6.815 Digital and Computational Photography
6.865 Advanced Computational Photography

Graph Cut

Frédo Durand
MIT - EECS

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 α, then F&B
- **Solve for each pixel independently**
  - using a "data term"

More foreground background

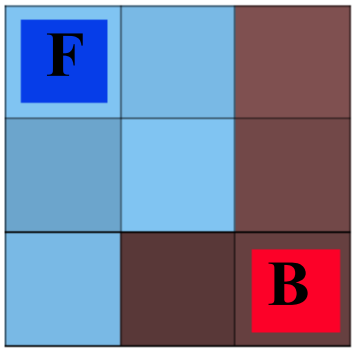
- **Today, we want to exploit both data and smoothness**
- **Smoothness**
  - The alpha value of a pixel is likely to be similar to that of its neighbors
  - Unless the neighbors have a very different color
- **Data**
  - Color distribution of foreground and background

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

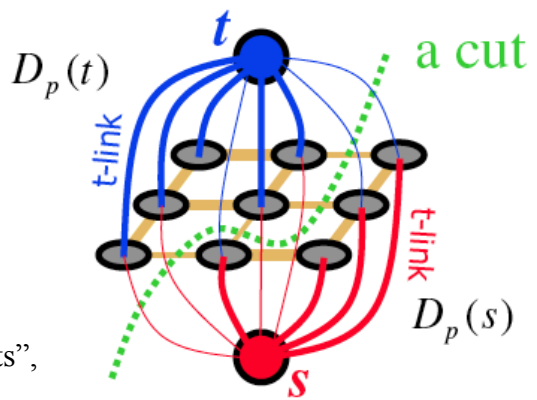
- **Interactive image segmentation using graph cut**
- **Binary label: foreground vs. background**
- **User labels some pixels**
  - similar to trimap, usually sparser
- **Exploit**
  - Statistics of known Fg & Bg
  - Smoothness of label
- **Turn into discrete graph optimization**
  - Graph cut (min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



F F B

F F B

F B B



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

Refs

- **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

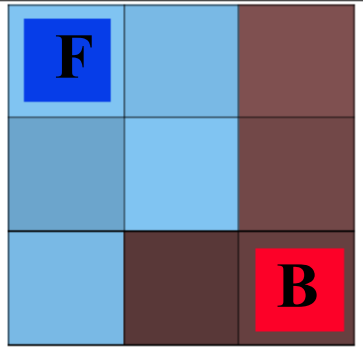
- **Stereo depth reconstruction**
- **Texture synthesis**
- **Video synthesis**
- **Image denoising**

3D model of scene

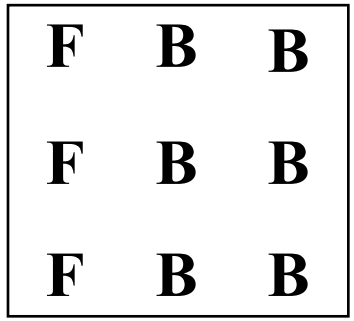
Questions?

Energy function

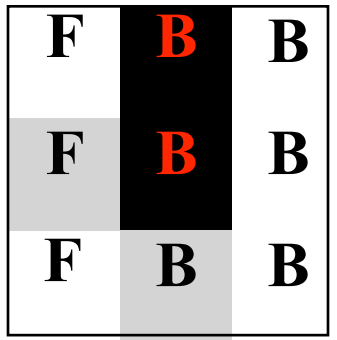
- **Labeling: one value per pixel, F or B**
- **Energy(labeling) = data + smoothness**
  - Very general situation
  - Will be minimized
- **Data: for each pixel**
  - Probability that this color belongs to F (resp. B)
  - Similar in spirit to Bayesian matting
- **Smoothness (aka regularization): per 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



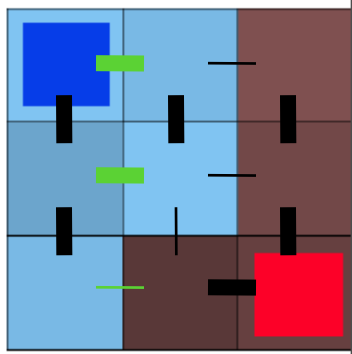
One labeling (ok, not best)



Data

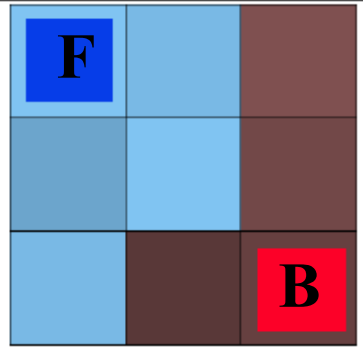


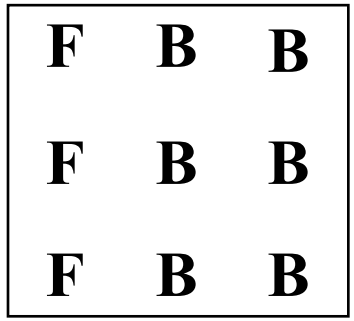
Smoothness

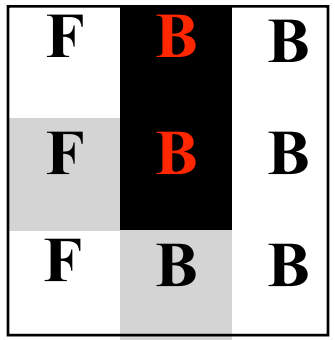


Data term

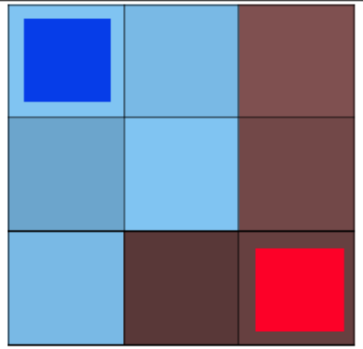
- A.k.a regional term
  (because integrated over full region)
- $D(L)=\sum_i -\log h[L_i](C_i)$
- Where *i* is a pixel
  L_i is the label at *i* (F or B),
  C_i is the pixel value
  $h[L_i]$ is the histogram of the observed Fg (resp Bg)
- Note the minus sign







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)

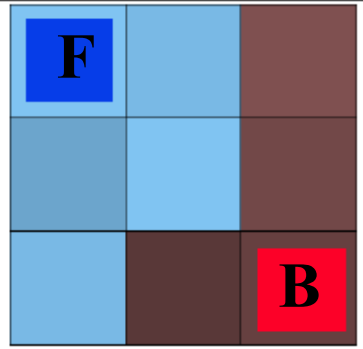
F	B	B
F	B	B
F	B	B

F	B	B
F	B	B
F	B	B

- 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

Histograms

Hard constraints

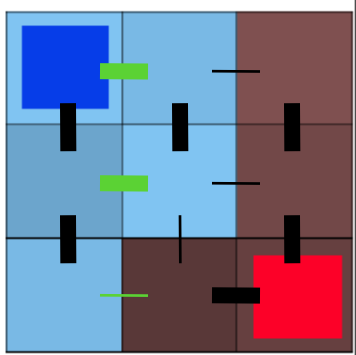


- **The user has provided some labels**
- **The quick and dirty way to include constraints into optimization is to replace the data term by a huge penalty K if not respected.**
- **D(Li)=0 if respected**
- **D(Li) = K if not respected**
  - e.g. K= - #pixels

Smoothness term

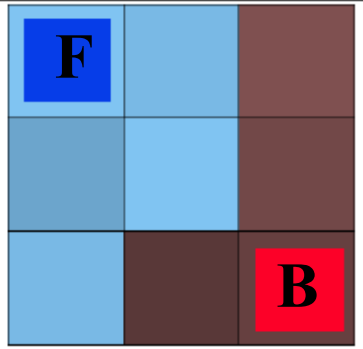
- **a.k.a boundary term, a.k.a. regularization**
- **$S(L)=\sum_{\{j,\, i\}2\ N} B(C_i,C_j)\ \delta(L_i-L_j)$**
- **Where i,j are neighbors**
  - e.g. 8-neighborhood
    (but I show 4 for simplicity)
- **$\delta(L_i-L_j)$ is 0 if $L_i=L_j$, 1 otherwise**
- **$B(C_i,C_j)$ is high when C_i and C_j are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

F	**B**	**B**
F	**B**	**B**
F	**B**	**B**

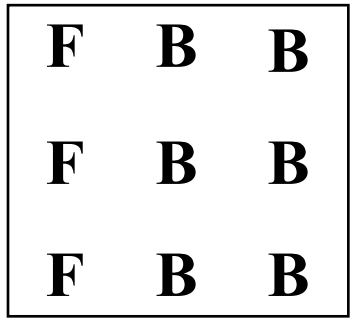


Recap: Energy function

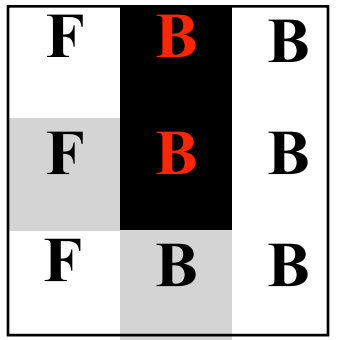
- **Labeling: one value Li per pixel, F or B**
- **Energy(labeling) = Data + Smoothness**
- **Data: for each pixel**
  - Probability that this color belongs to F (resp. B)
  - Using histogram
  - $D(L)=\sum_i -\log h[L_i](C_i)$
- **Smoothness (aka regularization): per neighboring pixel pair**
  - Penalty for having different label
  - Penalty is downweighted if the two pixel colors are very different
  - $S(L)=\sum_{\{j,\, i\}2\ N} B(C_i,C_j)\ \delta(L_i-L_j)$
    -



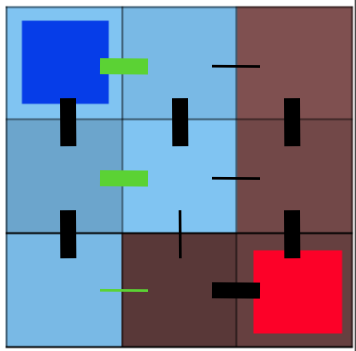
One labeling (ok, not best)



Data

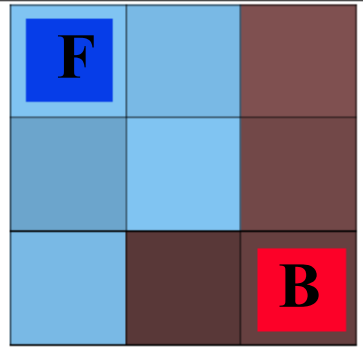


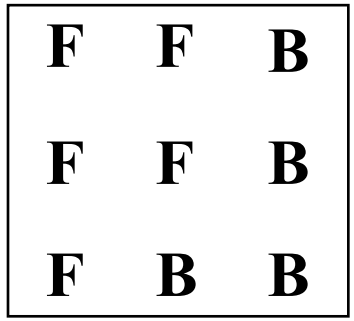
Smoothness

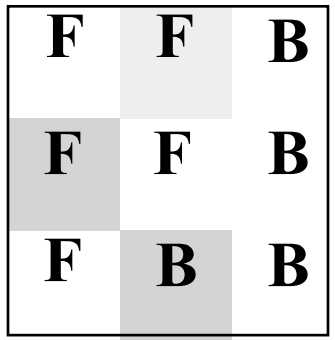


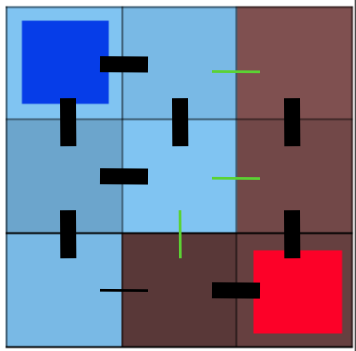
Optimization

- **E(L)=D(L)+λ S(L)**
- **λ 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?









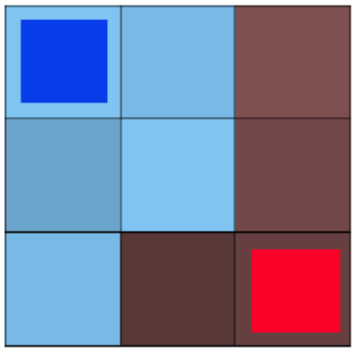
- **DISCUSS AREA VS PERIMTER SCALING**
- **and how it affects lambda**

Questions?

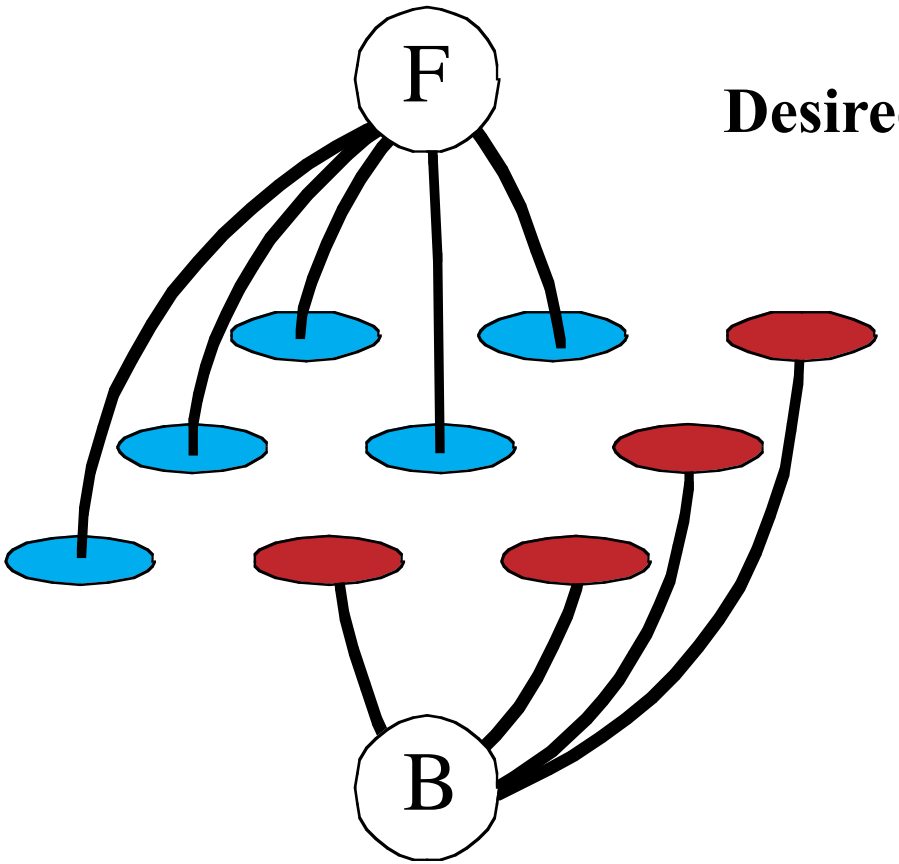
- **Recap:**
  - Labeling F or B
  - Energy(Labeling) = Data+Smoothness
  - Need efficient way to find labeling with lowest energy

Labeling as a graph problem

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



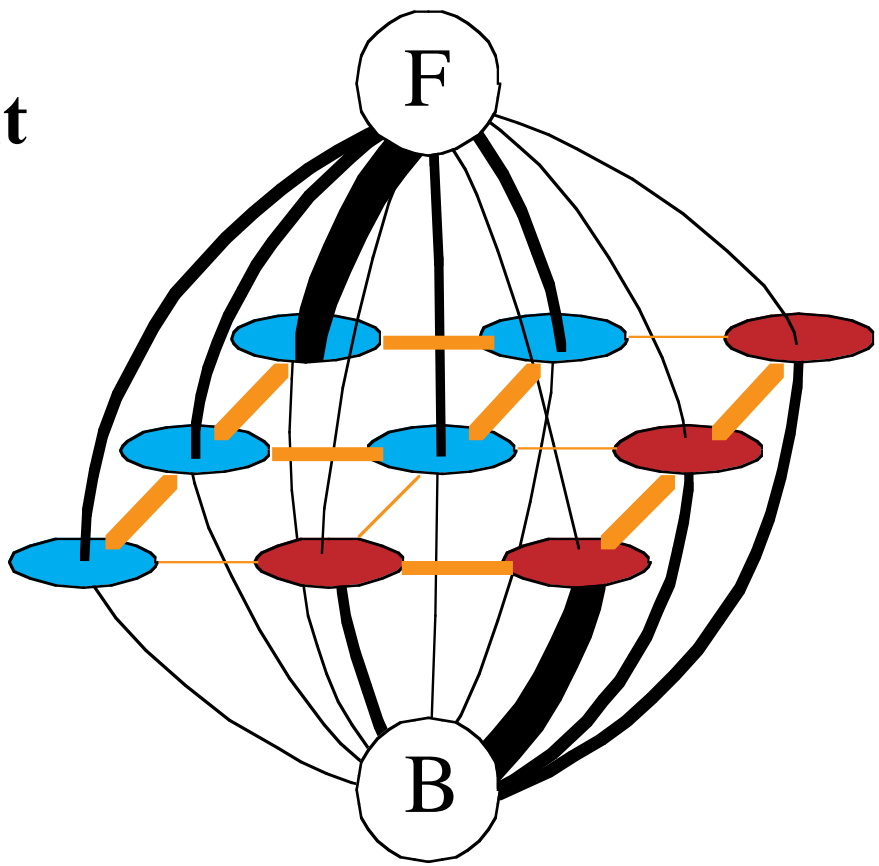
F	F	B
F	F	B
F	B	B



Desired result

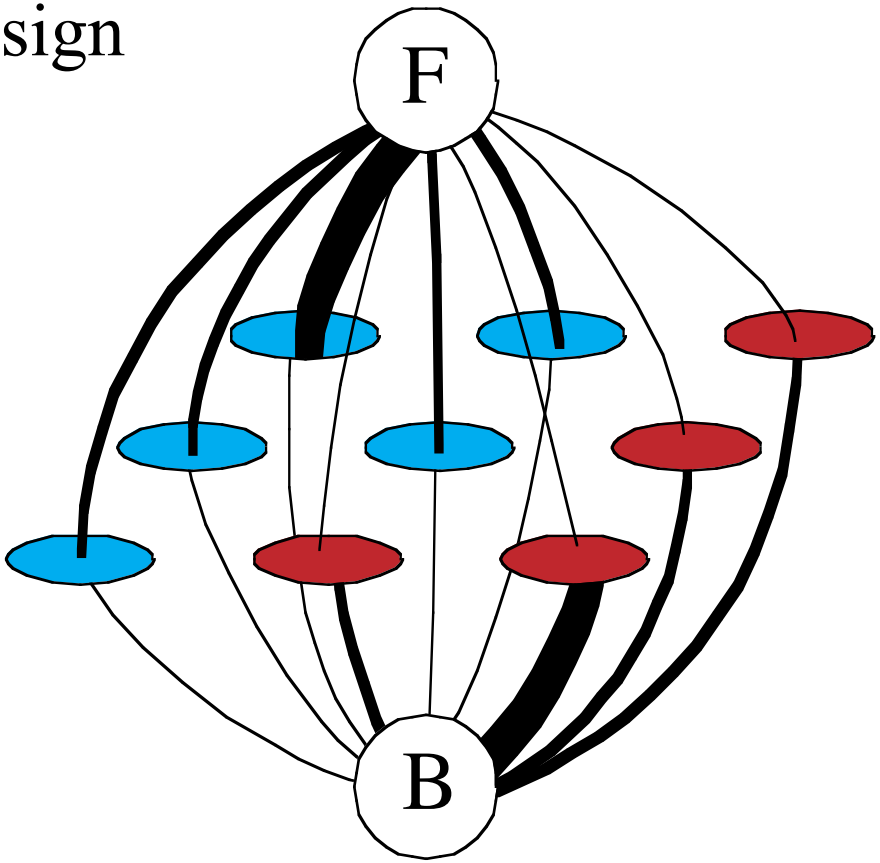
Idea

- **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**



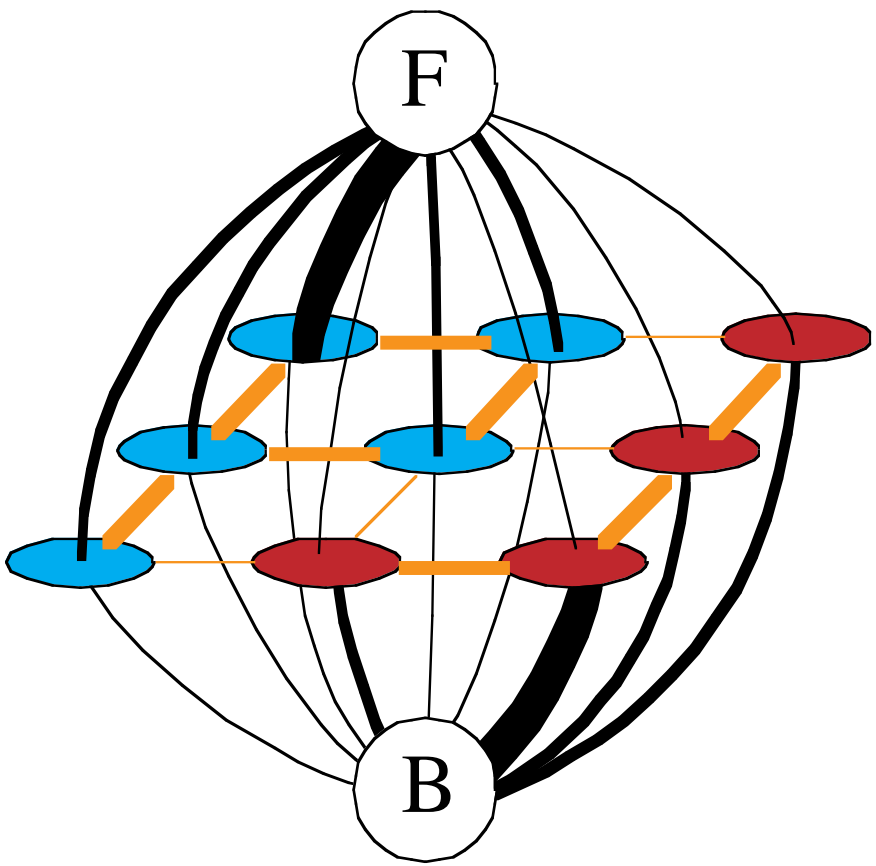
Data term

- **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



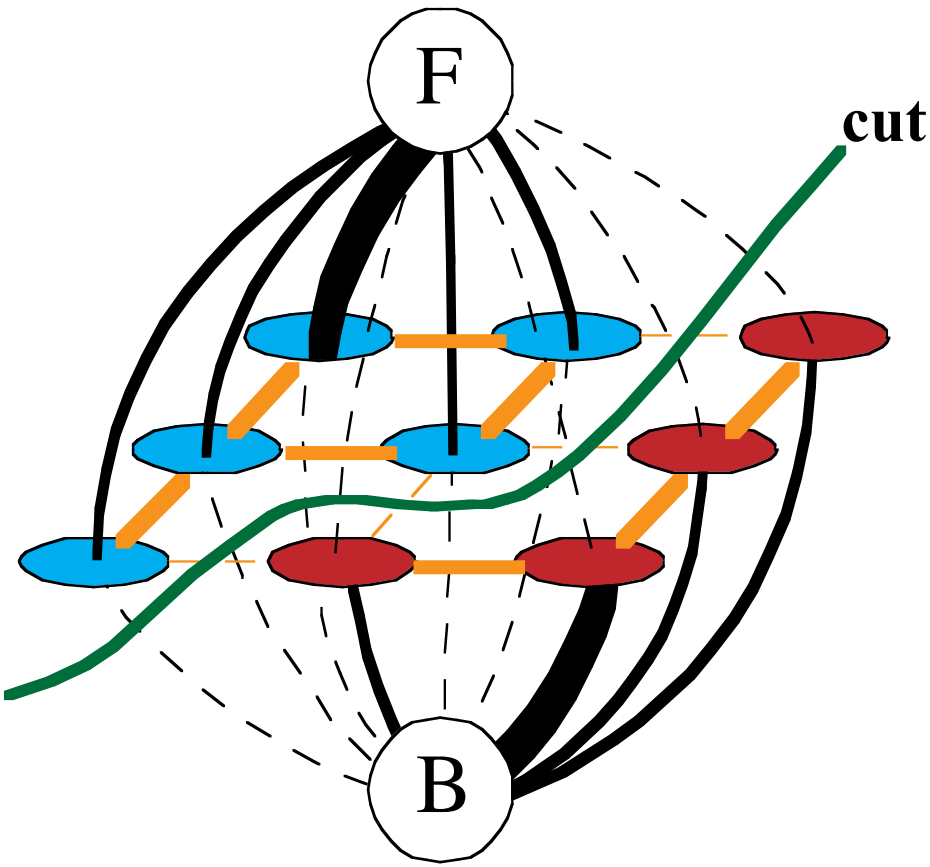
Smoothness term

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



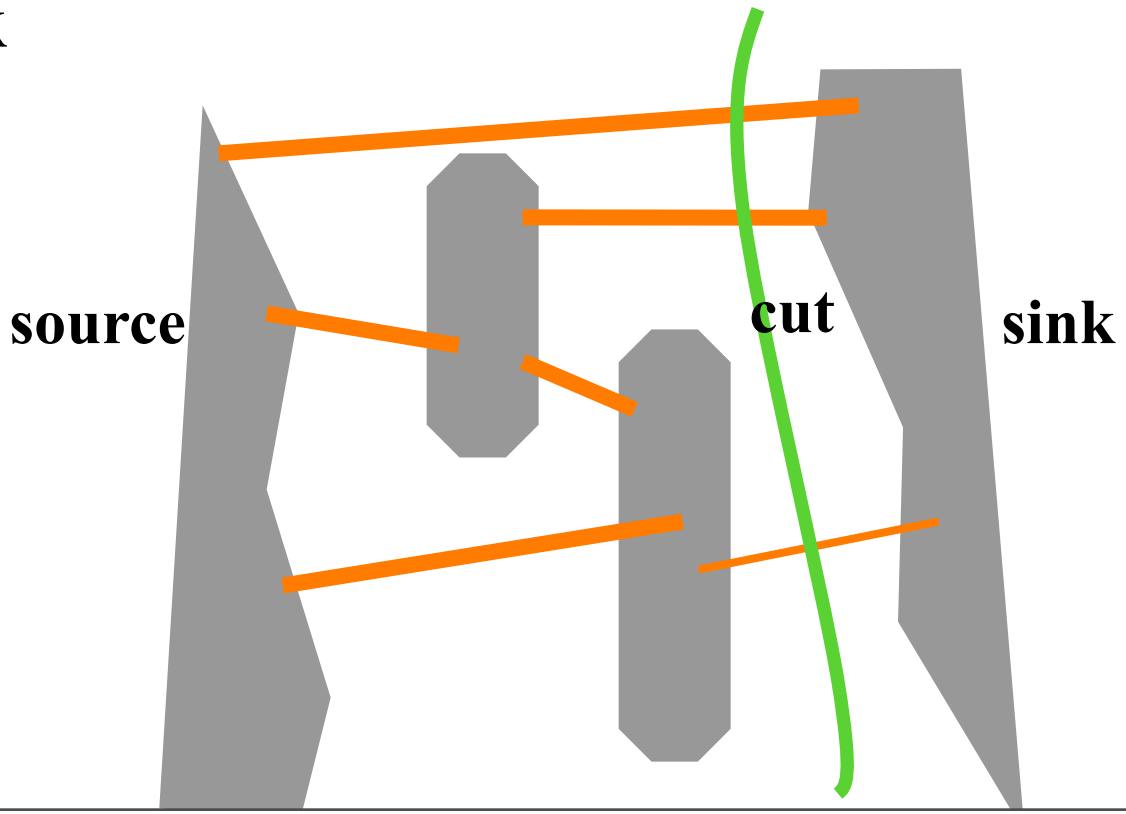
Min cut

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



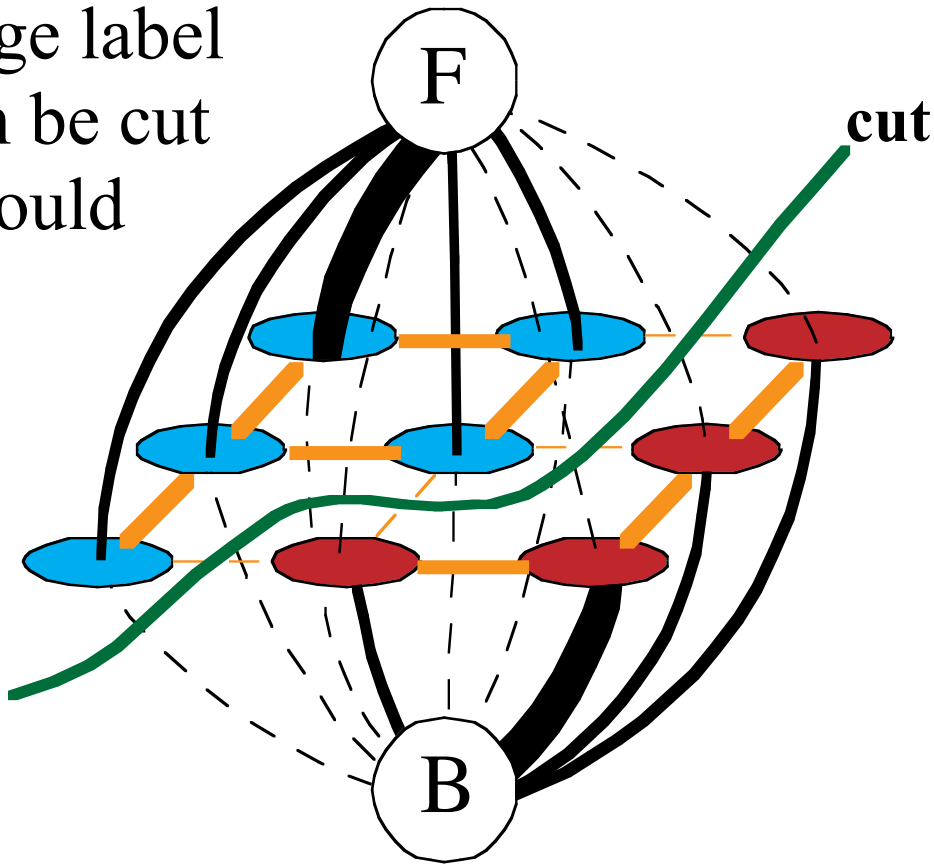
Min cut

- 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



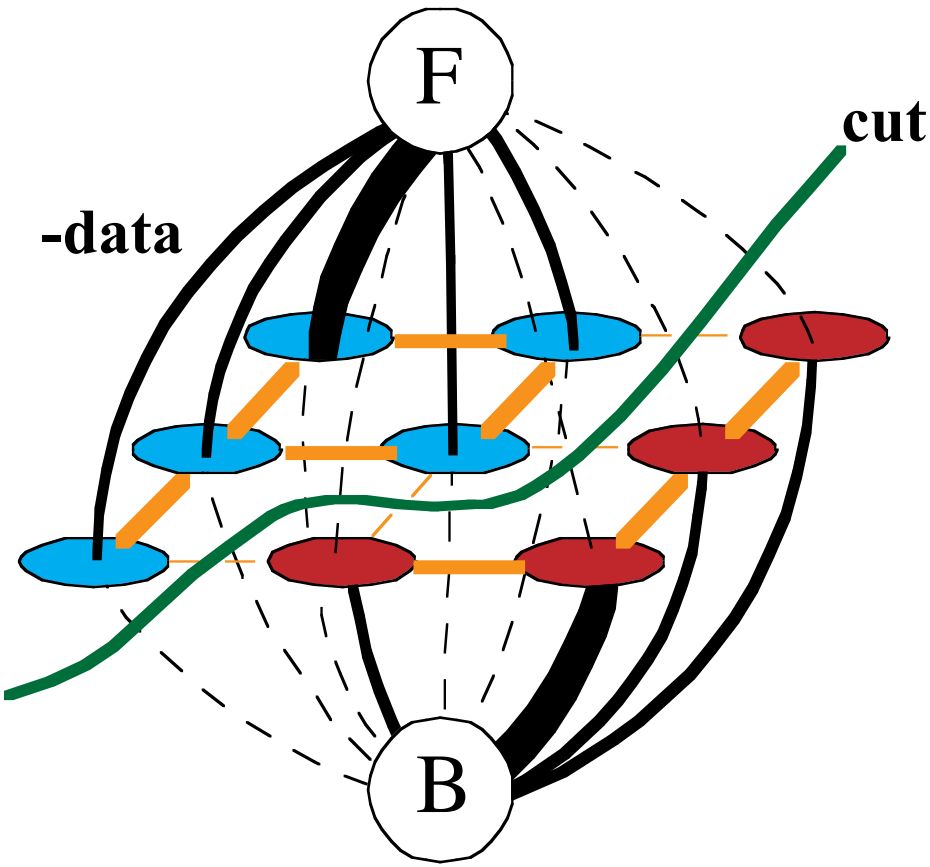
Min cut <=> labeling

- **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)



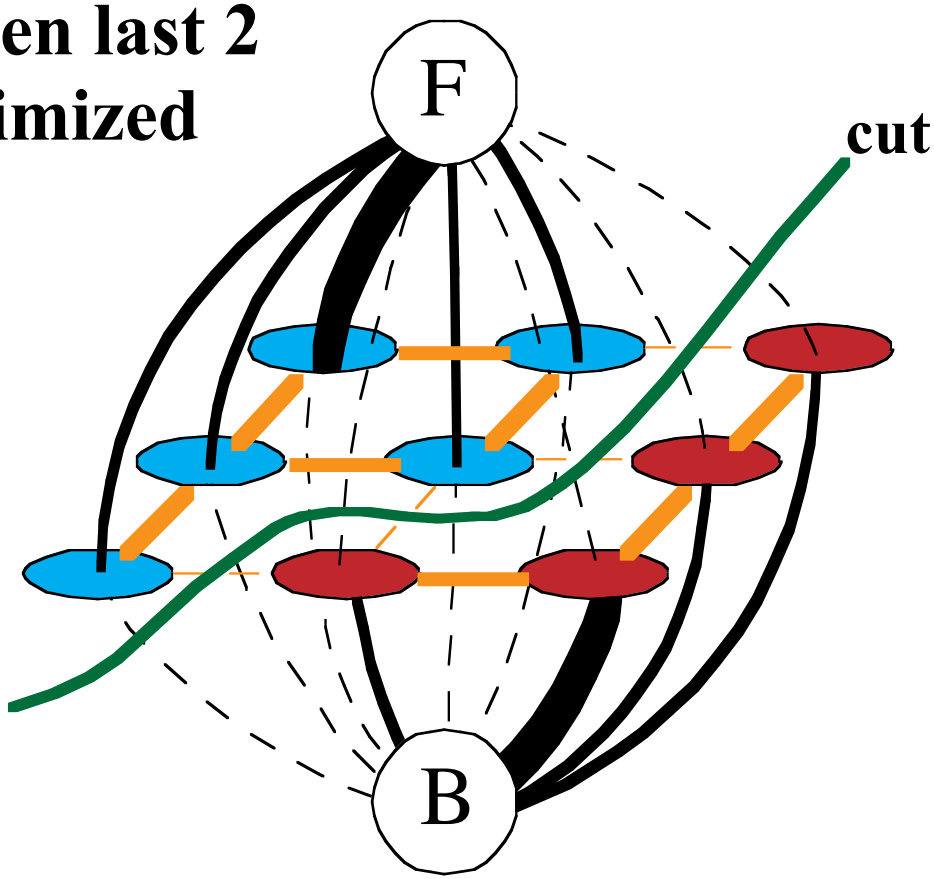
Min cut <=> optimal labeling

- **Energy = - Σ weight of remaining links to F & B**
  + Σ weight cut neighbor links



Min cut <=> optimal labeling

- **Energy = - Σ all weights to F & B**
  + Σ weight of cut links to F & B
  +Σ weight cut neighbor links
- **Minimized when last 2 terms are minimized**

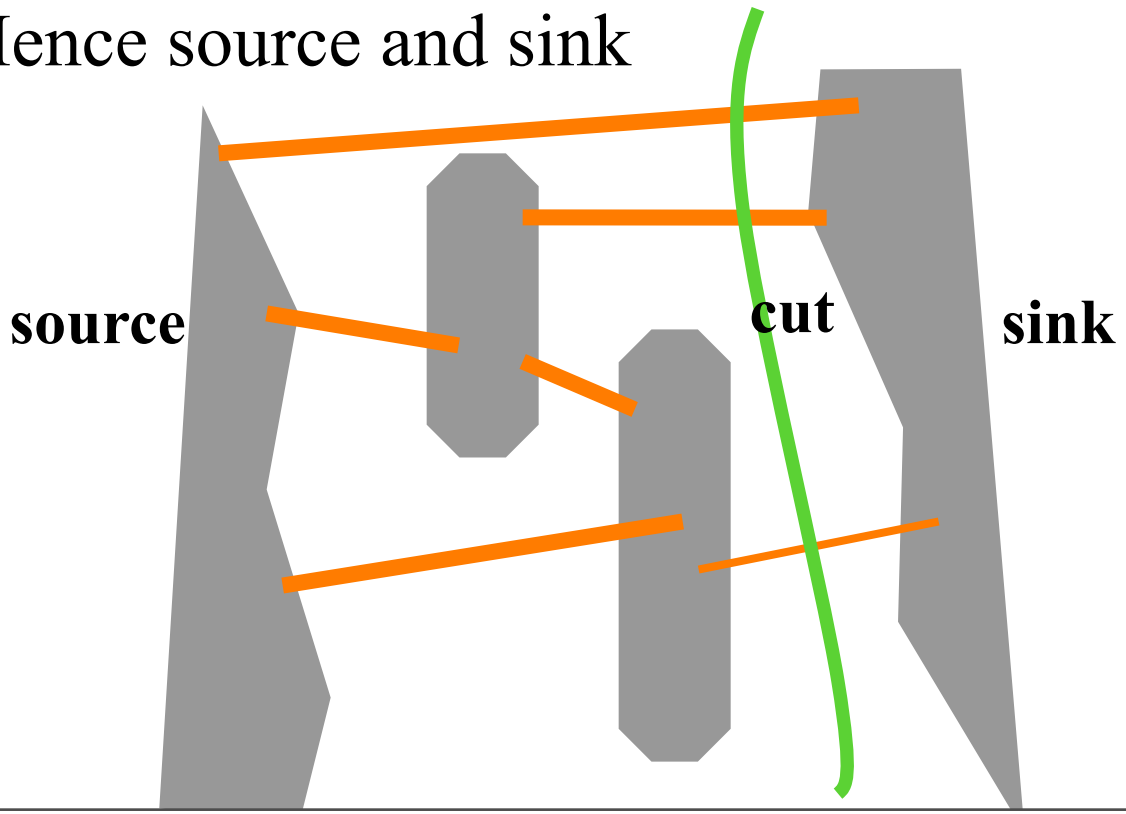


Questions?

- **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**

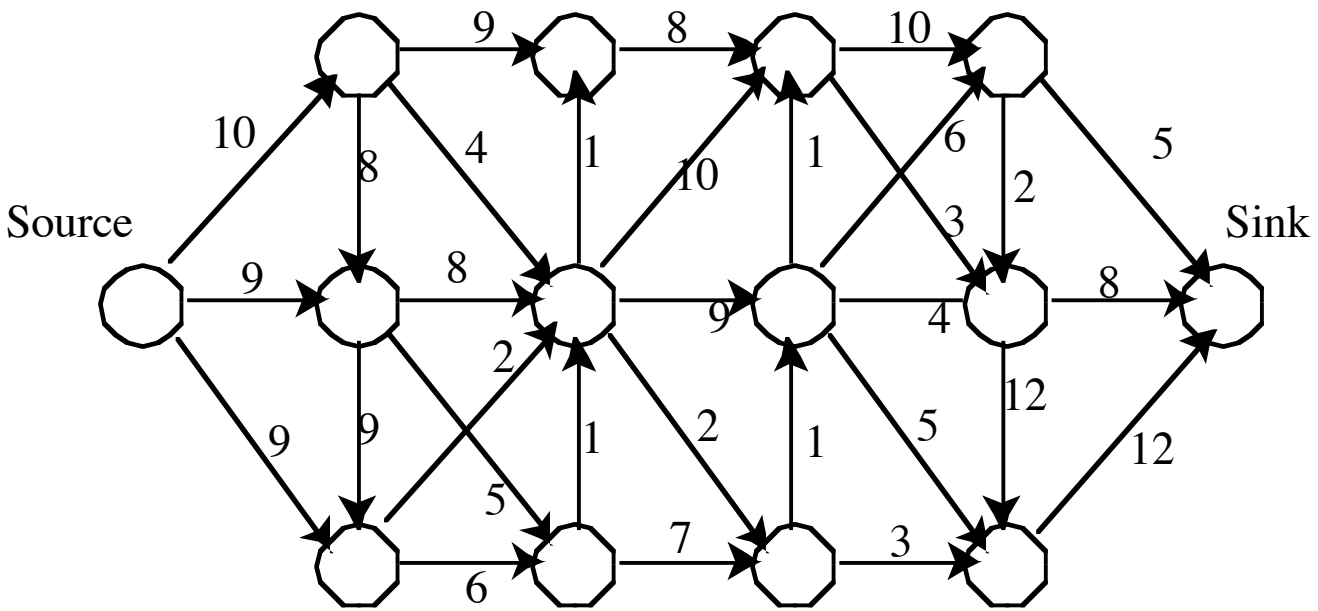
Min cut

- **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



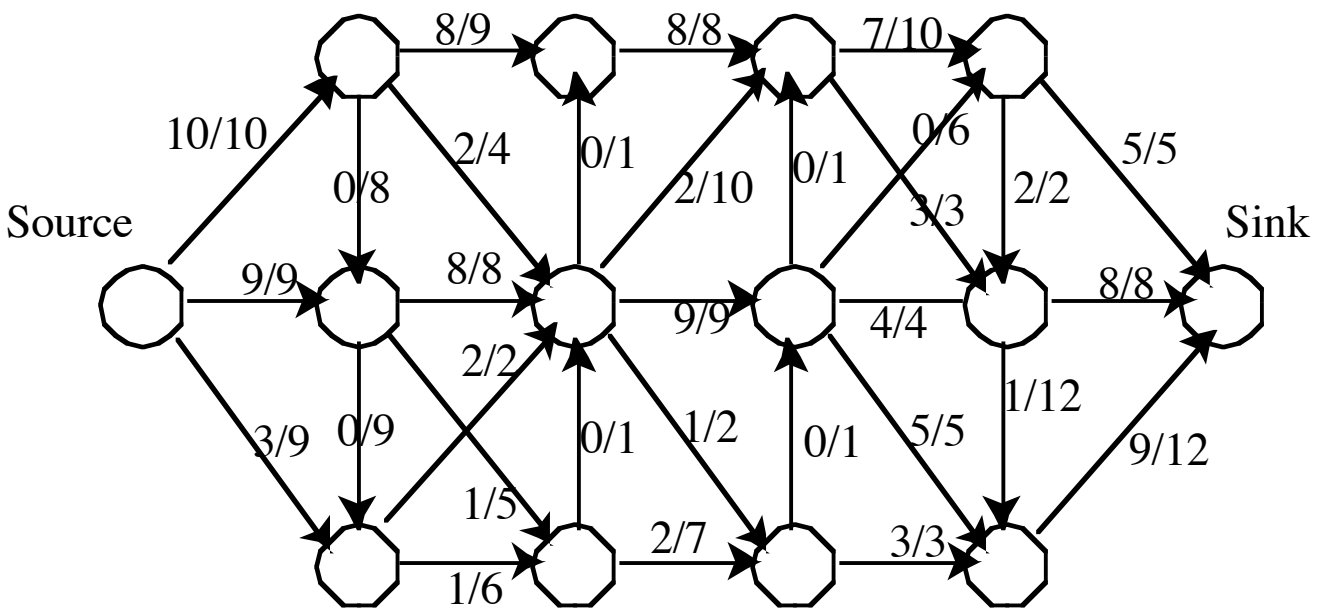
Max flow

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



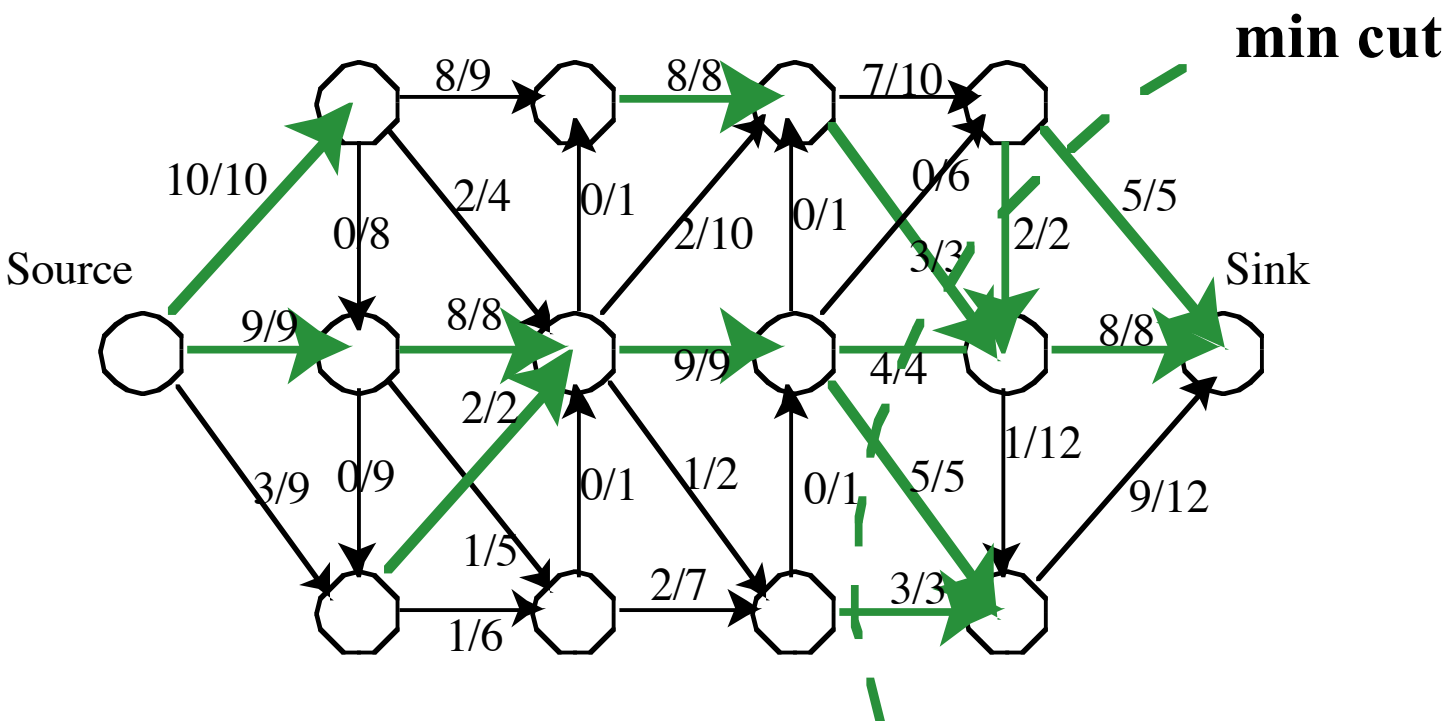
Max flow

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



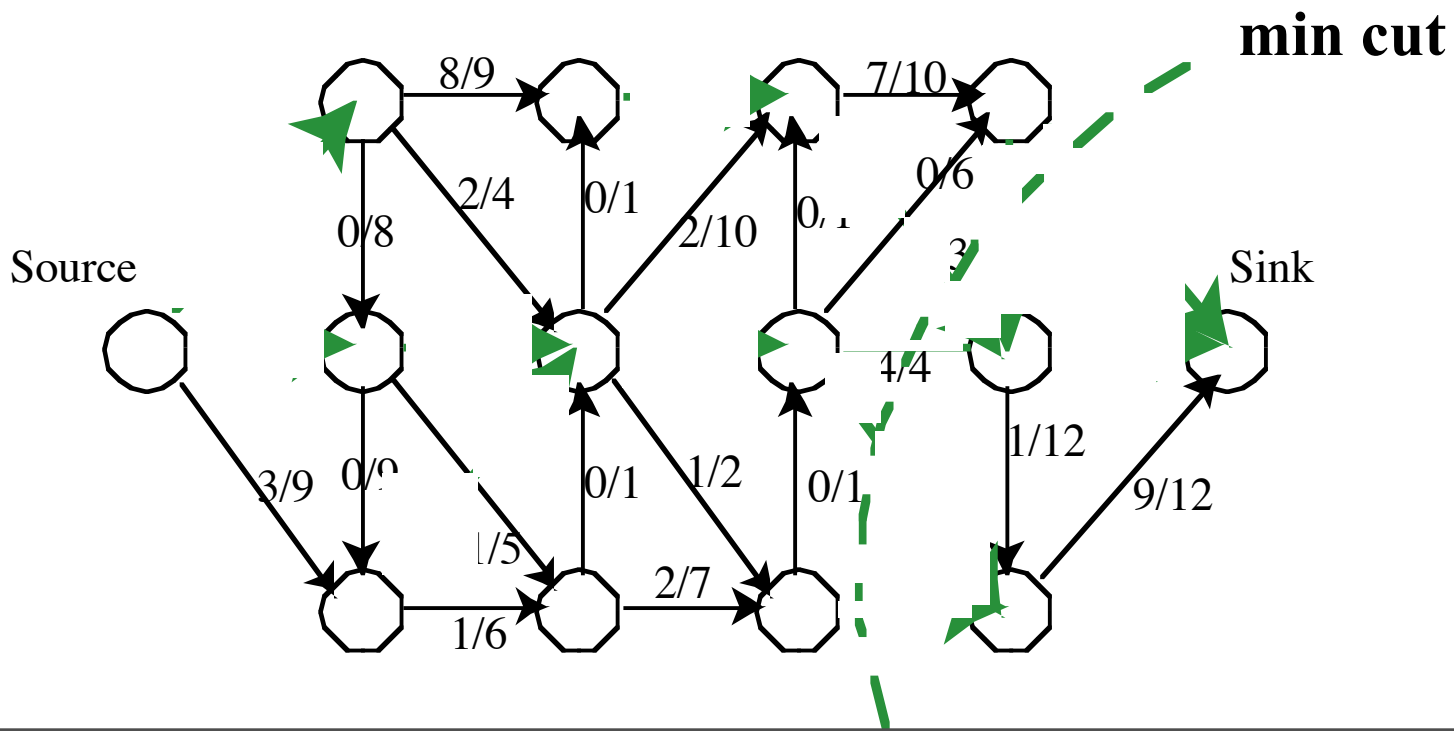
Max flow

- **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

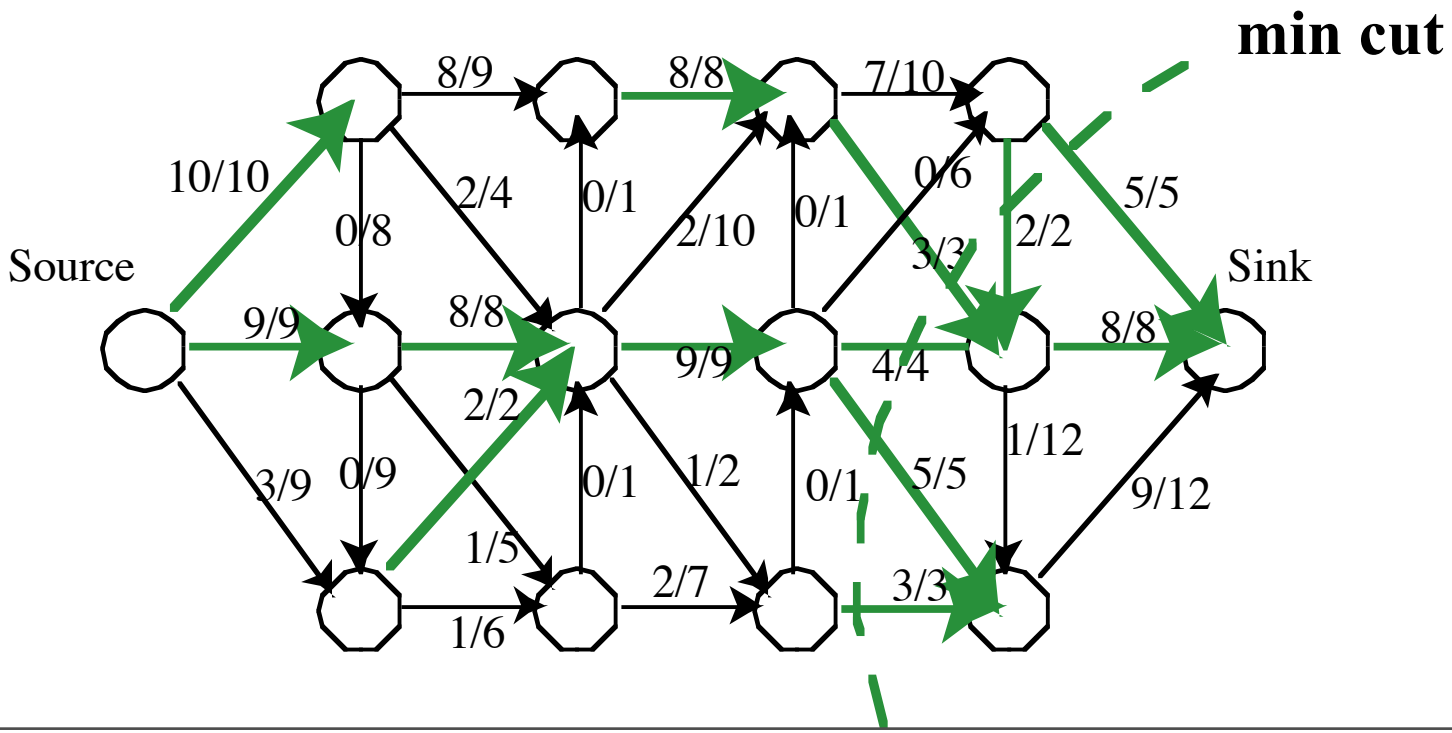
- **What is the max flow from source to sink?**
- **Look at residual graph**
  - remove saturated edges (gone here)
  - min cut is at boundary between 2 connected components



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

Shortest path anyone?

- **e.g. Dijkstra, A***

Efficiency concerns

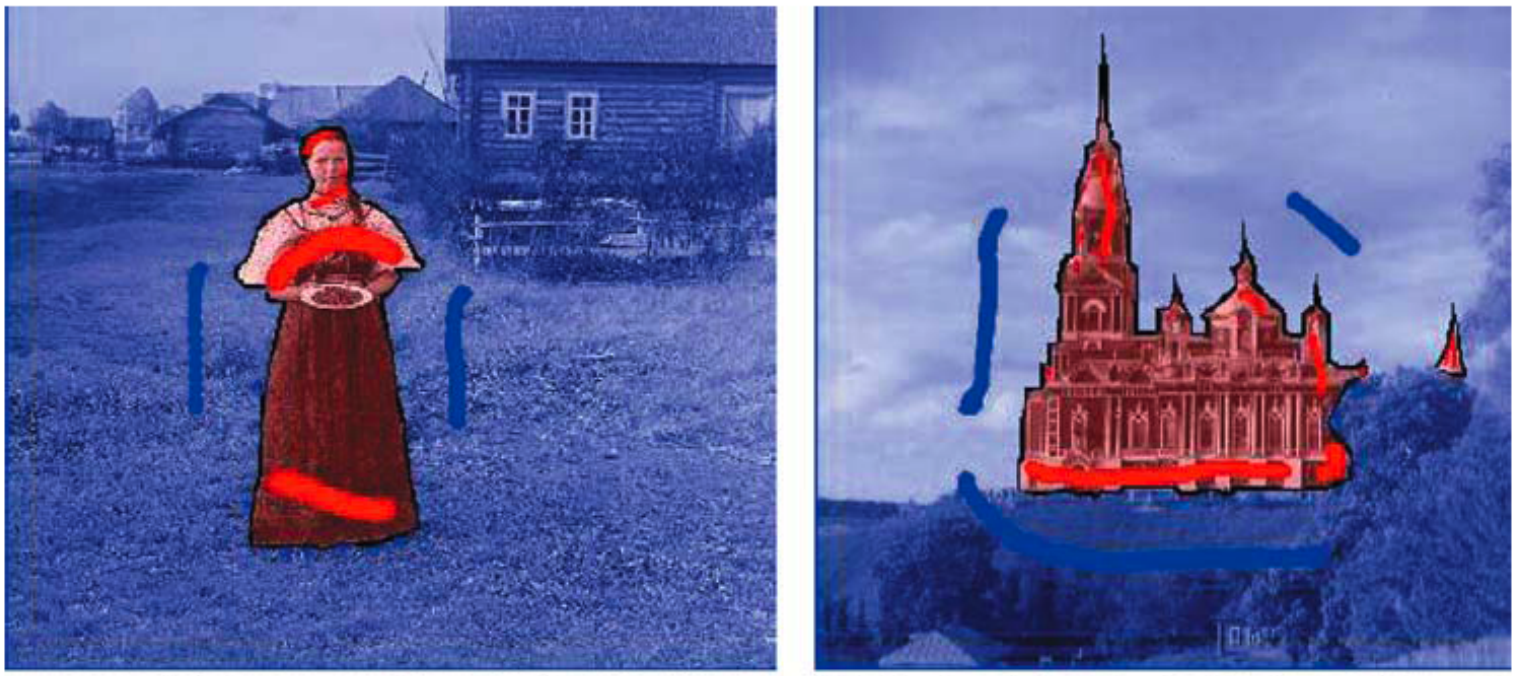
- **The search for a shortest path becomes prohibitive for the large graphs generated by images**
- **For practical vision/image applications, better (yet related) approaches exist**

  An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. Yuri Boykov, Vladimir Kolmogorov In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 9, Sept. 2004. http://www.csd.uwo.ca/faculty/yuri/Abstracts/pami04-abs.html

- **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**
- **In International Journal of Computer Vision (IJCV), vol. 70, no. 2, pp. 109-131, 2006 (accepted in 2004).**

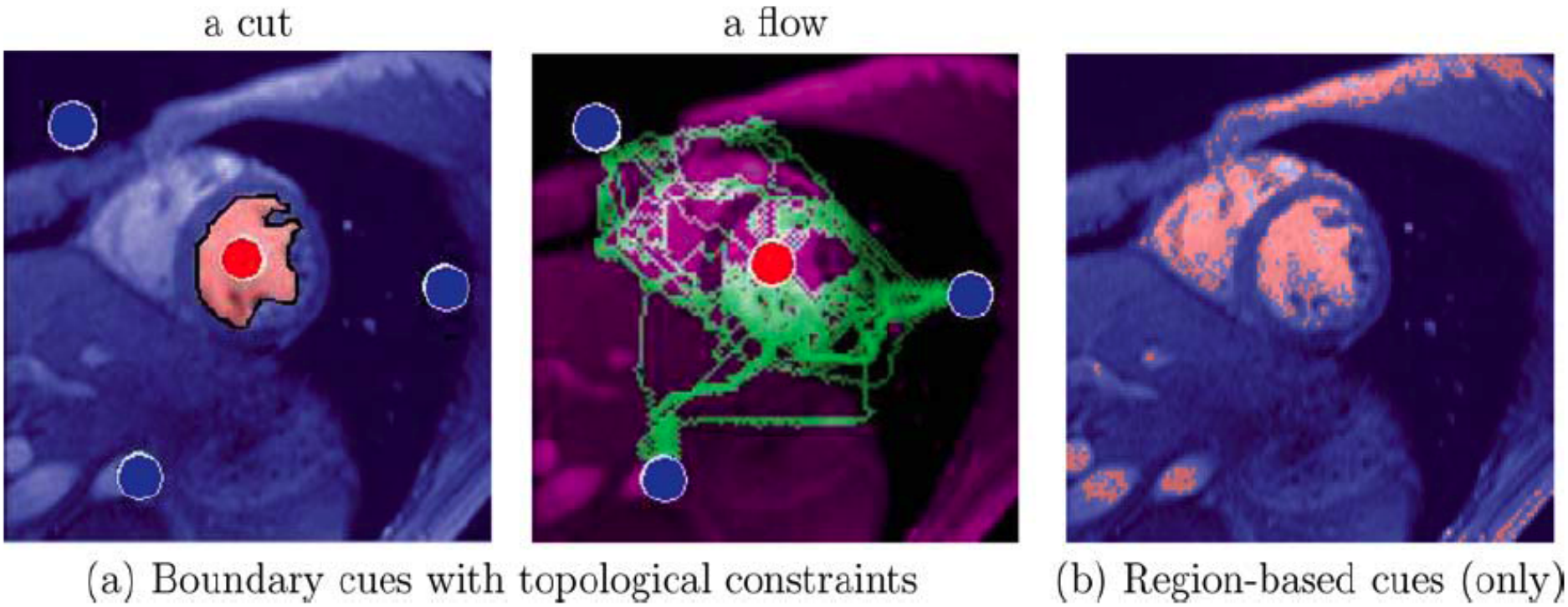


(a) A woman from a village

(b) A church in Mozhaisk (near Moscow)

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 × 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**

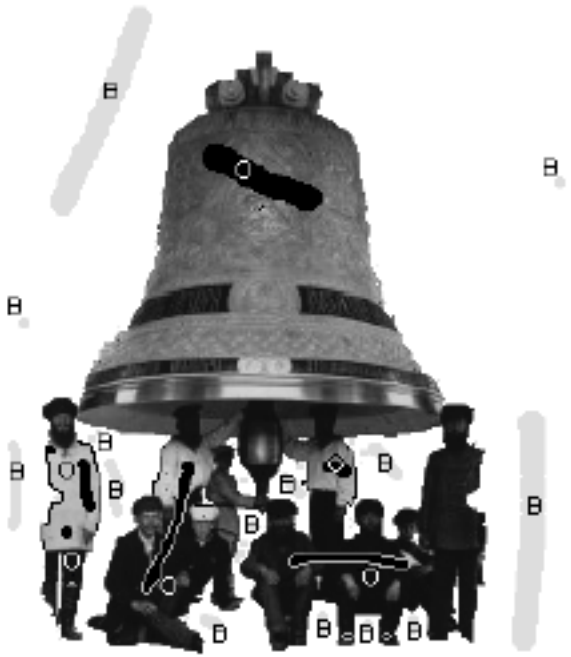


(a) Boundary cues with topological constraints (b) Region-based cues (only)

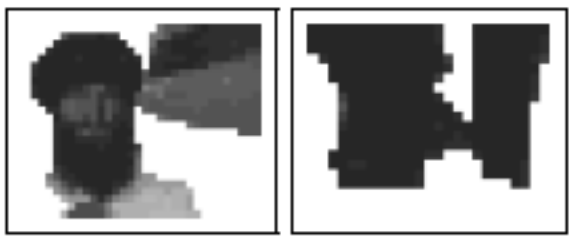
From Yuri Boykov, Gareth Funka-Lea

Data (regional) term

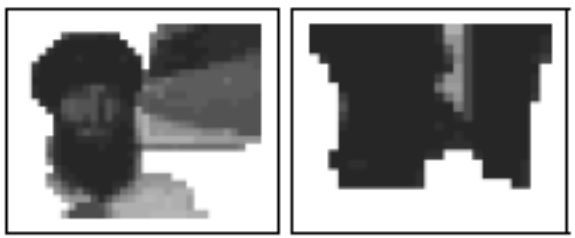
(a) Original B&W photo



(b) Segmentation results



(c) Details of segmentation with regional term



(d) Details of segmentation without regional term

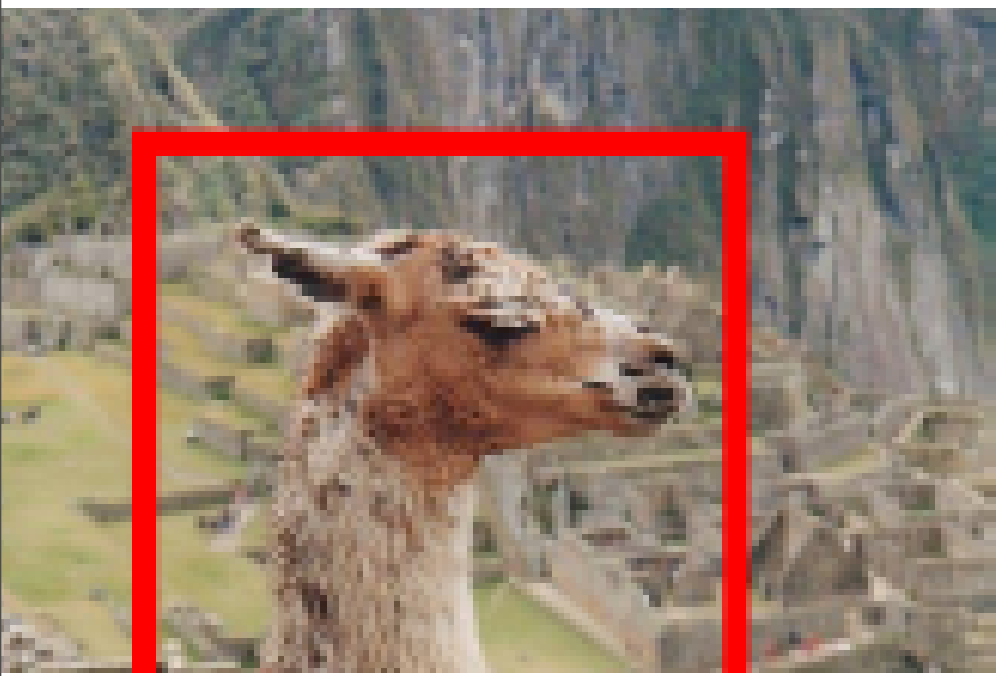
Questions?

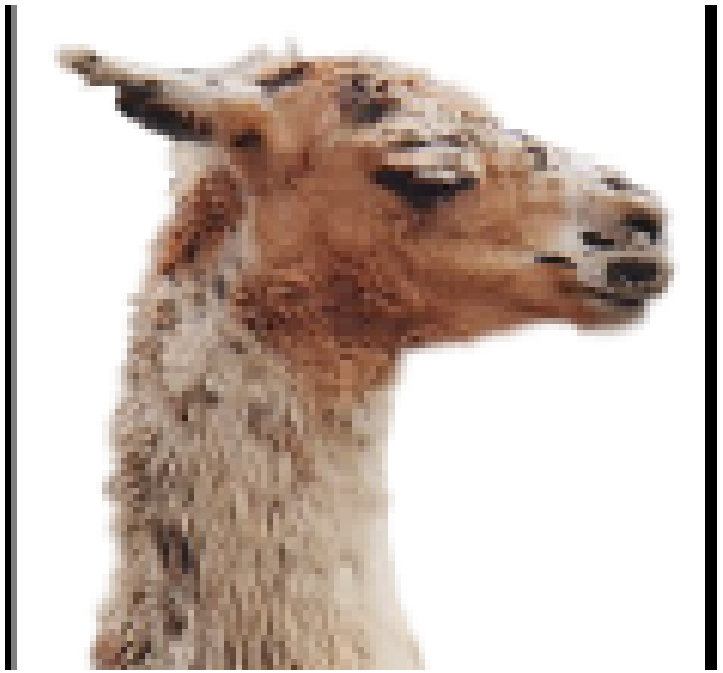
Grabcut

- **Rother et al. 2004**

- **Less user input: only rectangle**
- **Handle color**
- **Extract matte as post-process**

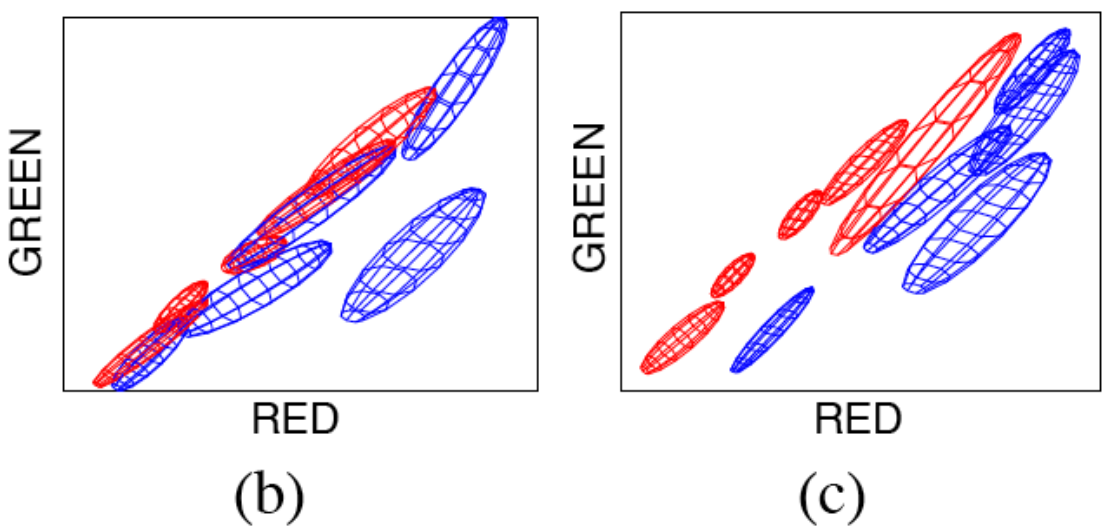
Figure 1: **Three examples of GrabCut .** The user drags a rectangle loosely around an object. The object is then extracted automatically.





Color data term

- **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

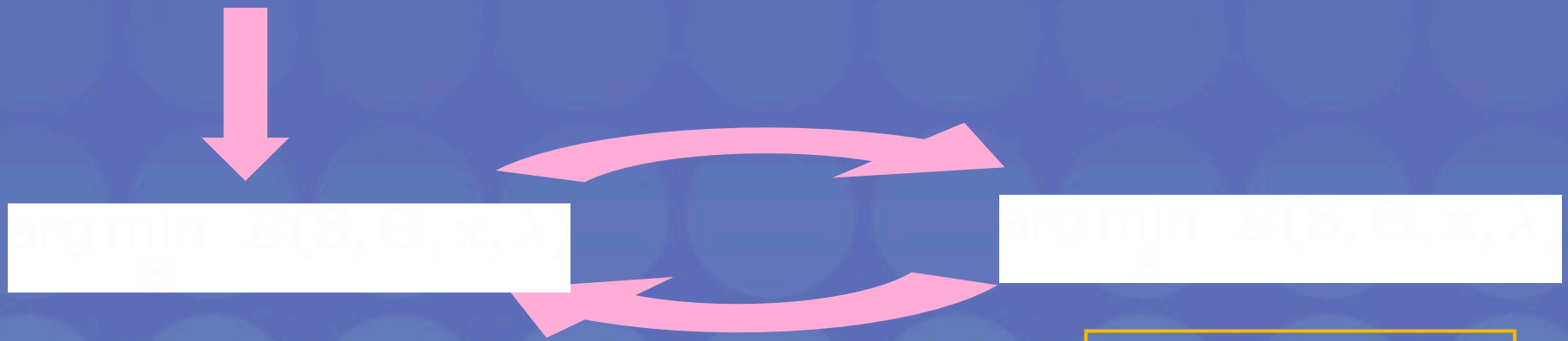
```
Take K random samples as seed centers
Iterate:
    For each sample
        Assign to closest cluster
    For each cluster
        Center = mean of samples in cluster
```

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

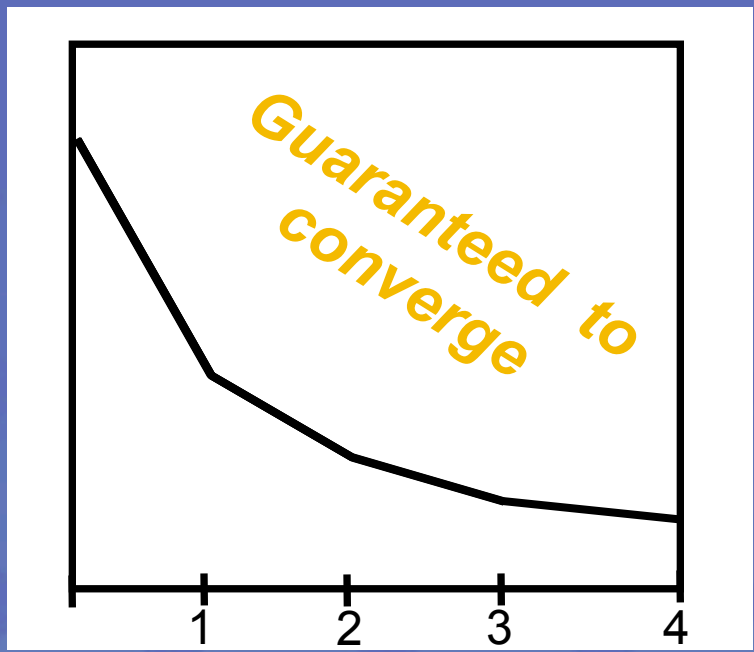
User Initialisation



K-means for learning colour distributions

Graph cuts to infer the segmentation

Iterated Graph Cuts



Result

Energy after each Iteration

Border matting

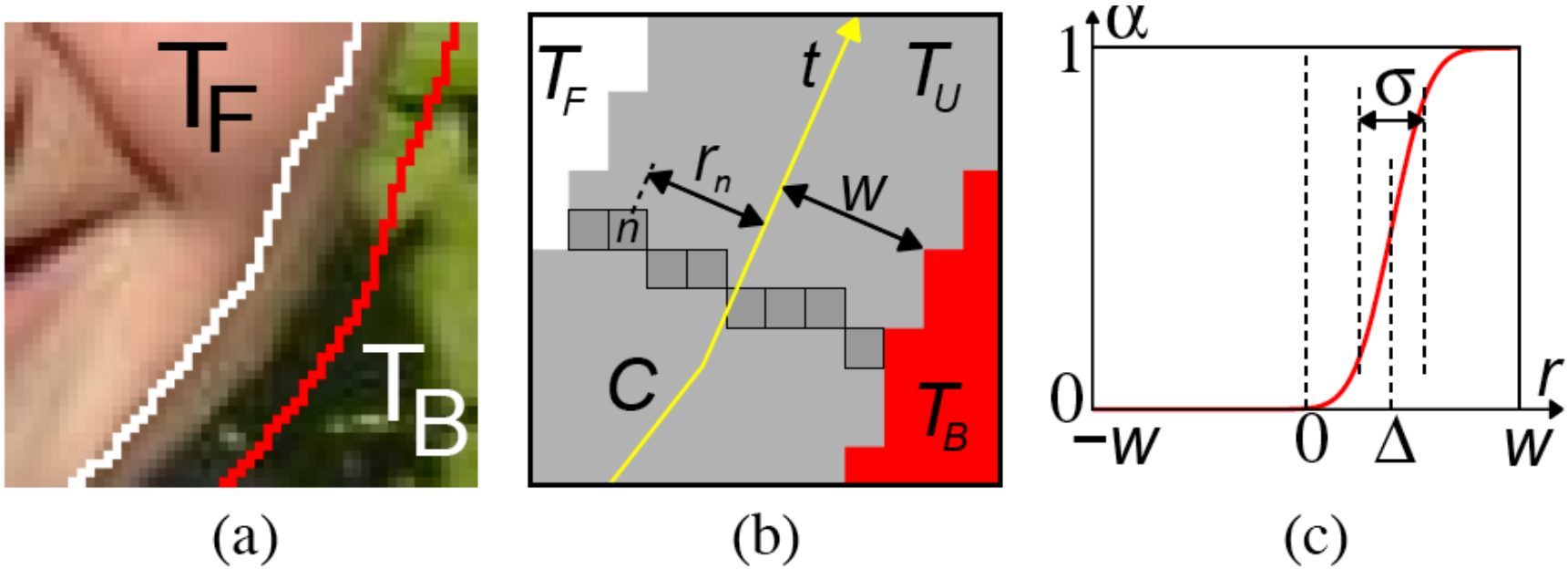


Figure 6: **Border matting.** (a) Original image with trimap overlaid. (b) Notation for contour parameterisation and distance map. Contour C (yellow) is obtained from hard segmentation. Each pixel in T_U is assigned values (integer) of contour parameter t and distance r_n from C. Pixels shown share the same value of t. (c) Soft step-function for α-profile g, with centre Δ and width σ.

Results

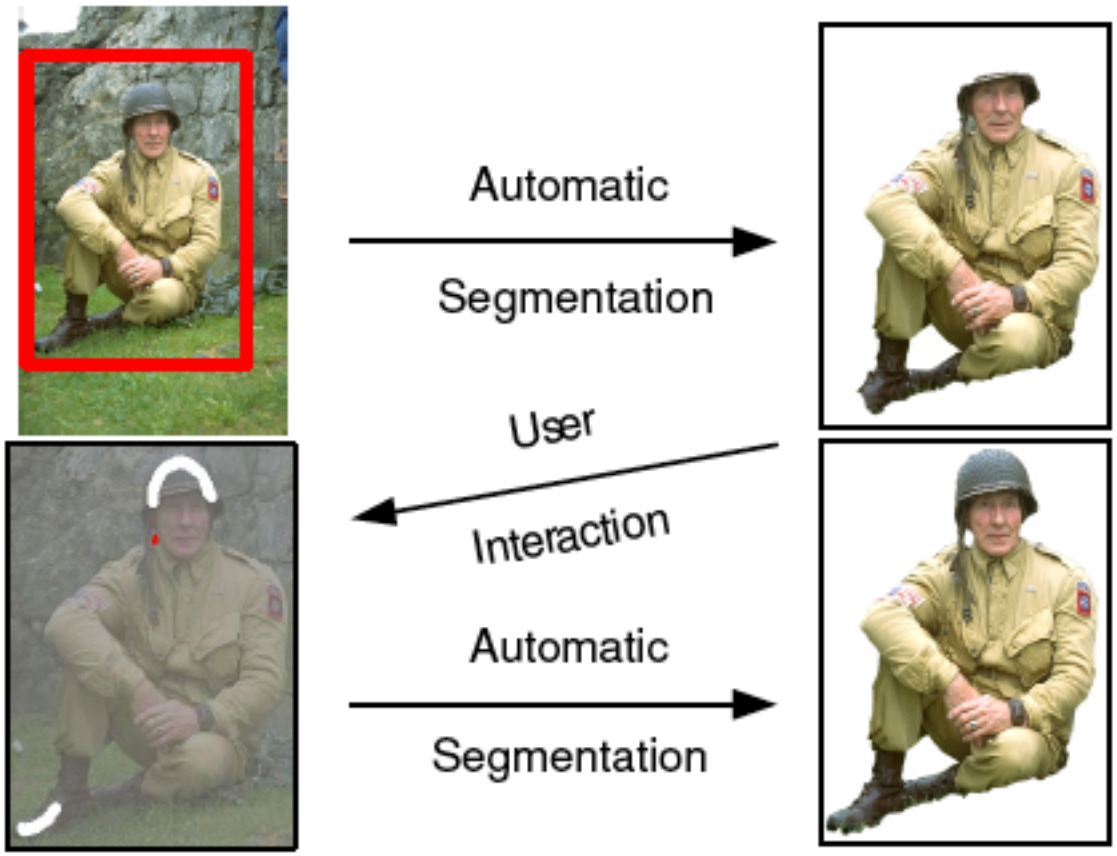
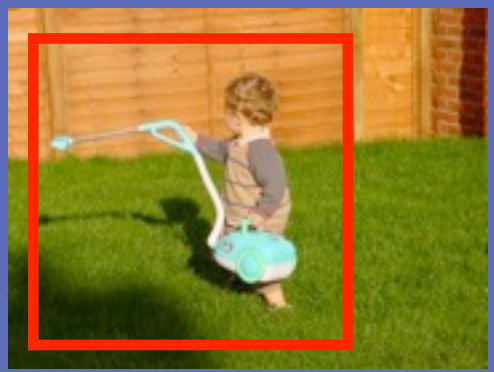
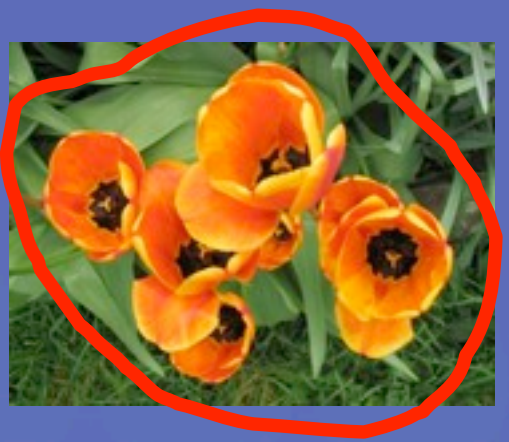


Figure 5: **User editing.** After the initial user interaction and segmentation (top row), further user edits (fig. 3) are necessary. Marking roughly with a foreground brush (white) and a background brush (red) is sufficient to obtain the desired result (bottom row).

Moderately straightforward examples

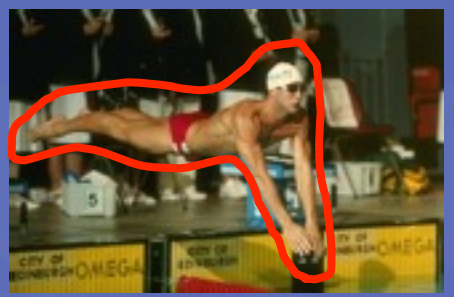





… GrabCut completes automatically

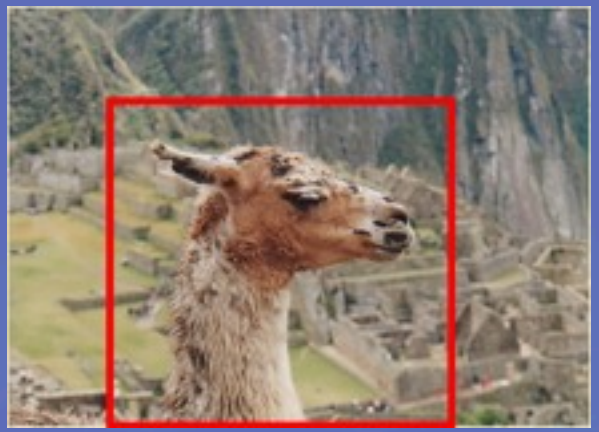
Difficult Examples

SIGGRAPH2004

	Camouflage & Low Contrast	Fine structure	No telepathy
Initial Rectangle			
Initial Result			

Comparison

SIGGRAPH2004

	Boykov and Jolly (2001)	GrabCut
User Input		
Result		
	Error Rate: 0.72%	Error Rate: 0.72%

Refs

- http://www.csd.uwo.ca/faculty/yuri/Abstracts/eccv06-tutorial.html
- **Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images.**
  Yuri Boykov and Marie-Pierre Jolly.
  In International Conference on Computer Vision, (ICCV), vol. I, 2001.
  http://www.csd.uwo.ca/~yuri/Abstracts/iccv01-abs.html
- http://www.cse.yorku.ca/~aaw/Wang/MaxFlowStart.htm
- http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm
- http://www.cc.gatech.edu/cpl/projects/graphcuttextures/
- **A Comparative Study of Energy Minimization Methods for Markov Random Fields. Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, Carsten Rother. ECCV 2006**
  www.cs.cornell.edu/~rdz/Papers/SZSVKATR.pdf