentation as Energy Minimization
07.05.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de

Announcements
• Reminder: Exam dates
  ▶ According to RWTH Online, the exam dates are
    ▶ 1st try Tue 20.08.2019 11:30 – 13:30h
    ▶ 2nd try Wed 25.09.2019 11:30 – 13:30h
• Exam registration should now work
  ▶ Please don’t forget to register for the exam!

Course Outline
• Image Processing Basics
• Segmentation
  ▶ Segmentation as Clustering
  ▶ Graph-theoretic Segmentation
• Recognition
  ▶ Global Representations
  ▶ Subspace representations
• Local Features & Matching
• Object Categorization
• 3D Reconstruction

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:
    \[
    p(\text{data}|\theta) = \prod_{n=1}^{N} p(x_n|\theta)
    \]
• 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
    \[
    \gamma_j(x_n) \leftarrow \frac{\pi_j \mathcal{N}(x_n; \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n; \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \quad n = 1, \ldots, N
    \]
  - M-Step: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \hat{N}_j = \sum_{n=1}^{N} \gamma_j(x_n) = \text{soft number of samples labeled } j
    \]
    \[
    \hat{\mu}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n) x_n
    \]
    \[
    \hat{\Sigma}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n) (x_n - \hat{\mu}_j^{\text{new}})(x_n - \hat{\mu}_j^{\text{new}})^T
    \]

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.
  \(\Rightarrow\) 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 \(W\):
    \[
    c = \frac{1}{\sum_{x \in W} \mathcal{H}(x)} \sum_{x \in W} x \mathcal{H}(x)
    \]
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

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

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

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
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

Markov Random Fields

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

MRF Nodes as Pixels

Original image
Degraded image
Reconstruction from MRF modeling pixel neighborhood statistics

Network Joint Probability

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

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 logarithm of it
  \[ -\log p(x, y) = -\sum_i \log \Phi(x_i, y_i) - \sum_{i,j} \log \Psi(x_i, x_j) \]
  \[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]
- This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call \( E(x, y) \) an energy function.
- \( \phi \) and \( \psi \) are called potentials.

Energy Formulation

- Energy function
  \[ E(x, y) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \]
  \[ \begin{align*}
    \text{Single-node} & \quad \text{Pairwise} \\
    \text{potentials} & \quad \text{potentials}
  \end{align*} \]
- 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)
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 a source-sink graph

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

Simple Example of Energy

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

Unary terms

Pairwise terms

\[ \phi_i(x) = \exp \left\{ \frac{-M_i}{2\sigma^2} \right\} \]

\[ \Delta \phi_i \]

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

(binary object segmentation)

Adding Regional Properties

Regional bias example

Suppose \( I \) and \( I' \) are given "expected" intensities of object and background

\[ \phi(s) = \exp \left\{ -\| I - I' \|^2 / 2\sigma^2 \right\} \]

\[ \phi(t) = \exp \left\{ -\| I - I' \|^2 / 2\sigma^2 \right\} \]

NOTE: hard constrains are not required, in general.

Adding Regional Properties

“expected” intensities of object and background \( I \) and \( I' \) can be re-estimated

\[ \phi(s) = \exp \left\{ -\| I - I' \|^2 / 2\sigma^2 \right\} \]

\[ \phi(t) = \exp \left\{ -\| I - I' \|^2 / 2\sigma^2 \right\} \]

EM-style optimization
Adding Regional Properties

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

Note: \( \phi(t) \) is the cost for the link to the s node! Why?

Given object and background intensity histograms

Adding Regional Properties

How to Set the Potentials? Some Examples

- Color potentials
  - e.g., modeled with a Mixture of Gaussians
  \[ \phi(x_i, y_i; \theta) = -\log \sum_k \theta_k \mu_k \nu_k(x_i) \]

- Edge potentials
  - E.g., a "contrast sensitive Potts model"
  \[ g_{ij}(y) = \theta_2 g_{ij}(y) \delta(x_i \neq x_j) \]

where

\[ g_{ij}(y) = e^{-\beta \|y - y_i\|^2} \]

Parameters \( \theta_1, \theta_2 \) need to be learned, too!

How Does the Code Look Like?

Graph *g*

For all pixels *p*

- Add a node to the graph
  \[ \text{nodeID}(p) = g \rightarrow \text{add_node}(); \]

- Set cost of terminal edges
  \[ \text{set_weights}(\text{nodeID}(p), \text{fgCost}(p), \text{bgCost}(p)); \]

end

for all adjacent pixels *p, q*

- Add weights
  \[ \text{add_weights}(\text{nodeID}(p), \text{nodeID}(q), \text{cost}(p, q)); \]

end

\[ g \rightarrow \text{compute_maxflow}(); \]

label \( = g \rightarrow \text{is_connected_to_source}(\text{nodeID}(p)); \)

is the label of pixel *p* (0 or 1)

\[ a_1 = \text{bg} \quad a_2 = \text{fg} \]

Example: MRF for Image Segmentation

- MRF structure

  \[ \text{Unary likelihood} \]

  \[ \text{Pairwise terms} \]

  \[ \text{MAP solution} \]

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

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

Edges \( E = \{v_i, v_j, \ldots\} \)

Costs \( C = \{c_{ij}, c_{kj}, \ldots\} \)

How Does it Work? The s-t-Mincut Problem

- Source

- Sink

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
### The s-t-Mincut Problem

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

**What is the cost of a s-t-cut?**
Sum of cost of all edges going from S to T

\[ 5 + 2 + 9 = 16 \]

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

\[ 2 + 1 + 4 = 7 \]

### How to Compute the s-t-Mincut?

1. Solve the dual maximum flow problem
2. 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

### History of Maxflow Algorithms

**Augmenting Path and Push-Relabel**

<table>
<thead>
<tr>
<th>Year</th>
<th>Discoverer(s)</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Ford &amp; Fulkerson</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1970</td>
<td>Edmonds &amp; Karp</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1973</td>
<td>Edmonds &amp; Karp</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1974</td>
<td>Edmonds &amp; Karp</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1975</td>
<td>Edmonds &amp; Karp</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1980</td>
<td>Goldberg &amp; Tarjan</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1986</td>
<td>Goldberg &amp; Tarjan</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>Dijkstra</td>
</tr>
</tbody>
</table>

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

### 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. Adjust the capacity of the used edges
4. Repeat until no path can be found

**Algorithms assume non-negative capacity**
Maxflow Algorithms

Flow = 0 + 2

Augmenting Path Based Algorithms

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

Algorithms assume non-negative capacity

---

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. Adjust the capacity of the used edges
4. Repeat until no path can be found

"Residual flows"

Algorithms assume non-negative capacity

---

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. Adjust the capacity of the used edges
4. 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. Adjust the capacity of the used edges
4. 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. Adjust the capacity of the used edges
4. Repeat until no path can be found

Algorithms assume non-negative capacity

Source

Sink

\[ v_1 \]

\[ v_2 \]

\[ 0 \]

\[ 2 \]

\[ 4 \]

\[ 5 \]

\[ 3 \]

\[ 2 \]

\[ 4 \]

\[ 0 \]

Slide credit: Pushmeet Kohli

B. Leibe

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. Adjust the capacity of the used edges
4. Repeat until no path can be found

Algorithms assume non-negative capacity

Source

Sink

\[ v_1 \]

\[ v_2 \]

\[ 0 \]

\[ 2 \]

\[ 1 \]

\[ 5 \]

\[ 4 \]

\[ 3 \]

\[ 2 \]

\[ 3 \]

\[ 4 \]

\[ 0 \]

Slide credit: Pushmeet Kohli

B. Leibe

Applications: Maxflow in Computer Vision

• Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity (m ~ 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

When Can s-t Graph Cuts Be Applied?

\[ E(L) = \sum_p E_p(L_p) + \sum_{p \in P} \sum_{n \in L} E(L_p, L_n) \]

- s-t graph cuts can only globally minimize binary energies that are submodular.

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - 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.
  \( \Rightarrow \) E.g. binary segmentation only...
- We would like to solve also multi-label problems.
  \( \Rightarrow \) 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.
  \( \Rightarrow \) But \( \alpha \)-Expansion has a guaranteed approximation quality (2-approx) and converges in a few iterations.

\( \alpha \)-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.

Example: Stereo Vision

- 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
  - Initial solution
    \( \Rightarrow \) expansion
    \( \Rightarrow \) expansion
    \( \Rightarrow \) expansion
    \( \Rightarrow \) expansion
    \( \Rightarrow \) expansion
  - Depth map

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 Minicut 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.

GrabCut: Data Model

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

GrabCut: Coherence Model

• An object is a coherent set of pixels:

\[ y(x, y) = \gamma \sum_{(m, n) \in C} \delta[x_m \neq x_n] e^{-(A - x)} \]

How to choose \( \gamma \)?

GrabCut: Example Results

• This is included in all MS Office versions since 2010!
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
  - C. Rother, V. Kolmogorov, Y. Boykov, A. Blake. *Interactive Foreground Extraction using Graph Cut*, Microsoft Research Tech Report MSR-TR-2011-46, March 2011

- Try the GraphCut implementation at [https://pub.ist.ac.at/~vnk/software.html](https://pub.ist.ac.at/~vnk/software.html)