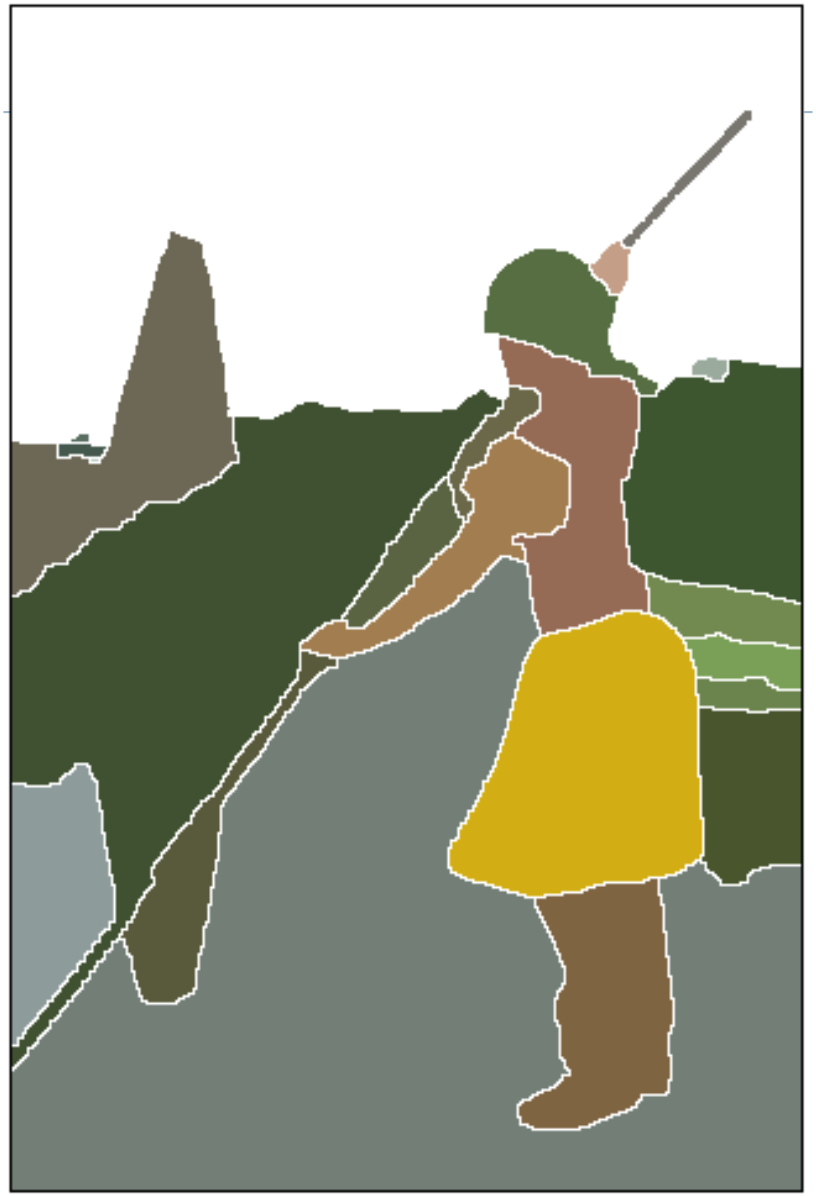
Segmentation



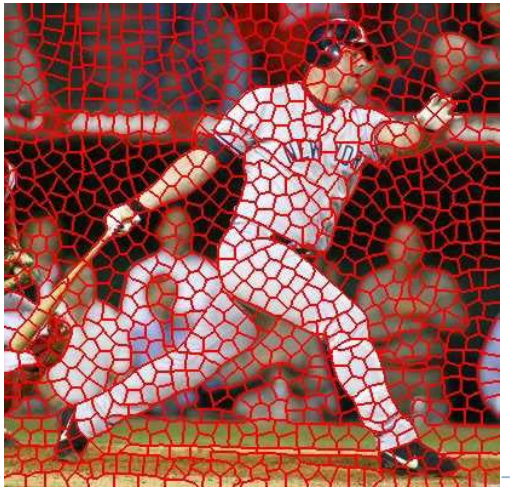
Slide credit: Ioannis Gkioulekas ,Kris Kitani, Fredo Durand, James Hays, Srinivasa Narasimhan Derek Hoiem, Lihi Zelnik

Image segmentation

Group pixels into meaningful or perceptually similar regions

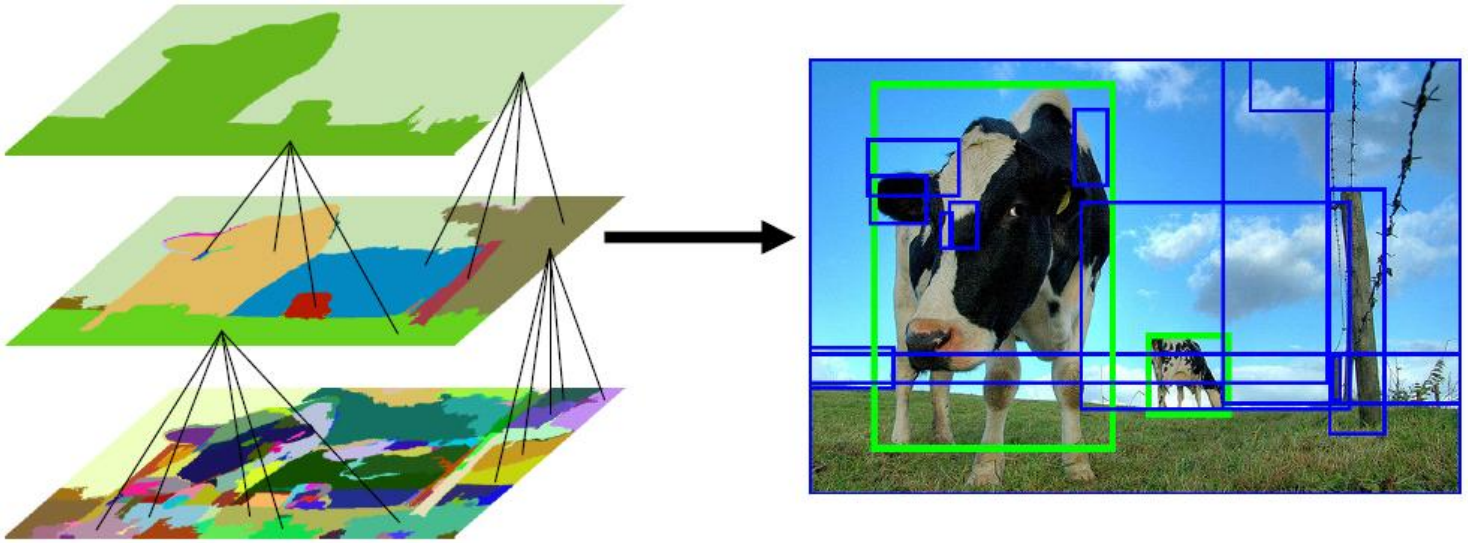
Segmentation for efficiency: “superpixels”

[Felzenszwalb and Huttenlocher 2004]

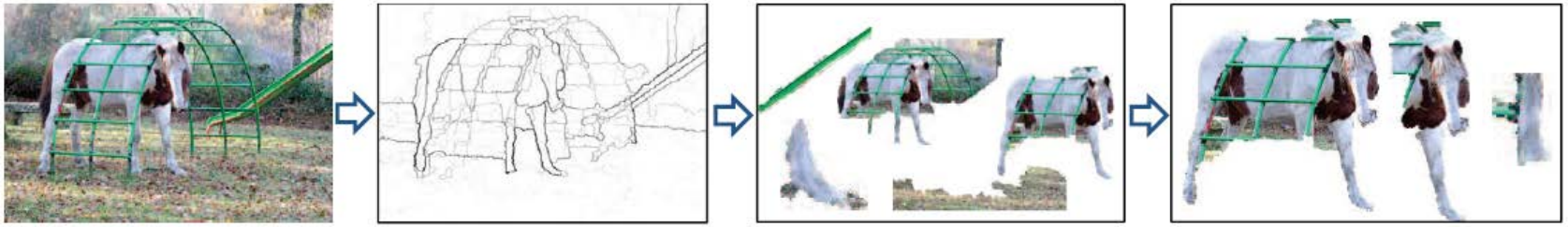


[Hoiem et al. 2005, Mori 2005]

Segmentation for object proposals



"Selective Search" [Sande, Uijlings et al. ICCV 2011, IJCV 2013]



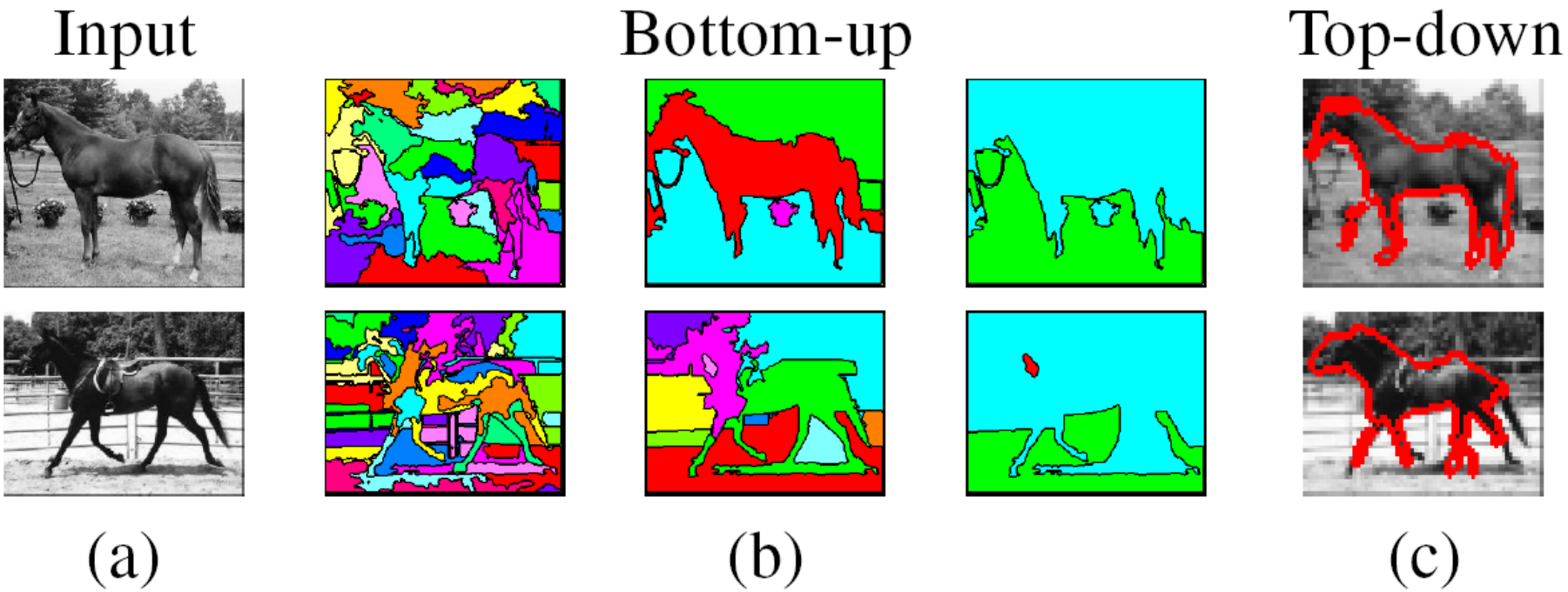
[Endres & Hoiem ECCV 2010, IJCV 2014]

Segmentation for image editing

Rother et al. 2004

Major processes for segmentation

- Bottom-up: group tokens with similar features
- Top-down: group tokens that likely belong to the same object

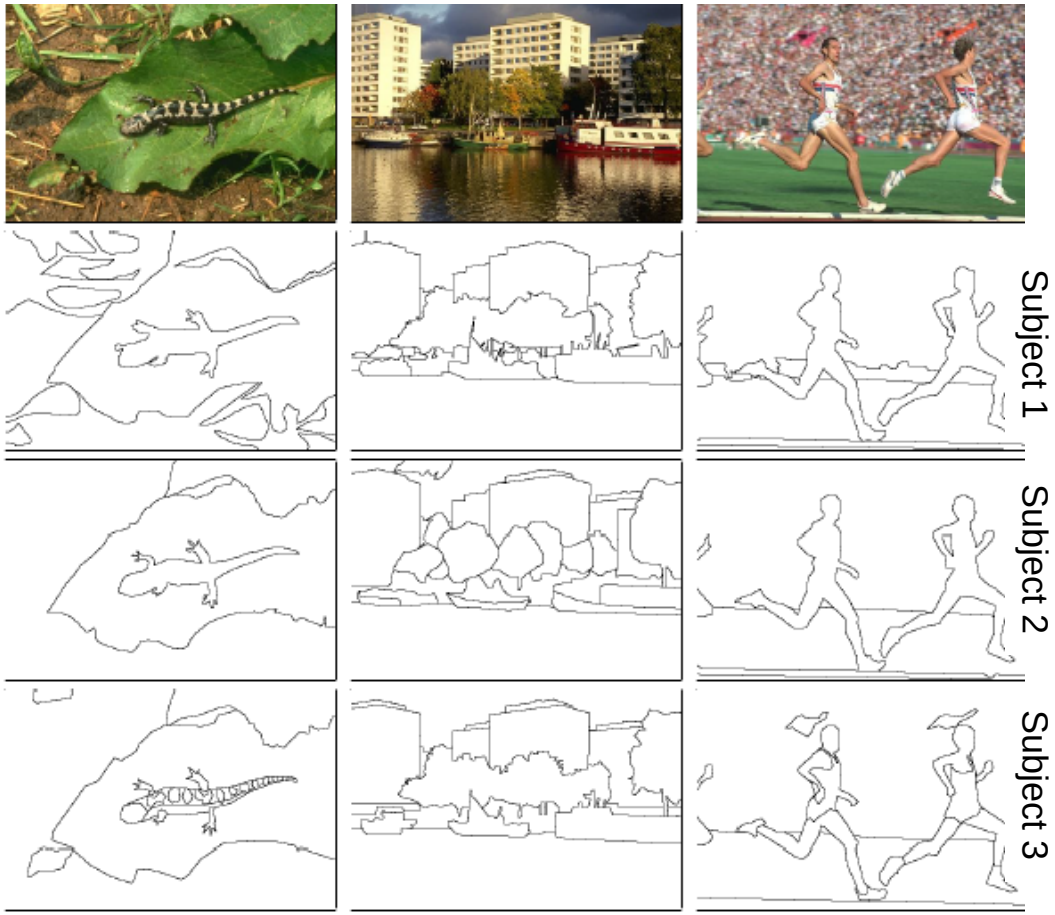


[Levin and Weiss 2006]

What is a “good” segmentation??

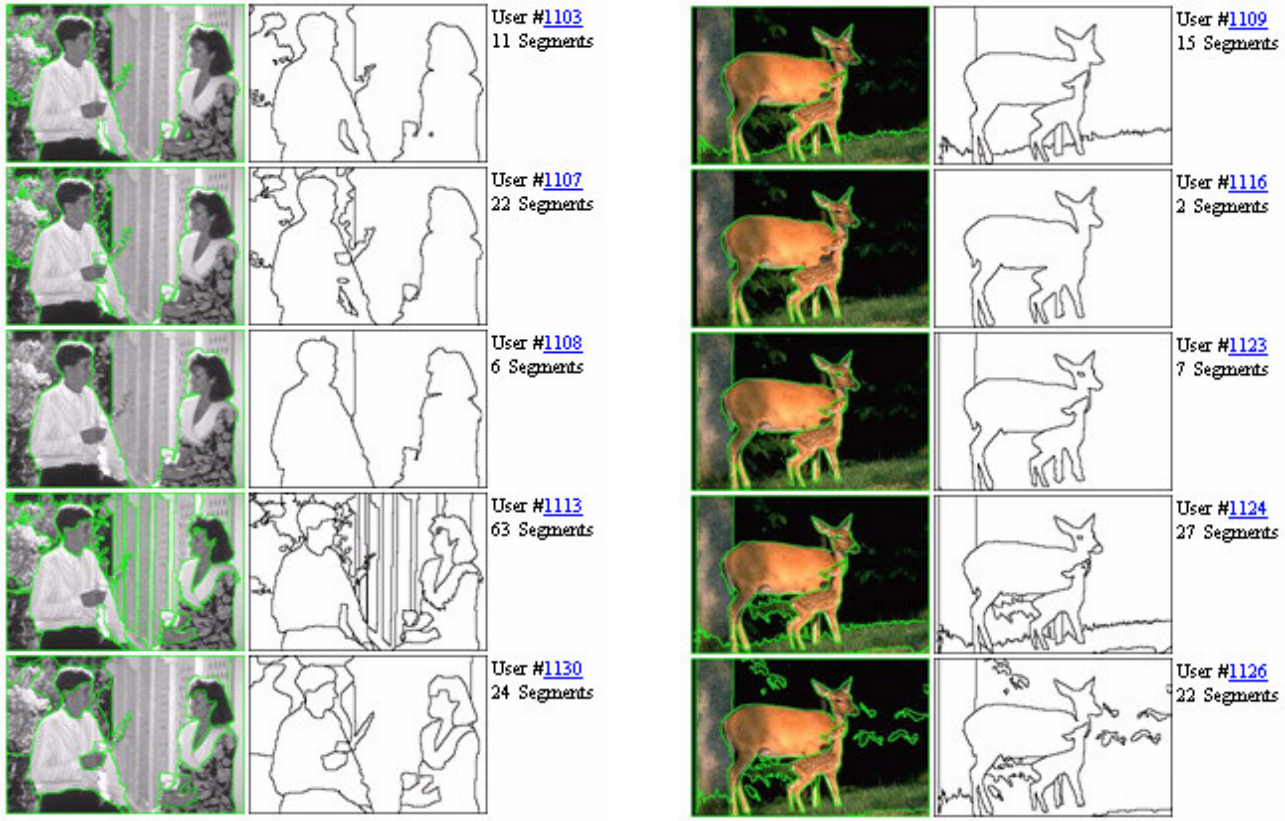
First idea: Compare to human segmentation or to “ground truth”

No objective definition of segmentation!



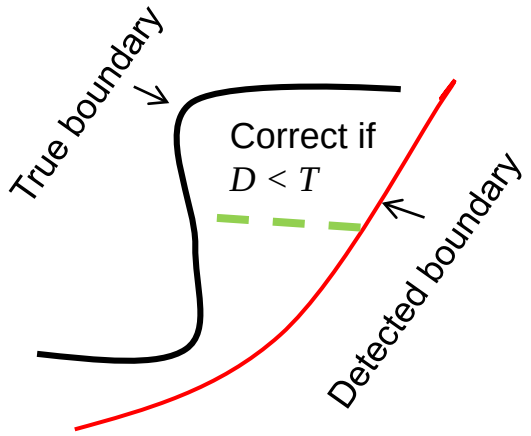
- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

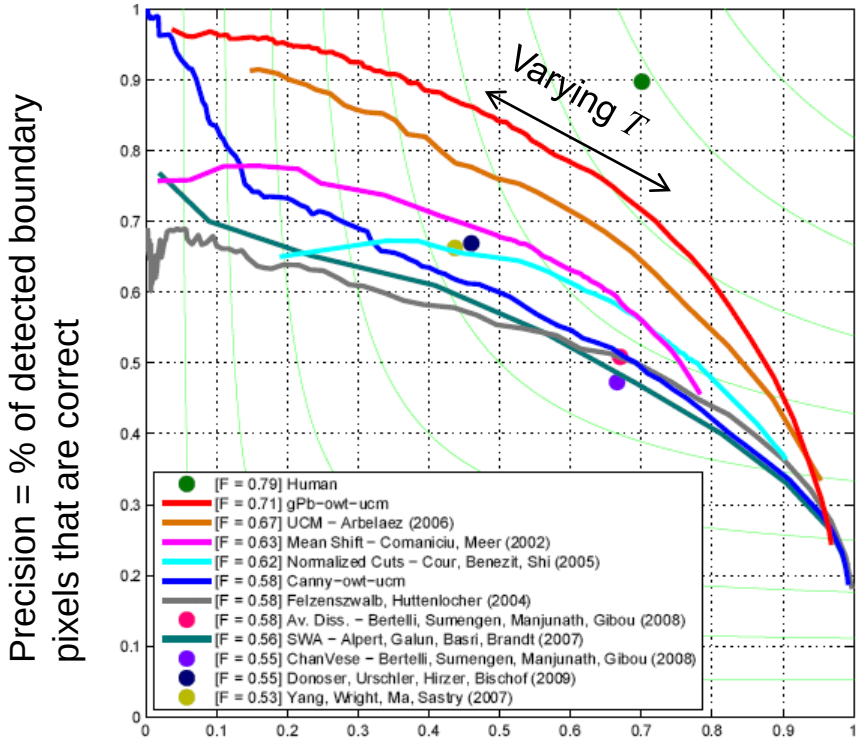
No objective definition of segmentation!



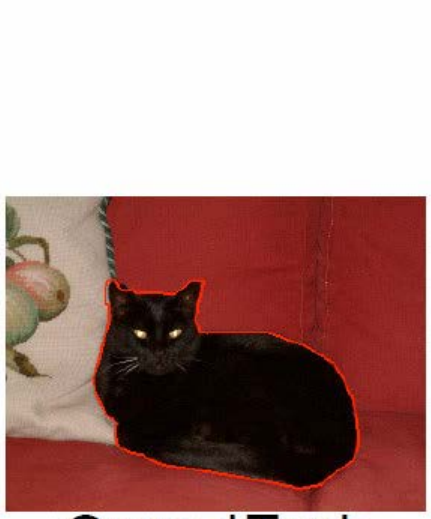
- http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/dataset/images/color/317080.html

Evaluation: Boundary agreement





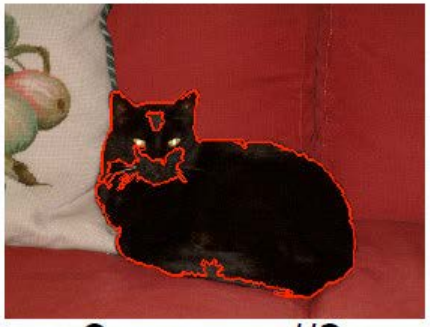
Evaluation: Region overlap with ground truth



Ground Truth

.825

Segment #1

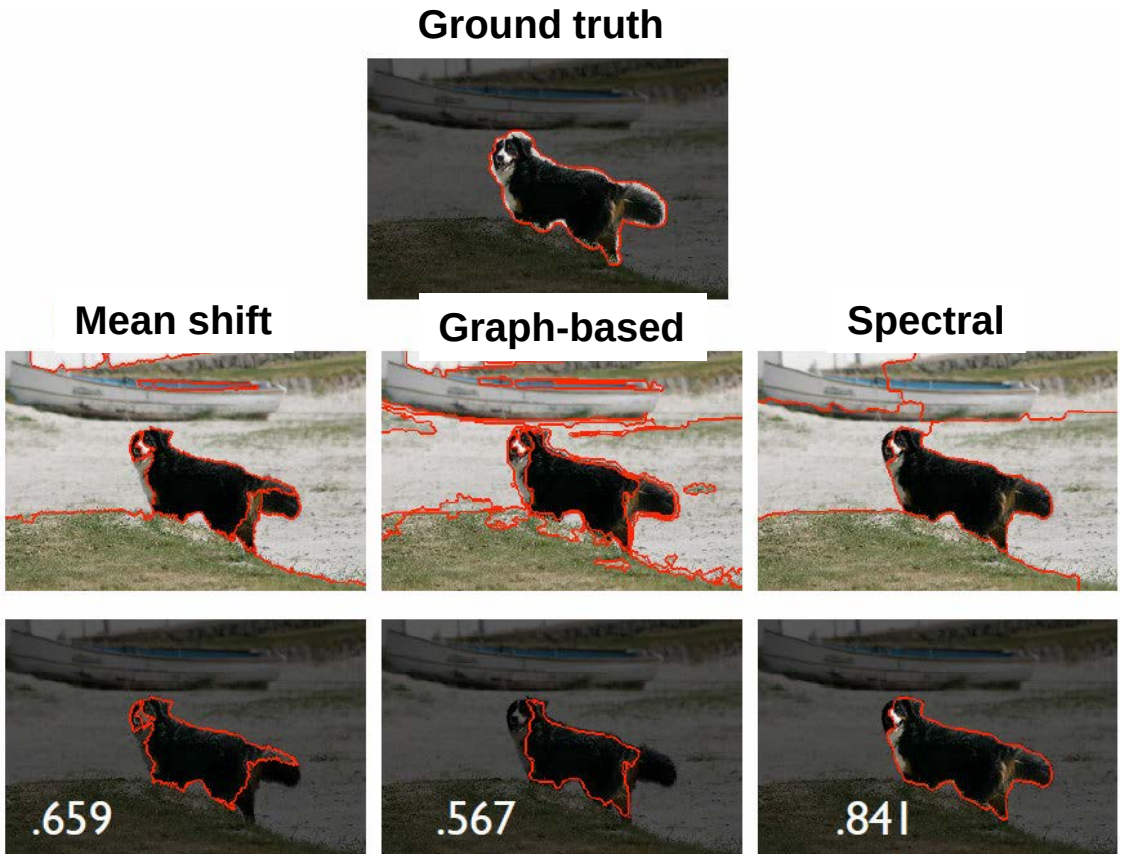


.892

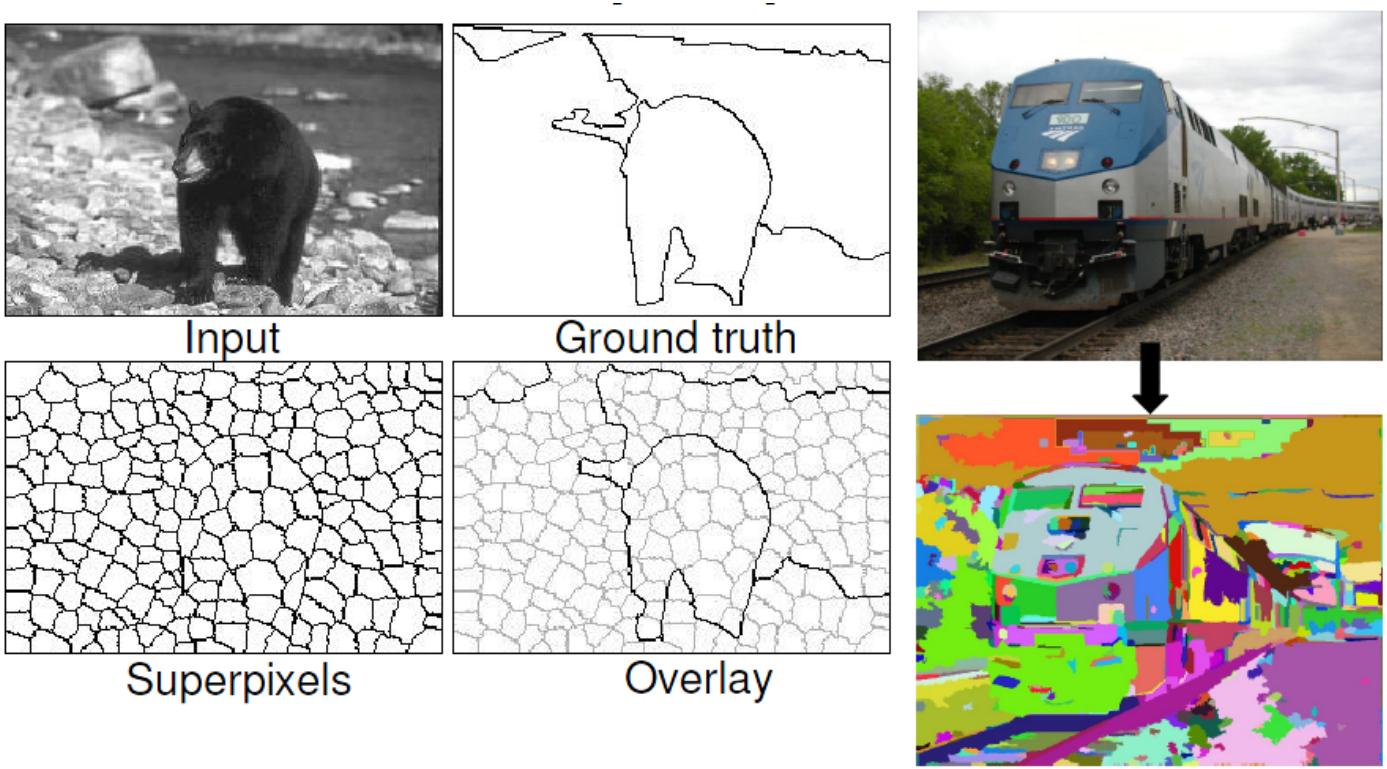
Segment #2

$$OS(S,G) = \frac{|S \cap G|}{|S \cup G|}$$

Evaluation: Region overlap with ground truth

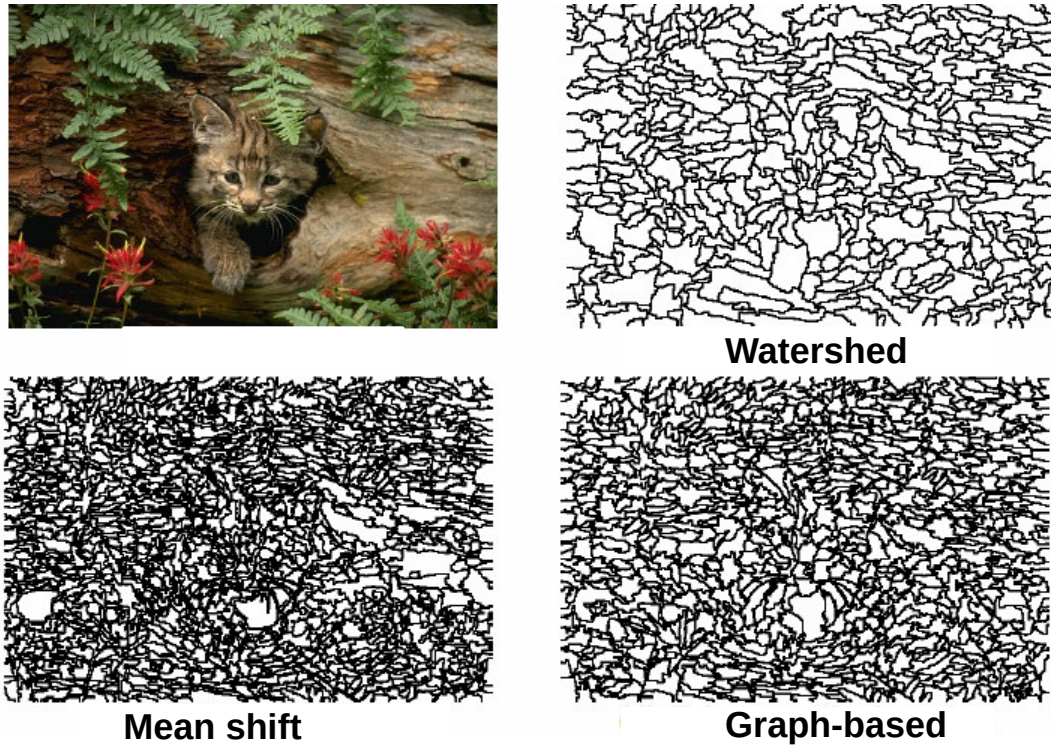


Second idea: Superpixels



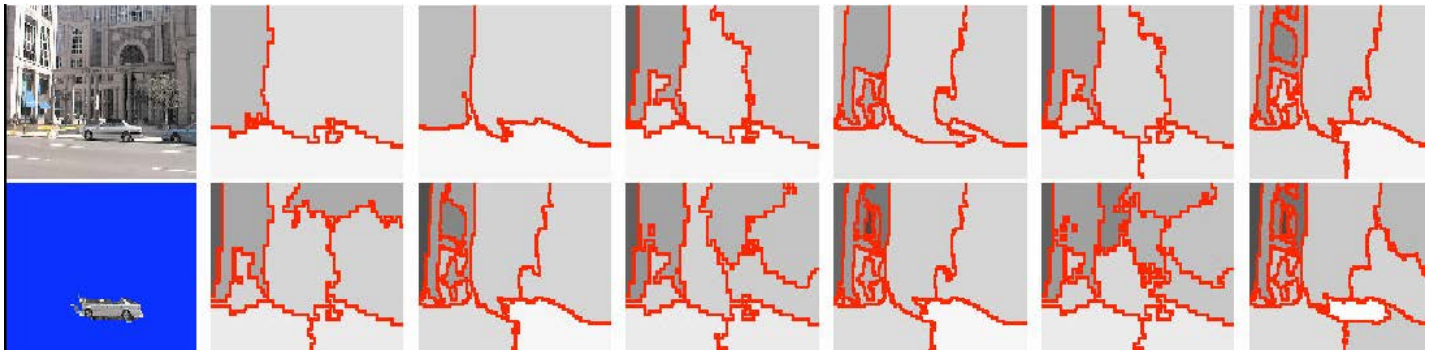
- Let's not even try to compute a "correct" segmentation
- Let's be content with an *oversegmentation* in which each region is very likely (formal guarantees are hard) to be uniform

Second idea: Superpixels



- Example from: How Do Superpixels Affect Image Segmentation?
- Progress in Pattern Recognition, Image Analysis and Applications. Springer LNCS. Volume 5197/2008.

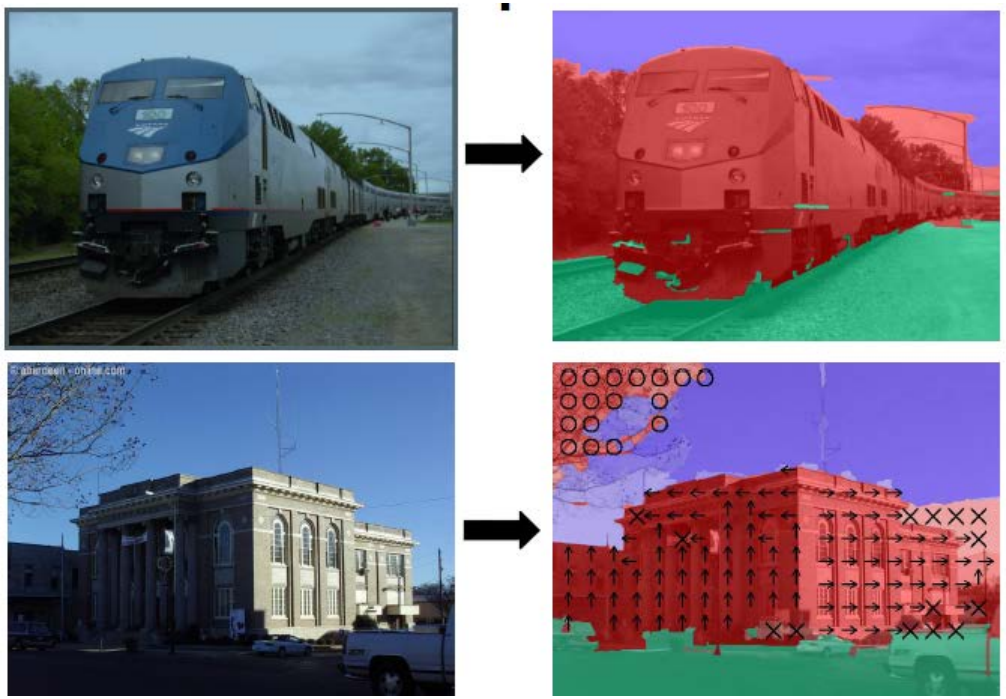
Third idea:
Multiple segmentations



- Generate many segmentations of the same image
- Even though many regions are "wrong", some consensus should emerge

Example: Improving Spatial Support for Objects via Multiple Segmentations Tomasz Malisiewicz and Alexei A. Efros. British Machine Vision Conference (BMVC), September, 2007.

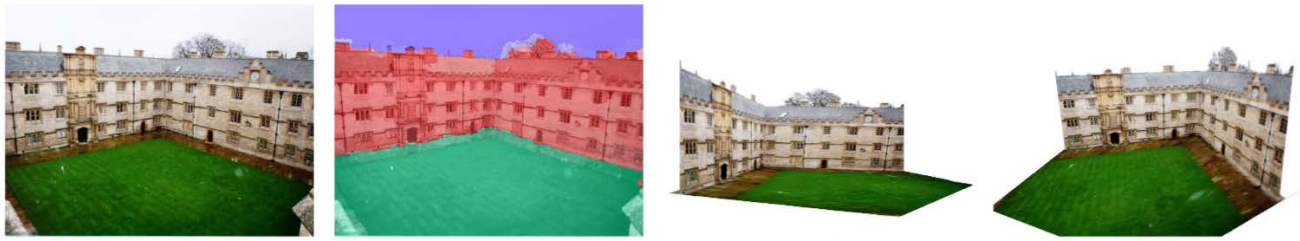
Multiple segmentations: Example



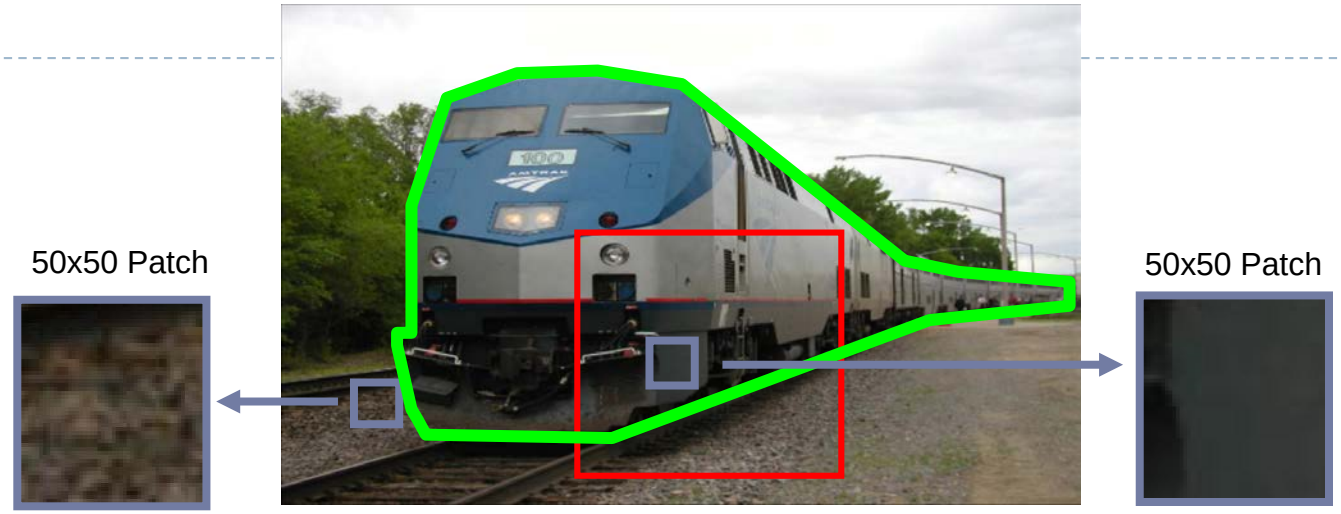
- Task:

Regions → Features → Labels

(horizontal, vertical, sky, etc.)

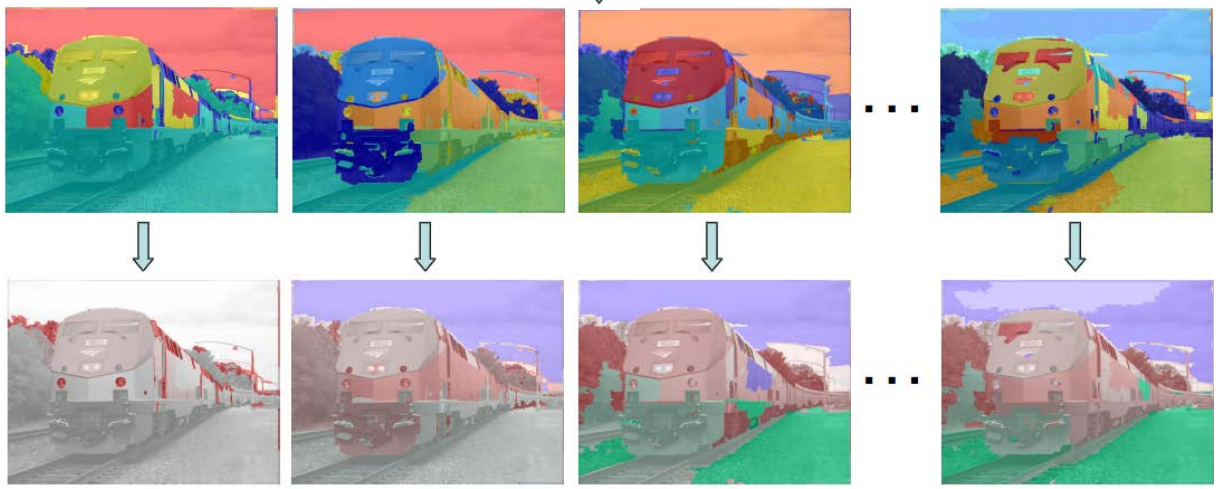


Input Labels Novel View Novel View



- Chicken and egg problem:
  - If we knew the regions, we could compute the features and label the right regions
  - But to know the right regions we need to know the labels!
- Solution:
  - Generate lots of segmentations
  - Combine the classifications to get consensus

Recovering Surface Layout from an Image. D. Hoiem, A.A. Efros, and M. Hebert. IJCV, Vol. 75, No. 1, October 2007.



P(*good segment* | *data*) x

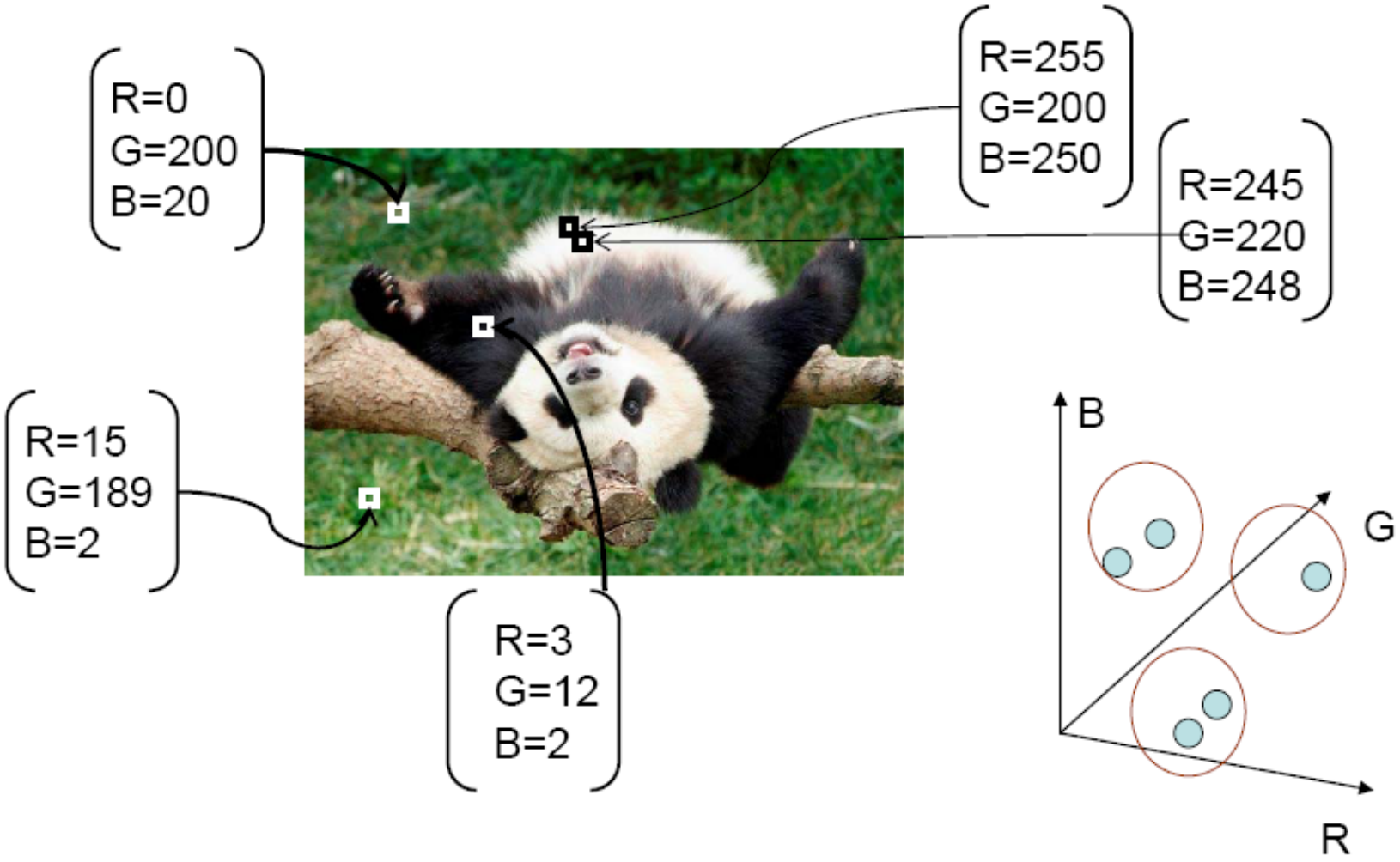
P(*label* | *good segment*, *data*)

$$P(label \mid data) \propto \sum_{segments} P(good\ seg \mid data) P(label \mid good\ seg, data)$$

Segmentation using clustering

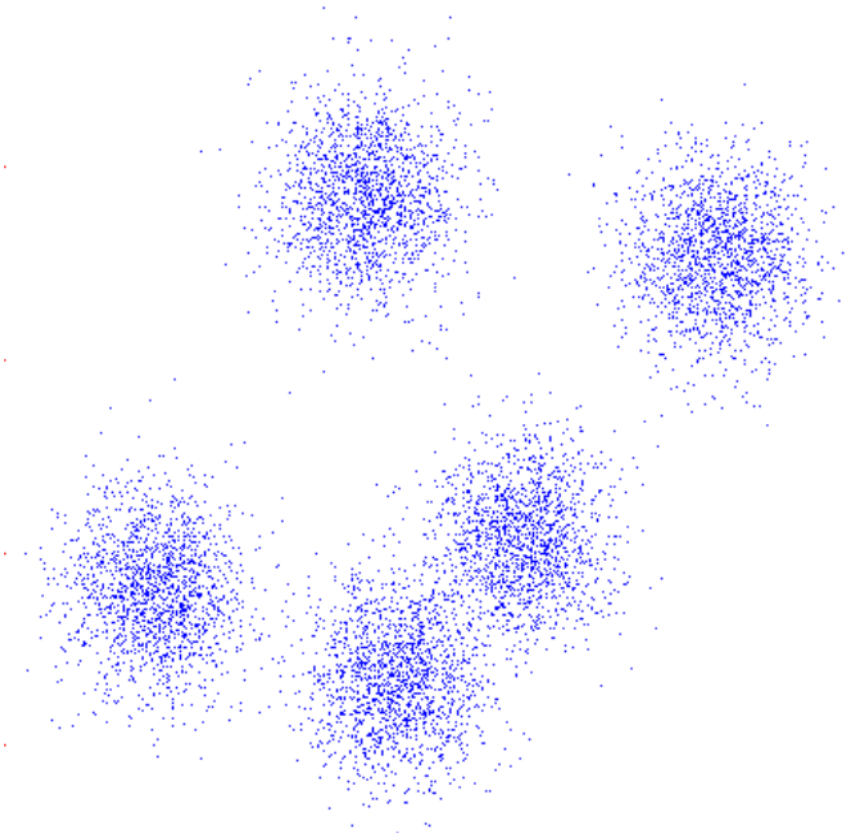
- Kmeans
- Mean-shift

Feature Space

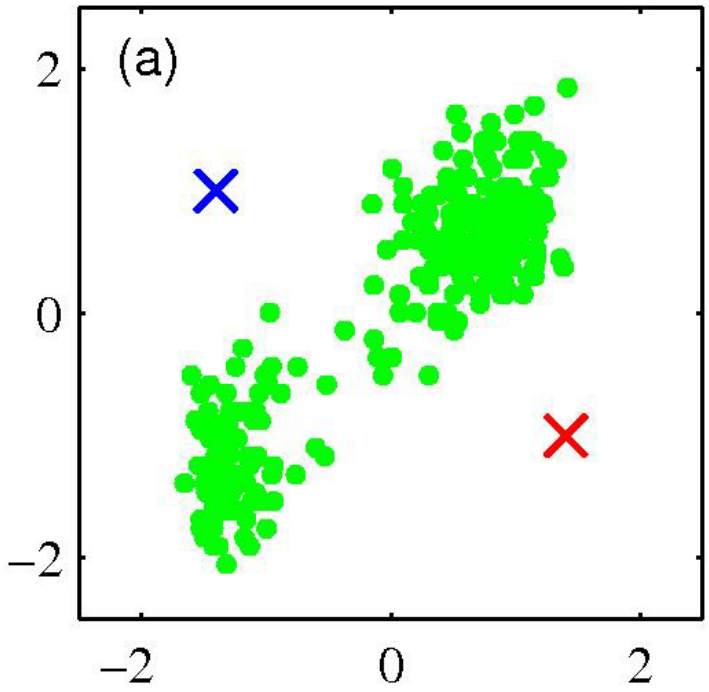


K-means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change

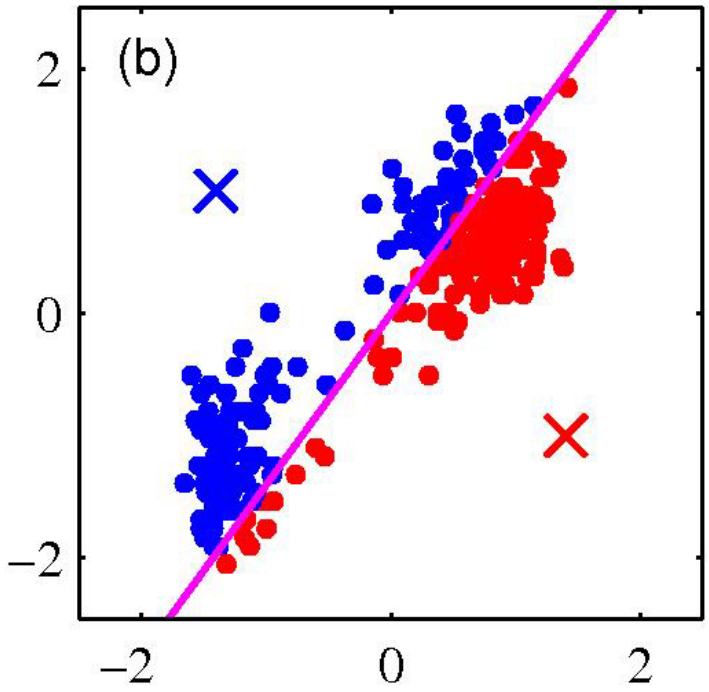


K-means



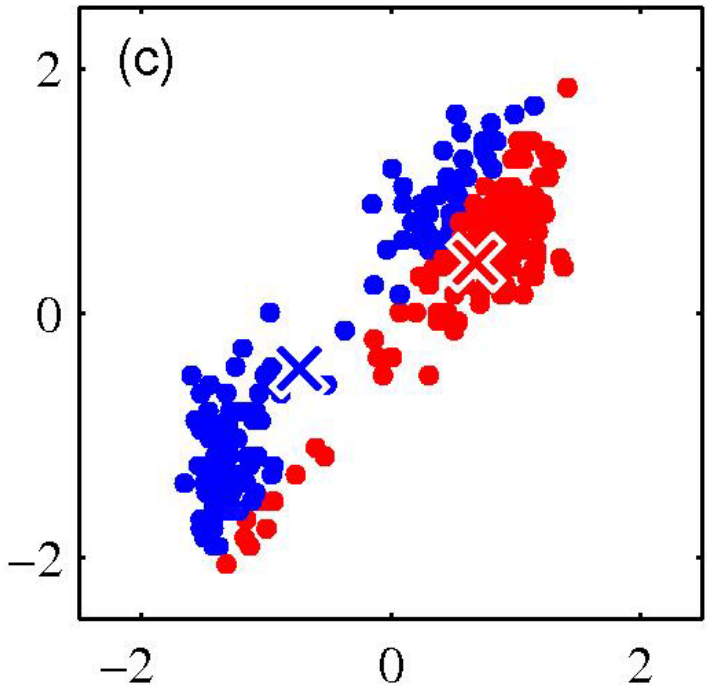
Pick *K* random points as cluster centers (means)
Shown here for *K*=2

K-means



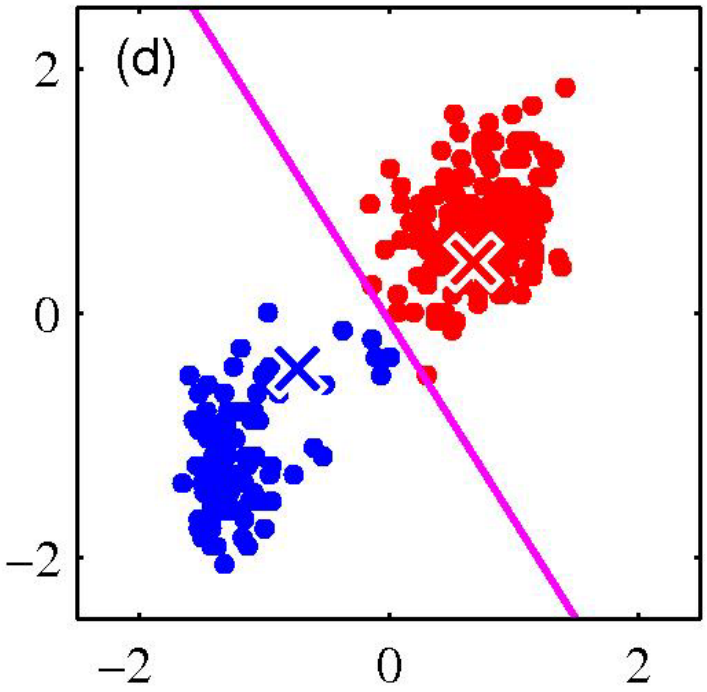
Assign data points to closest cluster center

K-means



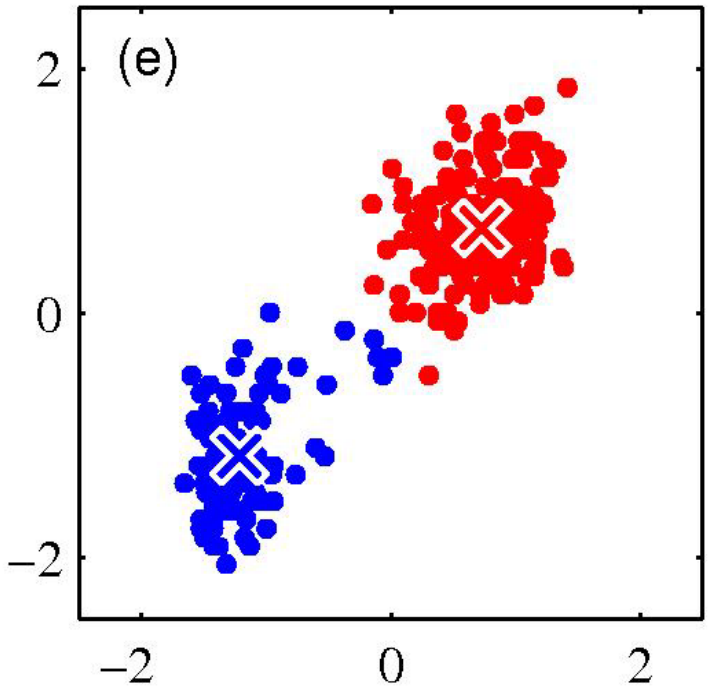
Change the cluster center to the average of the assigned points

K-means

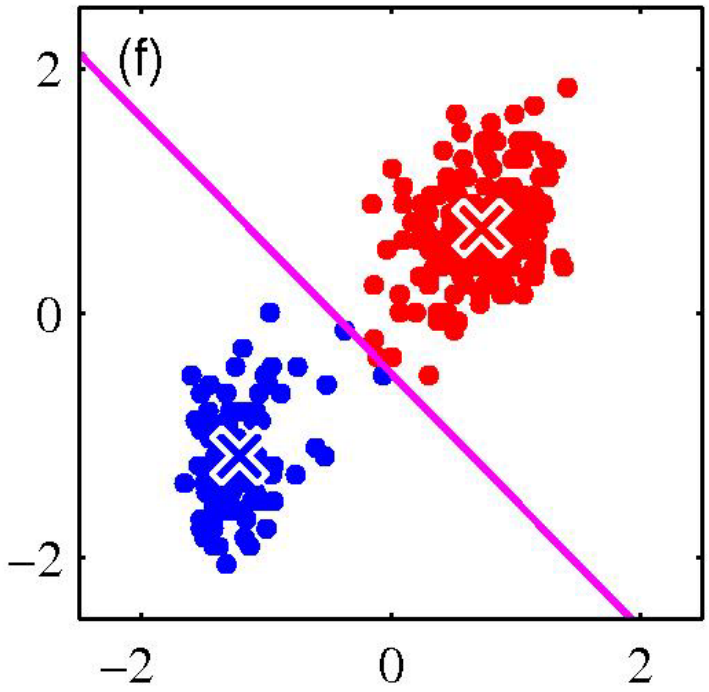


Repeat until convergence

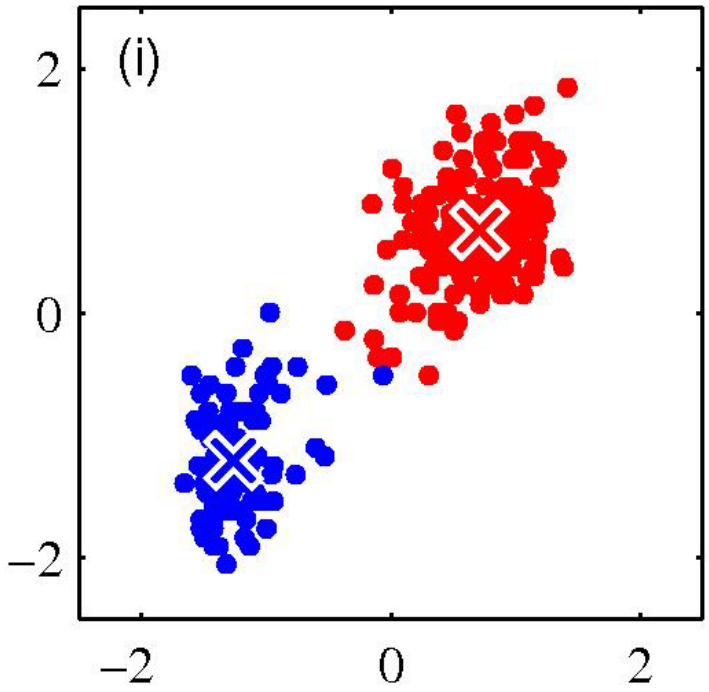
K-means



K-means



K-means



K-means clustering using intensity alone and color alone

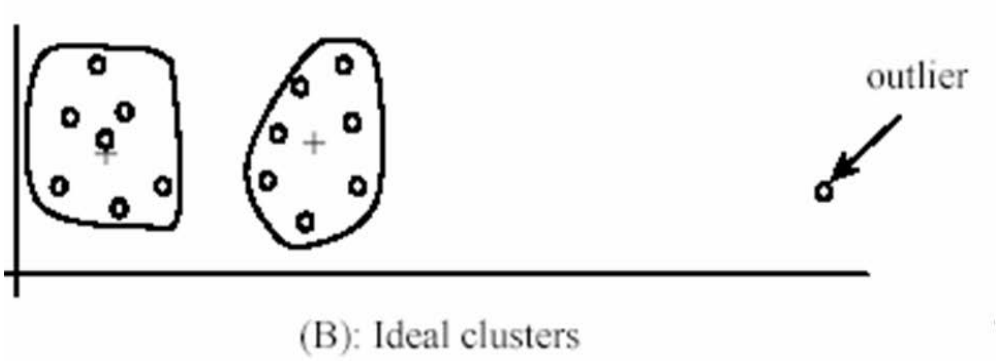
Input image

Clusters on intensity

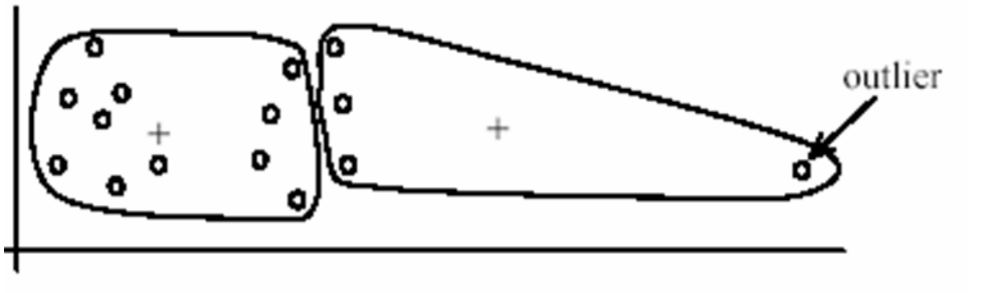
Clusters on color

K-Means pros and cons

- Pros
  - Simple and fast
  - Easy to implement



- Cons
  - Need to choose K
  - Sensitive to outliers

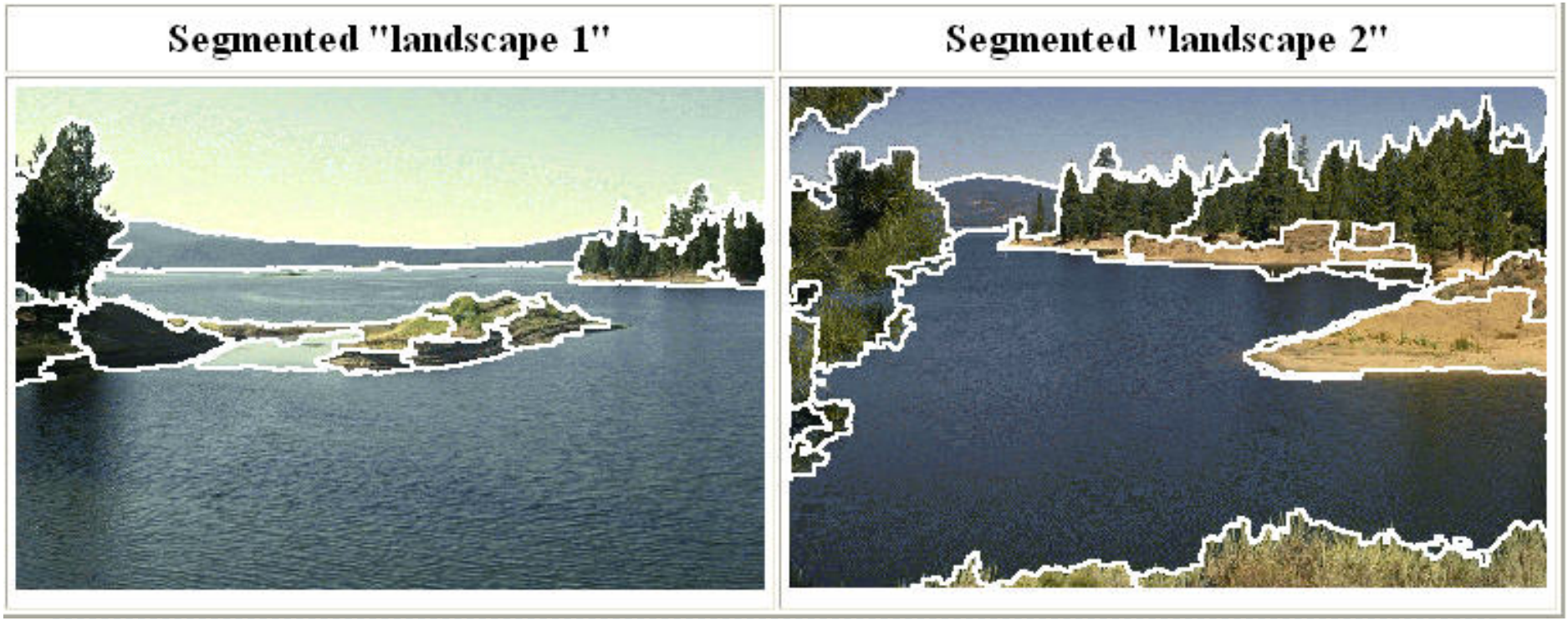


- Usage
  - Rarely used for pixel segmentation

Mean shift segmentation

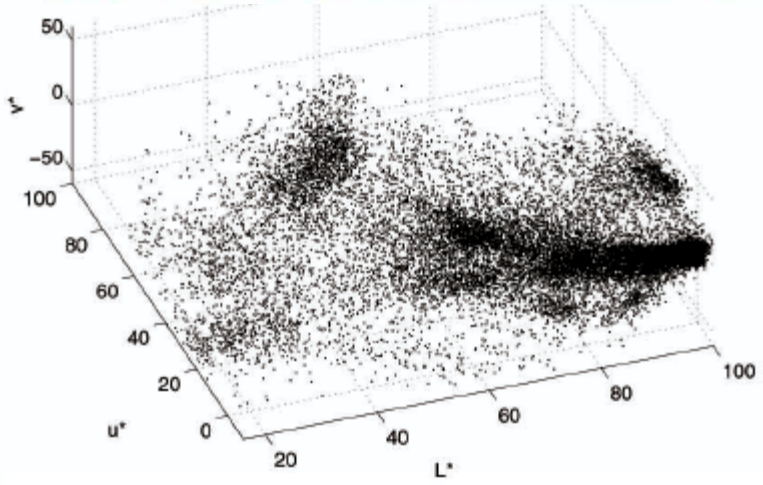
- D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

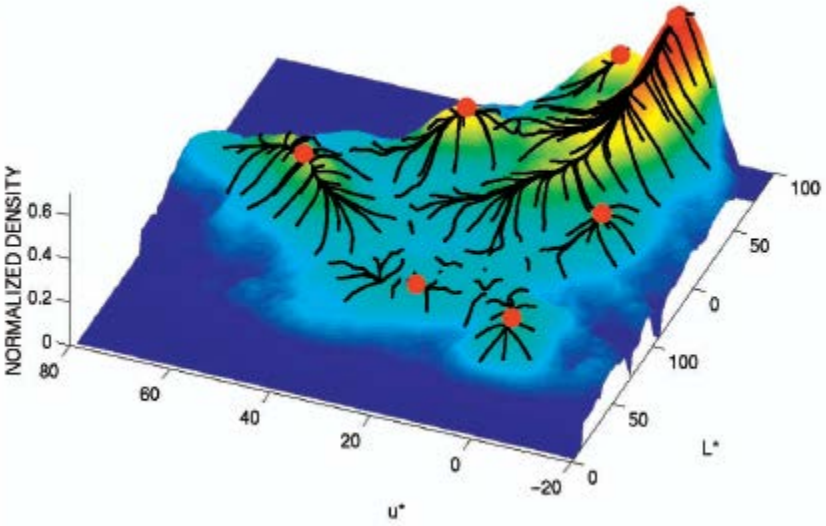
- Versatile technique for clustering-based segmentation

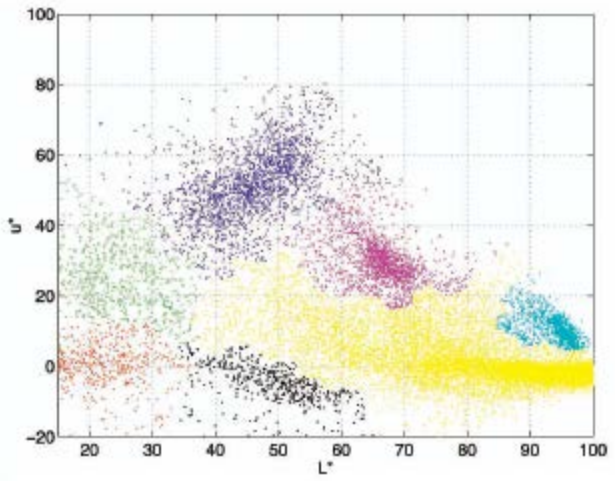


Mean shift algorithm

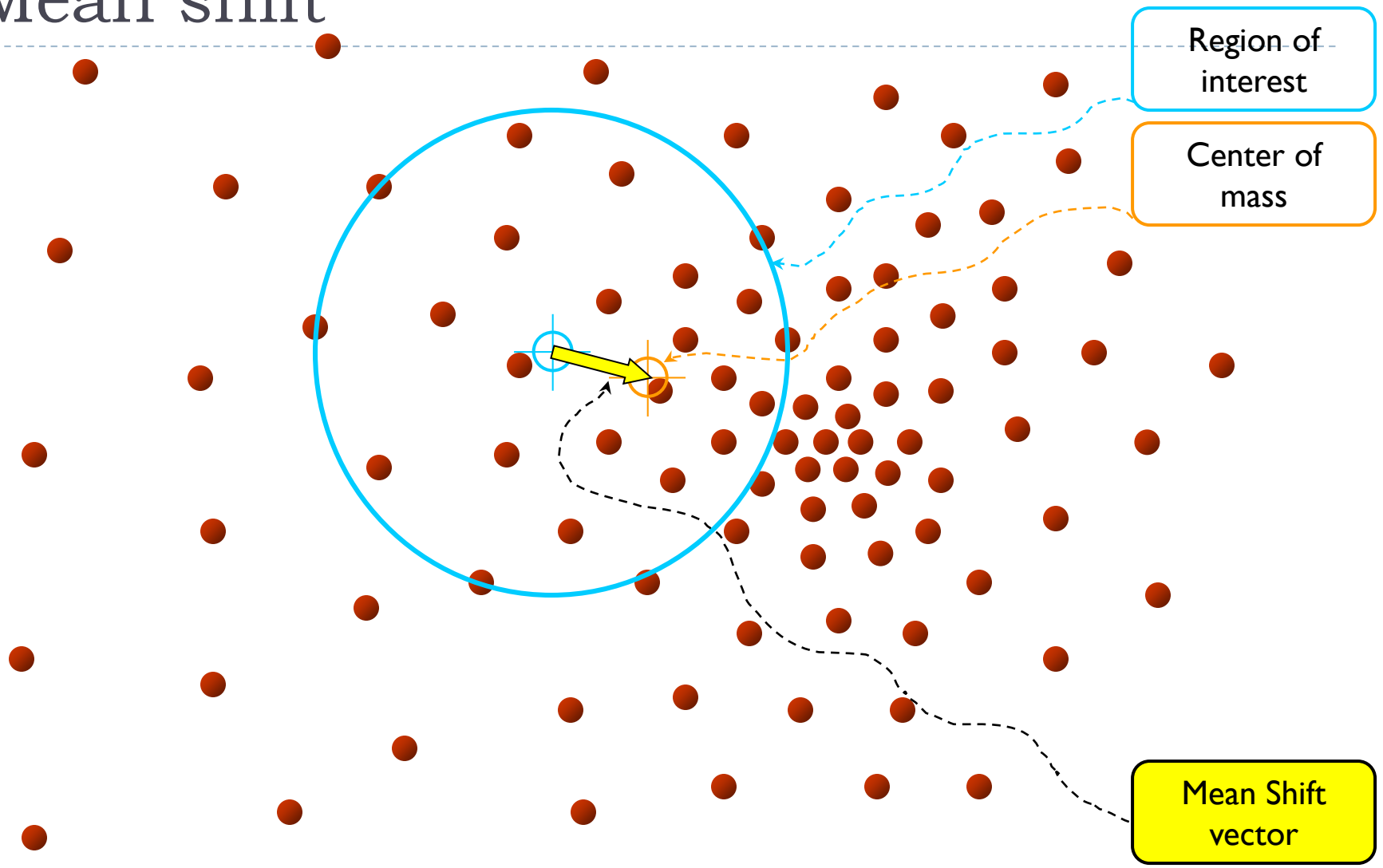
- Try to find *modes* of this non-parametric density





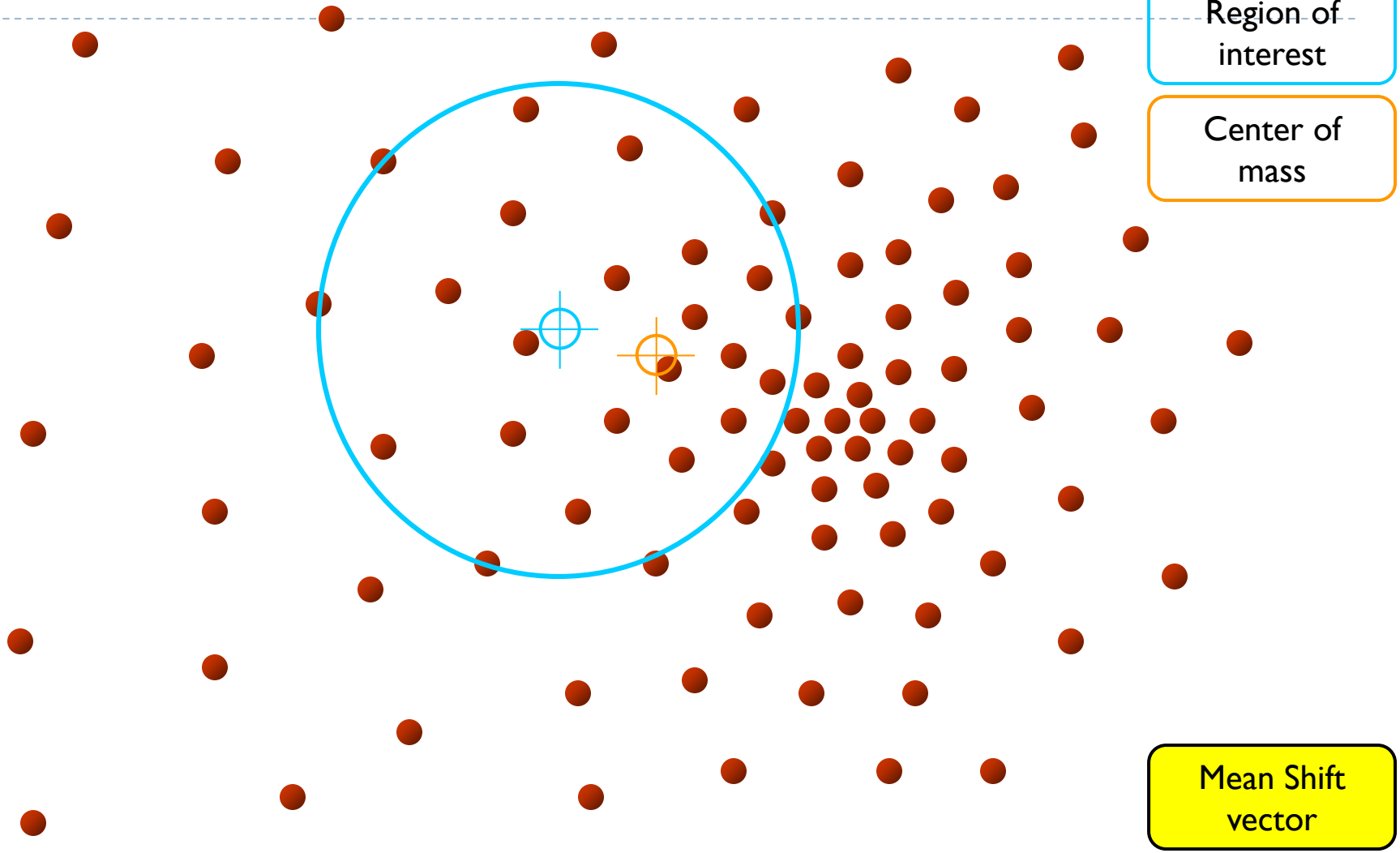


Mean shift



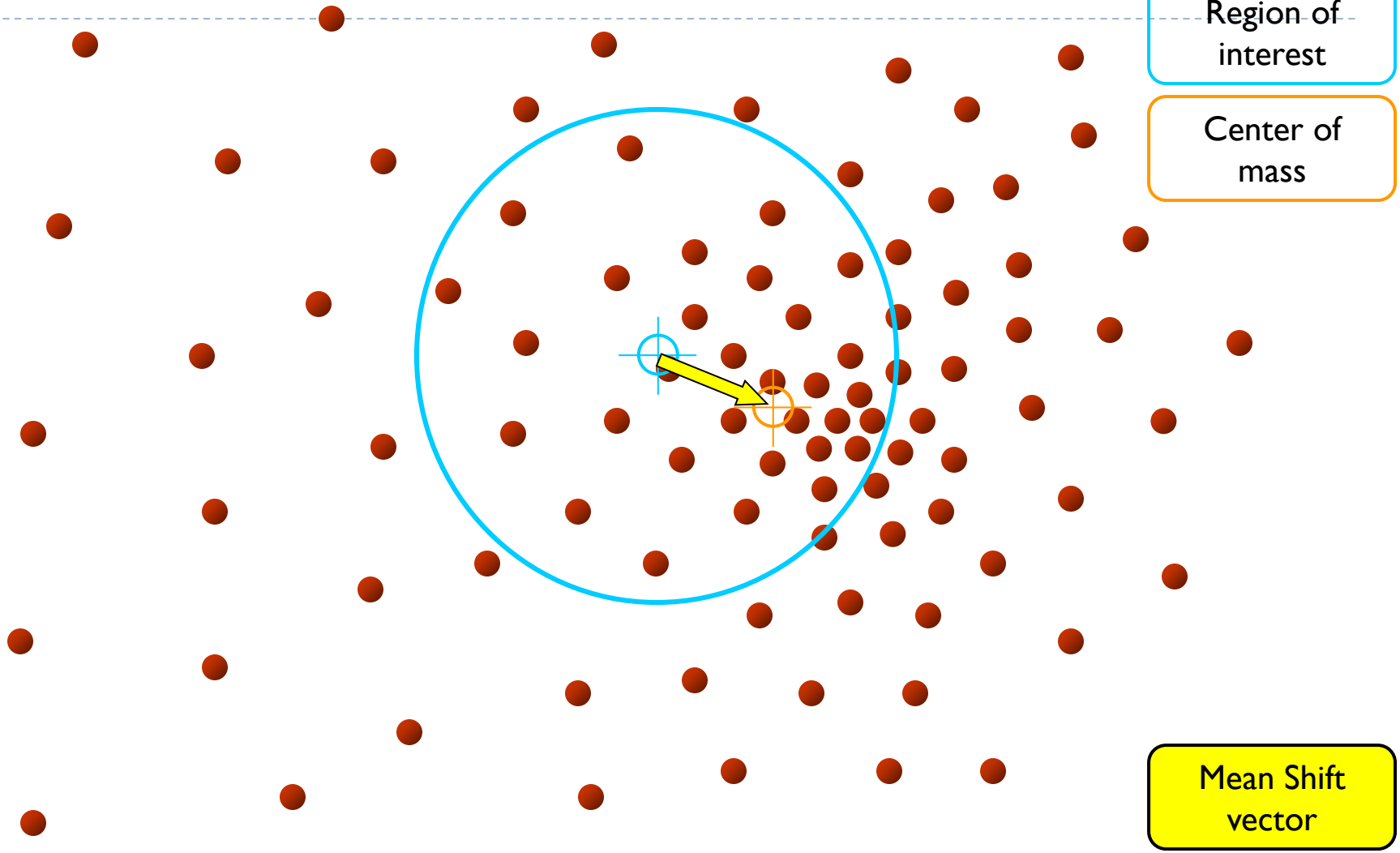
Slide by Y. Ukrainitz & B. Sarel

Mean shift



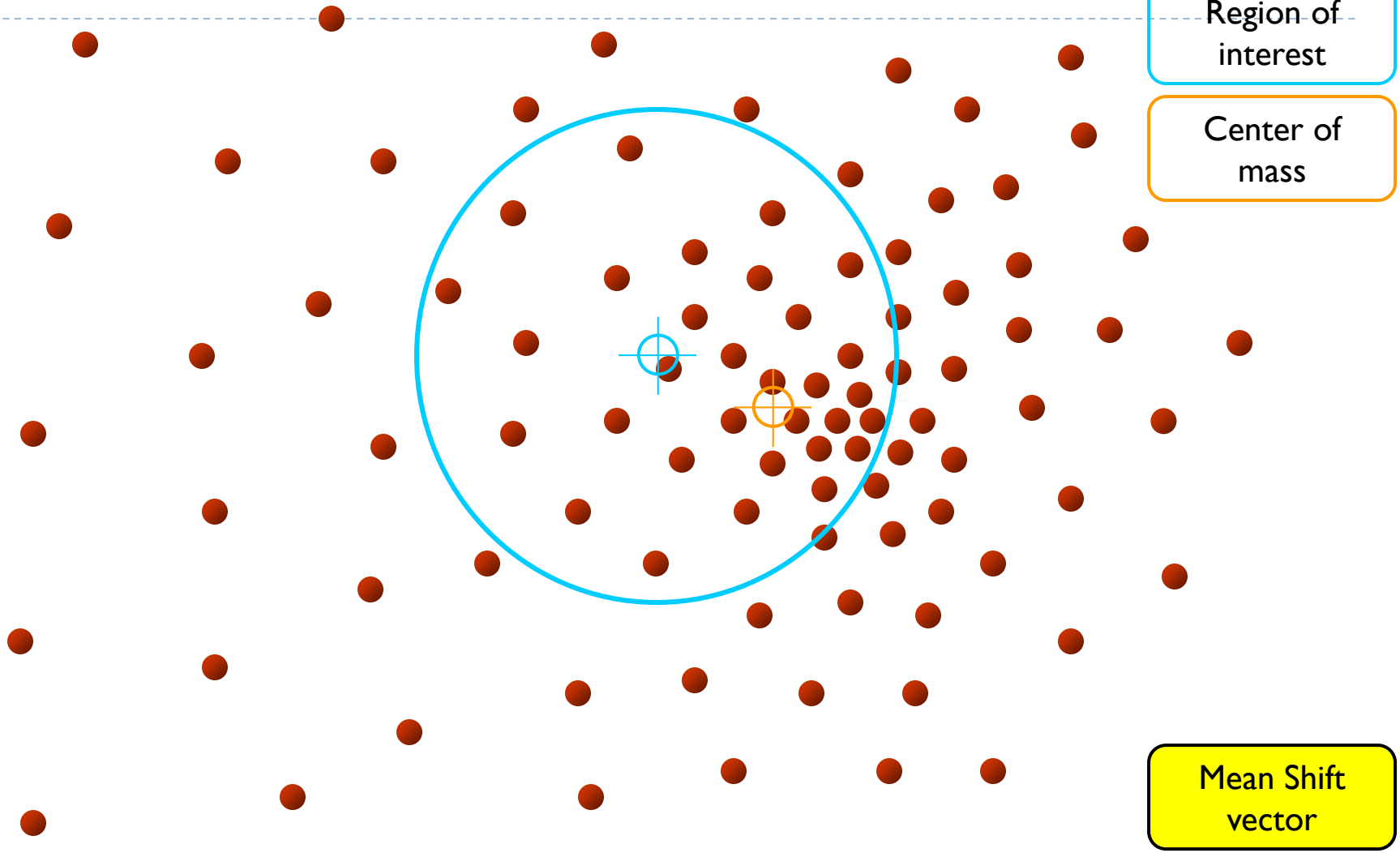
Slide by Y. Ukrainitz & B. Sarel

Mean shift



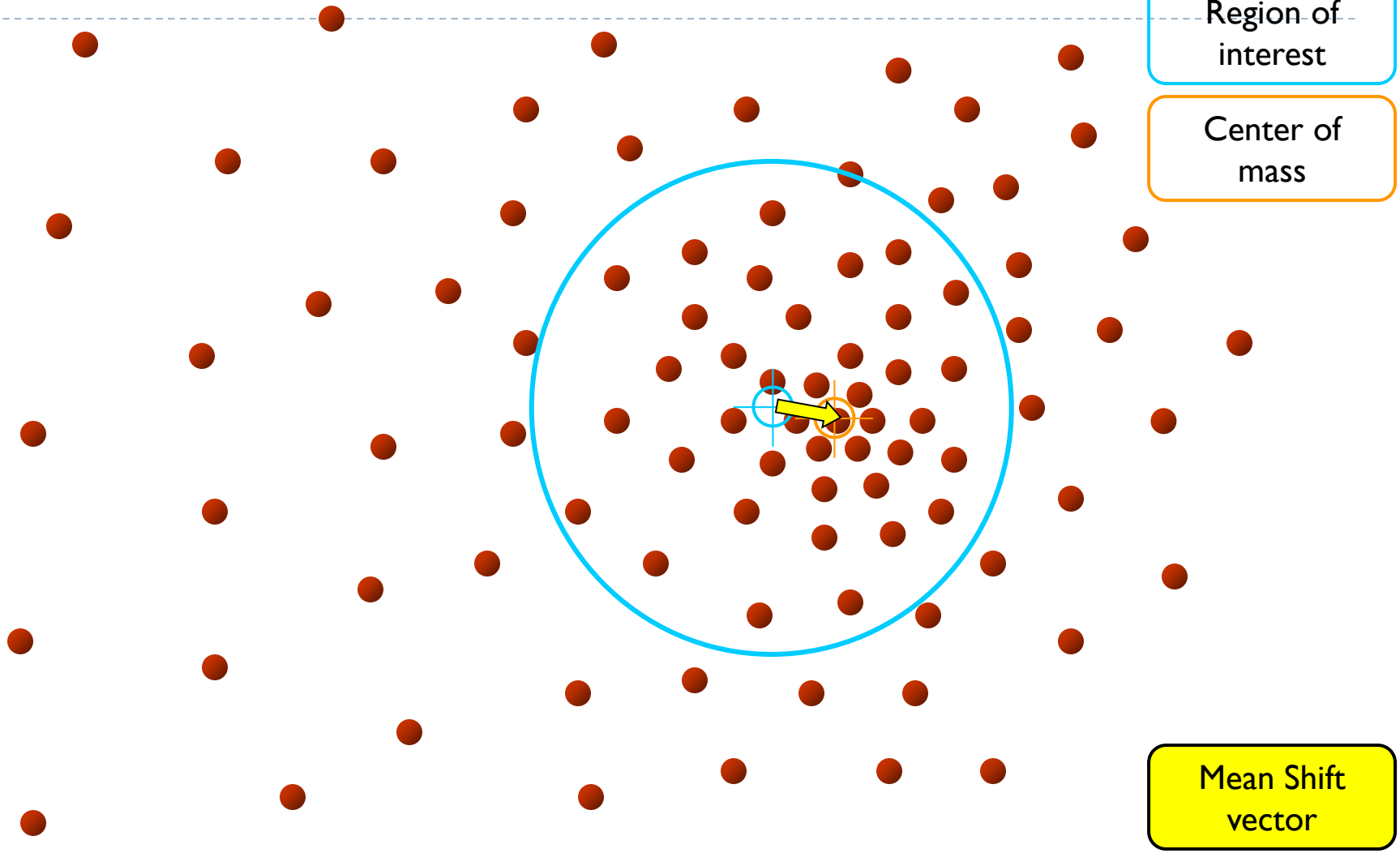
Slide by Y. Ukrainitz & B. Sarel

Mean shift



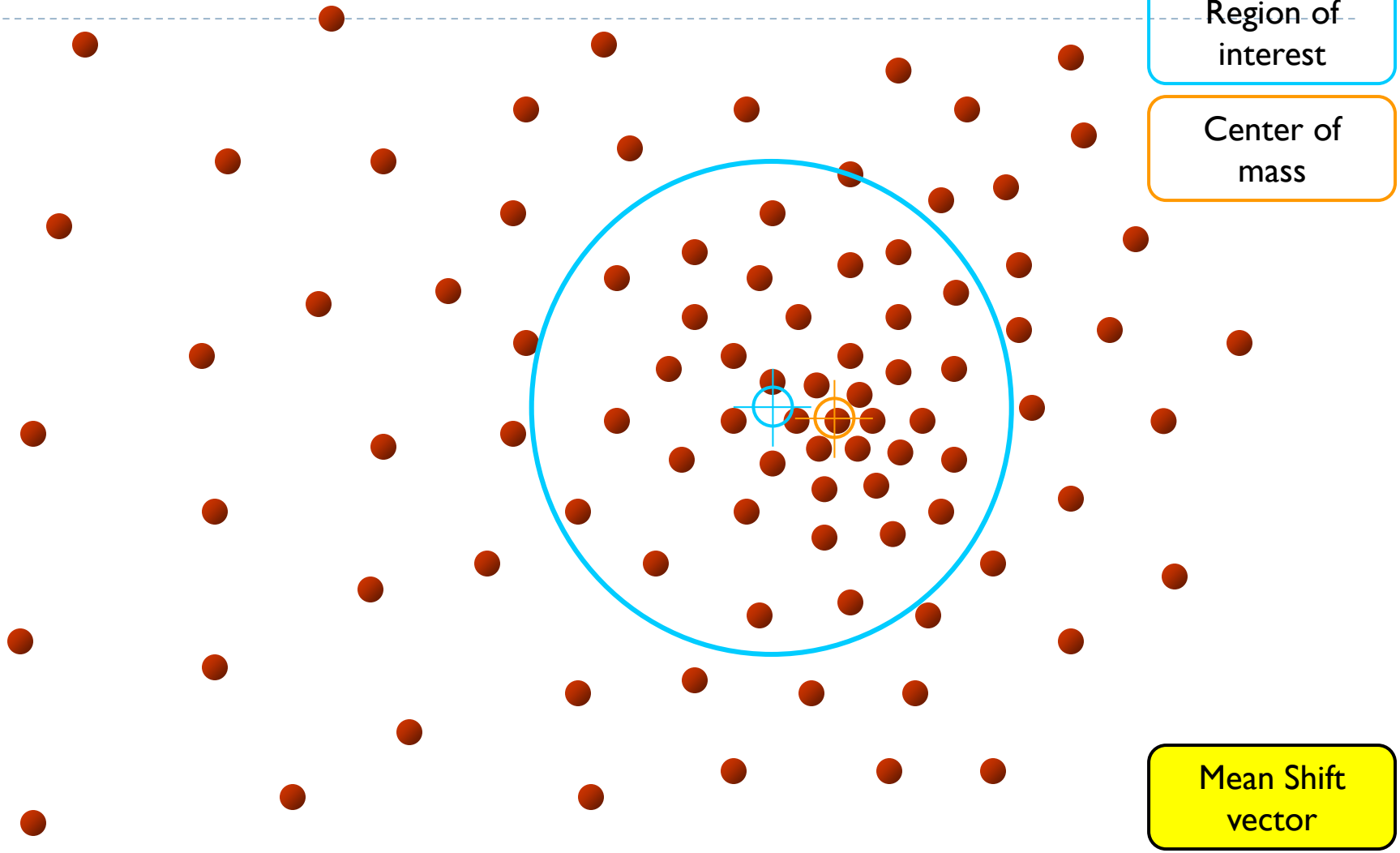
Slide by Y. Ukrainitz & B. Sarel

Mean shift



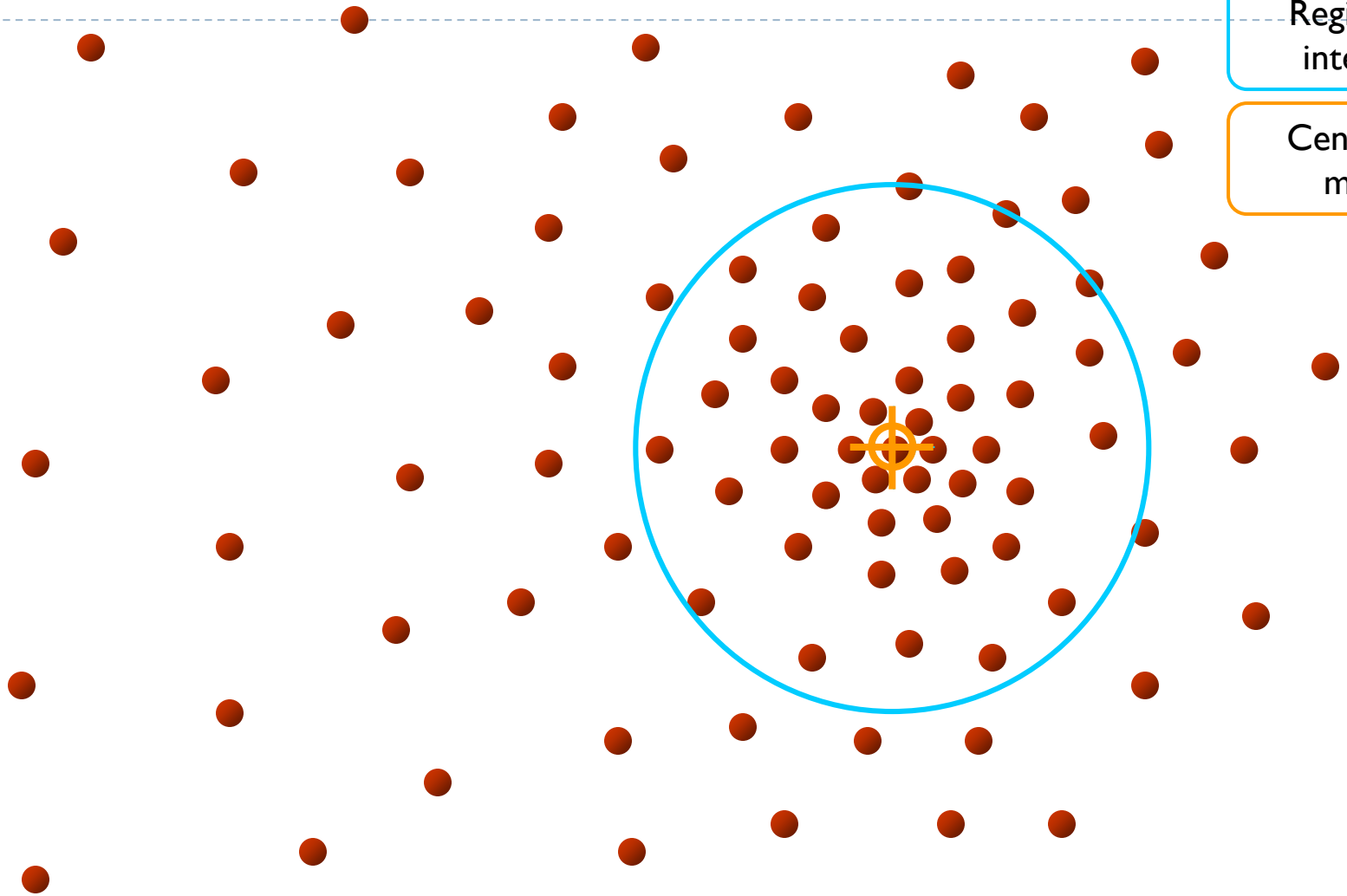
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

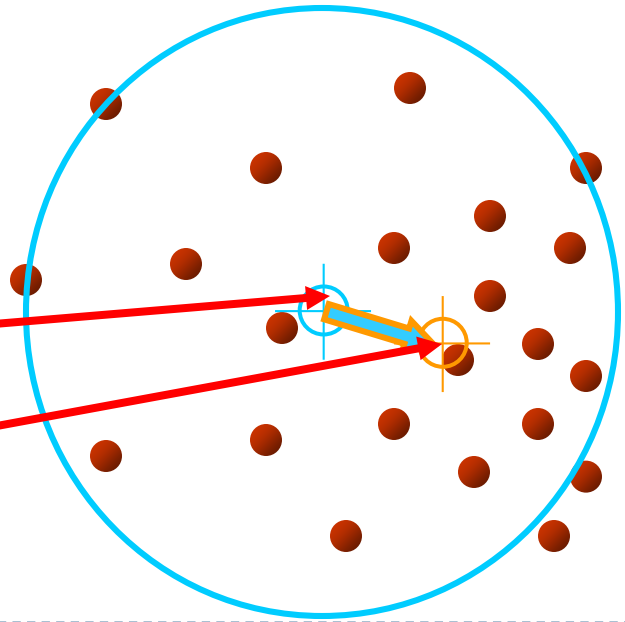
Computing the Mean Shift

Simple Mean Shift procedure:

- Compute mean shift vector

- Translate the Kernel window by **m(x)**

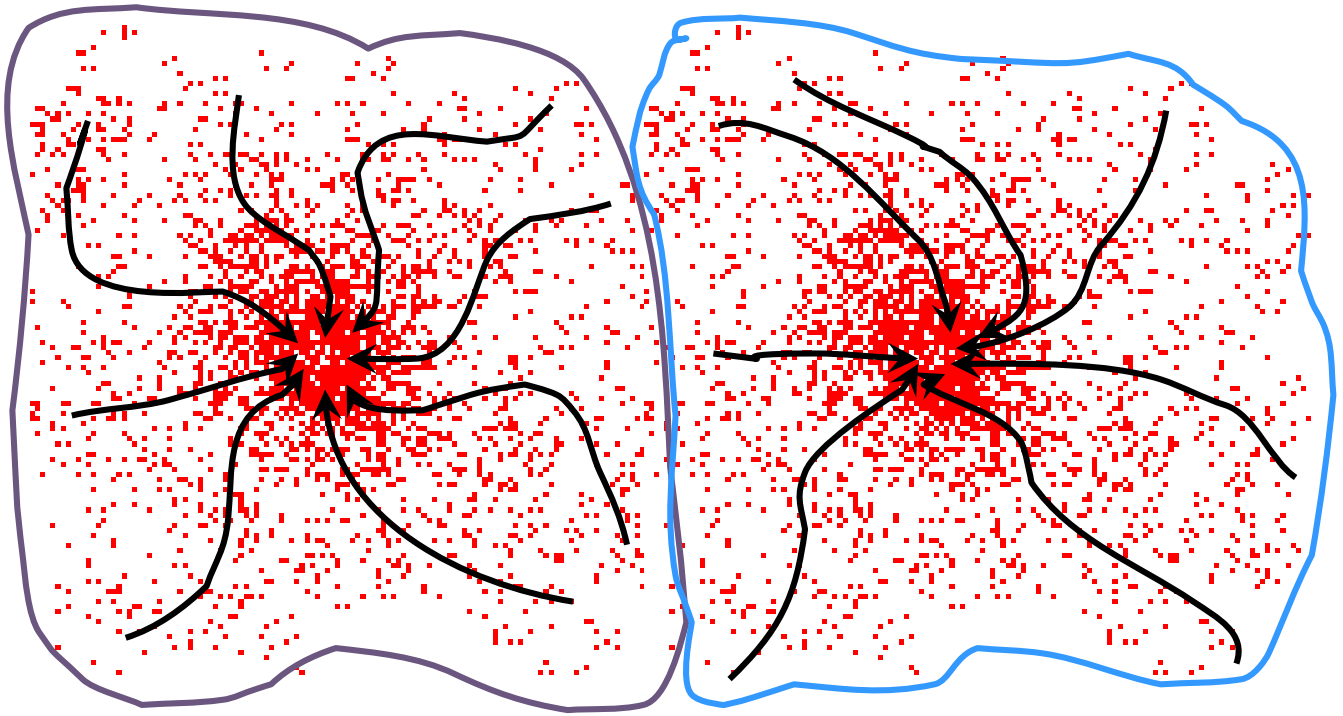
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h} \right)} \right] - \mathbf{x}$$



Slide by Y. Ukrainitz & B. Sarel

Attraction basin

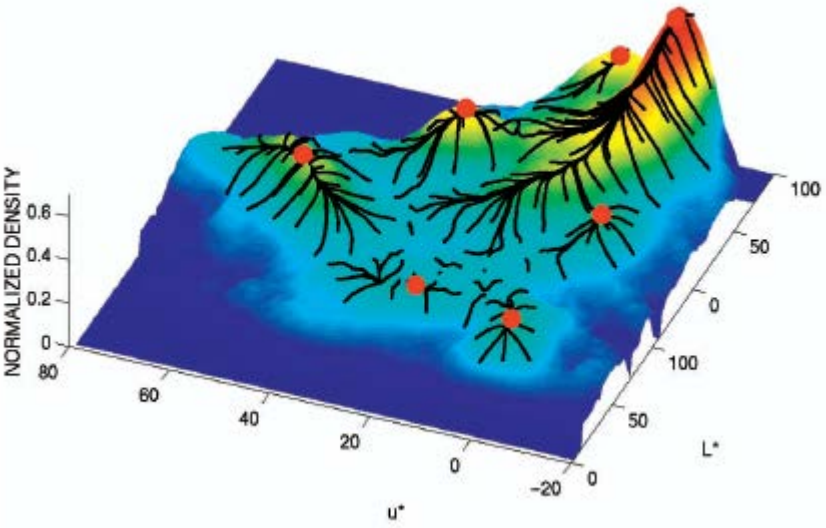
- Attraction basin: the region for which all trajectories lead to the same mode
- Cluster: all data points in the attraction basin of a mode

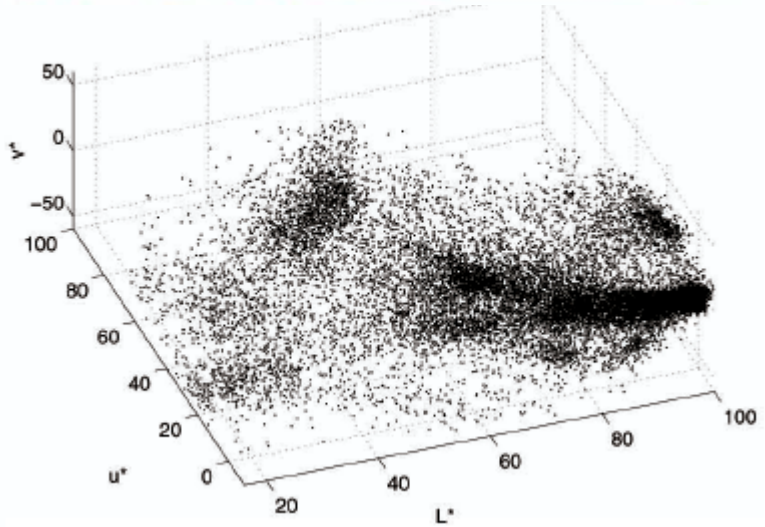


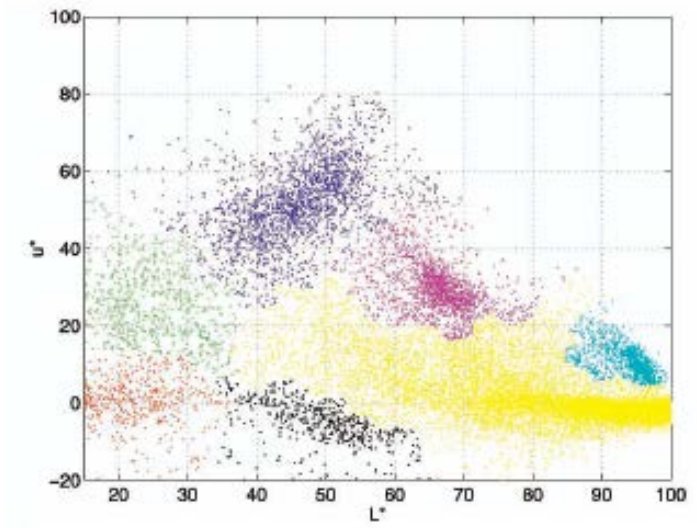
Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm

- Try to find *modes* of this non-parametric density



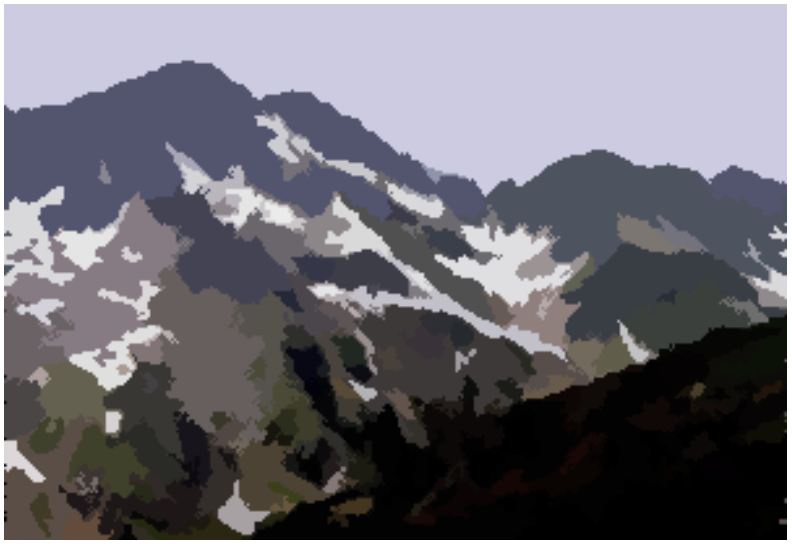




Mean shift clustering

- The mean shift algorithm seeks *modes* of the given set of points
  1. Choose kernel and bandwidth (kernel size for features K_f and position K_s)
  2. For each point:
     a) Center a window on that point
     b) Compute the mean of the data in the search window
     c) Center the search window at the new mean location
     d) Repeat (b,c) until convergence
  3. Assign points that lead to nearby modes (within width of K_f and K_s) to the same cluster

Mean shift segmentation results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean shift segmentation results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean-shift: other issues

- Speedups
  - Binned estimation – replace points within some "bin" by point at center with mass
  - Fast search of neighbors – e.g., k-d tree or approximate NN
  - Update all windows in each iteration (faster convergence)

- Other tricks
  - Use kNN to determine window sizes adaptively

- Lots of theoretical support

  D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean shift pros and cons

- Pros
  - Good general-purpose segmentation
  - Flexible in number and shape of regions
  - Robust to outliers
  - General mode-finding algorithm (useful for other problems such as finding most common surface normals)
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features
- When to use it
  - Oversegmentation
  - Multiple segmentations
  - Tracking, clustering, filtering applications
    - D. Comaniciu, V. Ramesh, P. Meer: *Real-Time Tracking of Non-Rigid Objects using Mean Shift*, *Best Paper Award*, IEEE Conf. Computer Vision and Pattern Recognition (CVPR'00), Hilton Head Island, South Carolina, Vol. 2, 142-149, 2000

Mean-shift reading

- Nicely written mean-shift explanation (with math)

http://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/

  - Includes .m code for mean-shift clustering

- Mean-shift paper by Comaniciu and Meer

http://www.caip.rutgers.edu/~comanici/Papers/MsRobustApproach.pdf

- Adaptive mean shift in higher dimensions

http://mis.hevra.haifa.ac.il/~ishimshoni/papers/chap9.pdf

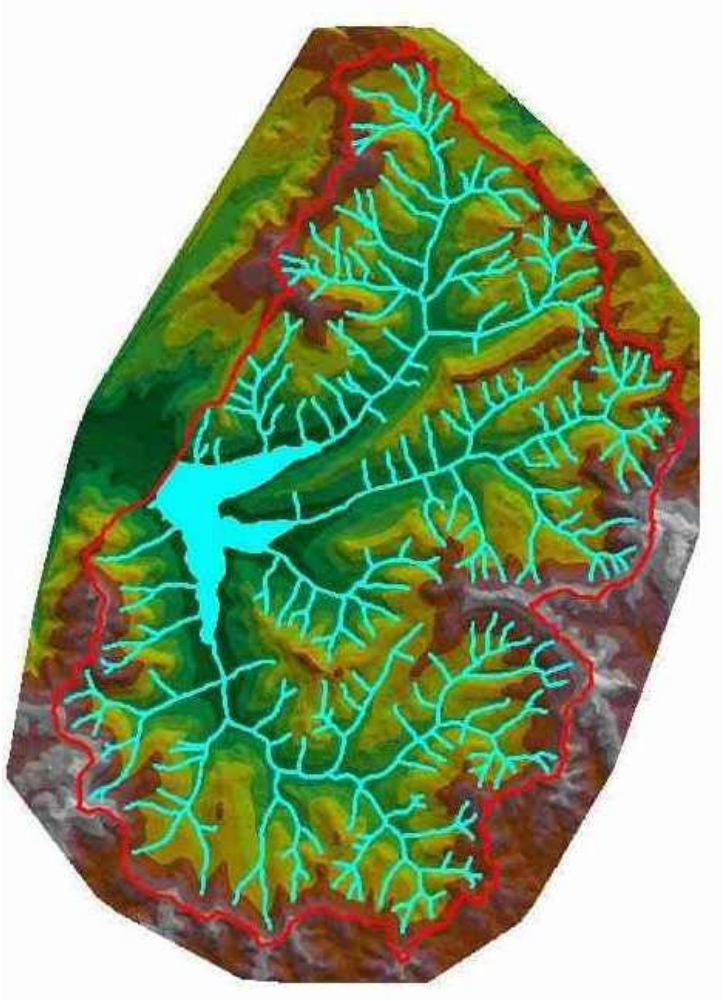
Image Over-Segmentation (Superpixels)

Superpixel algorithms

- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

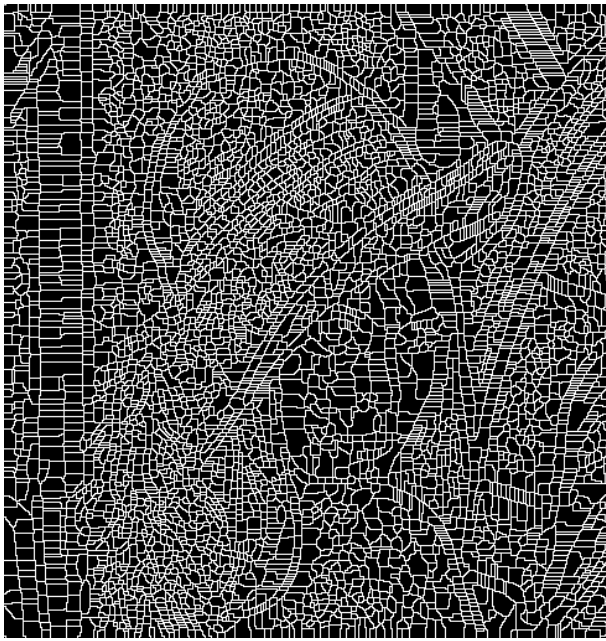
Watershed algorithm



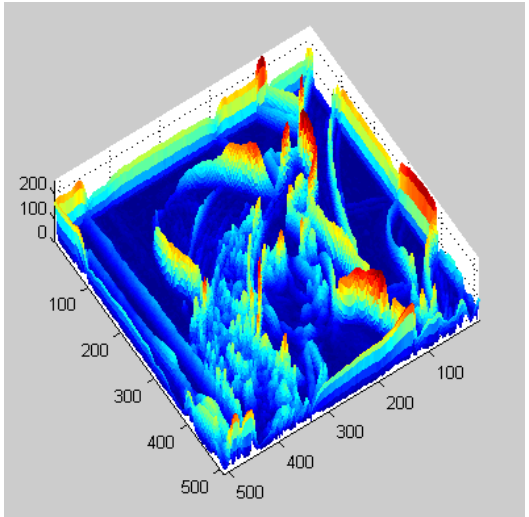
Watershed segmentation

Image

Gradient



Watershed boundaries



Meyer's watershed segmentation

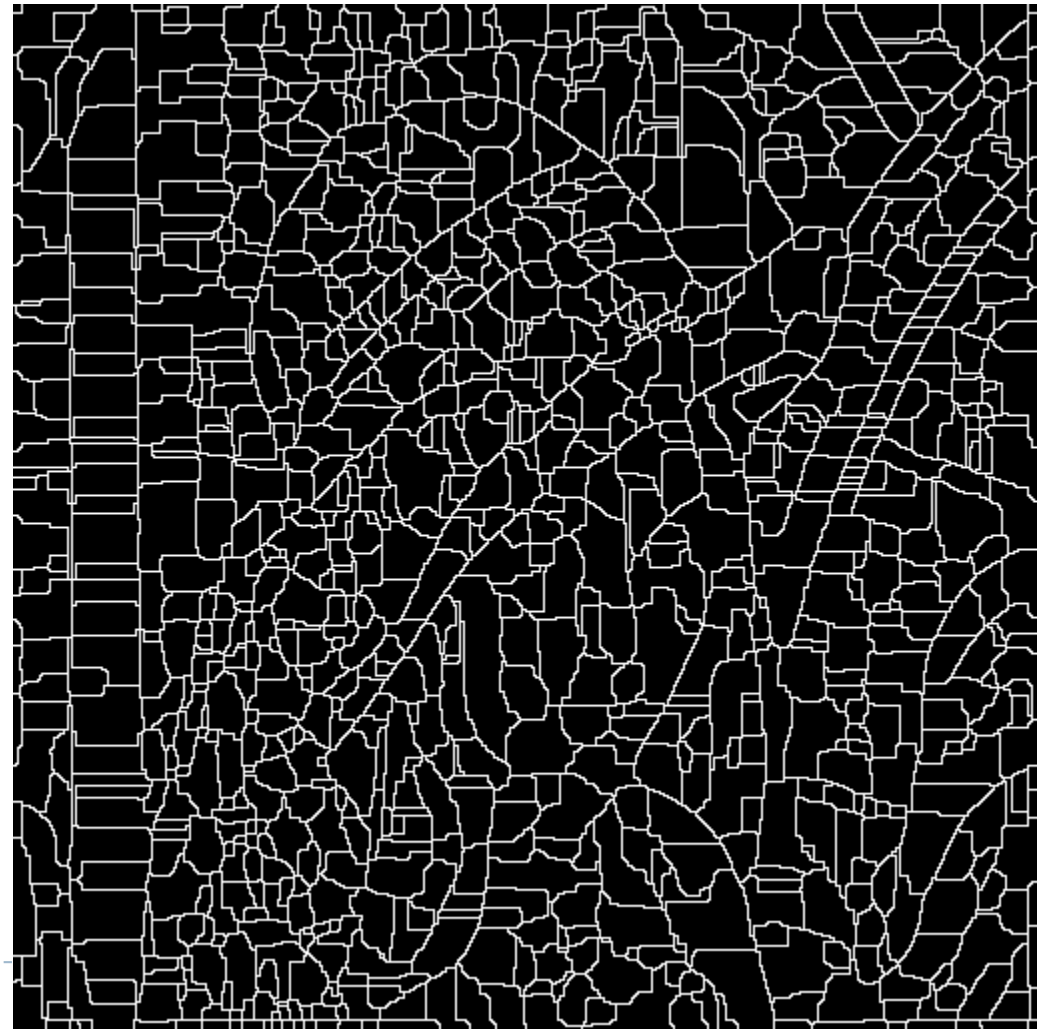
1. Choose local minima (in gradient image) as region seeds
2. Add neighbors to priority queue, sorted by value
3. Take top priority pixel from queue
    1. If all labeled neighbors have same label, assign that label to pixel
    2. Add all non-marked neighbors to queue
4. Repeat step 3 until finished (all remaining pixels in queue are on the boundary)

Matlab: seg = watershed(bnd_im)

[Meyer 1991]

Simple trick

- Use Gaussian or median filter to reduce number of regions



Watershed usage

- Use as a starting point for hierarchical segmentation
  - Ultra-metric contour map (Arbelaez 2006)

- Works with any soft boundaries
  - Pb (Berkeley contour detection) (w/o non-max suppression)
  - Canny (w/o non-max suppression)
  - Etc.

Watershed pros and cons

- Pros
  - Fast (< 1 sec for 512x512 image)
  - Preserves boundaries
- Cons
  - Only as good as the soft boundaries (which may be slow to compute)
  - Not easy to get variety of regions for multiple segmentations
- Usage
  - Good algorithm for superpixels, hierarchical segmentation

Superpixel algorithms

- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

Felzenszwalb and Huttenlocher: Graph-Based Segmentation

http://www.cs.brown.edu/~pff/segment/

Algorithm 1 *Segmentation algorithm.*

The input is a graph $G = (V, E)$, with n vertices and m edges. The output is a segmentation of V into components $S = (C_1, \ldots, C_r)$.

0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.

Felzenszwalb and Huttenlocher: Graph-Based Segmentation

http://www.cs.brown.edu/~pff/segment/

3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Felzenszwalb and Huttenlocher: Graph-Based Segmentation

+ Good for thin regions
+ Fast
+ Easy to control coarseness of segmentations
+ Can include both large and small regions
- Often creates regions with strange shapes
- Sometimes makes very large errors

Superpixel algorithms

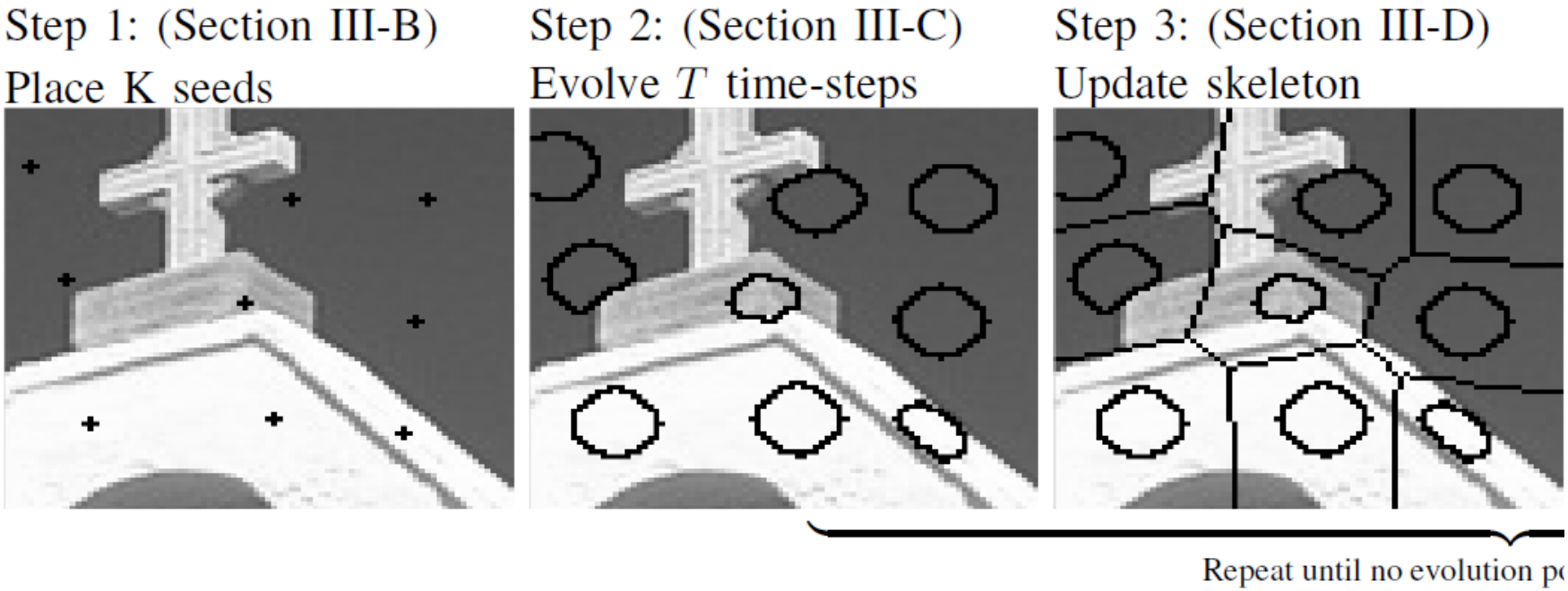
- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

Turbo Pixels: Levinstein et al. 2009

http://www.cs.toronto.edu/~kyros/pubs/09.pami.turbopixels.pdf

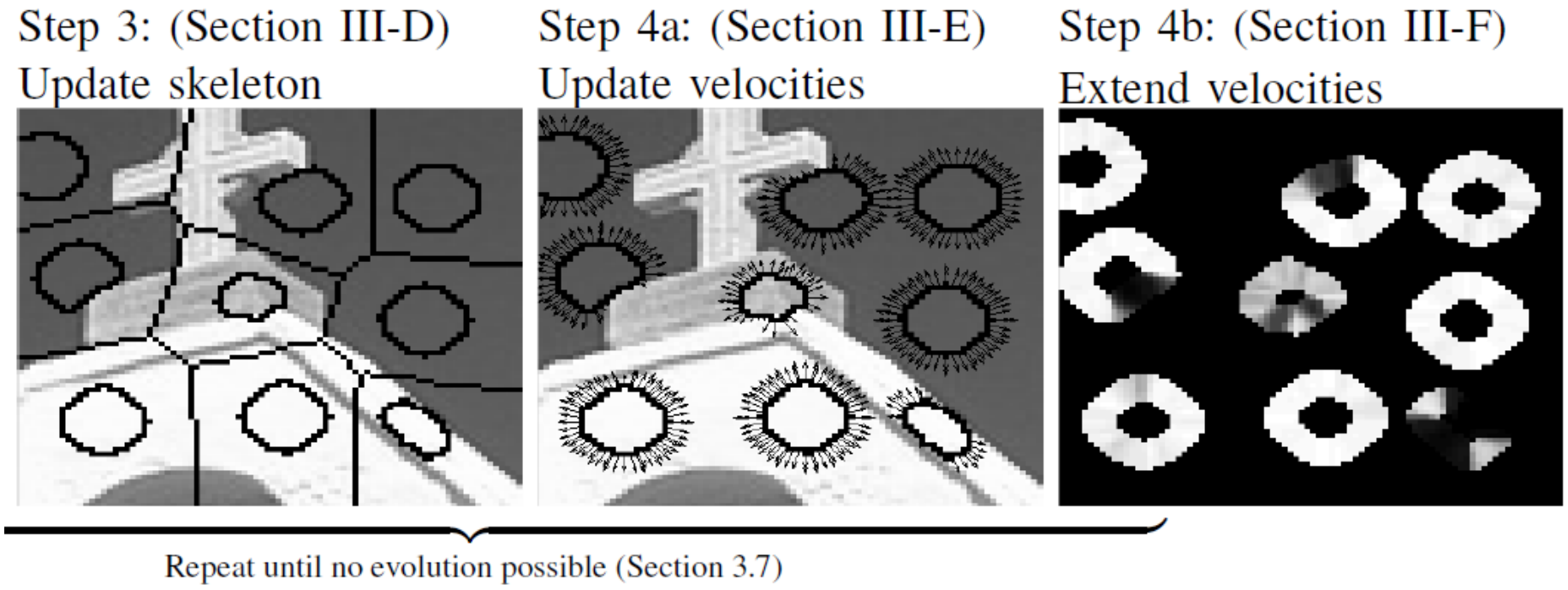
Tries to preserve boundaries like watershed but to produce more regular regions



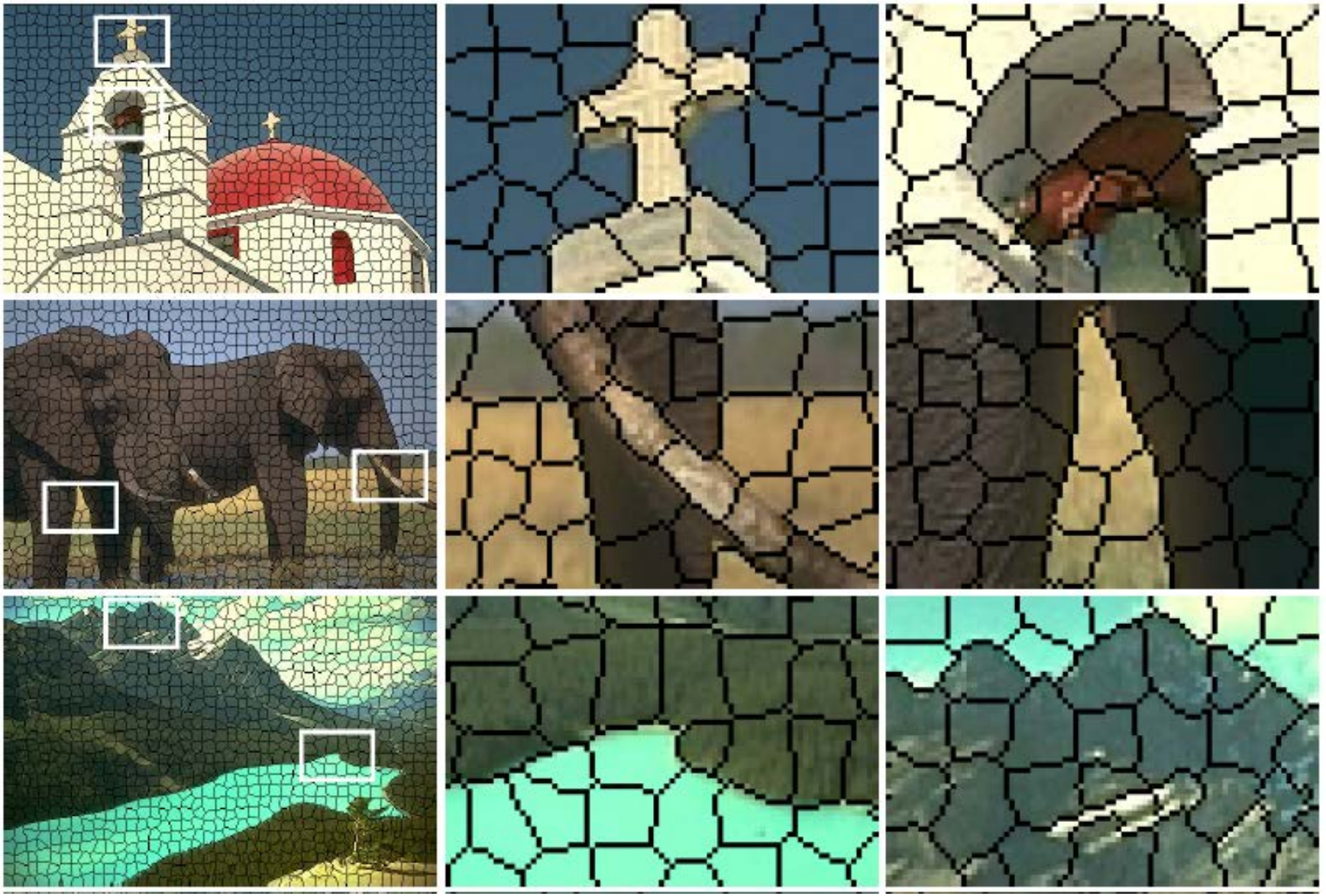
Turbo Pixels: Levinstein et al. 2009

http://www.cs.toronto.edu/~kyros/pubs/09.pami.turbopixels.pdf

Tries to preserve boundaries like watershed but to produce more regular regions



Turbo Pixels: Levinstein et al. 2009



Turbo Pixels: Levinstein et al. 2009

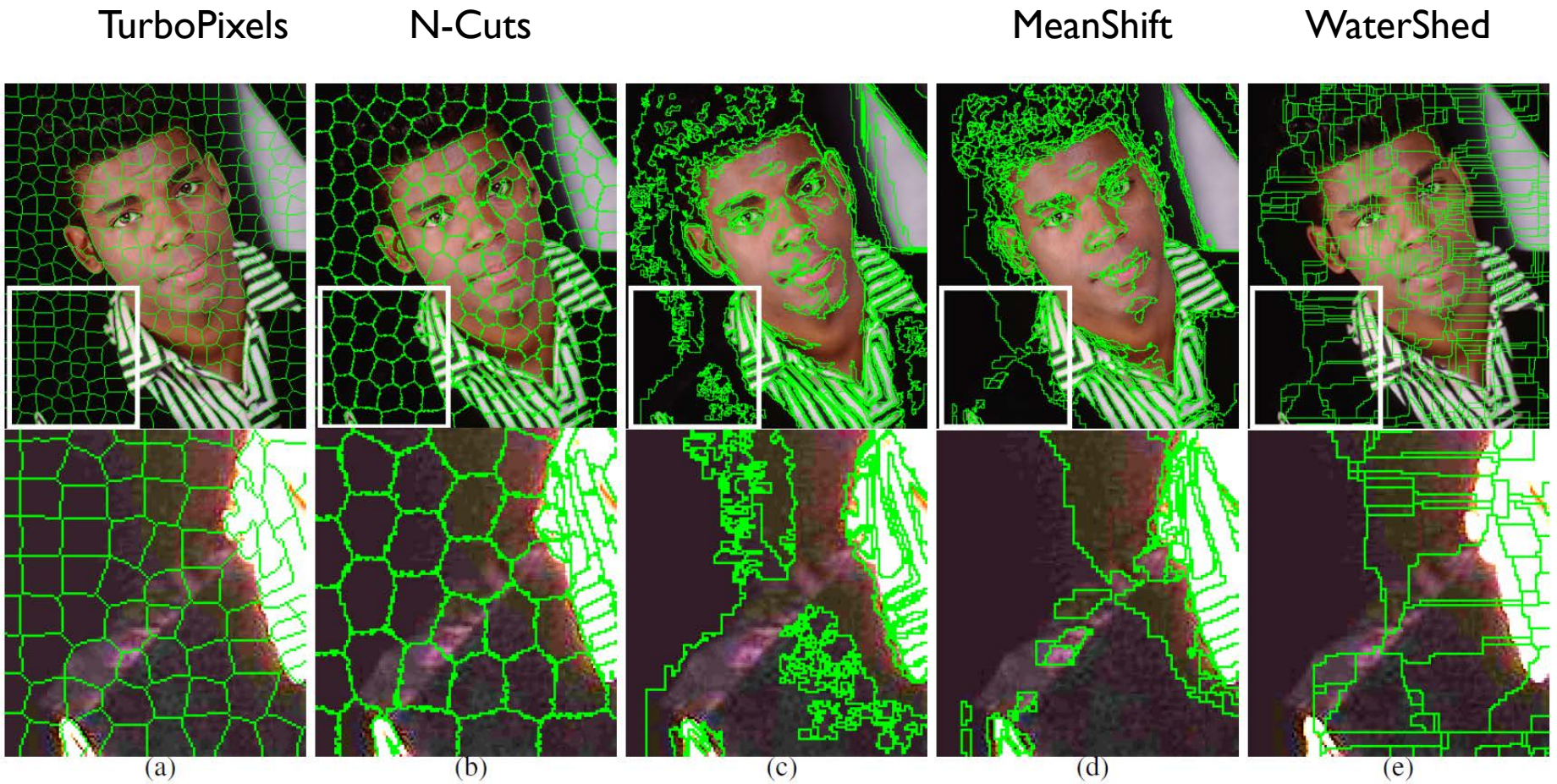


Fig. 1. Over-segmentations obtained with five algorithms: (a) TurboPixels (b) N-Cuts[27] (c) Local variation [6] (d) Mean-shift [3] (e) Watershed [7]. Each segmentation has (approximately) the same number of segments. The second row zooms in on the regions of interest defined by the white boxes.

Superpixel algorithms

- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

SLIC (Simple Linear Iterative Clustering) [Achanta et al. PAMI 2012]

http://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf

1. Initialize cluster centers on pixel grid in steps S
   - Features: Lab color, x-y position
2. Move centers to position in 3x3 window with smallest gradient
3. Compare each pixel to cluster center within $2S$ pixel distance and assign to nearest
4. Re-compute cluster centers as mean color/position of pixels belonging to each cluster
5. Stop when residual error is small

SLIC (Simple Linear Iterative Clustering)
[Achanta et al. PAMI 2012]

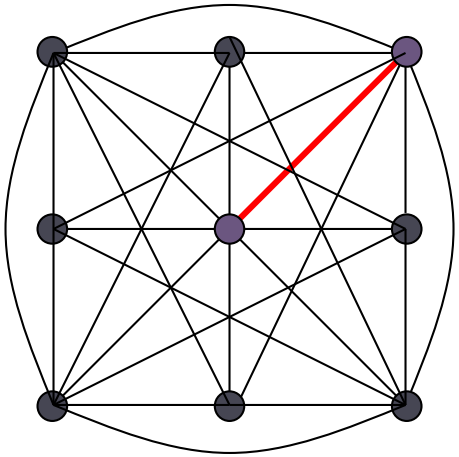
+ Fast 0.36s for 320x240
+ Regular superpixels
+ Superpixels fit boundaries
- May miss thin objects
- Large number of superpixels

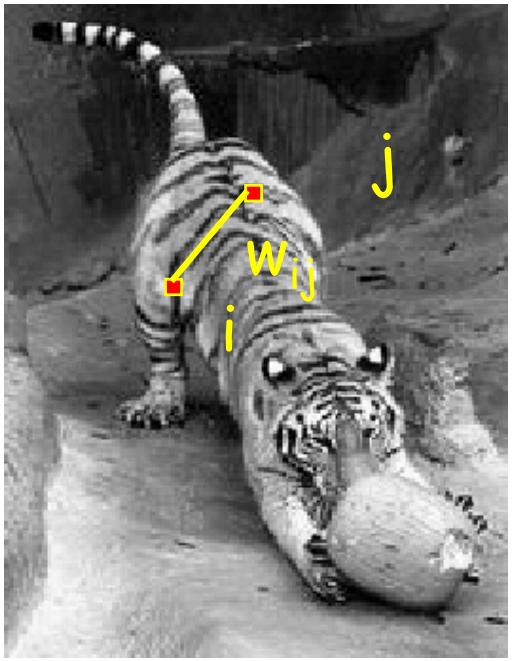
Graph Based Segmentation

- Normalized cuts
- Segmentation as energy minimization in a Markov random fields

Images as graphs

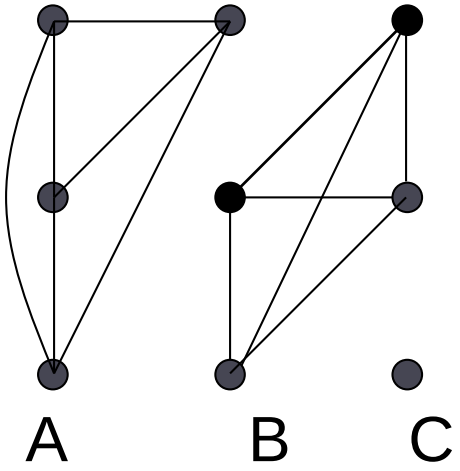
- Node for every pixel
- Edge between every pair of pixels (or every pair of "sufficiently close" pixels)
- Each edge is weighted by the ***affinity*** or ***similarity*** of the two nodes

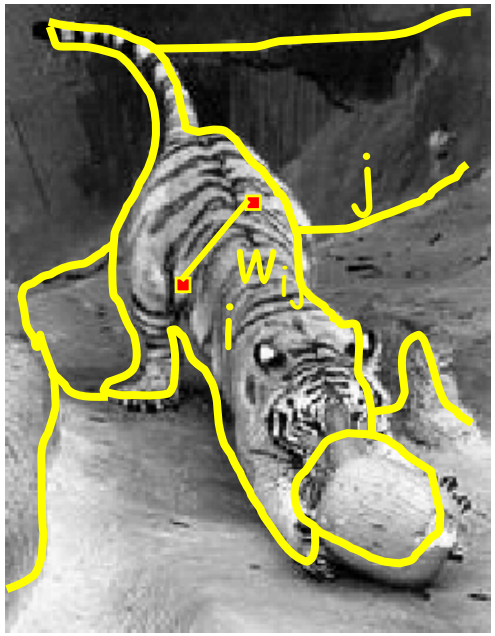




Segmentation by graph partitioning

- 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





Measuring affinity

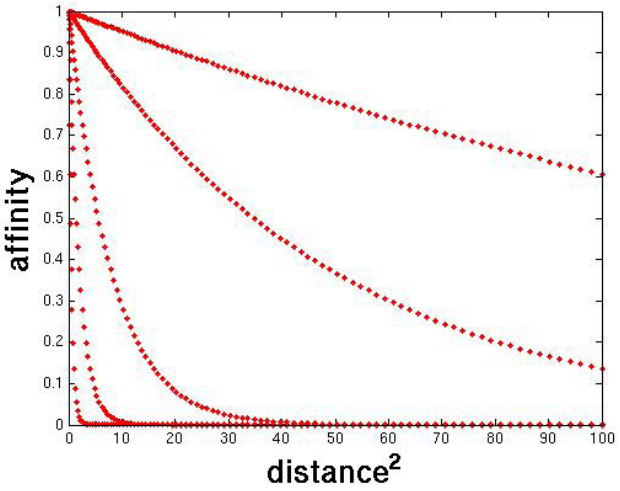
- Distance $aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\|x-y\|^2\right\}$
- Intensity $aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\|I(x)-I(y)\|^2\right\}$
- Color $aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\underbrace{dist\left(c(x),c(y)\right)}^{2}\right\}$

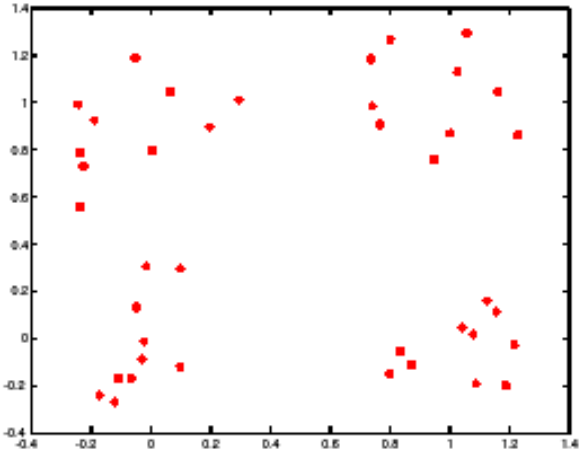
  (some suitable color space distance)
- Texture $aff(x,y) = \exp\left\{-\frac{1}{2\sigma_d^2}\underbrace{\|f(x)-f(y)\|}^{2}\right\}$

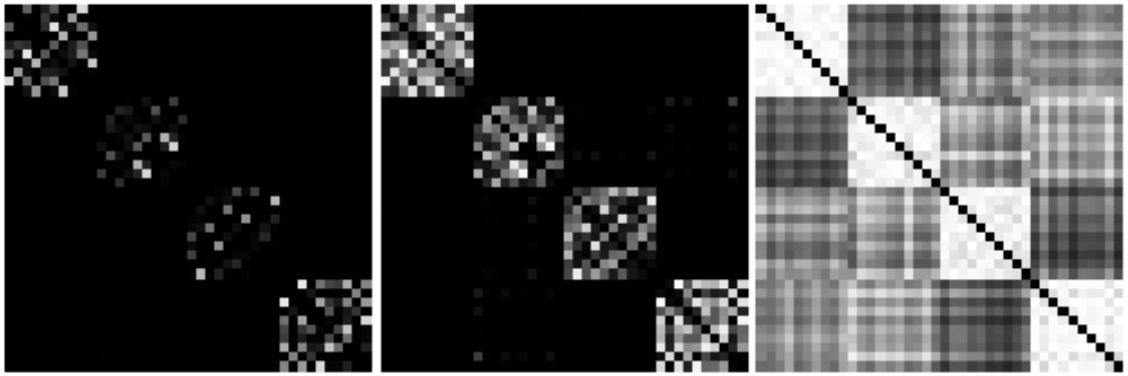
  (vectors of filter outputs)

Scale affects affinity

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

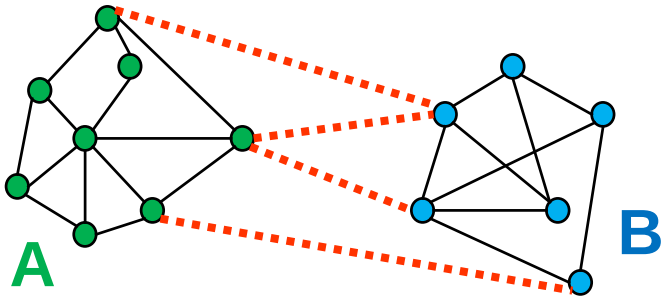






Graph cut

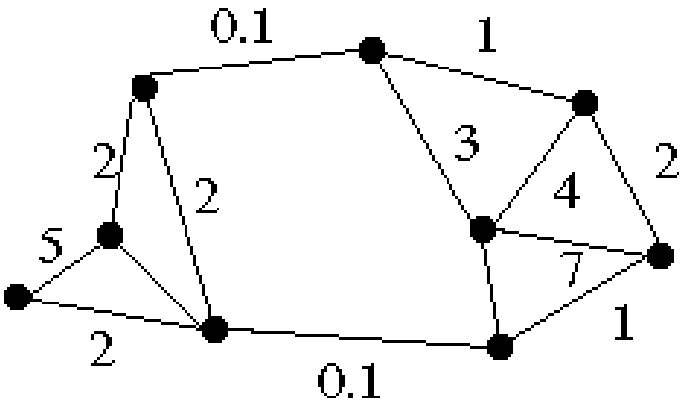
- Set of edges whose removal makes a graph disconnected
- **Cost of a cut: sum of weights of cut edges**
- A graph cut gives us a segmentation
  - What is a "good" graph cut and how do we find one?

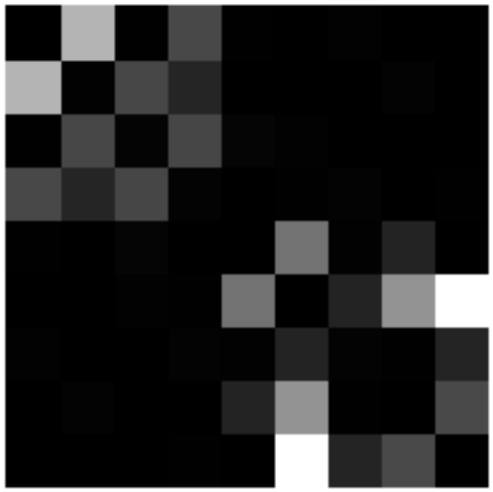


Minimum cut

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

Minimum cut example

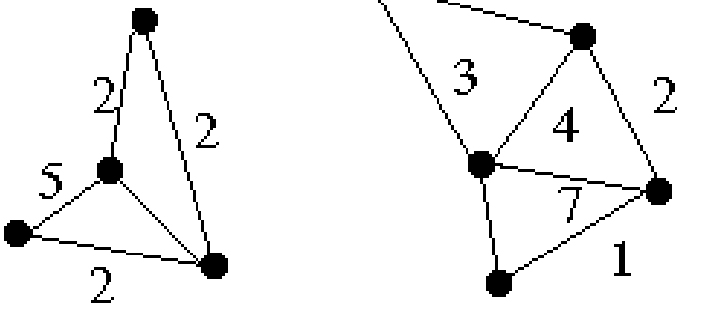


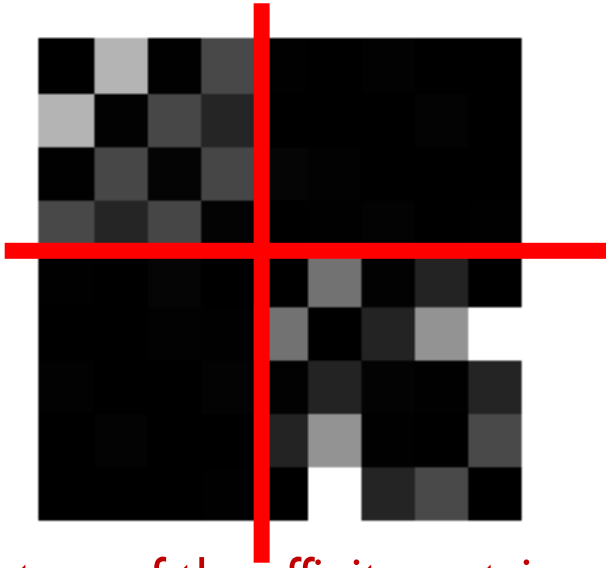


Minimum cut

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

Minimum cut example

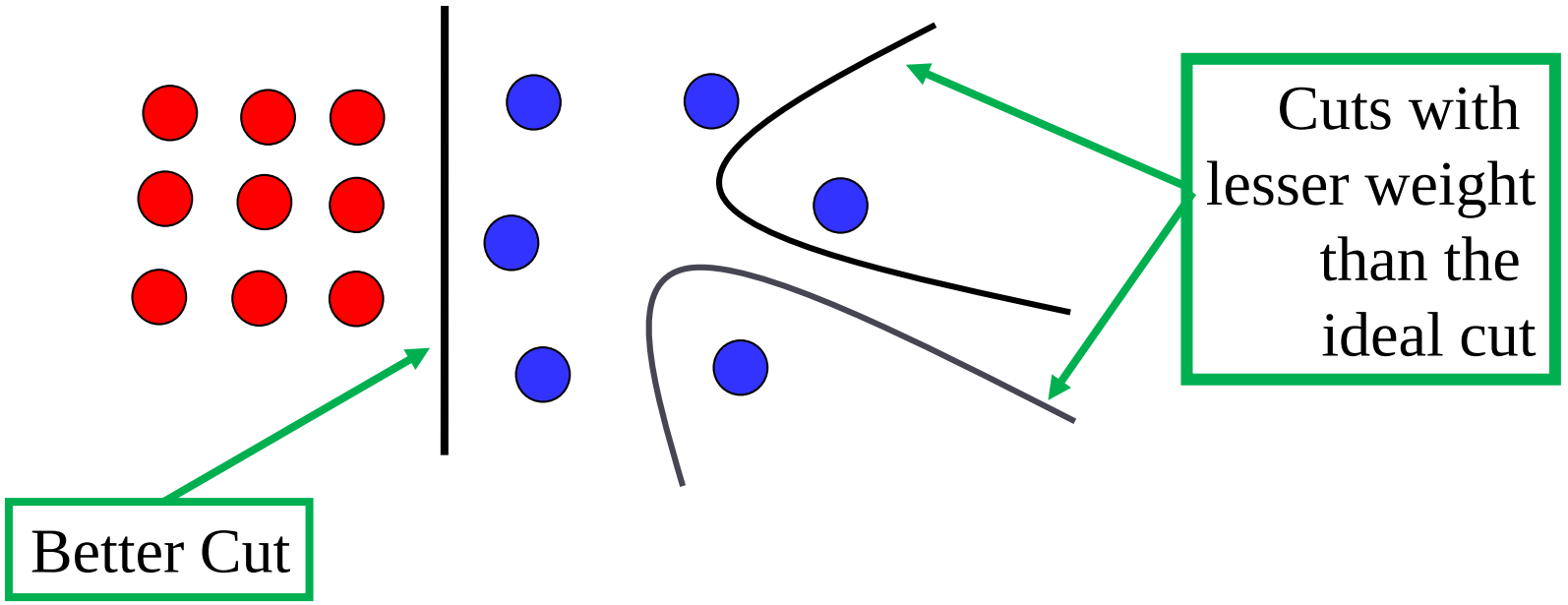




- The cut is defined by the block diagonal structure of the affinity matrix.
- Can this be generalized?

Minimum cut drawback

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



Normalized cut (Ncut)

- Normalize the cost by the weights:
  - Within each cluster
  - Between clusters

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

Normalized cut (Ncut)

- Normalize the cost by the weights:
  - Within each cluster
  - Between clusters

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

$assoc(A,V)$ = sum of weights of all edges **in *A***

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

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

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

$$D(i, i) = \Sigma_j W(i, j)$$

- Define the **cluster *A* vector**:

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

- Then the normalized cut cost can be written as

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

Proof required! see paper

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:
  - R*elax y* to take on arbitrary values, then solved by a *generalized eigenvalue problem*

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

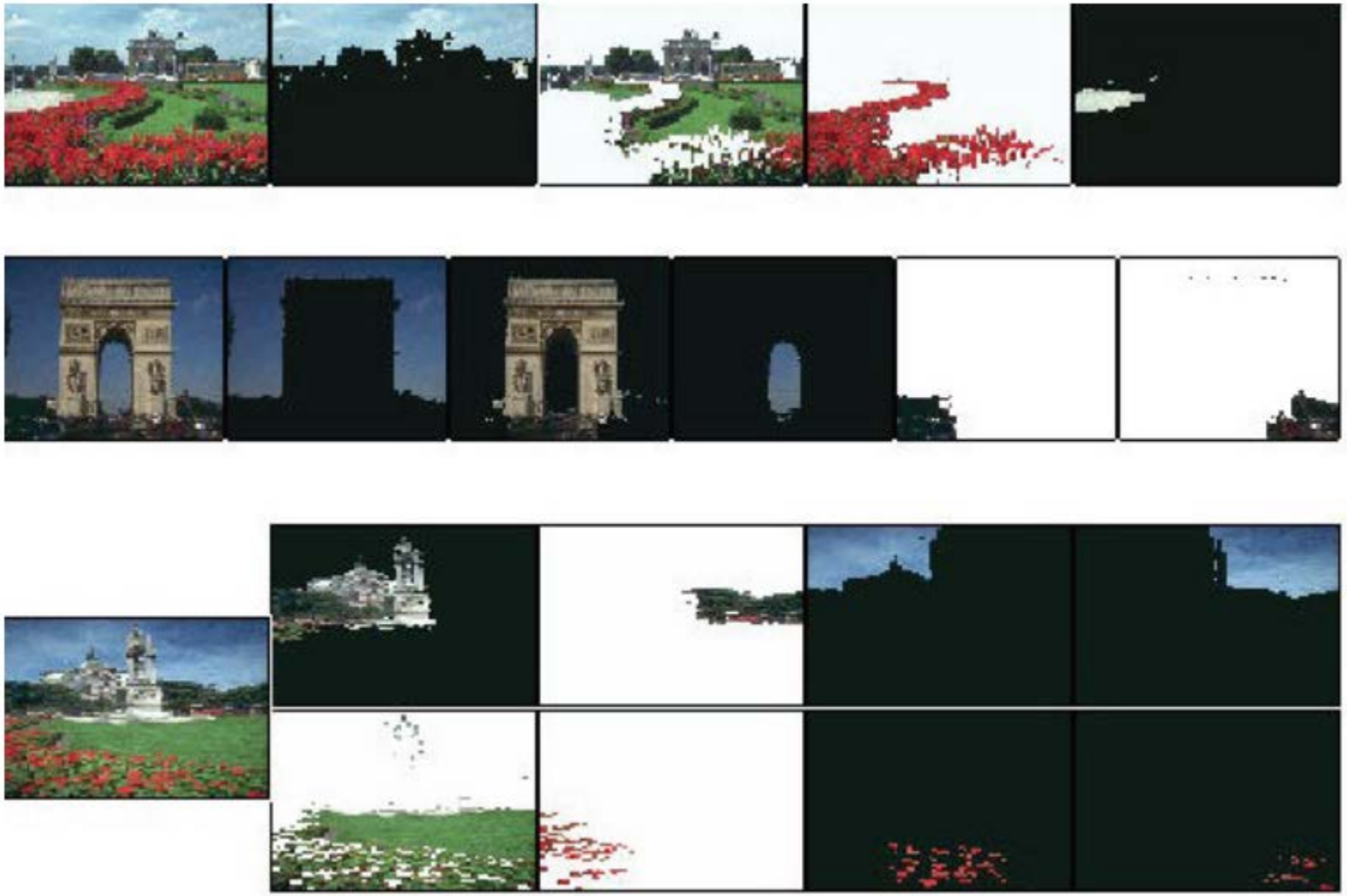
  - The solution *y* is given by the eigenvector corresponding to the second smallest eigenvalue
  - Continuous results need to be converted into discrete ➔ threshold

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 $D(i, i) = \Sigma_j W(i, j)$
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

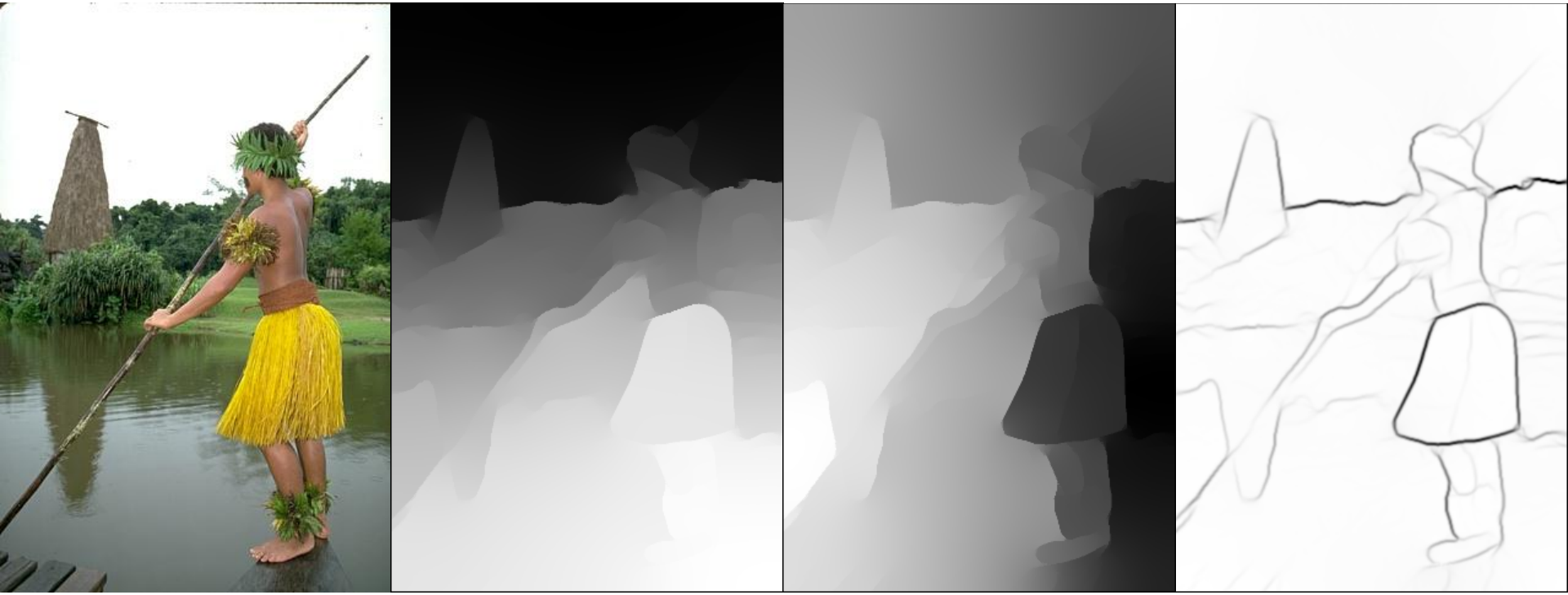
Results: Berkeley Segmentation Engine

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

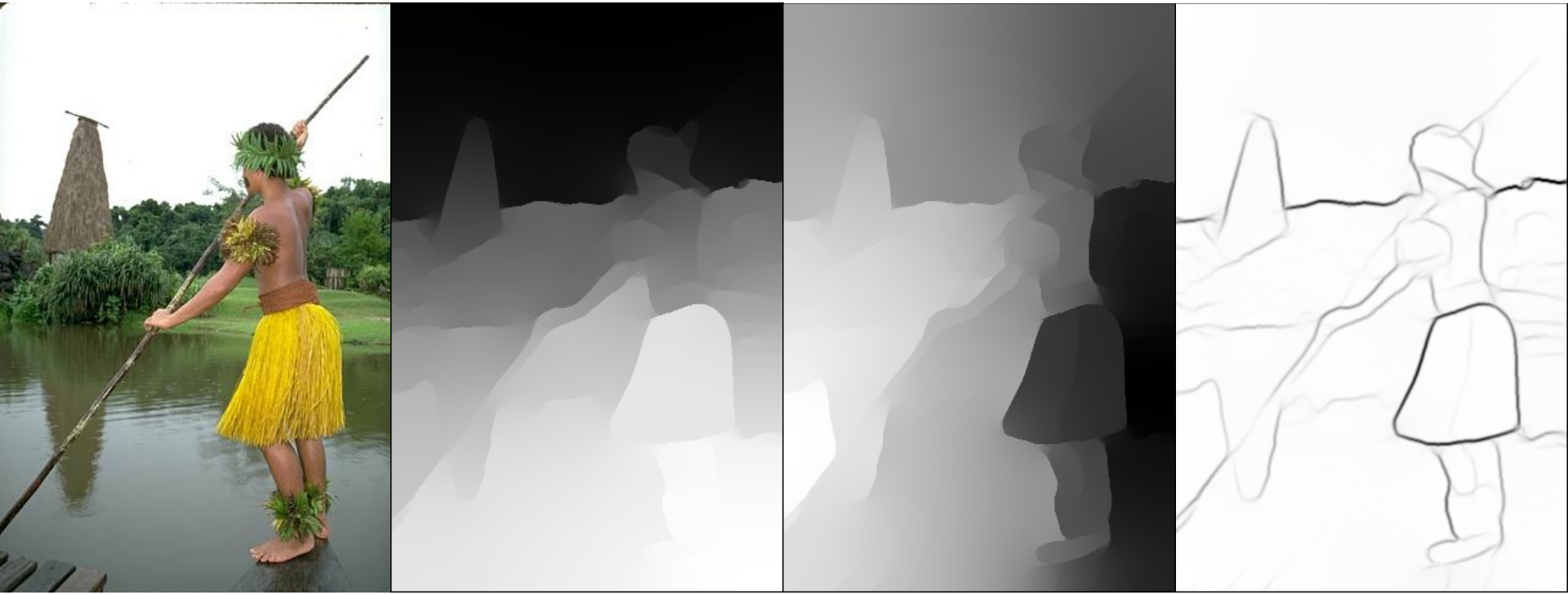
Normalized cuts: Pros and cons

- 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

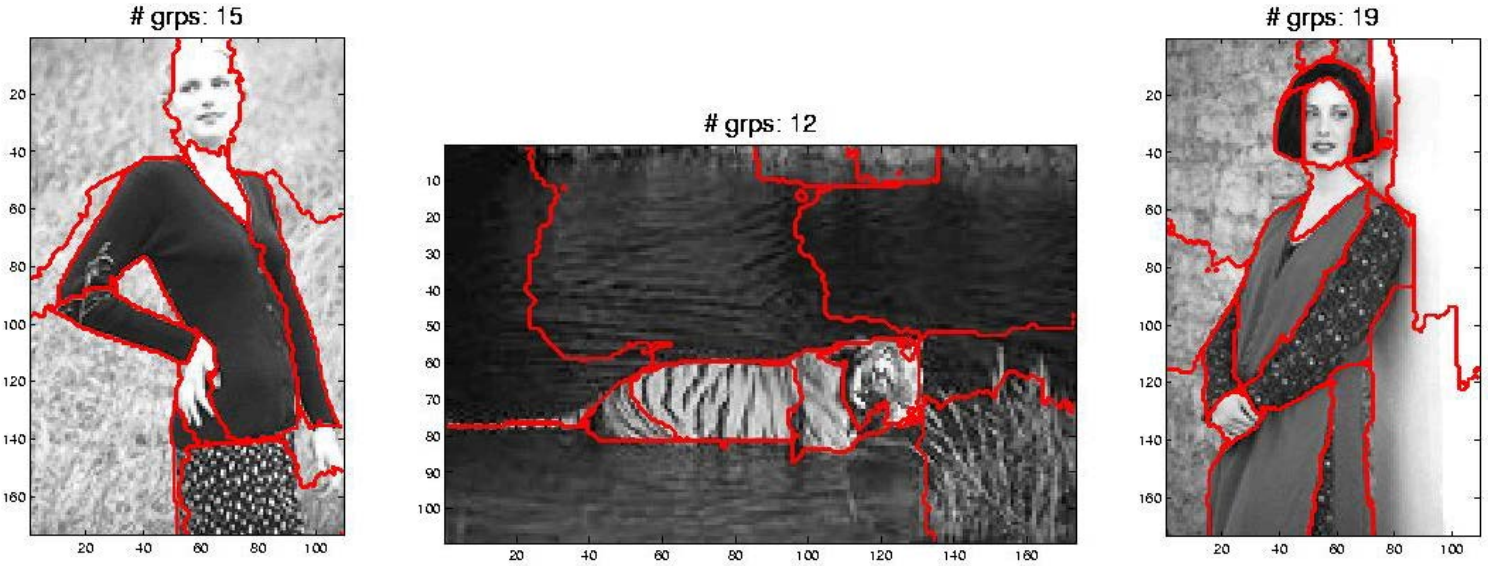


Eigenvectors carry contour information



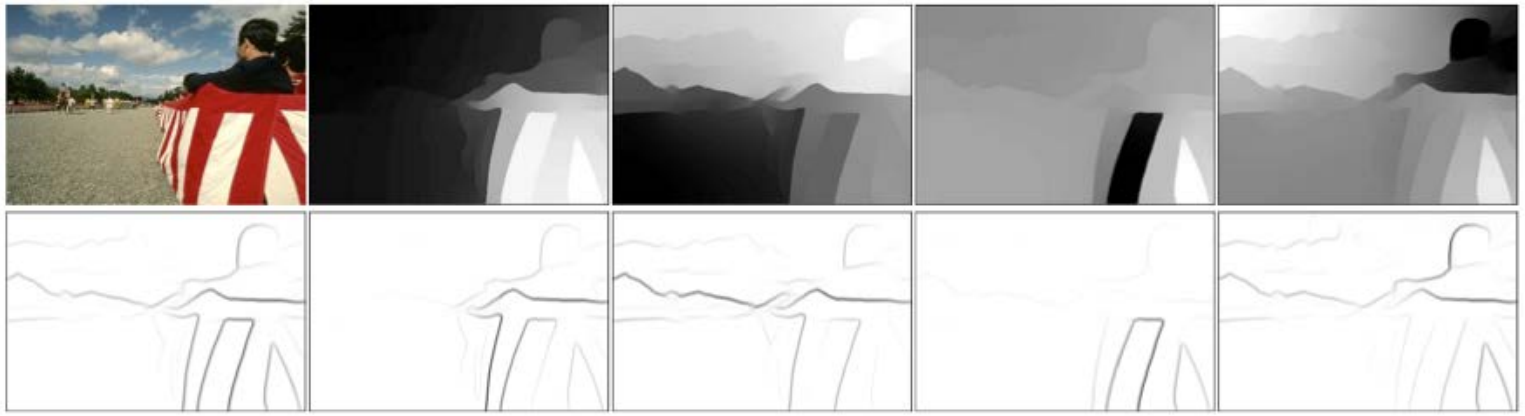
Avoiding Ncut drawback

- Do not try to find regions from the eigenvectors

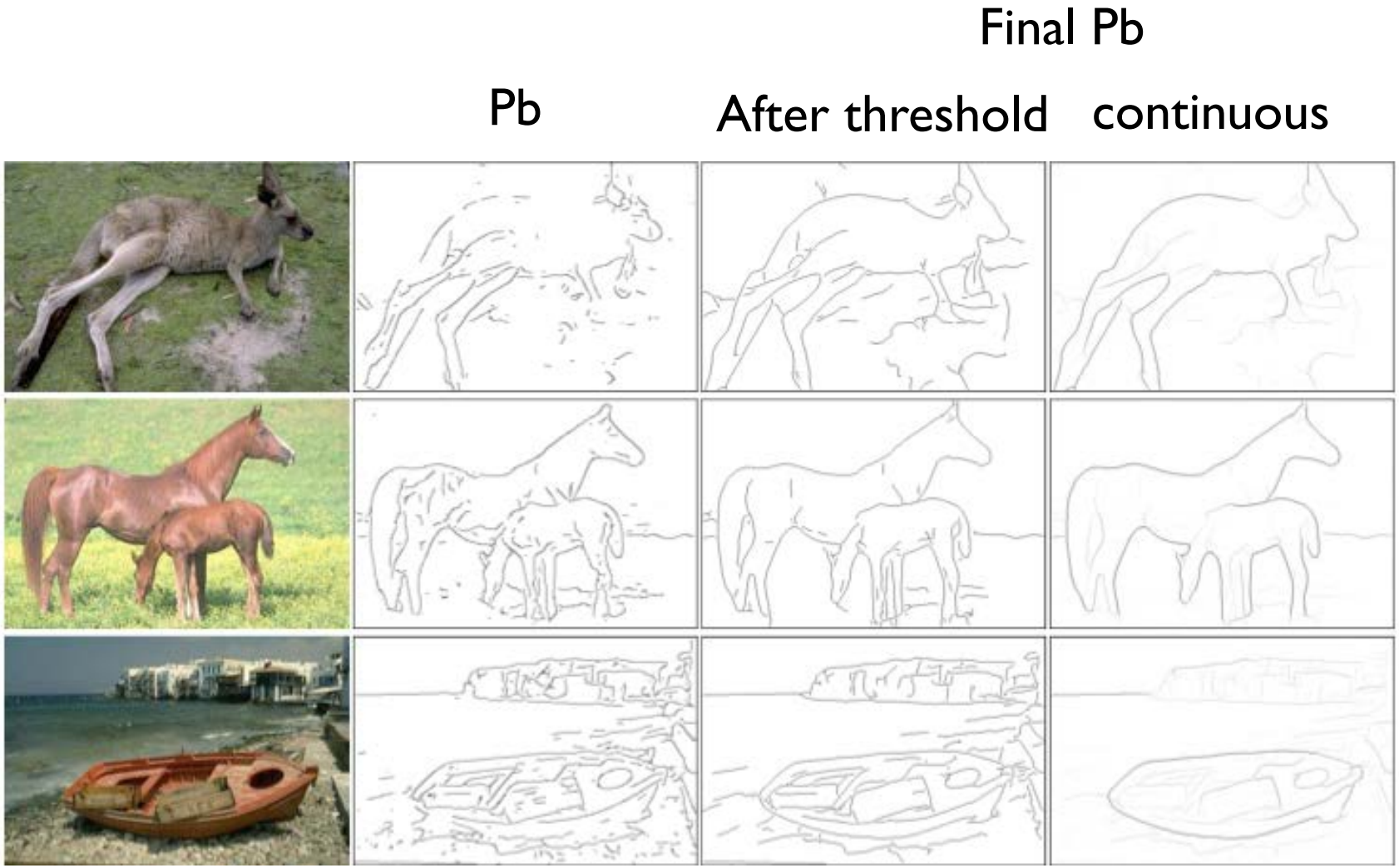


Avoiding Ncut drawback

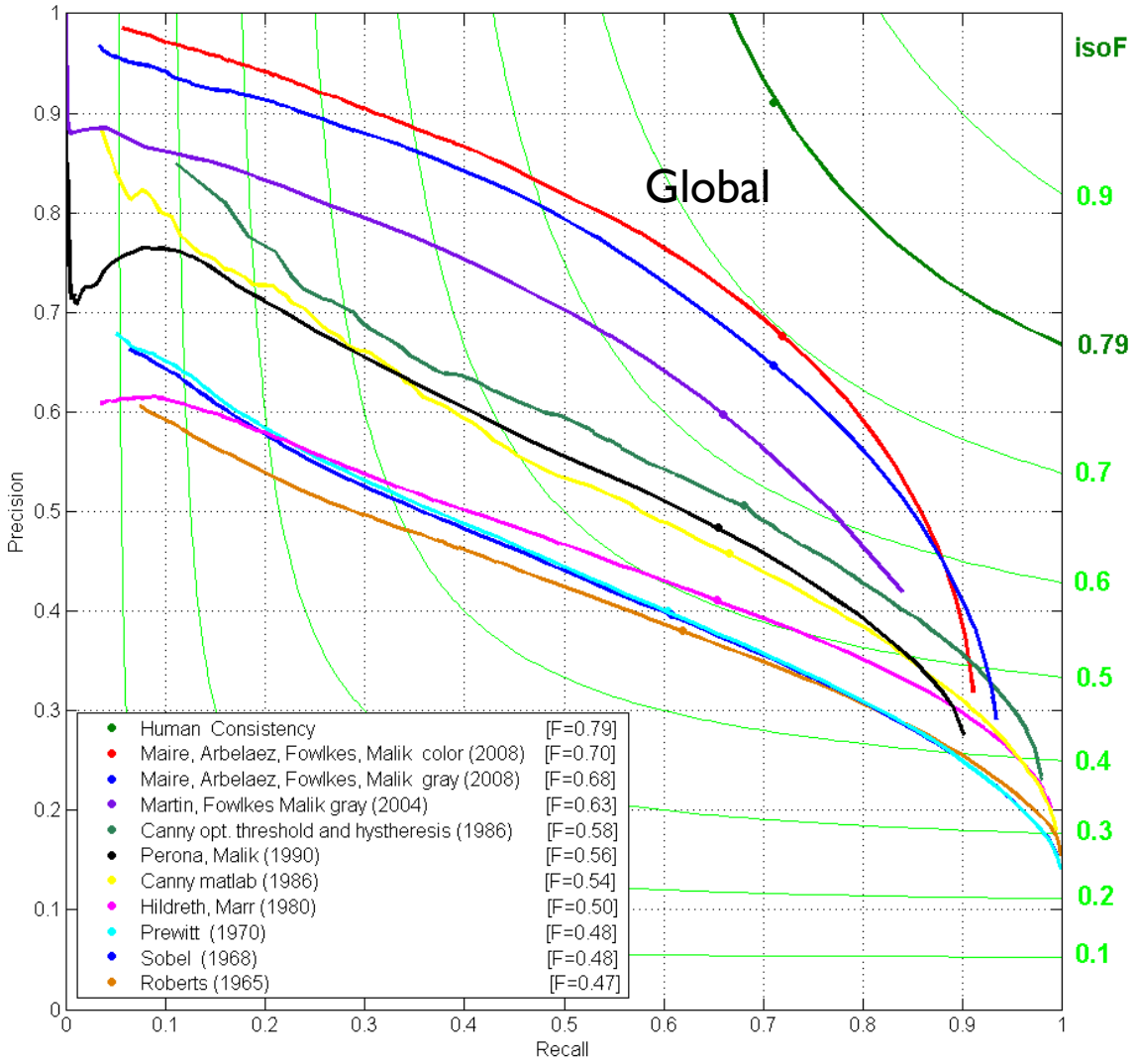
- Do not try to find regions from the eigenvectors
- Key idea:
  - Reshape eigenvectors into images
  - Compute edge probability P_b on eigenvector images
  - Final edge probability is the sum of all responses



Example results



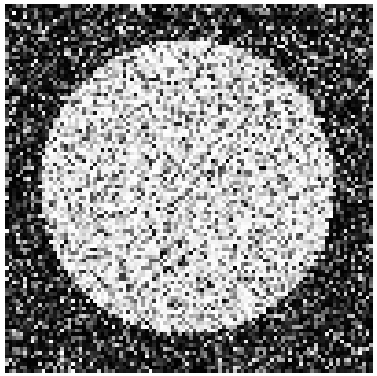
Contour detection ~2008 (color)



Graph Based Segmentation

- Normalized cuts
- Segmentation as energy minimization in a Markov random fields

Segmentation as energy minimization



P(foreground | image)

Normalizing constant called "partition function"

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

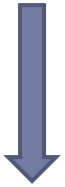
Labels to be predicted

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

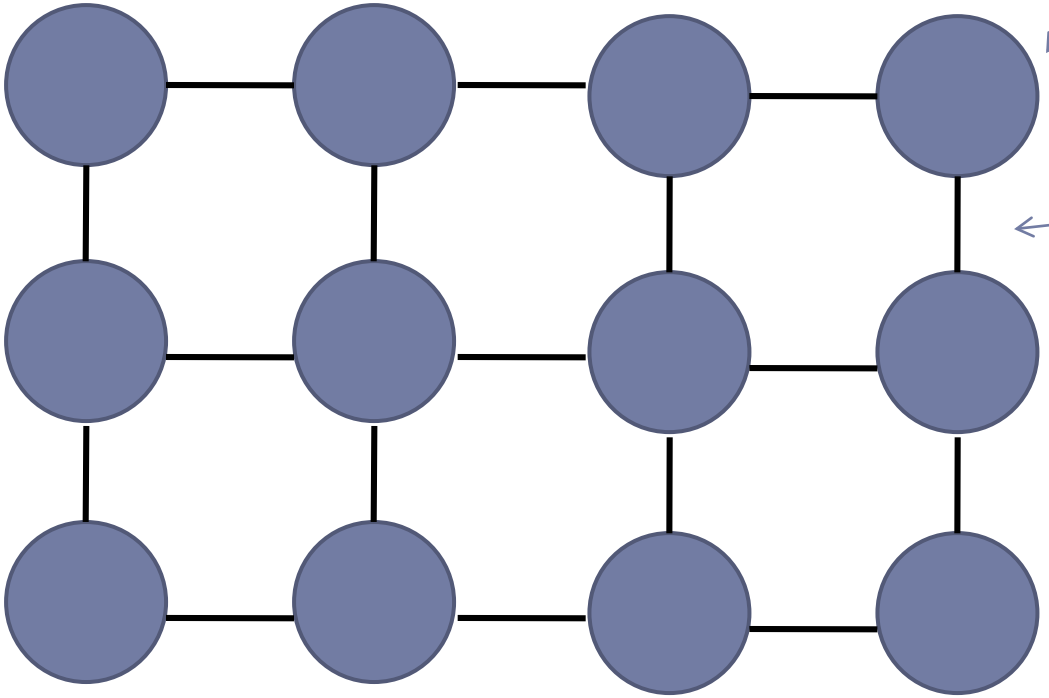
Cost of pairwise assignment y_i, y_j

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



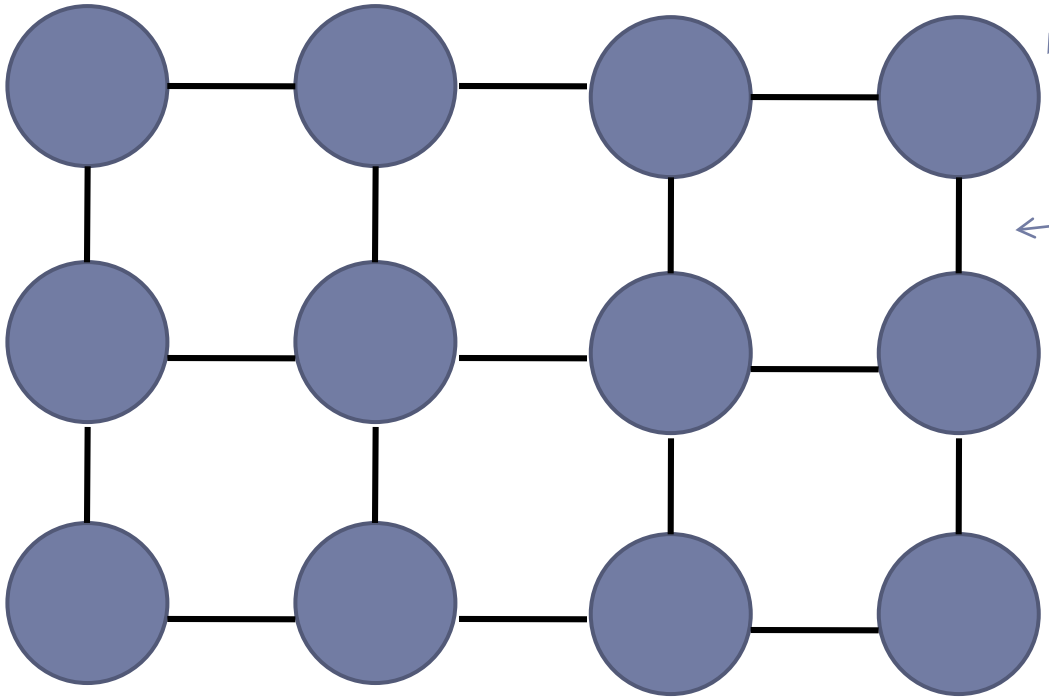
Cost to assign a label to each pixel

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)

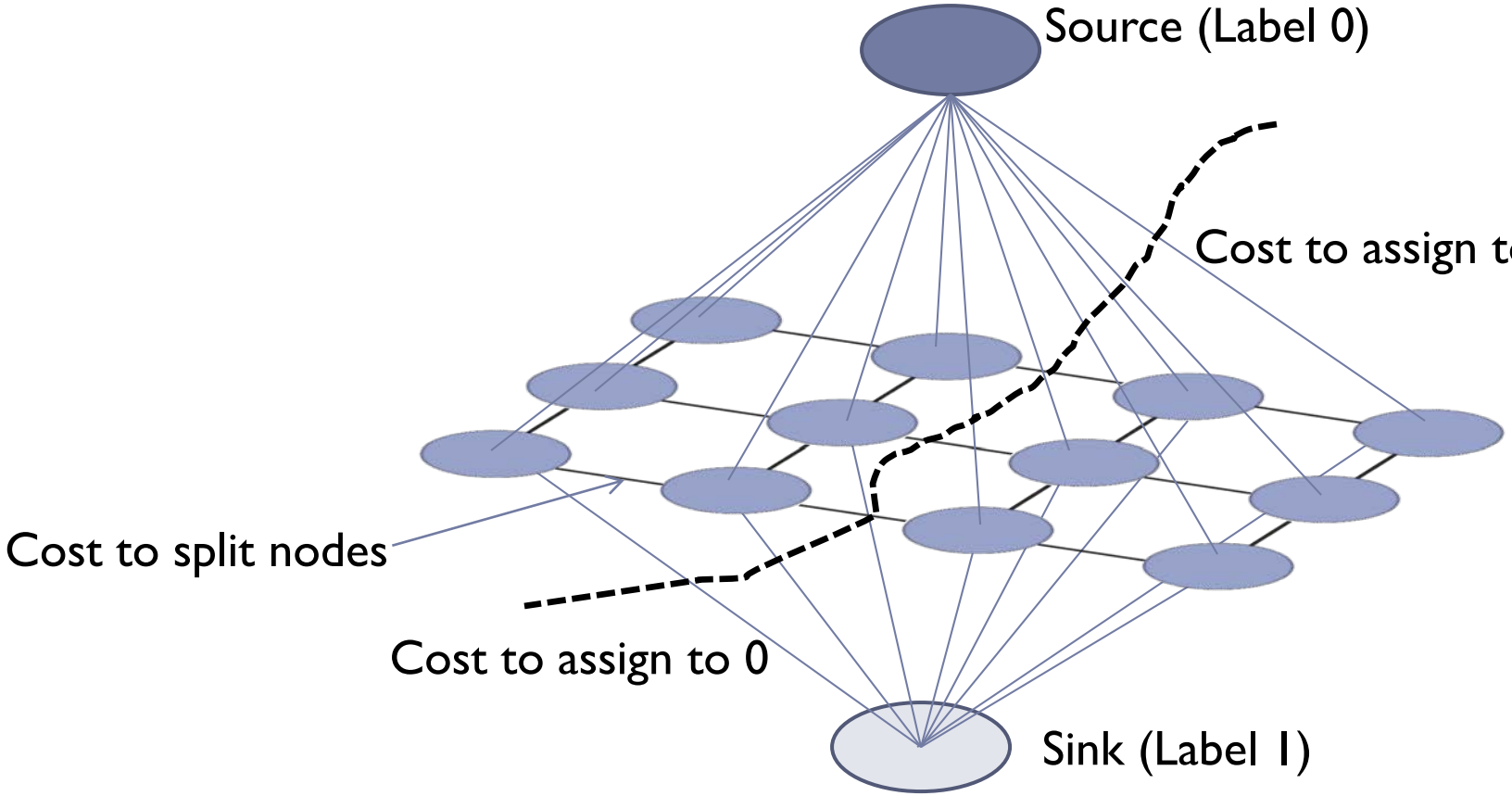
Pairwise Potential

	0	1
0	0	K
1	K	0

K>0

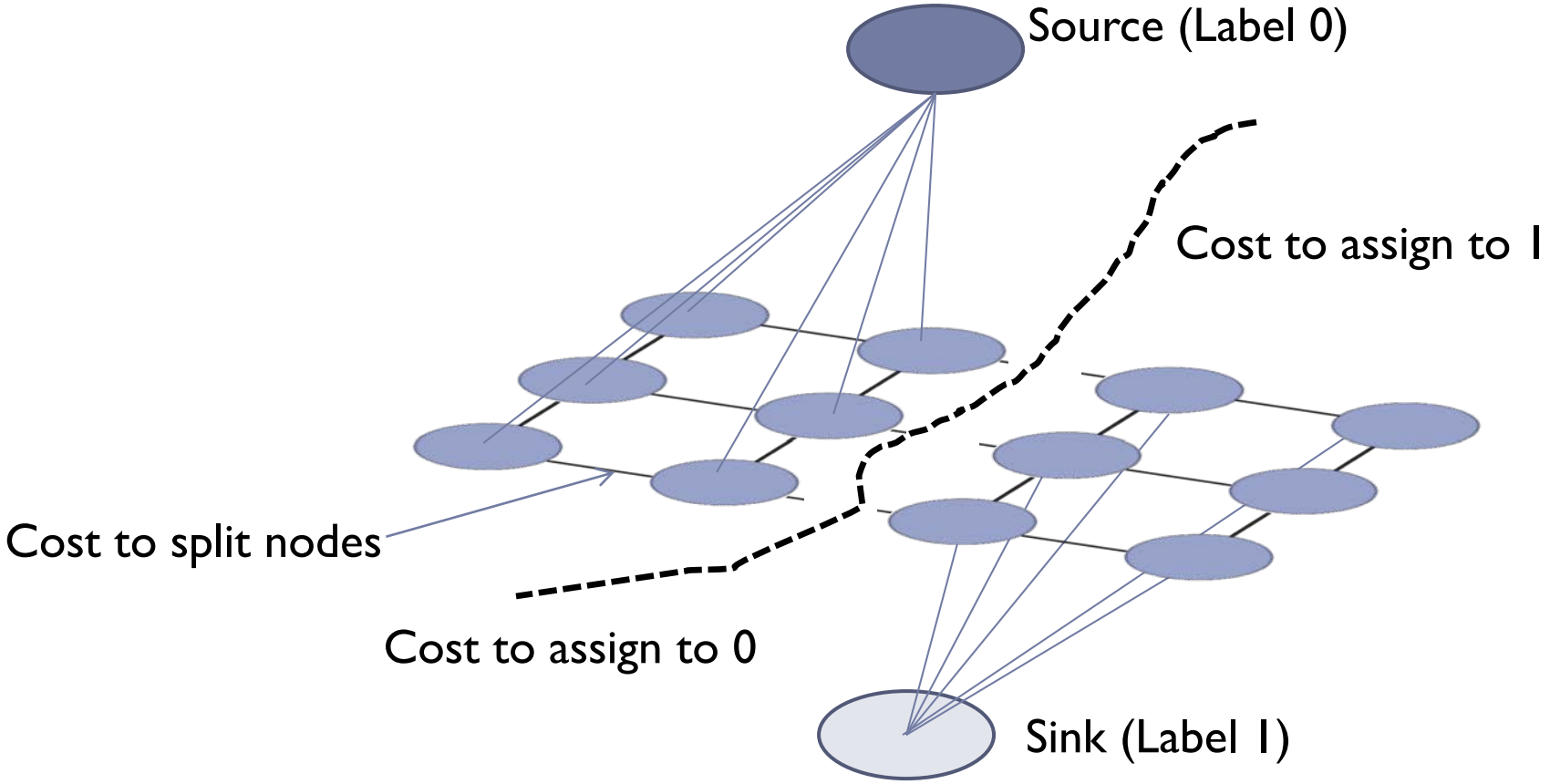
$$Energy(\mathbf{y};\theta,data) = \sum_i \psi_1(y_i;\theta,data) + \sum_{i,j\in edges} \psi_2(y_i,y_j;\theta,data)$$

Solving MRFs with graph cuts



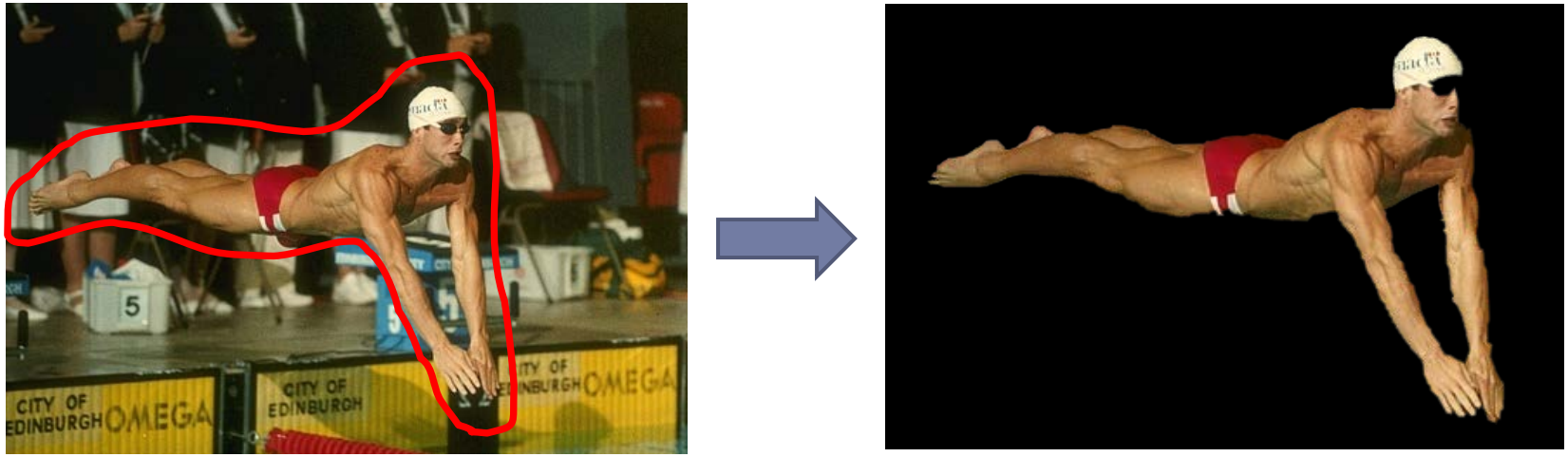
$$Energy(\mathbf{y};\theta,data)=\sum_i \psi_1(y_i;\theta,data)+\sum_{i,j\in edges}\psi_2(y_i,y_j;\theta,data)$$

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

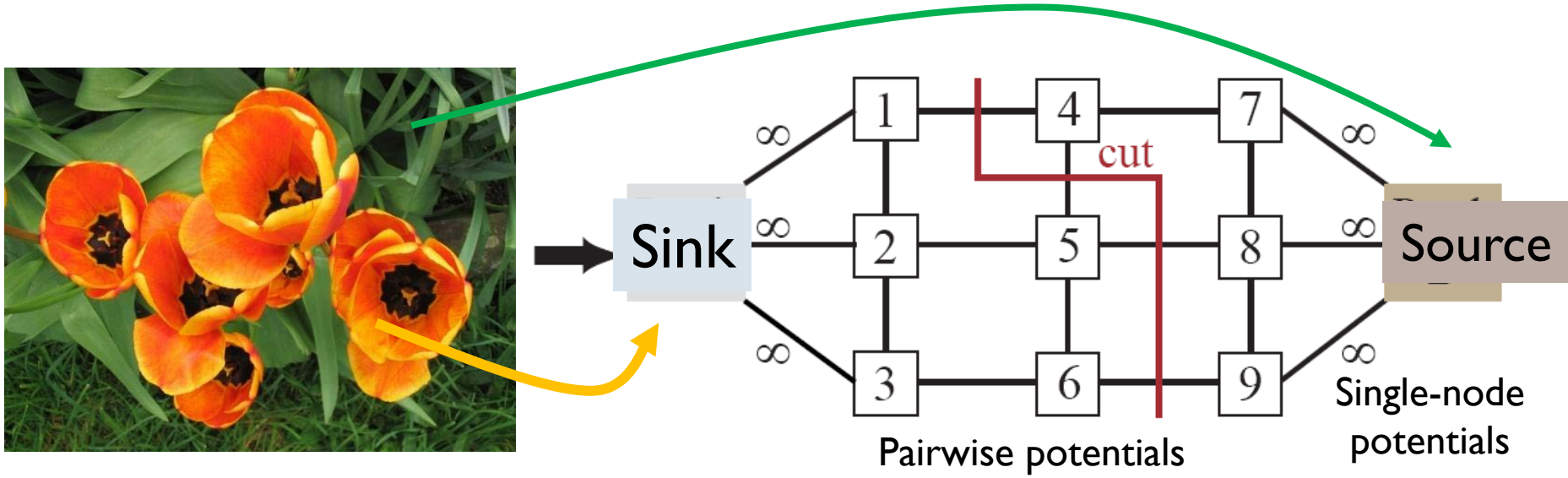


User provides rough indication of foreground region.

Goal: Automatically provide a pixel-level segmentation.

GrabCut

- Convert MRF into source-sink graph



- Minimun cost can be computed in polynomial time

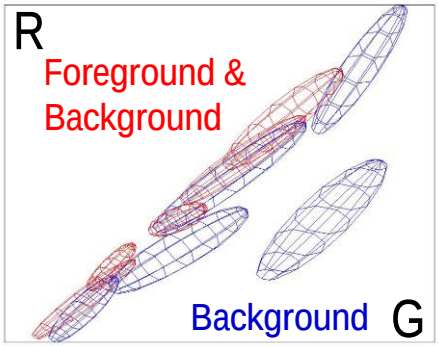
[Kwatra et al., SIGGRAPH 2003]

Binary segmentation as energy minimization

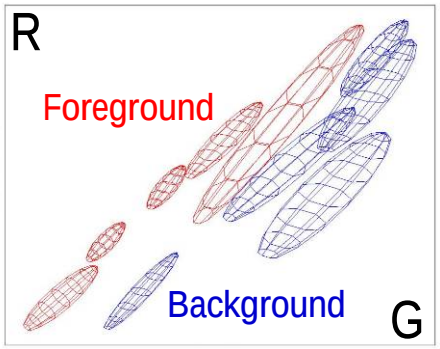
$$E(L) = E_d(L) + \lambda E_s(L)$$

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

Colour Model



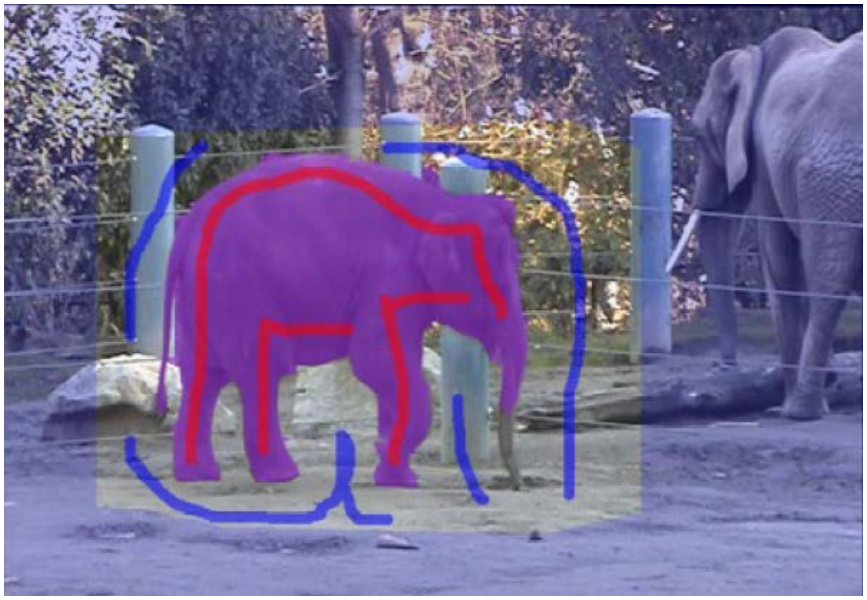
Iterated
graph cut

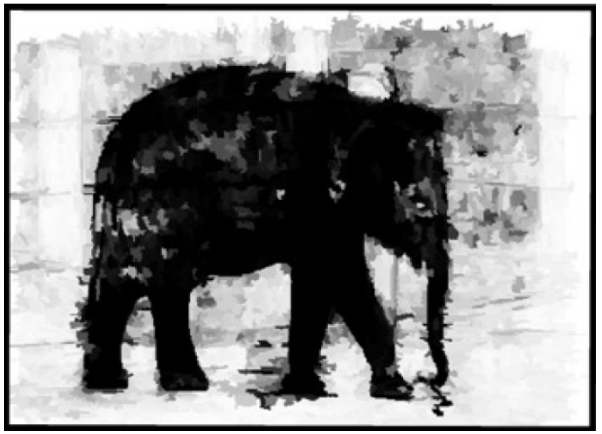


Gaussian Mixture Model (typically 5-8 components)

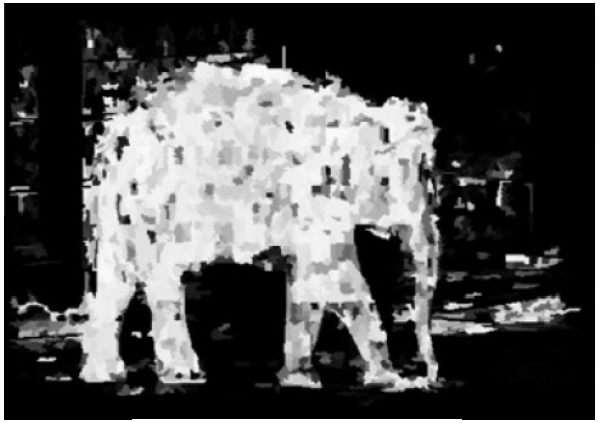
Source: Rother

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





$C'(x, y, 0)$



$C'(x, y, 1)$

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

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

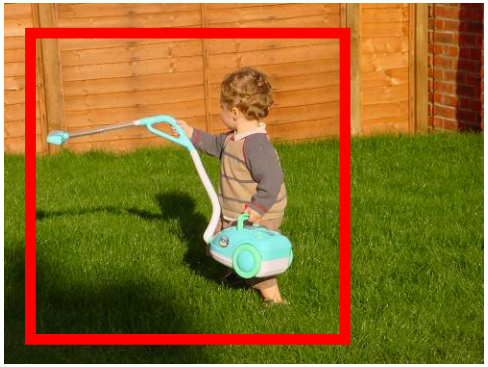
3. Define pairwise potentials

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

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

GrabCut

Grabcut [Rother et al., SIGGRAPH 2004]



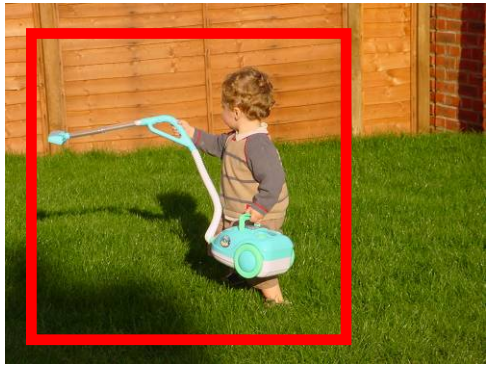



GrabCut

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

Reported results

Easier examples

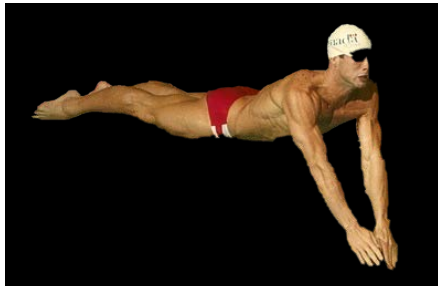


More difficult Examples

Camouflage &

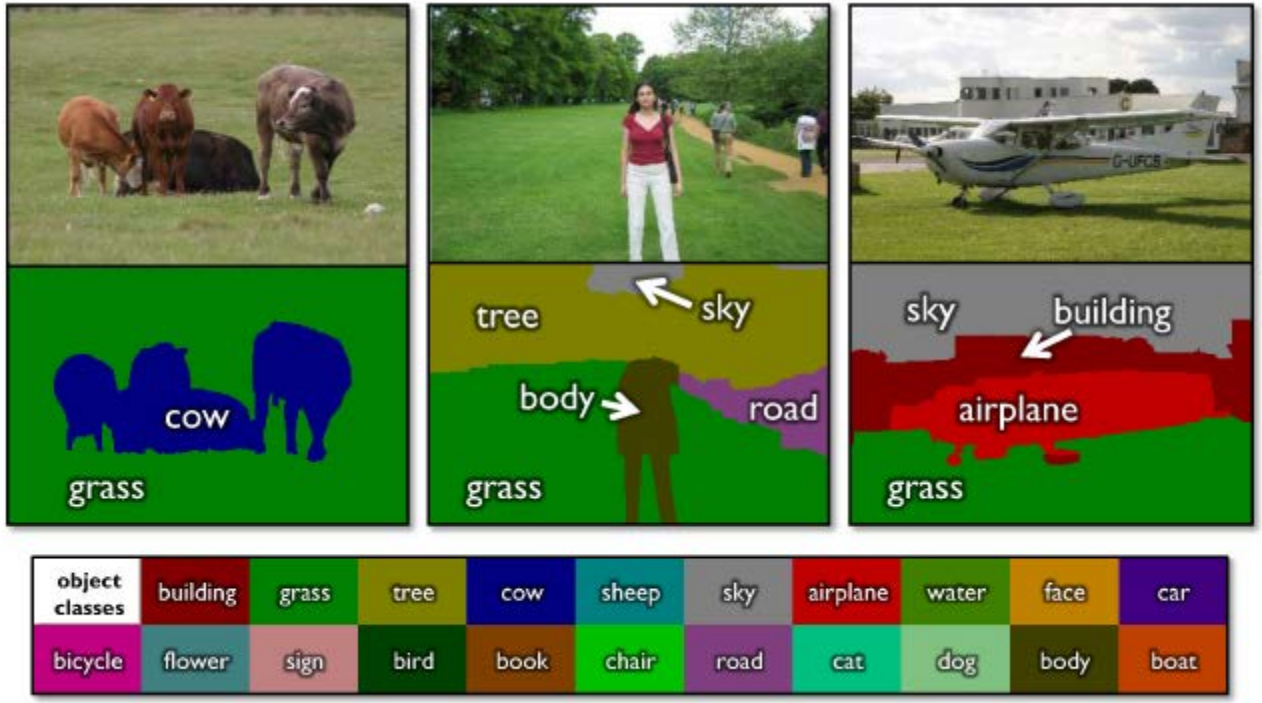
Fine structure

Harder Case



Other application: synthesis

Using graph cuts for recognition



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)
- 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/