Image Segmentation (in very general terms!):

- Partitioning the image into salient regions

- Salient regions (at least for human observers) tend to represent individual objects, object parts, or individual surfaces

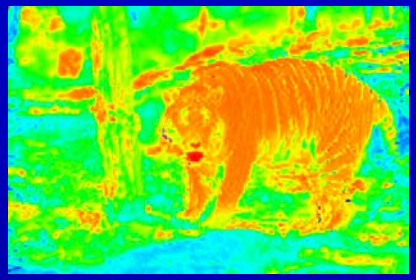
- But note: Humans use all sorts of tricks to perform this task!

Could we segment this image[2] without knowing what it is we're Looking at?

2- Prof. Richard L. Gregory's famous Dalmatian Image

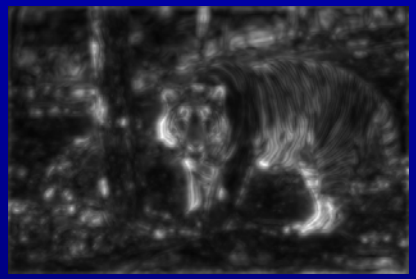
Similarity Measures. What cues are at hand?

- Image brightness



- Colour

- Edge energy



- Texture

- Image Position (spatial info)

- Motion

- Stereo disparity

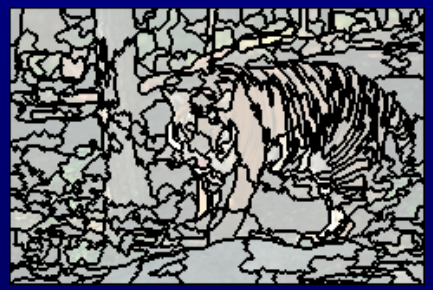
And now, the trick is deciding how to combine these cues.
For some insight see Malik et al. [MBLS01]

Brief aside: over- and under-segmentation

Two common terms used to describe general problems with segmentations[4]

Undersegmentation
(Mean-Shift)

Good segmentation
(human)

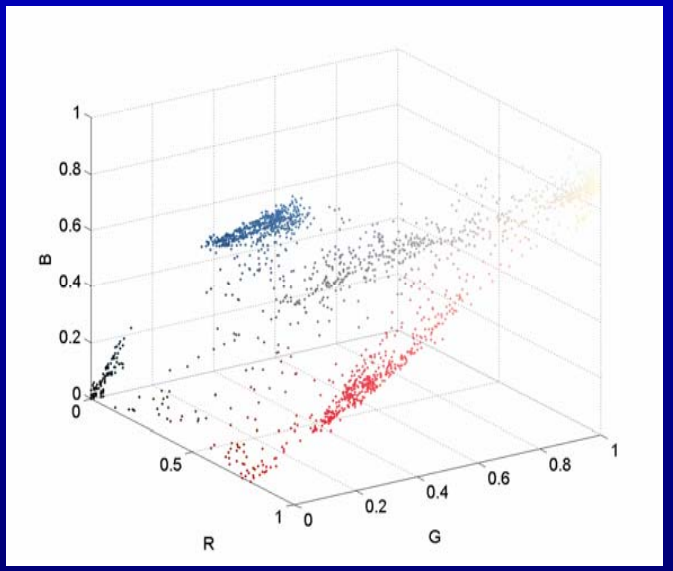


Oversegmentation
(Mean-Shift)

4 - Notice that under- and over segmentation are related, but not identical to the problem of deciding how much detail should be in a segmentation

Segmenting Images... feature space methods

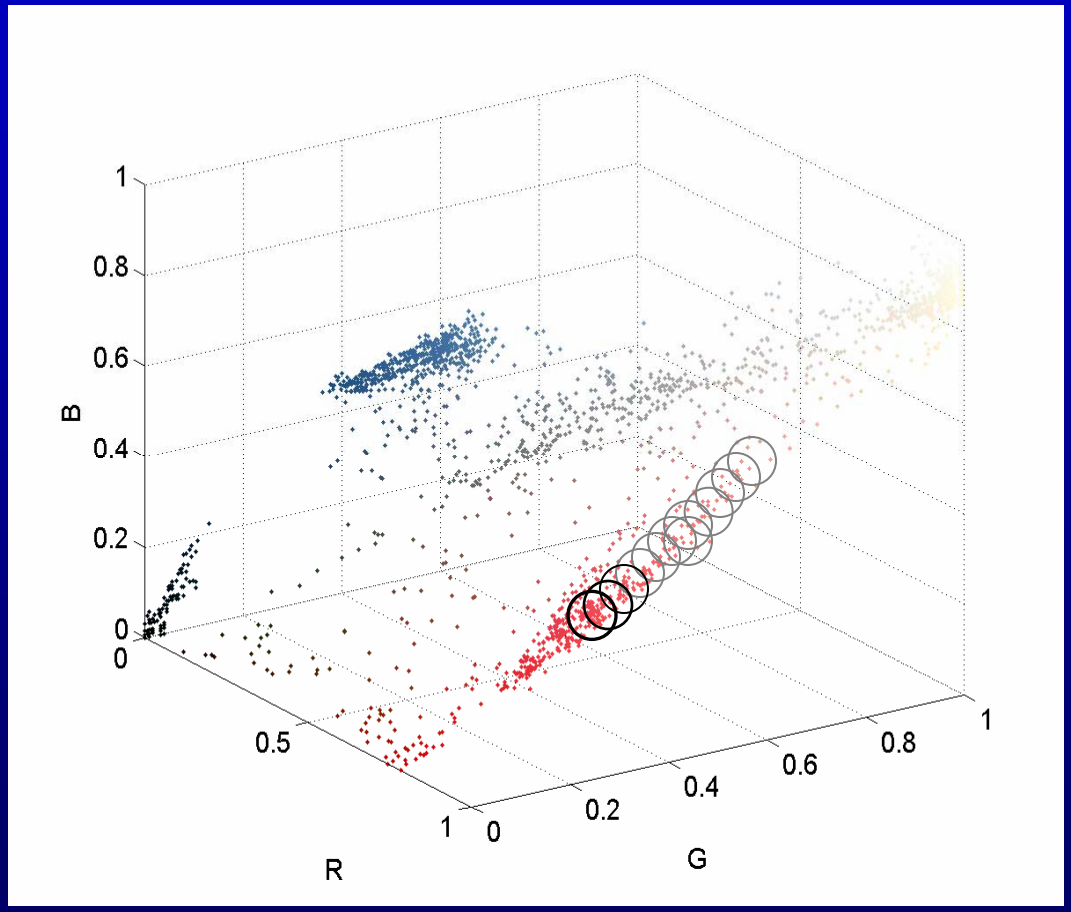
- Quick example, let's look at an RGB feature space for a simple image:



- The feature space has dimension 3. Notice that the principal image regions generate dense clusters in feature space.

Segmenting Images… feature space methods

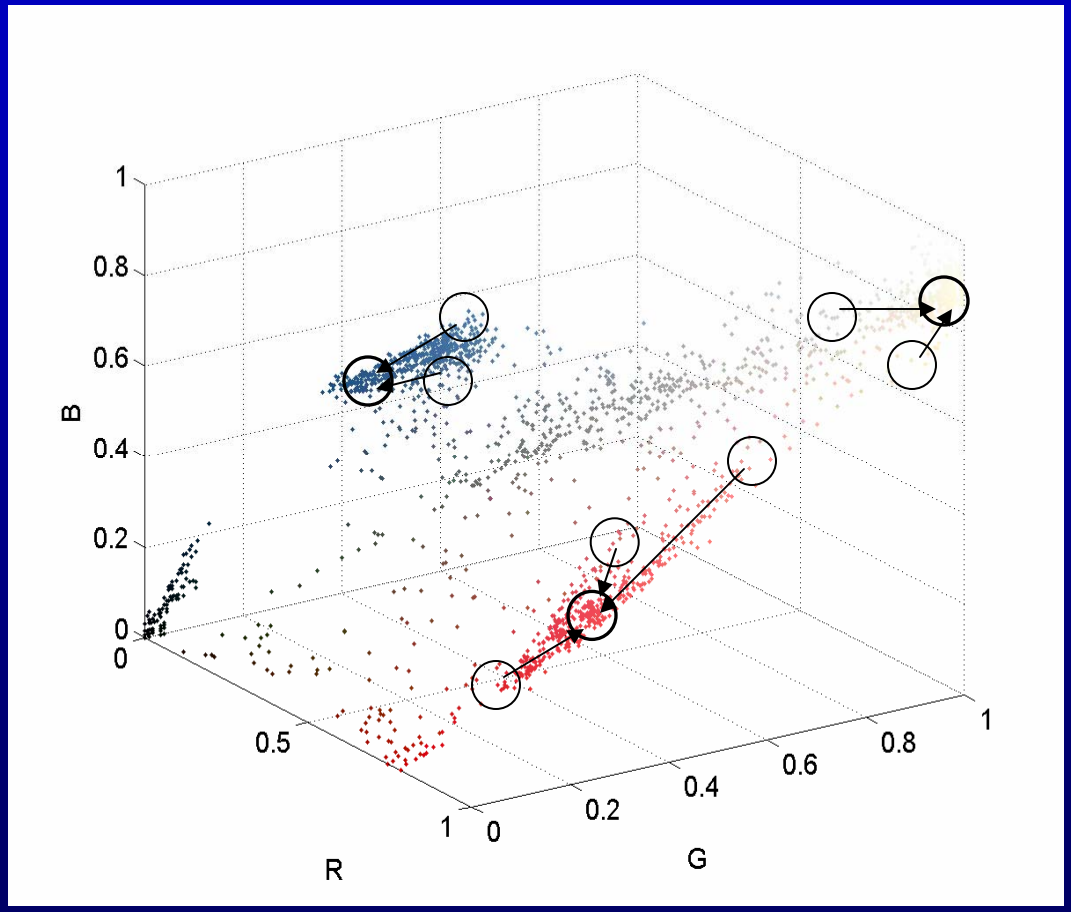
Mean-Shift algorithm



- The mean-shift iteration converges to a region of locally maximal density in feature space.

Segmenting Images… feature space methods

Mean-Shift algorithm



- A point of locally maximal density forms a basin of attraction for nearby feature vectors[6]

- These *domains of convergence* define the segments produced by mean-shift

6 – But notice that the range of attraction of this basin depends on the size and shape of the search window

Segmenting Images... feature space methods

Mean-Shift algorithm

- Sample segmentations[7]





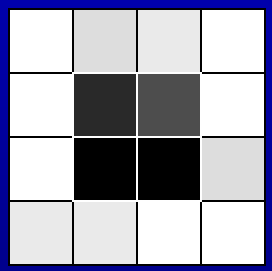

Input image | smaller search window | larger search window

7 – Using the EDISON system, see [EDISON]

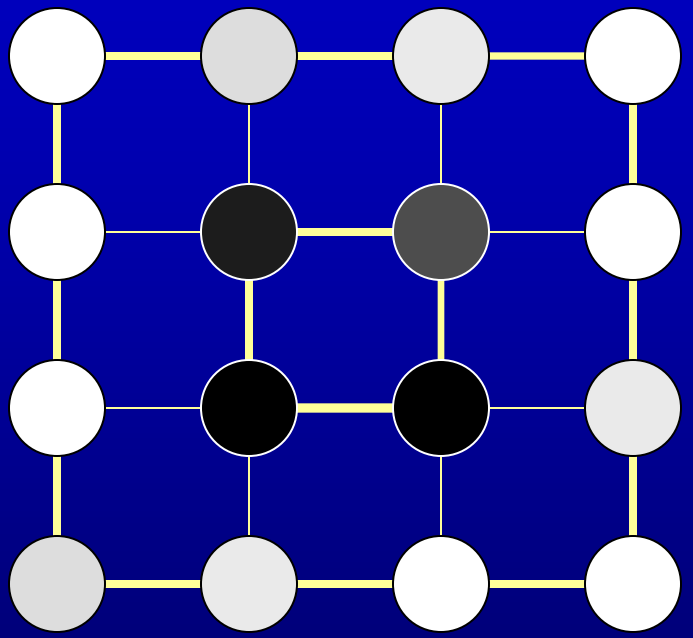
Segmenting Images... graph-theoretic methods

Images as graphs...

- An image I(x,y) is equivalent to a graph G(V,E)



Original image I(x,y)



Graph G(V,E)

- V is a set of *vertices* or *nodes*, each node represents one image element (e.g. individual pixels)

- E is a set of *edges* linking neighboring nodes together. The *weight* or *strength* of the edge is proportional to the similarity between the vertices it joins together.

Segmenting Images... graph-theoretic methods

Images as graphs...

- The image elements can be individual pixels, small regions, or other types of image features.

- Usually, only elements within a small neighborhood are connected. This provides spatial coherence and has computational advantages.

- Edge weight is also called pairwise *similarity*, or pairwise *affinity*.

- In general, given a graph G(V,E) graph-based segmentation methods attempt to find groups of nodes in G that are strongly connected to one another, but weakly connected to the rest of the graph.

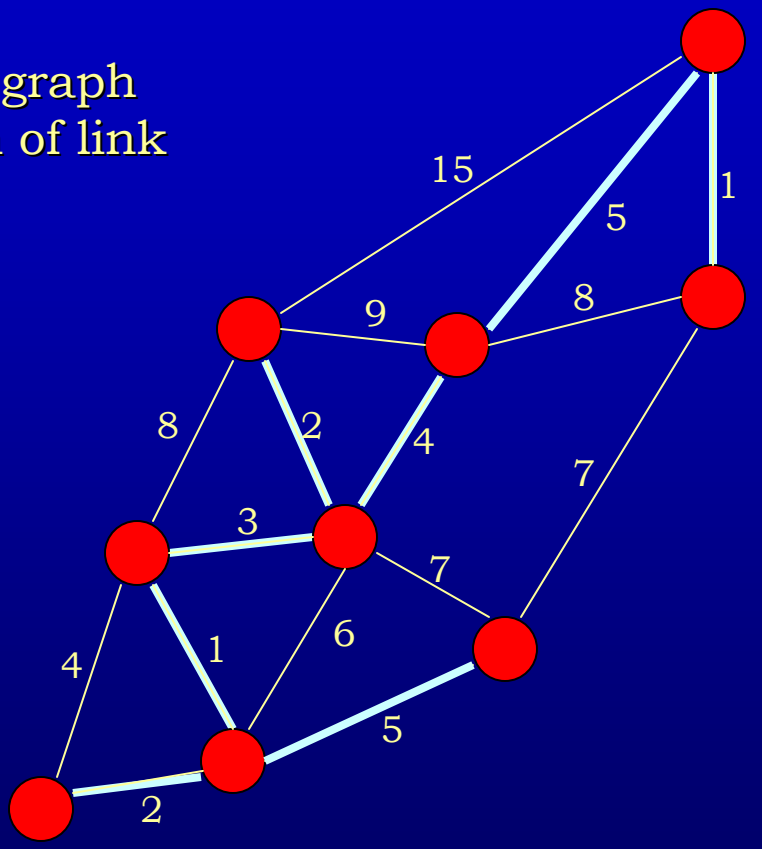
Segmenting Images… graph-theoretic methods

Minimal spanning trees (MSTs)

- Tree that connects all vertices of the graph with the minimum total weight (sum of link weight in the tree)

- Can be constructed efficiently using Kruskal's algorithm:

  - Start with a disconnected graph
  - Add edges in increasing order of weight as long as doing so doesn't introduce a cycle
  - Stop when all the vertices are connected



— Minimal Spanning Tree for this graph

Segmenting Images… graph-theoretic methods

The Local Variation algorithm (see Felzenszwalb & Huttenlocher [FH04])

- Partition the image so that for any pair of regions the variation between the regions should be larger than the variation within the regions.

- Extension of Kruskal's algorithm[8]:

  - Add edges one at a time in order of increasing weight. Maintain a list of disconnected MSTs

  - For each MST C_i compute a threshold $T(C_i) = w(C_i) + k/|C_i|$, where $w(C_i)$ is the maximum weight in the spanning tree, $|C_i|$ is the number of pixels in C_i, and $k>0$ is a user-defined constant)

  - If the next edge to be added joins two separate MSTs, the MSTs are merged only if

$$w(\vec{x}_k, \vec{x}_l) \leq \min(T(C_i), T(C_j))$$

8 – See also Jepson [Jep04] for more details

Segmenting Images... graph-theoretic methods

The Local Variation algorithm

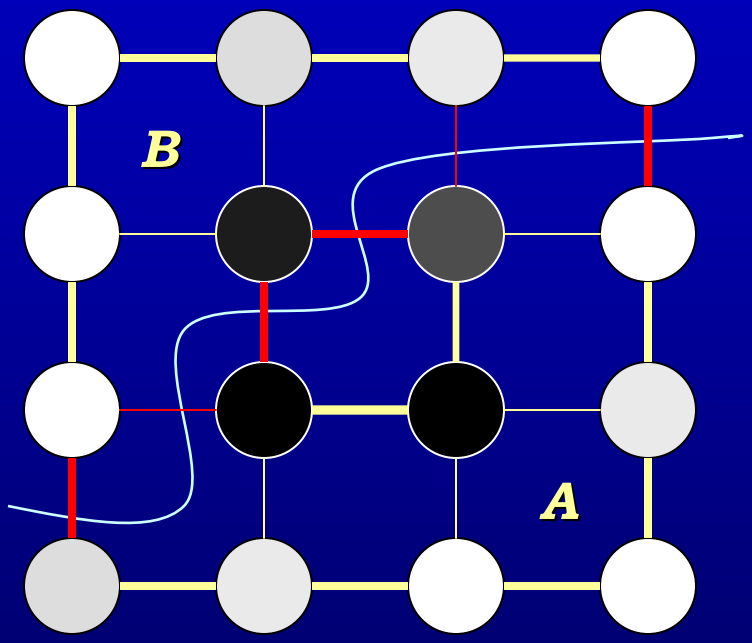
- Sample segmentations[9]

Input image | smaller k | larger k

9 – Generated using the implementation provided by the authors of [FH04], see [FH_code]

Segmenting Images… graph-theoretic methods

Graph Cuts



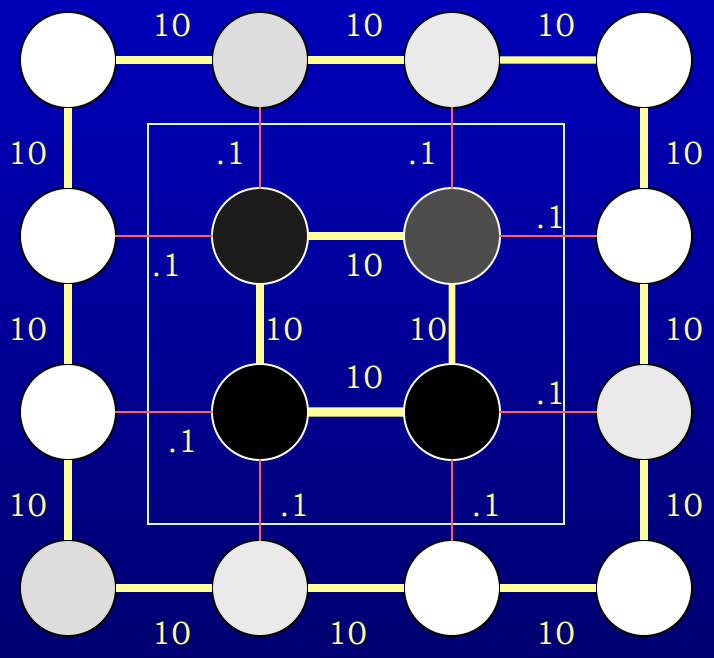
Graph G(V,E)

- A *cut* through a graph is defined as the total weight of the links that must be removed to divide the graph into two separate components.

$$cut(A,B) = \sum_{i \in A, j \in B} w_{i,j}$$

Segmenting Images… graph-theoretic methods

Minimum Cut method (see Wu & Leahy [WL93])



$$MinCut(A,B) = 8*(.1) = .8$$

- Find the cut through the graph that has the overall minimum weight

$$MinCut(A,B) = \min_{A,B}(cut(A,B))$$

- Should correspond to the subset of edges of least weight that can be removed to partition the graph

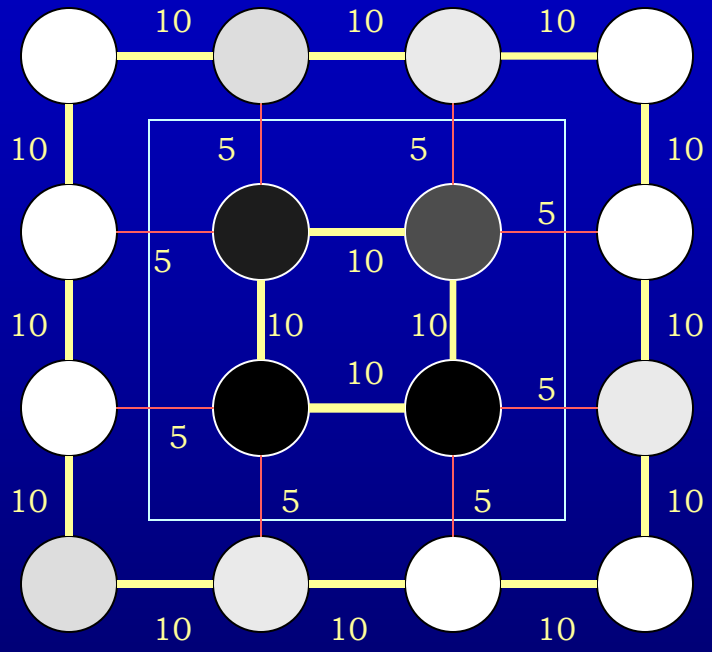
- Since weight encodes similarity, this should be equivalent to partitioning the graph along the boundary of least similarity

Segmenting Images... graph-theoretic methods

Minimum Cut method

- Can be computed efficiently

- However, it has a preference for *short* boundaries. Sometimes picks a trivial partition



$$cut(A,B) = 8*(5) = 40$$

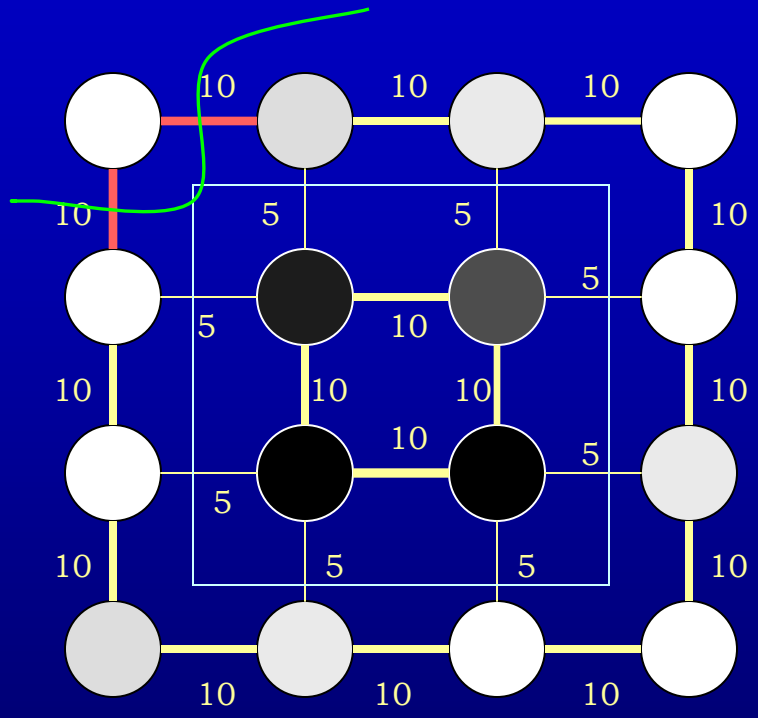
Segmenting Images… graph-theoretic methods

Minimum Cut method

- Can be computed efficiently

- However, it has a preference for *short* boundaries. Sometimes picks a trivial partition

- We have to somehow constrain the solution to avoid such trivial cuts



$$cut(A,B) = 8*(5) = 40$$

$$MinCut(A,B) = 2*(10) = 20$$

Segmenting Images... graph-theoretic methods

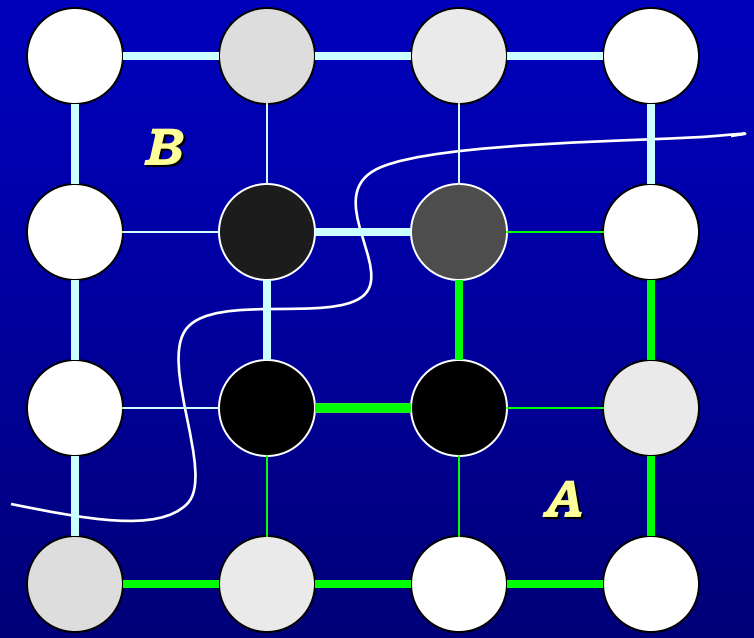
Normalized Cuts (see Shi & Malik [SM00])

- Find the cut through the graph that minimizes the normalized cut measure

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

- Here the term $assoc(A,V)$ is the total weight of the connections between the region A and the rest of the nodes in the graph

$$assoc(A,V) = \sum_{i \in A, j \in V, (v_i, v_j) \in E} w_{i,j}$$



Graph G(V,E)

Segmenting Images… graph-theoretic methods

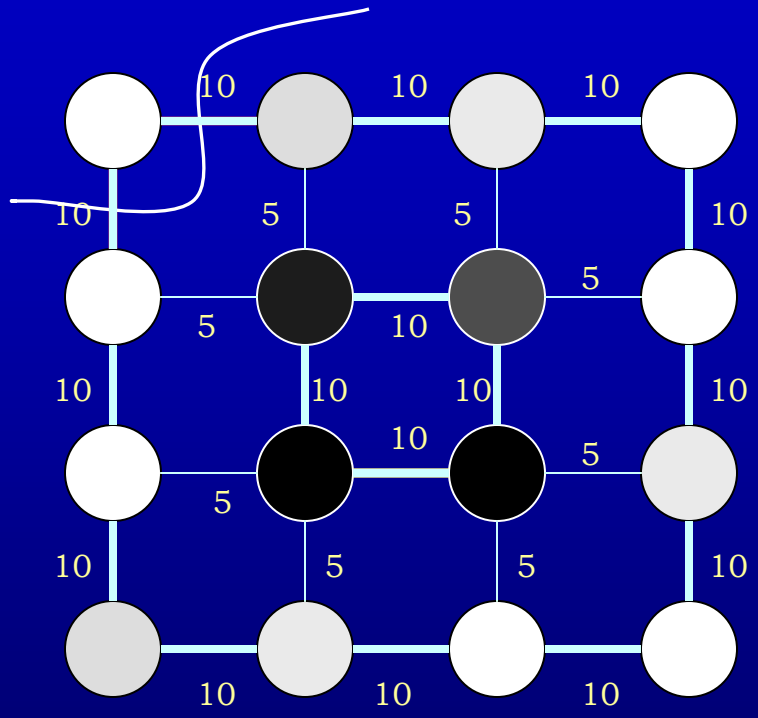
Normalized Cuts

$$cut(A,B) = 2*(10) = 20$$

$$assoc(A,V) = 2*(10) = 20$$

$$assoc(B,V) = 16*10 + 8*5 = 200$$

$$NCut(A,B) = \frac{20}{20} + \frac{20}{200} = 1.1$$



- Trivial cuts such as this one don't minimize the normalized cut

Segmenting Images... graph-theoretic methods

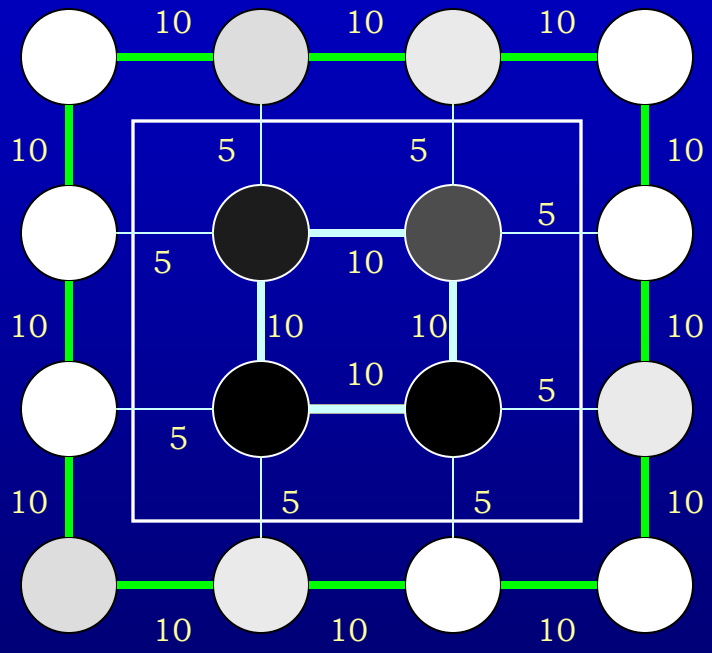
Normalized Cuts

$$cut(A,B) = 8*(5) = 40$$

$$assoc(A,V) = 12*10+8*4 = 160$$

$$assoc(B,V) = 4*10+8*5 = 80$$

$$NCut(A,B) = \frac{40}{160} + \frac{40}{80} = .75$$



- The normalization by *assoc(R,V)* should (in general) prevent the bias toward short boundaries

Segmenting Images… graph-theoretic methods

Normalized Cuts

- Solving for the optimal *NCut* exactly is NP-complete

- However, we can obtain an approximate solution. Start with an *affinity matrix W* s.t. *W(i,j)* contains the weight of the edge linking nodes *i* and *j*

- Notice that if we have an image (n x m) pixels in size, *W* will be an $(n*m)^2$ matrix. Fortunately, it is sparse

- Also, define a diagonal matrix *D* s.t. $D(i,i) = \sum_j w_{i,j}$

Segmenting Images… graph-theoretic methods

Normalized Cuts

- Shi and Malik show that an approximate solution is given by the eigenvector with the smallest eigenvalue of the system[10]

$$(D-W)\vec{y} = \lambda D\vec{y}$$

- Furthermore, succeeding eigenvectors can be used to split previous partitions

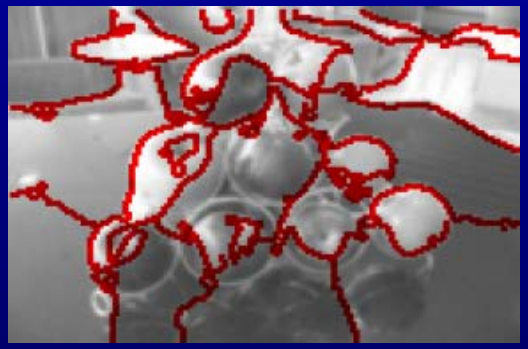
- General NCut procedure (there are several variants!)
  - Compute W and D
  - Solve the above system for the k eigenvectors with smallest eigenvalues
  - Threshold each eigenvector, from the 2^{nd} smallest on, to obtain a partition of the image
  - The segmentation is the intersection of the k partitions generated in this way

10 – Meila and Shi [MS01] present a probabilistic interpretation of NCuts in terms of random walks

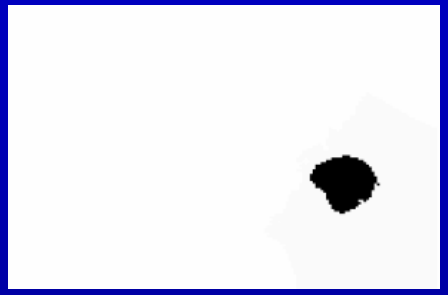
Segmenting Images… graph-theoretic methods

Normalized Cuts

Input image



Segmentation with 25 segments[11]



2nd eigenvector

3rd eigenvector

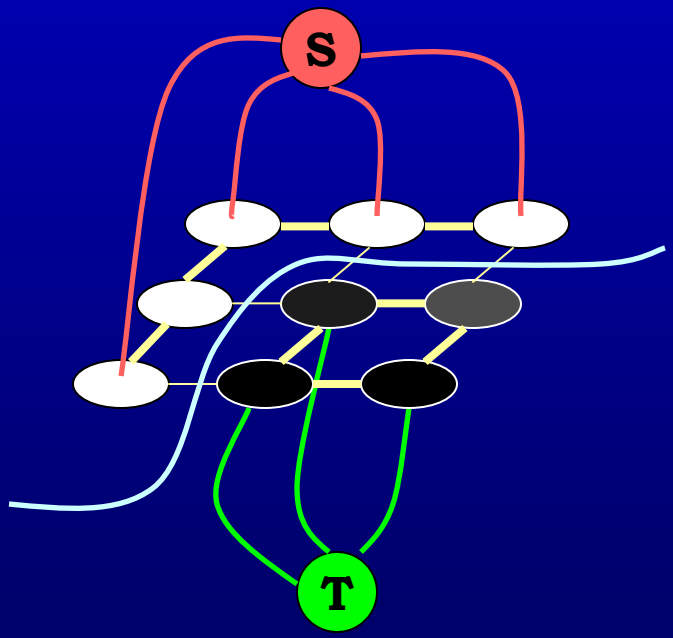
4th eigenvector

11 – Generated using the implementation provided by at [NCut_code]

Segmenting Images... graph-theoretic methods

S-T Min-Cut (see Boykov & Kolmogorov [BK01], and Boykov & Jolly [BJ01])

- Reduce the problem of trivial cuts by introducing two special nodes called *source (S)* and *sink (T)*



- *S* and *T* are linked to some image nodes by links of very large weight (so that they will never be selected in a cut)

- Find the minimum cut that separates the source from the sink

- Notice that the problem is deciding how to connect *S* and *T* to the image nodes

Segmenting Images… graph-theoretic methods

S-T Min-Cut

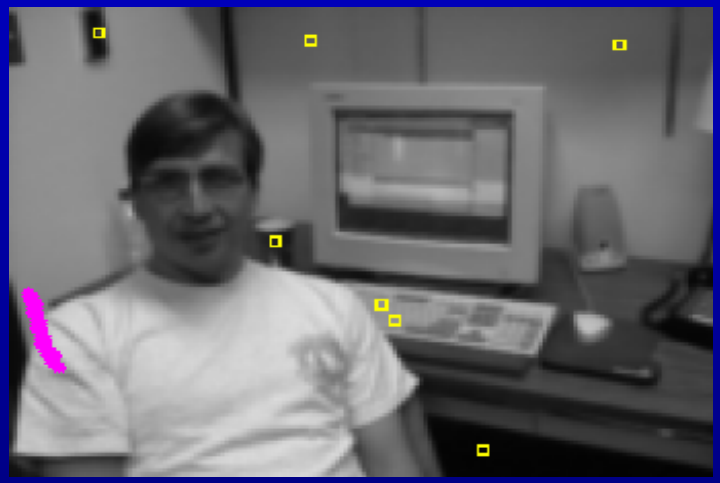
User-selected pixels connected to *S* (red) and pixels connected to *T* (cyan)

Minimum S-T cut

- If we have a good ‘guess’ (or a human observer) to tell us hot to link the source and sink to the image, we will get an optimal segmentation

Segmenting Images... graph-theoretic methods

S-T Min-Cut



Source pixels (purple) and sink pixels (yellow)

Minimum S-T cut

- With a bad guess, things don't go so well...

- And, how do we get this guess to begin with?

Segmenting Images... graph-theoretic methods

SE-MinCut (see Estrada, Jepson & Chennubhotla [EJC04])

- Use spectral clustering to identify small groups of similar pixels. These groups define *seed regions* for S-T MinCut

Input image

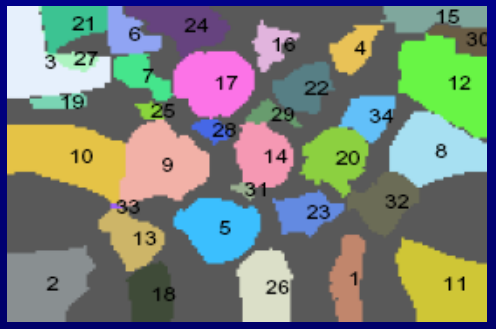
Seed regions found through spectral clustering. Combinations of these are used as source and sink regions for S-T MinCut

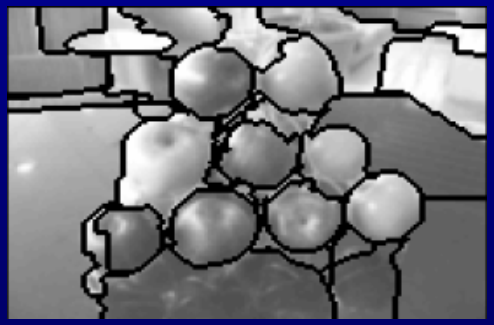
Final segmentation combines the partitions generated with each separate S-T cut.

Segmenting Images… graph-theoretic methods

SE-MinCut

- Sample segmentations



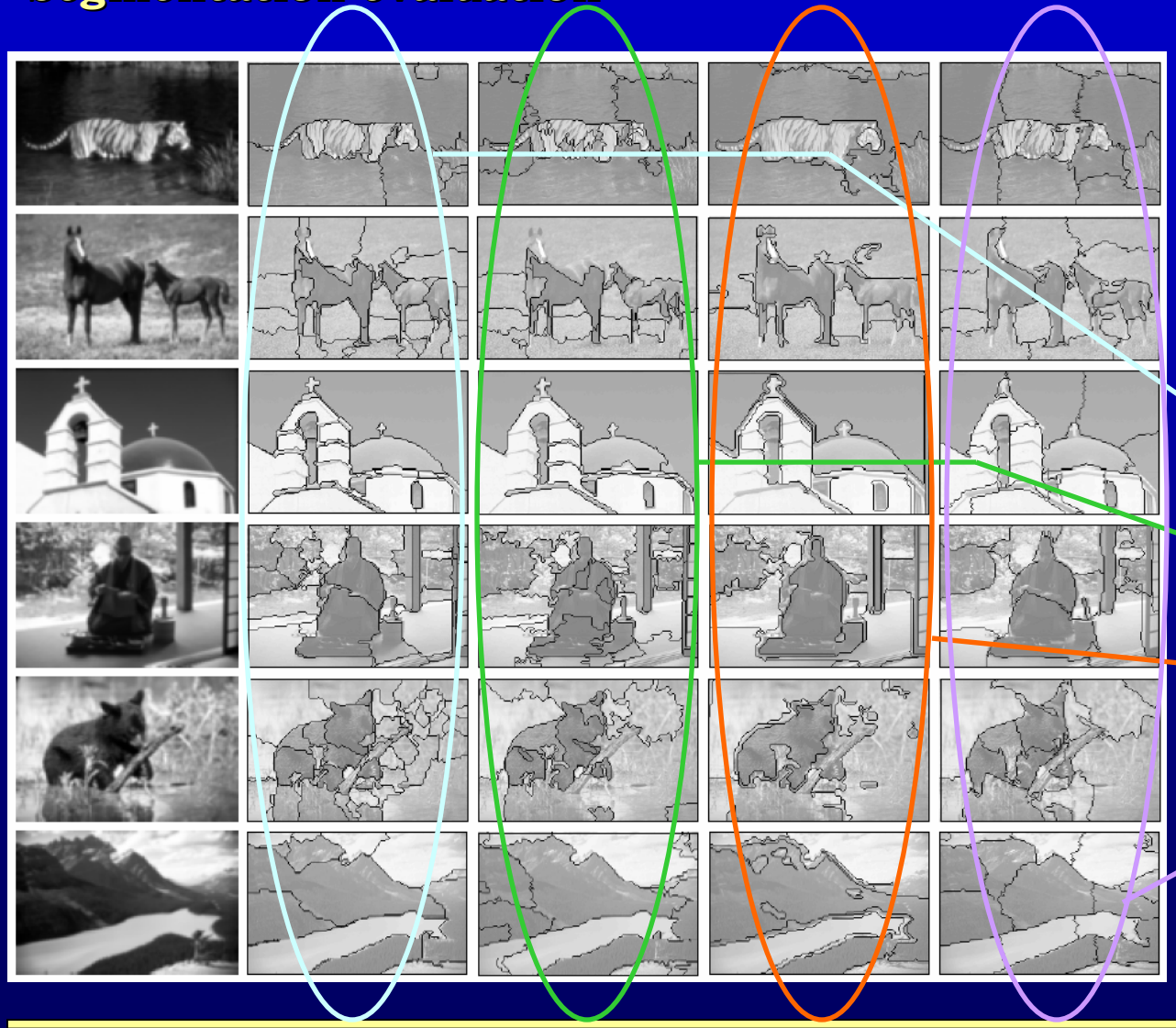


Input image

Seed regions

Segmentation

Segmentation evaluation



- Columns 2-5 show segmentations by different algorithms (each column corresponds to 1 method). Which segmentations are better?

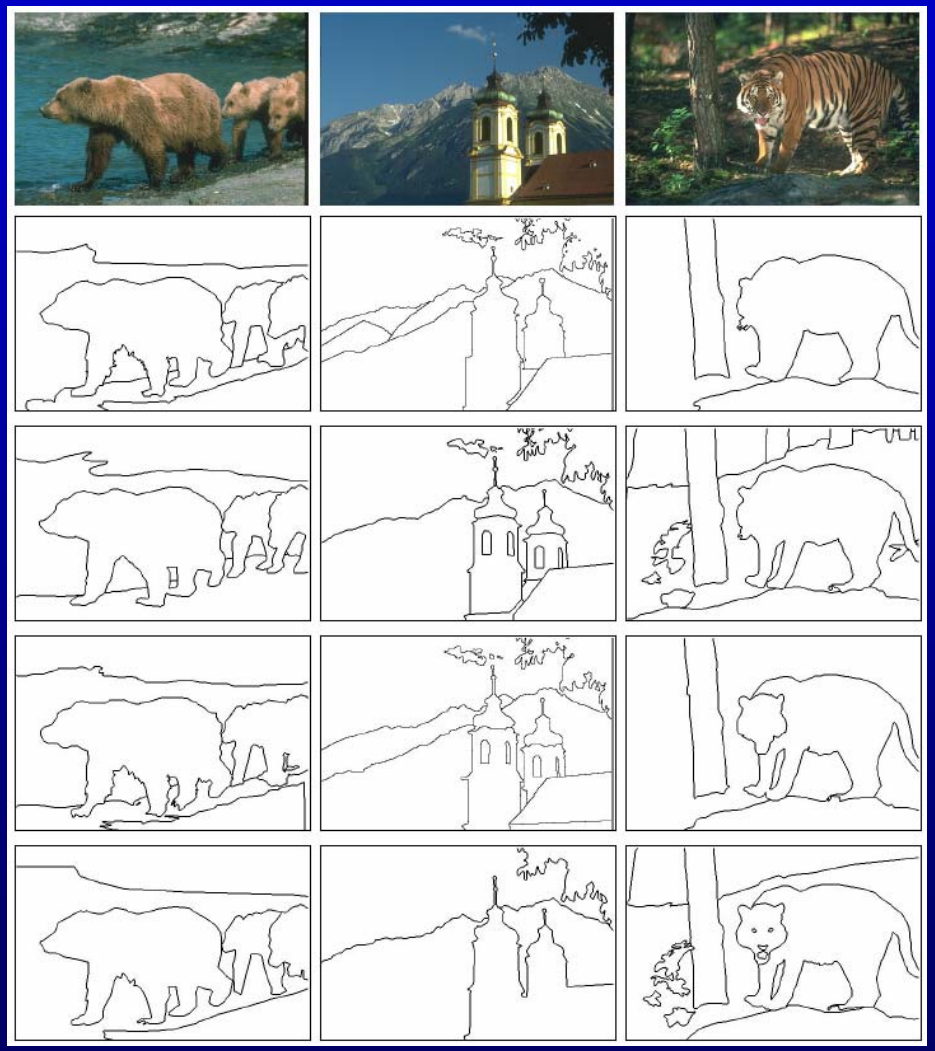
SE-MinCut

Mean-Shift

Local Variation

NCuts

Segmentation evaluation



- 200 training images, 100 testing images

- At least 4 human ground-truth segmentations per image

- Interesting observation: Human segmentations are consistent with one another.

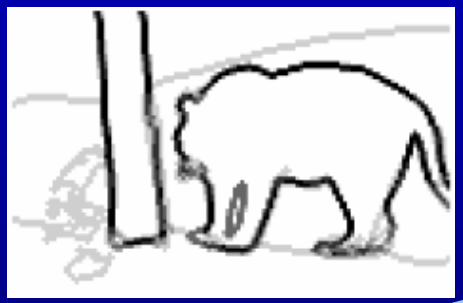
- Most variation concerns 'level of detail', notice that this is different from over- and under-segmentation

Segmentation evaluation

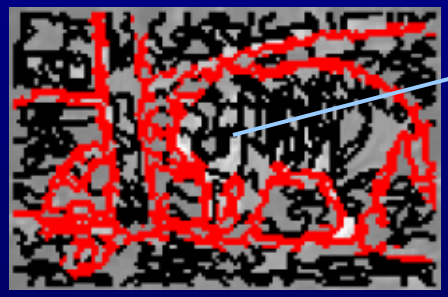
- Precision: Percentage of *detected* boundary pixels that correspond to *ground-truth* (human) boundary pixels

- Recall: Percentage of *ground-truth* (human) boundary pixels that were *detected* in the automatic segmentation

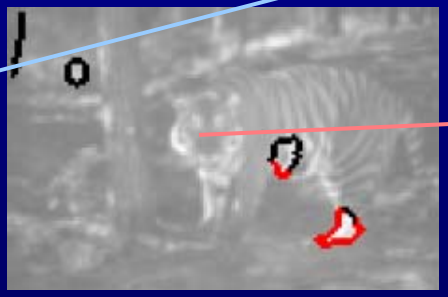
Original image



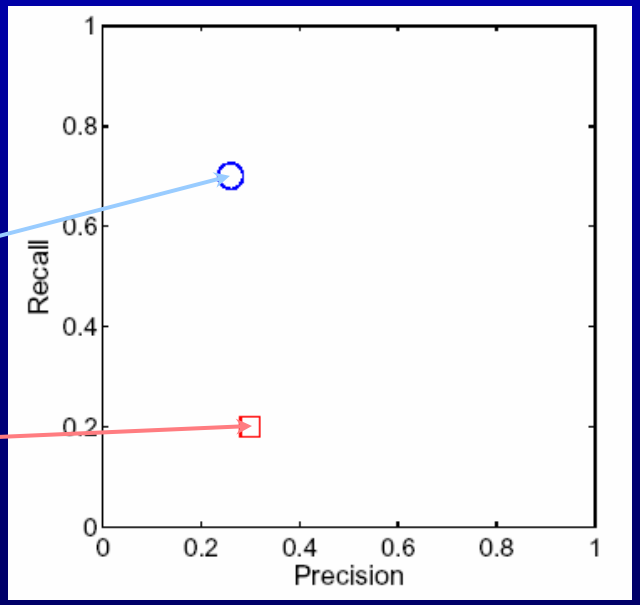
Overlayed human boundaries



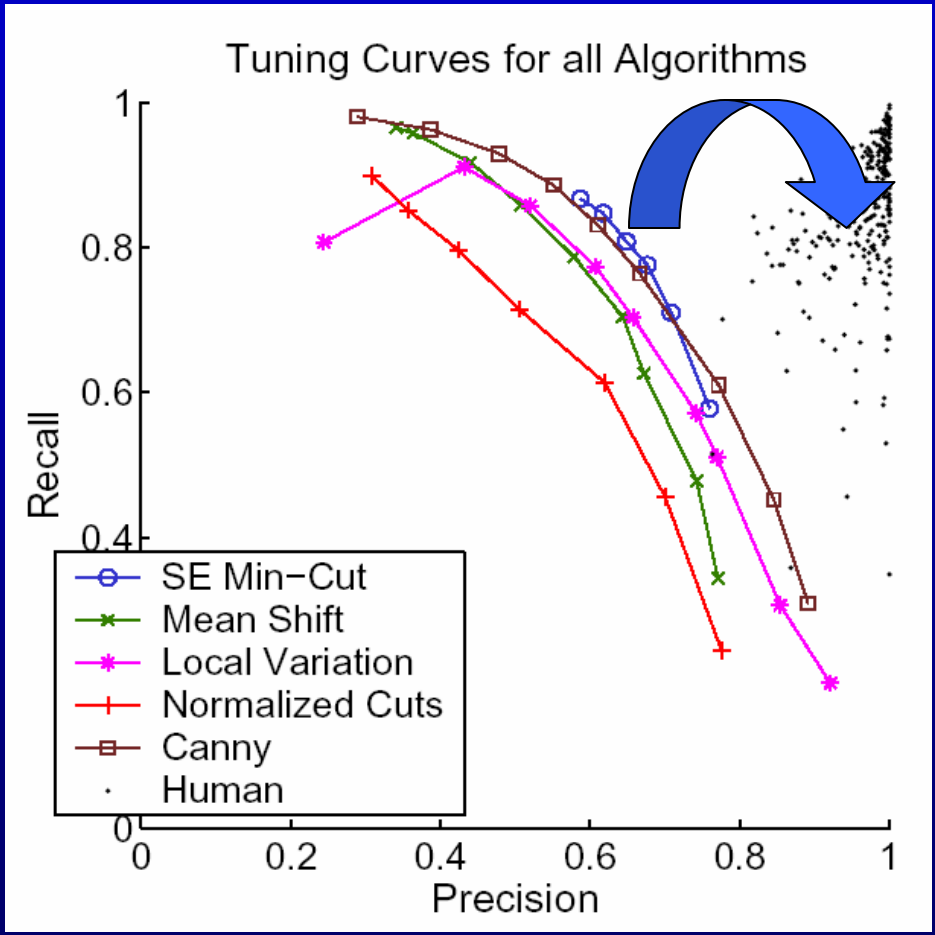
Mean-shift over-segmented



Mean-shift under-segmented



Segmentation evaluation



Tuning curves for several segmentation methods (see Estrada [Est05])

- Does this agree with our visual evaluation?

- What's the deal with Canny edges? Given these results, why even bother doing segmentation?

- What about this gap between human performance and automatic segmentation results? How do we get there?

Segmentation evaluation

- What about the gap in performance between human observers and automatic segmentation methods?

- It could be that we just don't know how to exploit available low-level cues appropriately

- Perhaps more likely, human observers use high-level knowledge to complete the segmentation task when low-level information is ambiguous or missing

- We should think about this: How much can we expect to be able to improve bottom-up segmentation methods? How good is good enough?

Segmentation... a few final thoughts

- Current segmentation algorithms seem reasonably capable

- We don't (yet) have many direct applications of segmentation, however it is quite popular in medical imaging

- Recent research has focused on the use of more mid-level information (i.e. shape priors, segmentation of known object classes, etc. see for example Kumar et al. [KTZ05])

- Ultimately, achieving human performance is likely to require a combination of bottom-up and top-down processing. We don't know how this should work at the present time