PSIVT2009
The 3rd Pacific-Rim Symposium on
Image and Video Technology

Tutorial 1

A Practical Introduction to Graph Cut

Hiroshi Ishikawa

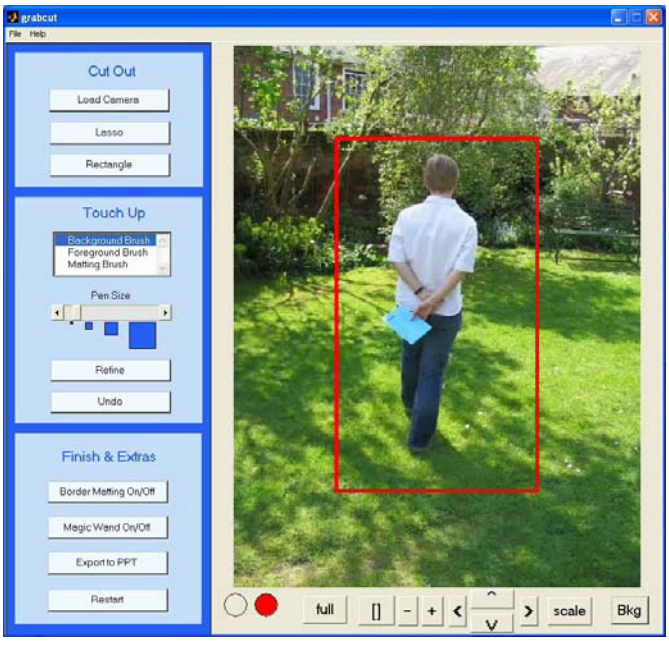
Department of Information and Biological Sciences
Nagoya City University

Contents

“Graph Cut”

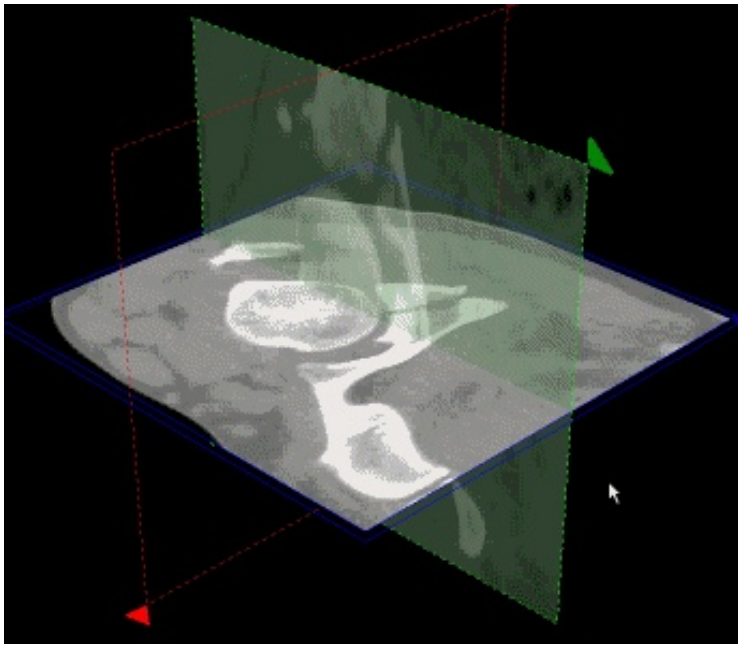
- A.K.A. s-t mincut
- An energy minimization method
  - Express tradeoffs by an energy
- Application areas
  - Image restoration
  - Stereo
  - Segmentation
  - Motion analysis
  - Texture synthesis
  - Photo montage

Interactive segmentation



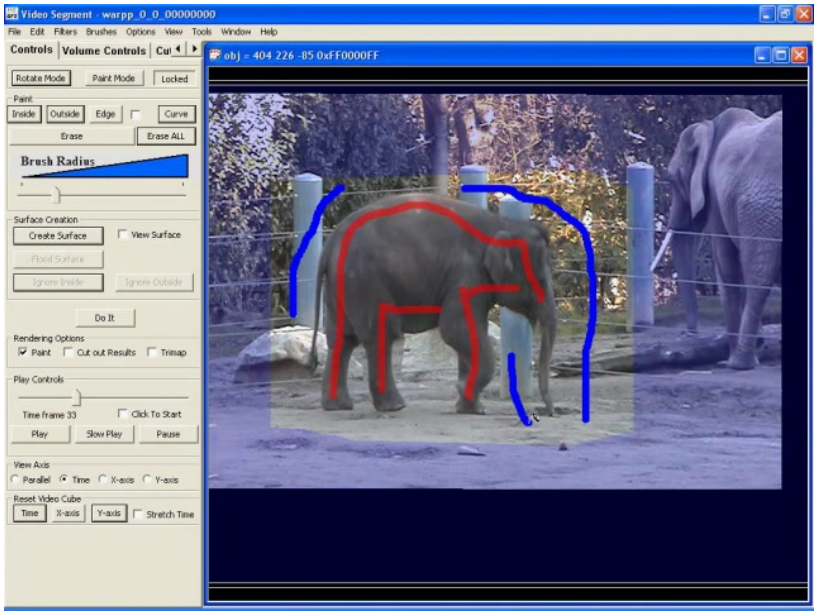
Rother et al.
SIGGRAPH2004

Interactive segmentation



Boykov&Jolly
ICCV2001

Interactive segmentation

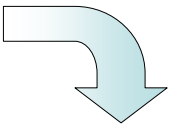


Wang et al. SIGGRAPH2005

Texture synthesis

Kwatra et al. SIGGRAPH2003

Texture synthesis



Kwatra et al. SIGGRAPH2003

Interactive photomontage

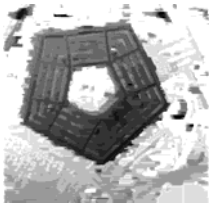

Agarwala et al. SIGGRAPH2004

Stereo

Left

Right

Elevation map



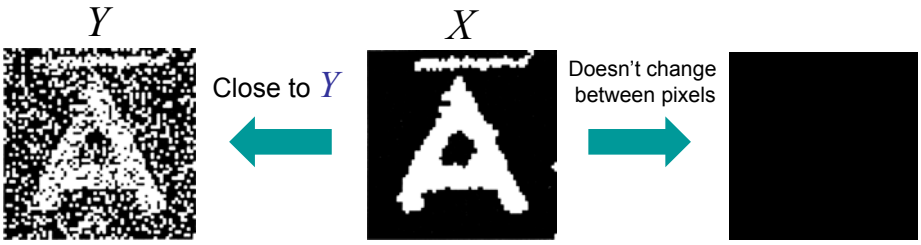
History

- Probablistic methods (SA, ICM,…) have been used for energy minimization
- OR has used graph cut for ever
- Was used for image processing in late 80's
- Introduced in vision in late 90's
  - Some theoretical results rediscovered (some were new)
- Also used in graphics in recent years

Contents

Example: Binary image restoration

- Denoising
  - Only the noisy image Y is given
- On what basis do we know what is noise?
  - We assume that the original image does not change too much between pixels (Smoothness assumption)
  - Close to the given image Y but not too rough
- Express the Tradeoffs by an energy $E(X)$



Example: Binary image restoration

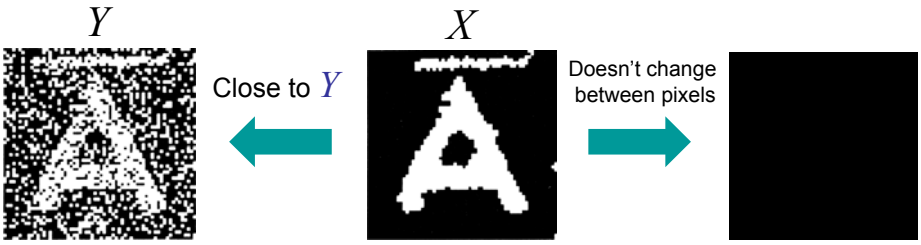
- Find X that minimizes the energy

Assigns X_v (= 0 or 1) to each pixel v

$$E(X) = \sum_{v \in V} \lambda \, | Y_v - X_v | + \sum_{(u,v) \in E} \kappa \, | X_u - X_v |$$

All pixels

Neighboring pixels



Example: Binary image restoration

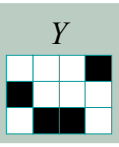
- Find X that minimizes the energy

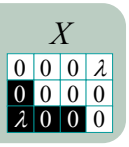
Assigns X_v (= 0 or 1) to each pixel v

$$E(X) = \sum_{v \in V} \lambda \, | Y_v - X_v | + \sum_{(u,v) \in E} \kappa \, | X_u - X_v |$$

All pixels

0 if $Y_v = X_v$
λ if $Y_v \neq X_v$





Close to Y

Doesn't change between pixels

Example: Binary image restoration

- Find X that minimizes the energy

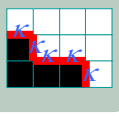
Assigns X_v (= 0 or 1) to each pixel v

$$E(X) = \sum_{v \in V} \lambda \, | Y_v - X_v | + \sum_{(u,v) \in E} \kappa \, | X_u - X_v |$$

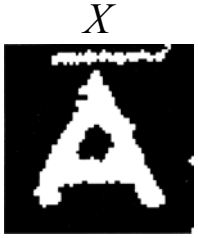
Neighboring pixels

0 if neighbors are the same; κ if not

X



Close to Y



Doesn't change between pixels

Energy minimization

- In general, consider an energy of the form:

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

Data term Smoothing term

where V is a set of sites

E is a set of neighboring pairs of sites

X assigns a label to each site in V

- First order Markov Random Field (MRF)
- Problem: Find the X that minimizes $E(X)$

Energy minimization

- Problem: Find the X that minimizes $E(X)$
- The number of possible Xs is vast
  - Assignments of a label to each site in V
  - If V is 64×64 and labels are binary, 2^{4096} ($>10^{1233}$) ways
- The general problem is NP-Hard (Discovery of less-than-exponential-time algorithm unlikely)
- Traditional method: Monte Carlo
- In some cases, Graph Cut can minimize globally

Contents

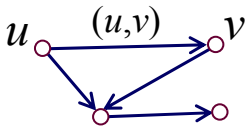
- Overview / Brief history
- Energy minimization
- Graphs and their minimum cuts
- Energy minimization via graph cuts
  - Global minimization
  - Approximation methods
- Energy design for graph cut

Graphs and their minimum cuts

- Directed graph

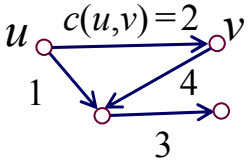
  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

  $\mathcal{V}$: Finite set (Vertices) $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ (Edges)



- "Weight" on each edge

  $c: \mathcal{E} \rightarrow \mathbf{R}$



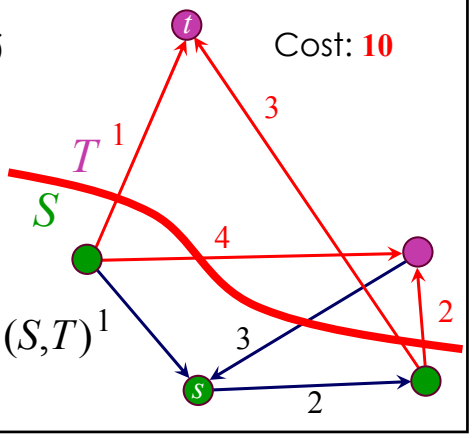
Graphs and their minimum cuts

- Take two vertices s, t of a weighted directed graph $\mathcal{G}$
- A cut of $\mathcal{G}$ w.r.t s and t is a partition of the vertices into two groups (S,T) such that

$$\mathcal{V} = S \cup T, \quad S \cap T = \varnothing$$

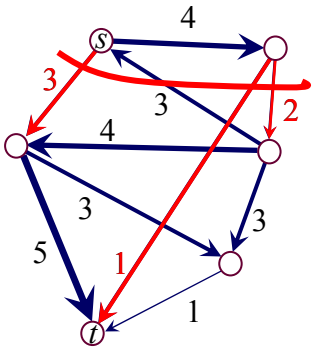
$$s \in S, \quad t \in T$$

- Total sum of the weights of the edges going from S to T is called the cost of (S,T)



Minimum cut

- Among the cuts of $\mathcal{G}$ with respect to s *and* t, ones with the smallest cost are called the minimum cuts
- Mincut = Maxflow
  - Ford & Fulkerson Theorem (1956)
  - Consider water flow from s to t, supposing the weight as the capacity of pipes
  - Minimum cuts are given by the saturated edges for maximum flows



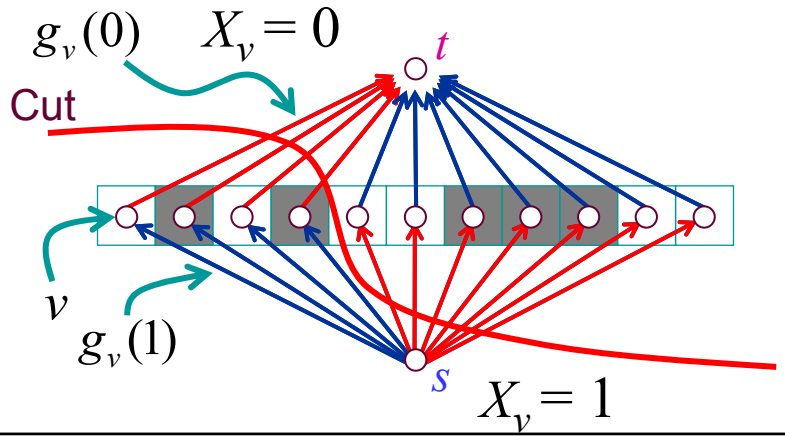
Minimum cut

- Among the cuts of G with respect to s *and* t, ones with the smallest cost are called the minimum cuts
- Mincut = Maxflow
- When the weights are all nonnegative, Maxflow can be found in polynomial time
- The number of possible cuts ~ $2^{\text{number of vertices}}$

Contents

Graph-cut minimization (binary)

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} \kappa |X_u - X_v|$$



Graph-cut minimization (binary)

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} \kappa |X_u - X_v|$$

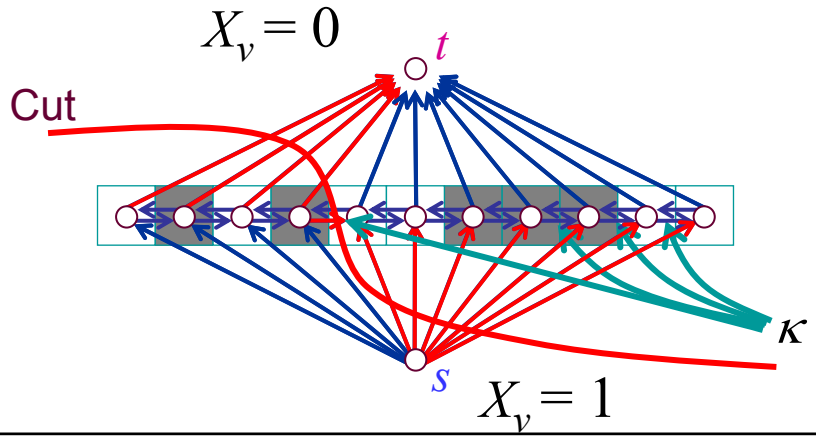
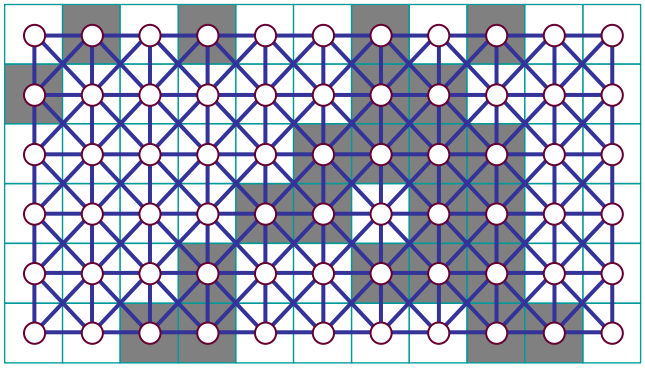
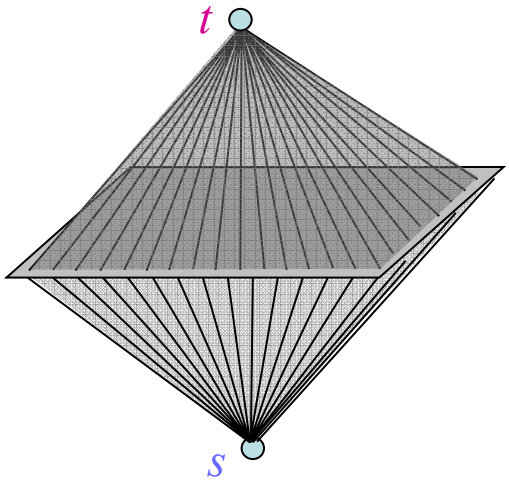


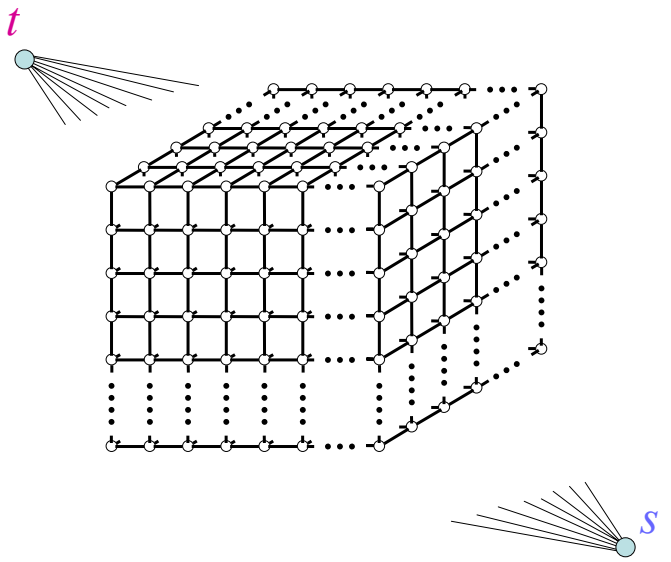
Image plane graph



2D case

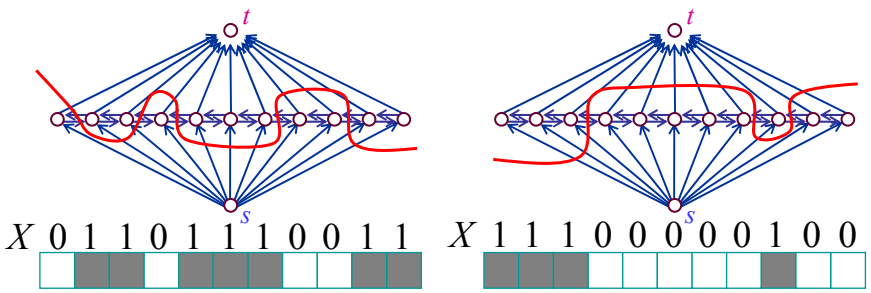


3D case



Graph-cut minimization (binary)

- One to one correspondence between X and cut



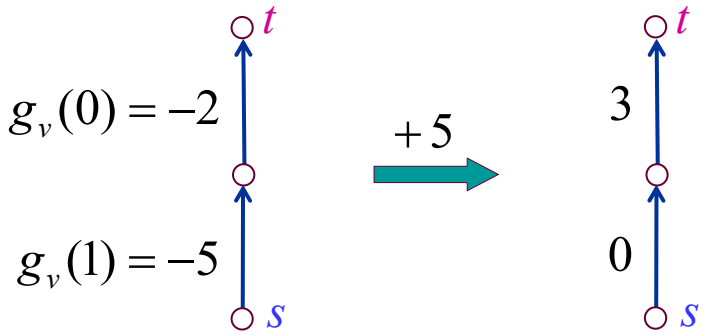
- Energy = Cut cost
- Find minimum cut to minimize energy
- The weights must be all nonnegative

Graph-cut minimization (binary)

- The weights must be all nonnegative

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

- $g_v(x)$ can be arbitrary



Graph-cut minimization (binary)

- The weights must be all nonnegative

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

- $g_v(x)$ can be arbitrary
- $h_{uv}(x, y)$ must satisfy

Submodularity condition

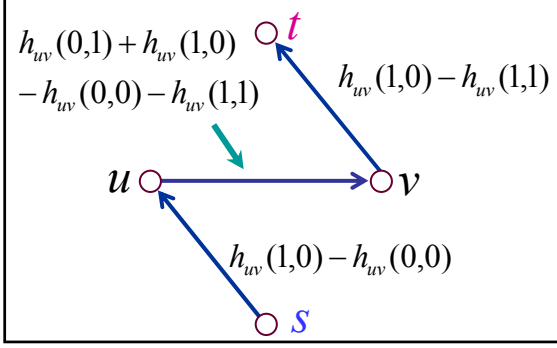
$$h_{uv}(0,0) + h_{uv}(1,1) \leq h_{uv}(0,1) + h_{uv}(1,0)$$

Graph-cut minimization (binary)

- The weights must be all nonnegative

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

$$h_{uv}(0,0) + h_{uv}(1,1) \le h_{uv}(0,1) + h_{uv}(1,0)$$



u	v	
0	0	$h_{uv}(1,0) - h_{uv}(1,1)$
0	1	$h_{uv}(0,1) + h_{uv}(1,0) - h_{uv}(0,0) - h_{uv}(1,1)$
1	0	$2h_{uv}(1,0) - h_{uv}(0,0) - h_{uv}(1,1)$
1	1	$h_{uv}(1,0) - h_{uv}(0,0)$

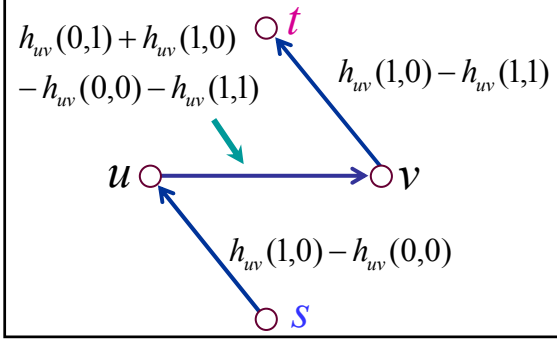
Graph-cut minimization (binary)

- The weights must be all nonnegative

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

$$h_{uv}(0,0) + h_{uv}(1,1) \le h_{uv}(0,1) + h_{uv}(1,0)$$

Add same value for all 4 cases



u	v	
0	0	$h_{uv}(0,0)$
0	1	$h_{uv}(0,1)$
1	0	$h_{uv}(1,0)$
1	1	$h_{uv}(1,1)$

Graph-cut minimization (binary)

- The weights must be all nonnegative

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

- $g_v(x)$ can be arbitrary
- $h_{uv}(x, y)$ must satisfy

  Submodularity condition

$$h_{uv}(0,0) + h_{uv}(1,1) \leq h_{uv}(0,1) + h_{uv}(1,0)$$

- Long known in OR

Graph-cut minimization (multi-value)

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

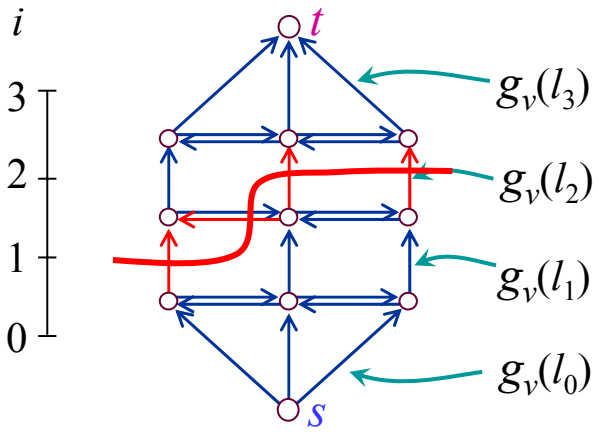
- When there are >2 labels

- If labels have an order : $L = \{l_0, l_1, \cdots, l_k\}$

  Globally minimizeable $\Leftrightarrow$ $h_{uv}(l_i, l_j)$ is a convex function of $i - j$

Graph-cut minimization (multi-value)

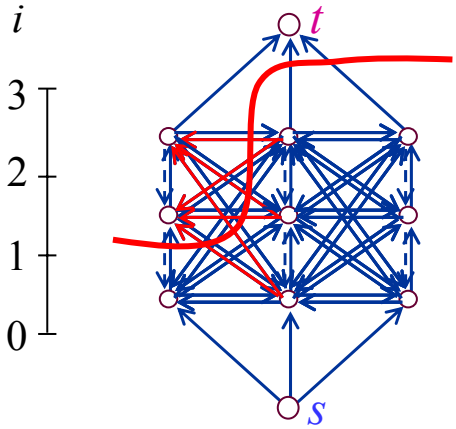
$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$



$$h_{uv}(l_i, l_j) = \kappa |i - j|$$

Graph-cut minimization (multi-value)

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$



$$\tilde{h}_{uv}(i-j+1) - 2\tilde{h}_{uv}(i-j) + \tilde{h}_{uv}(i-j-1)$$

In general: $h_{uv}(l_i, l_j) = \tilde{h}_{uv}(i-j)$, $\tilde{h}_{uv}$:convex

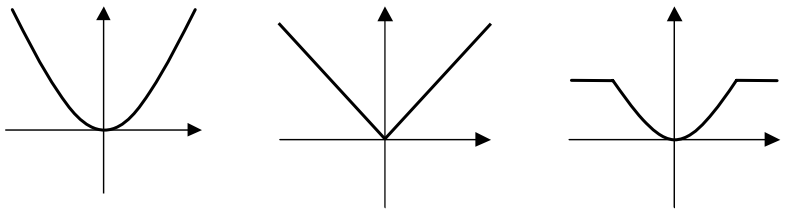
Contents

- Overview / Brief history
- Energy minimization
- Graphs and their minimum cuts
- Energy minimization via graph cuts
  - Global minimization
  - Approximation methods
- Energy design for graph cut

Graph-cut minimization (multi-value)

- The smoothing term must be convex

good good bad



- Approximation algorithms
  - α-β swap / α expansion (Boykov, Veksler, Zabih)
  - Guaranteed to get within the factor $2c$ of global minima

$$c = \max_{u,v \in V} \left(\frac{\max_{X_u \neq X_v} h_{uv}(X_u, X_v)}{\min_{X_u \neq X_v} h_{uv}(X_u, X_v)} \right)$$

  - The Potts model (discriminate only same or not): $c = 1$

Graph-cut minimization (multi-value)

- Approximation by iteration
- Decide whether or not to move the assignment, all at once for all pixels, by using binary graph cut
- α-β swap
  - Allow swap only when current label is α or β
  - Applicable when the following holds for all $\alpha,\beta \in L$

$$h_{uv}(\alpha,\alpha)+h_{uv}(\beta,\beta) \le h_{uv}(\alpha,\beta)+h_{uv}(\beta,\alpha)$$

- α expansion
  - Allow only change to α
  - Applicable when the following holds for all $\alpha,\beta,\gamma \in L$

$$h_{uv}(\alpha,\alpha)+h_{uv}(\beta,\gamma) \le h_{uv}(\alpha,\gamma)+h_{uv}(\beta,\alpha)$$

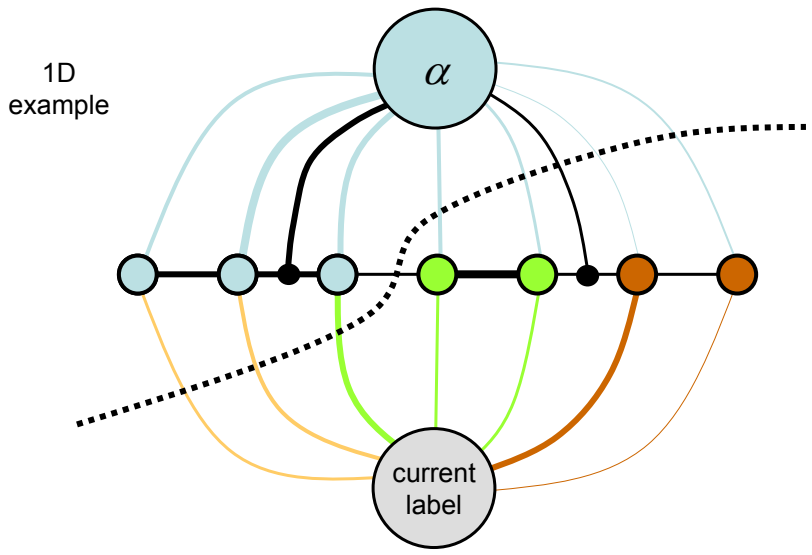
α expansion moves

In each α expansion a given label "α" grabs space from other labels

For each move we choose expansion that gives the largest decrease in the energy: **binary optimization problem**

Courtesy Yuri Boykov

Optimal α expansion move



Courtesy Yuri Boykov

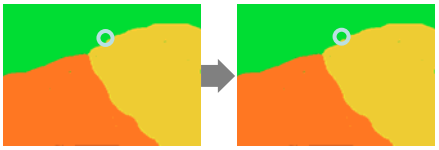
α expansion algorithm

1. Start with any initial solution
2. For each label α in any (e.g. random) order
    1. *Compute optimal α 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*

Courtesy Yuri Boykov

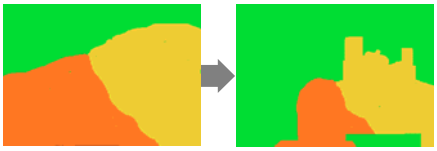
α expansion algorithm vs. standard discrete energy minimization techniques

Single "one-pixel" move
(Simulated Annealing, ICM,…)



- Only one pixel can change its label at a time
- Finding an optimal move is computationally trivial

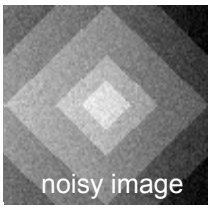
Single α expansion move



- Large number of pixels can change their labels simultaneously
- Finding an optimal move is computationally intensive

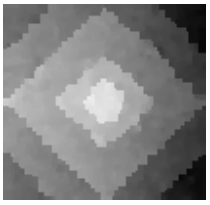
Courtesy Yuri Boykov

α-expansion move vs."standard" moves



Potts energy minimization

a local minimum
w.r.t. expansion moves



a local minimum
w.r.t. "one-pixel" moves

Courtesy Yuri Boykov

Contents

- Overview / Brief history
- Energy minimization
- Graphs and their minimum cuts
- Energy minimization via graph cuts
  - Global minimization
  - Approximation methods
- Energy design for graph cut

5 decisions for energy design

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

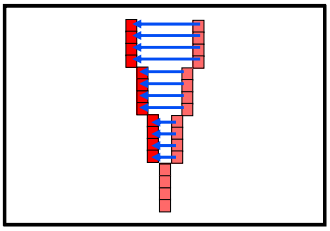
Example: Denoising

- Sites
  - Pixels
- Neighborhood structure
  - Pixel neighborhood structure
- Labels
  - Pixel colors

} The space of X : Assignmnets of a label to each site

- Data term: How to reflect the given data
  - Make X closer to the given (noisy) image
- Smoothing term : Desired propeties for X
  - Make neighboring pixels closer

Example 1: Stereo



Disparity

Boykov et al. PAMI 2001

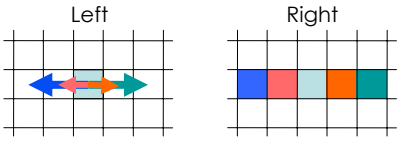
Simulated Annealing α-β swap α expansion

Example 1: Stereo

$$E(X) = \sum_{v \in V} | I_v - I_{v+X_v} | + \sum_{(u,v) \in E} \lambda_{uv} \, \text{Potts}(X_u, X_v)$$

- Sites
  - Pixels of one of the images
- Neighborhood structure
  - Pixel neighborhood structure
- Labels
  - Disparities
- Data term
  - Compare pixels displaced according to the disparity
- Smoothing term
  - Smooth the disparities across neighbors
  - But also want to allow discontinuities at object boundaries
  - Make λ_{uv} smaller where intensity changes abruptly

$$\text{Potts}(l, m) = \begin{cases} 0 & (l = m) \\ 1 & (l \neq m) \end{cases}$$



Example 2: Segmentation

$$E(X) = \sum_{v \in V} g_v(X_v) + \sum_{(u,v) \in E} h_{uv}(X_u, X_v)$$

- Sites
  - Pixels
- Neighborhood structure
  - Pixel neighborhood structure
- Labels
  - Foreground or background (0 or 1)
- Data term
  - Locally assesses from the color of the pixel whether it is more like the foreground or the background
- Smoothing term
  - Smooth the labels across neighbors

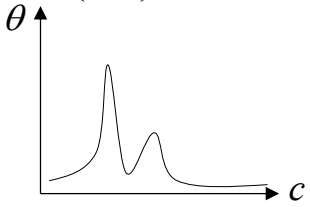
Example 2: Segmentation

- Data term: Creates histograms of the user-specified fore- and background sample pixels
- Evaluates how likely to be fore- or background

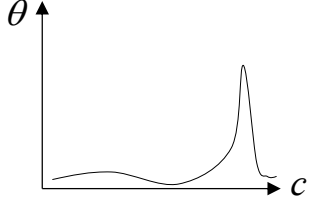
$$g_v(l) = -\log \theta(I(v), l) \quad l = 0,1$$

θ is normalized $\sum_c \theta(c,0) = \sum_c \theta(c,1) = 1$

$\theta(c,0)$: FG histogram



$\theta(c,1)$: BG



Example 2: Segmentation

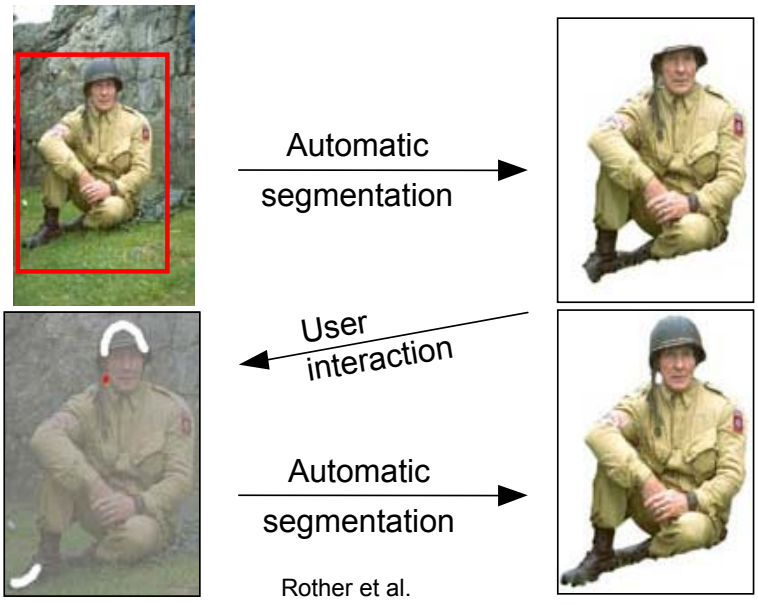
- Smoothing term: **Potts** (Penalty is smaller if the color changes more between neighbors)

$$h_{uv}(l,l') = \begin{cases} 0 & (l = l') \\ \dfrac{\lambda e^{-\kappa\{I(u)-I(v)\}^2}}{\mathrm{dist}(u,v)} & (l \neq l') \end{cases}$$

Does not depend on labels

- Meant to cut where there is a strong contrast

Example 2: Segmentation



Rother et al.
SIGGRAPH2004

Example 3: Photomontage

Agarwala et al. SIGGRAPH2004

Example 3: Photomontage

Agarwala et al. SIGGRAPH2004

Example 3: Photomontage

- Sites/Neighborhood
  - Pixel/Pixel neighborhood structure
- Labels
  - Which source picture $(1, 2, ..., k)$

- Data term
  - Where a source picture is specified, a constant penalty for other labels

$$g_v(l) = \begin{cases} 0 & (l = l') \\ M & (l \neq l') \end{cases}$$

  - 0 elsewhere



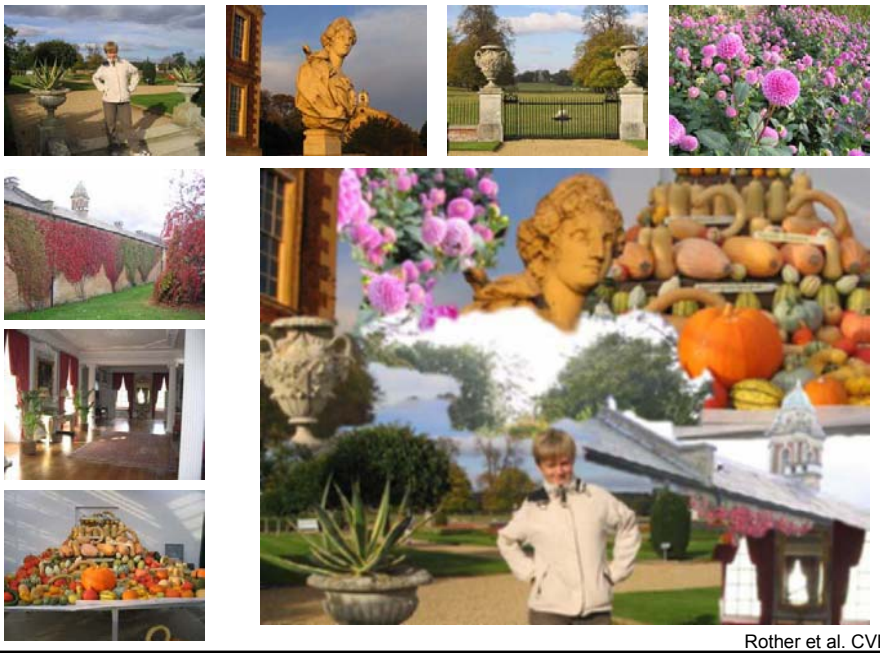
Example 3: Photomontage

- Smoothing term
  - Makes boundary invisible (Reverse of segmentation)

$$h_{uv}(l, m) = \text{dist}(I_l(u), I_m(u)) + \text{dist}(I_l(v), I_m(v))$$

  - Penalty is smaller if the color is closer
  - Other cues (e.g. color gradient) could be matched

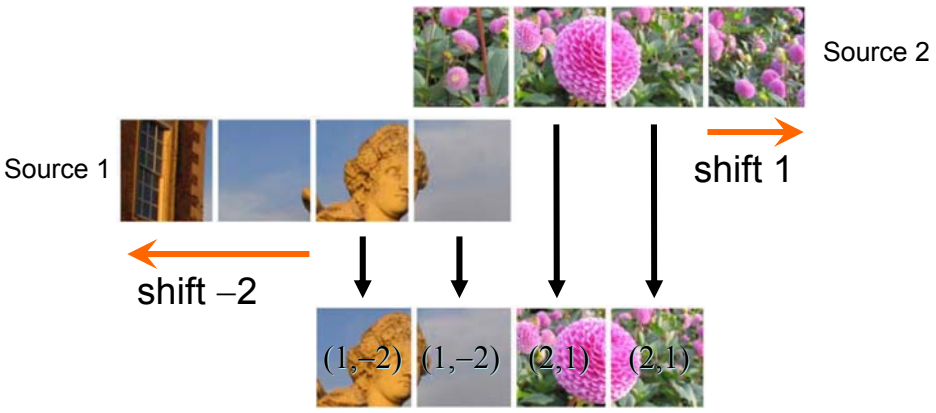

Example 4: Digital Tapestry



Rother et al. CVPR2005

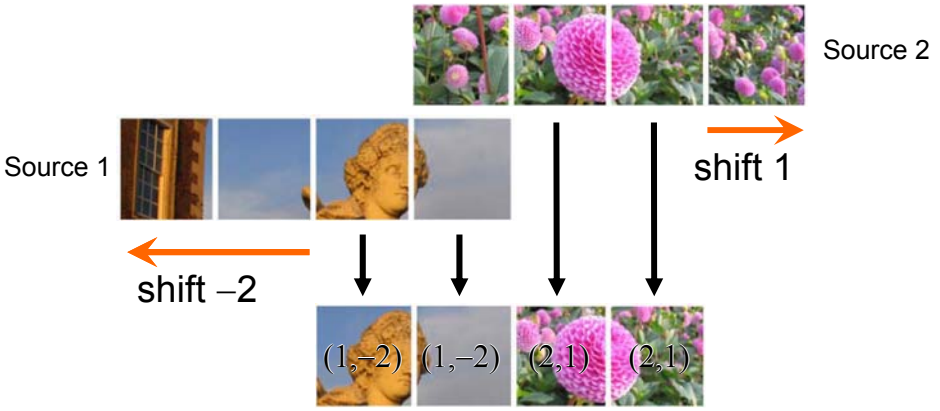
Example 4: Digital Tapestry

- Divide both tapestry and source into blocks
- Site: tapestry blocks
- Neighborhood: All pairs&block neighbors (later)
- Labels: Pairs of (source image, block shift)



Example 4: Digital Tapestry

- Data term: Block saliency (really the contrast)
  - Central blocks are considered more salient than peripherals
- Smoothing term:
  - Each source block used once (All sites are neighbors)
  - Neighboring block compatibility (Neighbor on tapestry)



This has been an introduction, but...

- Graph cut is an ongoing research area
- Recent topics include
  - α-β range moves (Veksler CVPR2007)

    Uses multi-label GC to minimize truncated convex potential
  - QPBO (Boros, Hammer, et al. RUTCOR Report RRR 10-2006; Kolmogorov&Rother CVPR2007)

    Minimizes even nonsubmodular energies where possible
  - Fusion moves (Lempitsky et al. ICCV2007)

    Fuses current and proposed configurations *à la* α expansion
  - Higher-order energies (Ramalingam et al. CVPR2008)

Conclusion

- Graph cut: energy minimization
- Energy form dictates applicable algorithm
  - Binary (Submodular → global optimization)
  - Multi label (Global optimization in some cases)
  - Multi label (Approximation algorithms)
- Some energy can be minimized globally
- Approximations applicable to more general cases are often better than classical methods, e.g., SA
- Code available at Kolmogorov's website:

  `http://www.adastral.ucl.ac.uk/~vladkolm/software.html`