Segmentation

Bottom up Segmentation
Semantic Segmentation

Bottom Up Segmentation

Segmentation as clustering

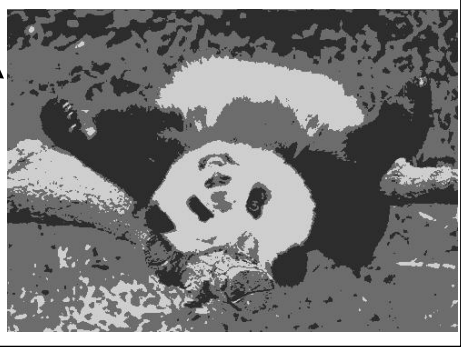
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

K=2

K=3

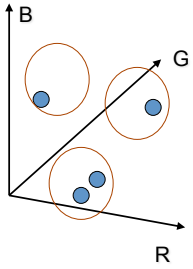


quantization of the feature space; segmentation label map

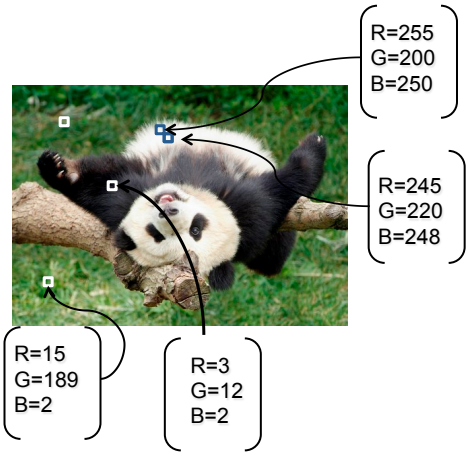
Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



Feature space: color value (3-d)



Kristen Grauman

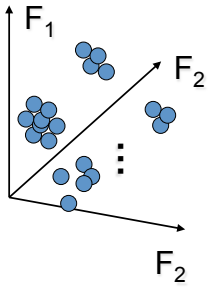
Segmentation as clustering

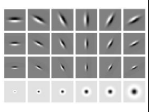
- Color, brightness, position alone are not enough to distinguish all regions…

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **texture** similarity





Filter bank of 24 filters

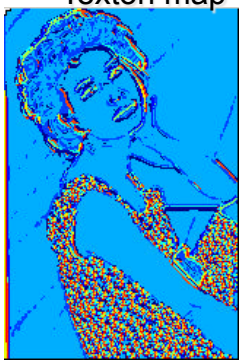
Feature space: filter bank responses (e.g., 24-d)

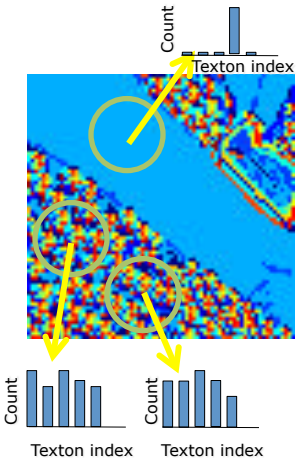
Segmentation with texture features

- Find "textons" by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*

Image

Texton map





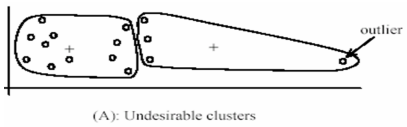
Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

K-means: pros and cons

Pros

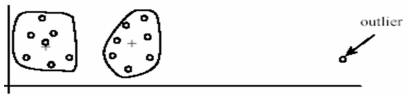
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error



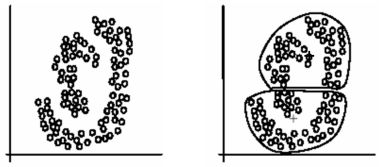
(A): Undesirable clusters

Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



(B): Ideal clusters

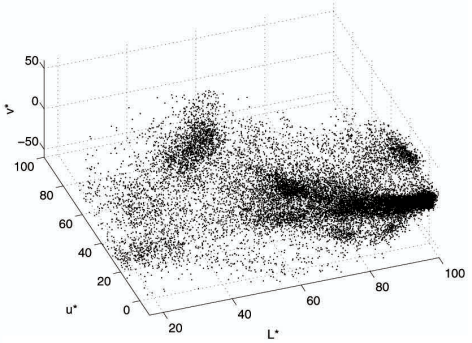

(A): Two natural clusters (B): *k*-means clusters

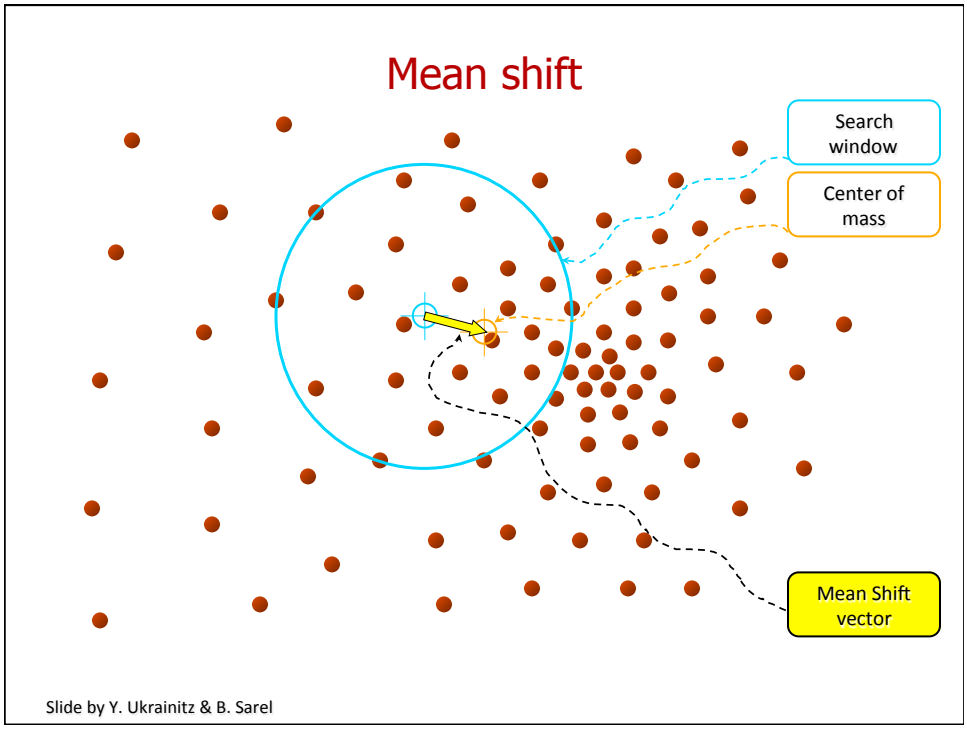
Mean shift algorithm

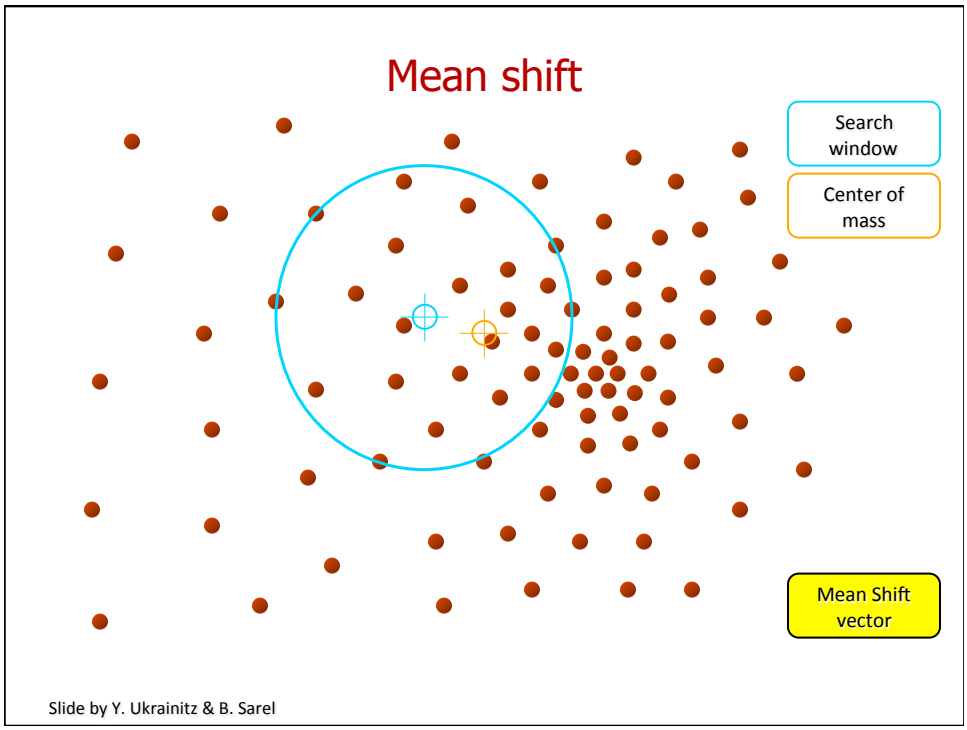
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

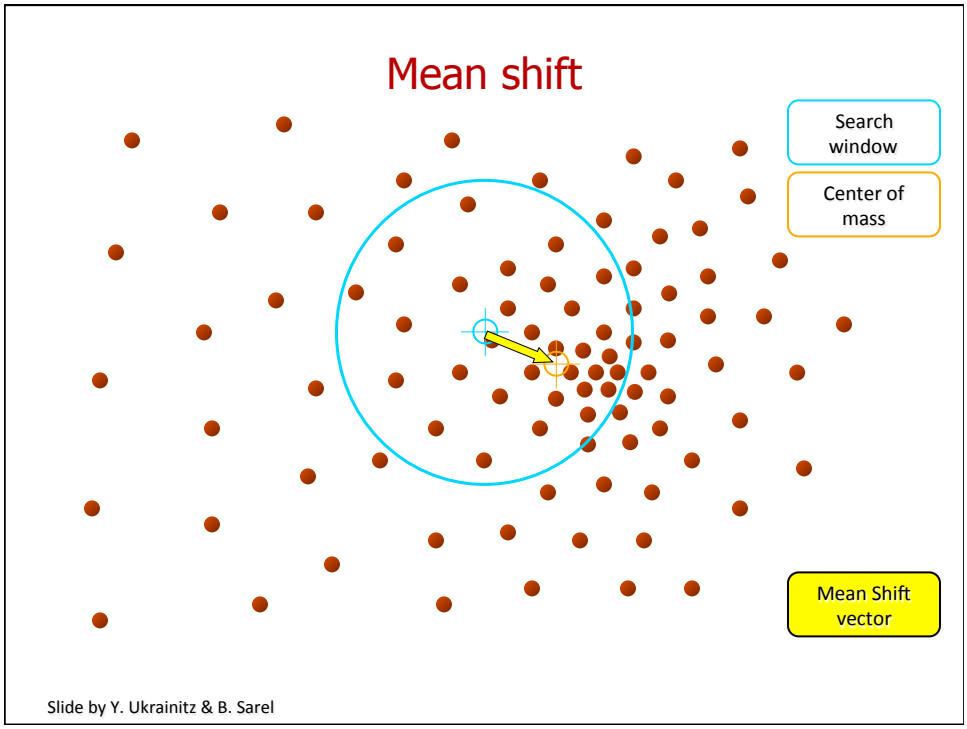
image

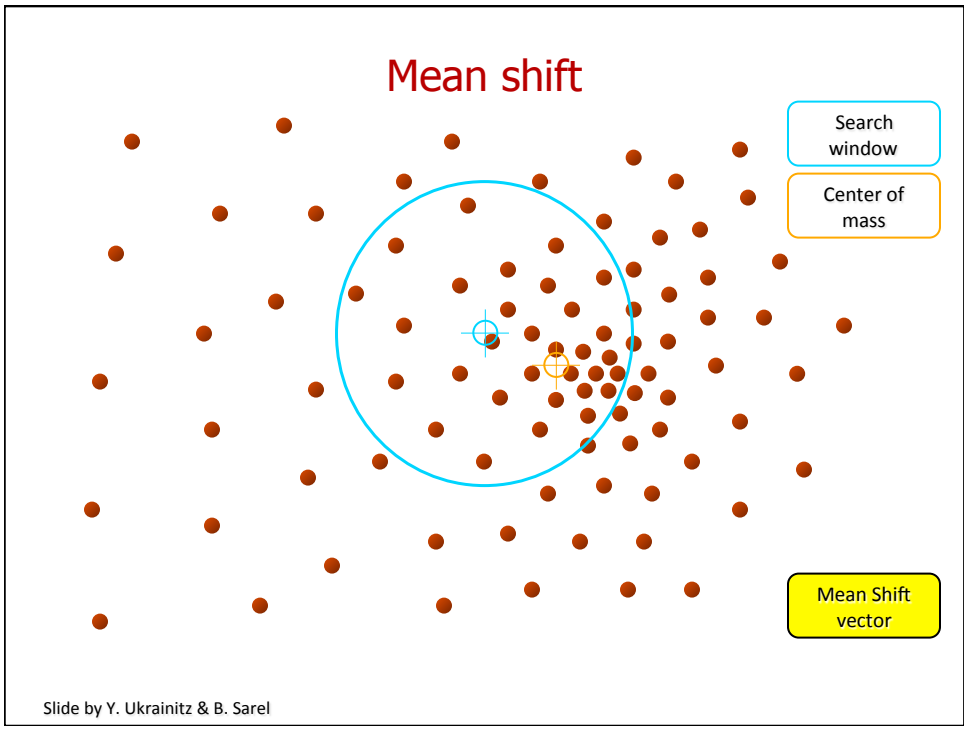
Feature space (L*u*v* color values)

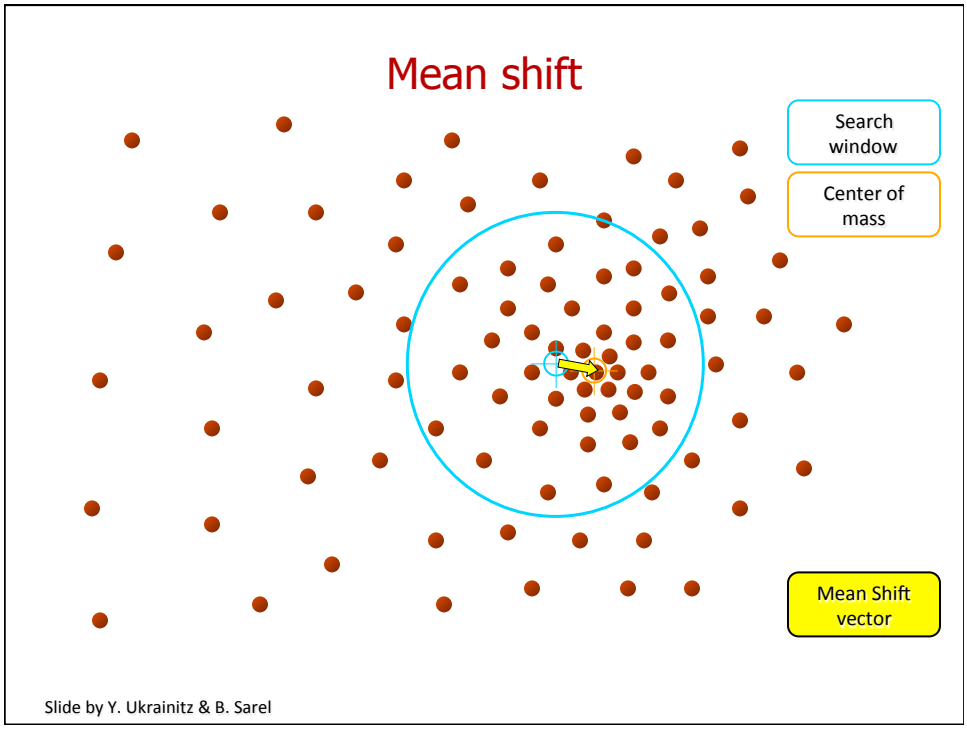


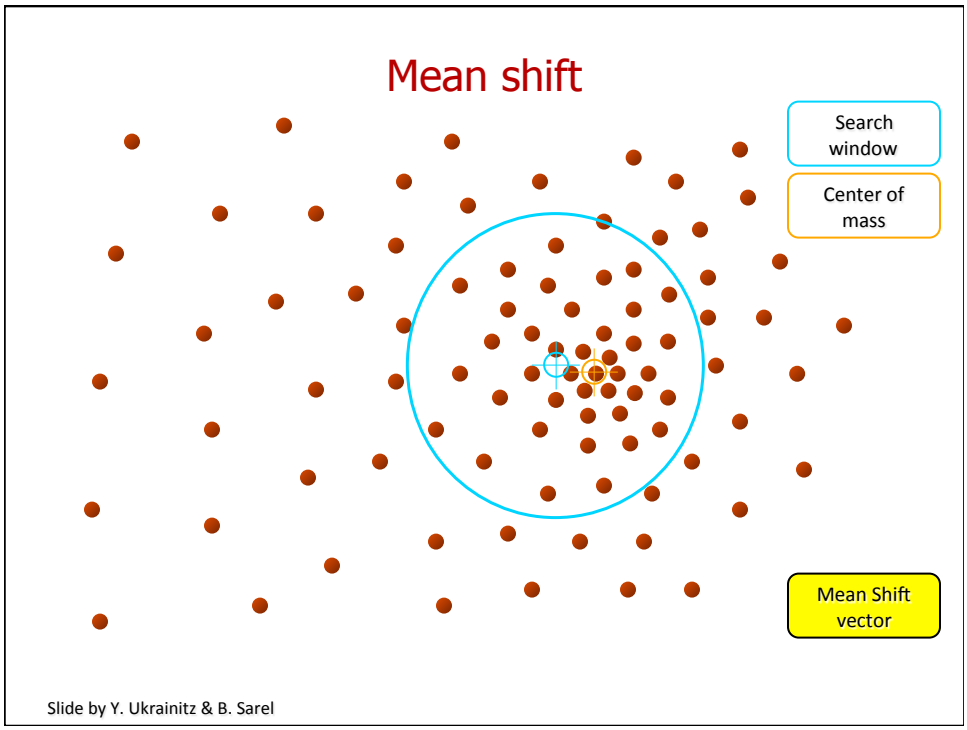

Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel


Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel

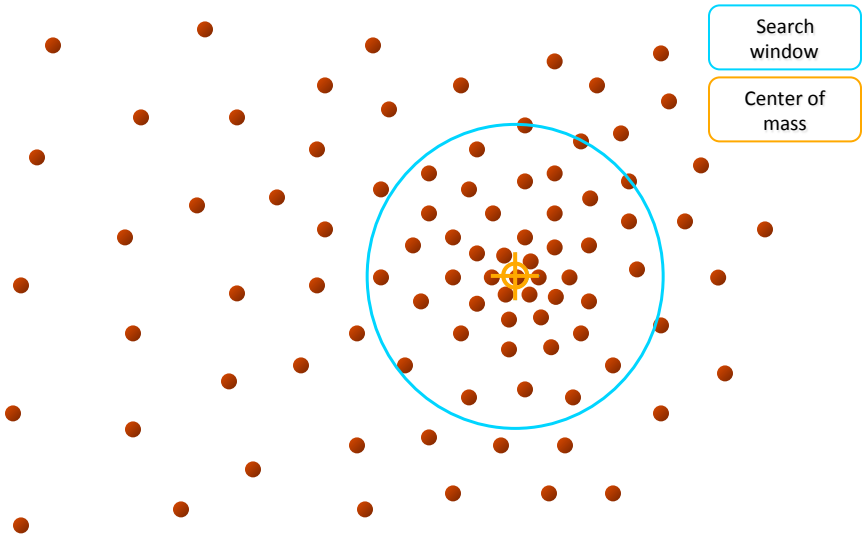

Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel


Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel


Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel


Mean shift
Search window
Center of mass
Mean Shift vector
Slide by Y. Ukrainitz & B. Sarel

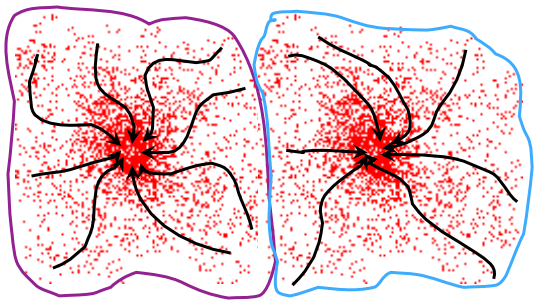
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

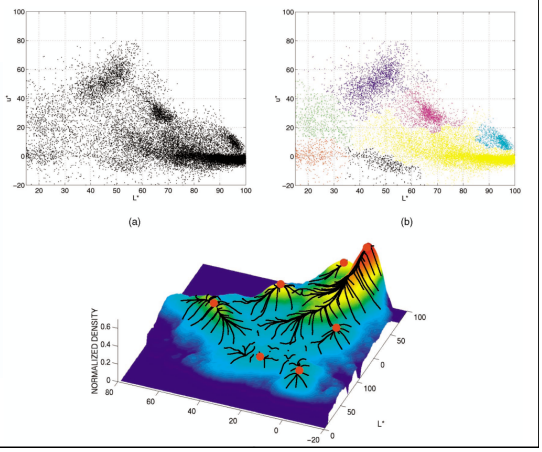
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



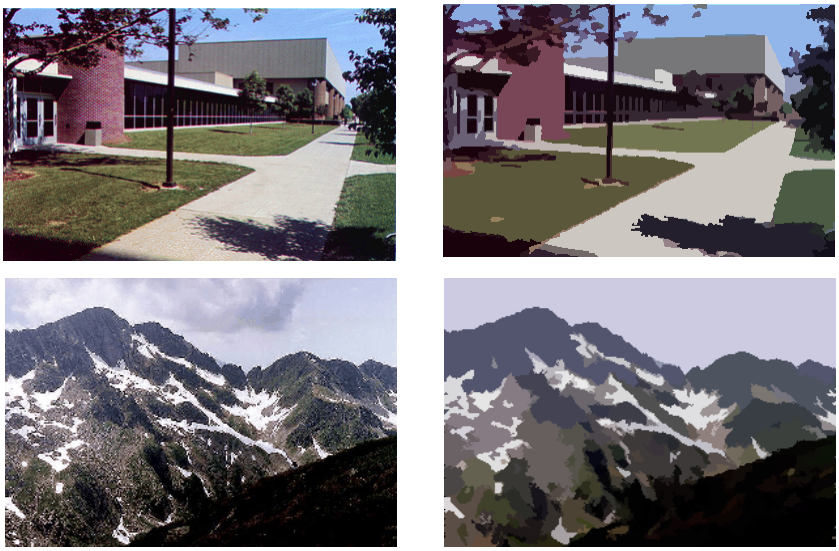
Slide by Y. Ukrainitz & B. Sarel

Mean shift 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 segmentation results



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

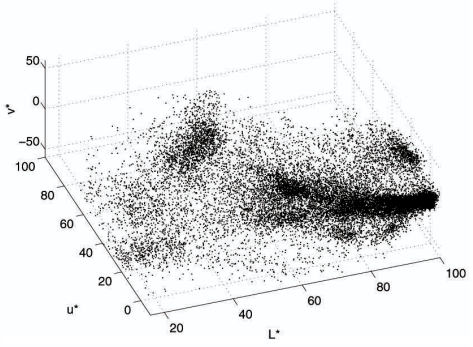
Mean-Shift Segmentation Results

Mean shift algorithm

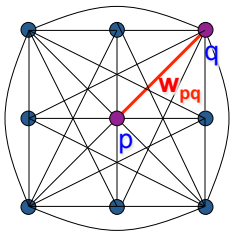
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

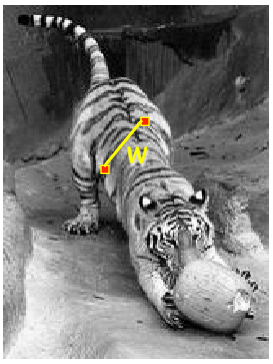
image

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



Images as graphs





- *Fully-connected* graph
  - node (vertex) for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - affinity weight $\mathbf{w_{pq}}$ for each link (edge)
    - $\mathbf{w_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference (in color and position…)

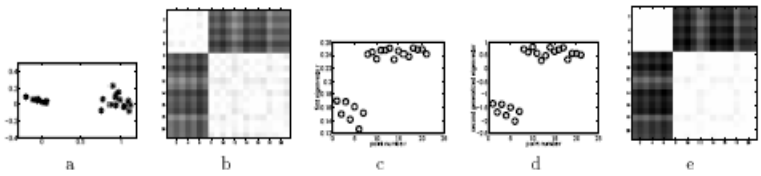
Source: Steve Seitz

Segmentation as Graph Partitioning

- (Shi & Malik ‘’ 97) Normalized Cut Algorithm
- Idea – each pixel in the image is a node in the graph
- Arcs represent similarities between adjacent pixels
- Graph is fully connected
- Goal – partition the graph into a sets of vertices (regions), such that the similarity within the region is high – and similarity across the regions is low.

Segmentation

- Toy example
- Bright entries in the affinity matrix high
- Likely to belong together

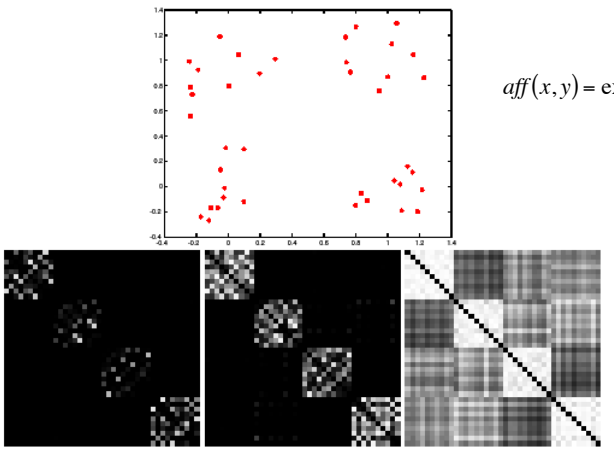


- one possible affinity based on distance

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$

Scale affects affinity

Depending on the scale the blocks are more (middle)
Or less obvious (left and right)



$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$

$$\sigma_d = \{0.1, 0.2, 1\}$$

Measuring Affinity

Intensity

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_i^2\right)\left(\|I(x) - I(y)\|^2\right)\right\}$$

Distance

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x - y\|^2\right)\right\}$$

Texture

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_t^2\right)\left(\|c(x) - c(y)\|^2\right)\right\}$$

Eigenvectors and segments

- Consider A – affinity matrix
- Simplest idea: we want a vector 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
- To get single Good Cluster - maximize

$$a^T A a$$

- Where a is an vector linking elements to n-th cluster

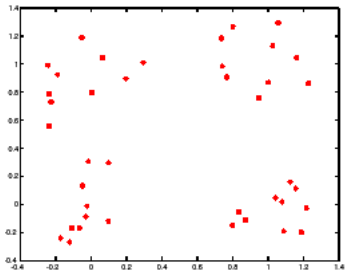
(i-th entry associates element with the cluster)

- In order to prevent arbitrary large association weights – impose constraint

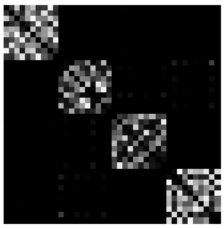
$$a^T a = 1$$

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

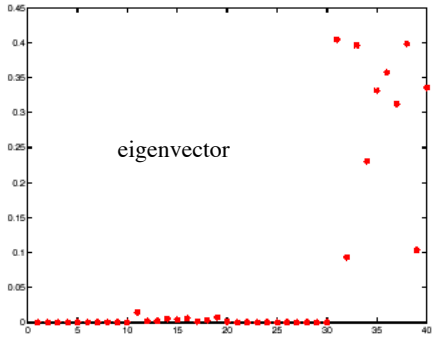
Example eigenvector



points



matrix



- Entries associated with the largest cluster are high
- Searching for additional clusters – look at
- Eigenvectors associated with second largest Eigenvalue etc

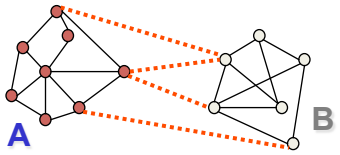
More than two segments

- Reasoning about other eigenvectors - consider that affinity matrix is block diagonal.
- Until there are sufficient clusters pick eigenvector associated with the largest eigenvalue, zero the elements which were clustered, threshold elements with large association weights - those will form a new cluster
- Keep going until there is sufficient number of clusters and all elements have been accounted for
- Spectral Clustering Techniques (A. Ng and M. Jordan)
- Problems - if the eigenvalues are similar - eigenvectors do not reveal the clusters
- Normalized cut - graph cut - alternative optimization criterion J. Shi and J. Malik – derivation on the board

Normalized Cuts

- Previous method – we can get eigenvectors which do not split clusters – e.g. in cases when there is no dominant eigenvalue
- Alternative approach find the best cut of the graph into two connected components, such that the cost of the cut is small

Cuts in a graph: Min cut



- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut:

$$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

Source: Steve Seitz

Minimum cut

- Problem with minimum cut:
  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

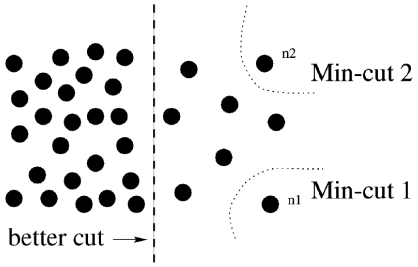
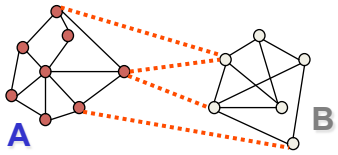


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut



Normalized Cut

- fix bias of Min Cut by **normalizing** for size of segments:

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

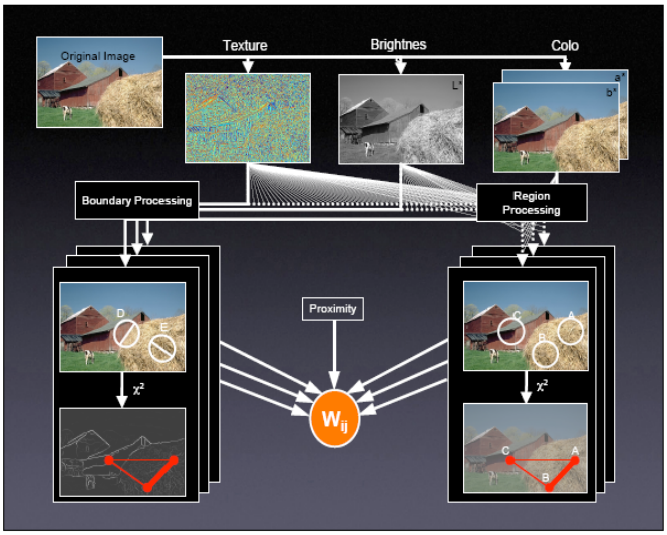
assoc(A,V) = sum of weights of all edges that touch A

- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Source: Steve Seitz

Cue Combination



J. Kosecka

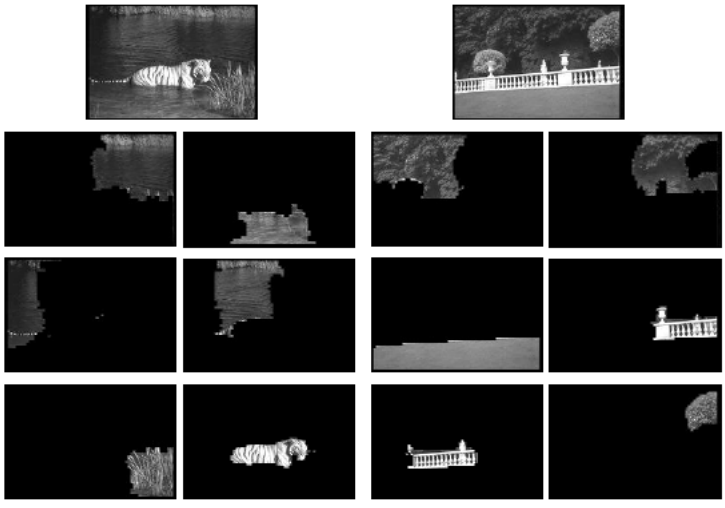


Figure from "Image and video segmentation: the normalised cut framework", by Shi and Malik, copyright IEEE, 1998

Example results

Results: Berkeley Segmentation Engine

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

Graph Based Segmentation

- Given representation of an image as a graph G(V,E)
- Partition the graph into C components, such that all the nodes within a component are similar
- Minimum weight spanning tree algorithm

1. Start with pixels as vertices, edge as similarity between neigbours, gradualy build connected components of the graph
2. Pick the lowest cost edge, the decision whether to join the two components or keep them apart depends on similarity within the components is smaller then similarity between the components. If the difference between the components is small then join them, else do nothing
3. Stopping criterion number of desired segments

Ref. P. Felzenswalb and D. Huttenlocher Efficient Graph Based Image Segmentation

MWST algorithm

- Idea – merge regions in decreasing order of the edges separating them
- Regions connected by weak edges merged first
- Each region characterized by internal difference

$$Int(R) = \min_{\{e \in R\}} w(e)$$

- Region difference – minimum weight connecting two regions

$$Diff(R_1, R_2) = \min_{\{v1 \in R1, v2 \in R2\}} w(e)$$

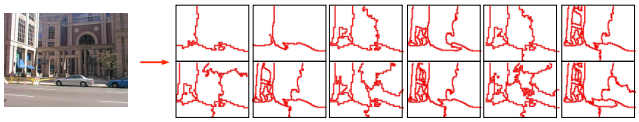
- Merge any two regions who's difference is smaller then the minimum internal difference of these regions

$$MInt(R_1, R_2) = \min(Int(R_1) + \tau(R_1), Int(R_2) + \tau(R_2))$$

MWST graph based segmentation

- Example results

Segments as primitives for recognition

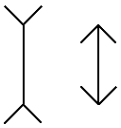


- B. Russell et al., "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections," CVPR 2006

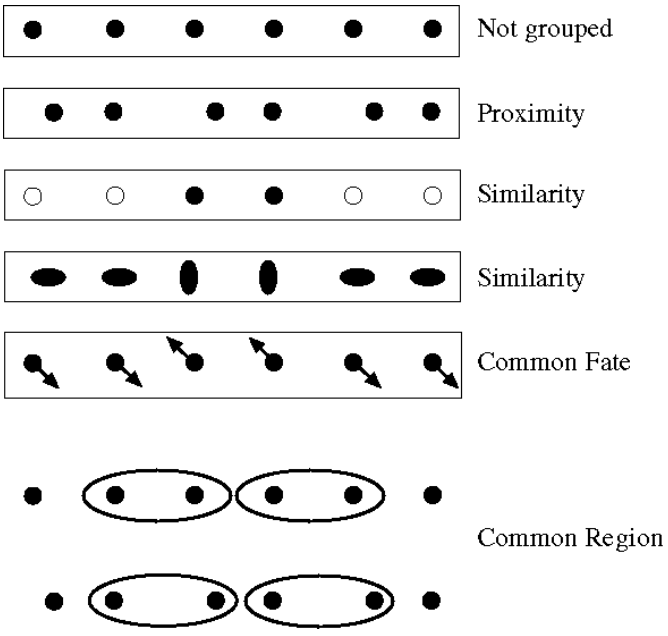
Slide credit: Lana Lazebnik

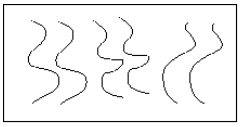
Grouping in humans

- Figure-ground discrimination
  - grouping can be seen in terms of allocating some elements to a figure, some to ground
  - impoverished theory

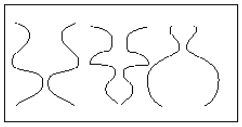


- Gestalt properties
  - elements in a collection of elements can have properties that result from relationships (Muller-Lyer effect)
    - Gestalt-qualitat
  - A series of factors affect whether elements should be grouped together
    - Gestalt factors

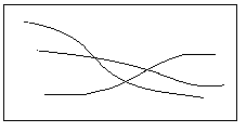




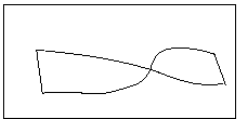
Parallelism



Symmetry



Continuity



Closure