GRAPH CUT SELECTIONS

Stephen J. Guy
Photomontage

GrabCut – Interactive Foreground Extraction
Agenda

- Background
  - Segmentation Intro
  - Graph Cut for Segmentation (Boykov & Jolly ‘01)
- Copying
  - Lazy Snapping (Li et al. ’04)
  - GrabCut (Rother et al. ’04)
- Other
  - Texture & Video Synthesis (Kwatra et al. ’04)
Goal

- Splitting Image into similar, meaningful (?) regions
- E.g. Clustering, Edge Detection, *Graph Cuts*

$E(A) = \lambda \cdot R(A) + B(A)$

where

$R(A) = \sum_{p \in \mathcal{P}} R_p(A_p)$

$B(A) = \sum_{\{p,q\} \in \mathcal{N}} B_{\{p,q\}} \cdot \delta(A_p, A_q)$

and

$\delta(A_p, A_q) = \begin{cases} 1 & \text{if } A_p \neq A_q \\ 0 & \text{otherwise.} \end{cases}$

- **Goal:** Find Segmentation, $A$, which minimizes $E(A)$

- **A** – Proposed Segmentation
- **E(A)** – Overall Energy
- **R(A)** – Degree to which pixels fits model
- **B(A)** – Degree to which the cuts breaks up similar pixels
- $\lambda$ – Balance $A()$ and $B()$
Graph Cut Basics

- **Image As Graph**
  - Each pixel is a node
  - Create links between each pixel
  - More nodes/links possible (e.g. foreground node)

- **Graph Cuts**
  - Assign energy for each link based on similarity
  - Break the links with least energy, but
  - Require each pixel is still connected to exactly one foreground or background node

- **Benefits**
  - No Topological constraints on regions (can be concave, unconnected, etc.)
  - Fast (polynomial time) solution – Ford-Fulkerson Algorithm
Graph Cut Overview

(a) Image with seeds.

(b) Graph.

(c) Cut.

(d) Segmentation results.
### Link Weights

<table>
<thead>
<tr>
<th>edge</th>
<th>weight (cost)</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>${p, q}$</td>
<td>$B_{{p,q}}$</td>
<td>${p, q} \in \mathcal{N}$</td>
</tr>
<tr>
<td>${p, S}$</td>
<td>$\lambda \cdot R_p(\text{“bkg”})$</td>
<td>$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$</td>
</tr>
<tr>
<td></td>
<td>$K$</td>
<td>$p \in \mathcal{O}$</td>
</tr>
<tr>
<td></td>
<td>$0$</td>
<td>$p \in \mathcal{B}$</td>
</tr>
<tr>
<td>${p, T}$</td>
<td>$\lambda \cdot R_p(\text{“obj”})$</td>
<td>$p \in \mathcal{P}, p \notin \mathcal{O} \cup \mathcal{B}$</td>
</tr>
<tr>
<td></td>
<td>$0$</td>
<td>$p \in \mathcal{O}$</td>
</tr>
<tr>
<td></td>
<td>$K$</td>
<td>$p \in \mathcal{B}$</td>
</tr>
</tbody>
</table>

- Pixel links based on color/intensity similarities
- Source/Target links based on histogram models of fore/background

\[ K = 1 + \max_{p \in \mathcal{P}} \sum_{q: \{p, q\} \in \mathcal{N}} B_{\{p,q\}} \]
Gestalt Example

- Effect of $\lambda$
- Recall: $E(A) = \lambda \cdot R(A) + B(A)$

(a) Original image
(b) Result for $\lambda = 7 - 43$
(c) Result for $\lambda = 0$
(d) Result for $\lambda = 60$
Lazy Snapping

†Yin Li* ‡Jian Sun †Chi-Keung Tang ‡Heung-Yeung Shum
†Hong Kong University of Science and Technology ‡Microsoft Research Asia

(a) Input image  (b) Object Marking  (c) Boundary editing  (d) Output composition
Lazy Snapping Overview

- **Main Steps:**
  1. Mark strokes as foreground & background
  2. Perform Boykov & Jolly style graphcut
  3. *Edit Boundary*

- **Familiar Formulation**
  - Minimize Gibbs Energy:
    \[
    E(X) = \sum_{i \in V} E_1(x_i) + \lambda \sum_{(i,j) \in E} E_2(x_i, x_j)
    \]
  - \(E_1(X)\) – Cluster (K-means) known foreground/background colors to find representative colors
  - \(E_2(X)\) – Similar to Boykov & Jolly
Lazy Snapping – Issue

- Too Slow to allow for real-time refinement
- Solution? Superpixels!
**Lazy Snapshot - Speedup**

<table>
<thead>
<tr>
<th>Image</th>
<th>Dimension</th>
<th>Nodes Ratio</th>
<th>Edges Ratio</th>
<th>Lag with Pre-segmentation</th>
<th>Lag without Pre-segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boy</td>
<td>(408, 600)</td>
<td>10.7</td>
<td>16.8</td>
<td>0.12s</td>
<td>0.57s</td>
</tr>
<tr>
<td>Ballet</td>
<td>(440, 800)</td>
<td>11.4</td>
<td>18.3</td>
<td>0.21s</td>
<td>1.39s</td>
</tr>
<tr>
<td>Twins</td>
<td>(1024, 768)</td>
<td>20.7</td>
<td>32.5</td>
<td>0.25s</td>
<td>1.82s</td>
</tr>
<tr>
<td>Girl</td>
<td>(768, 1147)</td>
<td>23.8</td>
<td>37.6</td>
<td>0.22s</td>
<td>2.49s</td>
</tr>
<tr>
<td>Grandpa</td>
<td>(1147, 768)</td>
<td>19.3</td>
<td>30.5</td>
<td>0.22s</td>
<td>3.56s</td>
</tr>
</tbody>
</table>
Boundary Editing

- Graph Cut is only step 1
- Allow user to edit boundary directly
- Formulate as Graph Cut
- Allow user to guide boundary regions w/ brush

(a) Foreground seeds $\mathcal{F}$
(b) Background seeds $\mathcal{B}$
(c) Uncertain regions $\mathcal{U}$
(d) Pixels ignored by graph cut
(e) Polygon vertices and lines

(a) (b)
Lazy Snapping – Video
User Study

- Compared Lazy Snapping to Photoshop
User Study

- Results
  - Easy of Use – Lazy Snapping 20% less mistakes
  - Speed – Lazy Snapping 60% less time
  - Accuracy – Lazy Snapping 60% less pixels wrong

- Feedback
  - “Almost Magic”
  - “Much Easier”

- Suggestions
  - Get rid of 2 step process
  - Make it easy to switch between graph cut strokes and boundary editing
More Results

(a) Girl (4/2/12)
(b) Ballet (4/7/14)
(c) Boy (6/2/13)

(c) Grandpa (4/2/11)
(d) Twins (4/4/12)
GrabCut
Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother
Vladimir Kolmogorov
Andrew Blake

Microsoft Research Cambridge-UK
Big Changes

- New model of foreground & background pixels:
  - GMM vs. K-mean colors vs. Intensity Histogram

- More automation:
  - No need to specify foreground
  - Iterative Graph Cuts
What GrabCut does

Magic Wand (198?)

Intelligent Scissors
Mortensen and Barrett (1995)

GrabCut

User Input

Regions

Boundary

Regions & Boundary
1. Assign Selection box or Lasso as background
2. Assign other pixels as unknown
3. Learn Model of Fore/Background
4. Assign Link Energy, GraphCut (Boykov & Jolly)
5. Use GraphCut results to assign Fore/Background
6. Goto Step 3
7. Allow user to add cleanup strokes
Graph Cuts

Boykov and Jolly (2001)

Image

Foreground (source)

Background (sink)

Min Cut

**Cut:** separating source and sink; Energy: collection of edges

**Min Cut:** Global minimal energy in polynomial time
Iterated Graph Cut

User Initialisation

K-means for learning colour distributions

Graph cuts to infer the segmentation
Iterated Graph Cuts

Guaranteed to converge

Result

Energy after each Iteration
Colour Model

Gaussian Mixture Model (typically 5-8 components)

Iterated graph cut

GrabCut – Interactive Foreground Extraction
Moderately straightforward examples

... GrabCut completes automatically
Difficult Examples

Camouflage & Low Contrast

Fine structure

No telepathy
Comparison

Boykov and Jolly (2001)

GrabCut

User Input

Result

Error Rate: 0.72%

Error Rate: 0.72%
Border Matting

- Hard Segmentation
- Automatic Trimap
- Soft Segmentation

GrabCut – Interactive Foreground Extraction
Natural Image Matting

Solve

Ruzon and Tomasi (2000): Alpha estimation in natural images
Fit a smooth alpha-profile with parameters
Results

GrabCut – Interactive Foreground Extraction
Summary

Magic Wand (198?)
Intelligent Scissors Mortensen and Barrett (1995)
Graph Cuts Boykov and Jolly (2001)
LazySnapping Li et al. (2004)
GrabCut Rother et al. (2004)
<table>
<thead>
<tr>
<th></th>
<th>Boykov&amp;Jolly</th>
<th>Lazy Snapping</th>
<th>GrabCut</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Foreground &amp; Background Model</strong></td>
<td>Histogram of Intensities</td>
<td>K-means clustering, 64 representative colors</td>
<td>K-mean</td>
</tr>
<tr>
<td><strong>Background Source</strong></td>
<td>User Strokes</td>
<td>User Strokes</td>
<td>Initial Box (Lasso) + User Strokes</td>
</tr>
<tr>
<td><strong>Foreground Source</strong></td>
<td>User Strokes</td>
<td>User Strokes</td>
<td>None or User Strokes</td>
</tr>
<tr>
<td><strong>Boundary Term</strong></td>
<td>Similarity Penalty</td>
<td>Similarity Penalty</td>
<td>Similarity Penalty</td>
</tr>
<tr>
<td><strong>Segmentation</strong></td>
<td>GraphCut</td>
<td>Superpixels + GraphCut</td>
<td>Iterative GraphCut</td>
</tr>
<tr>
<td><strong>Refinement</strong></td>
<td>User Strokes</td>
<td>Boundary Editing</td>
<td>User Strokes</td>
</tr>
</tbody>
</table>
Texture Synthesis Revisited

Graphcut Textures: Image and Video Synthesis Using Graph Cuts

Vivek Kwatra  Arno Schödl  Irfan Essa  Greg Turk  Aaron Bobick

GVU Center / College of Computing
Georgia Institute of Technology
http://www.cc.gatech.edu/cpl/projects/graphcuttextures

SIGGRAPH
Basic Idea

- Copy overlapping patches one at a time
- Find the seam which minimize change in neighboring pixels
- Formulate as Graph Cut

\[ M(s, t, A, B) = \|A(s) - B(s)\| + \|A(t) - B(t)\| \]
Extends well into 3D

Input Video

Window in which seam computed

Similar Frames

Computed Seam

Input Video

Output Video

Shown in 3D

Seams In Video

3D visualization of seam surface