Graph Cut

Frédo Durand
MIT - EECS
Debugging

• **Doubt everything**
• **Debug pieces in isolation**
  – Binary search/divide and conquer
• **Display/print everything**
  – In particular intermediate results
• **Create simple inputs**
  – where you can easily manually compute the result
  – e.g. constant image, edge image, etc.
  – use small images (e.g. 3x3)
  – including (especially) inputs to intermediate stages
  – use input that isolates different failure modes
• **Change one thing at a time**
  – e.g. to verify that a given command has the effect you want, modify it to break it

Tuesday, March 8, 2011
Last Tuesday: optimization

• Relied on a smoothness term
  – values are assumed to be smooth across image
• User provided boundary condition
Last Thursday: Bayesian Matting

- Separation of foreground & background
  - Partial coverage with fractional alpha
  - User provides a trimap
  - Bayesian approach
    - Model color distribution in F & B
    - Alternatively solve for $\alpha$, then F&B

- Solve for each pixel independently
  - using a “data term”
More foreground background

• Today, we want to exploit both data and smoothness

• Smoothness
  – The alpha value of a pixel is likely to be similar to that of its neighbors
  – Unless the neighbors have a very different color

• Data
  – Color distribution of foreground and background
Multiple options

• *Keep using continuous optimization*
  – See e.g. Chuang’s dissertation, Levin et al. 2006
  – Pros: Good treatment of partial coverage
  – Cons: requires the energy/probabilities to be well behaved to be solvable

• *Quantize the values of alpha & use discrete optimization*
  – Pros: allows for flexible energy term, efficient solution
  – Cons: harder to handle fractional alpha
Today’s overview

- Interactive image segmentation using graph cut
- Binary label: foreground vs. background
- User labels some pixels
  - similar to trimap, usually sparser
- Exploit
  - Statistics of known Fg & Bg
  - Smoothness of label
- Turn into discrete graph optimization
  - Graph cut (min cut / max flow)

Images from
European Conference on Computer Vision 2006: “Graph Cuts vs. Level Sets”, Y. Boykov (UWO), D. Cremers (U. of Bonn), V. Kolmogorov (UCL)
References

• Combination of

• Yuri Boykov, Marie-Pierre Jolly Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images In International Conference on Computer Vision (ICCV), vol. I, pp. 105-112, 2001

• C. Rother, V. Kolmogorov, A. Blake. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04), 2004
Cool motivation

• The rectangle is the only user input
• [Rother et al.’s grabcut 2004]
Graph cut is a very general tool

- Stereo depth reconstruction
- Texture synthesis
- Video synthesis
- Image denoising

3D model of scene
Questions?
Energy function

- **Unknown**: labeling: one value per pixel, F or B
- **Energy(labeling) = data + smoothness**
  - Very general situation
  - Will be minimized
- **Data**: for each pixel
  - Probability that this color belongs to F (resp. B)
  - Similar in spirit to Bayesian matting
- **Smootheness (aka regularization)**: per neighboring pixel pair
  - Penalty for having different label
  - Penalty is downweighted if the two pixel colors are very different
  - Similar in spirit to bilateral filter
Data term

• A.k.a regional term (because integrated over full region)

• \( D(L) = \sum_i -\log h[L_i](C_i) \)

• Where \( i \) is a pixel
  \( L_i \) is the label at \( i \) (F or B),
  \( C_i \) is the pixel value
  \( h[L_i] \) is the histogram of the observed Fg (resp Bg)

• Note the minus sign
Color Histograms

• Discretize R, G, B (e.g. 20x20x20)
  – store 20x20x20 counters

• For each pixel in the image
  – add one to the appropriate counter

Cool video

- [http://senchalabs.github.com/philoGL/PhiloGL/examples/histogram/](http://senchalabs.github.com/philoGL/PhiloGL/examples/histogram/)

**Real time color 3D histogram analysis**

Visualize frame by frame color decomposition. Choose from different color schemes below. Feel free to pause the video and to zoom/pan the histogram below.

Choose a Color Scheme:

- RGB Cube
- Chroma Luma Hue
- Chroma Value Hue

Other Options:

- Grain: None, Small, Large
- Background: 0.3

Copyright © Sencha Inc. - Authors: Maria Luz Cabello and Nicolas Garcia Belmonte.
Data term

- A.k.a regional term (because integrated over full region)

- $D(L) = \sum_i -\log h[L_i](C_i)$

  Where $i$ is a pixel
  $L_i$ is the label at $i$ (F or B),
  $C_i$ is the pixel value

  $h[L_i]$ is the histogram of the observed Fg (resp Bg)

- Here we use the histogram while in Bayesian matting we used a Gaussian model. This is partially because discrete optimization has fewer computational constraints. No need for linear least square
Hard constraints

• The user has provided some labels
• The quick and dirty way to include constraints into optimization is to replace the data term by a huge penalty $K$ if not respected.
  - $D(L_i)=0$ if respected
  - $D(L_i) = K$ if not respected
    - e.g. $K= -\#pixels$
Smoothness term

- a.k.a boundary term, a.k.a. regularization
- \( S(L) = \sum_{\{i, j\}_2 N} B(C_i, C_j) \delta(L_i - L_j) \)
- Where \( i, j \) are neighbors
  - e.g. 8-neighborhood
    (but I show 4 for simplicity)
- \( \delta(L_i - L_j) = 0 \) if \( L_i = L_j \), 1 otherwise
- \( B(C_i, C_j) \) is high when \( C_i \) and \( C_j \) are similar, low if there is a discontinuity between those two pixels
  - e.g. \( \exp\left(-\|C_i - C_j\|^2/2\sigma^2\right) \)
  - where \( \sigma \) can be a constant or the local variance
- Note positive sign
Recap: Energy function

- **Labeling:** one value $L_i$ per pixel, F or B
- **Energy(labeling) = Data + Smoothness**
- **Data:** for each pixel
  - Probability that this color belongs to F (resp. B)
  - Using histogram
  - $D(L) = \sum_i -\log h[L_i](C_i)$
- **Smoothness (aka regularization):** per neighboring pixel pair
  - Penalty for having different label
  - Penalty is downweighted if the two pixel colors are very different
  - $S(L) = \sum_{\{i, j\} \in N} B(C_i, C_j) \delta(L_i-L_j)$

One labeling (ok, not best)

Data

Smoothness

Tuesday, March 8, 2011
Optimization

• $E(L) = D(L) + \lambda S(L)$
  – $\lambda$ is a black-magic constant

• Find the labeling that minimizes $E$

• In this case, how many possibilities?
  – $2^9$ (512)
  – We can try them all!
  – What about megapixel images?
Discussion about lambda

- Lambda balances the smoothness and data terms
- Bad news: not invariant to image size

- The smoothness (boundary) term scales linearly
  – Because it depends on the length
  – You only pay it at the boundary between Fg and Bg
- The data (regional) term scales quadratically
  – Because it depends on the area

- Makes life difficult.
Questions?

• Recap:
  – Labeling F or B
  – Energy(Labeling) = Data+Smoothness
  – Need efficient way to find labeling with lowest energy
Labeling as a graph problem

• Each pixel = node
• Add two label nodes F & B
• Labeling: link each pixel to either F or B

F F B
F F B
F B B

Desired result

Tuesday, March 8, 2011
Idea

- Start with a graph with too many edges
  - Represents all possible labeling
  - Strength of edges depends on data and smoothness terms
- solve as min cut
Data term

- Put one edge between each pixel and both F & G
- Weight of edge = minus data term
  - Don’t forget huge weight for hard constraints
  - Careful with sign
Smoothness term

- Add an edge between each neighbor pair
- Weight = smoothness term
Min cut

- Energy optimization equivalent to graph min cut
- Cut: remove edges to disconnect F from B
- Minimum: minimize sum of cut edge weight
Min cut

- Graph with one source & one sink node
- Edge = bridge
- Edge label = cost to cut bridge
- What is the min-cost cut that separates source from sink
Min cut $\iff$ labeling

- In order to be a cut:
  - For each pixel, either the F or G edge has to be cut
- In order to be minimal
  - Only one edge label per pixel can be cut (otherwise could be added)
Min cut <=> optimal labeling

- Energy = - \( \sum \) weight of remaining links to F & B  
  + \( \sum \) weight cut neighbor links
Min cut <=> optimal labeling

- Energy = - $\sum$ all weights to F & B
  + $\sum$ weight of cut links to F & B
  + $\sum$ weight cut neighbor links

- Minimized when last 2 terms are minimized
Questions?

• Recap: We have turned our pixel labeling problem into a graph min cut
  – nodes = pixels + 2 labels
  – edges from pixel to label = data term
  – edges between pixels = smoothness
• Now we need to solve the min cut problem
Min cut

- Graph with one source & one sink node
- Edge = bridge; Edge label = cost to cut bridge
- Find the min-cost cut that separates source from sink
  - Turns out it’s easier to see it as a flow problem
  - Hence source and sink
Max flow

- Directed graph with one source & one sink node
- Directed edge = pipe
- Edge label = capacity
- What is the max flow from source to sink?
Max flow

- Graph with one source & one sink node
- Edge = pipe
- Edge label = capacity
- What is the max flow from source to sink?
Max flow

- What is the max flow from source to sink?
- Look at residual graph
  - remove saturated edges (green here)
  - min cut is at boundary between 2 connected components
Max flow

• What is the max flow from source to sink?
• Look at residual graph
  – remove saturated edges (gone here)
  – min cut is at boundary between 2 connected components

Source

Sink
Equivalence of min cut / max flow

The three following statements are equivalent

- The maximum flow is $f$
- The minimum cut has weight $f$
- The residual graph for flow $f$ contains no directed path from source to sink
Questions?

• Recap:
  – We have reduced labeling to a graph min cut
    • vertices for pixels and labels
    • edges to labels (data) and neighbors (smoothness)
  – We have reduced min cut to max flow

  – Now how do we solve max flow???
Max flow algorithm

• We will study a strategy where we keep augmenting paths (Ford-Fulkerson, Dinic)
• Keep pushing water along non-saturated paths
  – Use residual graph to find such paths
Max flow algorithm

Set flow to zero everywhere

Big loop
  compute residual graph
  Find path from source to sink in residual
    If path exist add corresponding flow
    Else
      Min cut = {vertices reachable from source; other vertices}
      terminate

Animation at
http://www.cse.yorku.ca/~aaw/Wang/MaxFlowStart.htm

Tuesday, March 8, 2011
Shortest path anyone?

- e.g. Dijkstra, A*
Efficiency concerns

- The search for a shortest path becomes prohibitive for the large graphs generated by images
- For practical vision/image applications, better (yet related) approaches exist


- Maintain two trees from sink & source.
- Augment tree until they connect
- Add flow for connection
- Can require more iterations because not shortest path
  But each iteration is cheaper because trees are reused
Questions?
Graph Cuts and Efficient N-D Image Segmentation

Yuri Boykov, Gareth Funka-Lea


Figure 8. Segmentation of photographs (early 20th century). Initial segmentation for a given set of hard constraints (seeds) takes less than a second for most 2D images (up to $1000 \times 1000$). Correcting seeds are incorporated in the blink of an eye. Thus, the speed of our method for photo editing mainly depends on time for placing seeds. An average user will not need much time to enter seeds in (a) and (b).
• **Importance of smoothness**

   - a cut
   - a flow

   (a) Boundary cues with topological constraints
   (b) Region-based cues (only)

---

**From Yuri Boykov, Gareth Funka-Lea**
Data (regional) term

(a) Original B&W photo
(b) Segmentation results
(c) Details of segmentation with regional term
(d) Details of segmentation without regional term
Questions?
Recap: Energy for images

• **Continuous optimization**
  – e.g. Poisson, colorization & other scribbles
  – easy when linear least square

• **Graphcut**
  – e.g. segmentation, stereo
  – only discrete values
  – more general energy
Questions?
Grabcut

- Rother et al. 2004
- Less user input: only rectangle
- Handle color
- Extract matte as post-process

Figure 1: Three examples of GrabCut. The user drags a rectangle loosely around an object. The object is then extracted automatically.
• **Model 3D color histogram with Gaussians**
  – Because brute force histogram would be sparse
    • Although I question this. My advice: go brute force, use a volumetric grid in RGB space and blur the histogram
  – Gaussian Mixture Model (GMM)
  – Just means histogram = sum of Gaussians
    • They advise 5 Gaussians
Getting a GMM

- Getting one Gaussian is easy: mean / covariance
- To get K Gaussians, we cluster the data
  - And use mean/covariance of each cluster
- The K-mean clustering algorithm can do this for us
  - Idea: define clusters and their center. Points belong to the cluster with closest center

Take K random samples as seed centers
Iterate:
  For each sample
    Assign to closest cluster
  For each cluster
    Center = mean of samples in cluster
Grabcut: Iterative approach

• Initialize
  – Background with rectangle boundary pixels
  – Foreground with the interior of rectangle

• Iterate until convergence
  – Compute color probabilities (GMM) of each region
  – Perform graphcut segmentation

• Apply matting at boundary

• Potentially, user edits to correct mistakes
Iterated Graph Cut

User Initialisation

K-means for learning
colour distributions

Graph cuts to infer the segmentation

slide: Rother et al.

GrabCut – Interactive Foreground Extraction
Iterated Graph Cuts

Result

Energy after each Iteration

Guaranteed to converge
Border matting

(a) Original image with trimap overlaid. (b) Notation for contour parameterisation and distance map. Contour $C$ (yellow) is obtained from hard segmentation. Each pixel in $T_U$ is assigned values (integer) of contour parameter $t$ and distance $r_n$ from $C$. Pixels shown share the same value of $t$. (c) Soft step-function for $\alpha$-profile $g$, with centre $\Delta$ and width $\sigma$. 
Figure 5: User editing. After the initial user interaction and segmentation (top row), further user edits (fig. 3) are necessary. Marking roughly with a foreground brush (white) and a background brush (red) is sufficient to obtain the desired result (bottom row).
Moderately straightforward examples

GrabCut completes automatically

slide: Rother et al.  
GrabCut – Interactive Foreground Extraction  
Tuesday, March 8, 2011
Difficult Examples

Camouflage & Low Contrast

Initial Rectangle

Initial Result

Fine structure

No telepathy

GrabCut – Interactive Foreground Extraction

slide: Rother et al.

Tuesday, March 8, 2011
Comparison

Boykov and Jolly (2001)

GrabCut

User Input

Result

Error Rate: 0.72%

Error Rate: 0.72%

slide: Rother et al.

GrabCut – Interactive Foreground Extraction
• http://www.csd.uwo.ca/faculty/yuri/Abstracts/eccv06-tutorial.html
• http://www.cse.yorku.ca/~aaw/Wang/MaxFlowStart.htm
• http://www.cc.gatech.edu/cpl/projects/graphcuttextures/