Image Analysis

Segmentation by Clustering

Christophoros Nikou

cnikou@cs.uoi.gr

Images taken from:
D. Forsyth and J. Ponce. Computer Vision: A Modern Approach, Prentice Hall, 2003.
Computer Vision course by Svetlana Lazebnik, University of North Carolina at Chapel Hill.
Computer Vision course by Kristen Grauman, University of Texas at Austin.

Segmentation by Clustering

- Grouping and fitting
- The Gestalt school of psychology
- K-means
- Mean shift
- Graph-based methods
- Spectral methods
- Segmentation as a first step to image understanding.

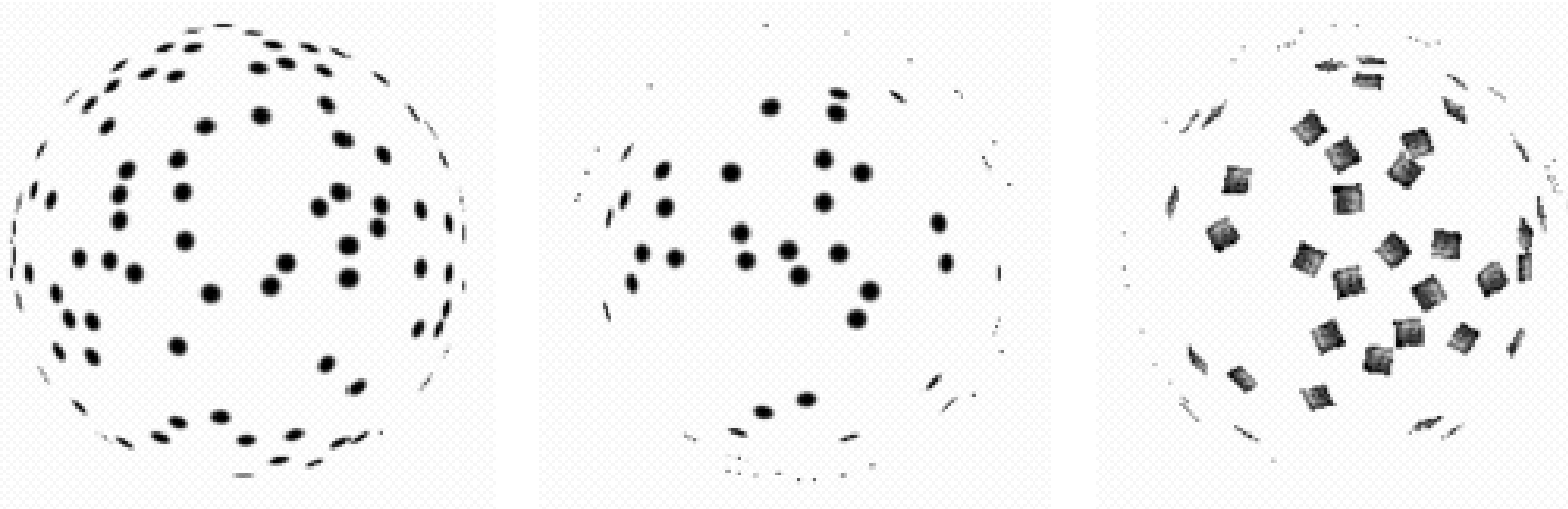
Segmentation and Grouping

- Image analysis and computer vision are inference problems
  - We have measurements and a model.
  - We would like to know what caused the measurement.
- Obtain a compact representation from an image/motion sequence/set of tokens.
  - Purely local pixel grouping (color)
  - Global relations between sets of pixels (e.g. all items lying on a straight line.

Segmentation and Grouping

- Grouping (or clustering)
  - collect together tokens (pixels, points surface elements…) that “belong together”.
- Fitting
  - associate a model with tokens.
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?

Perceptual Organization

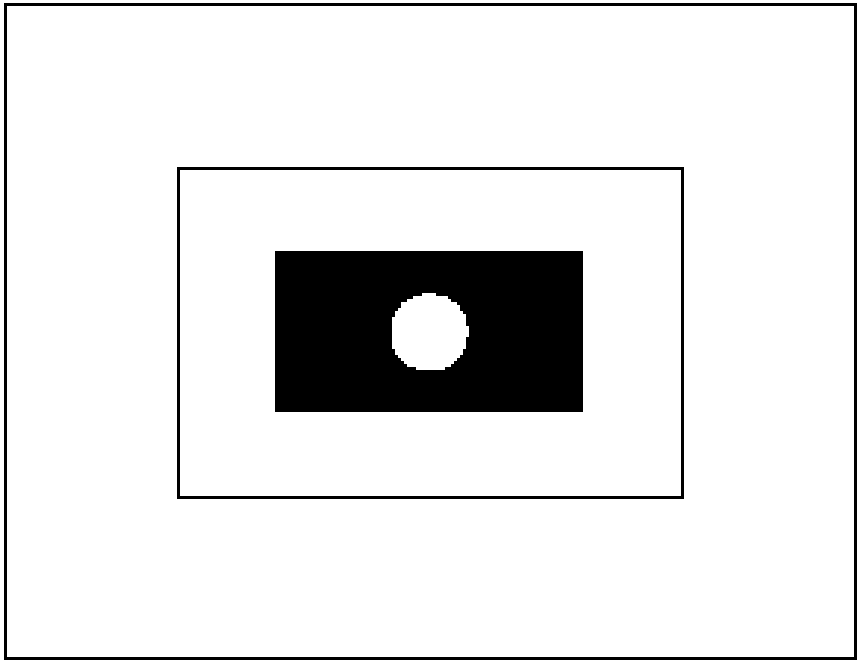


- Why do these tokens belong together?
- The human visual system performs surprisingly well
  - it recognizes the spheres
- How could we make a computer system “see” as a human?
  - why does it have to assemble the tokens?

Basic ideas of grouping in humans

- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground.
  - impoverished theory
- The Gestalt school
  - Gestalt: shape, whole or a group
  - grouping is the key to human visual perception.
  - A more complicated mechanism than figure/ground must explain visual perception

The Gestalt School (cont.)

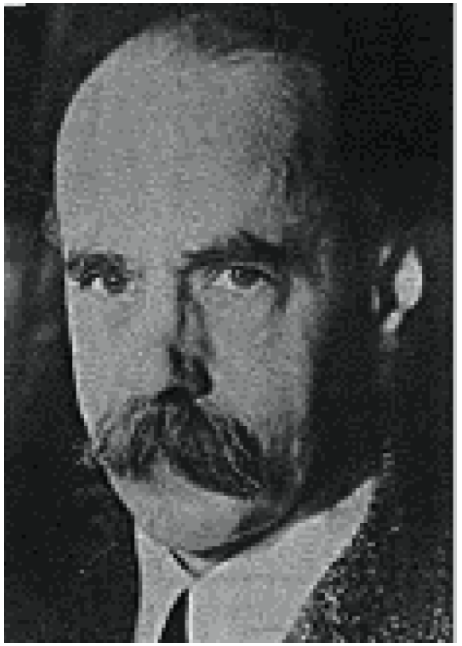


- Figure-ground discrimination is a relatively poor model
  - White circle on a black background?
  - A black rectangle with a hole on it?

The Gestalt School

Psychologists identified series of factors that predispose sets of elements to be grouped in the human visual system.

"I stand at the window and see a house, trees, sky. Theoretically, I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No, I have house, sky and trees."

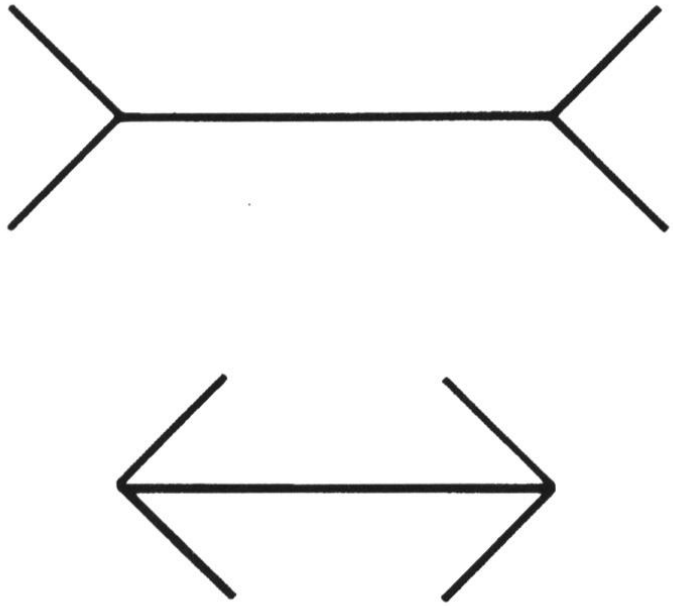


Max Wertheimer (1880-1943).

Untersuchungen zur Lehre von der Gestalt.
Psychologische Forschung, Vol. 4, pp. 301-350, 1923.

The Gestalt School (cont.)

- The Muller-Lyer illusion:
  - You can't look at this figure and ignore ("group") the arrowheads.



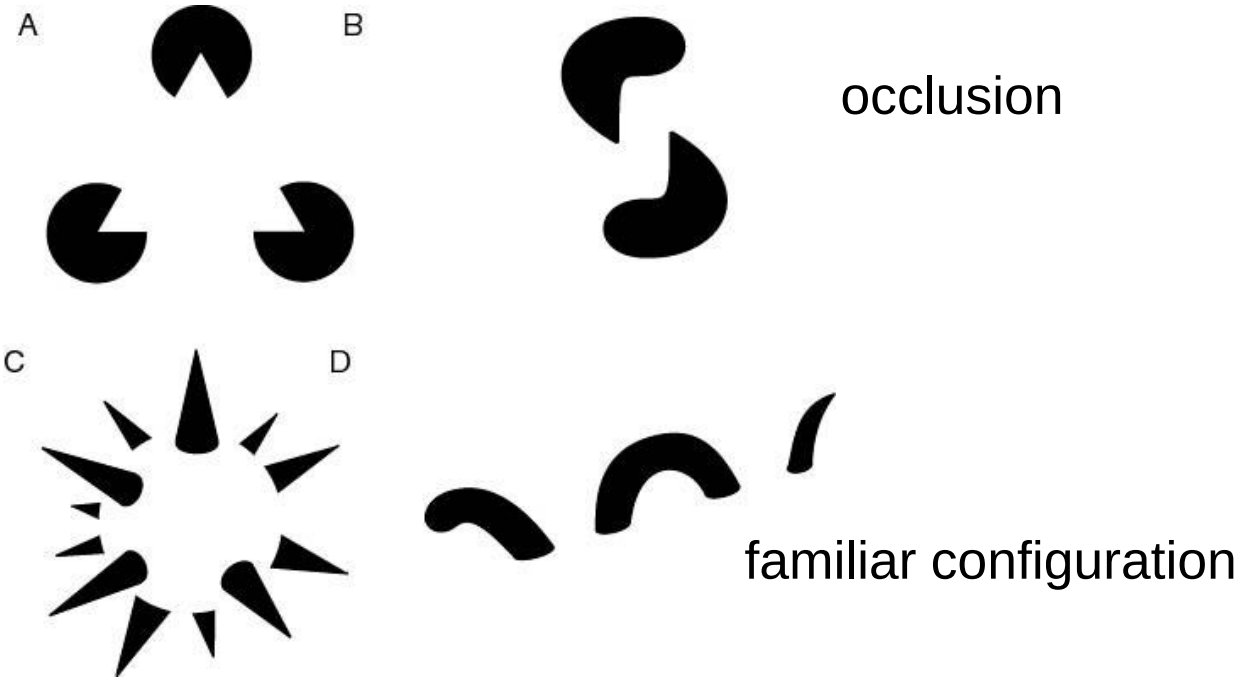
The Gestalt School (cont.)

- Subjective contours
  - Scattered tokens?
  - Occlusion?

The Gestalt School (cont.)

- Elements in a collection can have properties that result from relationships.
  - "The whole is greater than the sum of its parts"

subjective contours

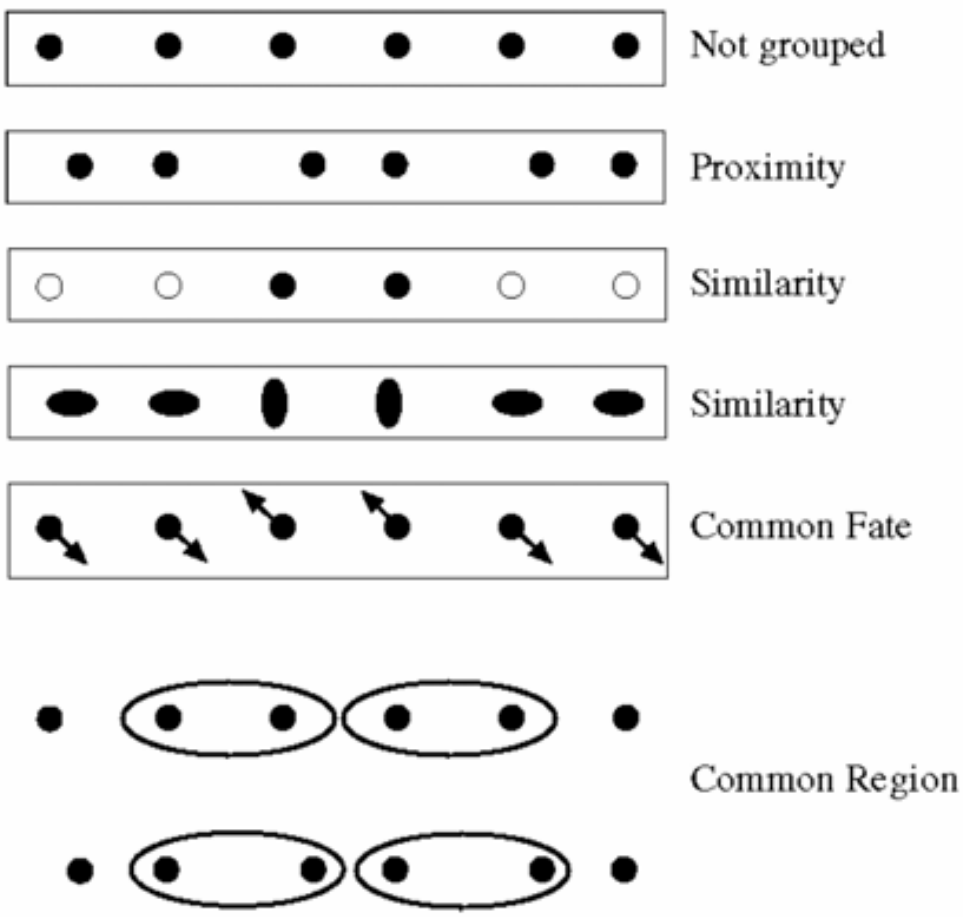


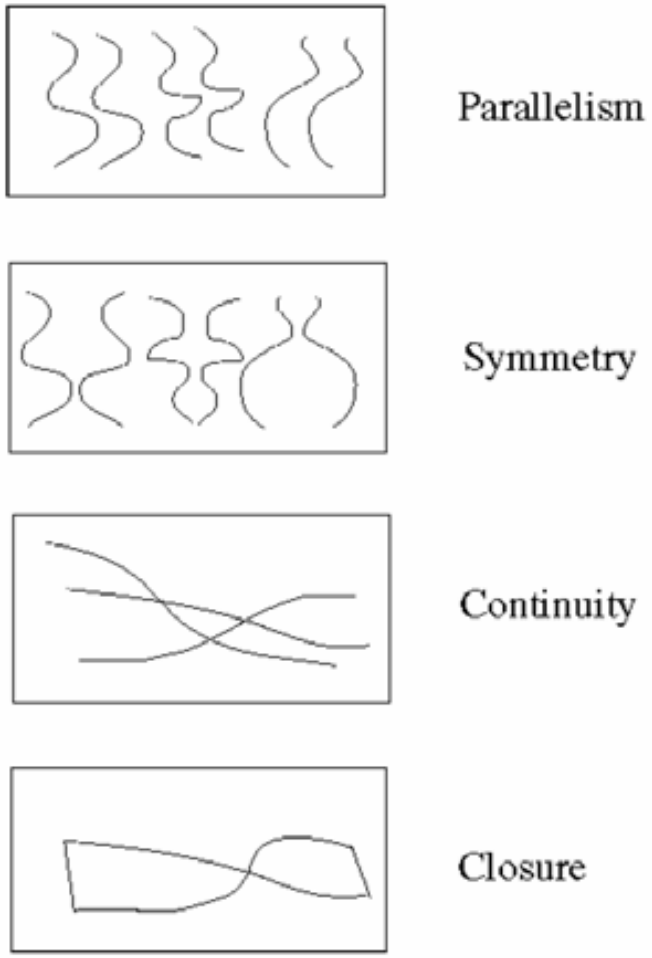
occlusion

familiar configuration

The Gestalt School (cont.)

Gestalt factors





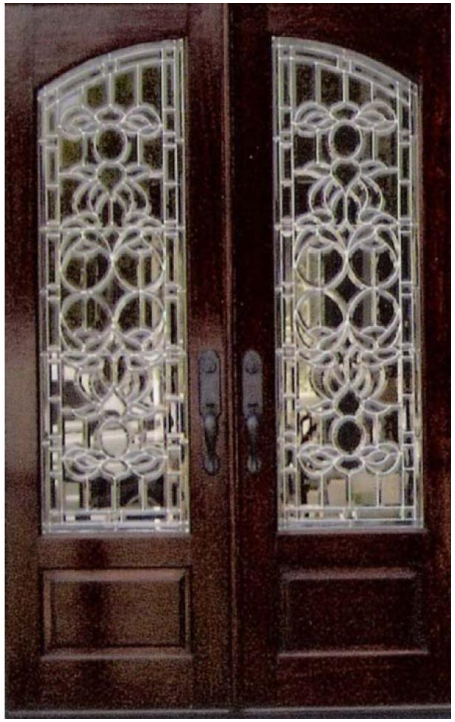
Intuitive factors leading to grouping are very difficult to translate into algorithms.

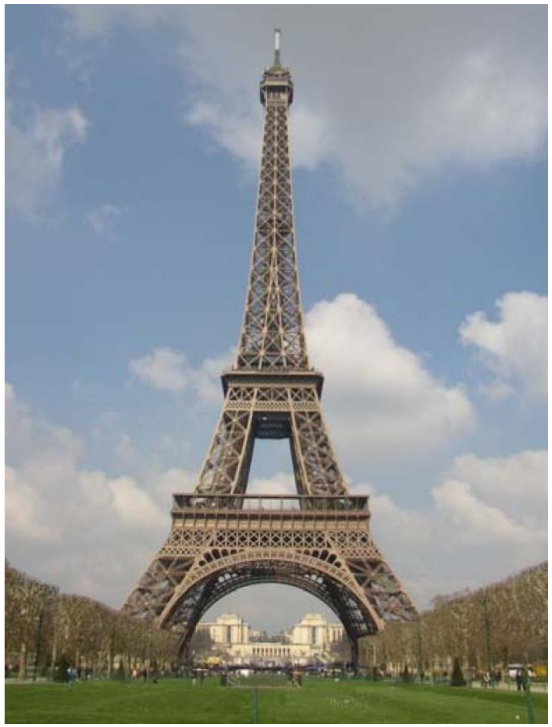
Gestalt factors in natural images

Similarity

Gestalt factors in natural images (cont.)

Symmetry





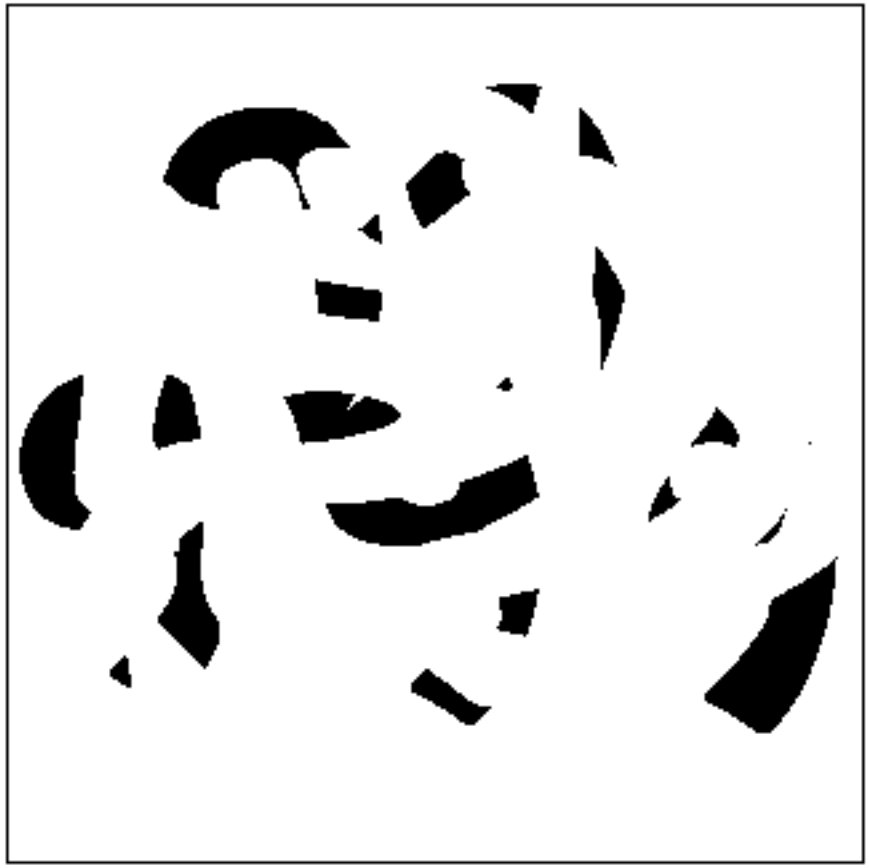
Gestalt factors in natural images (cont.)

Common fate

Gestalt factors in natural images (cont.)

Proximity

Gestalt factors (cont.)

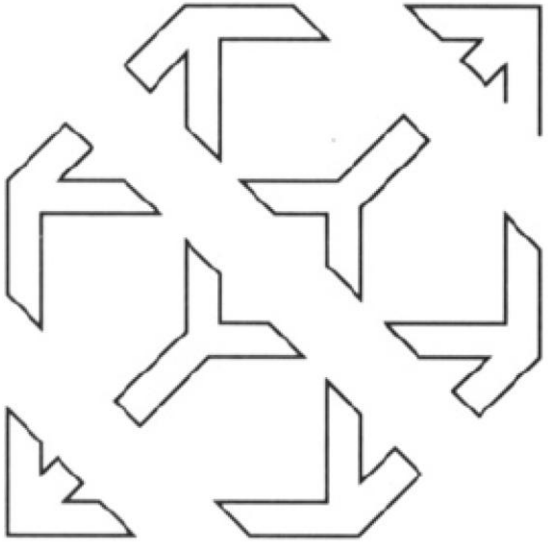


What can you see ?

Gestalt factors (cont.)

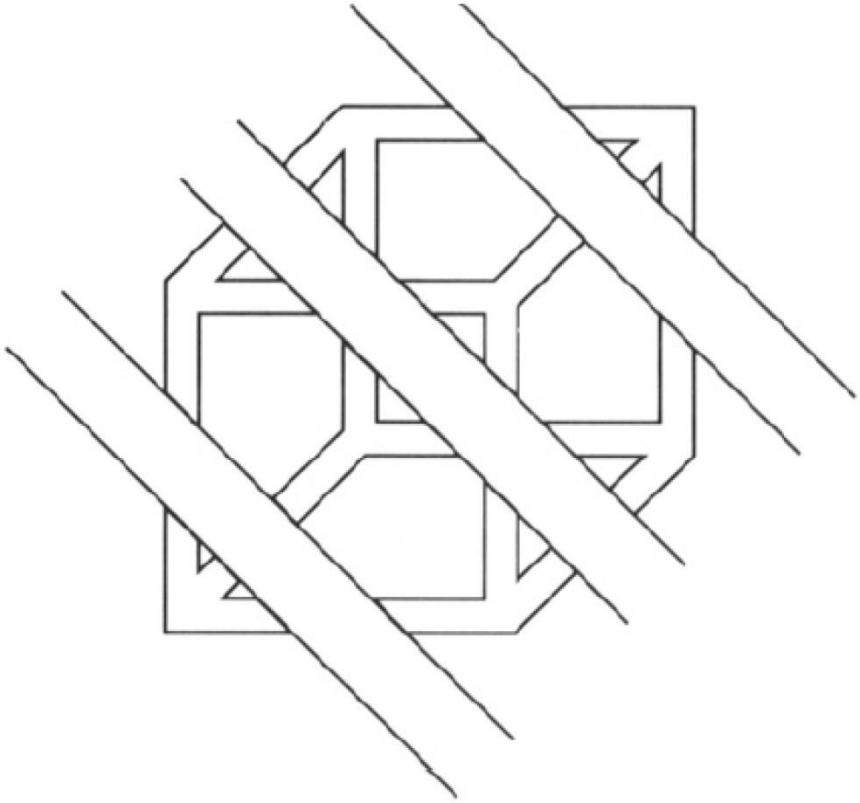
The visual system is helped by the evidence that the tokens are separated for a reason: **occlusion**.

Gestalt factors (cont.)



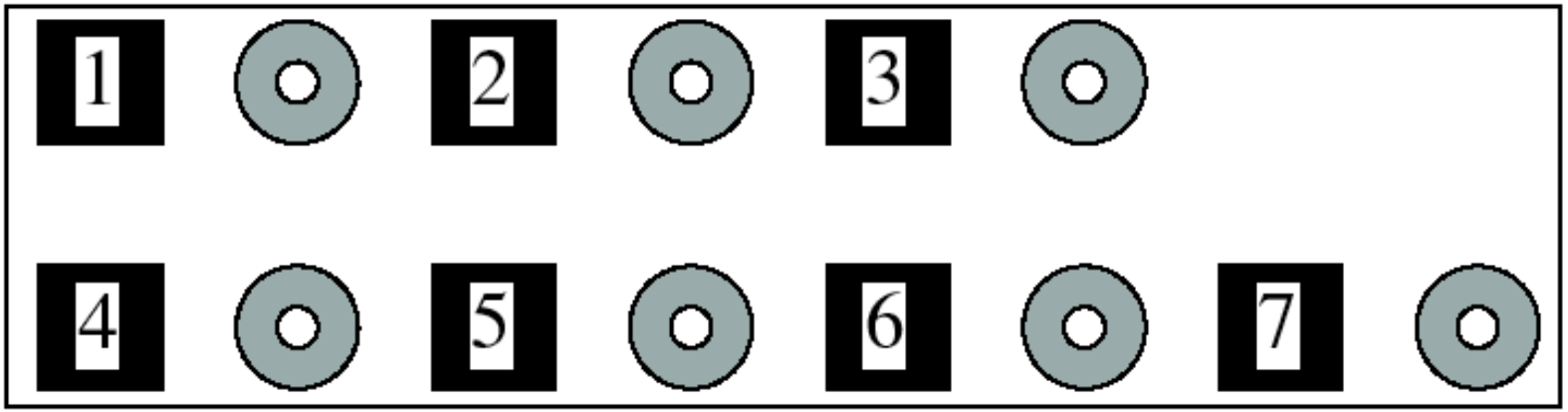
What can you see ?

Gestalt factors (cont.)



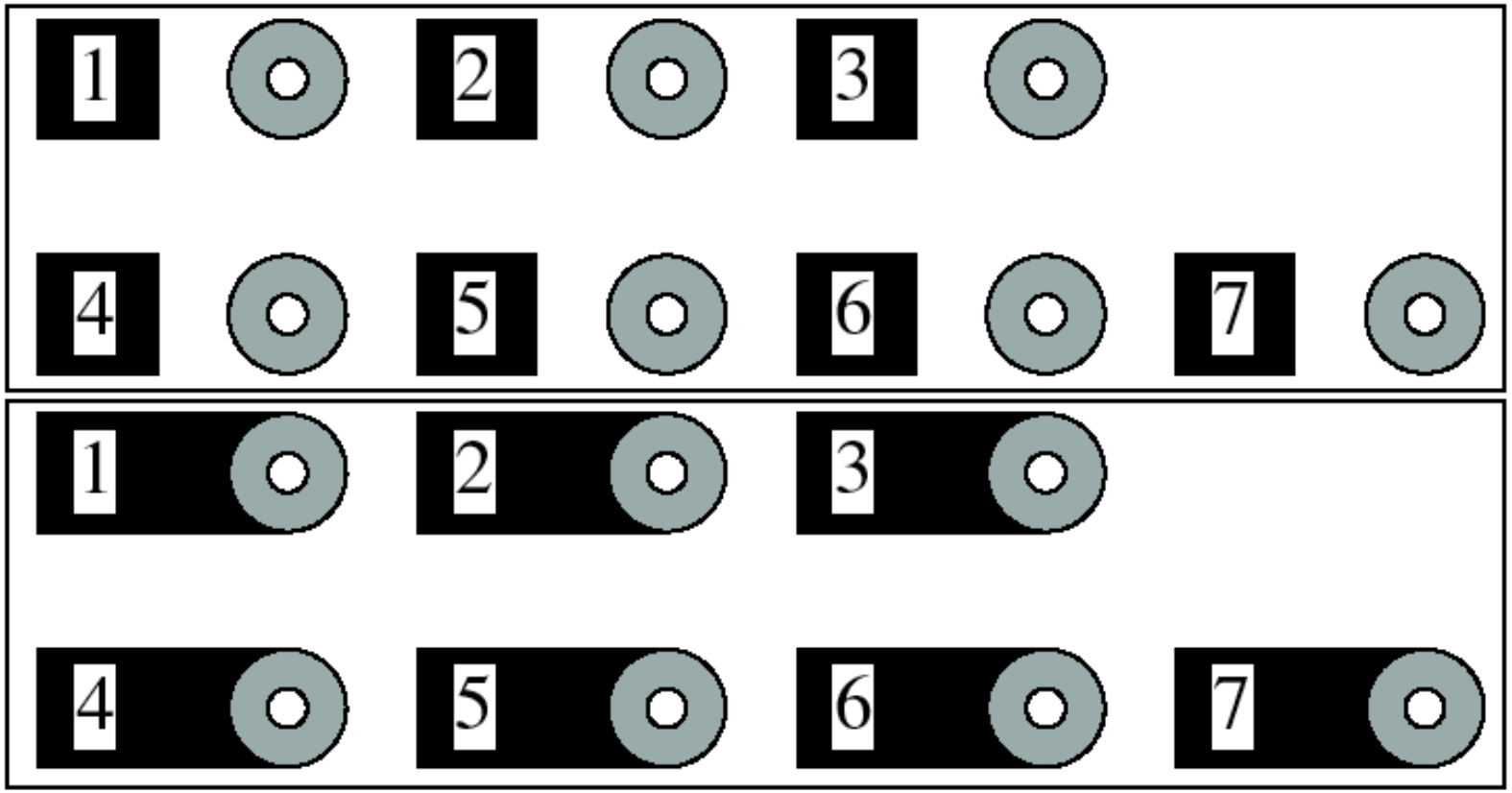
Continuity through **occlusion** (is it a cube?)

Gestalt factors (cont.)



Elevator buttons at Computer Science building, U.C. at Berkeley.

Gestalt factors (cont.)



Proximity cue has been disambiguated.

Gestalt factors (cont.)

- Even psychologists argued which factors are more dominant than the others.
- Today, Gestalt factors are important but they should be considered as consequences of a more general grouping mechanism and not as the mechanism itself.
- Which is this mechanism?
  - We cannot answer yet.

Application: Background Subtraction

- Simple segmentation algorithms work well when we know what to look for.
- If we know what the background looks like, it is easy to identify “interesting bits”.
- Applications
  - Person in an office
  - Tracking cars on a road
  - Surveillance

Application: Background Subtraction

- Approach:
  - use a weighted moving average (over time) to estimate background image.
    - Distant frames are attributed smaller weights (e.g. the weather changes smoothly from rain to sunshine).
    - This is a filter smoothing a function of time.
  - Subtract from current frame.
  - Large absolute values are interesting pixels.
  - The method is powerful at **coarse scales**.

Application: Background Subtraction

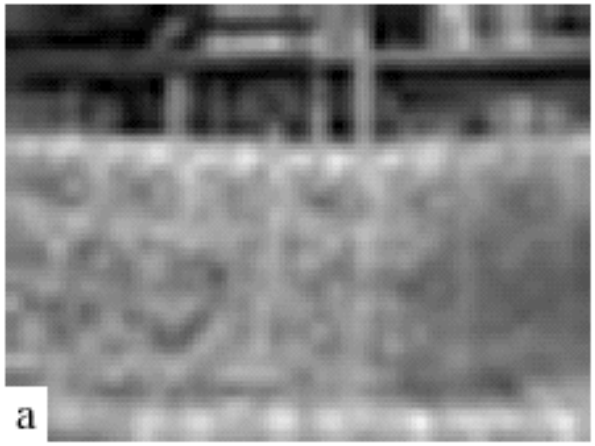
Form a background estimate $\mathcal{B}^{(0)}$. At each frame $\mathcal{F}$
  Update the background estimate, typically by
    forming $\mathcal{B}^{(n+1)} = \frac{w_a \mathcal{F} + \sum_i w_i \mathcal{B}^{(n-i)}}{w_c}$
    for a choice of weights w_a, w_i and w_c.
  Subtract the background estimate from the
    frame, and report the value of each pixel where
    the magnitude of the difference is greater than some
    threshold.
end

Algorithm 9.1: Background Subtraction.

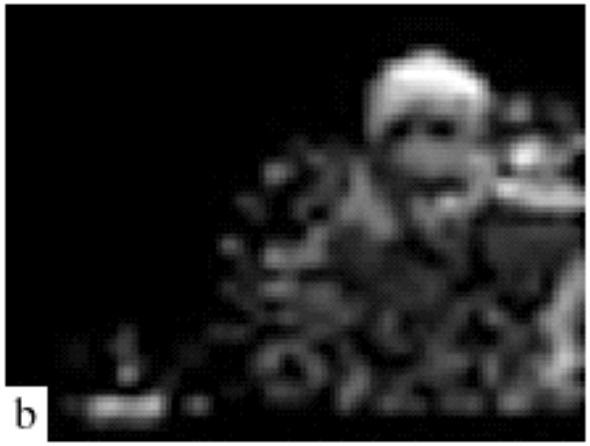
Application: Background Subtraction (cont.)

- Every 5th frame of the sequence is shown.
- The child moves from one side to the other.

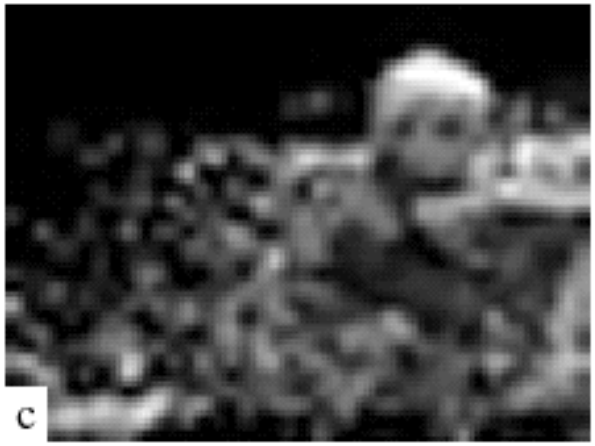
Application: Background Subtraction (cont.)



Background estimation averaging frames of size 80x60.
The child spent more time on the right side.



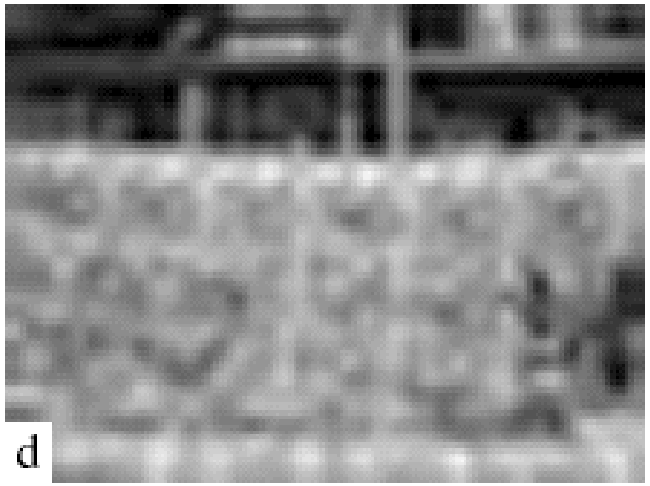
Pixels of a frame whose difference from the average exceed a *threshold*.



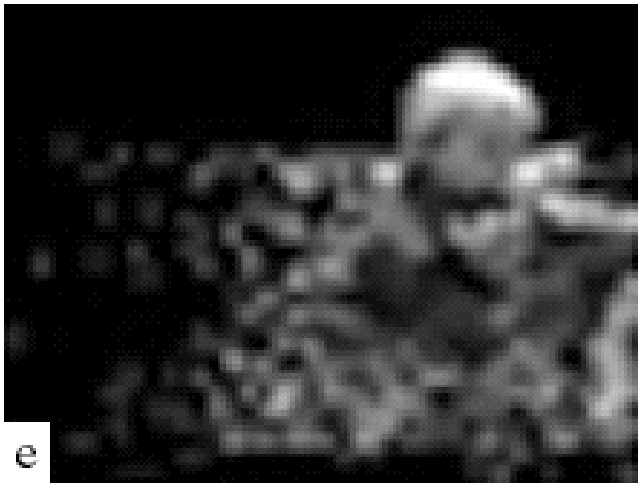
Pixels of a frame whose difference from the average exceed a *smaller threshold*.

In both thresholds, there are excess pixels and missing pixels.

Application: Background Subtraction (cont.)



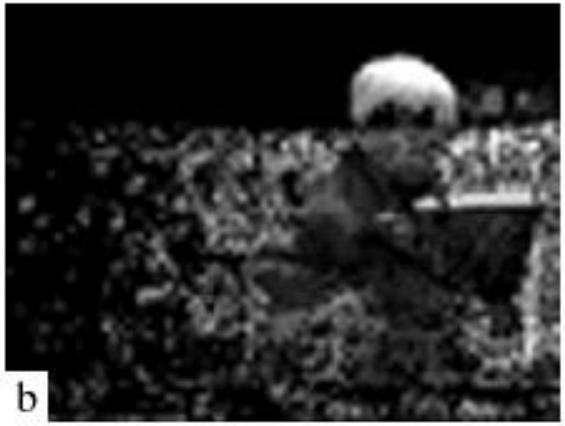
Background estimation by a more sophisticated method (EM).



Pixels of a frame whose difference from the average exceed a *threshold*.

There are also excess pixels and missing pixels.

Application: Background Subtraction (cont.)



The same results at a higher resolution (160x120).

The high frequency texture of the sofa pattern was mistaken for the child.

This is because small movements can cause **misregistration** of spatial content carrying high frequencies.

Application: Shot Boundary detection

- Video sequences are composed of shots:
  - Shorter subsequences showing largely the same object.
- It is very helpful to represent a video as a collection of shots
  - Each shot is represented by a *key frame*.
  - Video browsing retrieval.
- Approach: find frames that are significantly different from the previous frame.
  - Take into account that object and background may move.
  - Simple difference will not do the job.

Application: Shot Boundary detection (cont.)

- Other approaches:
  - Histogram based methods
    - Compute color histograms and take the difference.
    - Insensitive to motion and camera jitters.
  - Block comparison
    - Avoids difficulties of color histograms.
      - A red object disappearing in the bottom is equivalent to a red object appearing in the top.
  - Edge differencing
    - Compare edge maps (corresponding edges between frames).
- These are ad hoc methods but usually sufficient for standard problems.

Application: Interactive segmentation

- Goals
  - User cuts an object to paste into another image
  - User forms a matte to cope with e.g. hair
    - weights between 0 and 1 to mix pixels with background
- Interactions
  - mark some foreground, background pixels with strokes
  - put a box around foreground
- Technical problem
  - allocate pixels to foreground/background class
  - consistent with interaction information
  - segments are internally coherent and different from one another

Application: Interactive segmentation

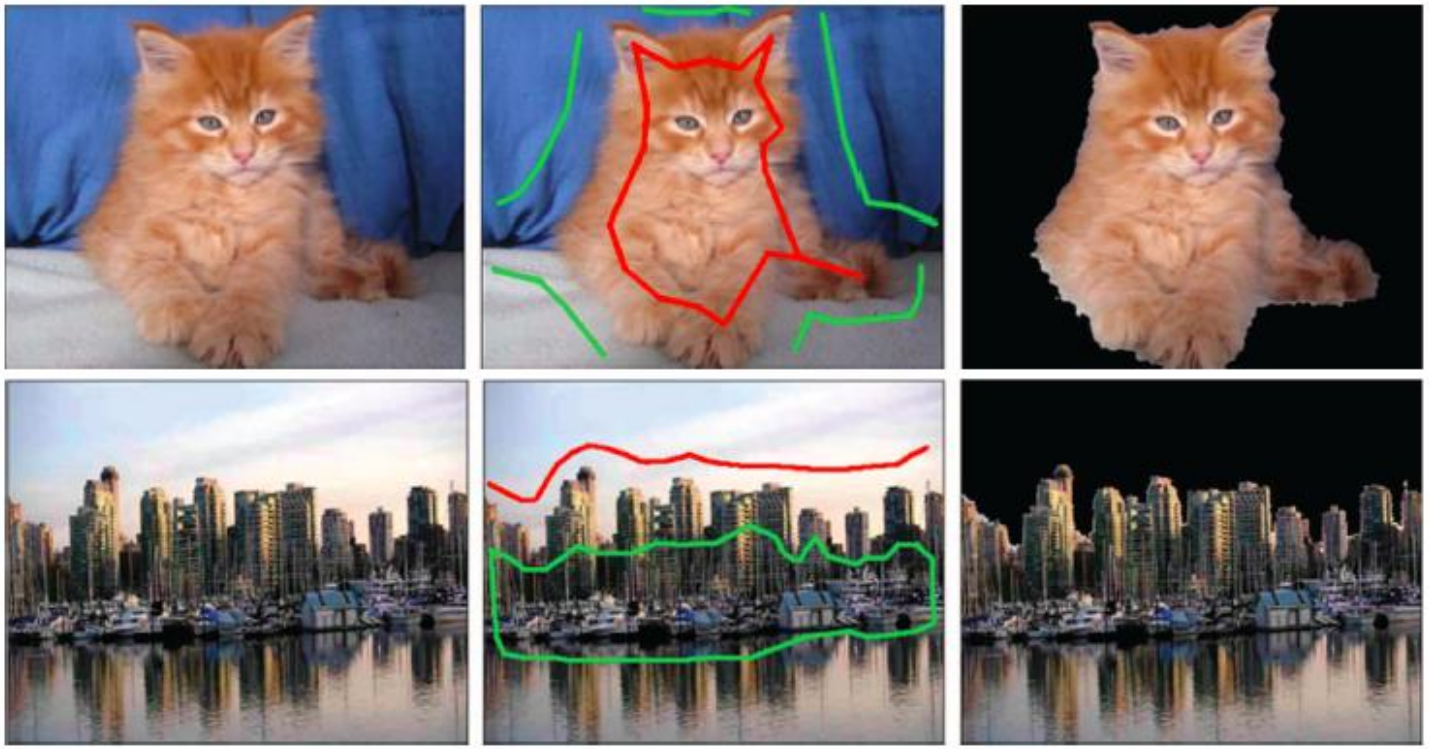


FIGURE 9.11: A user who wants to cut an object out of an image (**left**) could mark some foreground pixels and some background pixels (**center**), then use an interactive segmentation method to get the cut out components on the **right**. The method produces a model of foreground and background pixel appearance from the marked pixels, then uses this information to decide a figure ground segmentation. *This figure was originally published as Figure 9 of "Interactive Image Segmentation via Adaptive Weighted Distances," by Protiere and Sapiro, IEEE Transactions on Image Processing, 2007* © *IEEE, 2007.*

Application: Interactive segmentation

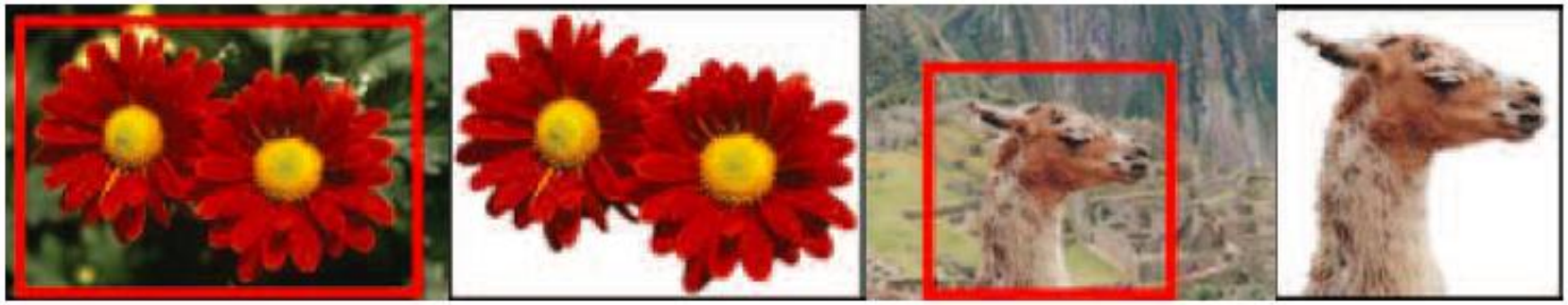


FIGURE 9.12: In a grabcut interface for interactive segmentation, a user marks a box around the object of interest; foreground and background models are then inferred by a clustering method, and the object is segmented. If this segmentation isn't satisfactory, the user has the option of painting foreground and background strokes on pixels to help guide the model. *This figure was originally published as Figure 1 of "GrabCut Interactive Foreground Extraction using Iterated Graph Cuts" by C. Rother, V. Kolmogorov, and A. Blake, Proc. ACM SIGGRAPH, 2004* © *ACM, 2004.*

Application: Interactive segmentation

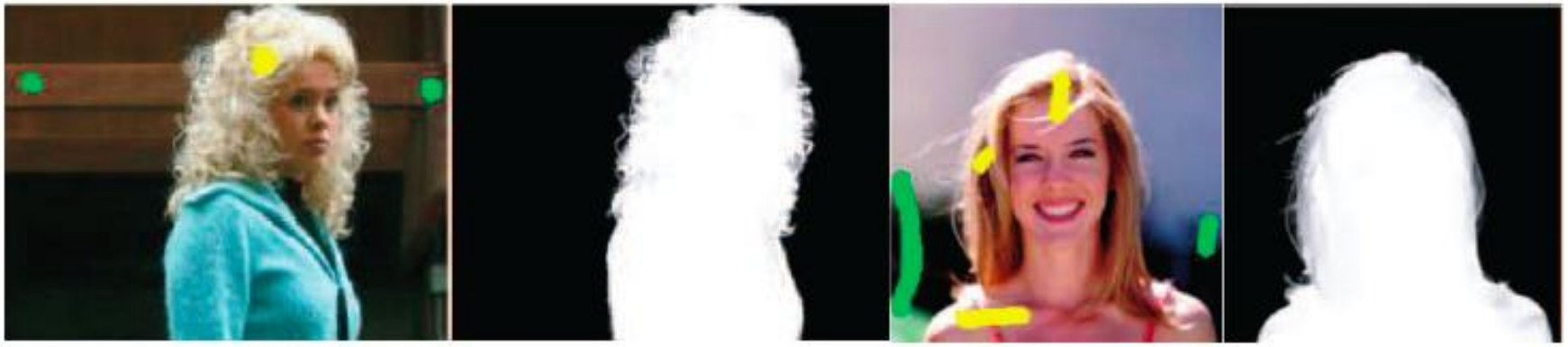


FIGURE 9.13: Matting methods produce a real-valued mask (rather than a foreground-background mask) to try and compensate for effects in hair, at occluding boundaries, and so on, where some pixels consist of an average of foreground and background values. The matte is bright for foreground pixels and dark for background pixels; for some pixels in the hair, it is gray, meaning that when the foreground is transferred to a new image, these pixels should become a weighted sum of foreground and background. The gray value indicates the weight. *This figure was originally published as Figure 6 of "Spectral Matting," by A. Levin, A. Rav-Acha, and D. Lischinski, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008* © *IEEE, 2008.*

Application: Forming image regions

- Pixels are too small and too detailed a representation for
  - recognition
  - establishing correspondences in two images
  - …
- Superpixels
  - Small groups of pixels that are
    - clumpy, coarse
    - similar
    - a reasonable representation of the underlying pixels

Application: Forming image regions

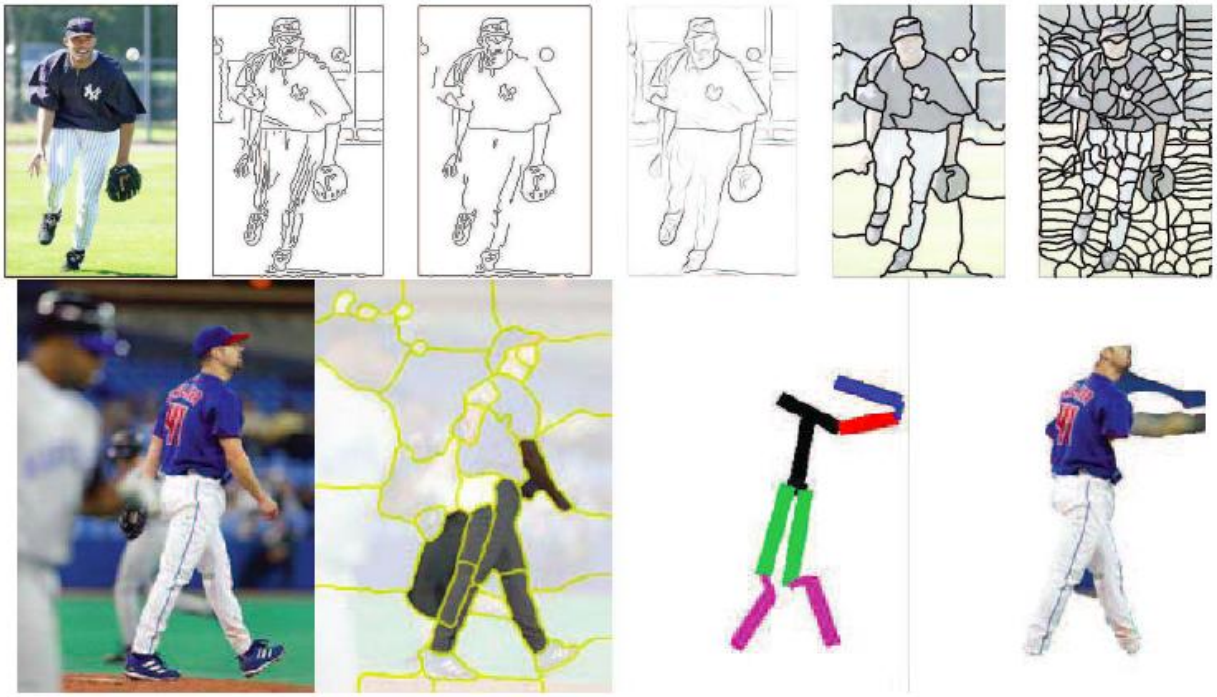
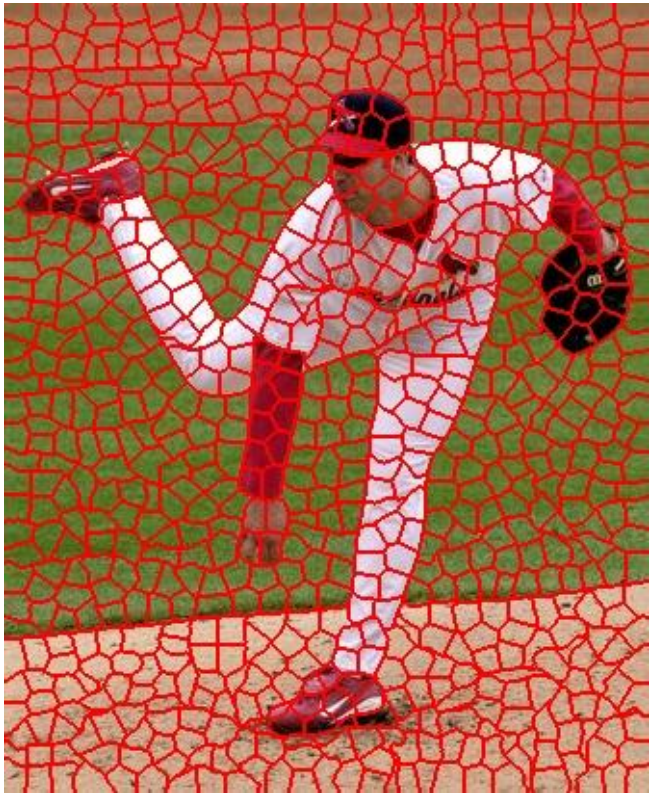
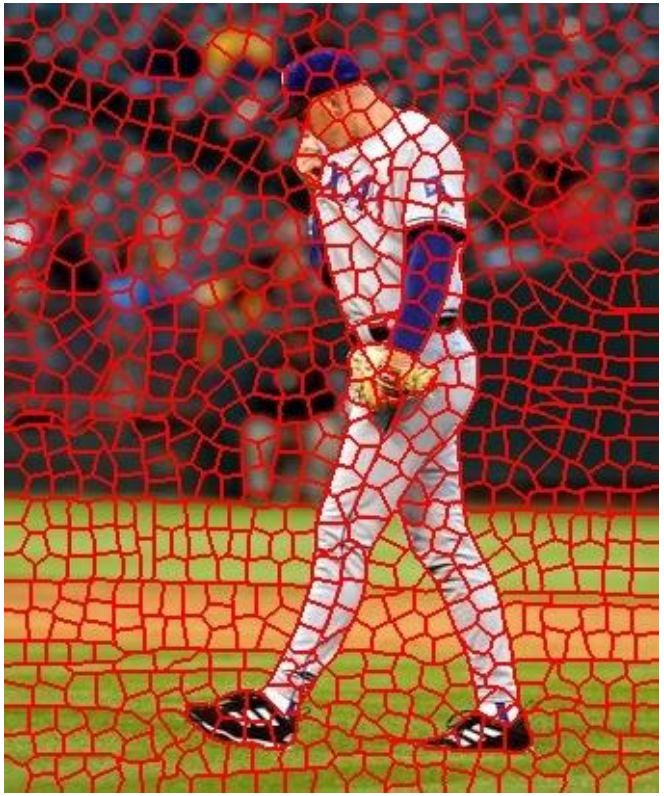


FIGURE 9.14: Superpixels often can expose structure in images that other representations conceal. Human body segments tend to appear as long, thin segments. In the **top** row, an image together with three different edge maps (the edge detector of Section 5.2.1, with two scales of smoothing, and the P_b of Section 17.1.3) and superpixels computed at two "scales" (in this case, the number of superpixels was constrained). Notice that the coarser superpixels tend to expose limb segments in a straightforward way. On the **bottom** row, another image, its superpixels, and two versions of the body layout inferred from the superpixel representation. *This figure was originally published as Figure 3 and part of Figure 10 of "Recovering human body configurations: Combining Segmentation and Recognition," by G. Mori, X. Ren, A. Efros, and J. Malik, Proc. IEEE CVPR, 2004* © *IEEE, 2004.*

Application: Forming image regions

Superpixels





Segmentation by Clustering

- Most image segmentation algorithms are based on clustering.
- Agglomerative clustering
  - Each data item is regarded as a cluster.
  - Clusters are recursively merged.
- Divisive clustering
  - The entire data set is regarded as a cluster.
  - Clusters are recursively split.

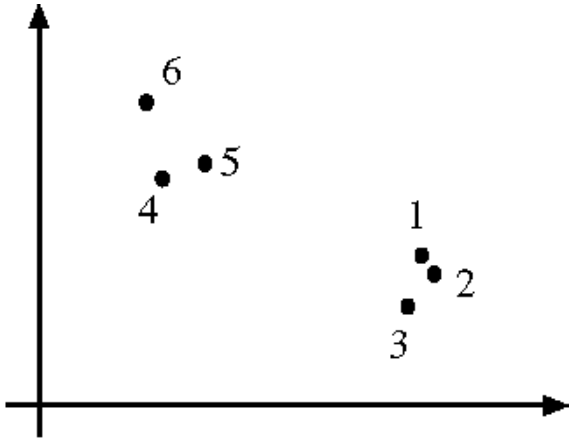
Segmentation by Clustering (cont.)

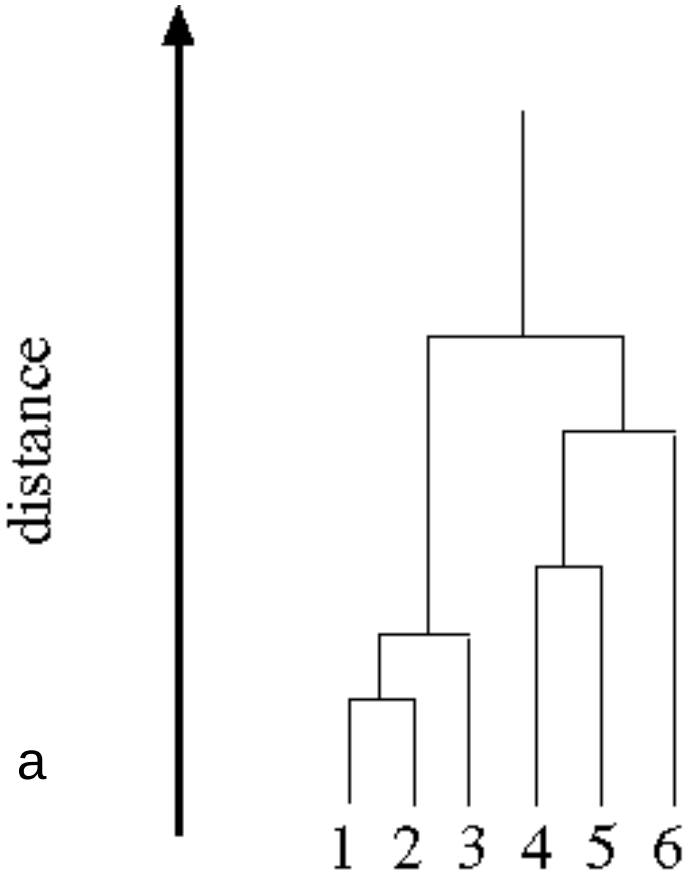
- Major issue:
  - What is a good inter-cluster distance?
    - The distance between the closest elements in the clusters (*single-link clustering*)
      - Tends to form extended clusters.
    - The maximum distance between an element of the first and one of the second cluster (*complete-link clustering*)
      - Tends to provide rounded clusters.
    - The average distance between elements in the clusters (*group average clustering*)
      - Tends to form rounded clusters also.

Segmentation by Clustering (cont.)

- Major issue:
  - How many clusters are there?
    - Difficult to answer if there is no model for the process that generated the data.
    - The hierarchy may be displayed in the form of a dendrogram.
      - A representation of cluster structure displaying cluster distances.
      - We may determine the number of clusters from the dendrogram.

Segmentation by Clustering (cont.)





- A dendrogram obtained by agglomerative clustering.
- Selecting a particular value of distance then a horizontal line at that distance splits the dendrogram into clusters.
- It gives some insight into how good are the clusters.

Segmentation by Clustering (cont.)

- Common distances in image analysis involve color, texture and difference in position (to provide blobby segments).
- Problem for using agglomerative or divisive clustering: there are a lot of pixels in the image.
  - Too big dendrograms.
  - Impractical to look for the best split (merge) of clusters).
    - Divisive methods are modified by using a summary of a cluster (histogram).
    - Agglomerative methods also use coordinates for inter-cluster distance computation, and merging is performed on neighboring clusters.

The watershed algorithm

- Agglomerative clustering with a special metric

An early segmentation algorithm that is still widely used is the *watershed* algorithm. Assume we wish to segment image $\mathcal{I}$. In this algorithm, we compute a map of the image gradient magnitude, $\|\nabla\mathcal{I}\|$. Zeros of this map are locally extreme intensity values; we take each as a seed for a segment, and give each seed a unique label. Now we assign pixels to seeds by a procedure that is, rather roughly, analogous to filling a height map with water (hence the name). Imagine starting at pixel (i, j); if we travel backward down the gradient of $\|\nabla\mathcal{I}\|$, we will hit a unique seed. Each pixel gets the label of the seed that is hit by this procedure.

The watershed algorithm (cont.)

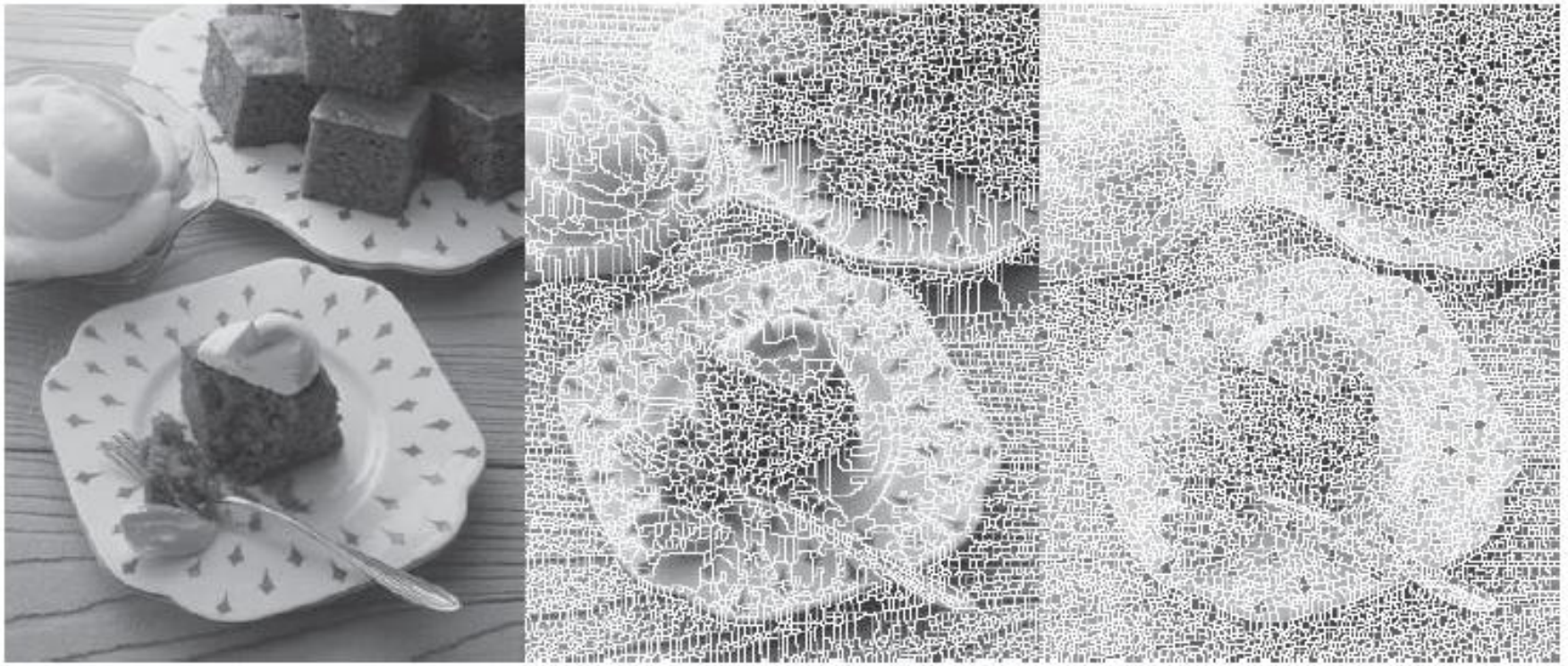


FIGURE 9.16: Segmentation results from the watershed algorithm, applied to an image by Martin Brigdale. **Center:** watershed applied to the image intensity; notice some long superpixels. **Right:** watershed applied to image gradient magnitude; this tends to produce rounder superpixels. *Martin Brigdale © Dorling Kindersley, used with permission.*

K-Means for Segmentation

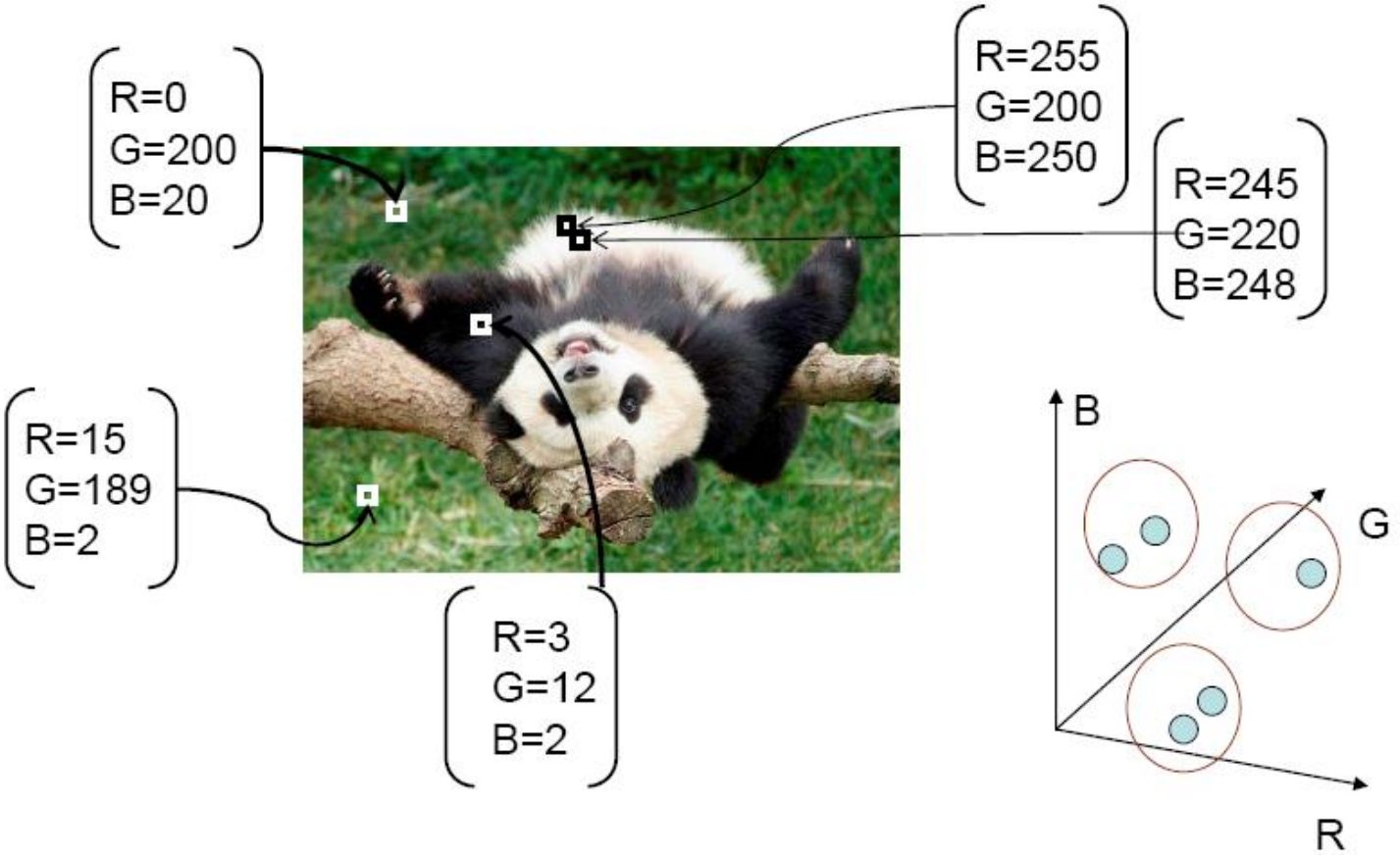
- Choose a fixed number of clusters.
- Choose cluster centers and point-cluster allocations to minimize error
- We can't do this by search, because there are too many possible allocations.
- It minimizes the dispersion of the data from the centers.

- Algorithm
  - fix cluster centers; allocate points to closest cluster.
  - fix allocation; compute best cluster centers
- We could use any set of features for which we can compute a distance (careful about scaling each dimension).

$$E(\text{clusters},\text{data}) = \sum_{i \in clusters} \left\{ \sum_{j \in i^{th}\ cluster} (x_j - c_i)^T (x_j - c_i) \right\}$$

K-Means for Segmentation (cont.)

- Cluster similar pixels (features) together



Source: K. Grauman

K-Means for Segmentation (cont.)

K=11 K=11

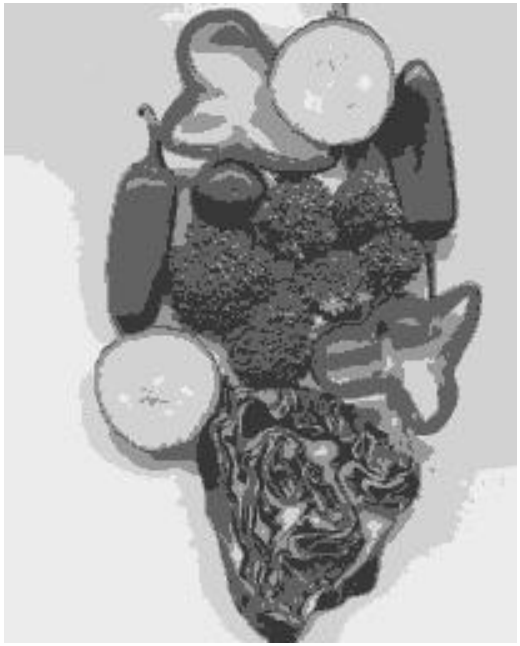


Image Intensity-based clusters Color-based clusters

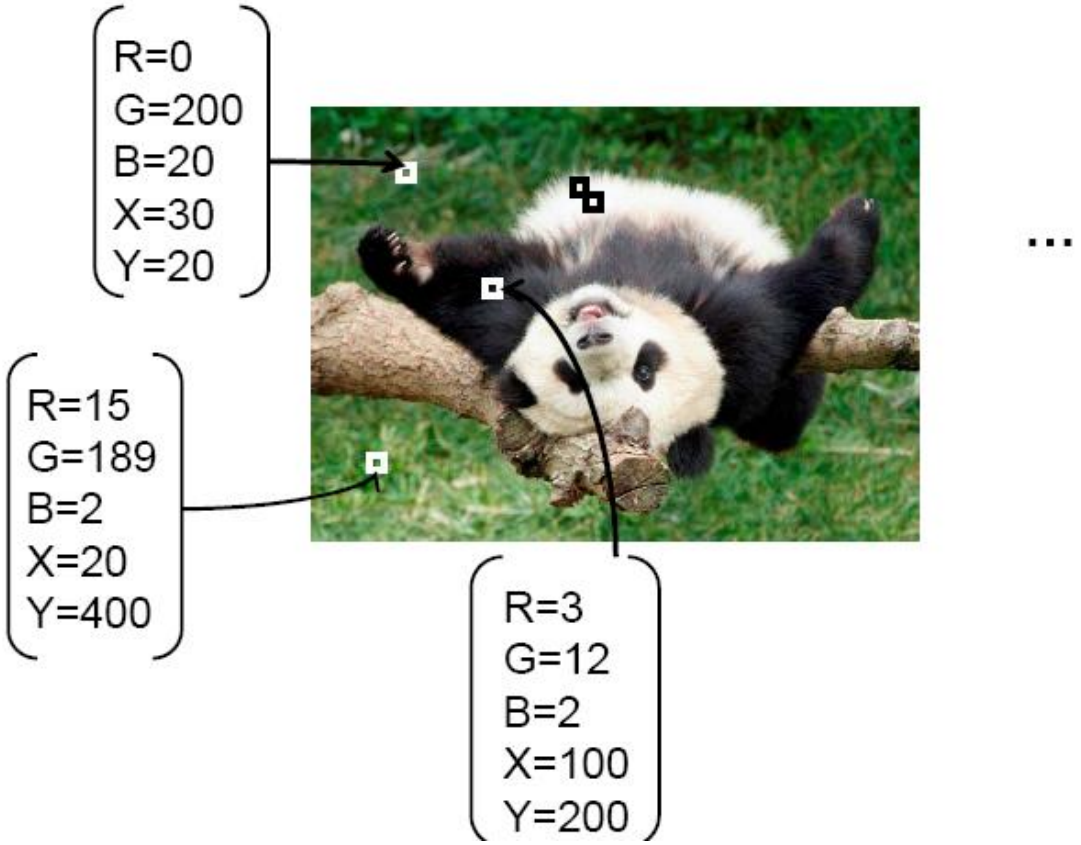
- Segments are not necessarily connected using only color.
- Absence of texture features is obvious (red cabbage).

K-Means for Segmentation (cont.)

K=11 (4 segments are shown here).

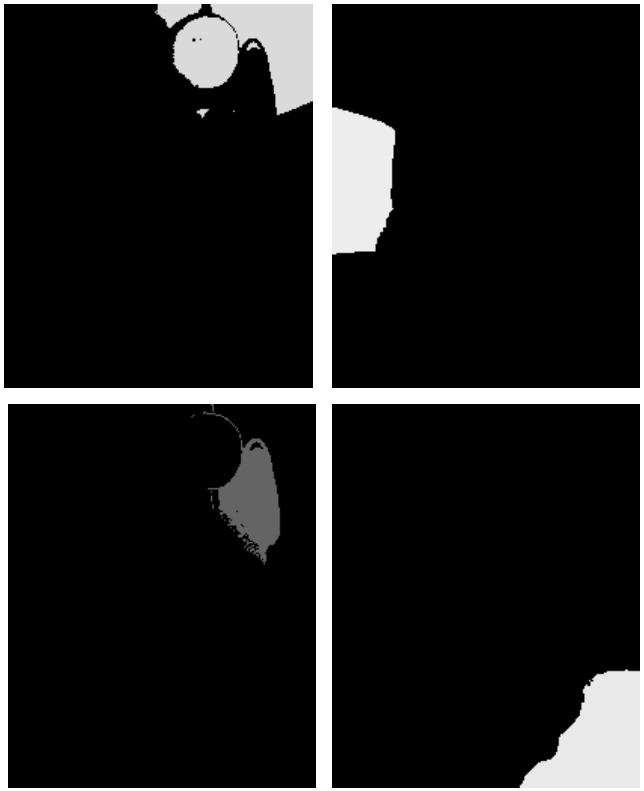
K-Means for Segmentation (cont.)

- Cluster similar pixels (features) together

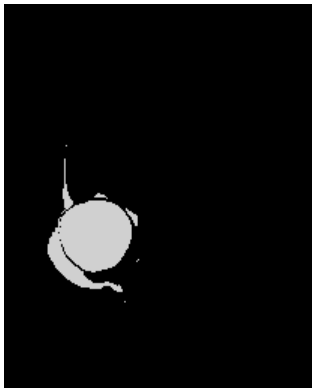


Source: K. Grauman

K-Means for Segmentation (cont.)



K=20 (5 segments are shown here).



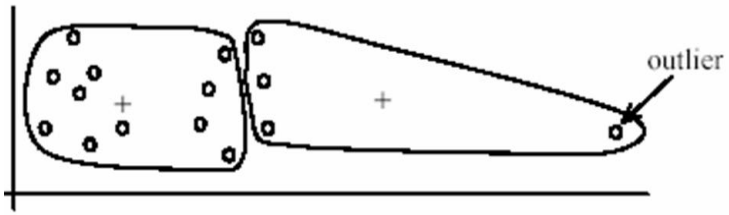
- Position is also used as part of the feature vector.
- Notice that the large background is broken.

K-Means for Segmentation (cont.)

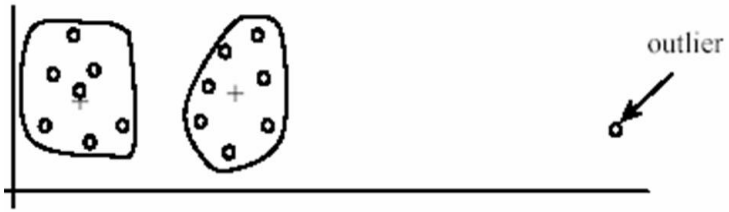
- Demos in the web
  - Information and visualization
    http://informationandvisualization.de/blog/kmeans-and-voronoi-tesselation-built-processing/demo
  - Alessandro Giusti home – Clustering
    http://www.leet.it/home/lale/joomla/component/option,com_wrapper/Itemid,50/
  - University of Leicester
    http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html

K-Means for Segmentation (cont.)

- Pros
  - Very simple method.
  - Converges to a local minimum.
- Cons
  - Memory-intensive.
  - Need to pick K.
  - Sensitive to initialization.
  - Sensitive to outliers.
  - Finds “spherical” clusters.



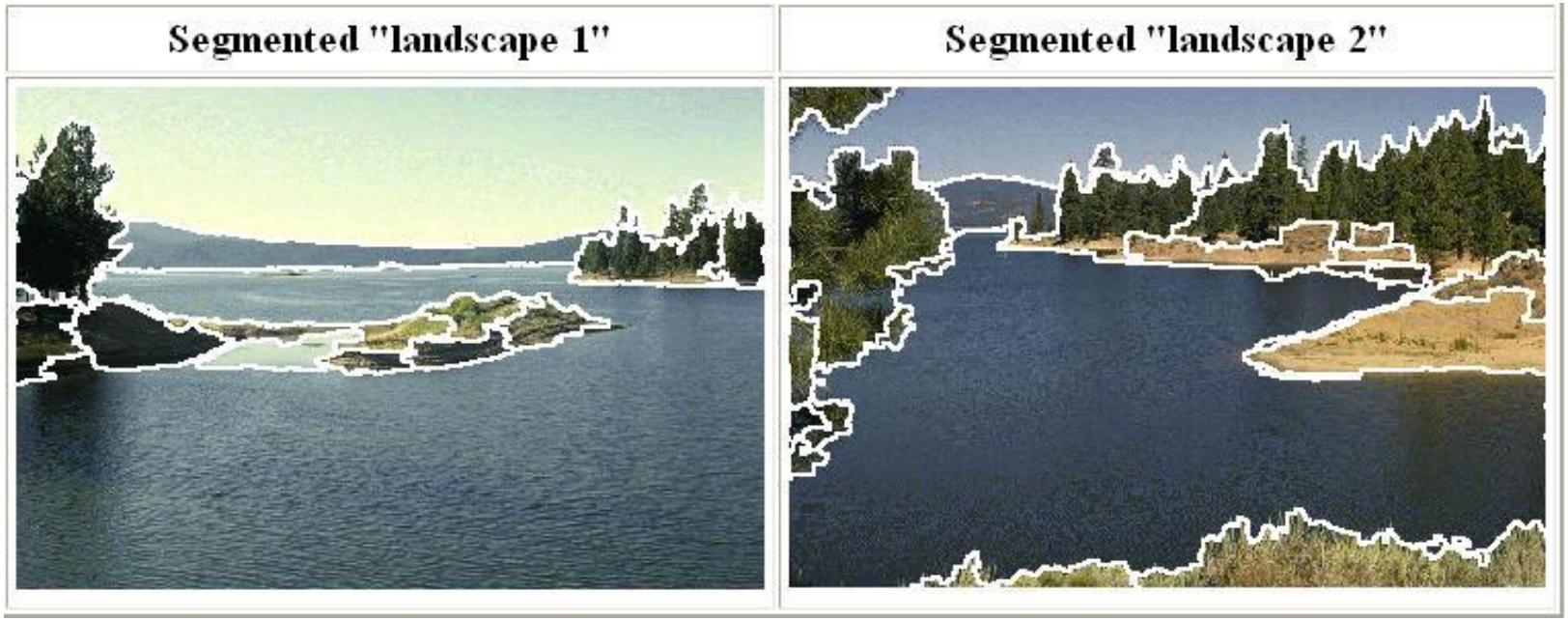
(A): Undesirable clusters



(B): Ideal clusters

Segmentation by Mean Shift

- An advanced and versatile technique for clustering-based segmentation.



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

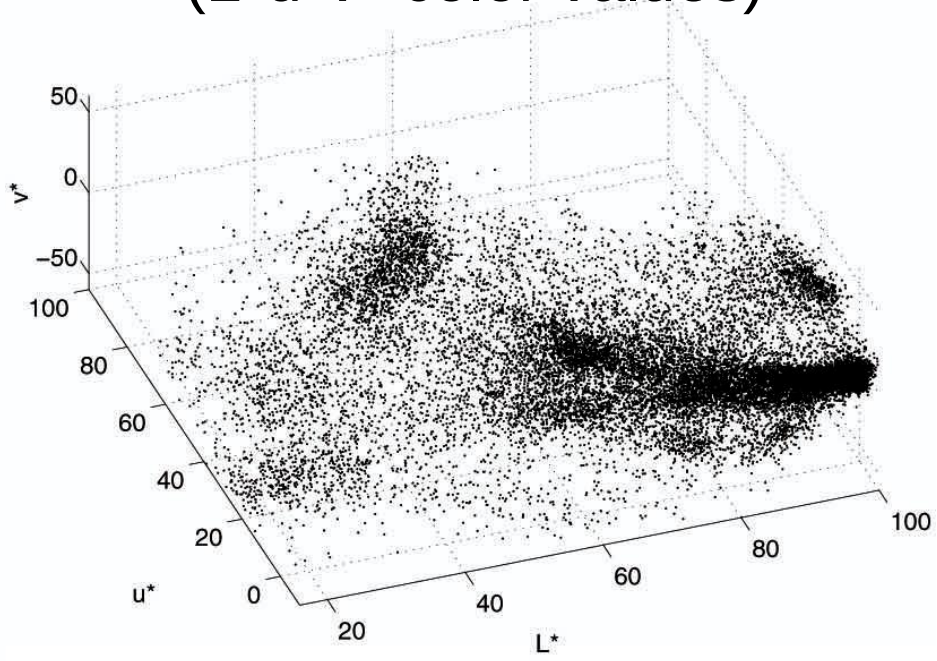
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002.

Mean shift algorithm (cont.)

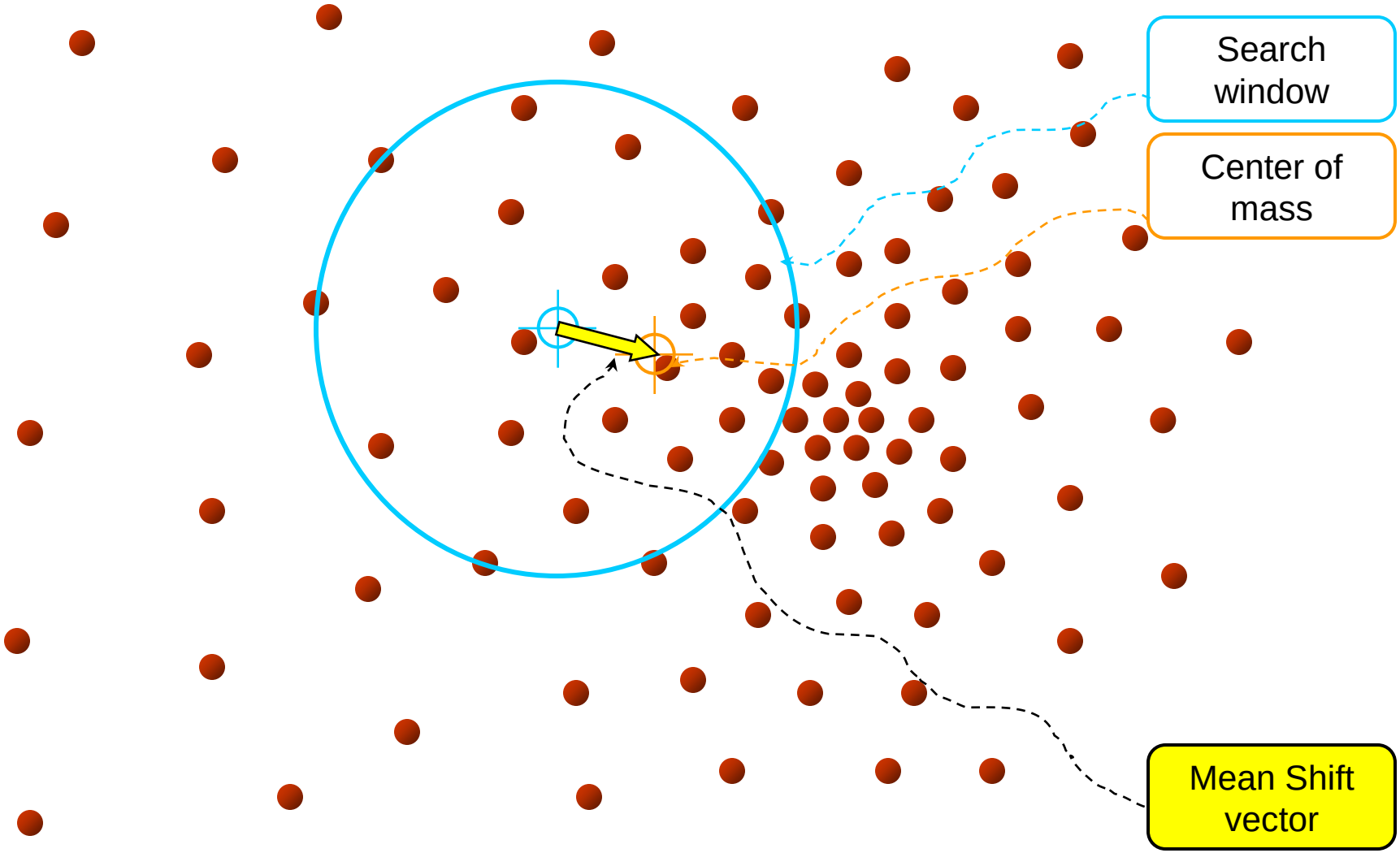
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space.

image

Feature space
(L*u*v* color values)

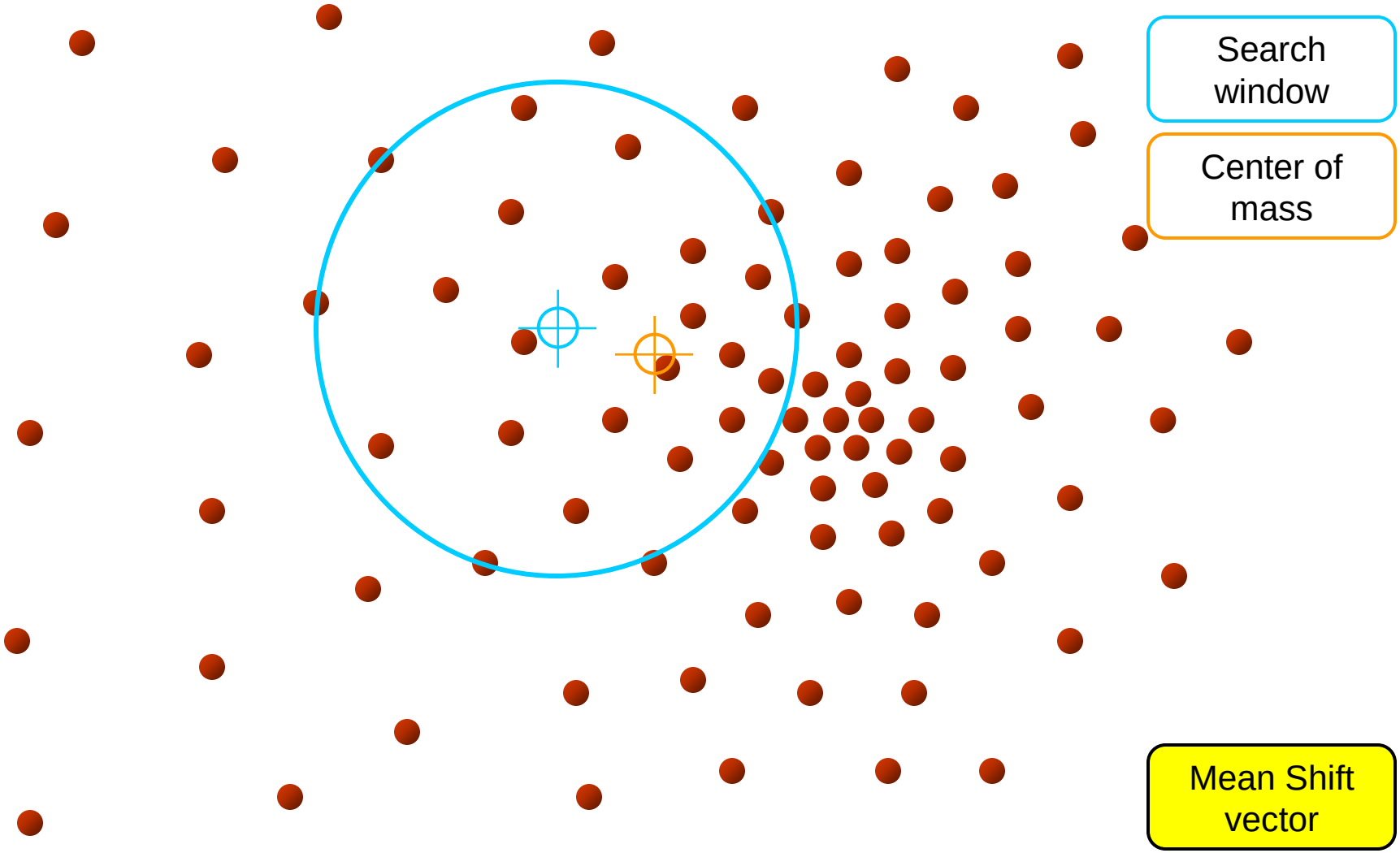


Mean shift algorithm (cont.)



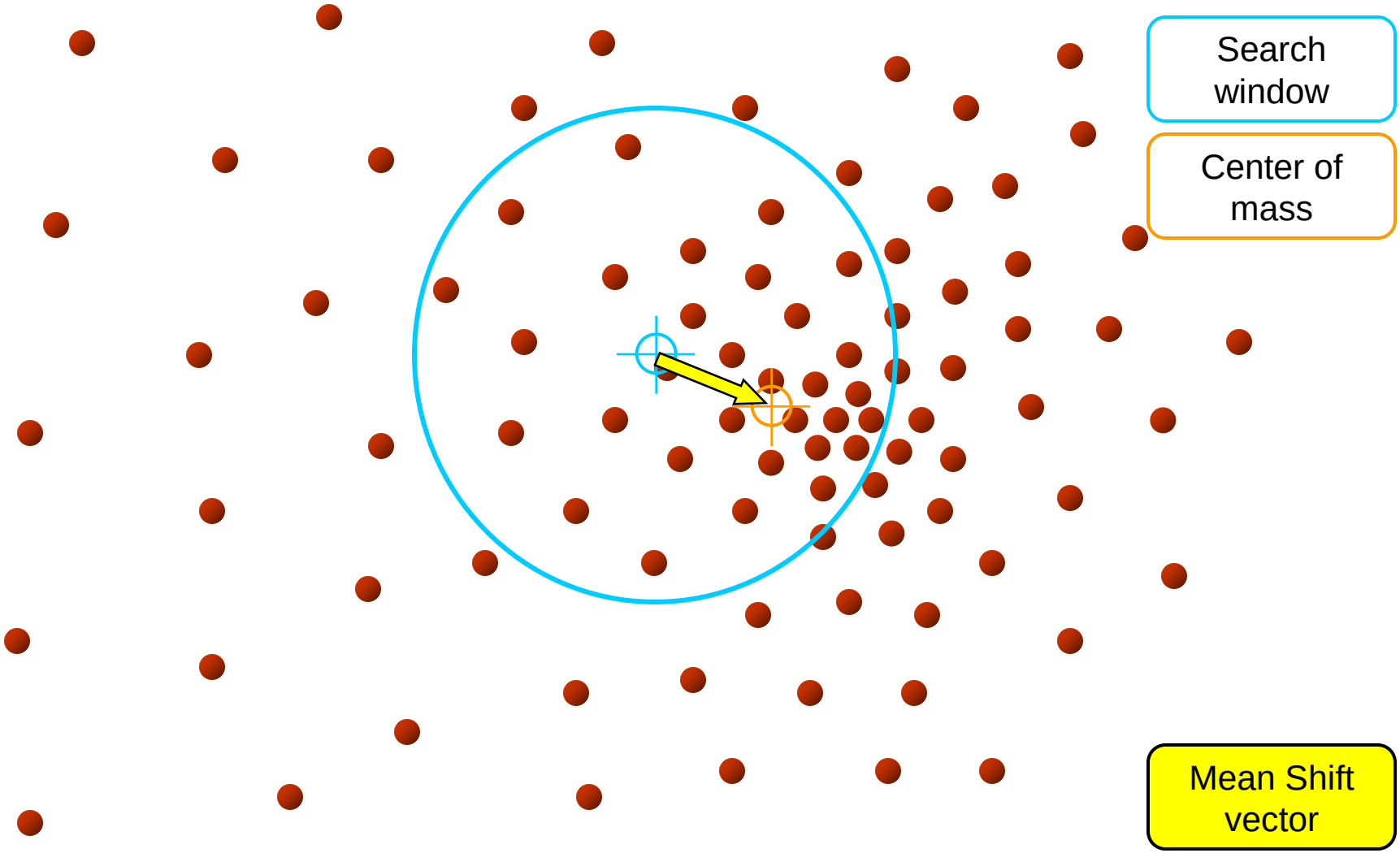
Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)



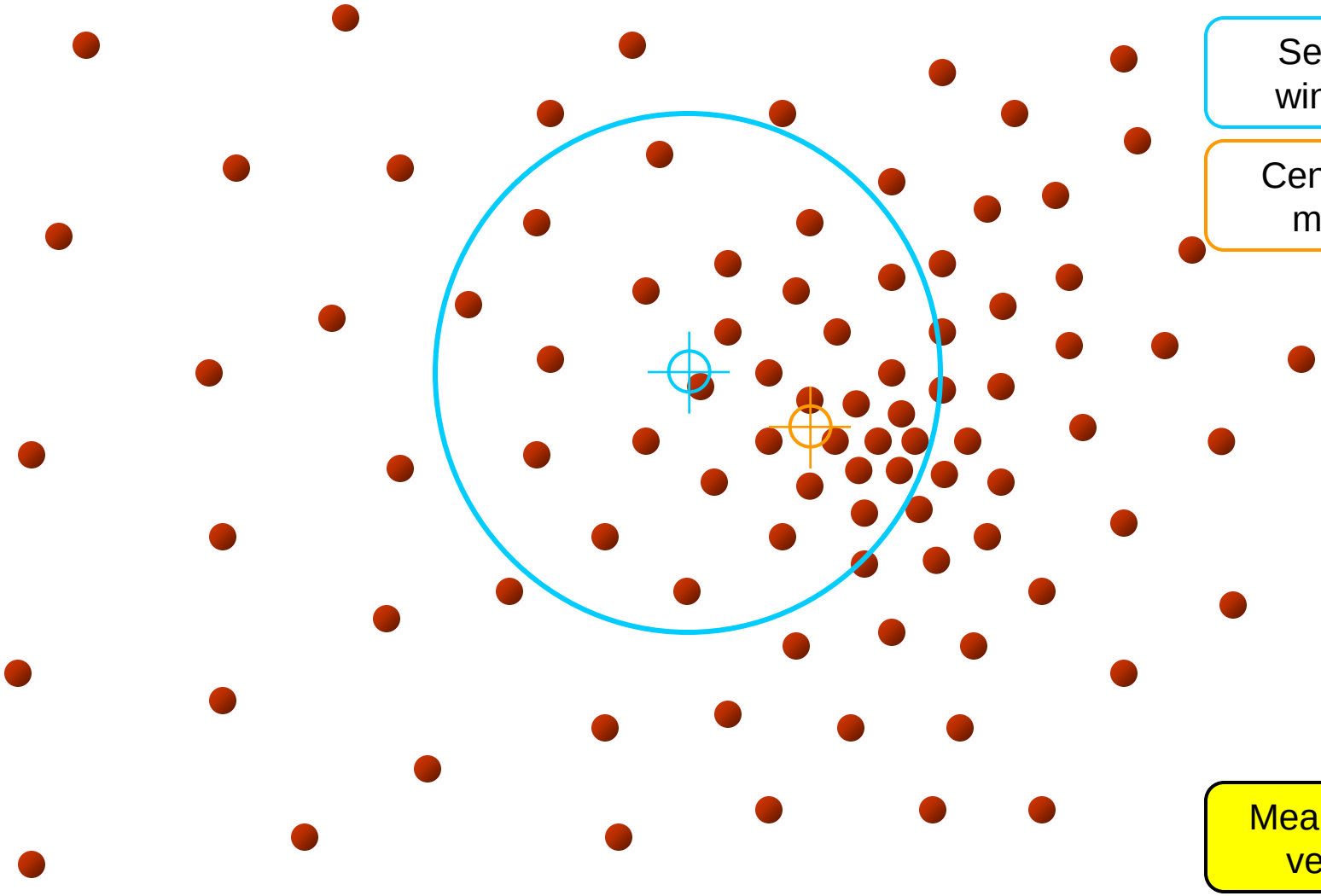
Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)



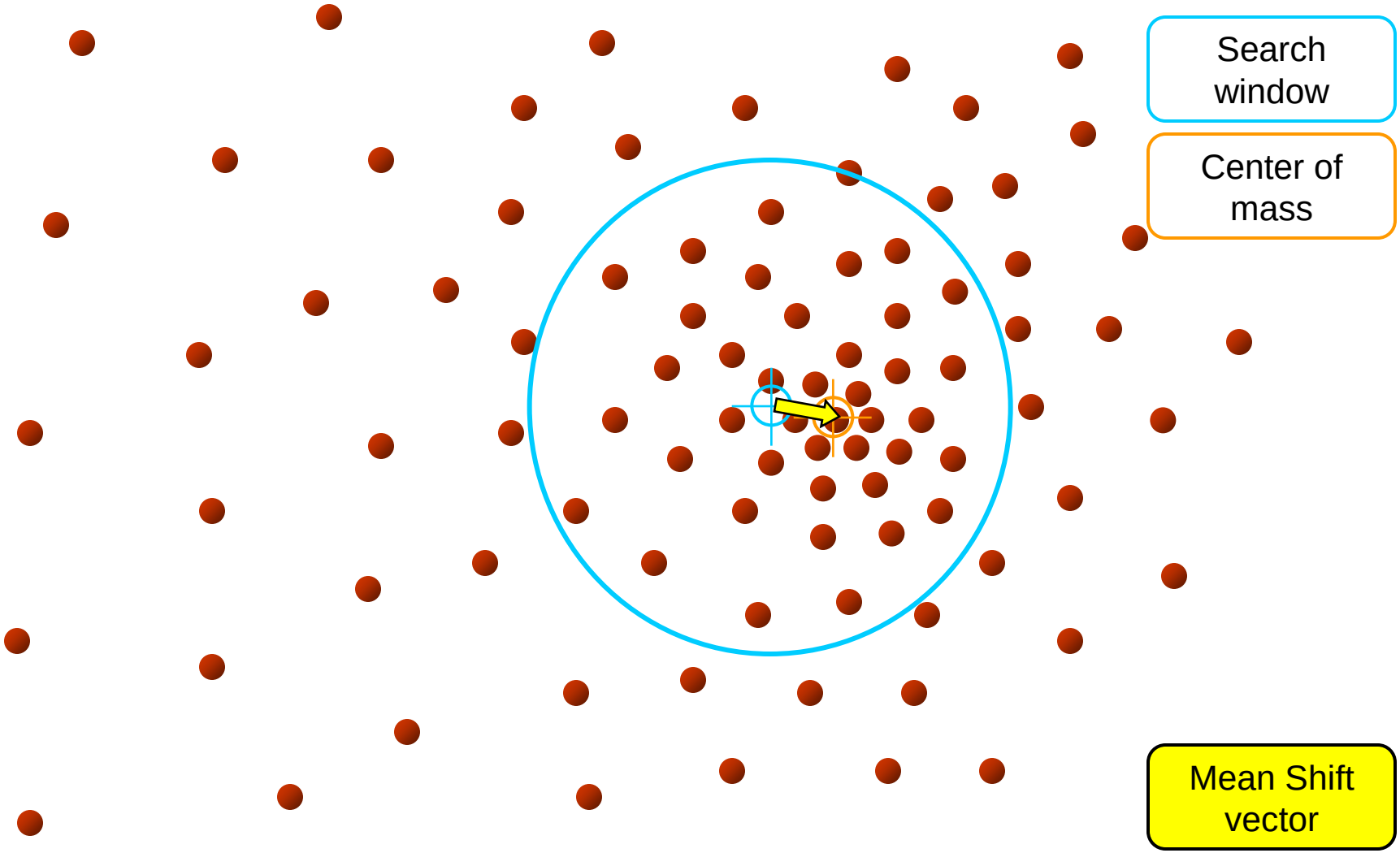
Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)



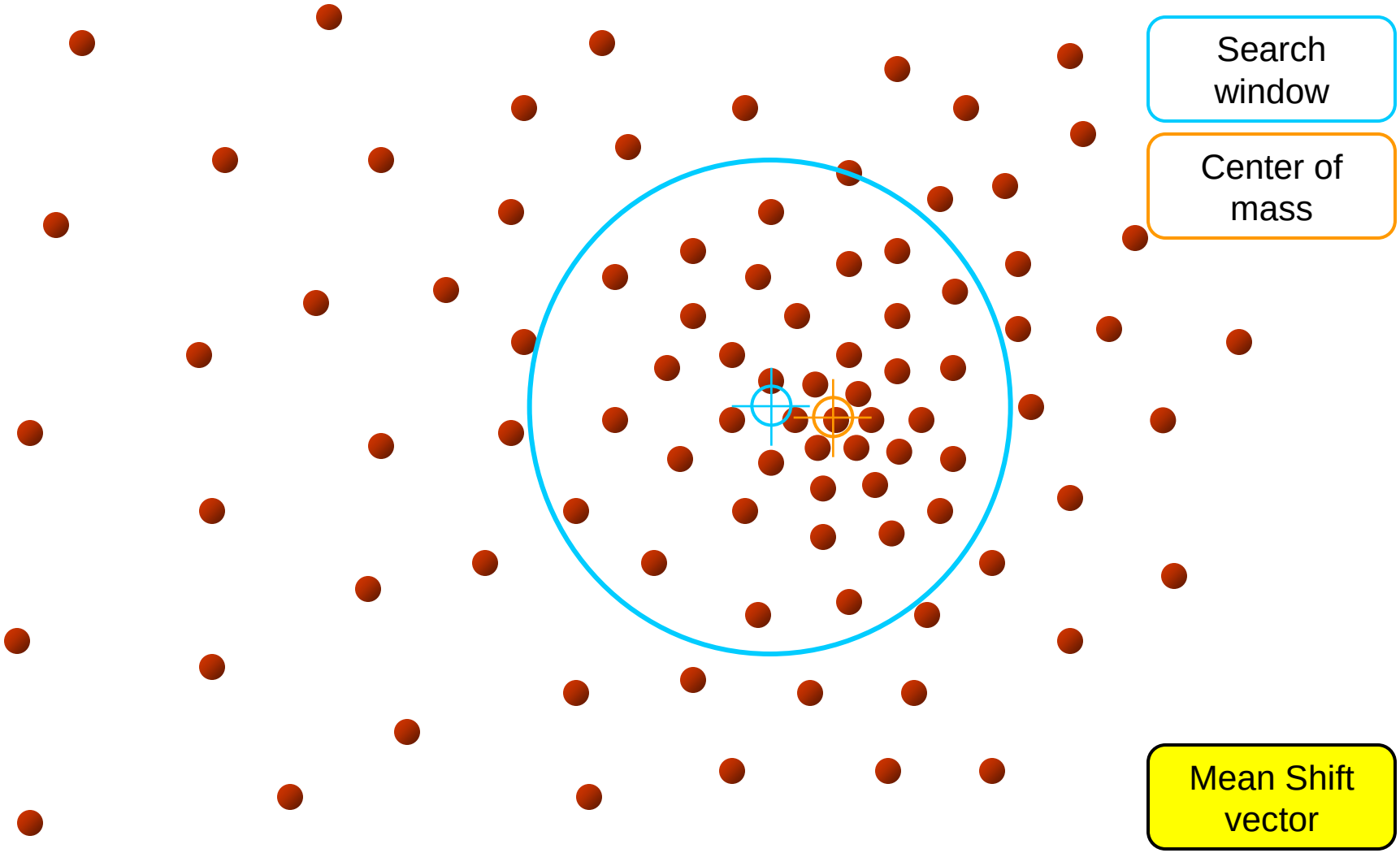
Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)

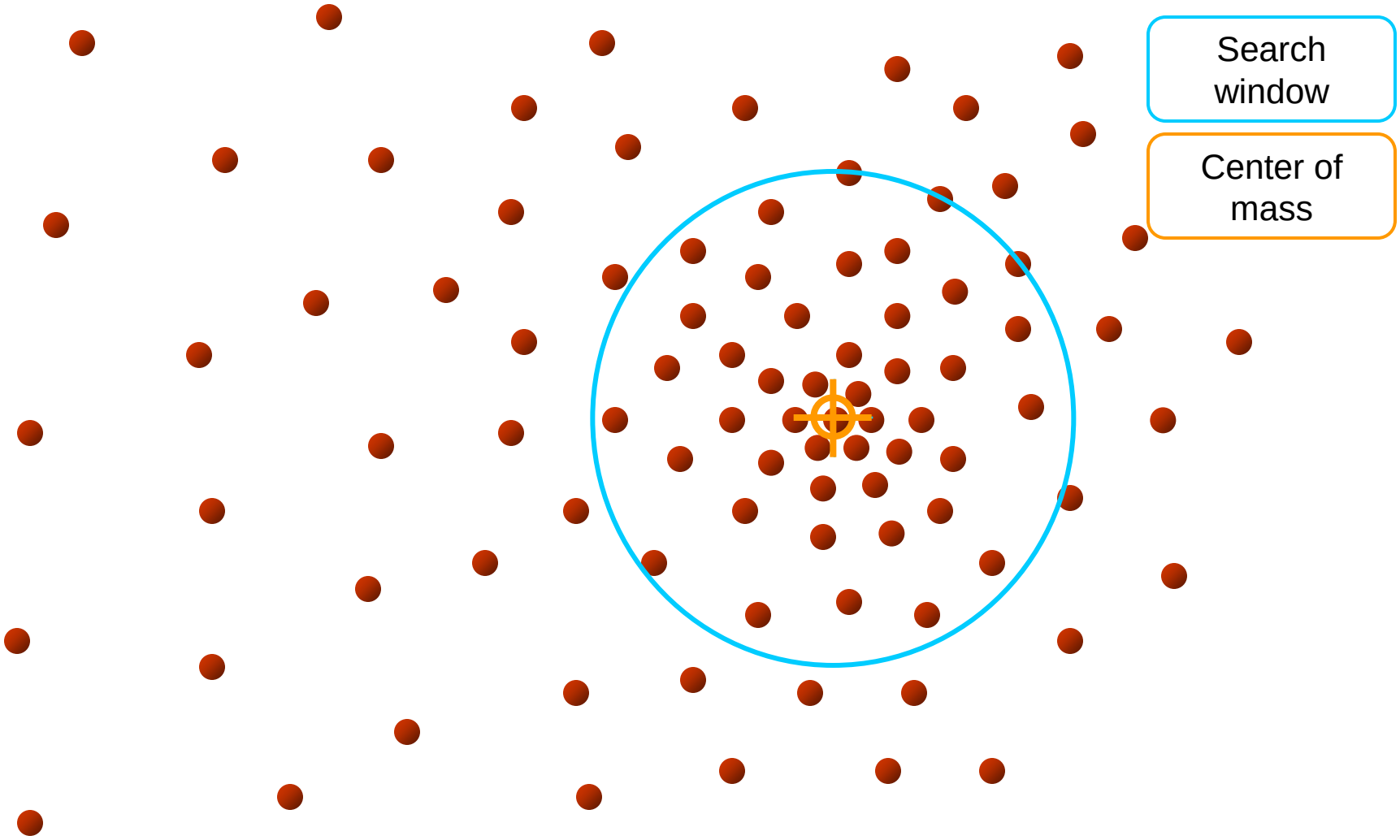


Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)



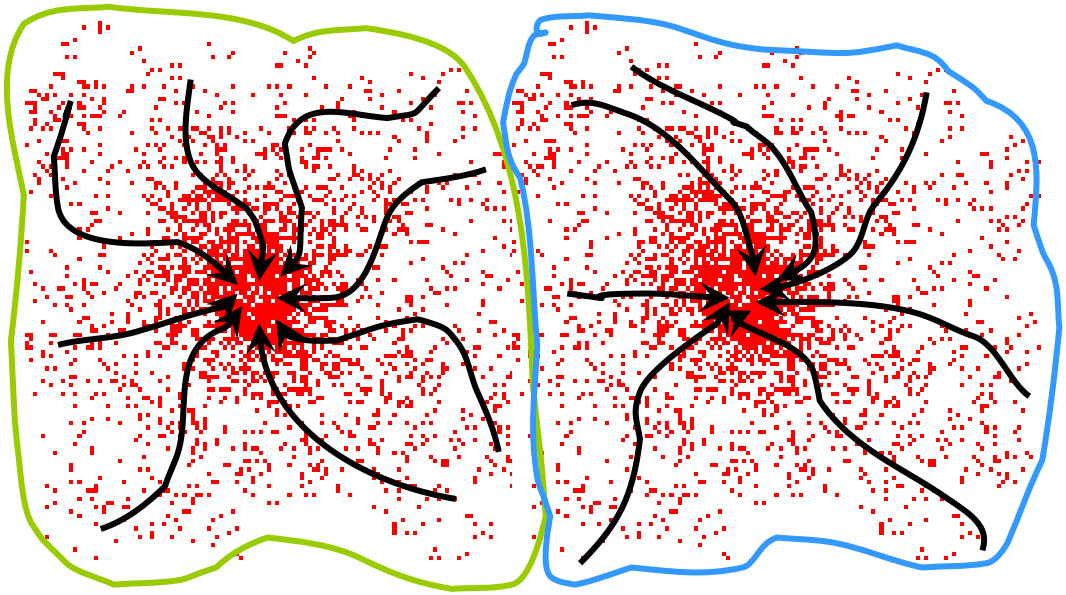
Mean shift algorithm (cont.)



Slide by Y. Ukrainitz & B. Sarel

Mean shift algorithm (cont.)

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



Slide by Y. Ukrainitz & B. Sarel

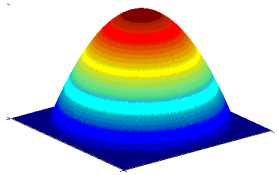
Mean Shift Fundamentals

$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

Estimate the density at a point ***x*** from a number of sample data points $x_1 \ldots x_n$

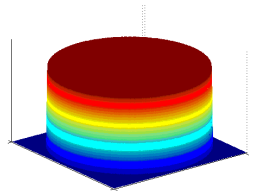
- Epanechnikov Kernel

$$K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$



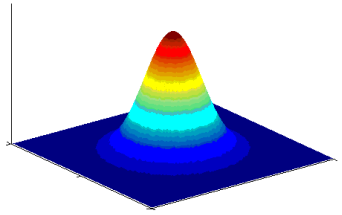
- Uniform Kernel

$$K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$



- Normal Kernel

$$K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$



Mean Shift Fundamentals (cont.)

$$\nabla P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla K(\mathbf{x} - \mathbf{x}_i)$$

Give up estimating the PDF !
Estimate **ONLY** the gradient

Using the Kernel form:

$$K(\mathbf{x} - \mathbf{x}_i) = ck\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

Size of window

We get :

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \cdot \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$$\mathbf{g}(\mathbf{x}) = -k'(\mathbf{x})$$

Mean Shift Fundamentals (cont.)

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\bullet\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

$$\mathrm{g}(\mathbf{x}) = -k'(\mathbf{x})$$

Mean Shift Fundamentals (cont.)

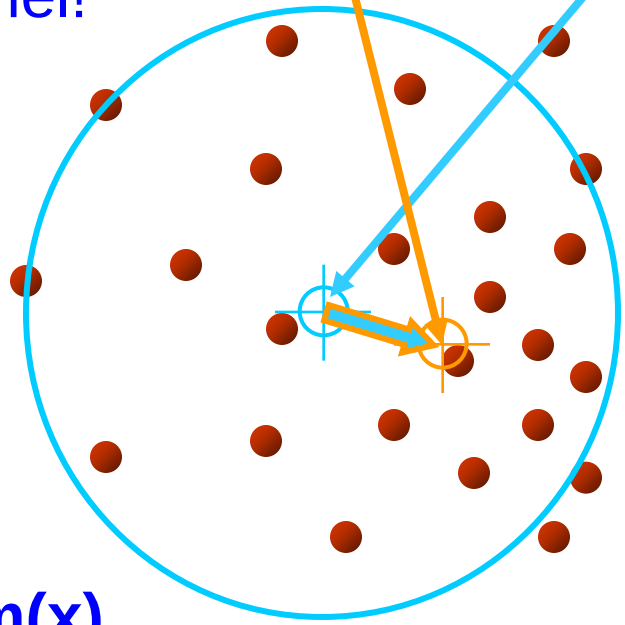
$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)\right]\cdot\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i\, g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}\right]$$

Density estimation with another kernel!

<u>Simple Mean Shift procedure</u>:

- Compute mean shift vector

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



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

$$g(\mathbf{x}) = -k'(\mathbf{x})$$

Mean shift clustering/segmentation

Start with an estimate of the mode $\boldsymbol{y}^{(0)}$ and a set of n data vectors $\boldsymbol{x}_i$ of dimension d, a scaling constant h, and g the derivative of the kernel profile

Until the update is tiny

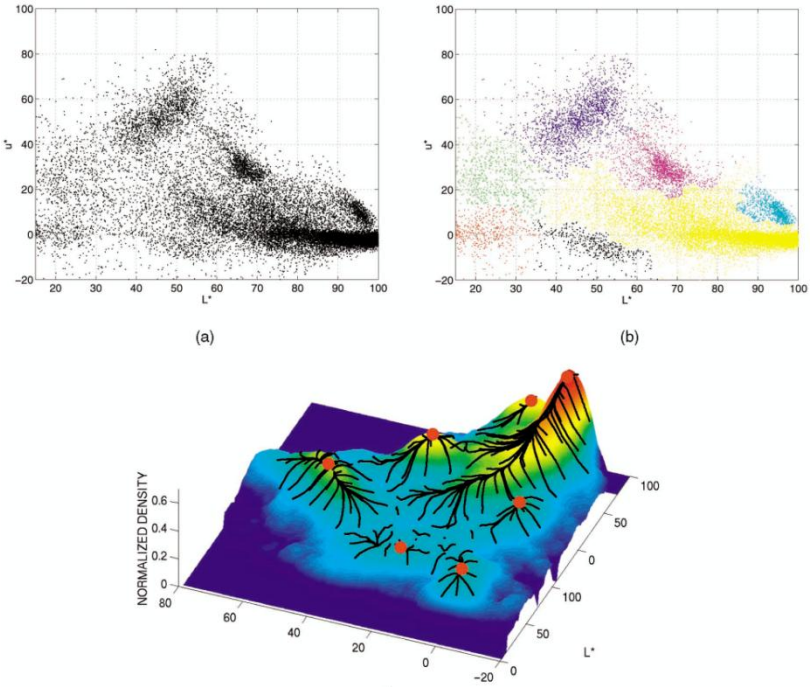
Form the new estimate

$$\boldsymbol{y}^{(j+1)} = \frac{\sum_i \boldsymbol{x}_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}{\sum_i g(\| \frac{\boldsymbol{x}_i - \boldsymbol{y}^{(j)}}{h} \|^2)}$$

Algorithm 9.5: Finding a Mode with Mean Shift.

Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc).
- Initialize windows at individual feature points.
- Perform mean shift for each window until convergence.
- Merge windows that end up near the same “peak” or mode.



Mean shift clustering/segmentation

- Color distances are not the same as position distances
- Use different kernels

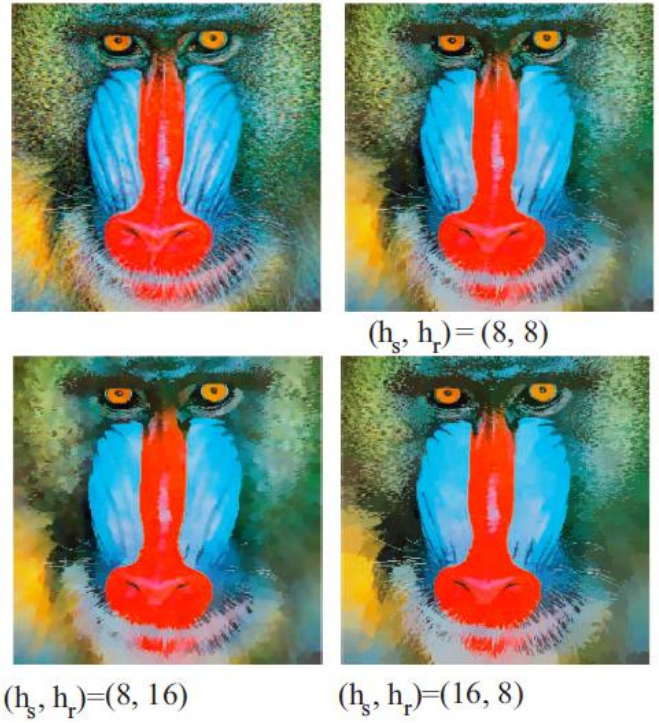
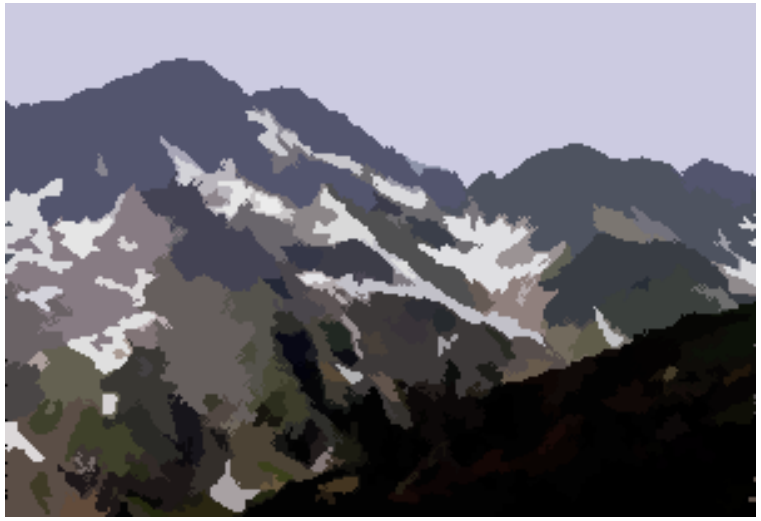


FIGURE 9.20: An image (**top left**) and mean shift modes obtained with different clustering scales for space h_s and appearance h_r. If h_s is small, the method must produce clusters that are relatively small and compact spatially because the kernel function smoothes over a relatively small radius and so will allow many distinct modes. If h_r is small, the clusters are compact in appearance; this means that small h_s and large h_r will produce small, blobby clusters that could span a range of appearances, whereas large h_s and small h_r will tend toward spatially complex and extended clusters with a small range of appearances. Cluster boundaries will try harder to follow level curves of intensity. *This figure was originally published as Figure 5 of "Mean Shift: A Robust Approach Toward Feature Space Analysis," by D. Comaniciu and P. Meer, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002* © *IEEE, 2002.*

Mean shift segmentation results



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

Mean shift segmentation results (cont.)

Mean shift segmentation results (cont.)

Mean shift pros and cons

- Pros
  - Does not assume spherical clusters.
  - Just a single parameter (window size).
  - Finds variable number of modes.
  - Robust to outliers.
- Cons
  - Output depends on window size.
  - Computationally expensive.
  - Does not scale well with dimension of feature space.

Relaxation labeling

- Probabilistic method.
  - Weights are assigned to pixels representing the probability to belong to a given class.
  - The weights are updated until stability is obtained.
- Convergence to probability 1 is not guaranteed for each pixel.
- However, it is commonly employed in computer vision applications.

Relaxation labeling (cont.)

- A set of M possible labels $L=\{l_1, l_2,\ldots, l_M\}$.
- $P_i(l_k)$ is the probability of the i^{th} pixel to belong to the k^{th} class.
  - They sum to one for each pixel.
- The labeling process starts with an initial estimate of probabilities for each pixel.
- A relaxation schedule modifies these probabilities.
- The procedure is repeated until stability.

Relaxation labeling (cont.)

- Popular methods often take stochastic approaches to update the probabilities.
- A simple approach is based on the compatibility of label probabilities with the ones of their neighboring pixels.
- Simple decision rule for the effect of a neighbor (e.g. in a foreground/background problem):
  - If P(neighbor being foreground)>0.5 then the contribution from the neighbor increments the same probability of the current pixel by a factor $c_p>0$.
  - Otherwise the probability decreases by $c_n<0$.

Relaxation labeling (cont.)

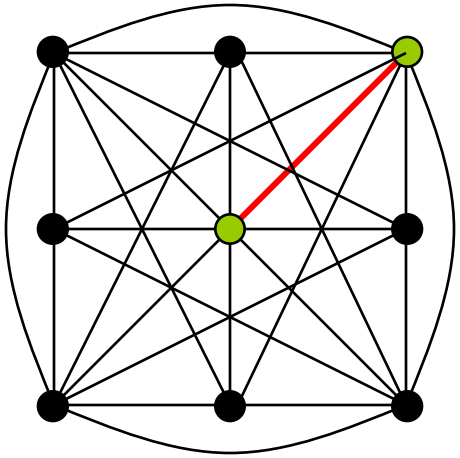
- Update in confidence for $P_i(l_k)$ for a two label problem:

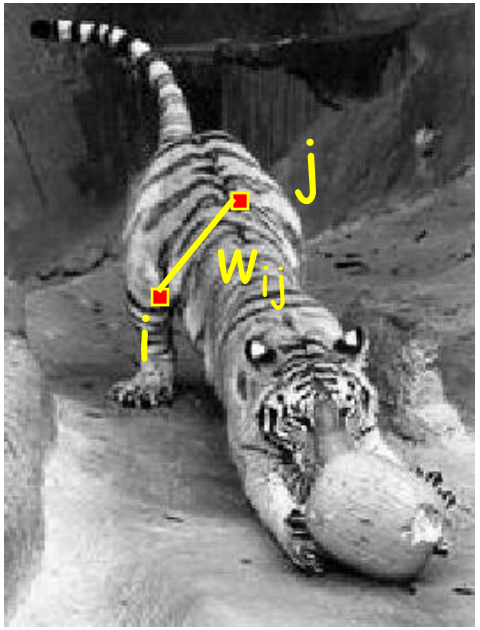
$$\delta P_i(l_k) = \frac{1}{|N_i|}\sum_{j\in N_i}\left[\sum_{l\in L} P(l_k \mid l_l)P_j(l_l)\right], \quad P(l_k \mid l_l) = \begin{cases} c_p, & P_j(l_l) > 0.5 \\ c_n, & \text{otherwise} \end{cases}$$

- Updated label:

$$P_i(l_k) = \frac{P_i(l_k)\left[1+\delta P_i(l_k)\right]}{\sum_l P_i(l_l)\left[1+\delta P_i(l_l)\right]}$$

Images as graphs



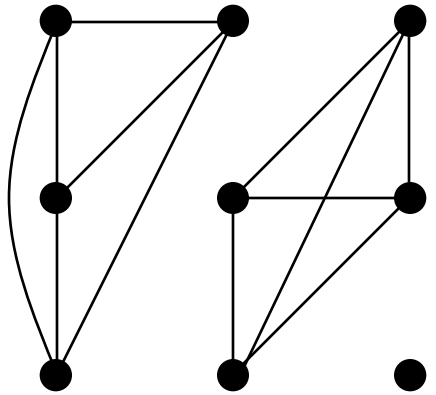


- Node for every pixel.
- Edge between every pair of pixels (or every pair of “sufficiently close” pixels).
- Each edge is weighted by the *affinity* or similarity of the two nodes.

Source: S. Seitz

Segmentation by graph partitioning

- Cut the 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.



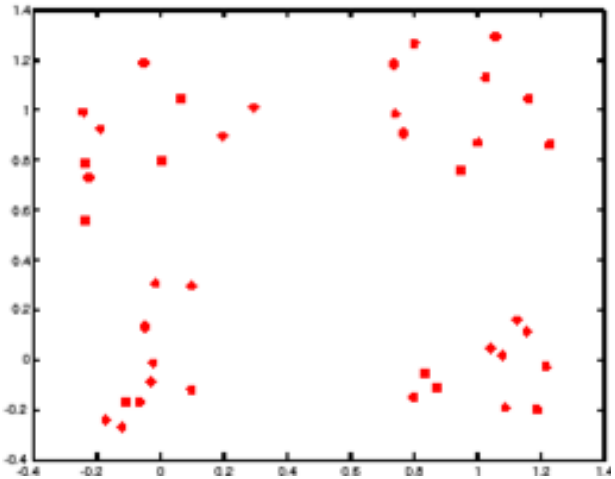
Source: S. Seitz

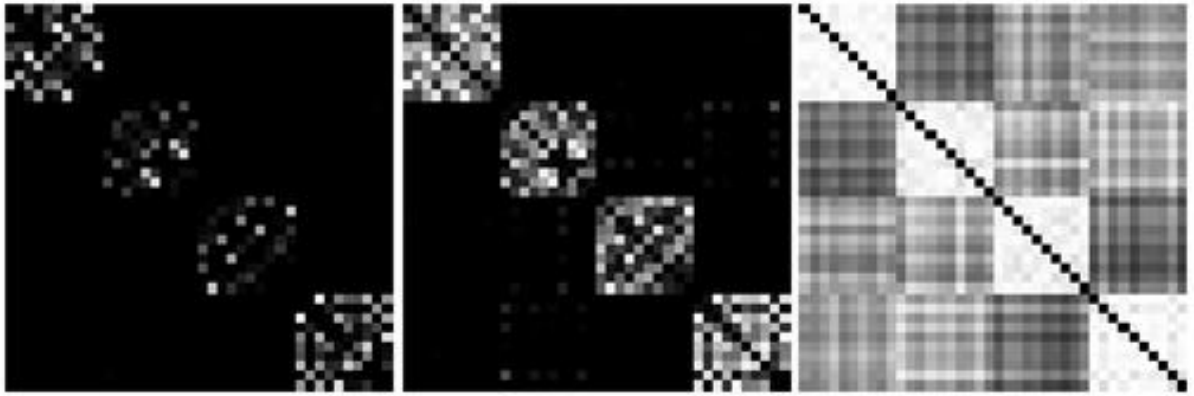
Measuring affinity

- Each pixel is represented by a feature vector, and a distance function is defined.
- We may convert the distance between two feature vectors into an affinity with the help of a generalized Gaussian kernel. The weight of the edge becomes:

$$w_{ij} = \exp\left(-\frac{1}{2\sigma^2} \mathrm{dist}(x_i, x_j)^2 \right)$$

Scale affects affinity

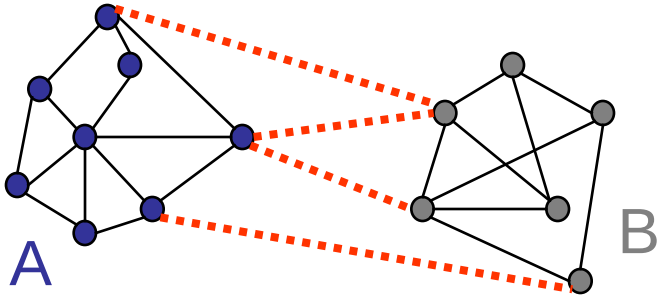




σ=0.1 σ=0.2 σ=1.0

- Small σ: group nearby points.
- Large σ: group distant points.

Graph cut



- Set of edges whose removal makes a graph disconnected.
- Cost of a cut: sum of weights of cut edges.

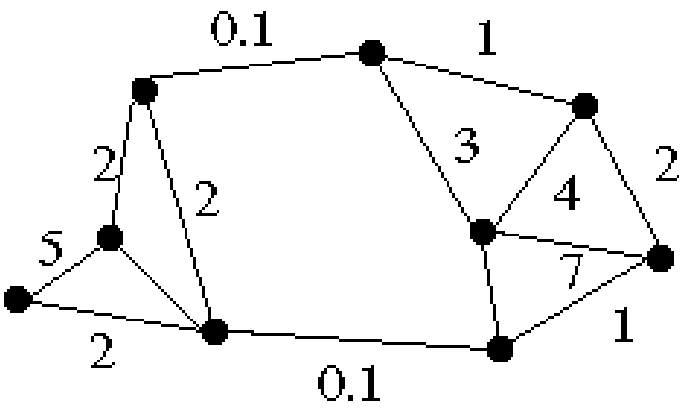
$$cut(A,B) = \sum_{i \in A,\, j \in B} w_{ij}$$

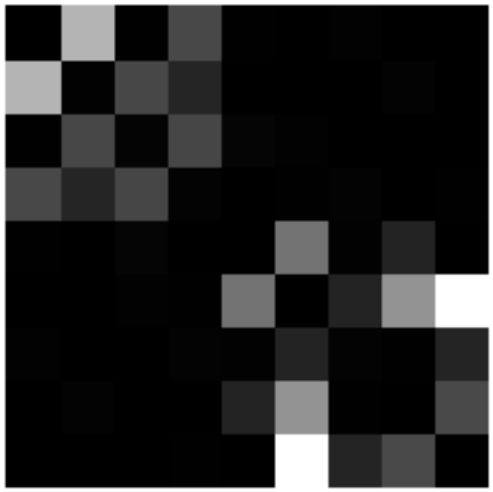
- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

Source: S. Seitz

Minimum cut

- We can have a segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist (Cormen et al. 2009)

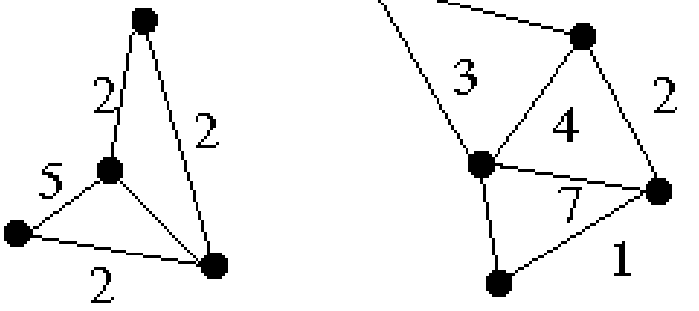


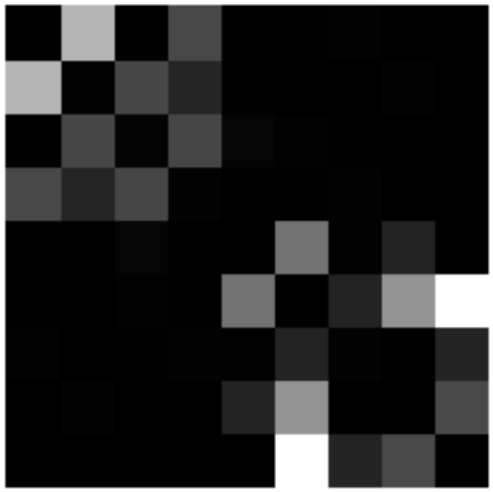


Minimum cut example

Minimum cut (cont.)

- We can have a segmentation by finding the *minimum cut* in a graph
  - Efficient algorithms exist (Cormen et al. 2009)

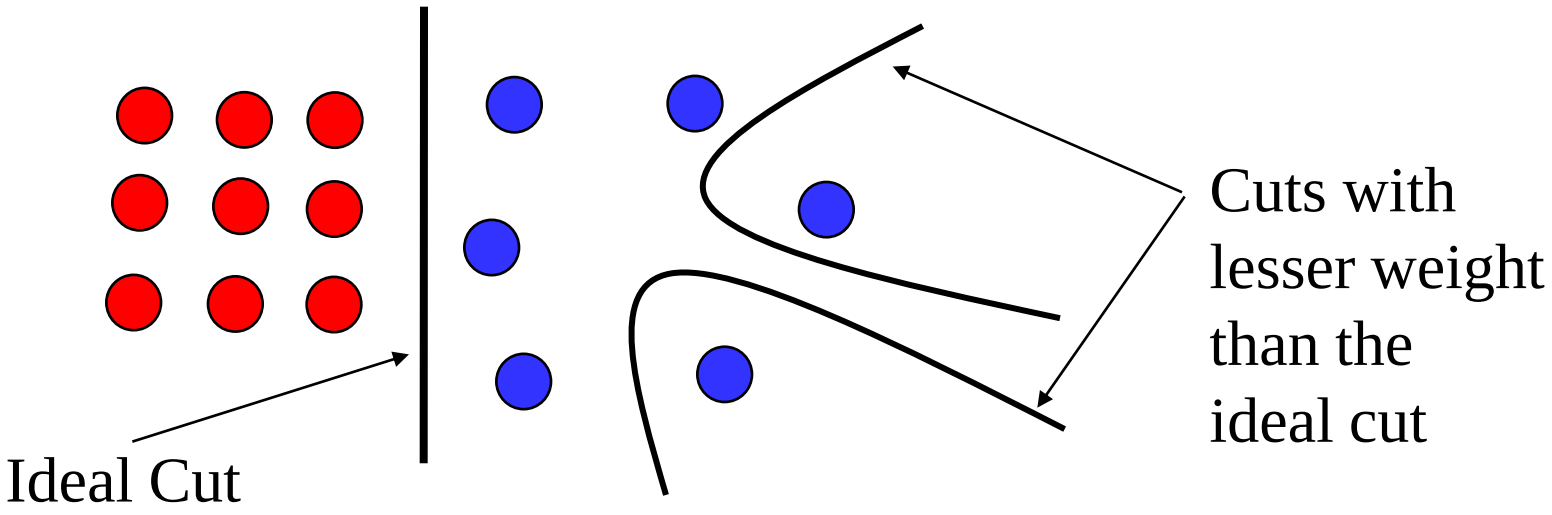




Minimum cut example

Minimum cut (cont.)

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



* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Agglomerative clustering with a graph

- Image as a weighted graph with edges between adjacent pixels measuring dissimilarity
  - Large weight means different pixels
- Every pixel forms a cluster
- Merge similar clusters until there are is no need to continue
- We need
  - Distance (difference) between two clusters for merging the closest ones
  - An index of how coherent a cluster is in order to stop clustering

Agglomerative clustering with a graph

- The difference between two clusters is the minimum weight edge connecting the clusters:

$$\mathrm{diff}(C_1, C_1) = \min_{v_1 \in C_1, v_2 \in C_2} w(v_1, v_2)$$

- The internal difference of a cluster is the largest weight of the minimum spanning tree of the cluster:

$$\mathrm{int}(C) = \max_{e \in M(C)} w(e)$$

- Fast algorithms exist for computing the minimum spanning tree (Cormen et al. 2009).

Agglomerative clustering with a graph

- The main idea is to merge clusters whose distance is smaller w.r.t. the internal distance of each one separately
- Small clusters
  - internal difference may be zero
  - comparing the difference between clusters to the internal distance of each one requires some care
  - In (Felzenswalb and Huttenlocher 2004) a function of two clusters is proposed:

$$\mathrm{Mint}(C_1, C_2) = \min\left(\mathrm{int}(C_1) + \tau(C_1), \mathrm{int}(C_2) + \tau(C_2)\right)$$

$\tau(C)$ is a term that biases $\mathrm{int}(C)$ upwards for small clusters, e.g. $\tau(C)=\mathrm{cnst}/|C|$

Agglomerative clustering with a graph (cont.)

Start with a set of clusters $\mathcal{C}_i$, one cluster per pixel.
Sort the edges in order of non-decreasing edge weight, so that $w(e_1) \geq w(e_2) \geq \ldots \geq w(e_r)$.

For $i = 1$ to r
    If the edge e_i lies inside a cluster
        do nothing
    Else
        One end is in cluster $\mathcal{C}_l$ and the other is in cluster $\mathcal{C}_m$
        If $diff(\mathcal{C}_l, \mathcal{C}_m) \leq MInt(\mathcal{C}_l, \mathcal{C}_m)$
            Merge $\mathcal{C}_l$ and $\mathcal{C}_m$ to produce a new set of clusters.

Report the remaining set of clusters.

Algorithm 9.8: Agglomerative Clustering with Graphs.

Agglomerative clustering with a graph (cont.)

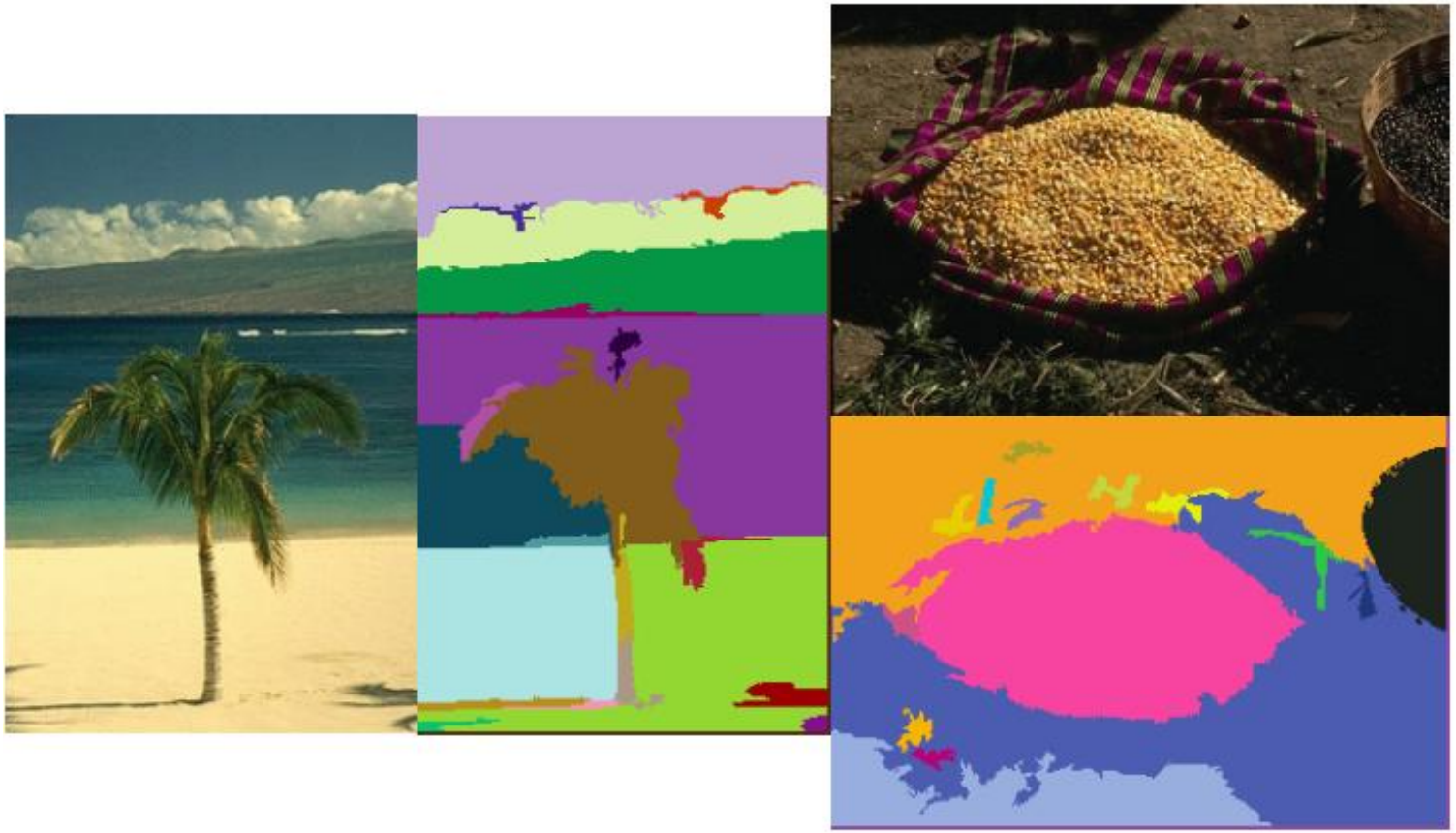


FIGURE 9.22: Images segmented using Algorithm 9.8, shown next to segments. Figures obtained from `http://people.cs.uchicago.edu/~pff/segment/`, by kind permission of Pedro Felzenszwalb.

Divisive clustering with a graph

- Useful in foreground/background segmentation
- We dispose a labeled map of pixels
  - Foreground
  - Background
  - Unknown
- We build models for foreground and background
- **Goal: label the unknown pixels as foreground or background**
- Important constraints on the labels
  - A pixel that looks like the foreground (background) examples should get a foreground (background) label
  - Neighboring pixels should tend to have similar labels

Divisive clustering with a graph (cont.)

- Boykov and Jolly (2001) rephrased the problem as the minimization of an energy function
- δ_i=-1 if the i-th pixel is background
- δ_i=1 if the i-th pixel is foreground
- $d_f(\boldsymbol{p}_i)$ a function comparing a pixel to foreground model
- $d_b(\boldsymbol{p}_i)$ a function comparing a pixel to background model
- $B(\boldsymbol{p}_i, \boldsymbol{p}_j)$ a non-negative symmetric encouraging neighboring pixels to have similar labels

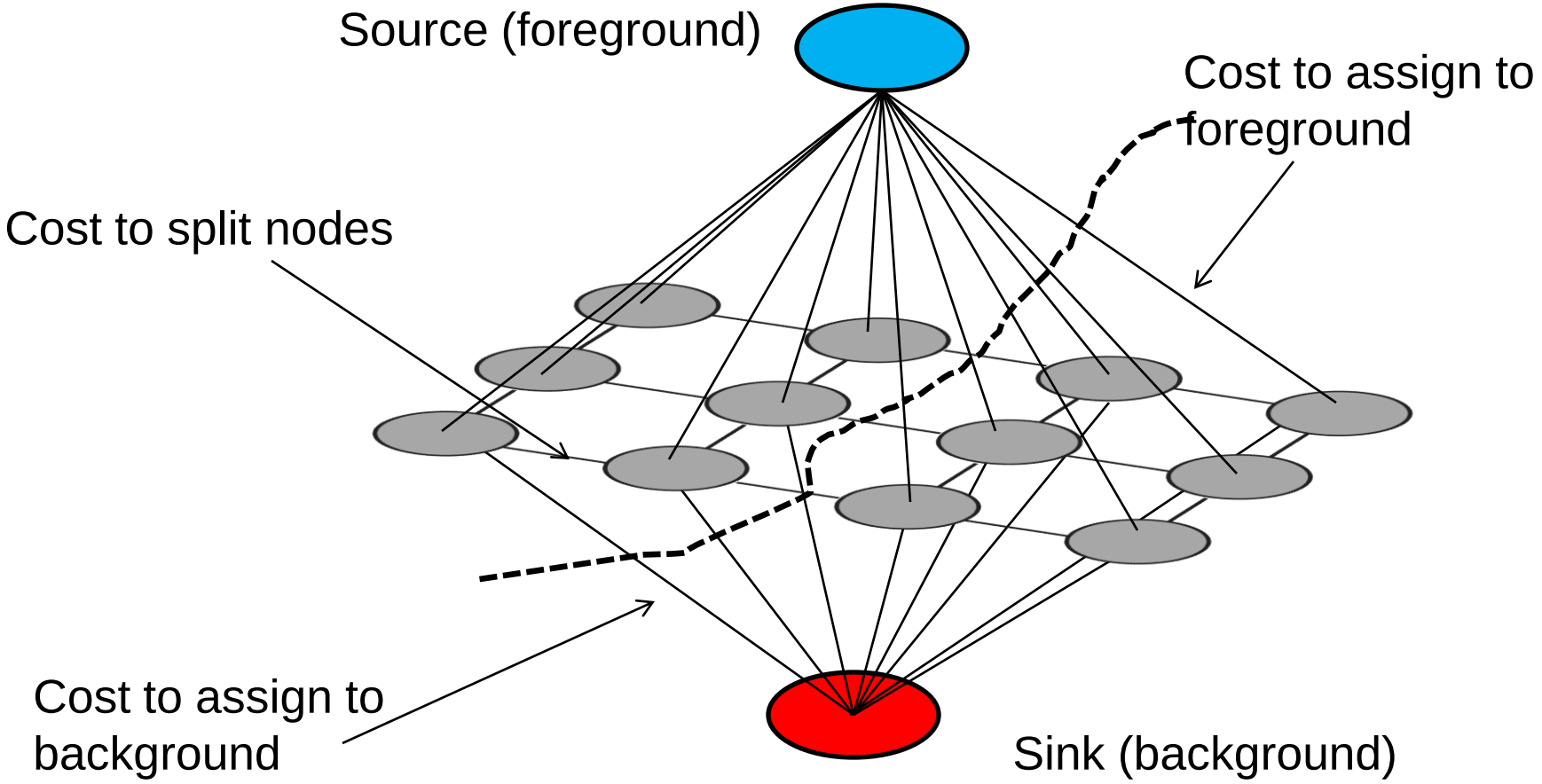
$$E(\delta) = \sum_{i \in I} d_f(\boldsymbol{p}_i) \frac{1}{2}(1+\delta_i) + \sum_{i \in I} d_b(\boldsymbol{p}_i) \frac{1}{2}(1-\delta_i) + \sum_{i \in I} \sum_{j \in N_i} B(\boldsymbol{p}_i, \boldsymbol{p}_j) \frac{1}{2}(1-\delta_i \delta_j)$$

Divisive clustering with a graph (cont.)

$$E(\delta) = \sum_{i \in I} d_f(\boldsymbol{p}_i) \frac{1}{2}(1+\delta_i) + \sum_{i \in I} d_b(\boldsymbol{p}_i) \frac{1}{2}(1-\delta_i) + \sum_{i \in I} \sum_{j \in N_i} B(\boldsymbol{p}_i, \boldsymbol{p}_j) \frac{1}{2}(1-\delta_i \delta_j)$$

- Hard minimization because it is a combinatorial problem (δ_i can take only two values).
- It may be rephrased as minimizing a cut on a graph (graph cut)
  - min-cut/max flow problem.
- The problem is polynomial and several specialized algorithms exist

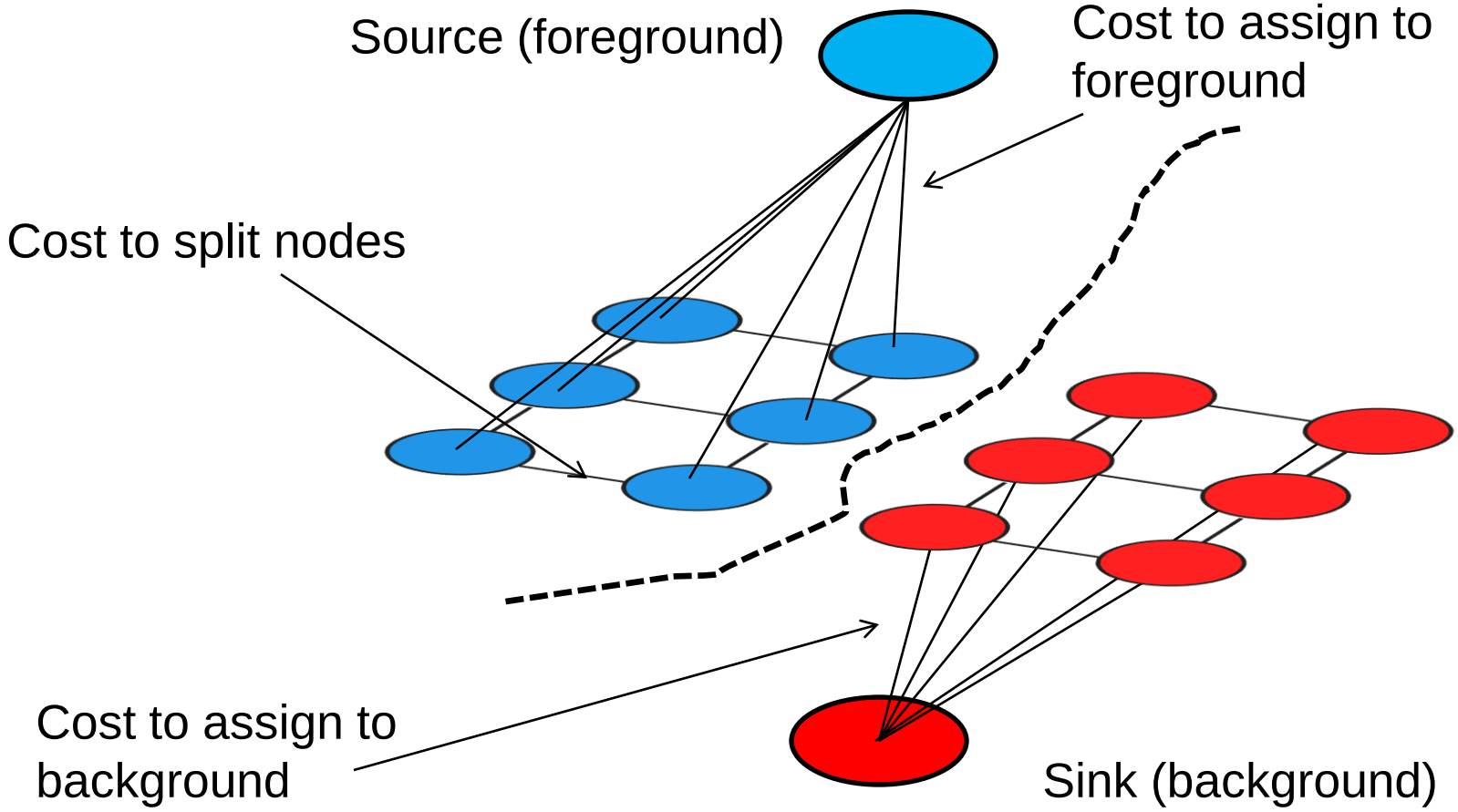
Divisive clustering with a graph (cont.)



$$E(\delta) = \sum_{i \in I} d_f(\boldsymbol{p}_i) \frac{1}{2}(1+\delta_i) + \sum_{i \in I} d_b(\boldsymbol{p}_i) \frac{1}{2}(1-\delta_i) + \sum_{i \in I} \sum_{j \in N_i} B(\boldsymbol{p}_i, \boldsymbol{p}_j) \frac{1}{2}(1-\delta_i \delta_j)$$

Figure: D. Hoeim, U. of Illinois

Divisive clustering with a graph (cont.)



$$E(\delta) = \sum_{i \in I} d_f(\boldsymbol{p}_i) \frac{1}{2}(1+\delta_i) + \sum_{i \in I} d_b(\boldsymbol{p}_i) \frac{1}{2}(1-\delta_i) + \sum_{i \in I} \sum_{j \in N_i} B(\boldsymbol{p}_i, \boldsymbol{p}_j) \frac{1}{2}(1-\delta_i \delta_j)$$

Figure: D. Hoeim, U. of Illinois

Divisive clustering with a graph (cont.)

Moderately straightforward examples

Figure: Microsoft Research

Divisive clustering with a graph (cont.)

Camouflage and low contrast

Fine structure

Harder case

More difficult examples

Figure: Microsoft Research

Eigenvectors and Segmentation

- Good foreground and background models are not generally available
  - Min cut will separate small groups of pixels
- Consider the image as a graph $G(V,E)$
  - with V being the set of vertices (pixels $i=1,\ldots N$).
  - E are the edges.
  - W is the affinity matrix between pixels.
- We want to segment it into two segments:
  - segment A containing "similar pixels" and segment and B containing the rest of the image.

Eigenvectors and Segmentation (cont.)

- We allow elements associated with cluster A to have a continuous weight a_i.
  - Large value for a_i means a strong connection to the cluster.
- A good cluster is one with elements having:
  - large weights a_i.
  - large values between them in the affinity matrix W.

Eigenvectors and Segmentation (cont.)

- An objective function expressing this assumption is:

$$E(a)=\sum_{i=1}^{N}\sum_{i=1}^{N}a_i w_{ij} a_j=\left[a_1 a_2 \ldots a_N\right]\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{N1} & w_2 & \cdots & w_{NN} \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}=a^T W a$$

- $E(a)$ is a sum of products of the form:
  - a_i : association of element i with the cluster.
  - w_{ij} : affinity between elements i and j.
  - a_j : association of element j with the cluster.

Eigenvectors and Segmentation (cont.)

- We now estimate vector a maximizing:

$$E(a) = a^T W a$$

subject to: $a^T a = \| a \|^2 = 1$

because scaling a scales the objective function.

- The Lagrangian is $J(a;\lambda) = a^T W a + \lambda(1 - a^T a)$

leading to the solution: $Wa = \lambda a$

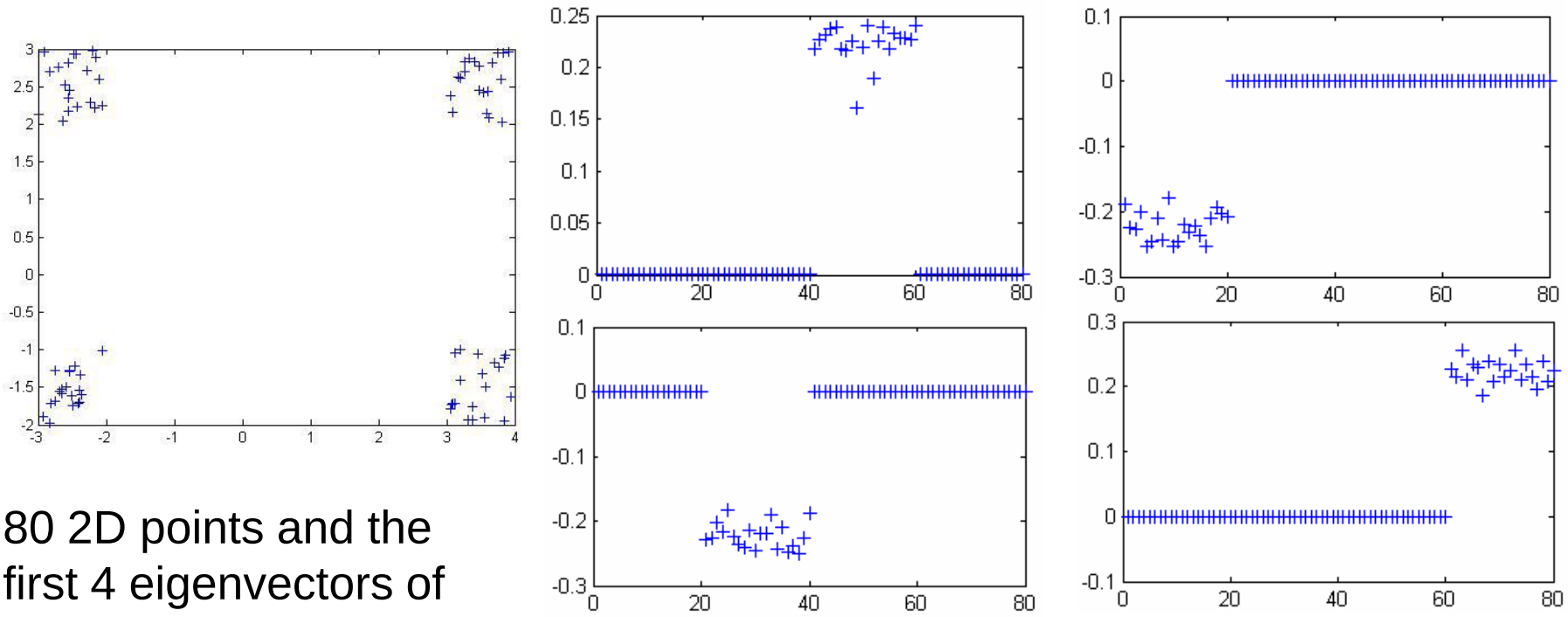
which is an eigenvector of W. The one maximizing J corresponds to the largest eigenvalue of W.

Eigenvectors and Segmentation (cont.

- Vector a is further thresholded
  - Elements of a over the threshold belong to the cluster.
  - Elements of a below the threshold are not associated with the cluster.
- More (M) segments may be obtained by
  - Recursively clustering the pixels associated with small values of vector a, or
  - Computing the first M eigenvectors of W and grouping their M -dimensional features (e.g. by K-means).

Eigenvectors and Segmentation (cont.

- Ideally, we expect that if here are M significant clusters, the eigenvectors corresponding to the M largest eigenvalues each represent a segment.
  - They would be vectors of a block padded with zeros.



80 2D points and the first 4 eigenvectors of the affinity matrix.

Average Association

- The presented technique may be reformulated in terms of an *association* between the elements of two clusters to be maximized:

$$\frac{assoc(A,A)}{|A|}+\frac{assoc(B,B)}{|B|},\quad assoc(A,A)=\sum_{i\in A,\, j\in A} w_{ij}$$

yielding the solution of the eigenvector corresponding to the largest eigenvalue of W:

$$Wa=\lambda a$$

Average Cut

- A similar approach minimizes the sum of edges to be cut in order to form two segments:

$$\frac{cut(A,B)}{|A|}+\frac{cut(A,B)}{|B|}$$

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

Average Cut (cont.)

- The associated cost function to be minimized, with respect to a, is:

$$E(a) = \sum_{i=1}^{N} \sum_{j=1}^{N} (a_i - a_j)^2 w_{ij}$$

- If pixels i and j are similar, that is w_{ij} is large, then if they do not belong to the same cluster, $E(a)$ is heavily penalized.
- If the pixels are not similar, the energy is not affected as w_{ij} approaches zero.

Average Cut (cont.)

To facilitate the proof let us define the diagonal matrix D with elements the sum of weights arriving at the i-th pixel:

$$d_{ii} = \sum_{j \in V} w_{ij}$$

$$E(a) = \sum_{i=1}^{N} \sum_{j=1}^{N} (a_i - a_j)^2 w_{ij}$$

$$= \sum_{i=1}^{N} a_i^2 \sum_{j=1}^{N} w_{ij} + \sum_{j=1}^{N} a_j^2 \sum_{i=1}^{N} w_{ij} - 2 \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_i w_{ij}$$

$$= \sum_{i=1}^{N} a_i^2 d_{ii} + \sum_{j=1}^{N} a_j^2 d_{jj} - 2 \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j w_{ij}$$

$$= 2a^T (D - W) a$$

Average Cut (cont.)

- Estimating vector a minimizing:

$$E(a) = 2a^T (D - W)a$$

subject to: $a^T a = \| a \|^2 = 1$

yields the solution of the eigenvector corresponding to the (second) *smallest* eigenvalue of D-W:

$$(D - W)a = \lambda a$$

Normalized cut

- Average association tends to find ‘tight” clusters in the graph.
- Average cut tends to keep a balance but it is not always guaranteed that that the two partitions will have a tight group similarity.
- The normalized cut (Ncut) tries to keep a balance between finding clustering (tight clusters) and segmentation (split clusters).
- It may be shown that both of the previous methods are approximations of Ncut.

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

Normalized cut (cont.)

- Ncut is defined as:

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

- The cost of the cut is a small fraction of the total affinity in each group.
- Minimization is NP-hard. An approximate solution may be found as in the previous cases.

Normalized cut (cont.)

- Create vector a having components the weights associating each pixel with a cluster.

$$a_i = \begin{cases} \dfrac{1}{assoc(A,V)} = \dfrac{1}{\sum_{i\in A,\, j\in V} w_{ij}}, & \text{if } i \in A \\ -\dfrac{1}{assoc(B,V)} = -\dfrac{1}{\sum_{i\in B,\, j\in V} w_{ij}} & \text{if } i \in B \end{cases}$$

- Each a_i is a cluster indicator of the corresponding pixel.

Normalized Cut (cont.)

- As before, we may show that:

The contribution is 0 for points in the same cluster

$$a^T(D-W)a = \frac{1}{2}\sum_{i\in V}\sum_{j\in V}(a_i - a_j)^2 w_{ij} = \sum_{i\in A}\sum_{j\in B}(a_i - a_j)^2 w_{ij}$$

$$= \left(\frac{1}{assoc(A,V)} + \frac{1}{assoc(B,V)}\right)^2 \sum_{i\in A}\sum_{j\in B} w_{ij}$$

$$= \left(\frac{1}{assoc(A,V)} + \frac{1}{assoc(B,V)}\right)^2 cut(A,B)$$

Normalized Cut (cont.)

- Also,

$$a^T Da = \sum_{i \in A} a_i^2 d_{ii} + \sum_{j \in B} a_j^2 d_{jj}$$

$$= \frac{1}{assoc(A,V)^2} assoc(A,V) + \frac{1}{assoc(B,V)^2} assoc(B,V)$$

$$= \frac{1}{assoc(A,V)} + \frac{1}{assoc(B,V)}$$

Normalized Cut (cont.)

- We now combine

$$a^T Da = \frac{1}{assoc(A,V)} + \frac{1}{assoc(B,V)} \quad \text{with}$$

$$a^T (D-W)a = \left(\frac{1}{assoc(A,V)} + \frac{1}{assoc(B,V)} \right)^2 cut(A,B)$$

$$= \left(a^T Da \right)^2 cut(A,B)$$

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

Normalized Cut (cont.)

It turns out that the Ncut objective function is equivalent to minimizing:

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

or minimizing:

$$E(a) = a^T(D-W)a + \lambda a^T Da$$

which corresponds to finding the eigenvector correponding to the (second) smallest eigenvalue of

$$D^{-1/2}(D-W)D^{-1/2}$$

Normalized Cut (cont.)

Remember the Average Cut minimization problem is equivalent to minimizing:

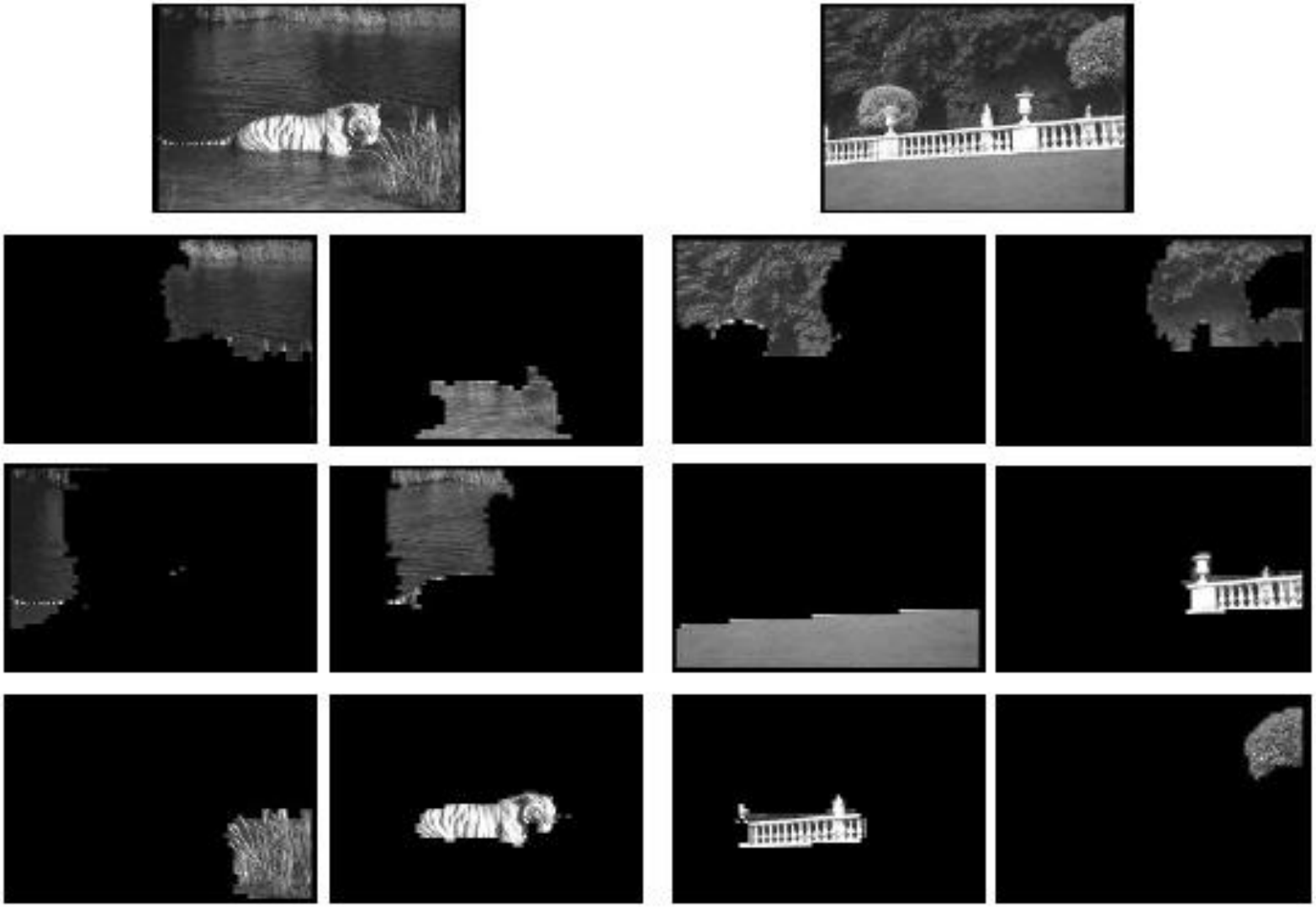
$$E(a) = a^T (D - W) a \quad \text{subject to} \quad a^T a = 1$$

The Ncut problem is equivalent to minimizing:

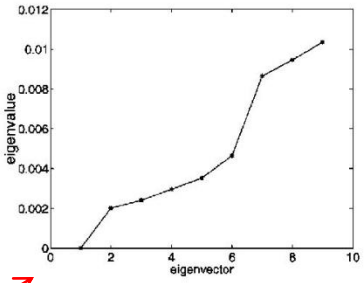
$$E(a) = a^T (D - W) a \quad \text{subject to} \quad a^T D a = 1$$

The larger the value of D_{ii}, the more important is the i-th sample.

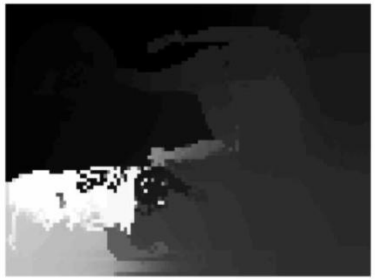
Normalized Cut (cont.)



Normalized Cut (cont.)



(a)



(b)

(c)

The magnitudes of the eigenvalues provide a hint for the number of clusters.

(d)

(e)

(f)

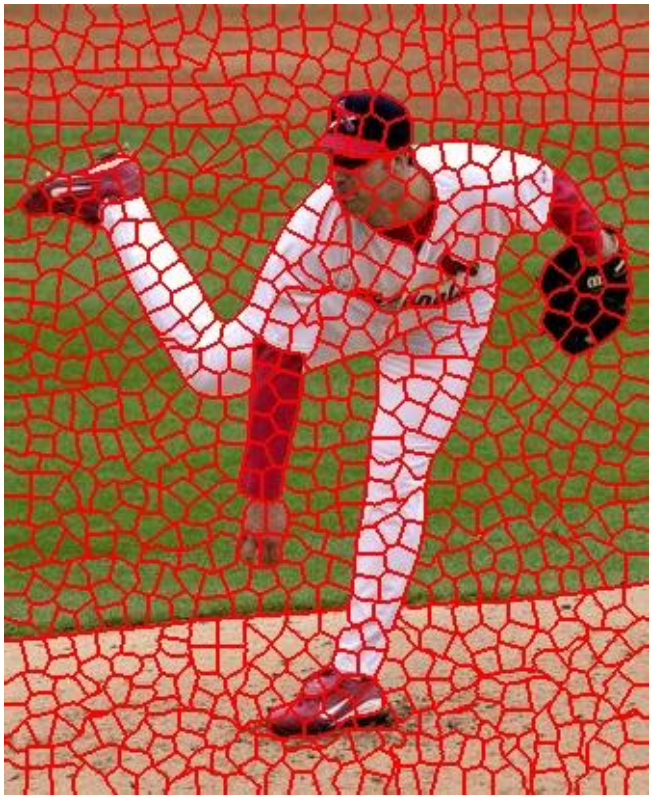
(g)

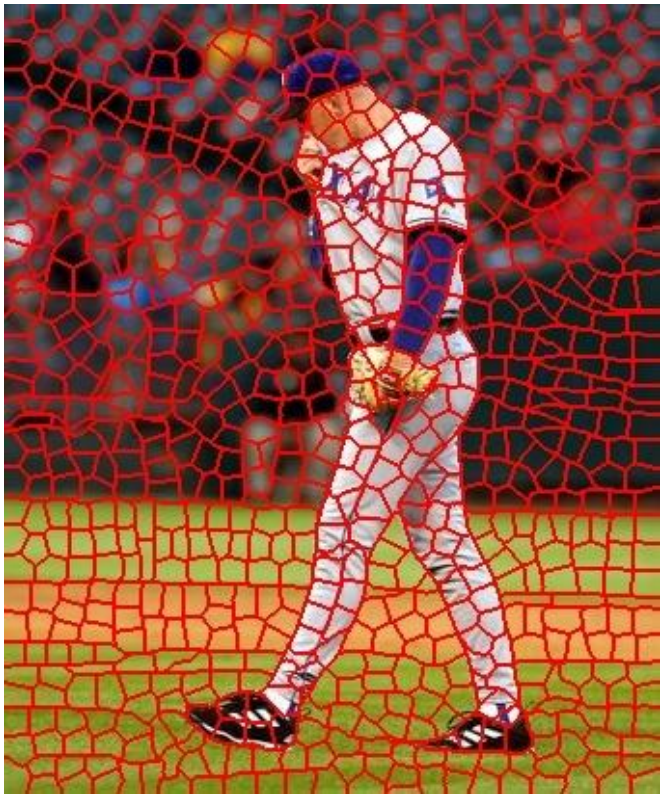
(h)

(i)

Normalized Cut (cont.)

- Superpixels.
- Unsupervised bottom-up process.





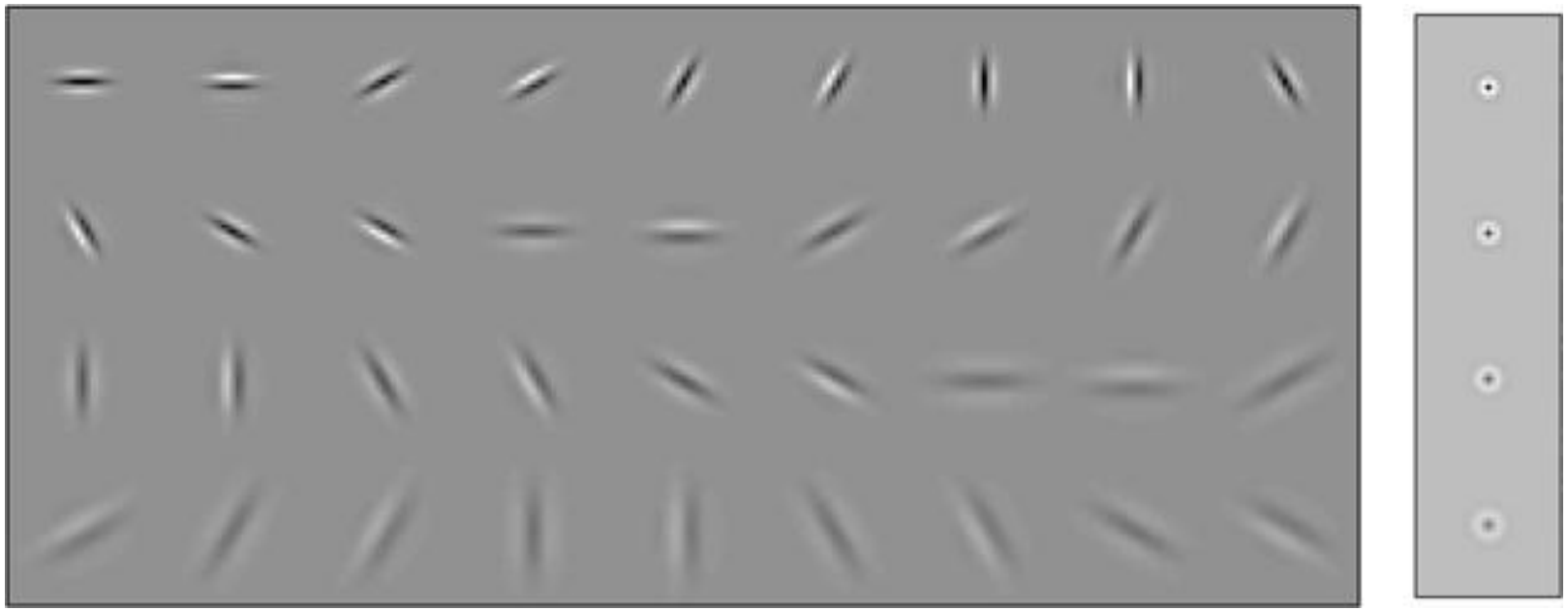
X. Ren and J. Malik. Learning a classification model for segmentation. ICCV 2003.

Using texture features for segmentation

- How to segment images that are a "mosaic of textures"?

Using texture features for segmentation (cont.)

- Convolve image with a bank of filters.



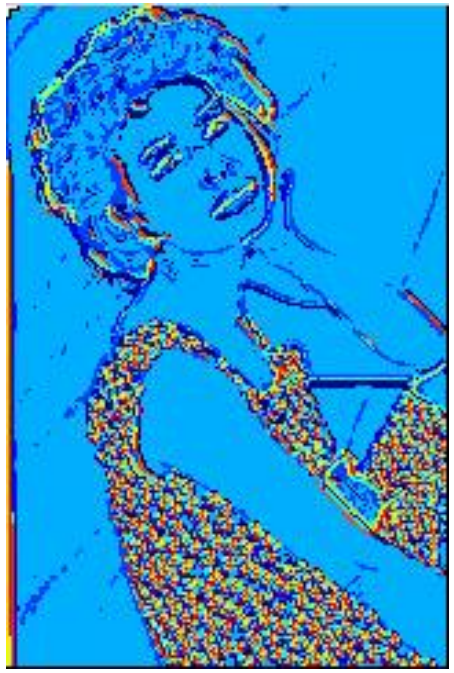
J. Malik, S. Belongie, T. Leung and J. Shi. "*Contour and Texture Analysis for Image Segmentation*". IJCV 43(1),7-27,2001.

Using texture features for segmentation (cont.)

- Find textons by clustering vectors of filter bank outputs.

Image

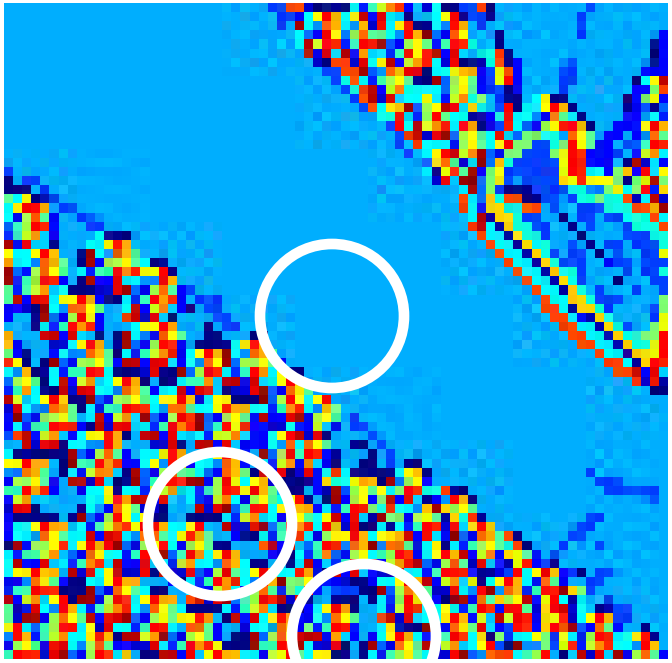
Texton map



J. Malik, S. Belongie, T. Leung and J. Shi. "*Contour and Texture Analysis for Image Segmentation*". IJCV 43(1),7-27,2001.

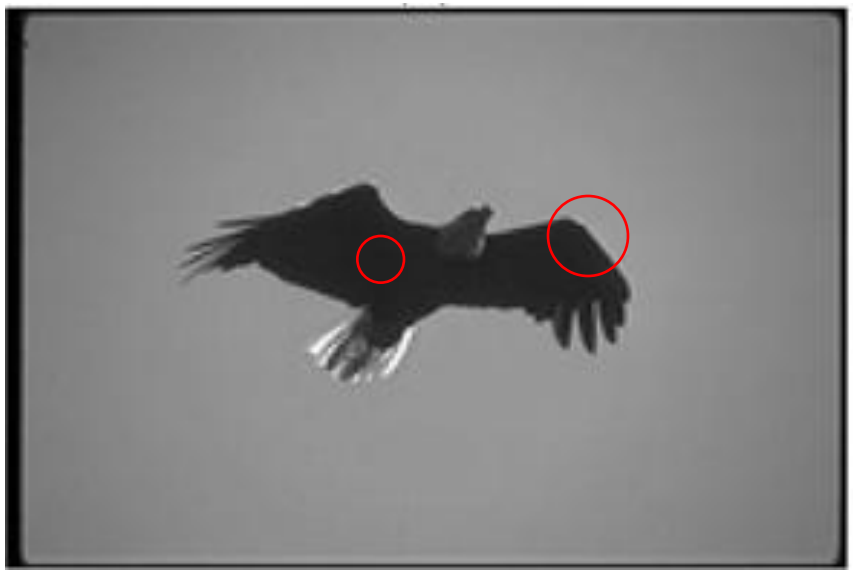
Using texture features for segmentation (cont.)

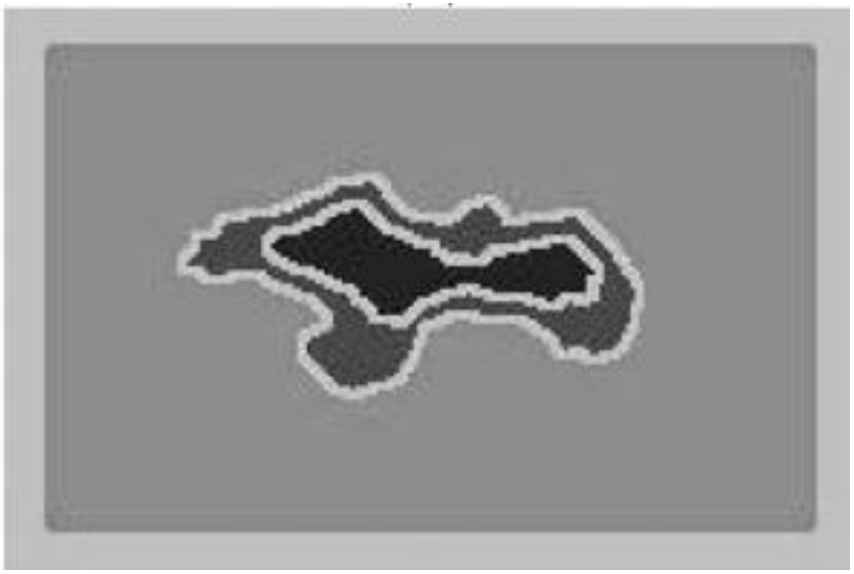
- The final texture feature is a texton histogram computed over image windows at some “local scale”.



J. Malik, S. Belongie, T. Leung and J. Shi. "*Contour and Texture Analysis for Image Segmentation*". IJCV 43(1),7-27,2001.

Pitfall of texture features





- Possible solution: check for “intervening contours” when computing connection weights.

J. Malik, S. Belongie, T. Leung and J. Shi. "*Contour and Texture Analysis for Image Segmentation*". IJCV 43(1),7-27,2001.

Example Results

Results: Berkeley Segmentation Engine

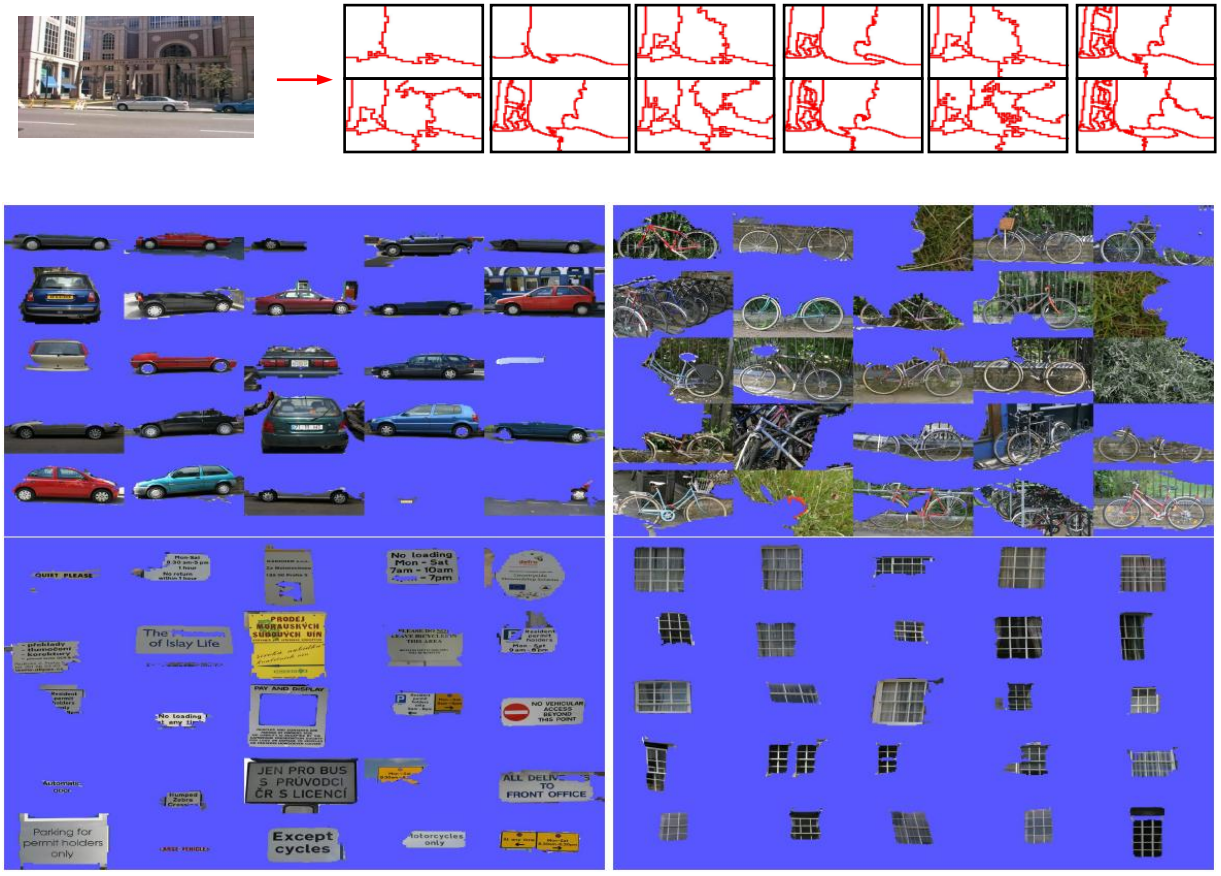
http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: Pros and cons

- Pros
  - Generic framework, can be used with many different features and affinity formulations.
- Cons
  - High storage requirement and time complexity.
  - Bias towards partitioning into equal segments.

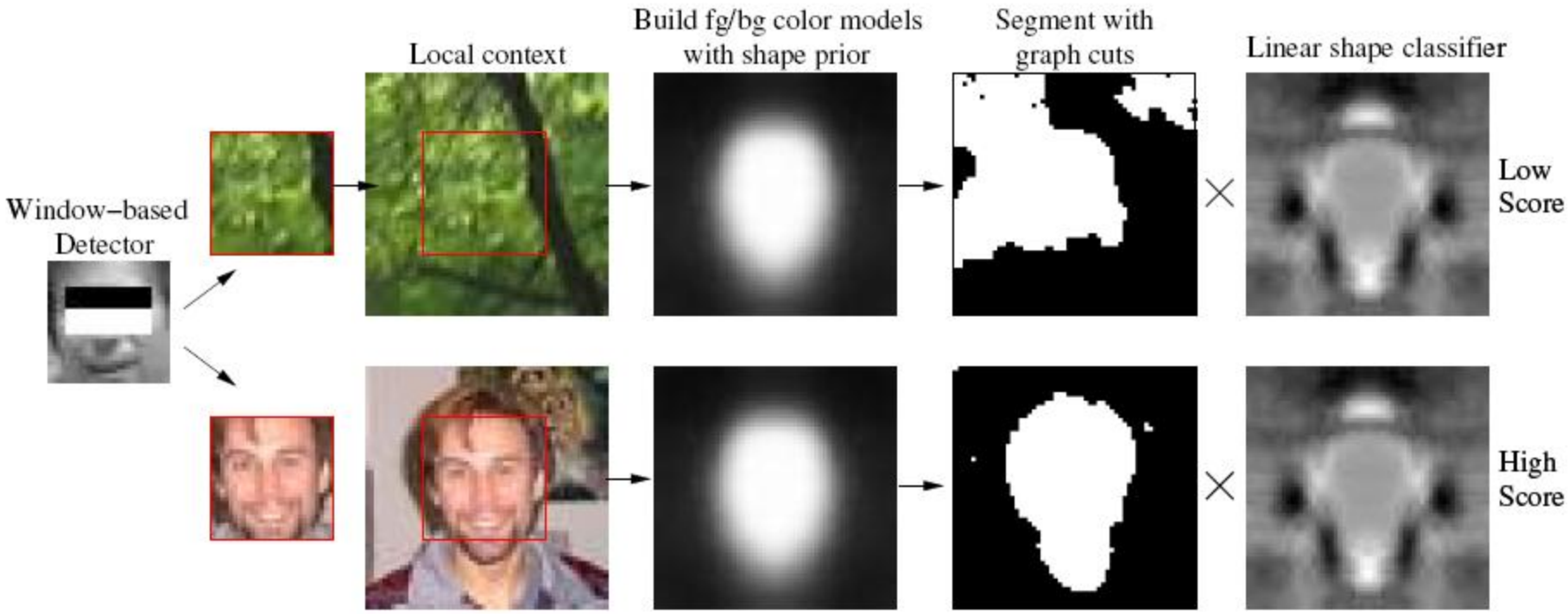
Segments as primitives for recognition?

Multiple segmentations



- B. Russell et al., "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections," CVPR 2006.

Object detection and segmentation

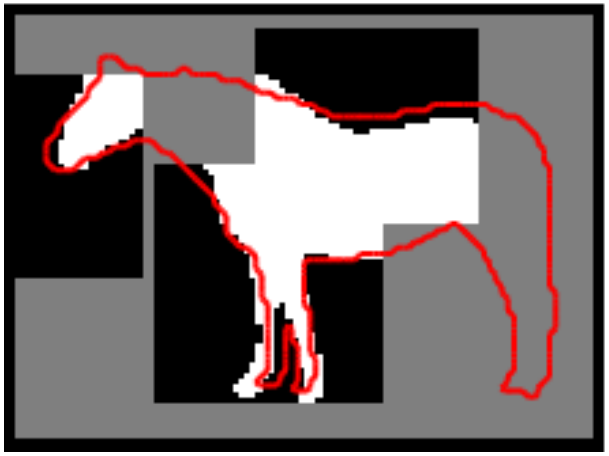


Segmentation energy:

$$E(L) = \sum_i -\log(P(l_i \mid class)) + \alpha \sum_{i,j \in N} \delta(l_i \neq l_j)$$

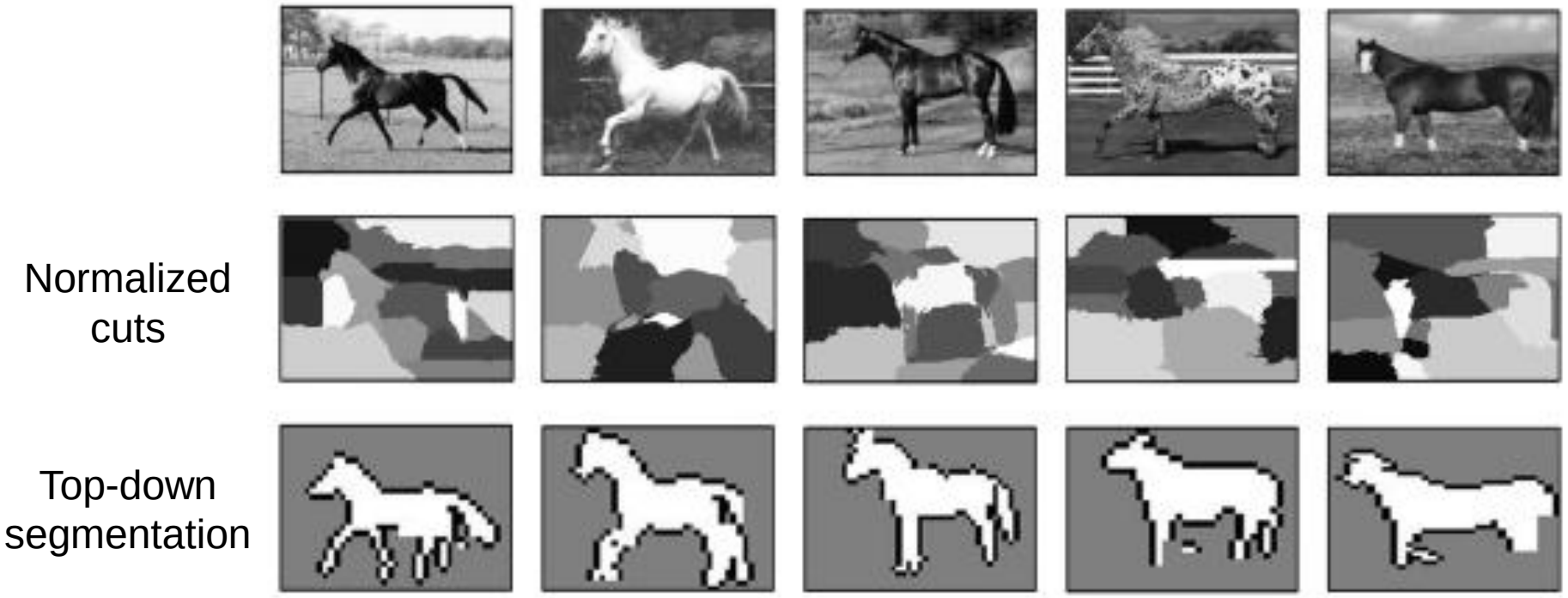
- D. Ramanan, "Using segmentation to verify object hypotheses," CVPR 2007.

Top-down segmentation



- E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002.
- A. Levin and Y. Weiss, "Learning to Combine Bottom-Up and Top-Down Segmentation," ECCV 2006.

Top-down segmentation (cont.)



- E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002
- A. Levin and Y. Weiss, "Learning to Combine Bottom-Up and Top-Down Segmentation," ECCV 2006.