CS 534: Computer Vision
Segmentation and Perceptual Grouping

Ahmed Elgammal
Dept of Computer Science
Rutgers University

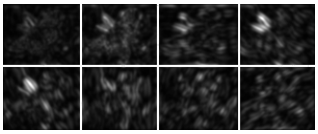
Outlines

- Mid-level vision
- What is segmentation
- Perceptual Grouping
- Segmentation by clustering
- Graph-based clustering
- Image segmentation using Normalized cuts

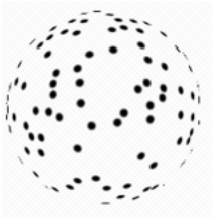
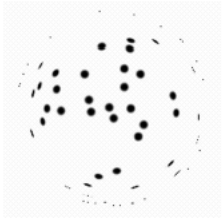
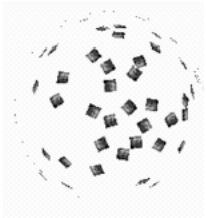
Mid-level vision

- Vision as an inference problem:
  - Some observation/measurements (images)
  - A model
  - Objective: what caused this measurement ?
- What distinguishes vision from other inference problems ?
  - A lot of data.
  - We don't know which of these data may be useful to solve the inference problem and which may not.
    - Which pixels are useful and which are not ?
    - Which edges are useful and which are not ?
    - Which texture features are useful and which are not ?





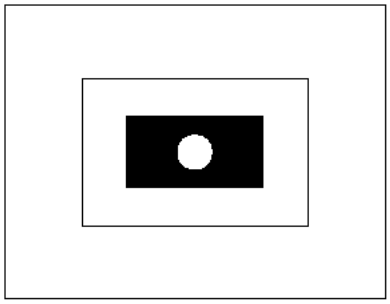
CS 534 – Segmentation - 3

Why do these tokens belong together?

It is difficult to tell whether a pixel (token) lies on a surface by simply looking at the pixel

CS 534 – Segmentation - 4

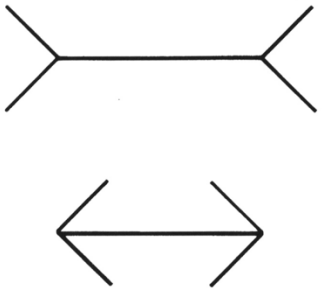


- One view of segmentation is that it determines which component of the image form the figure and which form the ground.
- What is the figure and the background in this image? Can be ambiguous.

Grouping and Gestalt

- Gestalt: German for form, whole, group
- Laws of Organization in Perceptual Forms (Gestalt school of psychology) Max Wertheimer 1912-1923

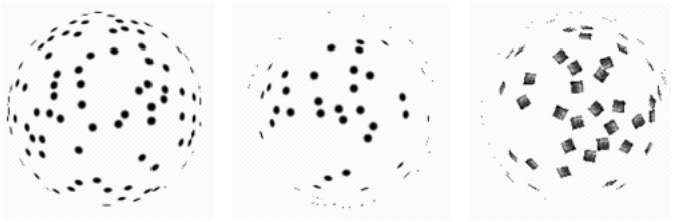
  "*there are contexts in which what is happening in the whole cannot be deduced from the characteristics of the separate pieces, but conversely; what happens to a part of the whole is, in clearcut cases, determined by the laws of the inner structure of its whole*"

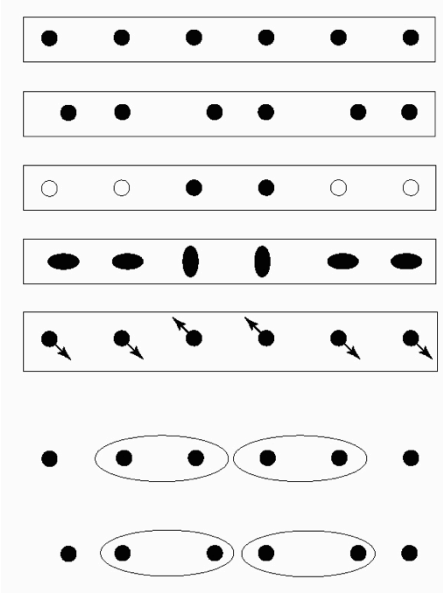


Muller-Layer effect:
This effect arises from some property of the relationships that form the whole rather than from the properties of each separate segment.

Grouping and Gestalt

- Can we write down a series of rules by which image elements would be associated together and interpreted as a group ?
- What are the factors that makes a set of elements to be grouped
- Human vision uses these factors in some way





Not Grouped

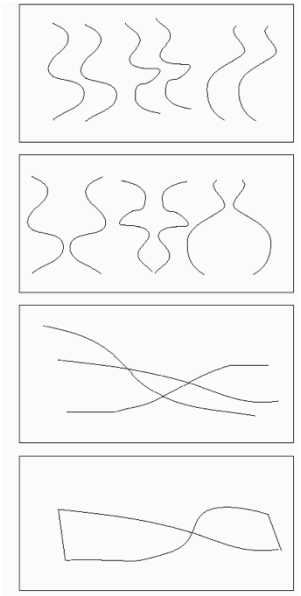
Proximity: Tokens that are nearby tend to be grouped together.

Similarity: Similar tokens tend to be grouped together.

Similarity

Common Fate: Tokens that have coherent motion tend to be grouped together

Common Region: Tokens that lie inside the same closed region tend to be grouped together

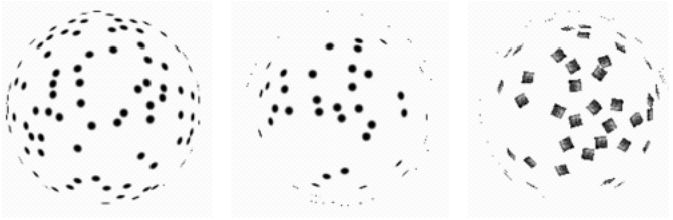


Parallelism: Parallel curves or tokens tend to be grouped together

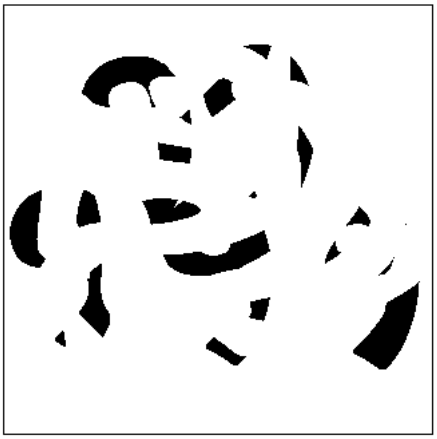
Symmetry: Curves that lead to symmetric groups are grouped together

Continuity: Tokens that lead to continuous curves tend to be grouped

Closure: Tokens or curves that tend to lead to closed curves tend to be grouped together.

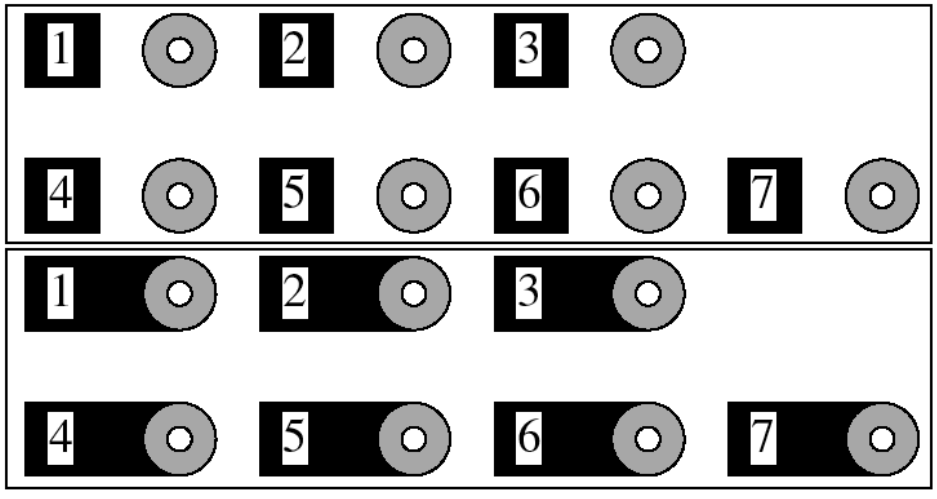


Familiar configuration: tokens that, when grouped, lead to a familiar object tend to be grouped



Occlusion appears to be a very important cue in grouping

Illusory contours: tokens are grouped together because they provide a cue to the presence of an occluding object

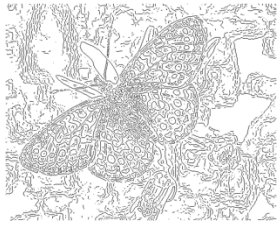


- These rules function as explanation only
- Very hard to form algorithms
- When one rule applied and when another ?

Segmentation

- Can we achieve a compact and suggestive representation of the interesting image data that emphasizes the properties that make it interesting
  - Segmentation
  - Grouping
  - Perceptual organization
  - Fitting
- What is interesting and what is not depends on the application

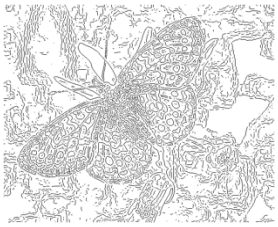




General ideas

- tokens
  - whatever we need to group (pixels, points, surface elements, etc., etc.)
- bottom up segmentation
  - tokens belong together because they are locally coherent
- top down segmentation
  - tokens belong together because they lie on the same object

- Grouping (or clustering)
  - collect together tokens that "belong together"
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?



Segmentation

Different problems – same problem: segmentation

- Summarizing a video: segment a video into shots, find coherent segments in the video, find key frames…
- Finding machine parts: finding lines, circles,…
- Finding people: find body segments, find human motion patterns
- Finding buildings from aerial imagery: find polygonal regions, line segments…
- Searching a collection of images: find coherent color, texture regions, shape…
- …

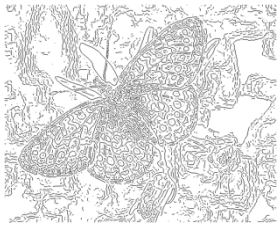
Segmentation

- Segmentation is a big topic

We will look into:

- Segmentation by clustering: Forming image segments:
  - How to decompose the image into "superpixels" image regions that are *coherent* in color and texture
  - Shape of the region is not that important while segmenting
- Segmentation by model fitting:
  - Fitting lines and curves to edge points:
  - Which points belong to which line, how many lines ?
  - What about more complicated models, e.g. fitting a deformable contour!





Segmentation as Clustering

- Objective: Which components of a data set naturally belong together
- This is a clustering problem which can be done in two ways:
- Partitioning – Decomposition:
  - Starting from a large data set how to partition it into pieces given some notion of association between data items
    - Decompose an image into regions that have coherent color and texture
    - Decompose a video sequence into shots
- Grouping
  - Collect sets of data item that make sense together given our notion of association
    - Collect together edge segments that seems to belong to a line
- Question: what is our notion of association ?


Image Segmentation as Clustering

- Cluster together (pixels, tokens, etc.) that belong together
- Pixels may belong together because they have the similar intensity, color, texture, and they are nearby.
- Framework:
  - (Representation) For each pixel extract a feature vector describing:
    - Intensity, color, texture (filter response)
    - Spatial location
  - (Clustering) Cluster the feature vectors
  - Replace each pixel by its cluster representation.



Image Segmentation as Clustering

- Progress:
- 1970s: Hierarchical Clustering
- 1980s: Markov Random Fields (MRF)
- 1990s:
  - Graph theoretic clustering – Graph cuts
  - Mean shift clustering (2000+)
- Recently: super-pixel / over-segmentation. Leave the final decision to a later process.

Evaluating Segmenters

- Collect "correct" segmentations
  - from human labellers
  - these may not be perfect, but ...
- Now apply your segmenter
  - Count
    - % real boundary points that were correctly marked -- recall
    - % correctly marked points out of all marked points-- precision

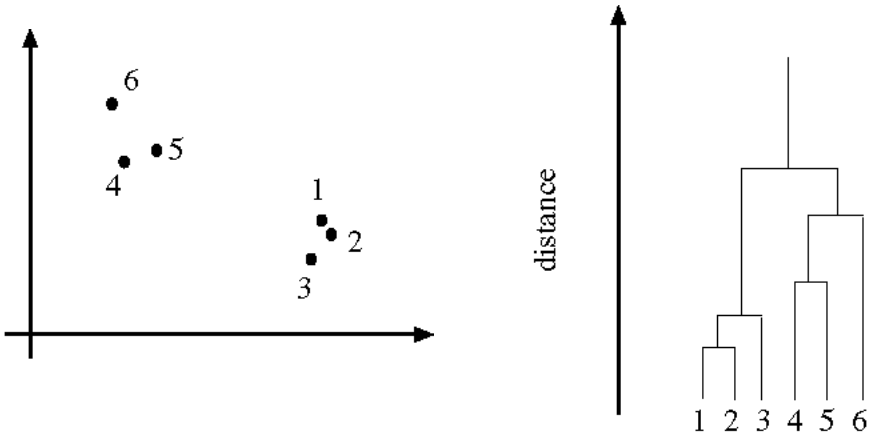
Segmentation as clustering

Hierarchical Clustering:

- Agglomerative clustering – clustering by merging – bottom-up
  - Each data point is assumed to be a cluster
  - Recursively merge clusters
  - Algorithm:
    - Make each point a separate cluster
    - Until the clustering is satisfactory
      - Merge the two clusters with the smallest *inter-cluster distance*
- Divisive clustering – clustering by splitting – top-down
  - The entire data set is regarded as a cluster
  - Recursively split clusters
  - Algorithm:
    - Construct a single cluster containing all points
    - Until the clustering is satisfactory
      - Split the cluster that yields the two components with the largest *inter-cluster distance*

Segmentation as clustering

- Two main issues:
- What is a good inter-cluster distance
  - single-link clustering: distance between the closest elements -> extended clusters
  - complete-link clustering: the maximum distance between elements –> rounded clusters
  - group-average clustering: Average distance between elements – rounded clusters
- How many clusters are there (model selection)
- Dendrograms
  - yield a picture of output as clustering process continues
- Issue: there is a lot of pixels, in reality hard to interpret the dendrogram, when to stop?



The watershed algorithm

- Think of an image (or gradient) as a topographic map

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.

- Work on image gradient or image intensity directly
- Tend to over-segment to small coherent regions (super-pixels)

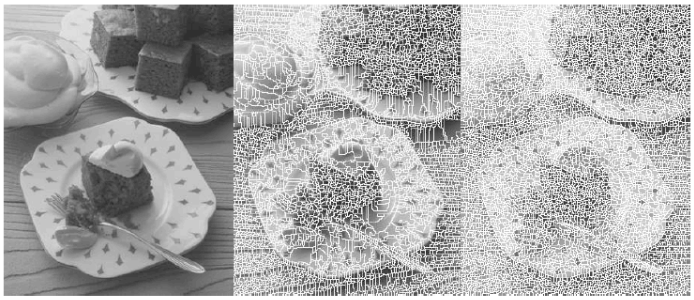


FIGURE 9.16: Segmentation results from the watershed algorithm, applied to an image by Martin Brigdale. **Center:** watershed applied to the image intensity; notice some long superpixels. **Right:** watershed applied to image gradient magnitude; this tends to produce rounder superpixels. *Martin Brigdale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

Clustering pixels using K-means

- represent pixels with
  - intensity vector; color vector; vector of nearby filter responses
  - perhaps position

Choose k data points to act as cluster centers
Until the cluster centers change very little
    Allocate each data point to cluster whose center is nearest.
    Now ensure that every cluster has at least
      one data point; one way to do this is by
      supplying empty clusters with a point chosen at random from
      points far from their cluster center.
    Replace the cluster centers with the mean of the elements
      in their clusters.
end

Algorithm 6.3: Clustering by K-Means.

K-Means

- Choose a fixed number of clusters K
- Each cluster has a center (mean) μ_i
- Choose
  - cluster centers and
  - point-cluster allocations

  to minimize error
- can't do this by search, because there are too many possible allocations.
- Algorithm:

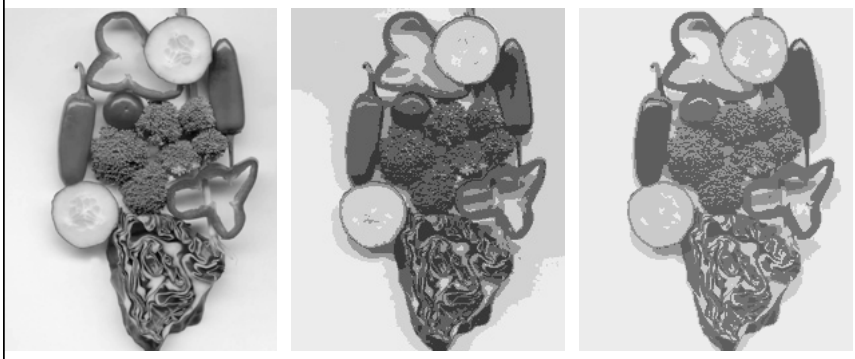
  Repeat until centers are unchanged:
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

Given $X = \{x_j\}$ Find a partition $S = \{S_1,...,S_K\}$

$$\arg\min_{S} \sum_{i=1}^{K} \sum_{x_j \in S_i} (x_j - \mu_i)$$

CS 534 – Segmentation - 30

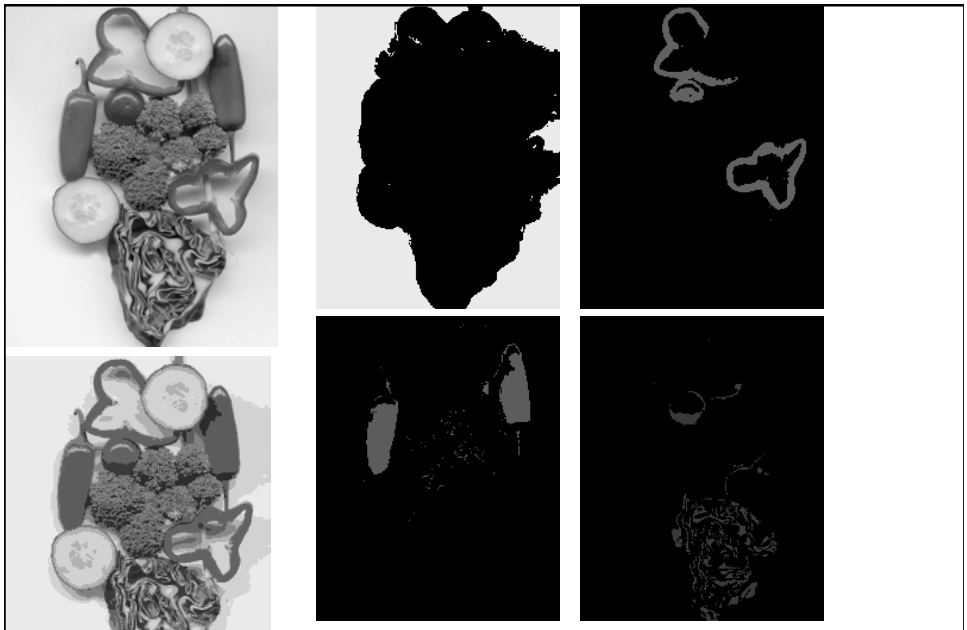
Image | Clusters on intensity | Clusters on color



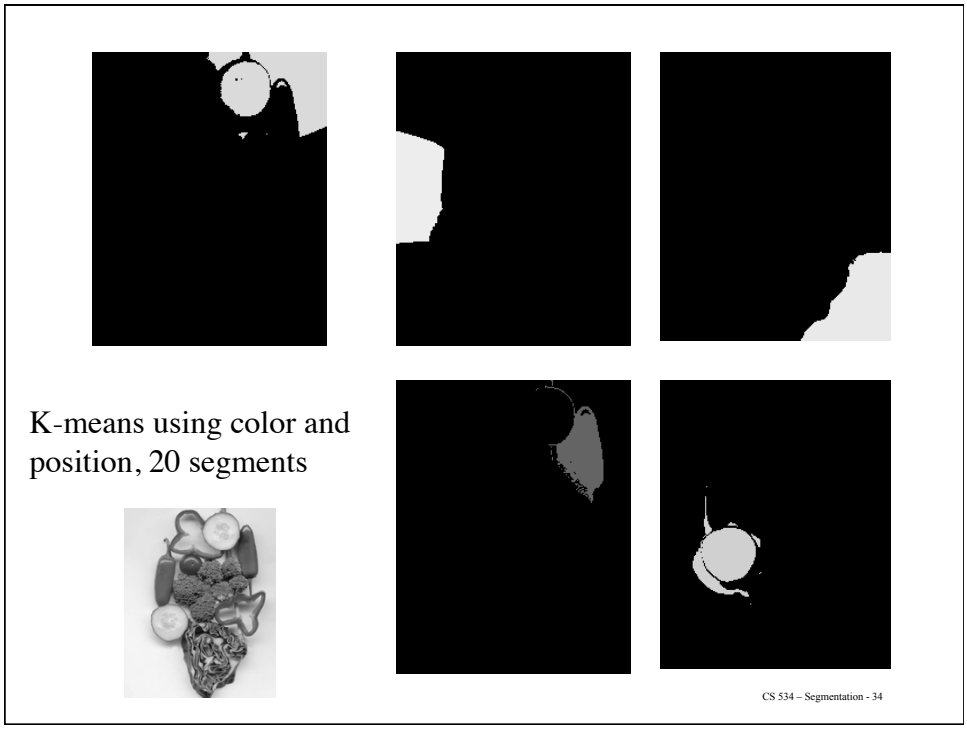
K-means clustering using intensity alone and color alone
K=5 segmented image is labeled with cluster means

Image | Clusters on color

K-means using color alone, 11 segments



K-means using color alone, 11 segments.



K-means using color and position, 20 segments

Graph-based Image Segmentation

- Represents image as a graph
- A vertex for each pixel
- Edges between pixels
- Weights on edges reflect
  - Dissimilarities (distance) (big weight = different, small weight = similar)
  - Similarity (affinities) (big weight = similar; small weight=different)
- Compute distances (or affinities) based on:
  - Brightness
  - Color
  - Texture
  - Distance
  - …
- Connectivity:
  - Fully connected: edges between every pair of pixels
  - Partially connected: edges between neighboring pixels

Measuring Affinity

Intensity

$$af f(x,y) \quad \exp \quad \frac{1}{2} \frac{2}{i} \left(\|I(x) \quad I(y)\|^2 \right)$$

Distance

$$af f(x,y) \quad \exp \quad \frac{1}{2} \frac{2}{d} \left(\|x \quad y\|^2 \right)$$

color

$$af f(x,y) \quad \exp \quad \frac{1}{2} \frac{2}{t} \left(\|c(x) \quad c(y)\|^2 \right)$$

Agglomerative strategy
grow clusters edge by edge

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.

$$diff(C_1, C_2) = \min_{v1 \in C_1, v2 \in C_2, (v1,v2) \in E} w(v1, v2)$$

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

M(C): minimum spanning tree of C

$$\text{MInt}(C_1, C_2) = \min(\text{int}(C_1), \text{int}(C_2))$$

Felzenszwalb and Huttenlocher 2004

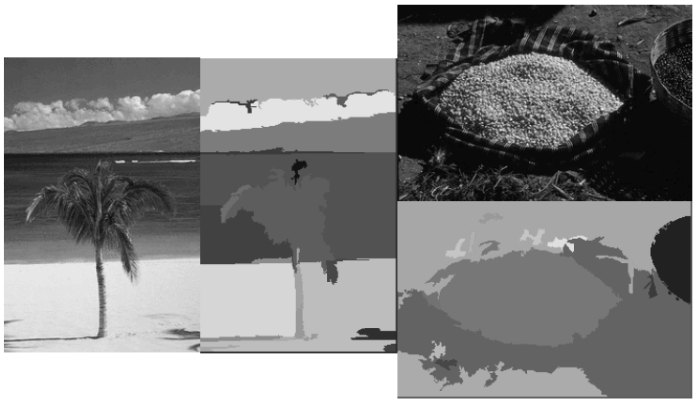


FIGURE 9.22: Images segmented using Algorithm 9.8, shown next to segments. Figures obtained from `http://people.cs.uchicago.edu/~pff/segment/`, by kind permission of Pedro Felzenszwalb.

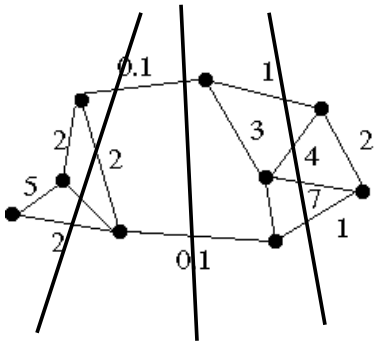
http://cs.brown.edu/~pff/segment/

Graph Cut

What is a Graph Cut:

- We have undirected, weighted graph $G=(V,E)$
- Remove a subset of edges to partition the graph into two disjoint sets of vertices A,B (two sub graphs):

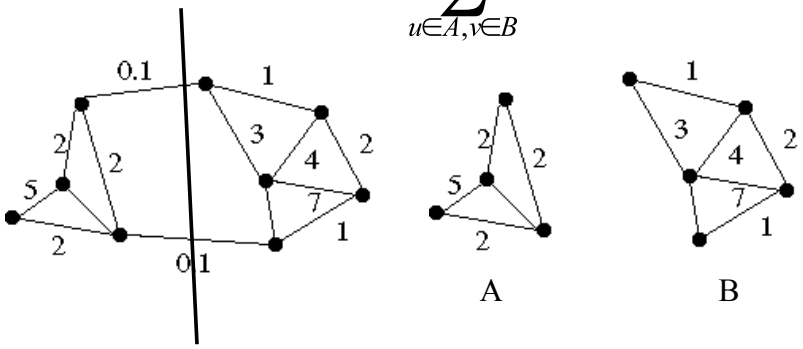
$A \cup B = V, A \cap B = \Phi$



Graph Cut

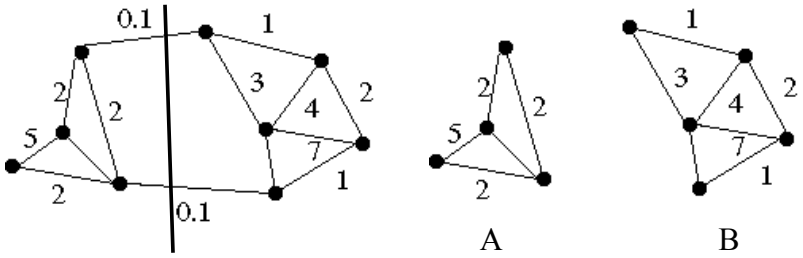
- Each cut corresponds to some cost (cut): sum of the weights for the edges that have been removed.

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$



Graph Cut

- In many applications it is desired to find the cut with minimum cost: *minimum cut*
- Well studied problem in graph theory, with many applications
- There exists efficient algorithms for finding minimum cuts

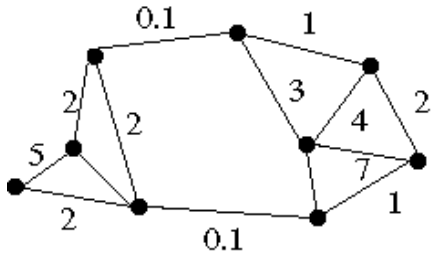


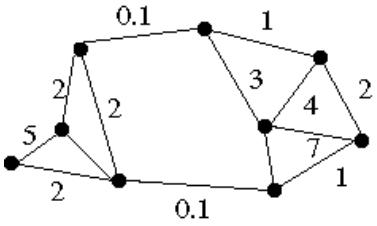
$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

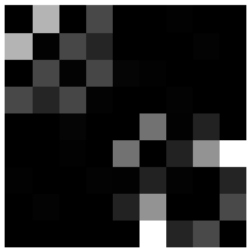
Graph theoretic clustering

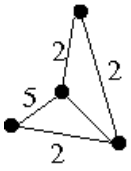
- Represent tokens using a weighted graph
  - Weights reflects similarity between tokens
  - *affinity* matrix
- Cut up this graph to get subgraphs such that:
  - Similarity within sets maximum.
  - Similarity between sets minimum.

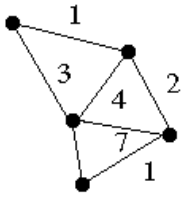
⇒ Minimum cut









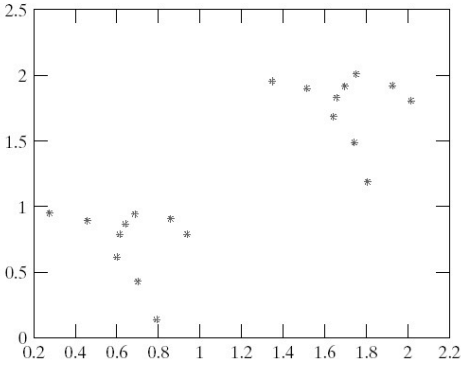


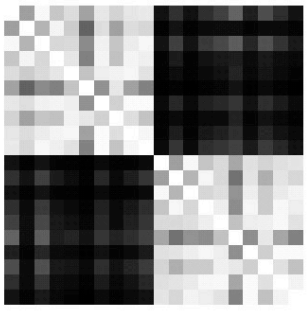
CS 534 – Segmentation II - 43

- Use exponential function for edge weights

$$w(x) = e^{-(d(x)/\sigma)^2}$$

d(x) : feature distance

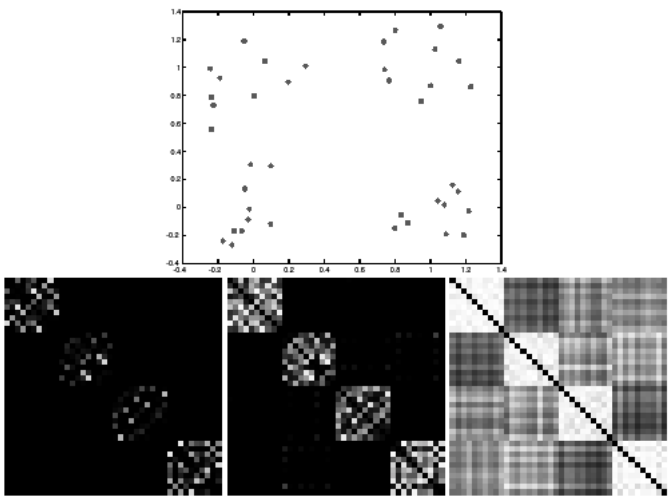




CS 534 – Segmentation II - 44

Scale affects affinity

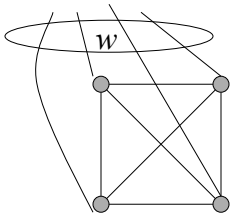
$$w(x) = e^{-(d(x)/\sigma)^2}$$



σ=0.1 σ=0.2 σ=1

Spectral clustering

- Simplest idea: we want a vector w giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize



$$w_n^T A w_n$$

Sum of

Association of element i with cluster n ×

Affinity between i and j ×

Association of element j with cluster n

Eigenvectors and clustering

- We could maximize $w_n^T A w_n$
- But need the constraint $w_n^T w_n = 1$
- Using Lagrange multiplier λ

$$w_n^T A w_n + \lambda(w_n^T w_n - 1)$$

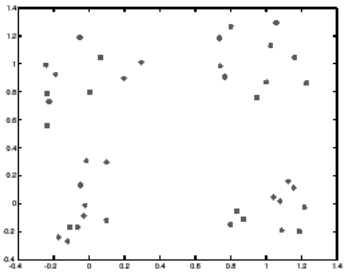
- Differentiation

$$A w_n = \lambda w_n$$

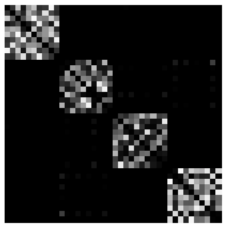
- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

CS 534 – Segmentation II - 47

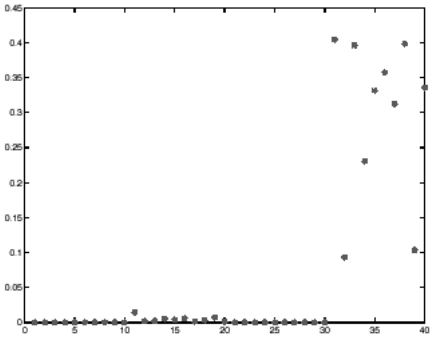
Example eigenvector



points



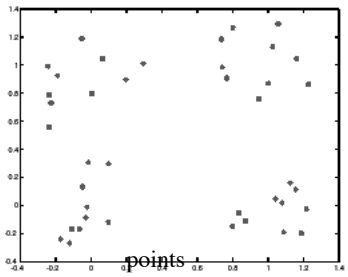
matrix



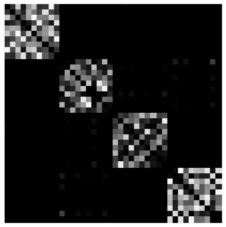
Eigenvector corresponding to the largest eigenvalue

CS 534 – Segmentation II - 48

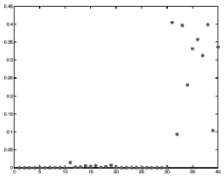
Example eigenvector



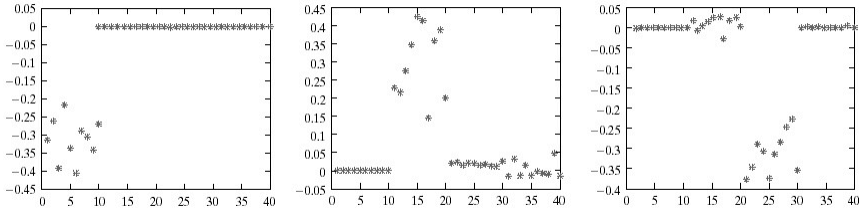
points



matrix

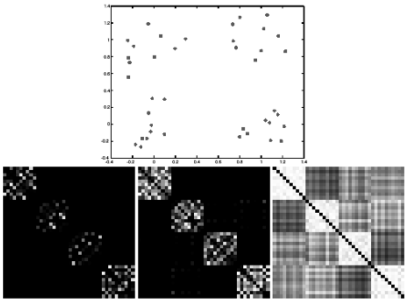


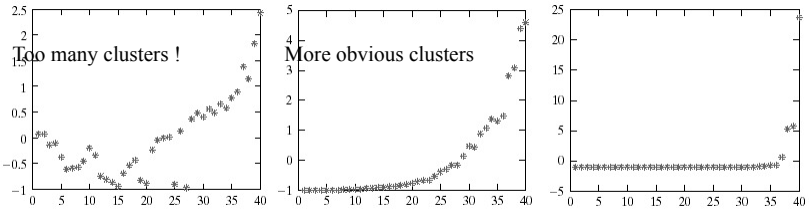
First eigenvectors



The three eigenvectors corresponding to the next three eigenvalues of the affinity matrix

CS 534 – Segmentation II - 49





Too many clusters !

More obvious clusters

eigenvalues for three different scales for the affinity matrix

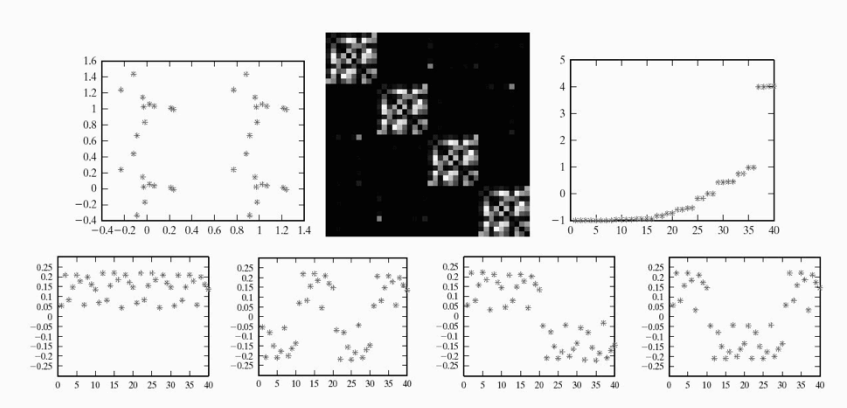
CS 534 – Segmentation II - 50

More than two segments

- Two options
  - Recursively split each side to get a tree, continuing till the eigenvalues are too small
  - Use the other eigenvectors

Algorithm

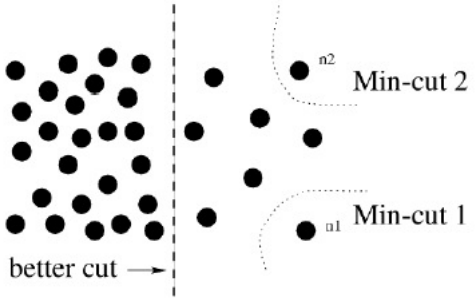
- Construct an Affinity matrix A
- Computer the eigenvalues and eigenvectors of A
- Until there are sufficient clusters
  - Take the eigenvector corresponding to the largest unprocessed eigenvalue; zero all components for elements already clustered, and threshold the remaining components to determine which element belongs to this cluster, (you can choose a threshold by clustering the components, or use a fixed threshold.)
  - If all elements are accounted for, there are sufficient clusters



We can end up with eigenvectors that do not split clusters because any linear combination of eigenvectors with the same eigenvalue is also an eigenvector.

Normalized Cuts

- Min cut is not always the best cut



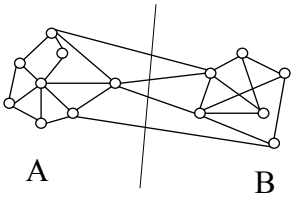
CS 534 – Segmentation II - 53

- Association between two sets of vertices: total connection between the two sets.

$$assoc(A,B) = \sum_{u \in A, t \in B} w(u,t)$$

$$assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$

$$assoc(B,V) = \sum_{u \in B, t \in V} w(u,t)$$

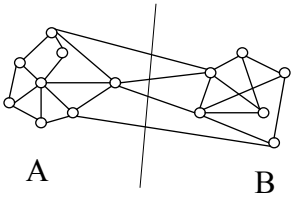


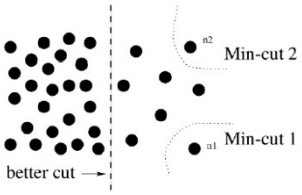
CS 534 – Segmentation II - 55

- Normalize the cuts: compute the cut cost as a fraction of the total edge connections to all nodes in the graph

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

- Disassociation measure. The smaller the better.



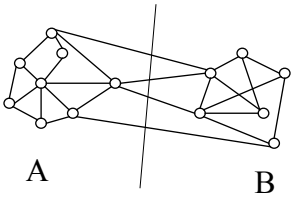


CS 534 – Segmentation II - 56

- Association measure: Normalized association within groups:

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

- This is a within group association measure: the bigger the better



CS 534 – Segmentation II - 57

$$cut(A,B) = assoc(A,B) = assoc(A,V) - assoc(A,A)$$

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ &= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} \\ &\quad + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\ &= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\right) \\ &= 2 - Nassoc(A,B). \end{aligned}$$

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

Total association (similarity) within groups, the bigger the better

CS 534 – Segmentation II - 58

- By looking for a cut that minimizes *Ncut(A,B),*
  - Minimize the disassociation between the groups,
  - Maximize the association within group

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ &= 2 - Nassoc(A,B) \end{aligned}$$

- Minimizing a normalized cut is NP-complete

Given a partition of nodes of a graph, V, into two sets A and B, let $\boldsymbol{x}$ be an $N = |V|$ dimensional indicator vector, $x_i = 1$ if node i is in A and -1, otherwise. Let $\boldsymbol{d}(i) = \sum_j w(i,j)$ be the total connection from node i to all other nodes. With the definitions $\boldsymbol{x}$ and $\boldsymbol{d}$, we can rewrite $Ncut(A,B)$ as:

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)} \\ &= \frac{\sum_{(\boldsymbol{x}_i>0,\boldsymbol{x}_j<0)} -w_{ij}\boldsymbol{x}_i\boldsymbol{x}_j}{\sum_{\boldsymbol{x}_i>0}\boldsymbol{d}_i} \\ &\quad + \frac{\sum_{(\boldsymbol{x}_i<0,\boldsymbol{x}_j>0)} -w_{ij}\boldsymbol{x}_i\boldsymbol{x}_j}{\sum_{\boldsymbol{x}_i<0}\boldsymbol{d}_i}. \end{aligned}$$

CS 534 – Segmentation II - 59

Normalized cuts

- W: cost matrix: w(i,j)
- D: sum of the costs for every vertex

$$D(i,i) = \sum_j W(i,j), \qquad D(i,j) = 0 \quad i \neq j$$

- Optimal Normalized cut can be found be solving for y that minimizes

$$\min_y \frac{y^T(D-W)y}{y^T D y} \qquad y \in \{1, -b\} \qquad y^T D 1 = 0$$

- NP-complete problem,
- approximate real-valued solution by solving a generalized eigenvalue problem

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

- Real-valued solution is the second smallest eigenvector
- look for a quantization threshold that maximizes the criterion --- i.e all components of y above that threshold go to one, all below go to -b

Example - brightness

$$w_{ij} = e^{\frac{-\|F_{(i)} - F_{(j)}\|_2^2}{\sigma_I^2}} * \begin{cases} e^{\frac{-\|X_{(i)} - X_{(j)}\|_2^2}{\sigma_X^2}} & \text{if } \|X(i) - X(j)\|_2 < r \\ 0 & \text{otherwise.} \end{cases}$$

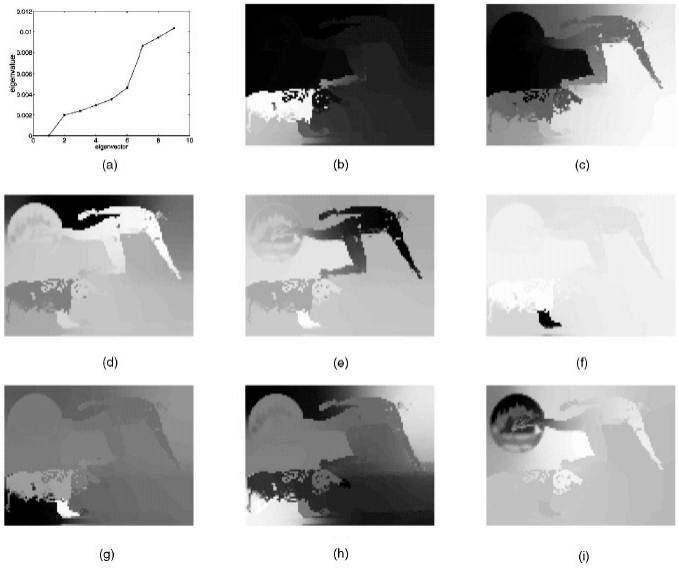


Figure from "Normalized cuts and image segmentation," Shi and Malik, copyright IEEE, 2000

Segmentation using Normalized cuts

Two algorithms:

- Recursive two-way Ncut
  - Use the second smallest eigenvector to obtain a partition to two segments.
  - Recursively apply the algorithm to each partition.
- Simultaneous K-way cut with multiple eigenvectors.
  - Use multiple (n) smallest eigenvectors as n dimensional class indicator for each pixel and apply simple clustering as k-means to obtain n clusters.

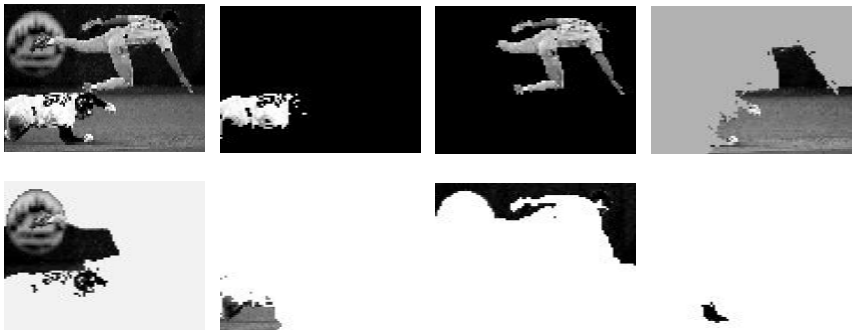


Figure from "Normalized cuts and image segmentation," Shi and Malik, copyright IEEE, 2000

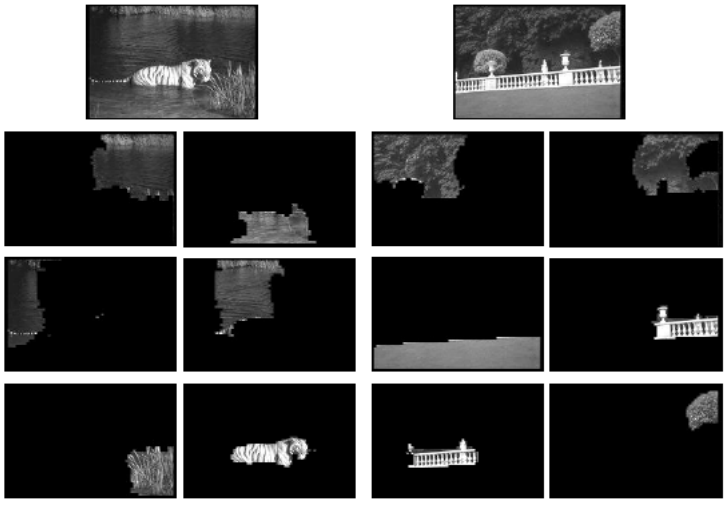
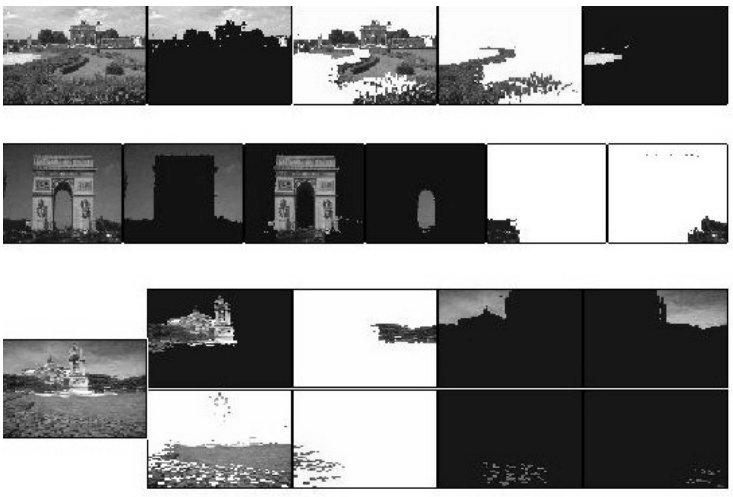


Figure from "Image and video segmentation: the normalised cut framework", by Shi and Malik, copyright IEEE, 1998

Figure from "Normalized cuts and image segmentation," Shi and Malik, copyright IEEE, 2000



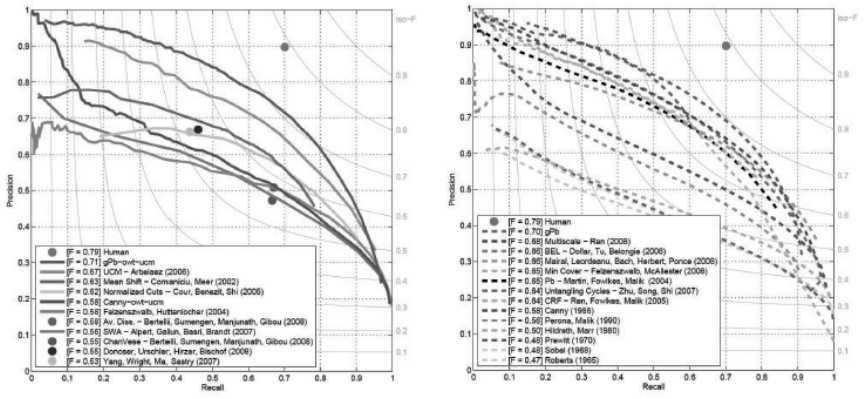


FIGURE 9.25: Segmenters and edge detectors can be evaluated by comparing the predicted boundaries to object boundaries that people mark on images. A natural comparison involves precision (the percentage of the marked boundary points that are real ones) and recall (the percentage of real boundary points that were marked); the F measure summarizes precision and recall into a single number $F = 2PR/(P + R)$. On the **left**, these measures for various segmenters; on the **right**, for various edge detectors. *This figure was originally published as Figures 1 and 2 of "Contour Detection and Hierarchical Image Segmentation" by P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, © IEEE, 2011.*

Interactive segmentation

- Goals
  - User cuts an object out of one image to paste into another
  - User forms a matte
    - weights between 0 and 1 to mix pixels with background
    - to cope with, say, hair
- 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

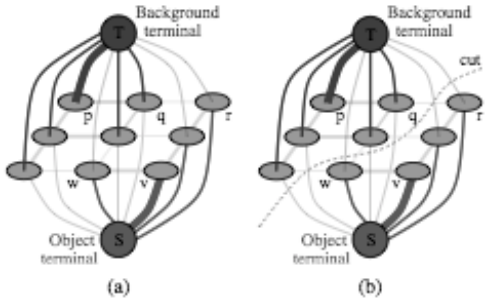


Figure 5.23 Graph cuts for region segmentation (Boykov and Jolly 2001) © 2001 IEEE: (a) the energy function is encoded as a maximum flow problem; (b) the minimum cut determines the region boundary.

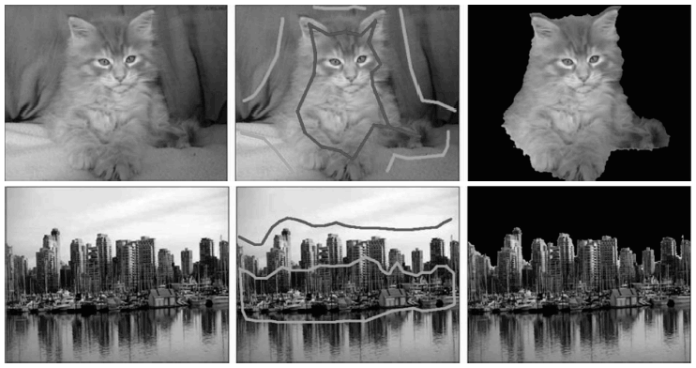


FIGURE 9.11: A user who wants to cut an object out of an image (**left**) could mark some foreground pixels and some background pixels (**center**), then use an interactive segmentation method to get the cut out components on the **right**. The method produces a model of foreground and background pixel 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.*

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.*

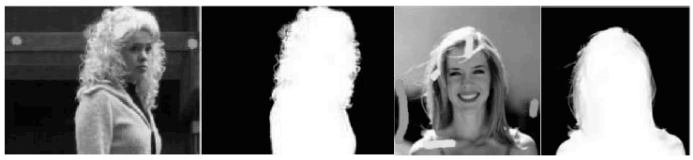


FIGURE 9.13: Matting methods produce a real-valued mask (rather than a foreground-background mask) to try and compensate for effects in hair, at occluding boundaries, and so on, where some pixels consist of an average of foreground and background values. The matte is bright for foreground pixels and 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.*

Superpixels

- Pixels are too small and too detailed a representation
  - for recognition
  - for some kinds of reconstruction
- Replace with superpixels
  - small groups of pixels that are
    - clumpy
    - like one another
    - a reasonable representation of the underlying pixels

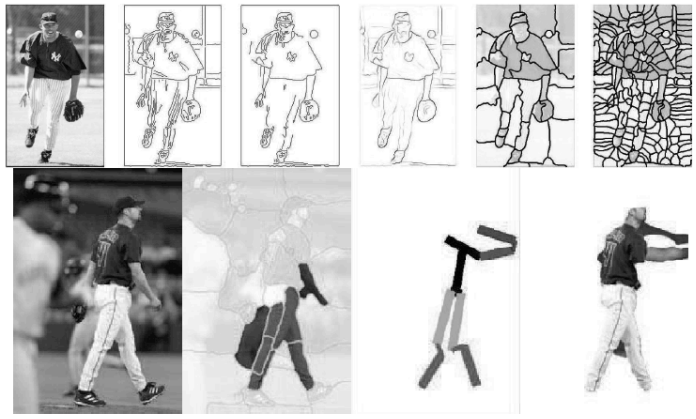


FIGURE 9.14: Superpixels often can expose structure in images that other representations conceal. Human body segments tend to appear as long, thin segments. In the **top** row, an image together with three different edge maps (the edge detector of Section 5.2.1, with two scales of smoothing, and the 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.*

Sources

- Forsyth and Ponce, Computer Vision a Modern approach: chapter 14.
- Slides by
  - D. Forsyth @ Berkeley
- Jianbo Shi and Jitendra Malik *"Normalized Cuts and Image Segmetnation"* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22 No. 0, August 2000