Image segmentation – part 2

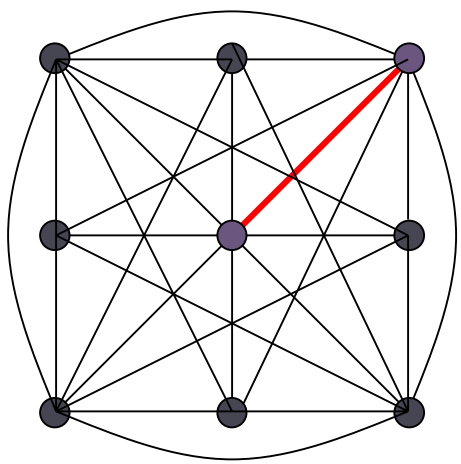
Lihi Zelnik-Manor, Computer Vision

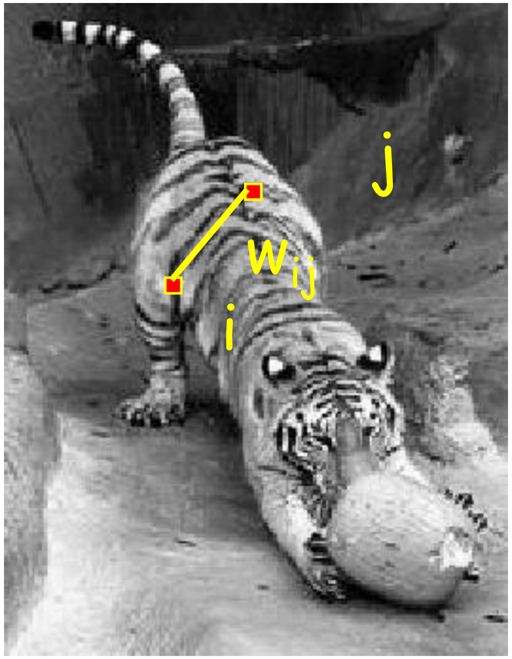
Today

- Graph theoretic segmentation
  - Normalized cuts
- Segmentation as energy minimization
  - Markov random fields

Images as graphs

- Node for every pixel
- Edge between every pair of pixels (or every pair of "sufficiently close" pixels)
- Each edge is weighted by the *affinity* or similarity of the two nodes

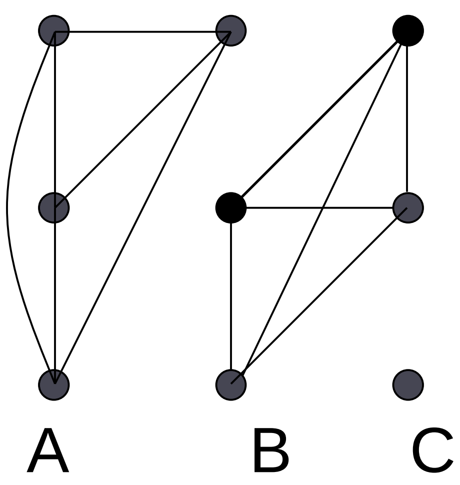


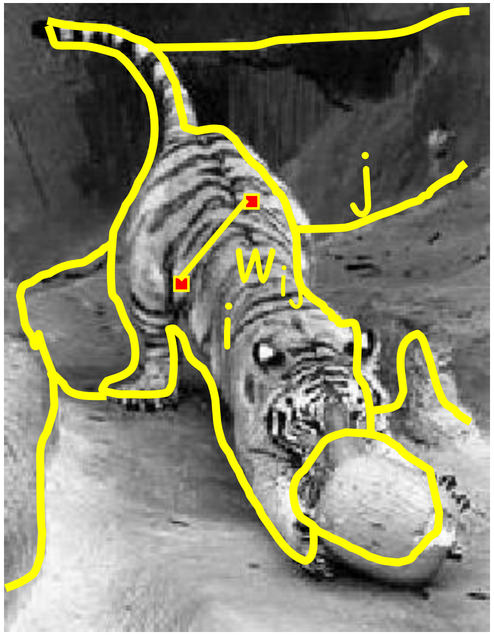


Source: S. Seitz

Segmentation by graph partitioning

- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low affinity
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments





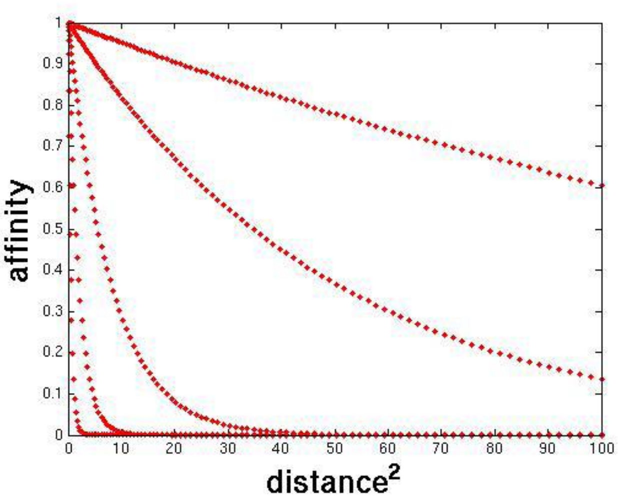
Source: S. Seitz

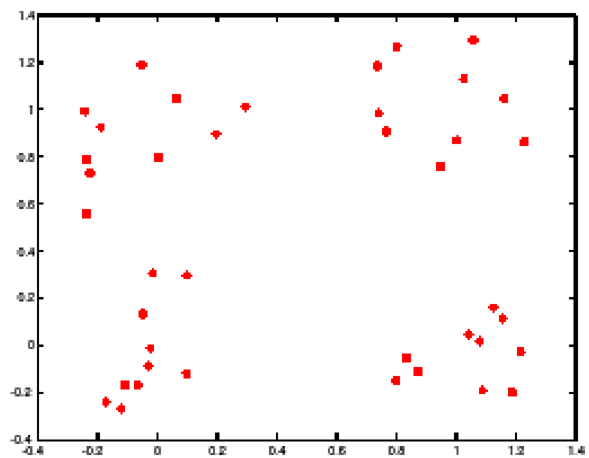
Measuring affinity

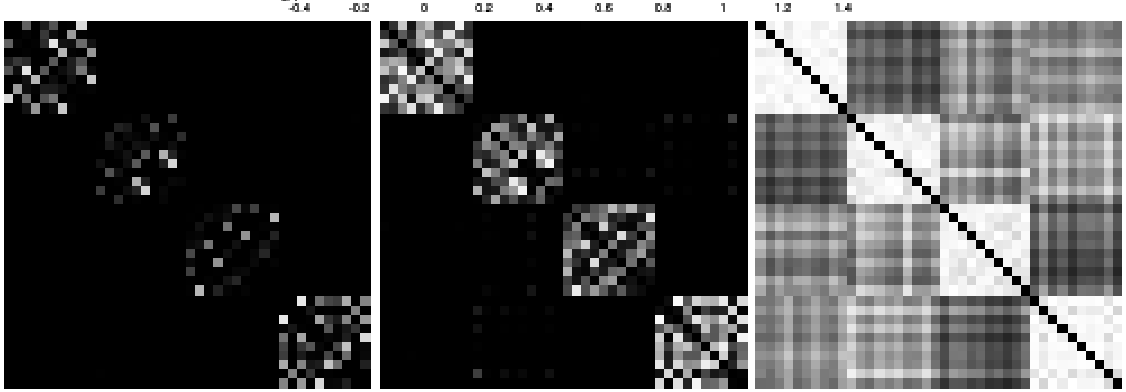
- Distance $$aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\left\|x-y\right\|^2\right\}$$
- Intensity $$aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\left\|I(x)-I(y)\right\|^2\right\}$$
- Color $$aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\underbrace{dist\left(c(x),c(y)\right)}_{\text{(some suitable color space distance)}}^2\right\}$$
- Texture $$aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\underbrace{\left\|f(x)-f(y)\right\|}_{\text{(vectors of filter outputs)}}^2\right\}$$

Scale affects affinity

- Small σ: group only nearby points
- Large σ: group far-away points

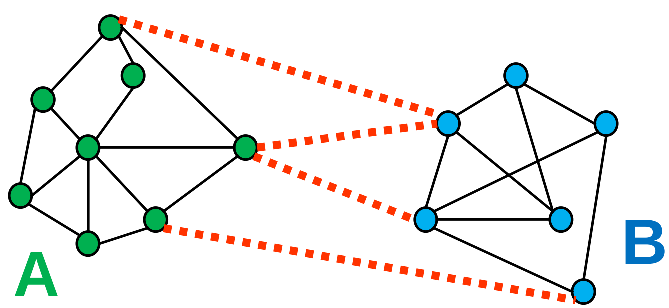






Graph cut

- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?

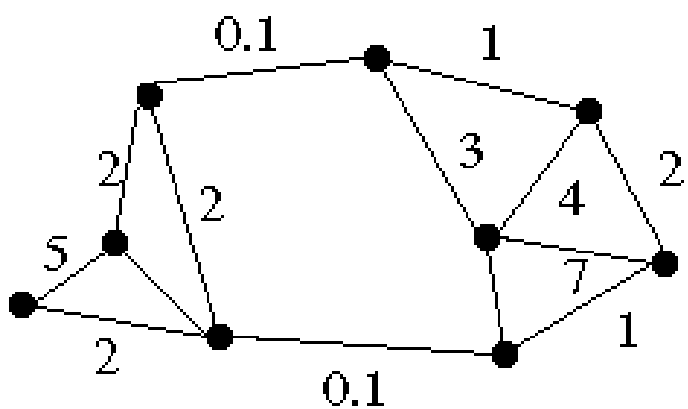


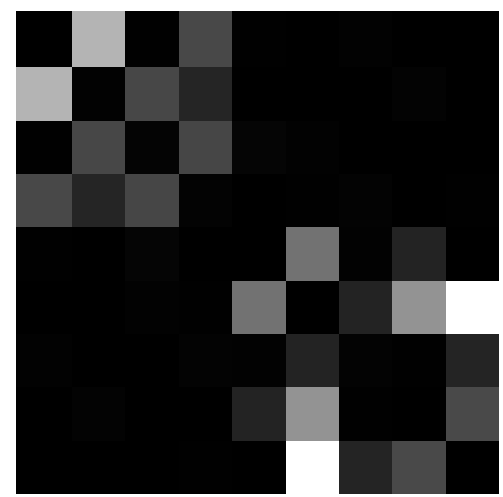
Source: S. Seitz

Minimum cut

- We can segment by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

Minimum cut example

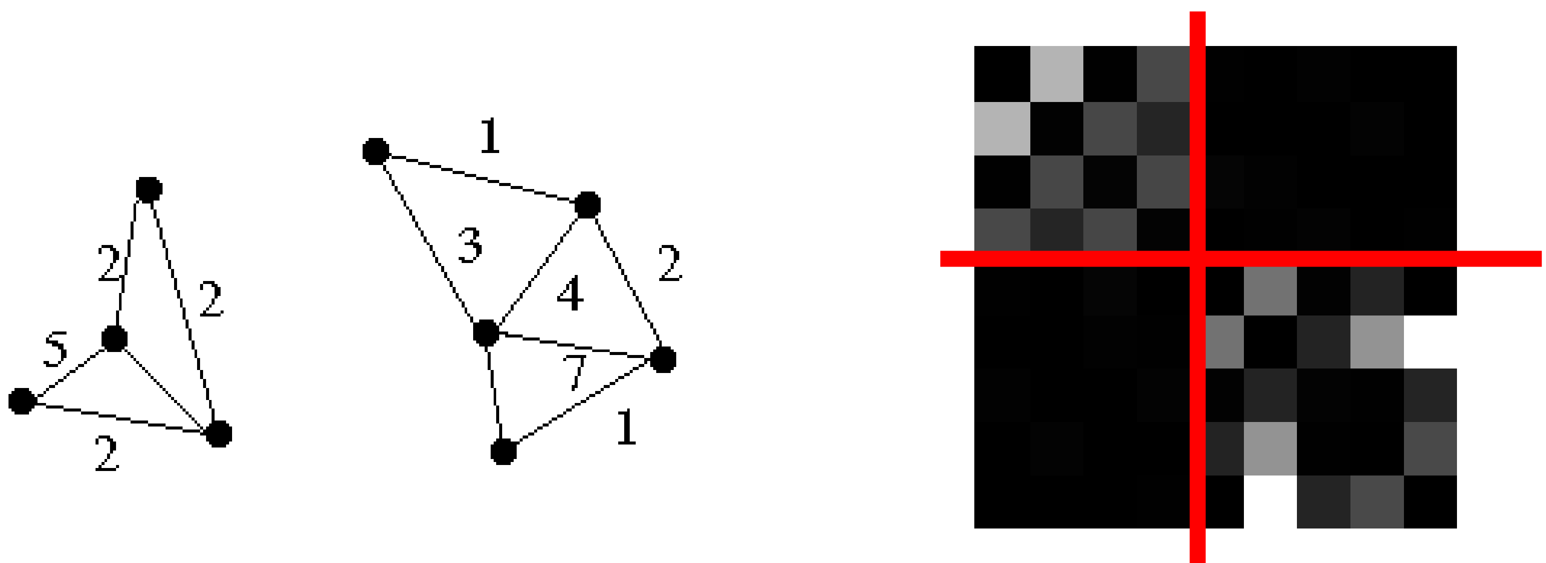




Minimum cut

- We can segment by finding the *minimum cut* in a graph
  - Efficient algorithms exist for doing this

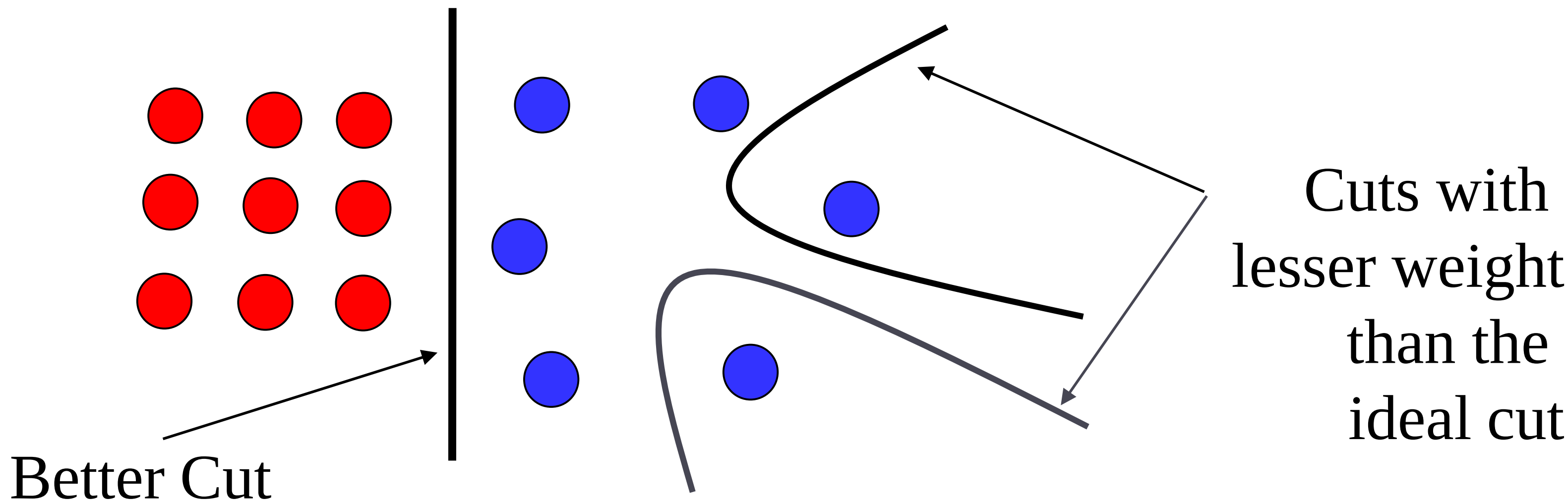
Minimum cut example



- The cut is defined by the block diagonal structure of the affinity matrix.
- Can this be generalized?

Minimum cut drawback

- Minimum cut tends to cut off very small, isolated components



Normalized cut (Ncut)

- The drawback of mincut can be fixed by normalizing the cost by the weight of all the edges incident to the segment
- The *normalized cut* cost is:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$cut(A,B)$= sum of weights of all edges between *A* and *B*

$assoc(A,\cancel{B}V)$ = sum of weights of all edges in V that touch *A*

$$Ncut(A,B) = cut(A,B)\frac{1}{\sum_{p\in A} w_{p,q}} + \frac{1}{\sum_{q\in B} w_{p,q}}$$

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Ncut as a generalized eigenvector problem

- Let W be the adjacency matrix of the graph
- Let D be the diagonal matrix with diagonal entries $D(i, i) = \Sigma_j W(i, j)$
- Then the normalized cut cost can be written as

$$Ncut(A, B) = \frac{y^T (D - W) y}{y^T D y}$$

$$y(p) = \begin{cases} 1 & p \in A \\ negative & otherwise \end{cases}$$

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000

Ncut as a generalized eigenvector problem

- Problem:
  Finding the exact minimum of the normalized cut cost is NP-complete (because y is discrete)
- Solution:
  - Relax *y* to take on arbitrary values, then solved by a *generalized eigenvalue problem*

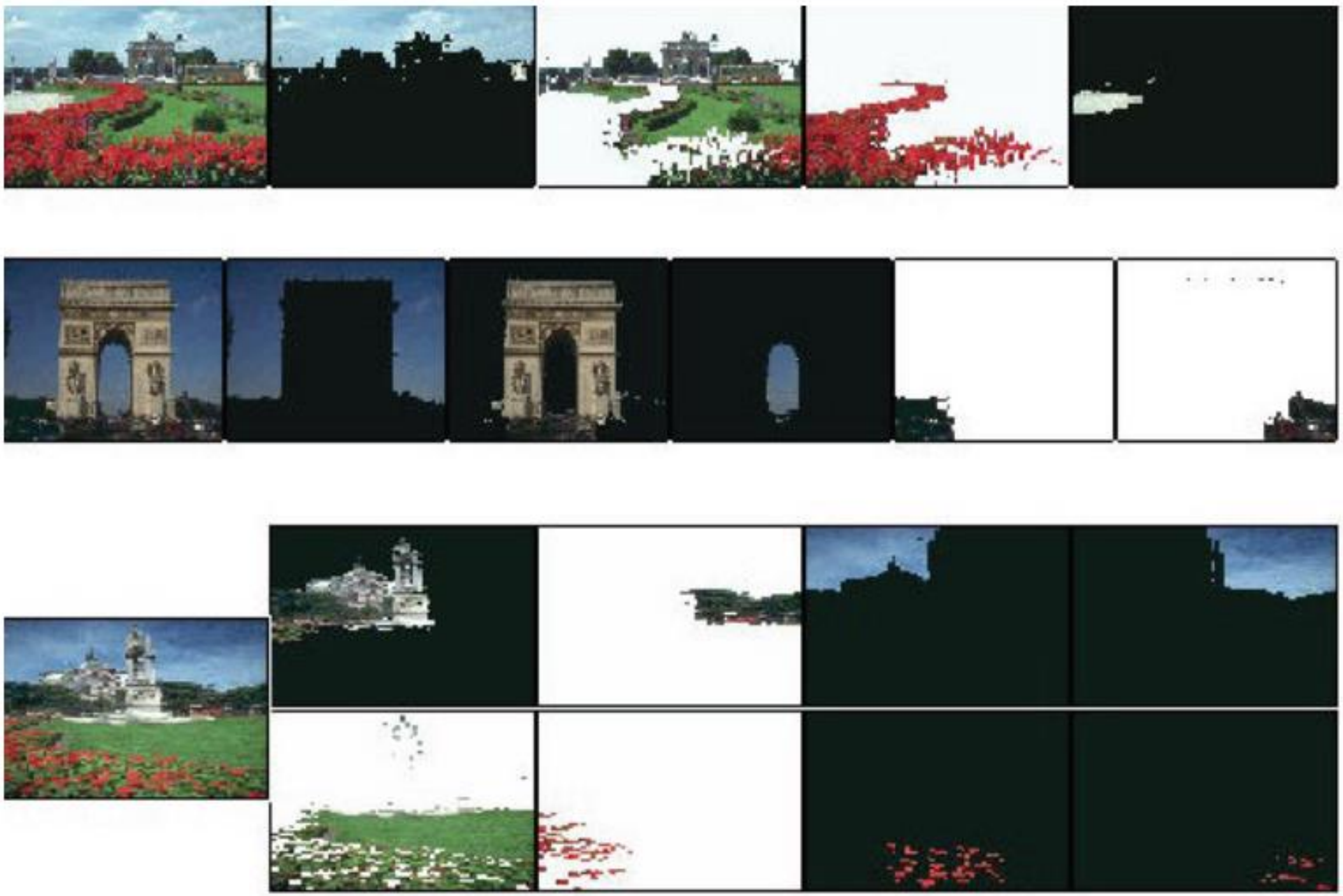
$$N(D-W)y = \lambda Dy$$

  - The solution *y* is given by the eigenvector corresponding to the second smallest eigenvalue
  - Continuous results need to be converted into discrete

Normalized cut algorithm

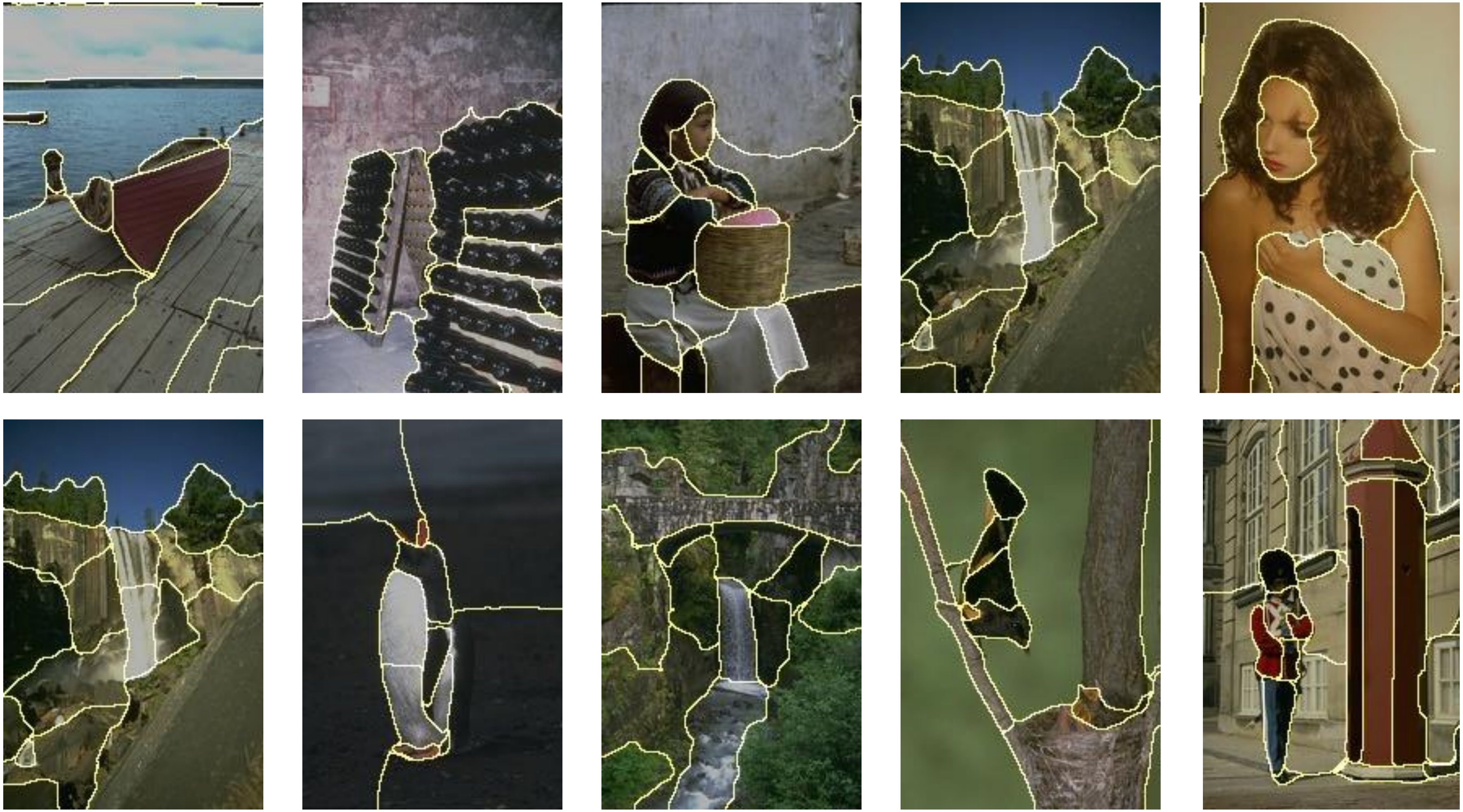
1. Construct a weighted graph $G = (V,E)$ from an image
2. Connect each pair of pixels, and assign weights
   $w(i,j) = prob(i,j \text{ belong to same region})$
3. Compute diagonal matrix $\mathrm{D(i, i)} = \Sigma_{\mathrm{j}} \mathrm{W(i, j)}$
4. Solve $(D - W)y = \lambda Dy$ for the eigenvector with the second smallest eigenvalue
5. Threshold eigenvector to get a discrete cut
6. Recursively partition the segmented parts, if necessary

Example results



Example results

Results: Berkeley Segmentation Engine

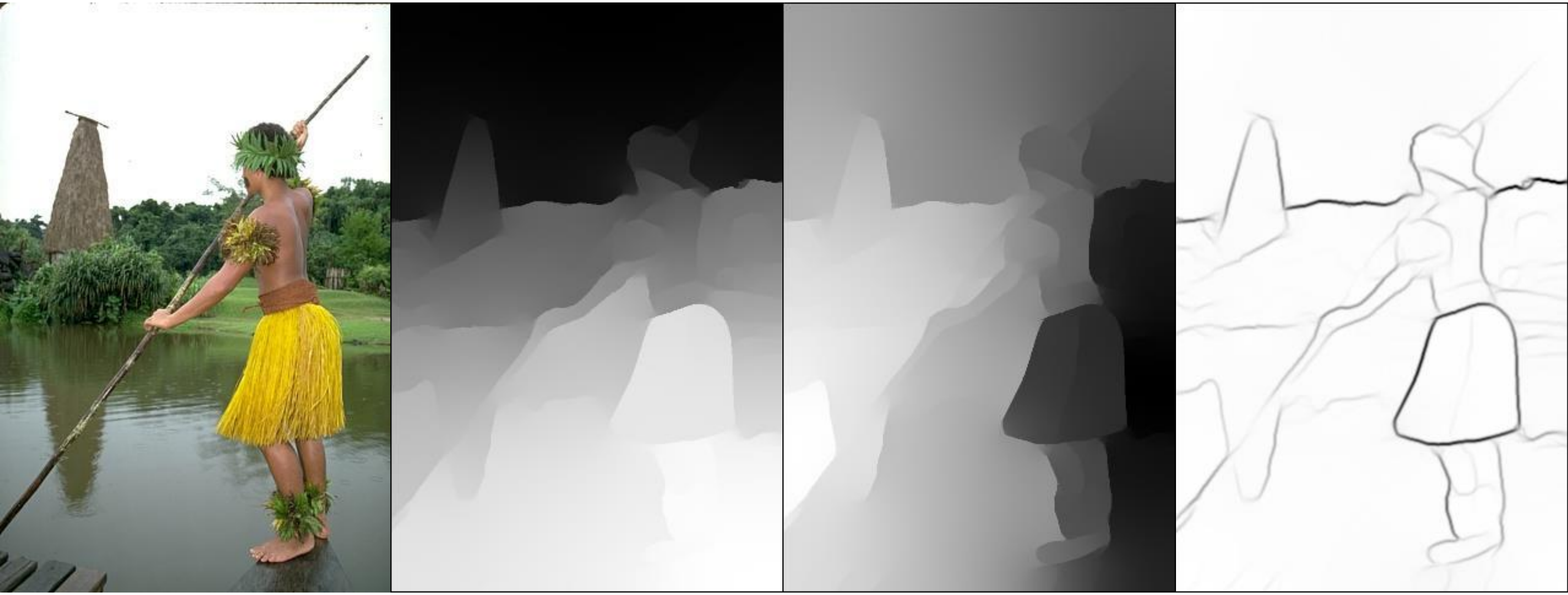


http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: Pros and cons

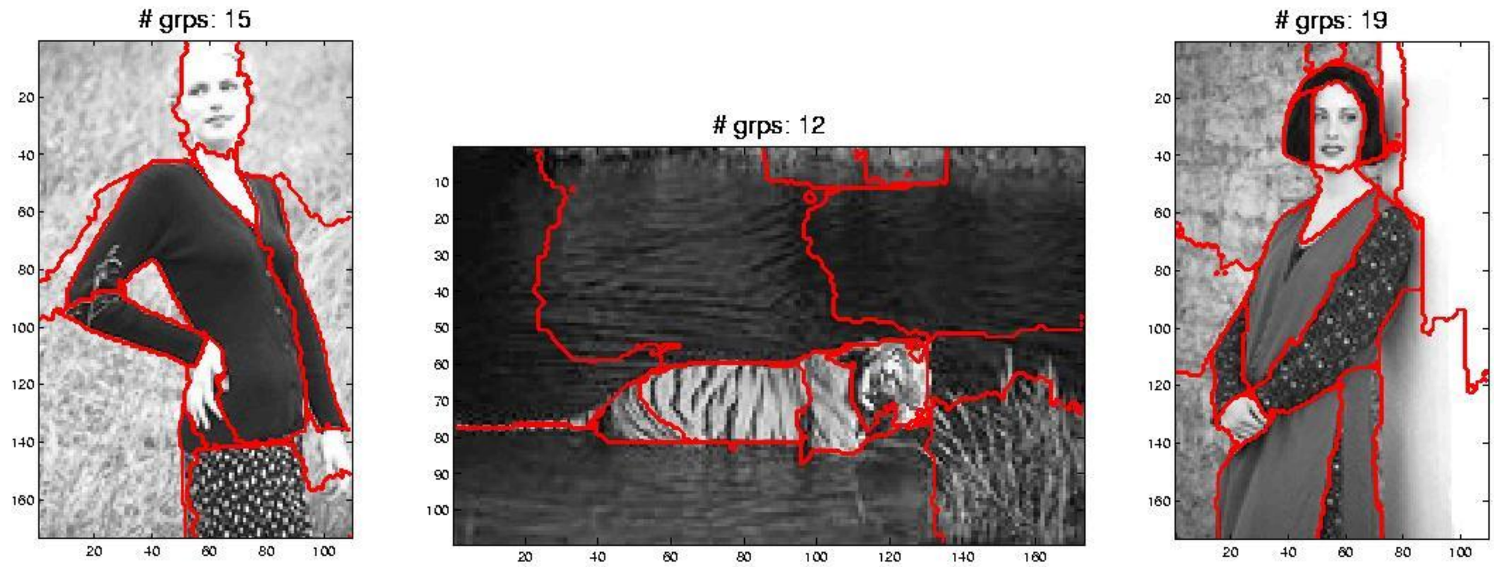
- Pros
  - Generic framework, can be used with many different features and affinity formulations
  - No model or data distribution
- Cons
  - High storage requirement and time complexity
  - Bias towards partitioning into equal segments

Eigenvectors carry contour information



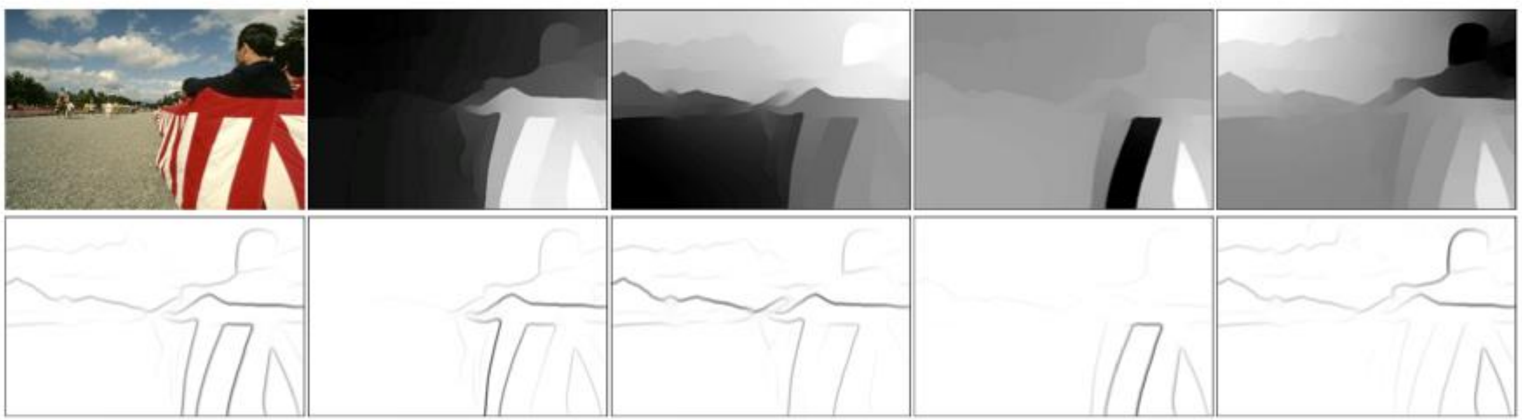
Avoiding Ncut drawback

- Do not try to find regions from the eigenvectors



Avoiding Ncut drawback

- Do not try to find regions from the eigenvectors
- Key idea:
  - Reshape eigenvectors into images
  - Compute edge probability P_b on eigenvector images
  - Final edge probability is the sum of all responses



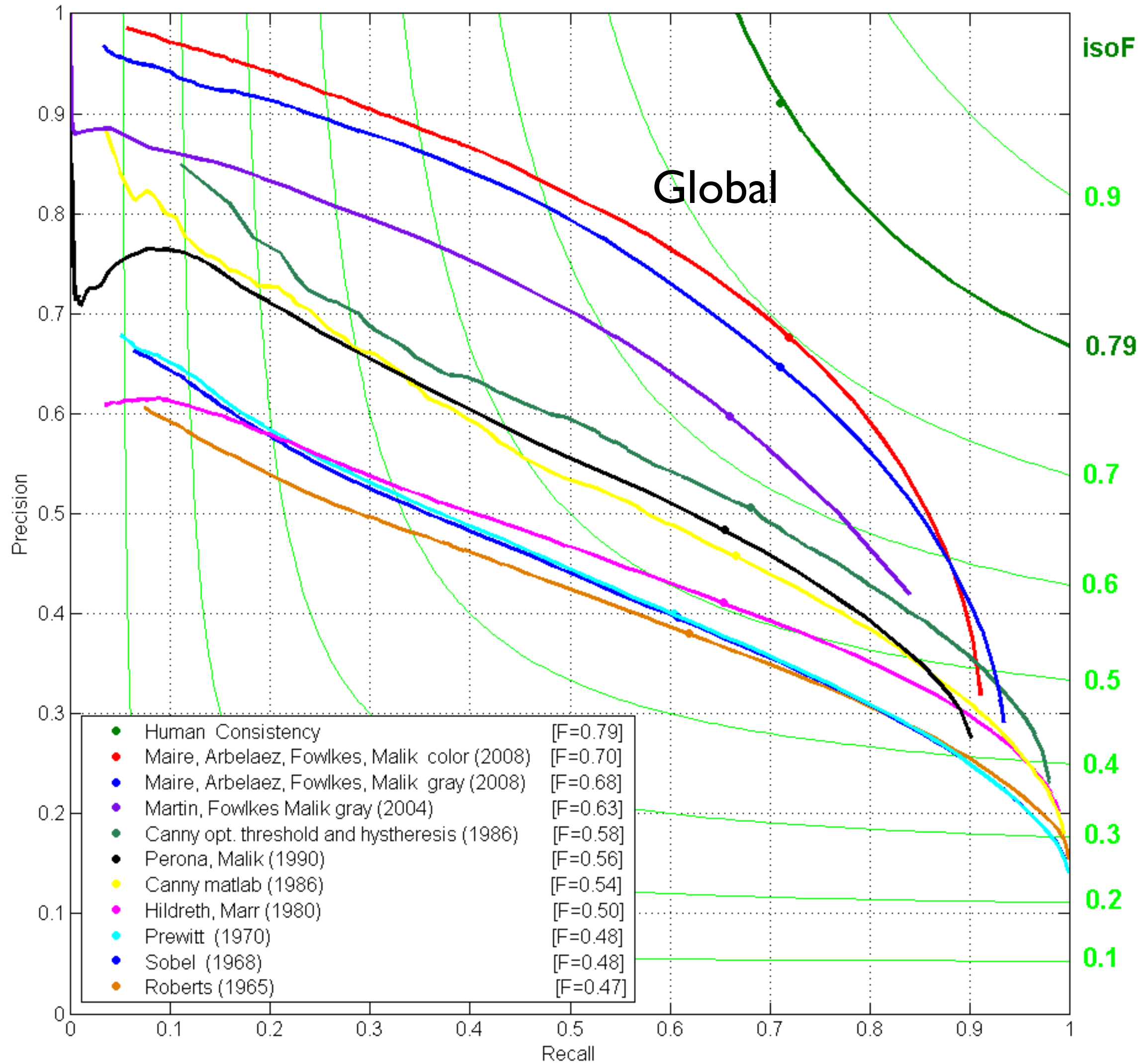
eigenvectors

edges

Example results

Pb | After threshold | Final Pb continuous

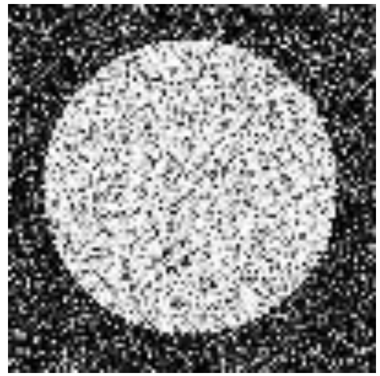
Contour detection ~2008 (color)



Today

- Graph theoretic segmentation
  - Normalized cuts
- Segmentation as energy minimization
  - Markov random fields

Segmentation as energy minimization



P(foreground | image)

Y , 0 1

Normalizing constant called "partition function"

$$P(\mathbf{y};\theta,data)=\frac{1}{Z}\prod_{i=1..N} f_1(y_i;\theta,data)\prod_{i,j\in edges} f_2(y_i,y_j;\theta,data)$$

Labels to be predicted

Individual predictions

Pairwise predictions

Writing Likelihood as an "Energy"

$$P(\mathbf{y};\theta,data)=\frac{1}{Z}\prod_{i=1..N}p_1(y_i;\theta,data)\prod_{i,j\in edges}p_2(y_i,y_j;\theta,data)$$

-log

$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

Cost of assignment y_i

Cost of pairwise assignment y_i, y_j

Notes on energy-based formulation

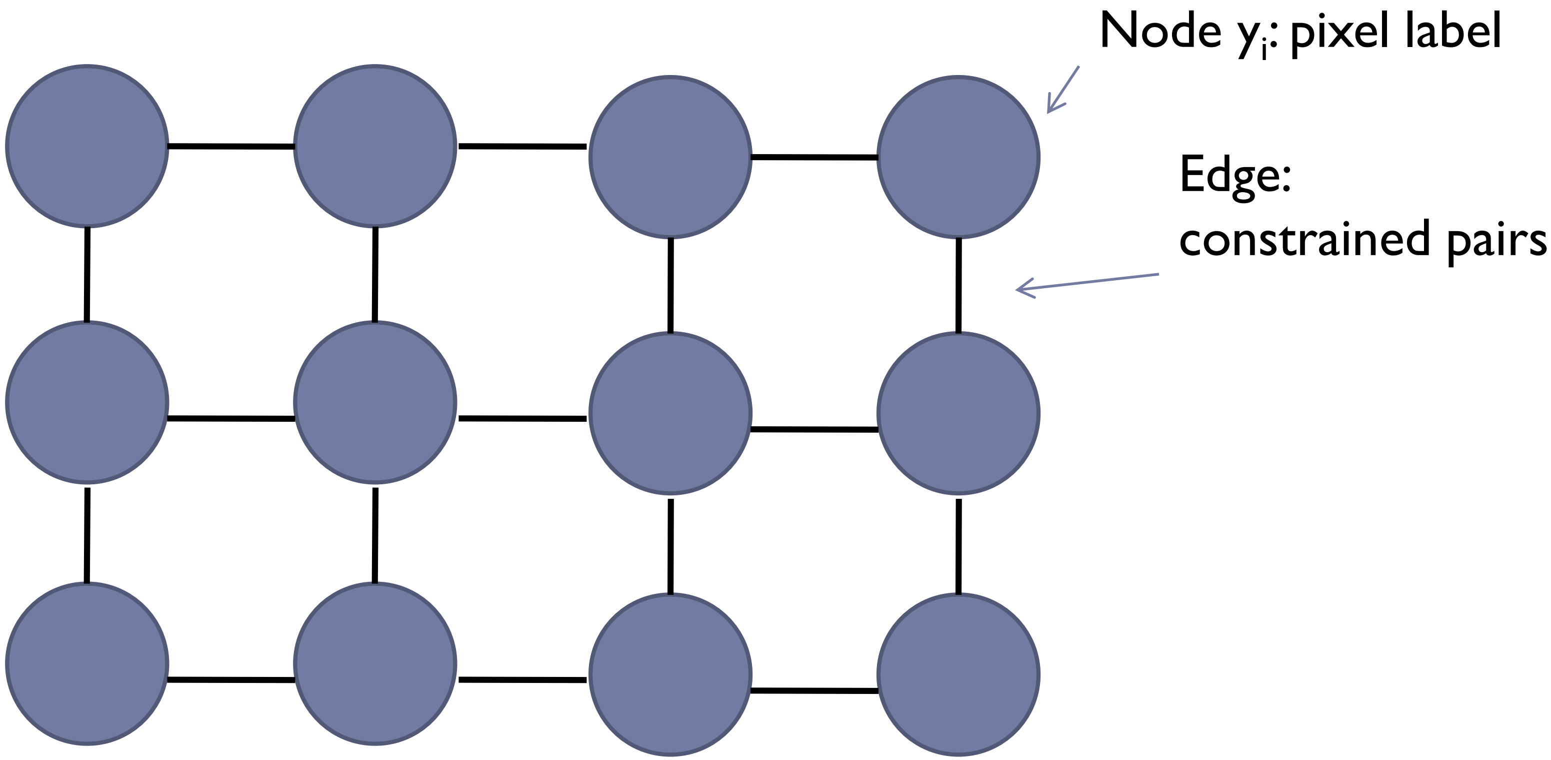
$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

- Primarily used when you only care about the most likely solution (not the confidences)

- Can think of it as a general cost function

- Can have larger "cliques" than 2
  - Clique is the set of variables that go into a potential function

Markov Random Fields



Cost to assign a label to each pixel

Cost to assign a pair of labels to connected pixels

$$Energy(\mathbf{y};\theta,data)=\sum_{i}\psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

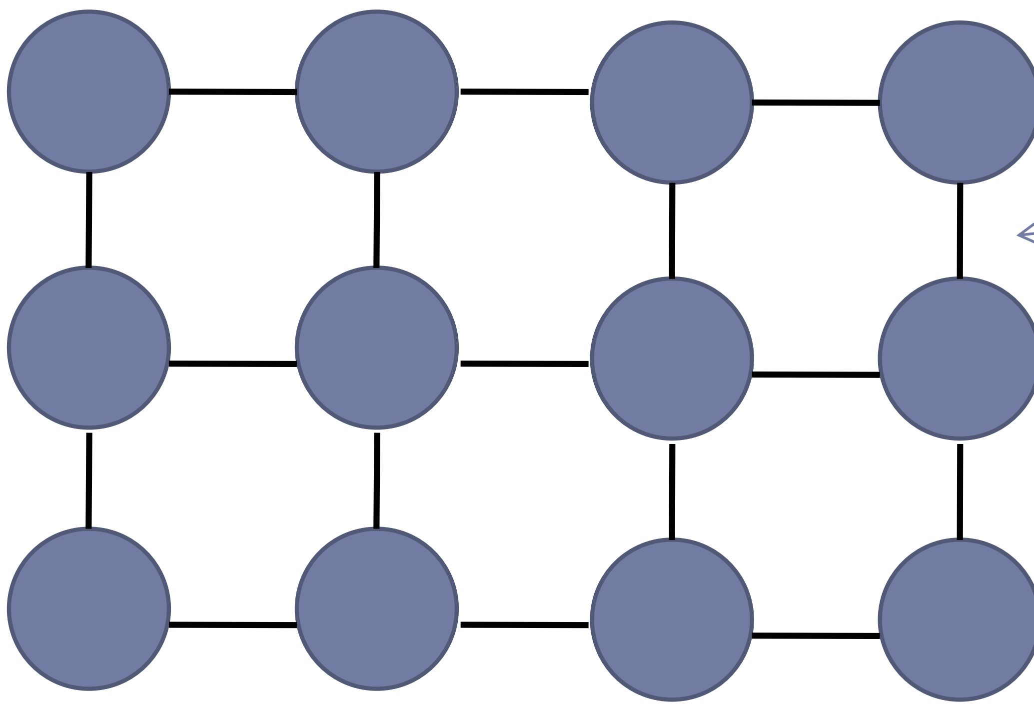
Markov Random Fields

- Example: "label smoothing" grid

Unary potential

0: -logP(y_i = 0 | data)

1: -logP(y_i = 1 | data)



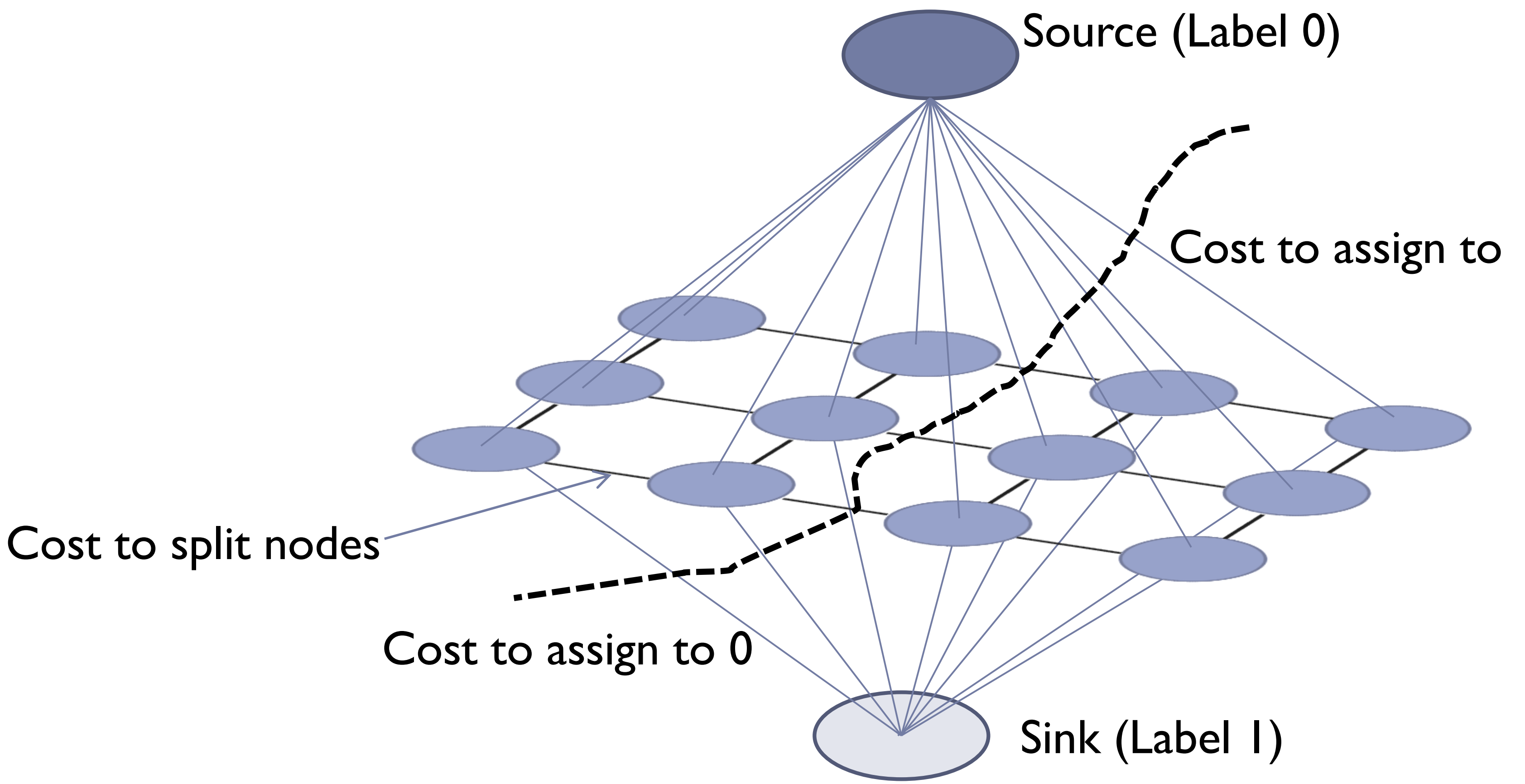
Pairwise Potential

	0	1
0	0	K
1	K	0

K>0

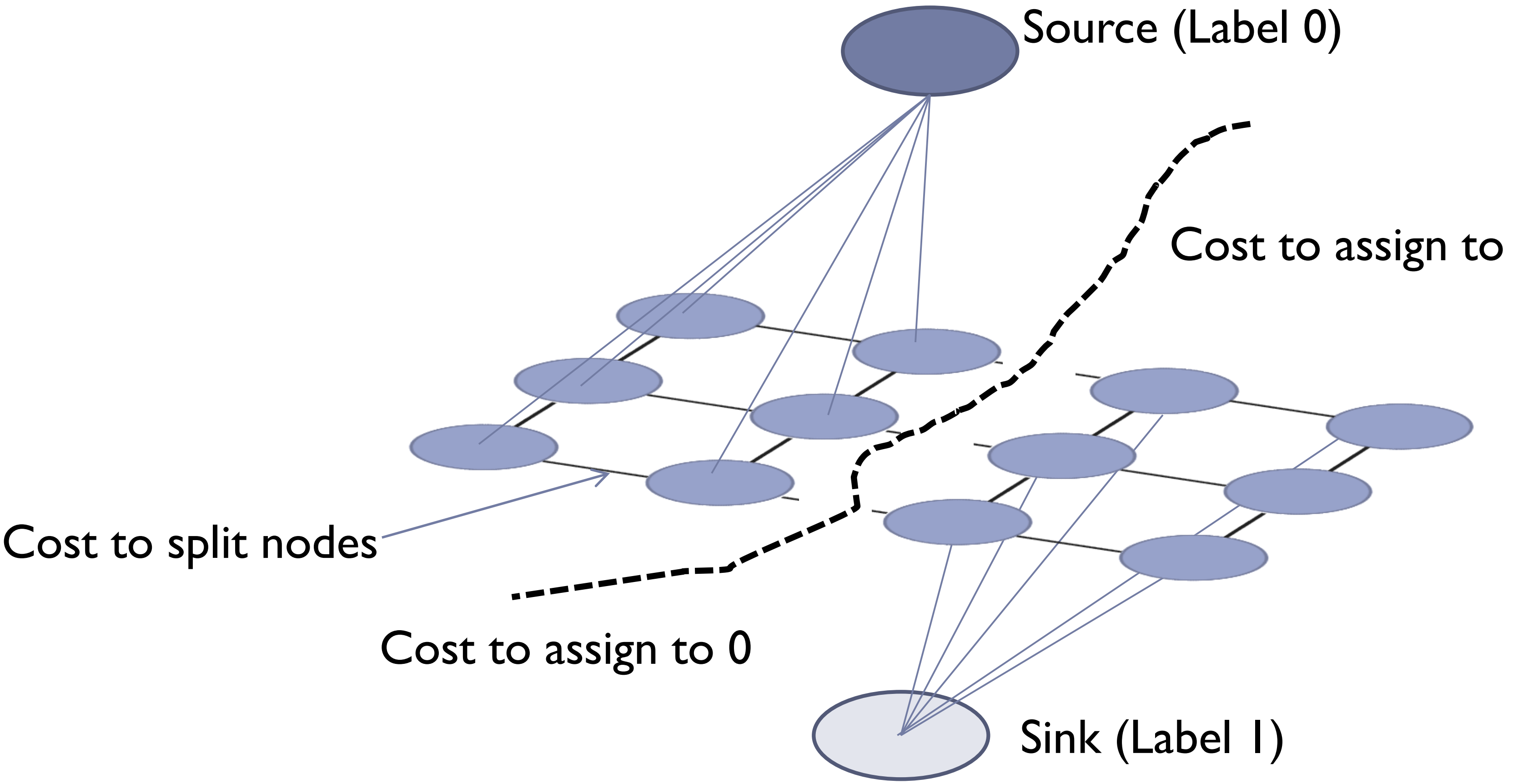
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j \in edges} \psi_2(y_i, y_j;\theta,data)$$

Solving MRFs with graph cuts



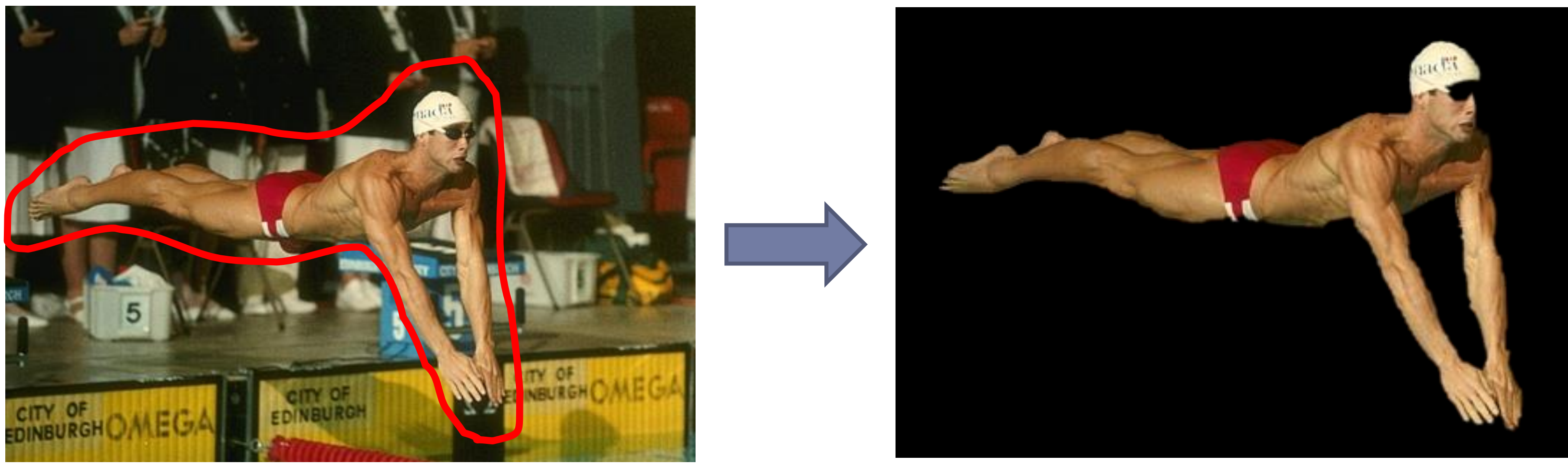
$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

Solving MRFs with graph cuts



$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

GrabCut segmentation

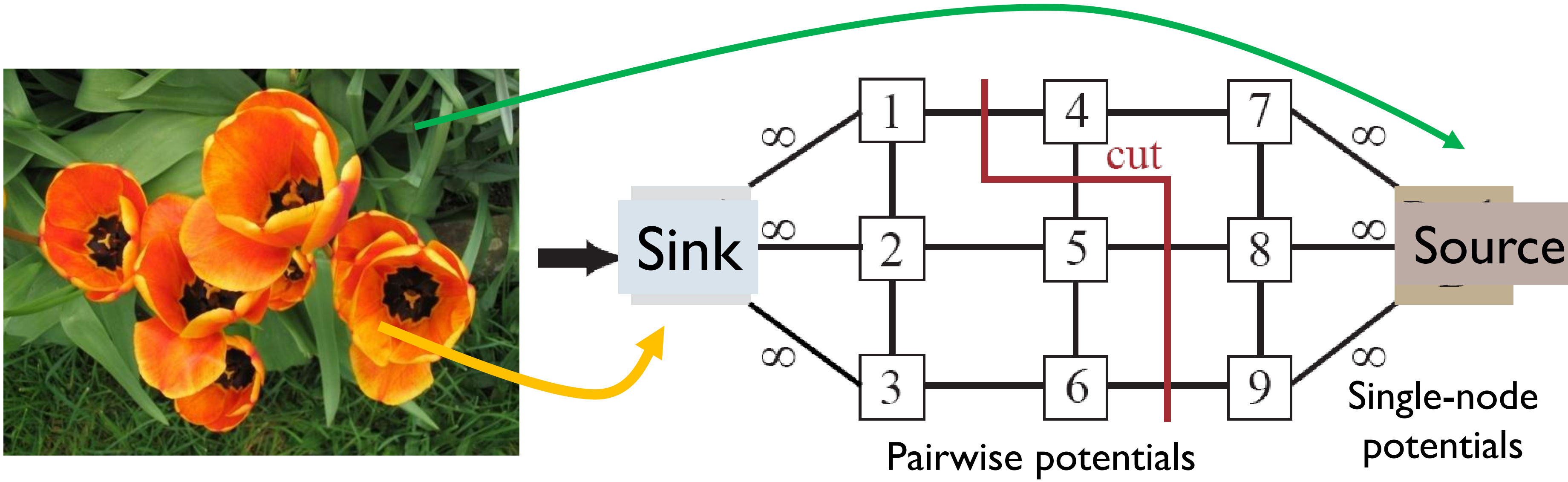


User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

GrabCut

- Convert MRF into source-sink graph



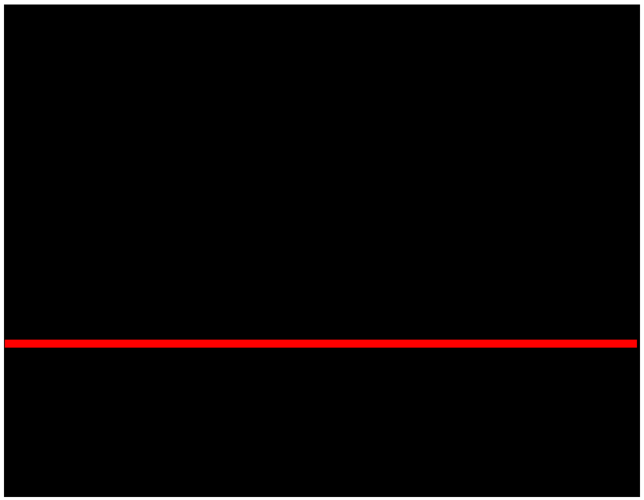
- Minimun cost can be computed in polynomial time

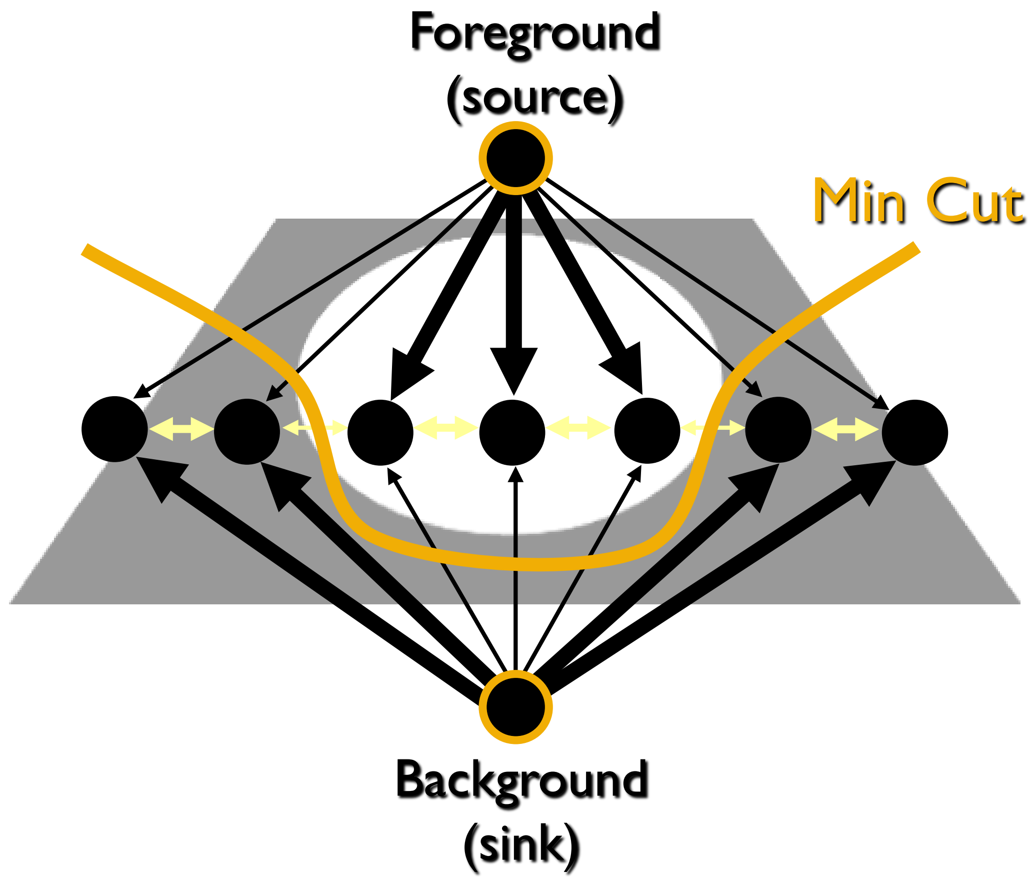
[Kwatra et al., SIGGRAPH 2003]

Graph cuts

Boykov and Jolly (2001)

Image





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

Min Cut: Global minimal energy in polynomial time

Source: Rother

Binary segmentation as energy minimization

- Define a labeling *L* as an assignment of each pixel with a 0-1 label (background or foreground)

- Problem statement: find the labeling *L* that minimizes

רוצים למזער
אנרגיה

משמעות, אנרגיה

אומר כמה טוב
מבוסס
לפיקסלים

$$E(L) = \underbrace{E_d(L)}_{\text{match cost}} + \lambda \underbrace{E_s(L)}_{\text{smoothness cost}}$$

(how similar is each pixel to the foreground / background?)

שואל פיקסלים עם
קרבה שמשויכות לא
כמה צריכים מתוכר ריש

או מנודקן יש

$$E(L) = E_d(L) + \lambda E_s(L)$$

$$E_d(L) = \sum_{(x,y)} C(x, y, L(x, y))$$

סימון של user $\tilde{L}(x, y)$

$$C(x, y, L(x, y)) = \begin{cases} \infty & \text{if } L(x, y) \neq \tilde{L}(x, y) \\ C'(x, y, L(x, y)) & \text{otherwise} \end{cases}$$

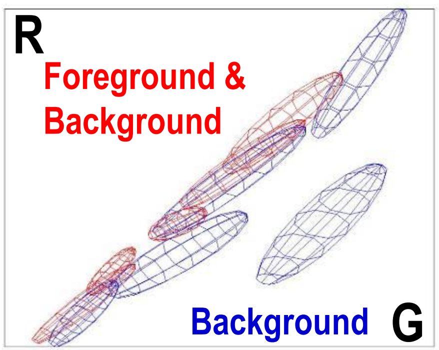
user

$C'(x, y, 0)$: "distance" from pixel to background pixels

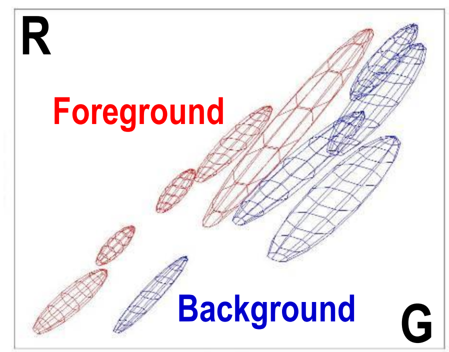
$C'(x, y, 1)$: "distance" from pixel to foreground pixels

usually computed by creating a color model from user-labeled pixels

Colour Model



Iterated graph cut



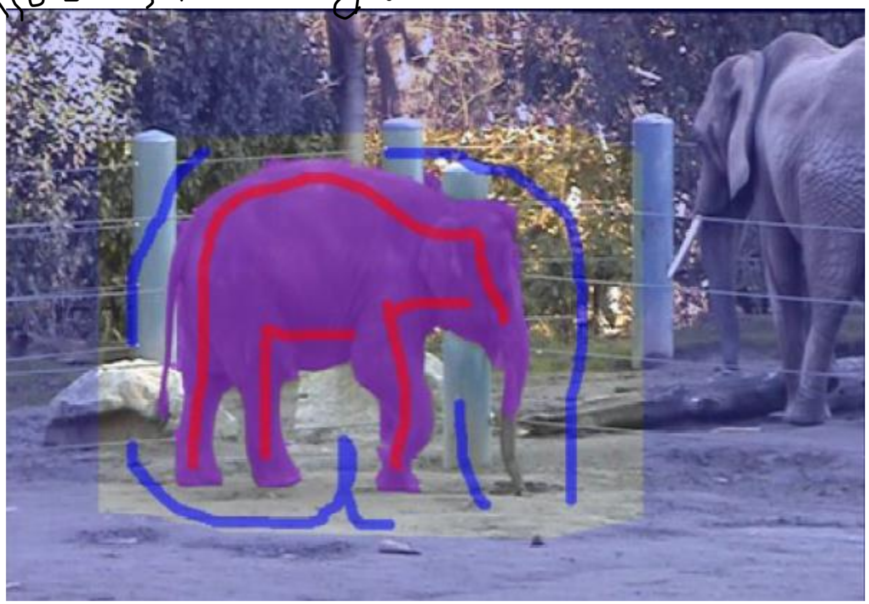
Gaussian Mixture Model (typically 5-8 components)

$$E(L) = E_d(L) + \lambda E_s(L)$$

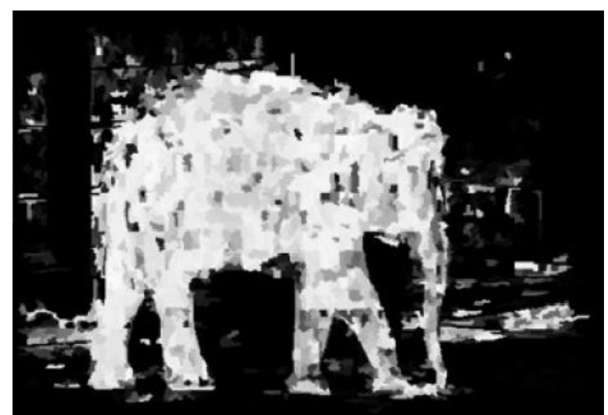
רווחית

צריך כסימון של מ רמזנו

משפיעים על מאב ב מתלונדת



$C'(x, y, 0)$



$C'(x, y, 1)$

$$E(L) = E_d(L) + \lambda E_s(L)$$

- Neighboring pixels should generally have the same labels
  - Unless the pixels have very different intensities

אלגו' הלכה רוצה לחבר בין פיקסלים
סמוכים

$$E_s(L) = \sum_{\text{neighbors } (p,q)} w_{pq} |L(p) - L(q)|$$

w_{pq} : similarity in intensity of *p* and *q*

$w_{pq} = 0.1$

$w_{pq} = 10.0$

גבוה

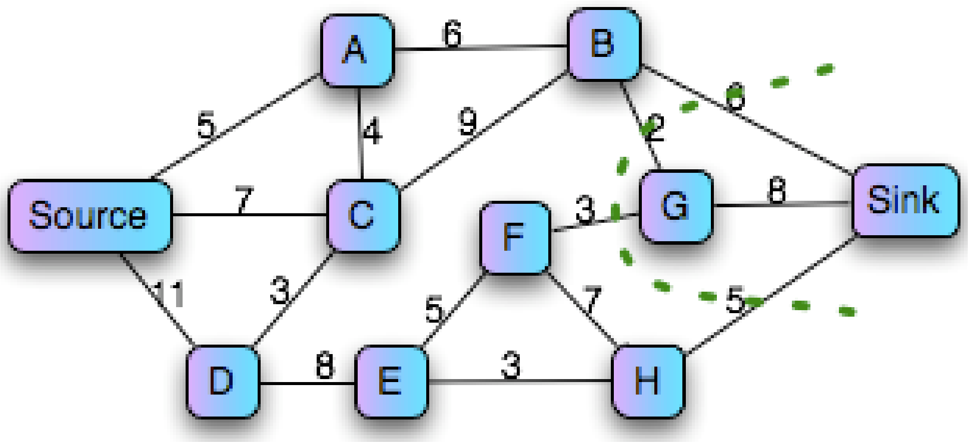
כשמספר יו' ה פרקטלן מעודד

Binary segmentation as energy minimization

$$E(L) = E_d(L) + \lambda E_s(L)$$

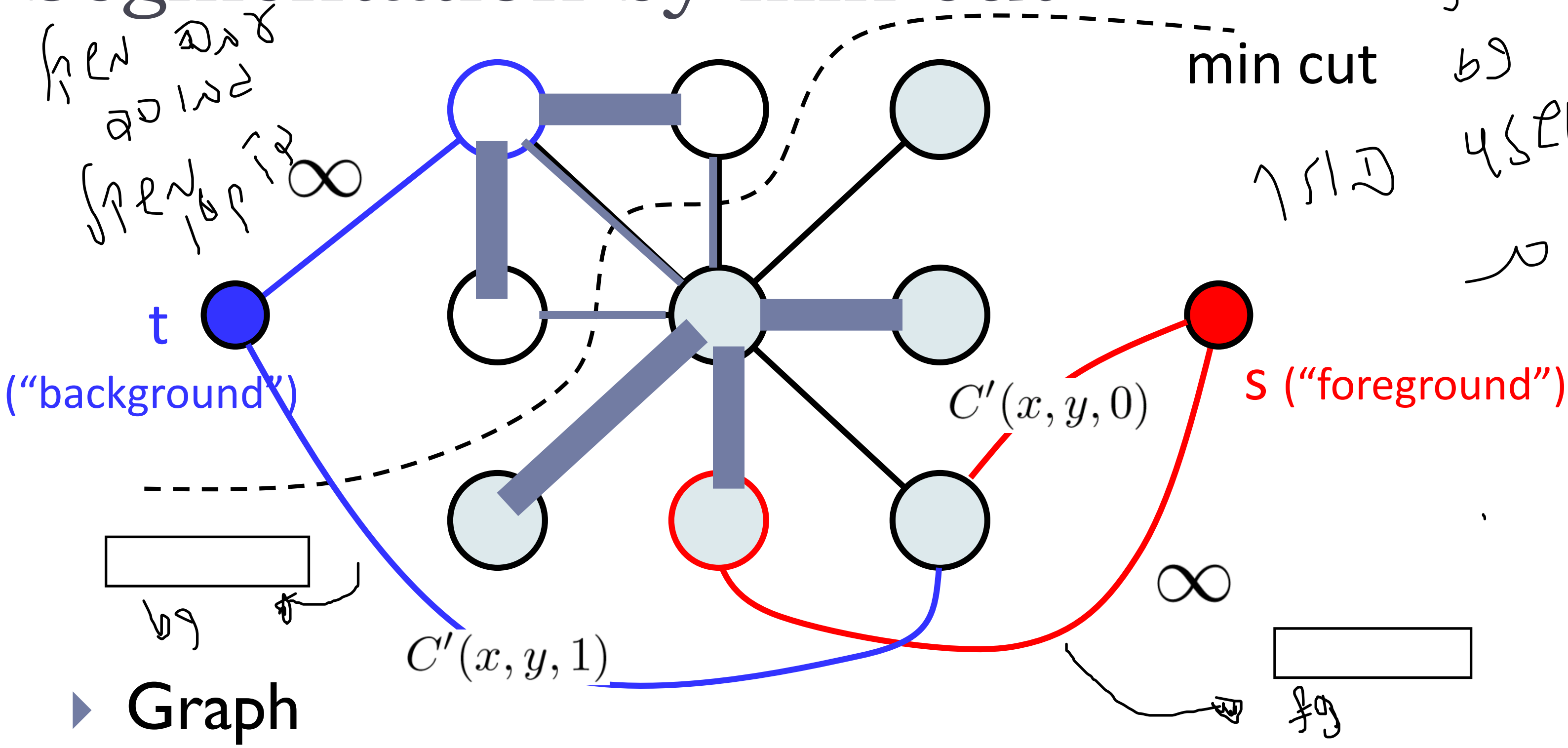
- For this problem, we can easily find the global minimum!
- Use max flow / min cut algorithm

Graph min cut problem



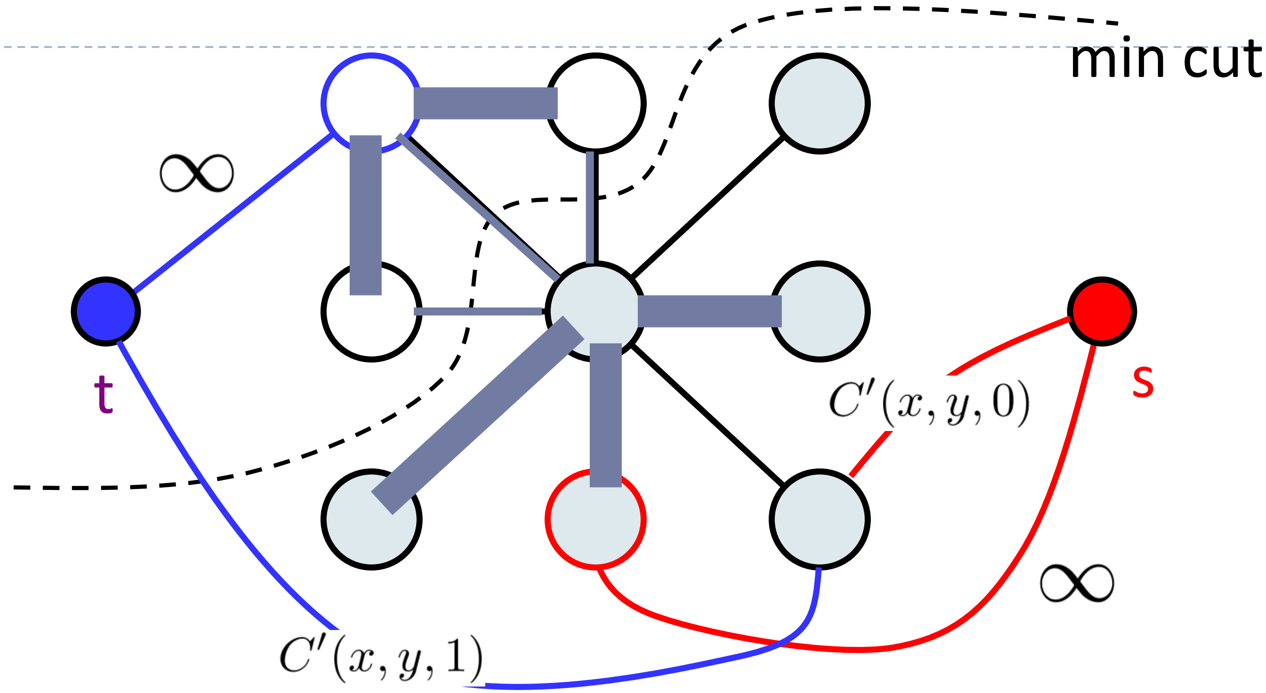
- Given a weighted graph *G* with source and sink nodes (*s* and *t*), partition the nodes into two sets, *S* and *T* such that the sum of edge weights spanning the partition is minimized
  - and $s \in S$ and $t \in T$

Segmentation by min cut



- Graph
  - node for each pixel, link between adjacent pixels
  - specify a few pixels as foreground and background
    - create an infinite cost link from each bg pixel to the *t* node
    - create an infinite cost link from each fg pixel to the *s* node
    - create finite cost links from *s* and *t* to each other node
  - compute min cut that separates s from t
    - The min-cut max-flow theorem [Ford and Fulkerson 1956]

Segmentation by min cut



- The partitions *S* and *T* formed by the min cut give the optimal foreground and background segmentation
- I.e., the resulting labels minimize

$$E(d) = E_d(d) + \lambda E_s(d)$$

GrabCut segmentation

1. Define graph
   - usually 4-connected or 8-connected
     - Divide diagonal potentials by sqrt(2)
2. Define unary potentials
   - Color histogram or mixture of Gaussians for background and foreground

$$unary\_potential(x) = -\log\left(\frac{P(c(x);\theta_{foreground})}{P(c(x);\theta_{background})}\right)$$

3. Define pairwise potentials

$$edge\_potential(x,y) = k_1 + k_2 \exp\left\{\frac{-\left\|c(x)-c(y)\right\|^2}{2\sigma^2}\right\}$$

4. Apply graph cuts
5. Return to 2, using current labels to compute foreground, background models

GrabCut

חוזרים על המבלין בגלל
הסימון של ם אשר

שיפור לאלגוריתם
המקורי

Grabcut [Rother et al., SIGGRAPH 2004]

GrabCut

- Implemented in MS office …. Let's try it

Reported results

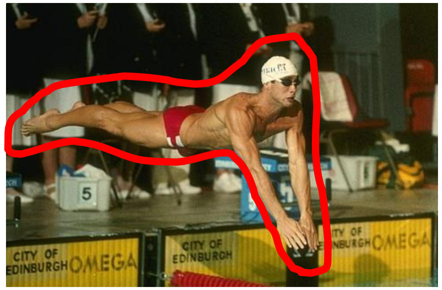
Easier examples

More difficult Examples

Camouflage &

Fine structure

Harder Case



Graph cuts with multiple labels

- Alpha expansion

  Repeat until no change

  For $\alpha = 1..M$

  Assign each pixel to current label or α (2-class graphcut)

  - Achieves "strong" local minimum

- Alpha-beta swap

  Repeat until no change

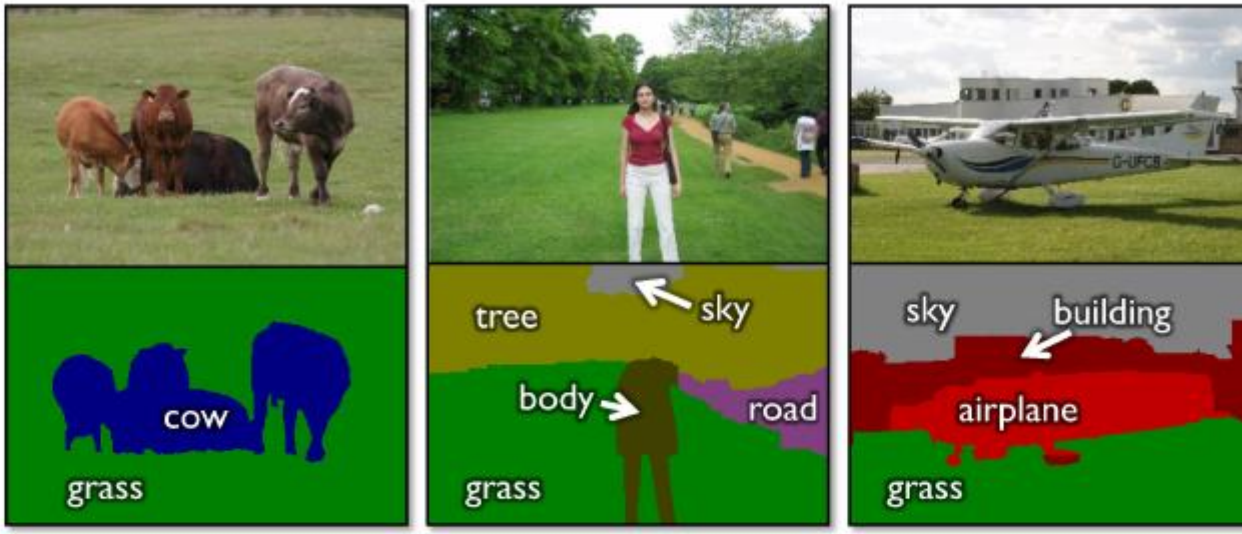
  For $\alpha = 1..M, \quad \beta = 1..M$ (except α)

  Re-assign all pixels currently labeled as α or β to one of those two labels while keeping all other pixels fixed

Other application: synthesis

מיזוג בין שתי תמונות בעזרת Graph Cuts

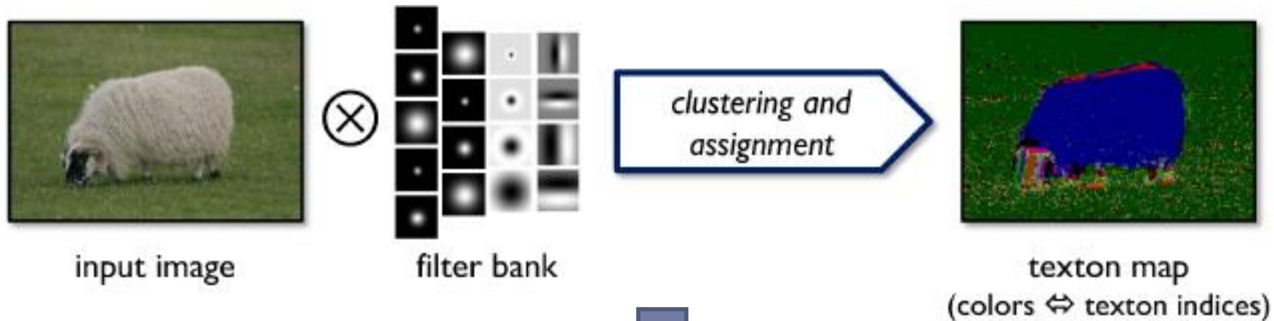
Using graph cuts for recognition



object classes	building	grass	tree	cow	sheep	sky	airplane	water	face	car
bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat

TextonBoost (Shotton et al. 2009 IJCV)

Using graph cuts for recognition



Unary Potentials from classifier

+

Alpha Expansion Graph Cuts

(note: edge potentials are from input image also; this is illustration from paper)

TextonBoost (Shotton et al. 2009 IJCV)

Summary: MRF and graph-cuts

- Pros:
  - Powerful, based on probabilistic model (MRF)
  - Applicable to a wide range of problems
  - Very efficient algorithms available for many problems
  - Becoming a standard for segmentation

- Cons
  - Graph-cuts can only solve a limited class of problems:
    - Sub-modular energy functions
    - Can only capture part of the power of MRF
  - Only approximate solutions available for multi-label case

Graph cuts: Pros and Cons

- Pros
  - Very fast inference
  - Can incorporate data likelihoods and priors
  - Applies to a wide range of problems (stereo, image labeling, recognition)
- Cons
  - Not always applicable (associative only)
  - Need unary terms (not used for bottom-up segmentation, for example)

user 개입
(with 최소)

- Use whenever applicable

Further reading and resources

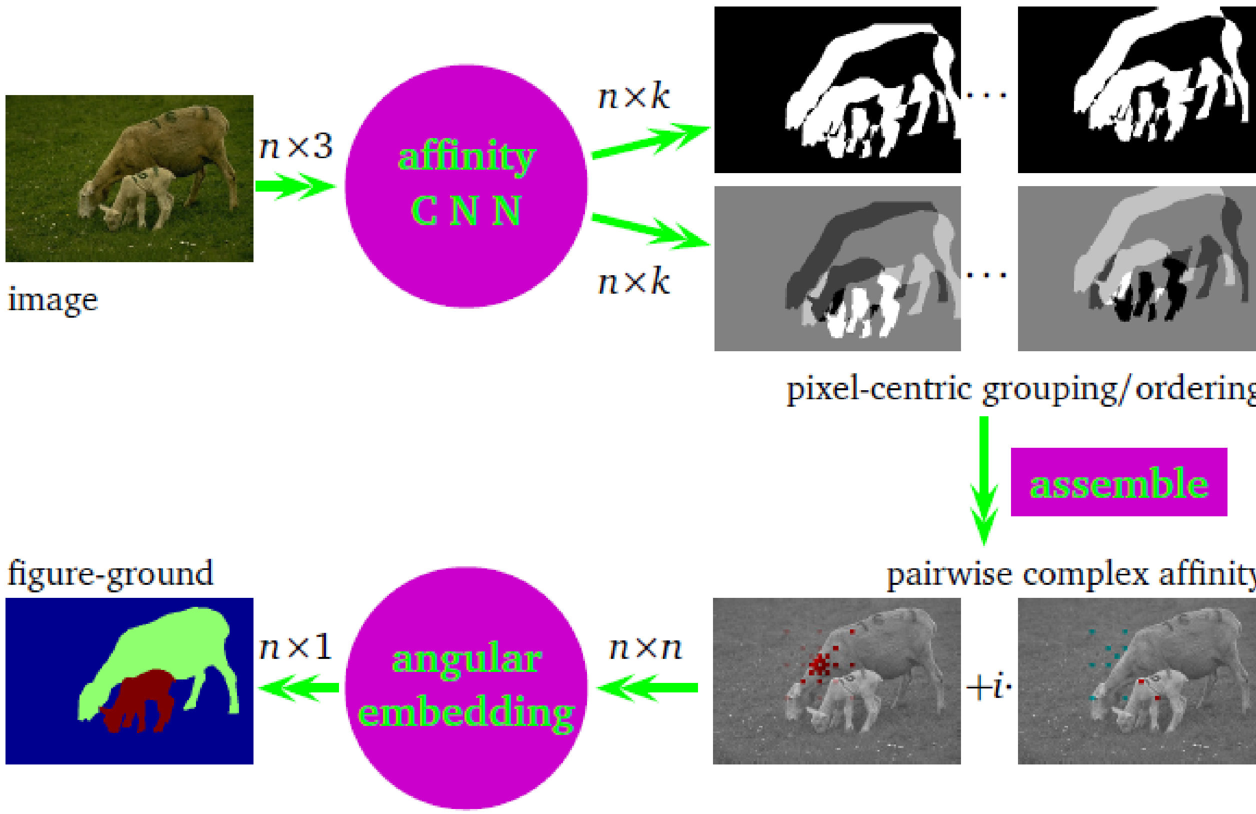
- Graph cuts
  - http://www.cs.cornell.edu/~rdz/graphcuts.html
  - Classic paper: What Energy Functions can be Minimized via Graph Cuts? (Kolmogorov and Zabih, ECCV '02/PAMI '04)

- Belief propagation

  Yedidia, J.S.; Freeman, W.T.; Weiss, Y., "Understanding Belief Propagation and Its Generalizations", Technical Report, 2001: http://www.merl.com/publications/TR2001-022/

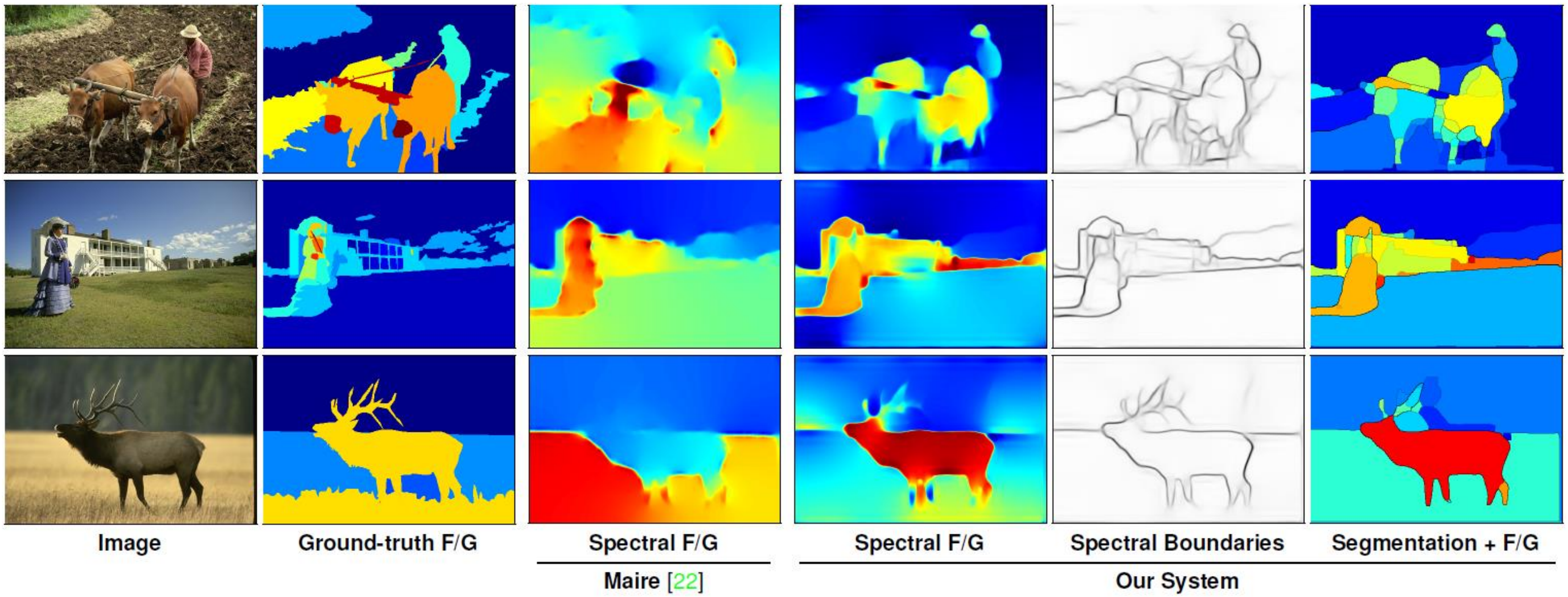
CNN for affinity learning

- Maire and Yu, 2015



CNN for affinity learning

- Maire and Yu, 2015



End – image segmentation part 2

Now you know how it works