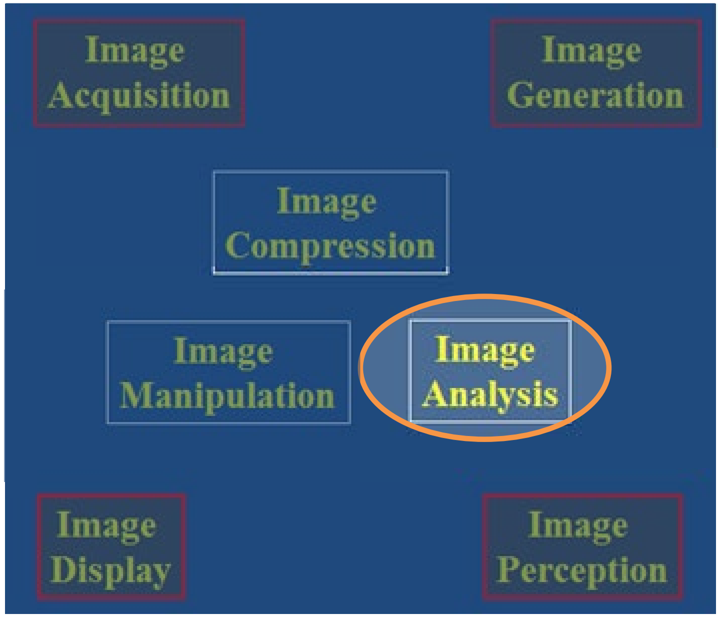
Digital Image Processing

Segmentation via Graphs



DRAFT

Brent M. Dingle, Ph.D. 2015
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

Material in this presentation is largely based on/derived from presentations by: Sventlana Lazebnik, and Noah Snavely

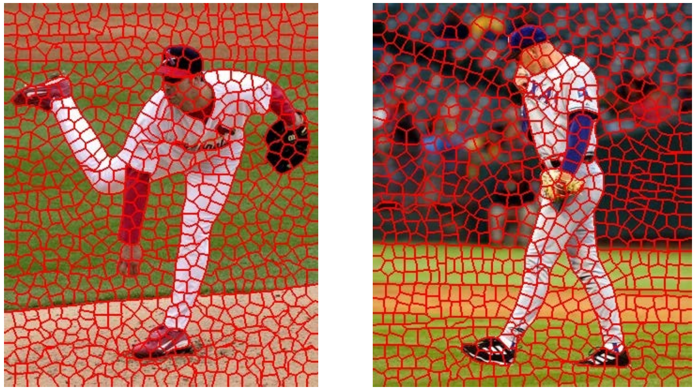
Lecture Objectives

- Previously
  - Image Manipulation and Enhancement
    - Filtering
    - Interpolation
    - Warping
    - Morphing
  - Image Compression
  - Image Analysis
    - Edge Detection
    - Smart Scissors
    - Stereo Image processing
    - Segmentation
- Today
  - Segmentation - Graph Based

source: Noah Snavely

Recall: Image Segmentation

- Group similar looking pixels together for efficiency of additional processing



- Separate image into coherent objects

Recall: Stereo as a Minimization Prob

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

Want each pixel to find a good match in the other image

Adjacent pixels should (usually) move about the same amount

Related: Binary Segmentation

- Separate an image into foreground and background

source: Noah Snavely

Related: Binary Segmentation

- Separate an image into foreground and background

User sketches some strokes on foreground and background

How do we classify the rest of the pixels based on those lines?

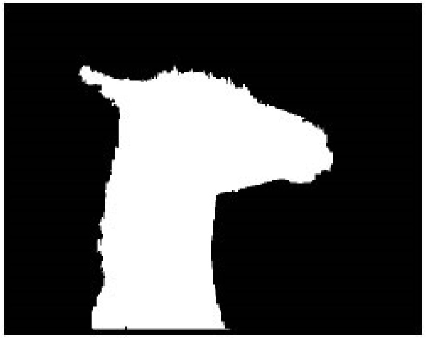
source: Noah Snavely

Binary Segmentation as Min Energy

- Define a labeling, L:
  - as an assignment of each pixel with a 0 or 1 label
    - background or foreground

- Problem:
  - Find the labeling that minimizes

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



Goal: establish a function to measure how similar a pixel is to the foreground (or background)

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

$\tilde{L}(x,y)$

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

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

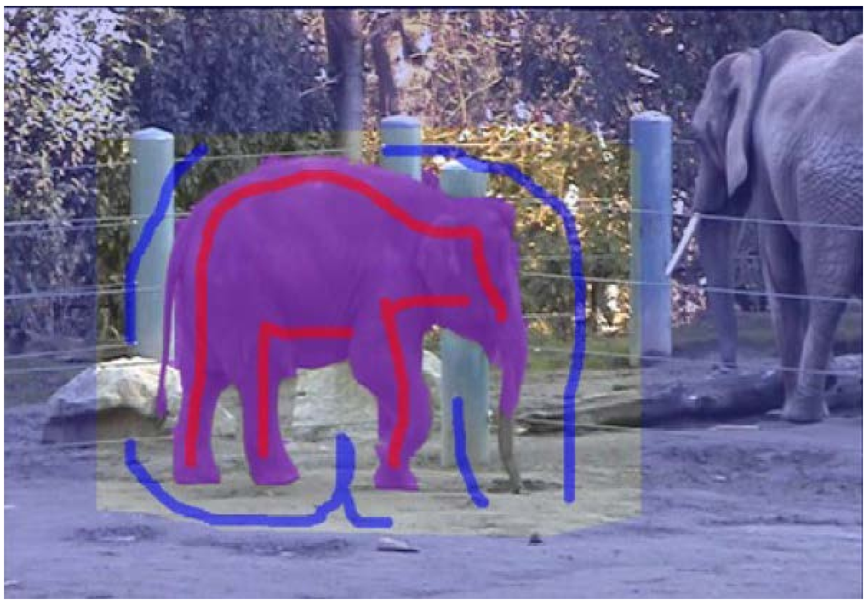
$C'(x,y,0)$: "distance" from pixel to background pixels

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

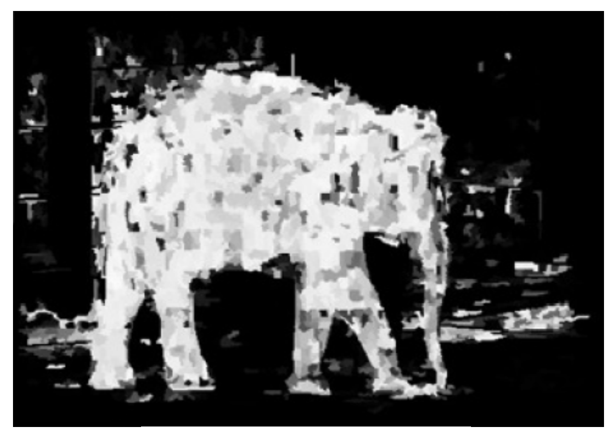
usually computed by creating a color model from user-labeled pixels

source: Noah Snavely

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



$C'(x, y, 0)$



$C'(x, y, 1)$

source: Noah Snavely

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

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

w_{pq} = 10.0

w_{pq} = 0.1

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

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

source: Noah Snavely

Solve with max flow / min cut

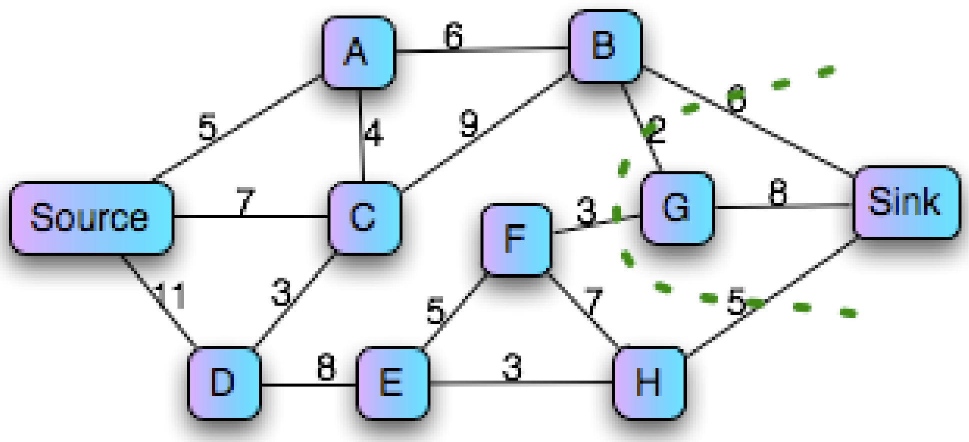
- Once all is in place

- We can solve

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

  – using max flow / min cut algorithm

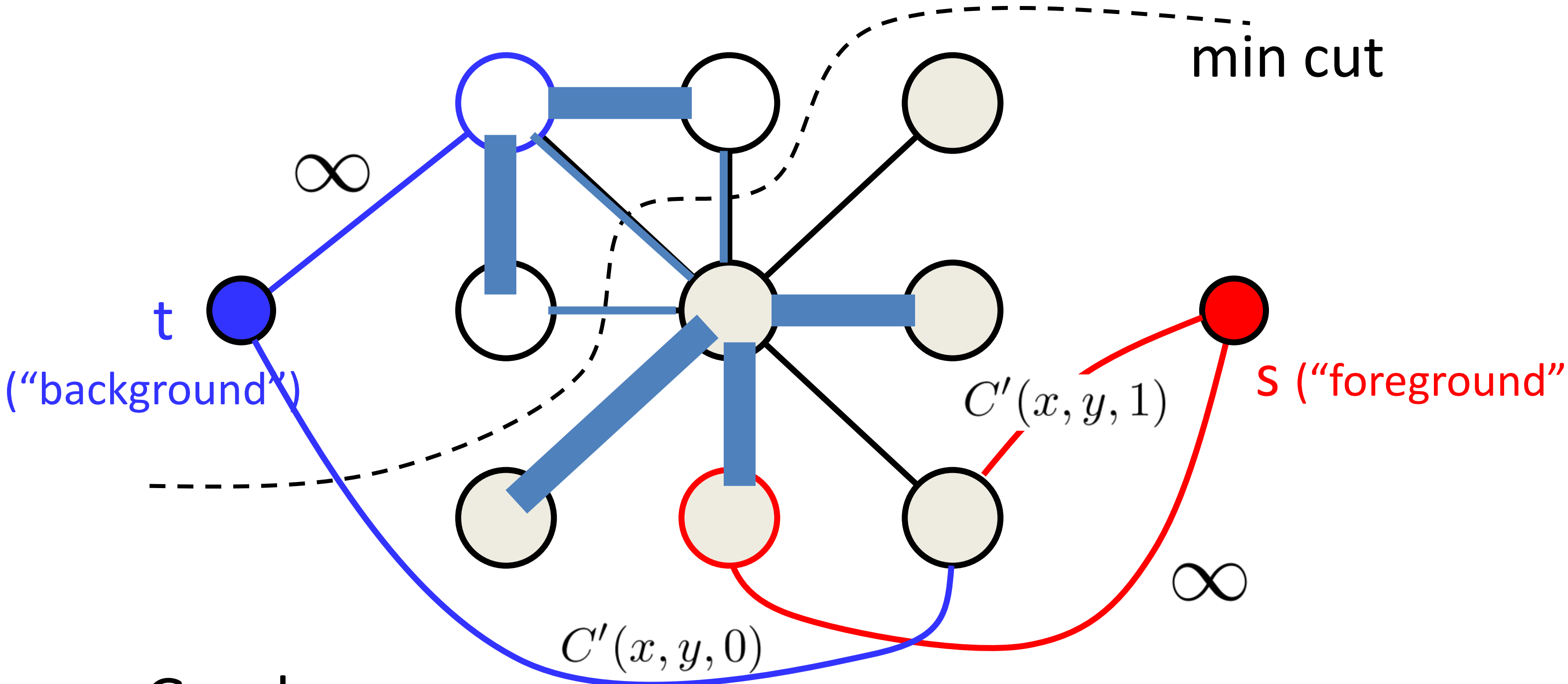
Graph min cut problem



- 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 $t \in T$

source: Noah Snavely

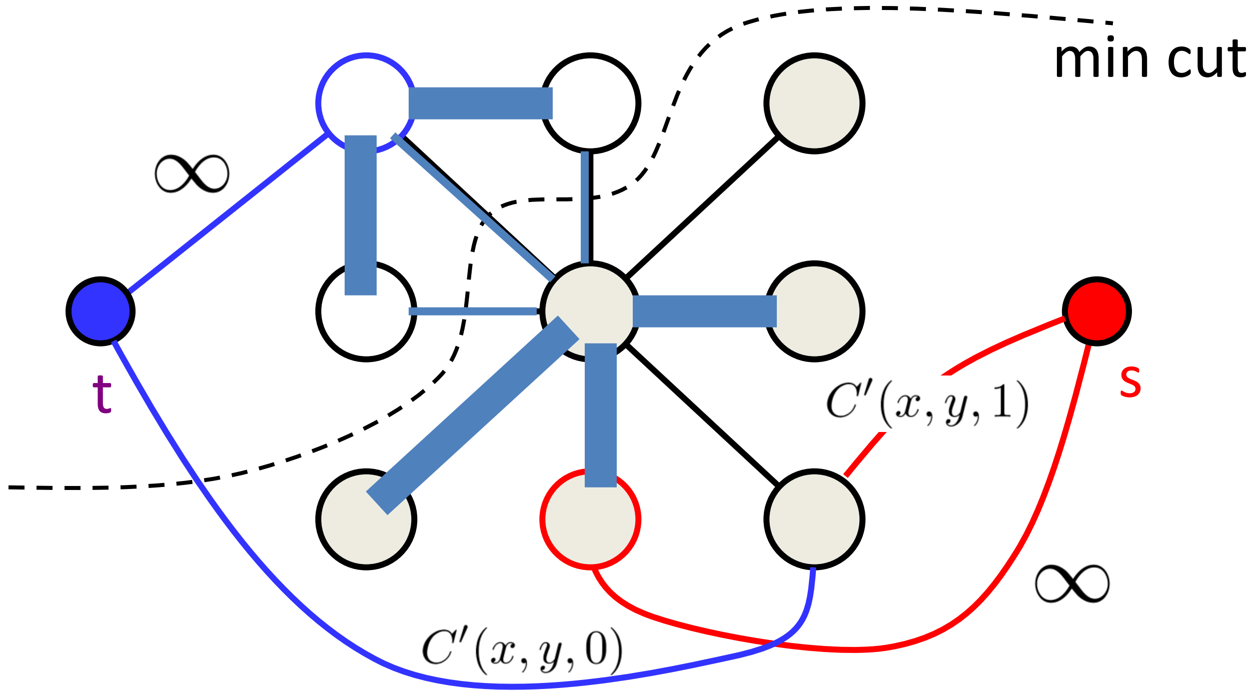
Segmentation by min cut



- 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]

source: Noah Snavely

Segmentation by min cut



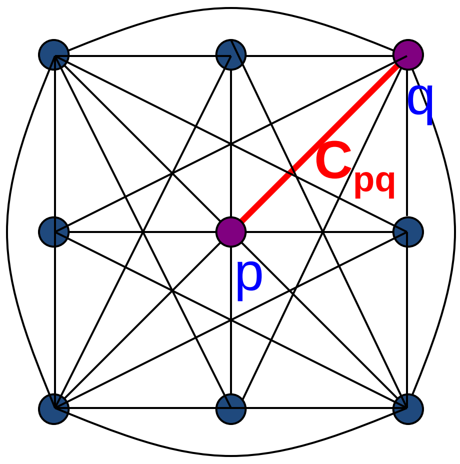
- The partitions *S* and *T* formed by the min cut give the optimal foreground and background segmentation
- i.e., the resulting labels minimize

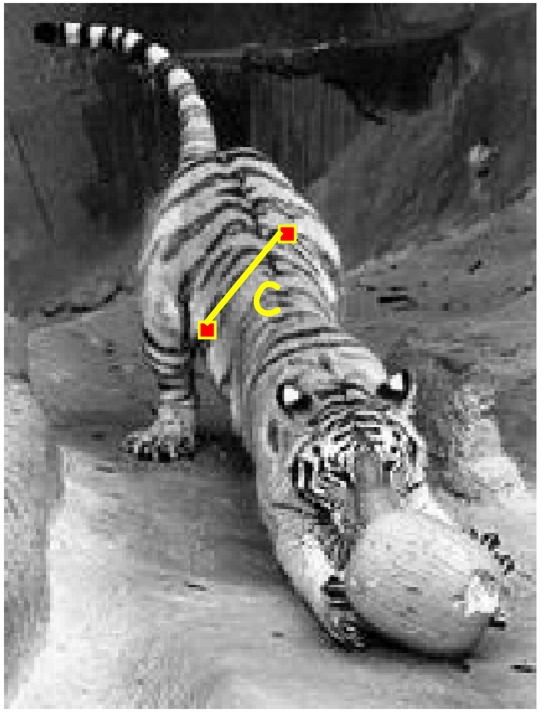
$$E(d) = E_d(d) + \lambda E_s(d)$$

Related / More info

- Grabcut
  - Rother et. al., SIGGRAPH 2004
- Intelligent Scissors
  - Mortensen et. al, SIGGRAPH 1995

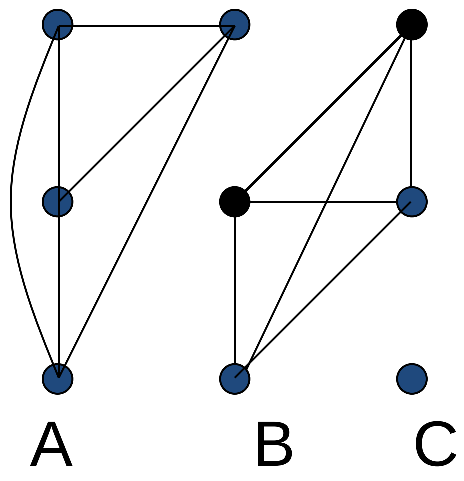
Fully Automatic [Shi and Malik]





- Fully connected graph
  - Node for every pixel
  - Edge between every pair of pixels
    - or every pair of sufficiently close pixels
    - Each edge is weighted by the similarity (affinity) of the two nodes
    - Similarity is inversely proportional to difference in color and position

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

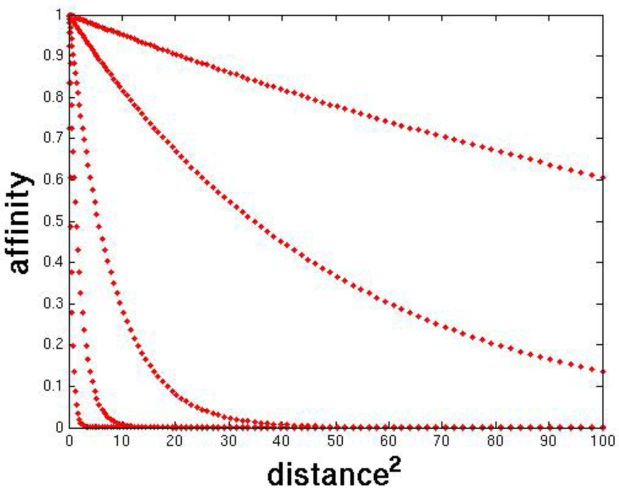
Measuring affinity

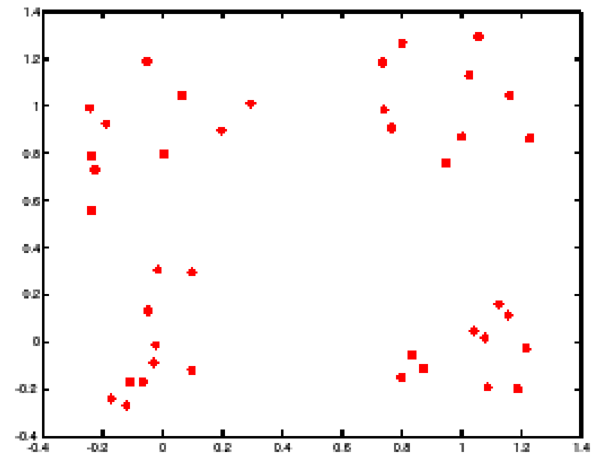
- Suppose we represent each pixel by a feature vector **x**, and define a distance function appropriate for this feature representation
- Then we can convert the distance between two feature vectors into an affinity with the help of a generalized Gaussian kernel:

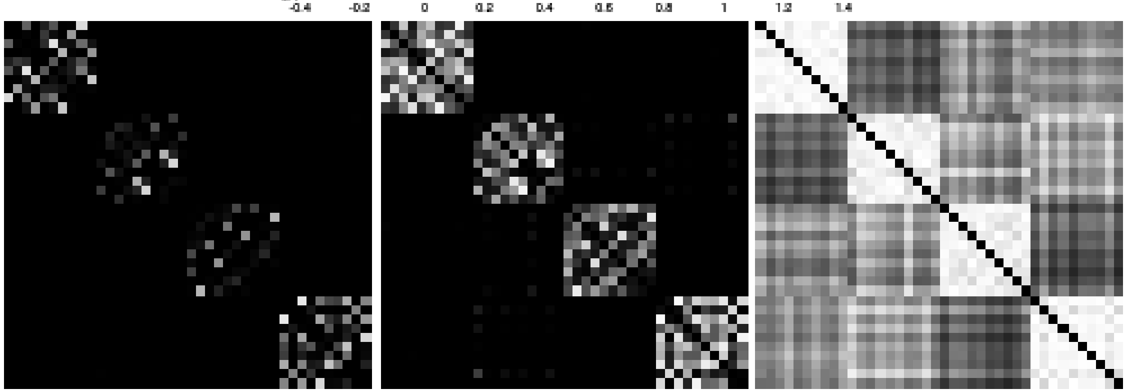
$$\exp\left(-\frac{1}{2\sigma^2}\mathrm{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Scale affects affinity

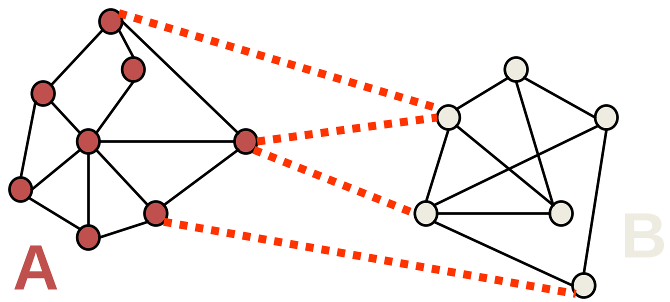
- Small σ: group only nearby points
- Large σ: group far-away points







Graph cut



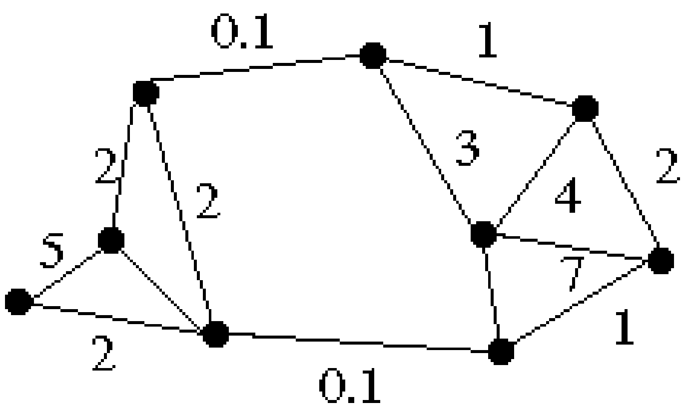
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges
- Any graph cut gives us a segmentation

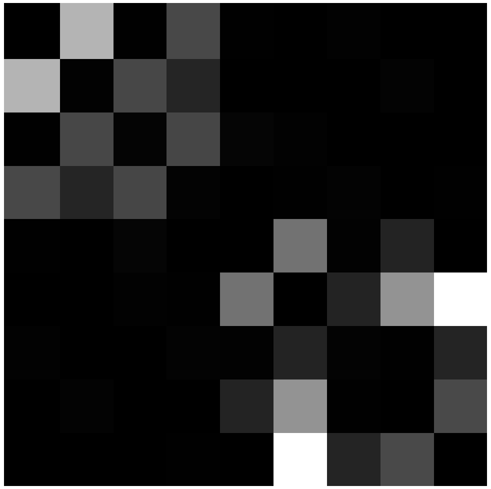
- **Find a minimum cut**
  - **Gives us a "good" segmentation**

Source: S. Seitz

Example: Minimum Cut

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

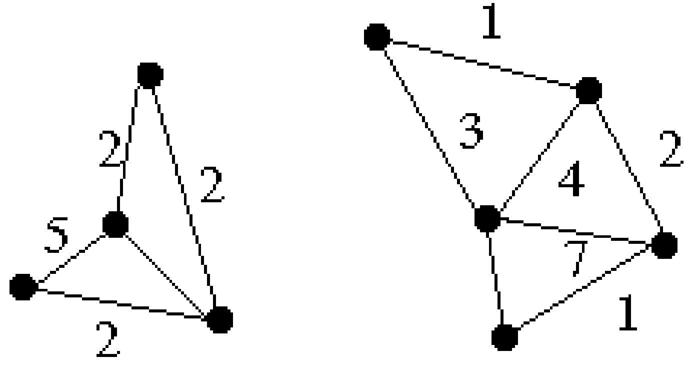


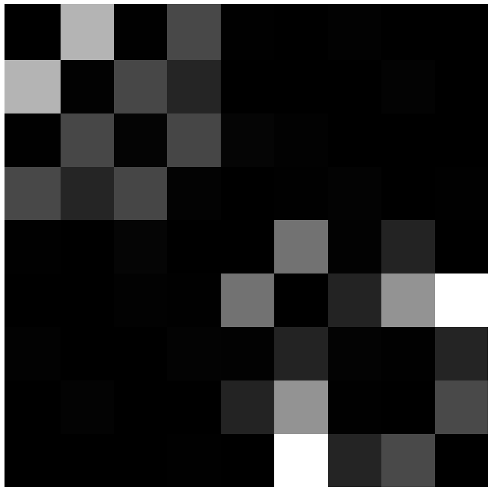


Minimum cut example

Example: Minimum Cut

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

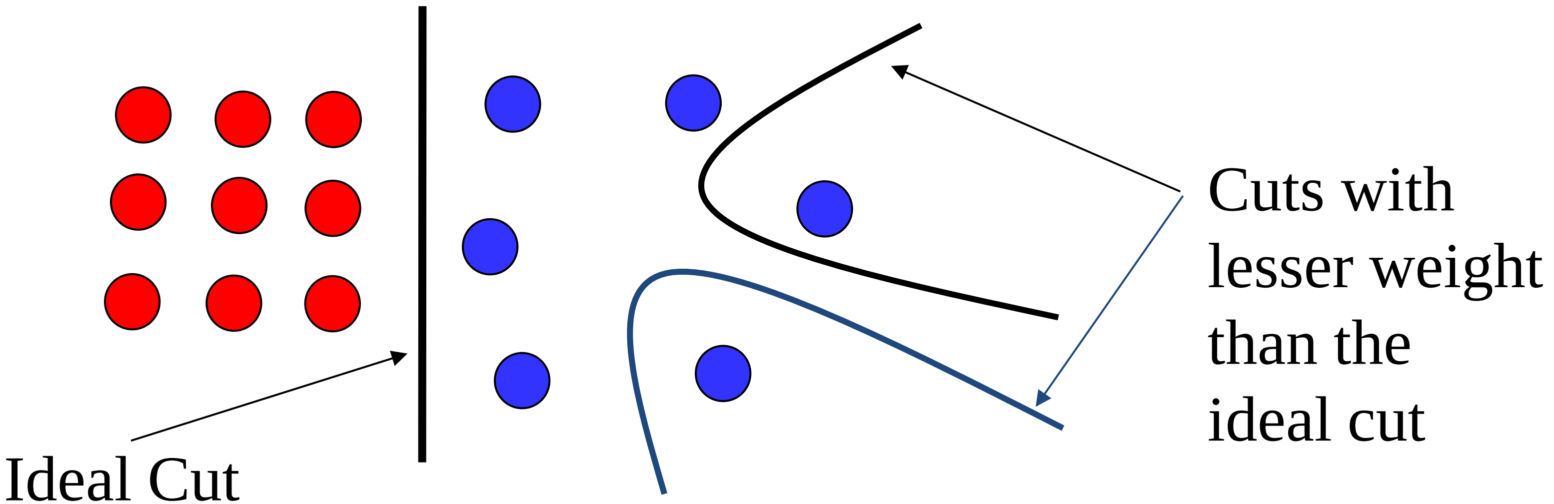




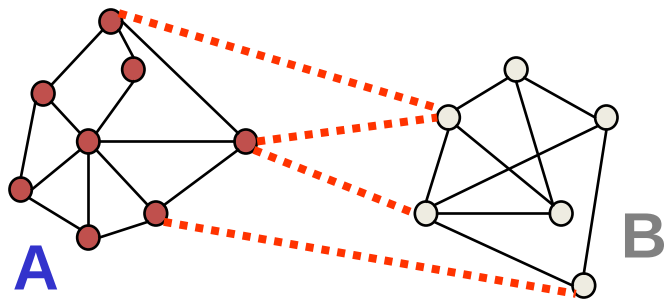
Minimum cut example

Normalized cut

- Drawback: minimum cut tends to cut off very small, isolated components



Normalized Cuts



Normalized Cut

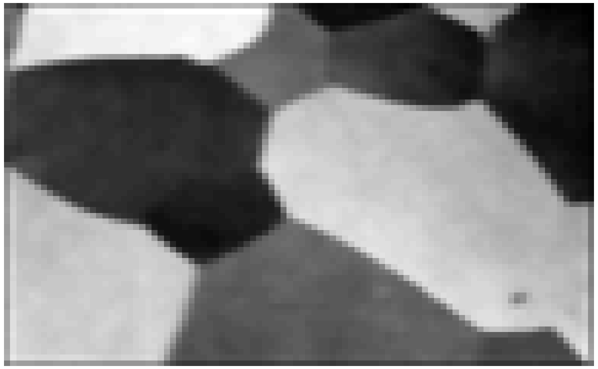
- a cut penalizes large segments
- fix by normalizing for size of segments

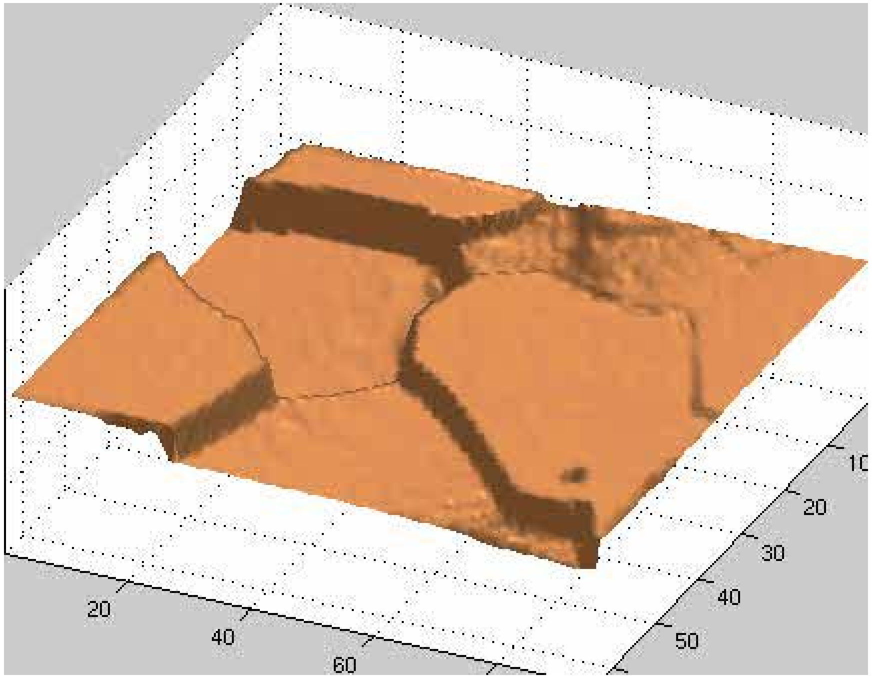
$$Ncut(A,B) = \frac{cut(A,B)}{volume(A)} + \frac{cut(A,B)}{volume(B)}$$

- volume(A) = sum of costs of all edges that touch A

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

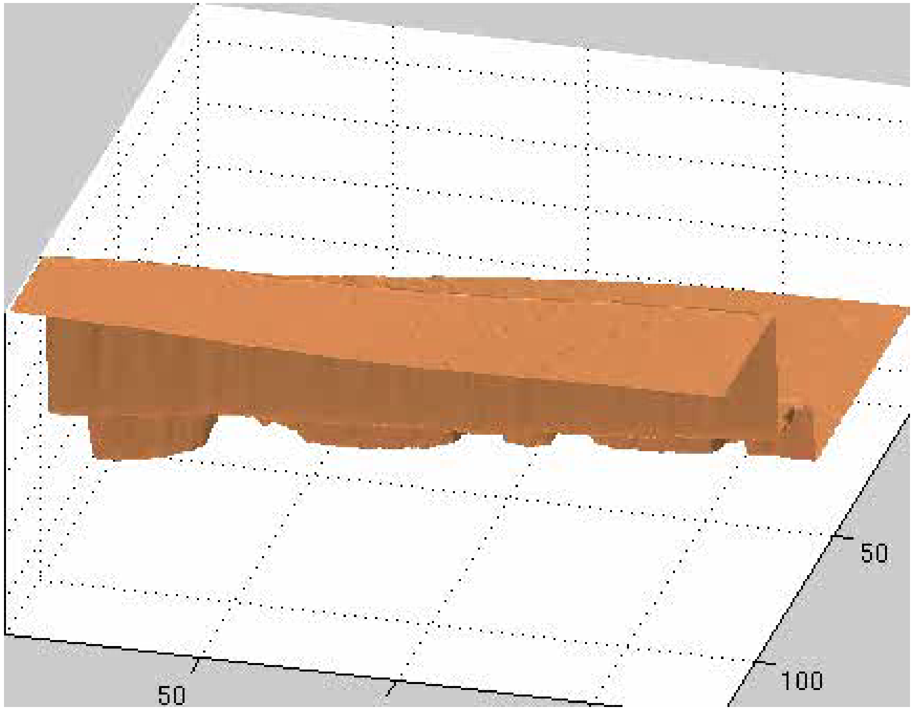
Interpretation as a Dynamical System





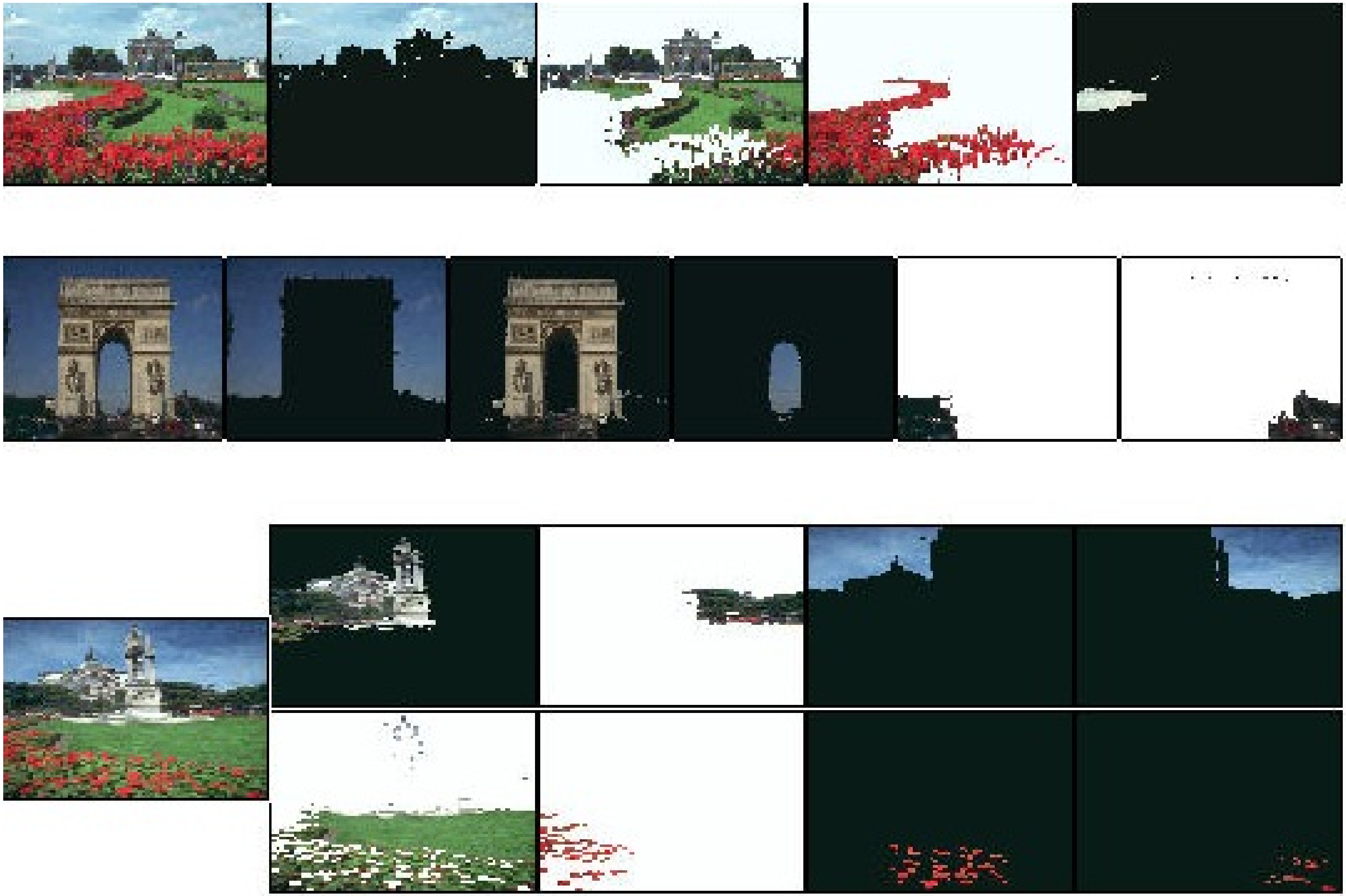
- Treat the links as springs and shake the system
  - elasticity proportional to cost
  - vibration "modes" correspond to segments
    - can compute these by solving an eigenvector problem
    - http://www.cis.upenn.edu/~jshi/papers/pami_ncut.pdf

Interpretation as a Dynamical System



- Treat the links as springs and shake the system
  - elasticity proportional to cost
  - vibration “modes” correspond to segments
    - can compute these by solving an eigenvector problem
    - http://www.cis.upenn.edu/~jshi/papers/pami_ncut.pdf

Color Image Segmentation



More Details: Normalized cut

- 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

$$\frac{y^T (D - W) y}{y^T D y}$$

where y is an indicator vector whose value should be 1 in the ith position if the ith feature point belongs to A and a negative constant otherwise

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

Normalized cut

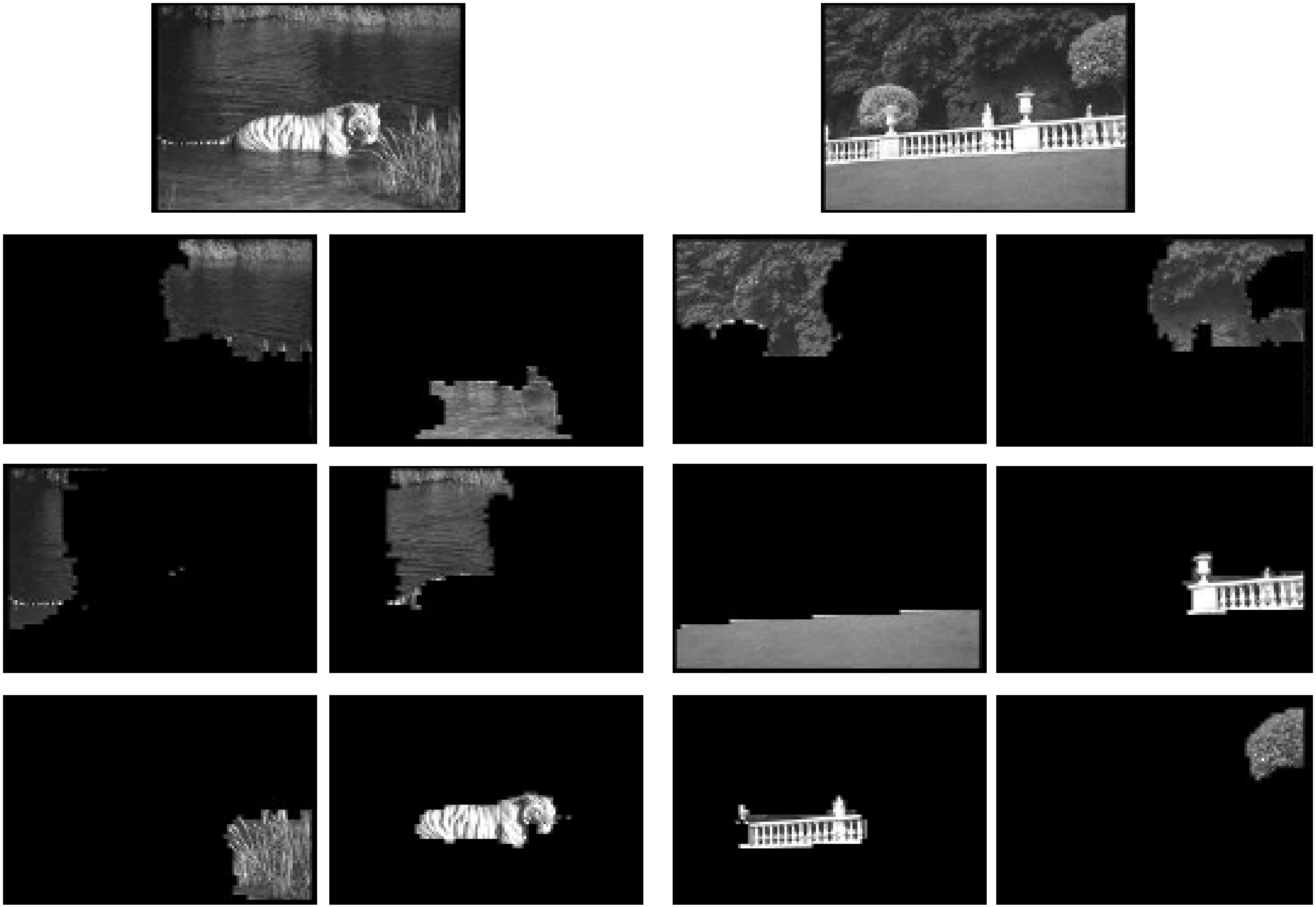
- Finding the exact minimum of the normalized cut cost is NP-complete, but if we *relax* y to take on arbitrary values, then we can minimize the relaxed cost by solving the *generalized eigenvalue problem* $(D - W)y = \lambda Dy$
- The solution y is given by the generalized eigenvector corresponding to the second smallest eigenvalue
- Intutitively, the ith entry of y can be viewed as a "soft" indication of the component membership of the ith feature
  - Can use 0 or median value of the entries as the splitting point (threshold), or find threshold that minimizes the Ncut cost

Normalized cut algorithm

1. Represent the image as a weighted graph $G = (V,E)$, compute the weight of each edge, and summarize the information in D and W
2. Solve $(D - W)y = \lambda Dy$ for the eigenvector with the second smallest eigenvalue
3. Use the entries of the eigenvector to bipartition the graph

- To find more than two clusters:
- Recursively bipartition the graph
- Run k-means clustering on values of several eigenvectors

Example result



Questions?

- Beyond D2L
  - Examples and information can be found online at:
    - *http://docdingle.com/teaching/cs.html*

    - *Continue to more stuff as needed*

Extra Reference Stuff Follows

Credits

- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing,* Gonzalez and Woods

- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Sventlana Lazebnik, UNC, 2010
  - Noah Snavely, Cornell University, 2012
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015