Unit 5 : Image Segmentation

- K-means Clustering
- Mean Shift Segmentation
- Normalized Cuts Segmentation

Ref: Szeliski Sec. 7.5

Image Segmentation

- Goal: Identify groups of pixels that go together

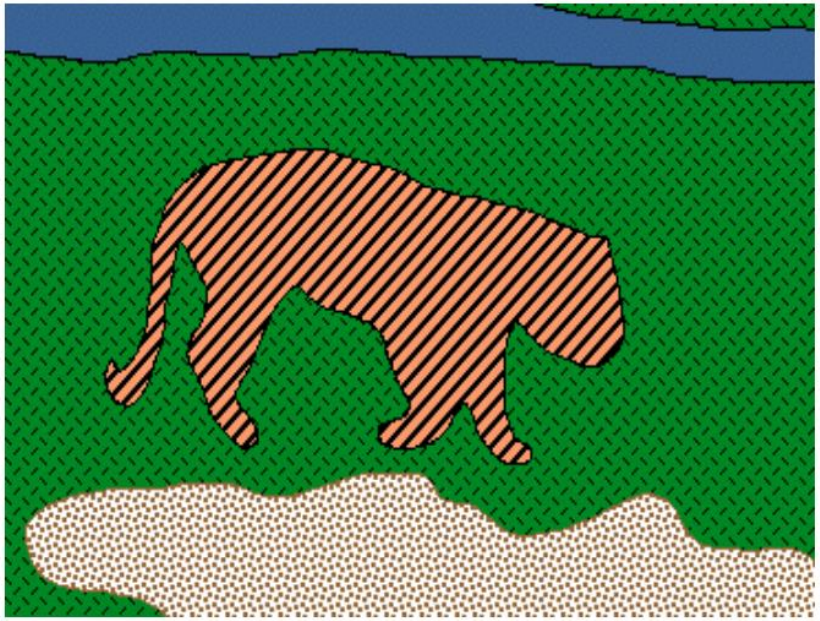


image credit: Steve Seitz, Kristen Grauman

Types of Image Segmentation

- Semantic Segmentation: Assign labels to the groups of pixels

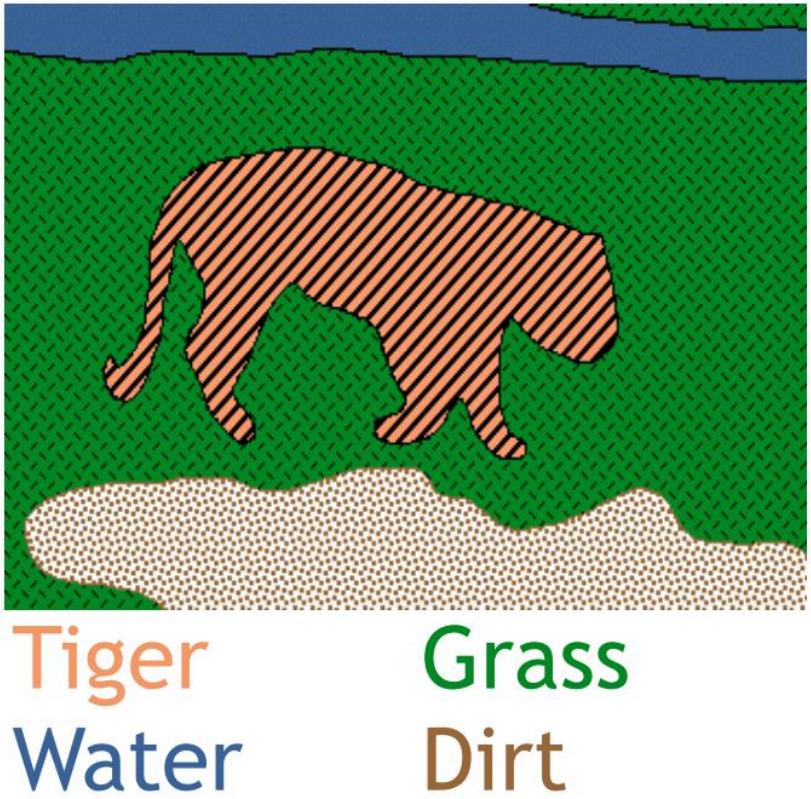


image credit: Steve Seitz, Kristen Grauman

Types of Image Segmentation

- Figure-ground segmentation: separate pixels into foreground or background classes

image credit: Carsten Rother

Types of Image Segmentation

- Co-segmentation: Segment common objects from a set of images

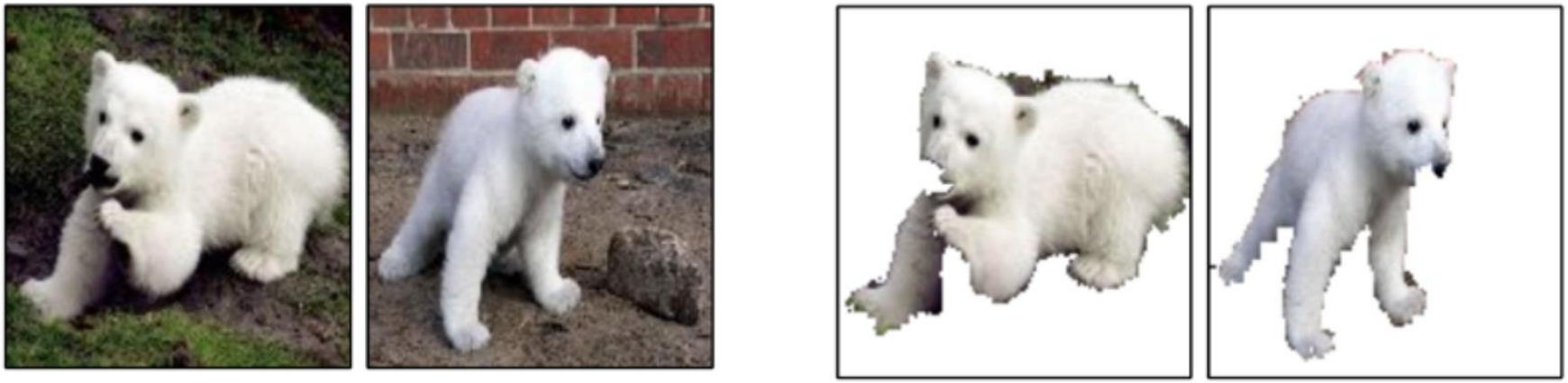


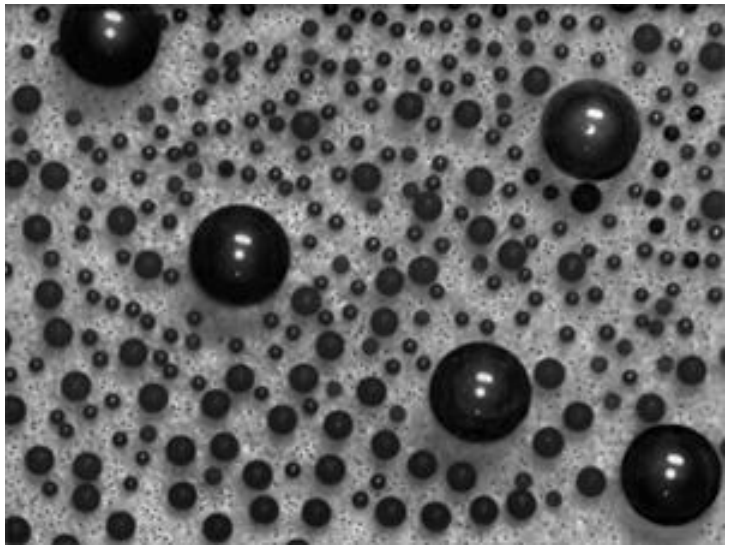
image credit: Armand Joulin

Applications of Image Segmentation

- Image editing
- Background replacement
- Pre-processing for classification
- Improve image classification accuracy
- Analysis of segmented regions

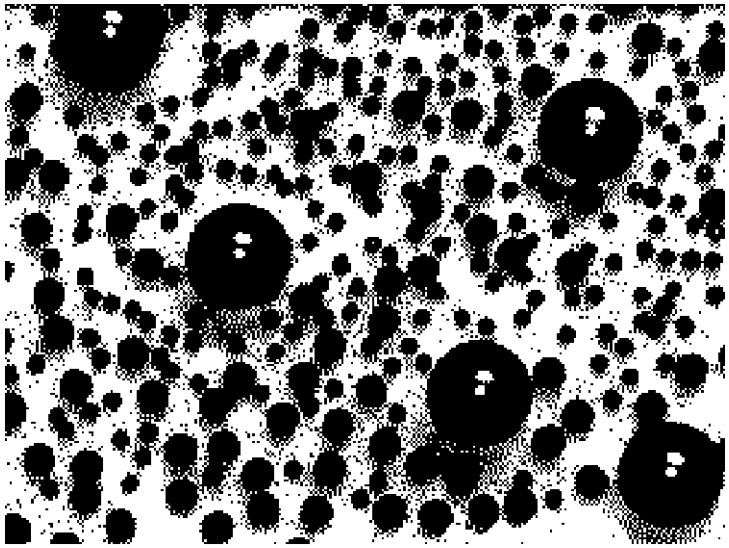
Histogram-based segmentation

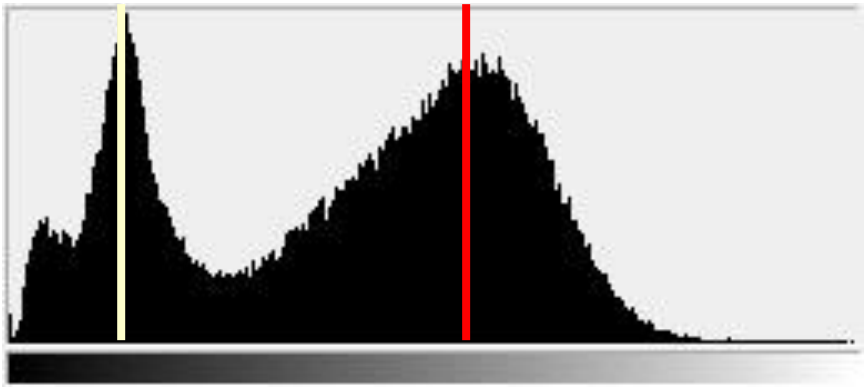
- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color



Histogram-based segmentation

- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color

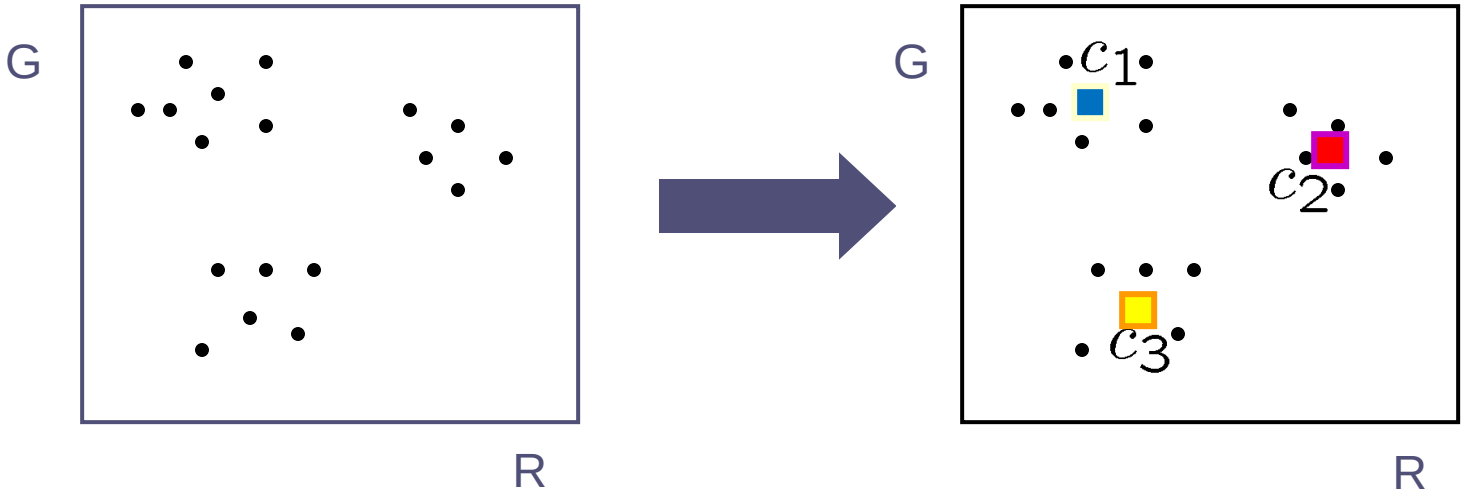




Here's what it looks like if we use two colors

Clustering

- How to choose the representative colors?
  - This is a clustering problem!



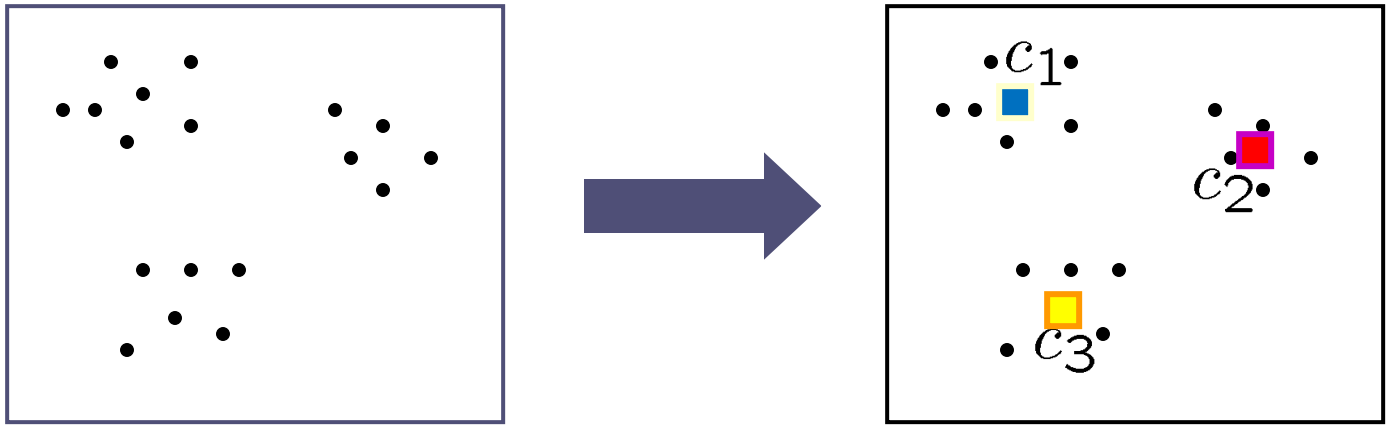
Objective

- Each point should be as close as possible to a cluster center
  - Minimize sum squared distance of each point to closest center

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Break it down into subproblems

- Suppose I tell you the cluster centers c_i
  - Q: how to determine which points to associate with each c_i?
  - A: for each point p, choose closest c_i



Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

K-means clustering

- K-means clustering algorithm
  1. Randomly initialize the cluster centers, c_1, ..., c_K
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, update c_i to be the mean of all points in cluster i
  4. If c_i have changed, repeat Step 2

- Properties
  - Will always converge to *some* solution
  - Can be a "local minimum"
    - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

K-Means++

- Can we prevent arbitrarily bad local minima?

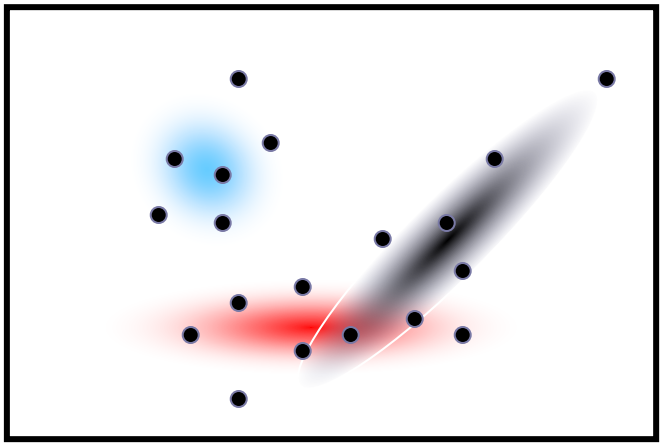
1. Randomly choose the first center.
2. Pick the next new center from dataset with prob. proportional to: $\sum_i \ ||p - c_i||^2$
   (contribution of *p* to total error)
3. Repeat until *k* centers are selected.

Probabilistic clustering

- Basic questions
  - What's the probability that a point **x** is in cluster m?
  - What's the shape of each cluster?
- K-means doesn't answer these questions

- Basic idea
  - instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function
  - This function is called a **generative model**
    - defined by a vector of parameters θ

Mixture of Gaussians



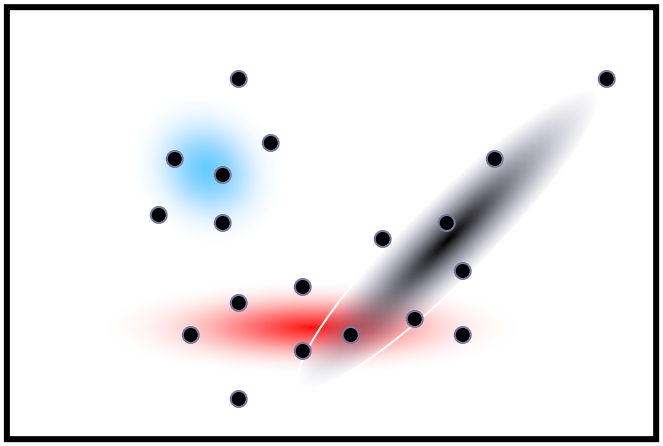
- One generative model is a mixture of Gaussians (MOG)
  - K Gaussian blobs with means $\boldsymbol{\mu_b}$ covariance matrices $\mathbf{V_b}$, dimension d
    - blob *b* defined by $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - blob *b* is selected with probability α_b
  - the likelihood of observing **x** is a weighted mixture of Gaussians

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b)$$

  - where

$$\theta = [\mu_1, \ldots, \mu_n, V_1, \ldots, V_n]$$

Expectation maximization (EM)



- Goal
  - find blob parameters θ that maximize the likelihood function:

$$P(data|\theta) = \prod_x P(x|\theta)$$

- Approach:
  1. E step: given current guess of blobs, compute ownership of each point
  2. M step: given ownership probabilities, update blobs to maximize likelihood function
  3. repeat until convergence

EM details

- E-step
  - compute probability that point **x** is in blob i, given current guess of **θ**

$$P(b|x,\mu_b,V_b) = \frac{\alpha_b P(x|\mu_b,V_b)}{\sum_{i=1}^{K}\alpha_i P(x|\mu_i,V_i)}$$

- M-step
  - compute probability that blob b is selected

$$\alpha_b^{new} = \frac{1}{N}\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)$$

  - mean of blob b

$$\mu_b^{new} = \frac{\sum_{i=1}^{N} x_i P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

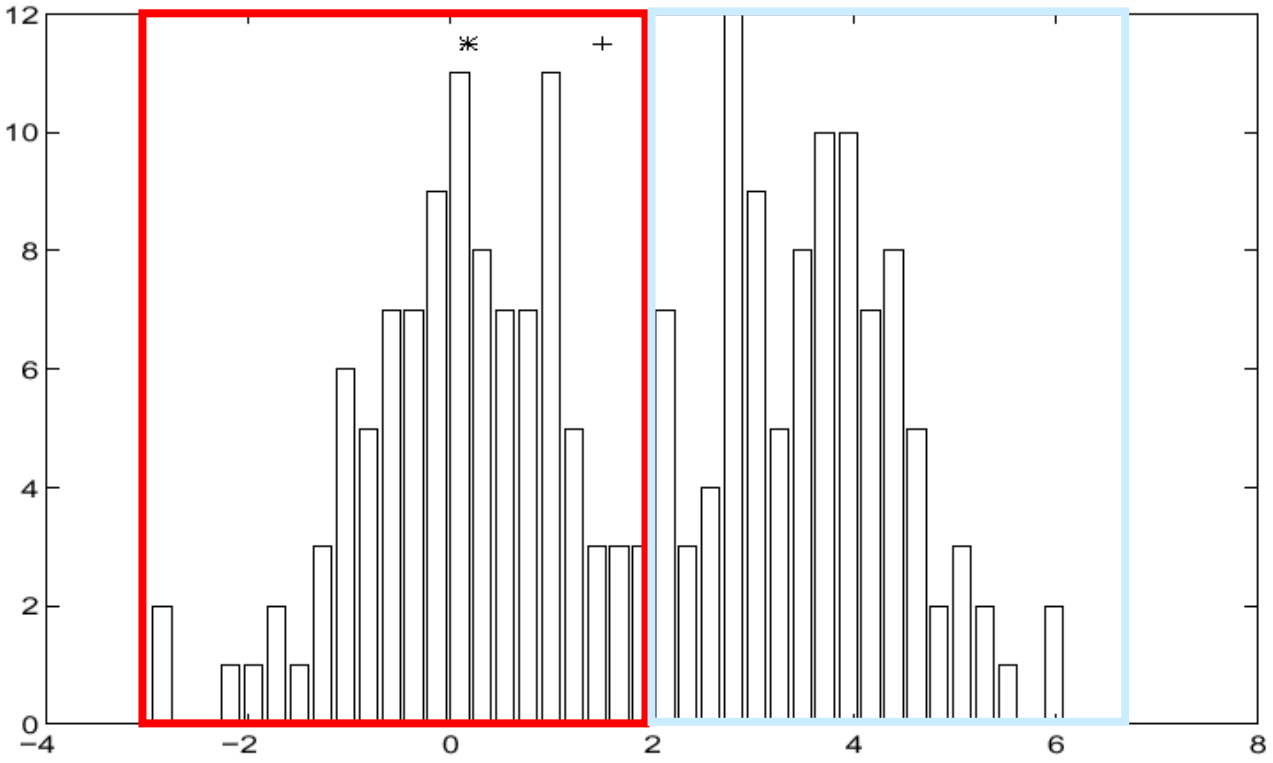
  - covariance of blob b

$$V_b^{new} = \frac{\sum_{i=1}^{N}(x_i-\mu_b^{new})(x_i-\mu_b^{new})^T P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

Applications of EM

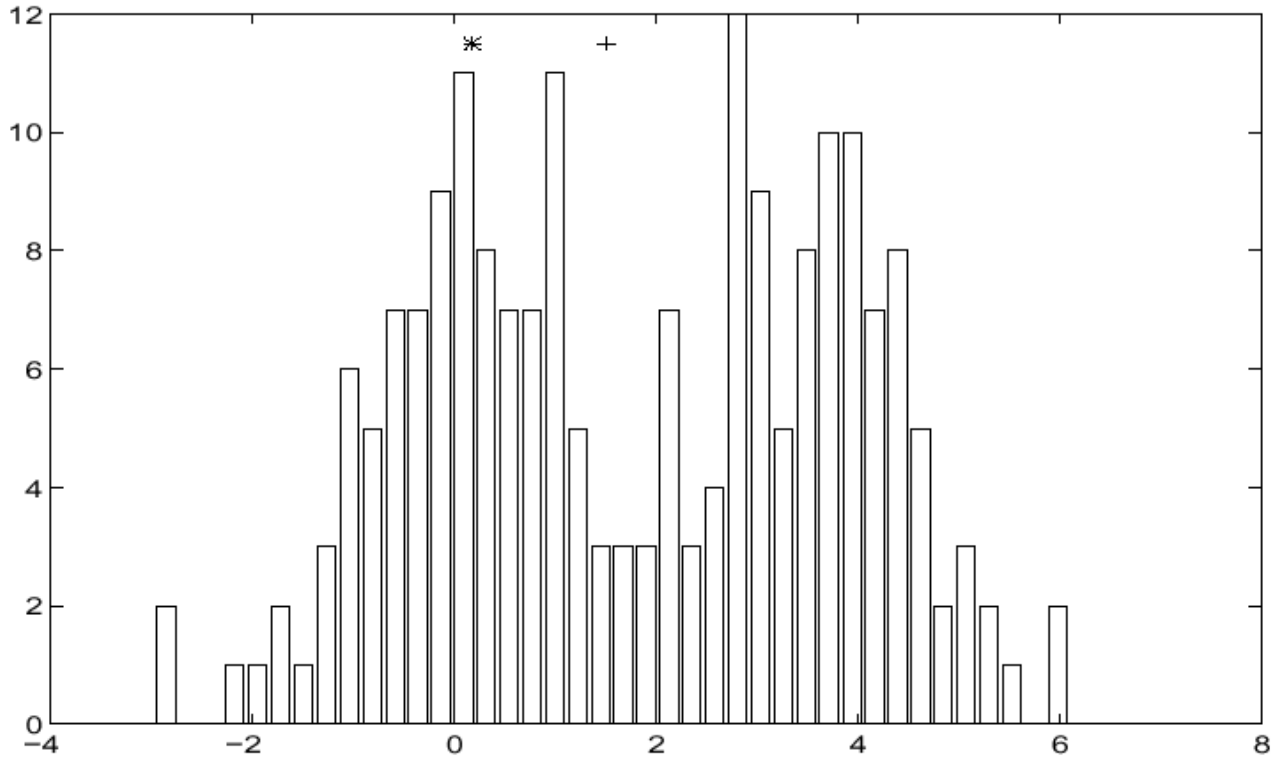
- Turns out this is useful for all sorts of problems
  - any clustering problem
  - any model estimation problem
  - missing data problems
  - finding outliers
  - segmentation problems
    - segmentation based on color
    - segmentation based on motion
    - foreground/background separation
  - ...

Finding Modes in a Histogram



- How Many Modes Are There?
  - Easy to see, hard to compute

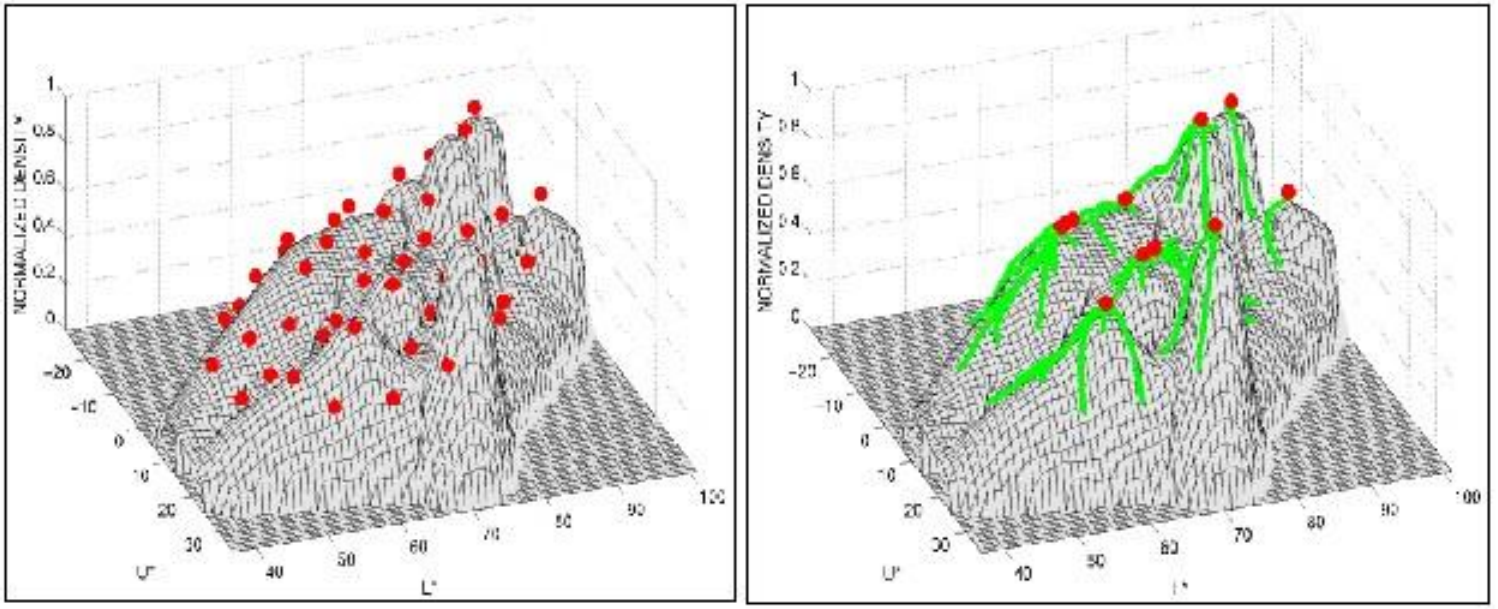
Mean Shift [Comaniciu & Meer]



- Iterative Mode Search
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} xH(x)$
  3. Translate the search window to the mean
  4. Repeat Step 2 until convergence

Mean-Shift

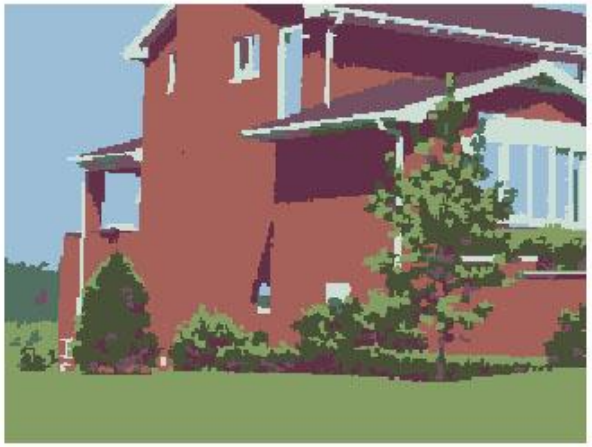
- Approach
  - Initialize a window around each point
  - See where it shifts—this determines which segment it's in
  - Multiple points will shift to the same segment

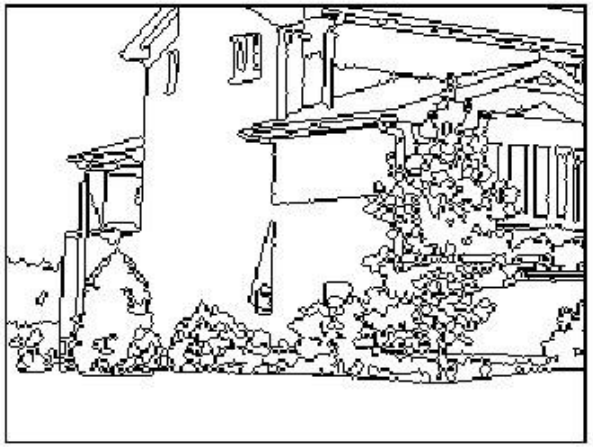


Mean shift trajectories

Mean-shift for image segmentation

- Useful to take into account spatial information
  - instead of (R, G, B), run in (R, G, B, x, y) space
  - D. Comaniciu, P. Meer, Mean shift analysis and applications, *7th International Conference on Computer Vision*, Kerkyra, Greece, September 1999, 1197-1203.
    - http://www.caip.rutgers.edu/riul/research/papers/pdf/spatmsft.pdf



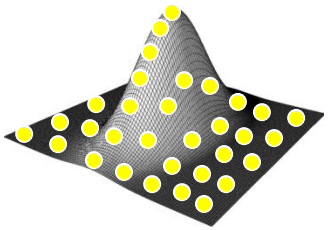


http://www.caip.rutgers.edu/~comanici/segm_images.html

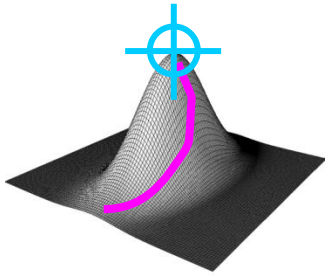
What is Mean Shift ?

A tool for:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in R^N

Non-parametric
Density Estimation

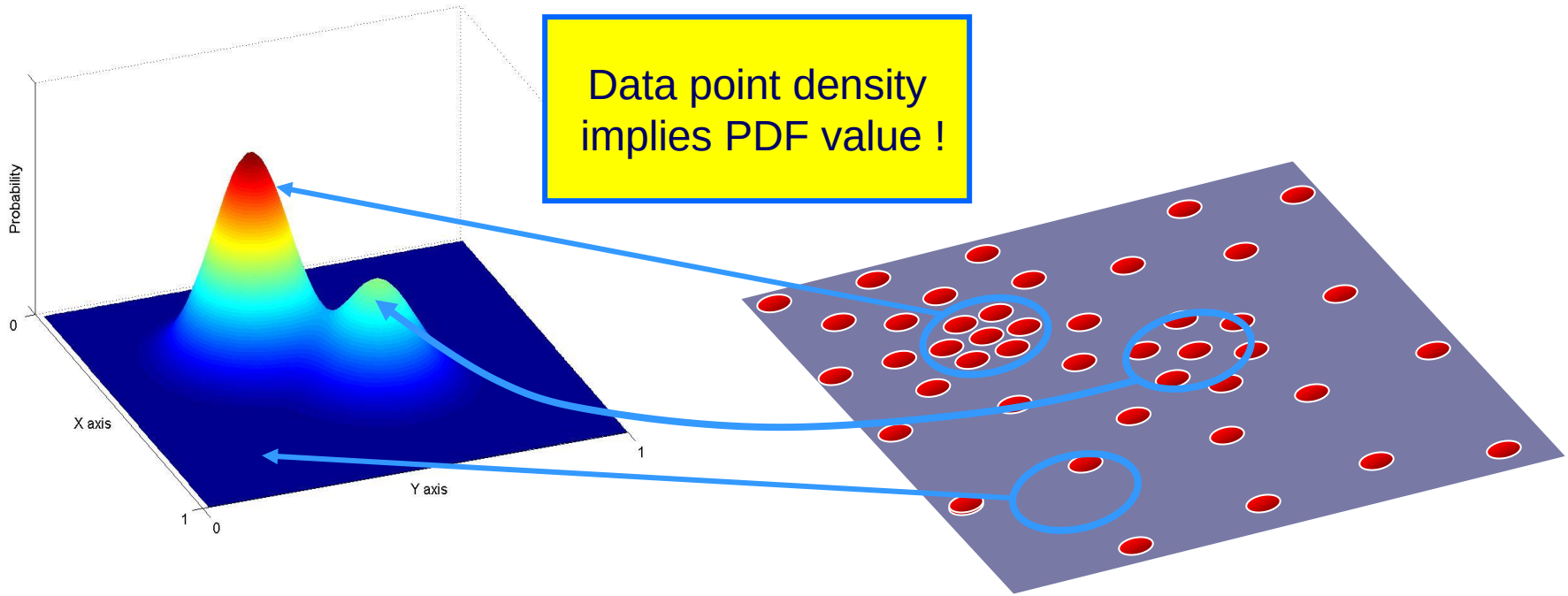


Non-parametric
Density **GRADIENT** Estimation
(Mean Shift)



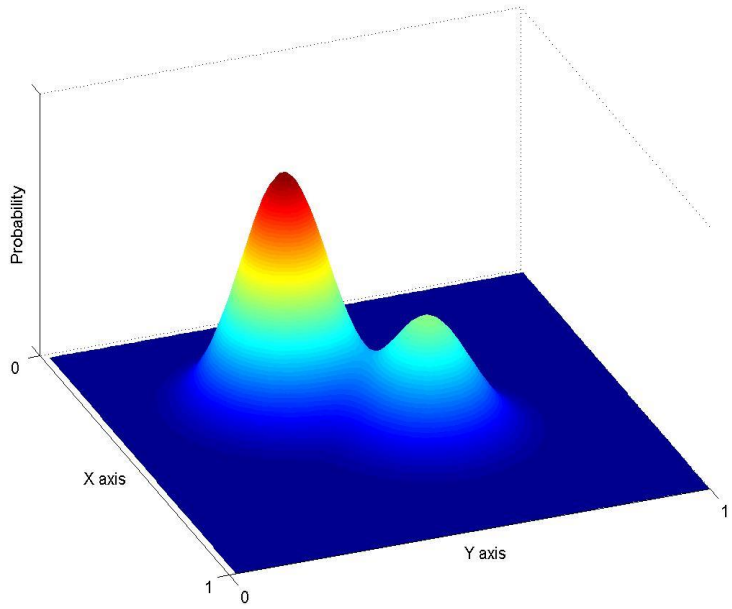
Non-Parametric Density Estimation

Assumption : The data points are sampled from an underlying PDF

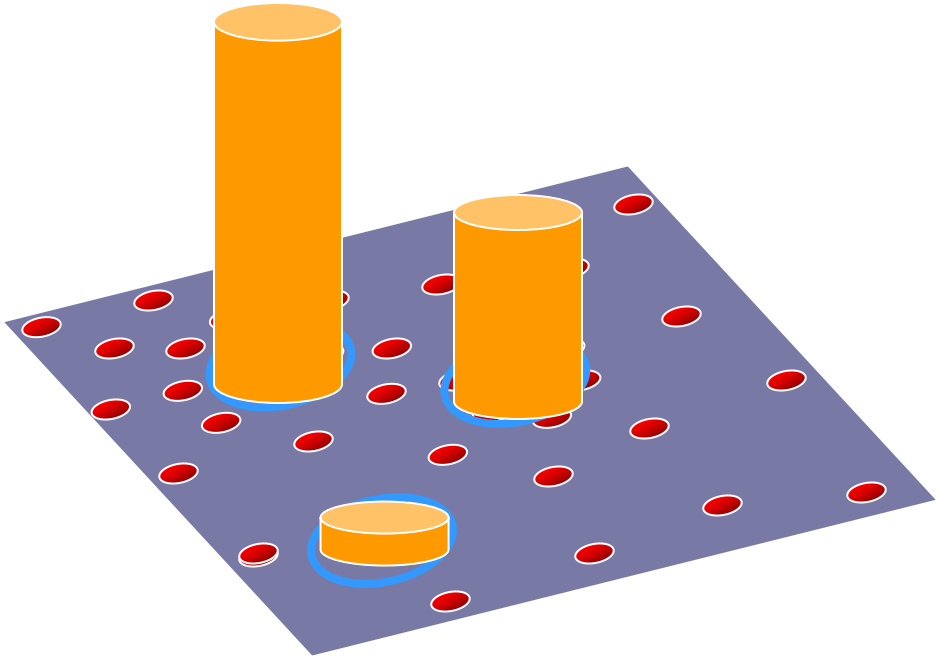


Assumed Underlying PDF **Real Data Samples**

Non-Parametric Density Estimation

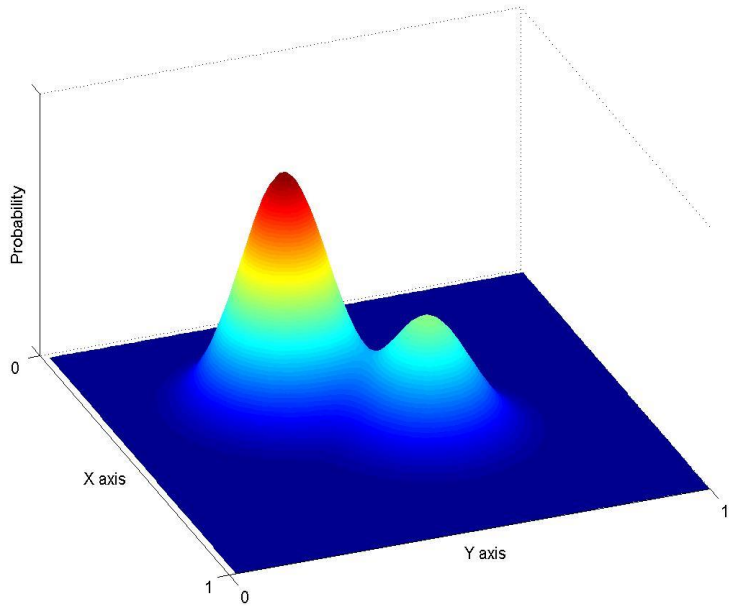


Assumed Underlying PDF

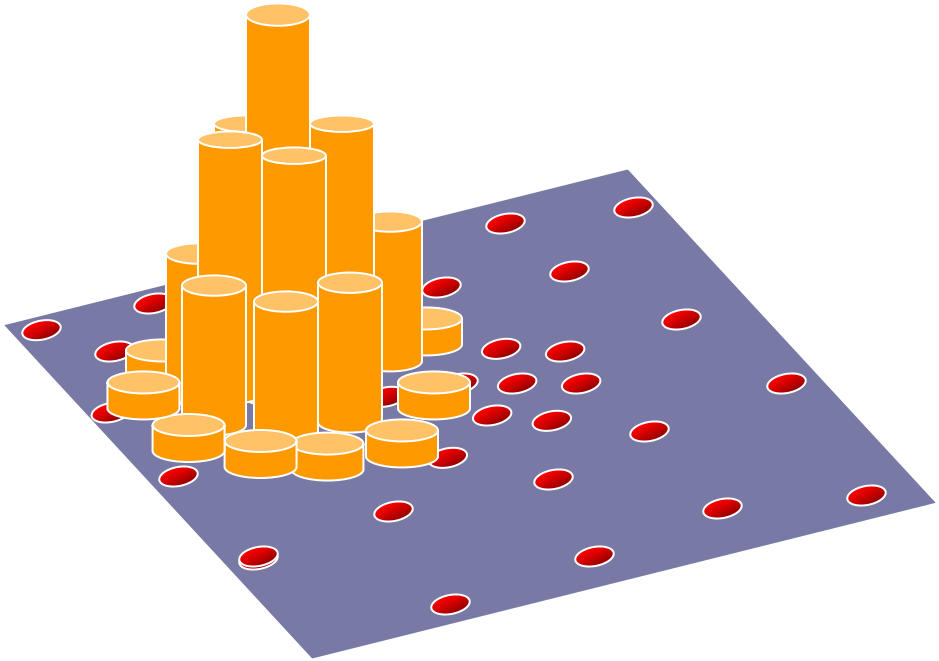


Real Data Samples

Non-Parametric Density Estimation



Assumed Underlying PDF

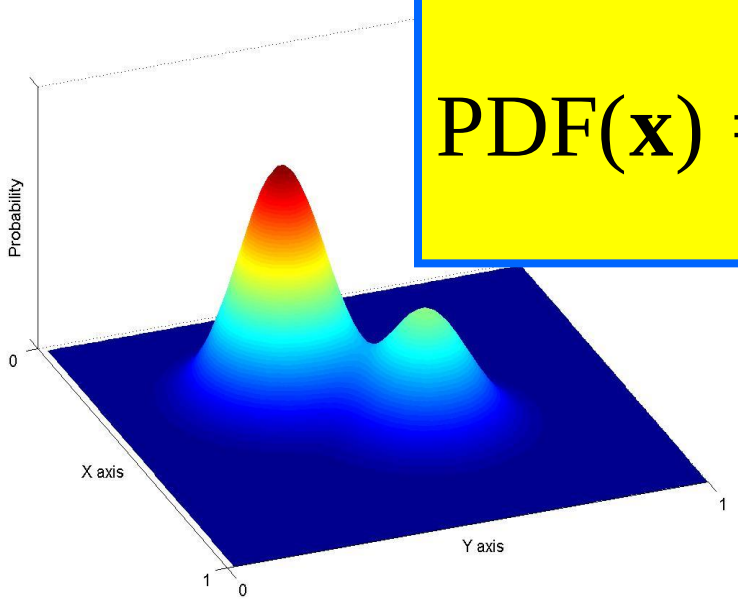


Real Data Samples

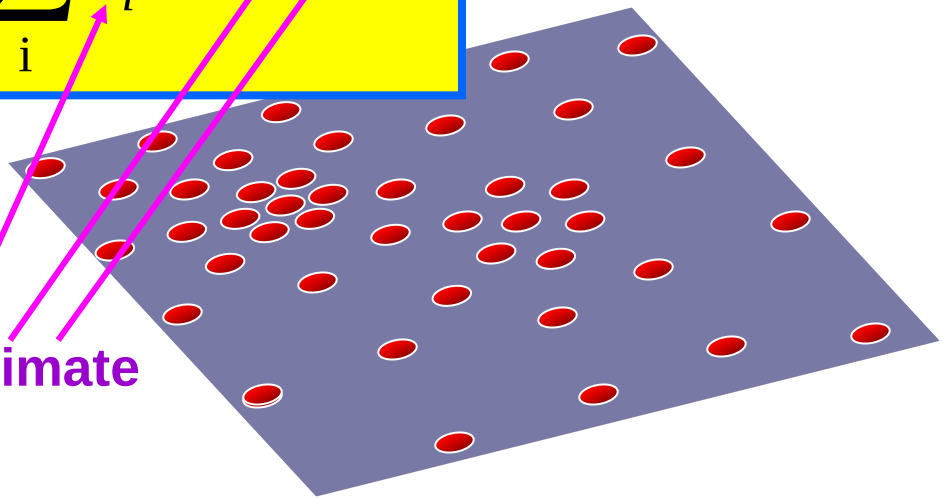
Parametric Density Estimation

Assumption : The data points are sampled from an underlying PDF

$$\mathrm{PDF}(\mathbf{x}) = \sum_{i} c_i \cdot e^{-\frac{(\mathbf{x}-\boldsymbol{\mu}_i)^2}{2\sigma_i^2}}$$



Estimate



Assumed Underlying PDF

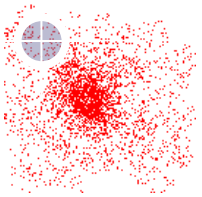
Real Data Samples

Kernel Density Estimation

Parzen Windows - Function Forms

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

A function of some finite number of data points $x_1 \ldots x_n$



In practice one uses the forms:

$$K(\mathbf{x}) = c\prod_{i=1}^{d} k(x_i)$$ or $$K(\mathbf{x}) = ck\left(\|\mathbf{x}\|\right)$$

Same function on each dimension

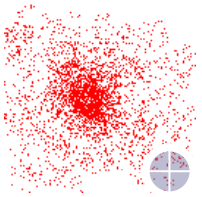
Function of vector length only

Kernel Density Estimation

Various Kernels

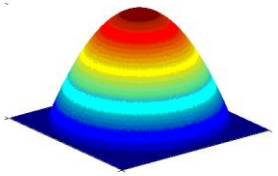
$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

A function of some finite number of data points $x_1 \ldots x_n$

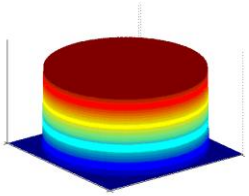


Examples:

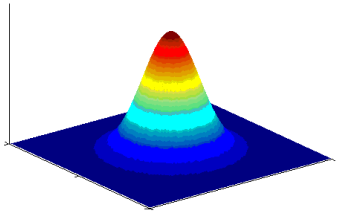
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



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



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



Kernel Density Estimation

Gradient

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

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

Using the Kernel form:

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

Kernel **bandwidth**

We get :

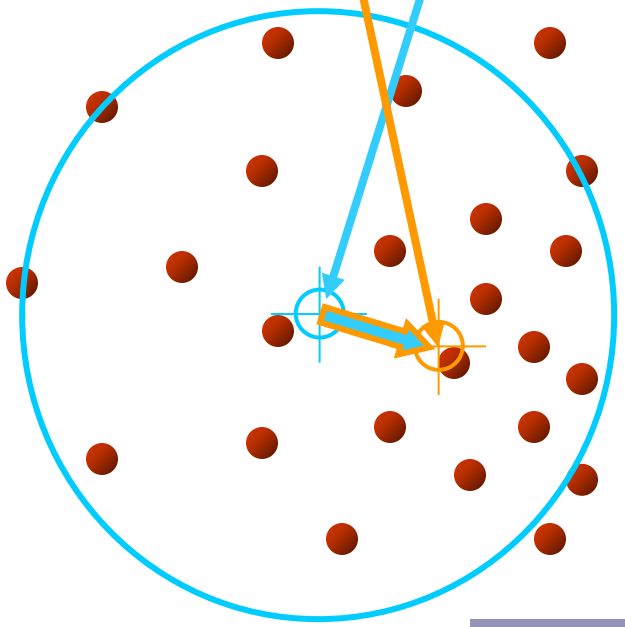
$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n} \nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

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

Computing The Mean Shift

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

Yet another Kernel density estimation !



Simple Mean Shift procedure:

- Compute mean shift vector

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

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

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

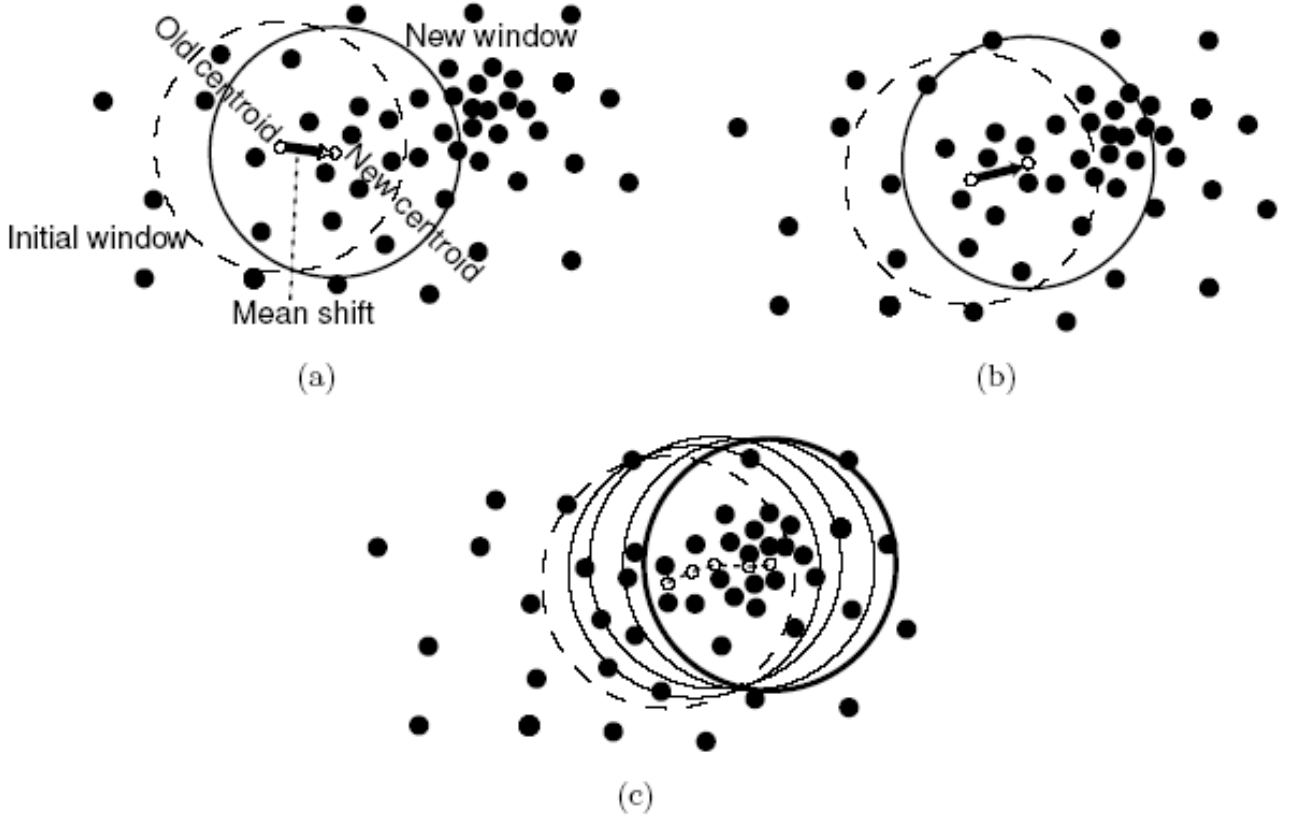
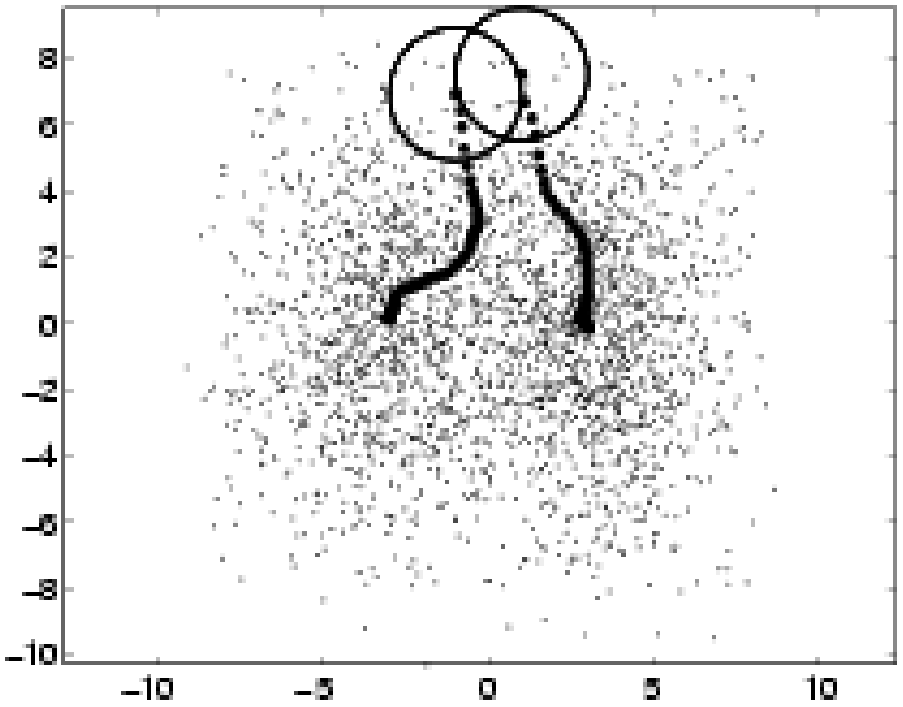


Figure 7.1: Principle of the mean shift procedure. The most dense region of data is identified in an iterative process. (a) The initial region of interest is randomly positioned over data and its centroid is determined. The new region is moved to the location of the identified centroid. The vector determining the region's positional change is the mean shift. (b) Next step of the mean shift procedure–a new mean shift vector is determined and the region is moved accordingly. (c) The mean shift vectors are determined in the remaining steps of the procedure until convergence. The final location identifies the local density maximum, or the local *mode*, of the probability density function.

Real Modality Analysis

An example



Window tracks signify the steepest ascent directions

Mean Shift Strengths & Weaknesses

Strengths :

- Application independent tool
- Suitable for real data analysis
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- *h* (kernel bandwidth) has a physical meaning, unlike K-Means

Weaknesses :

- The window size (bandwidth selection) is not trivial
- Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes

➔ Use adaptive window size

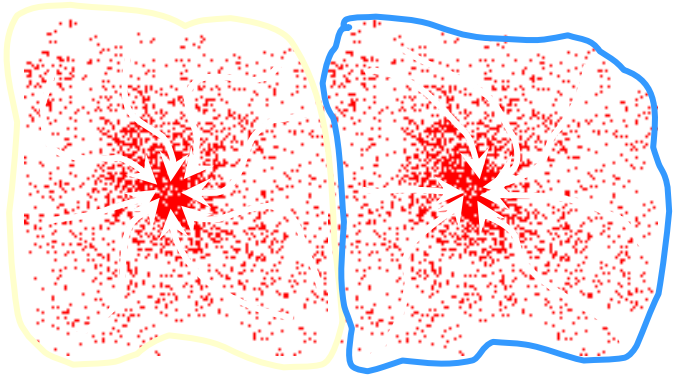
Mean Shift Applications

- Clustering
- Image segmentation
- Discontinuity-preserving filtering
- Mean-shift tracking

Clustering

Cluster : All data points in the ***attraction basin*** of a mode

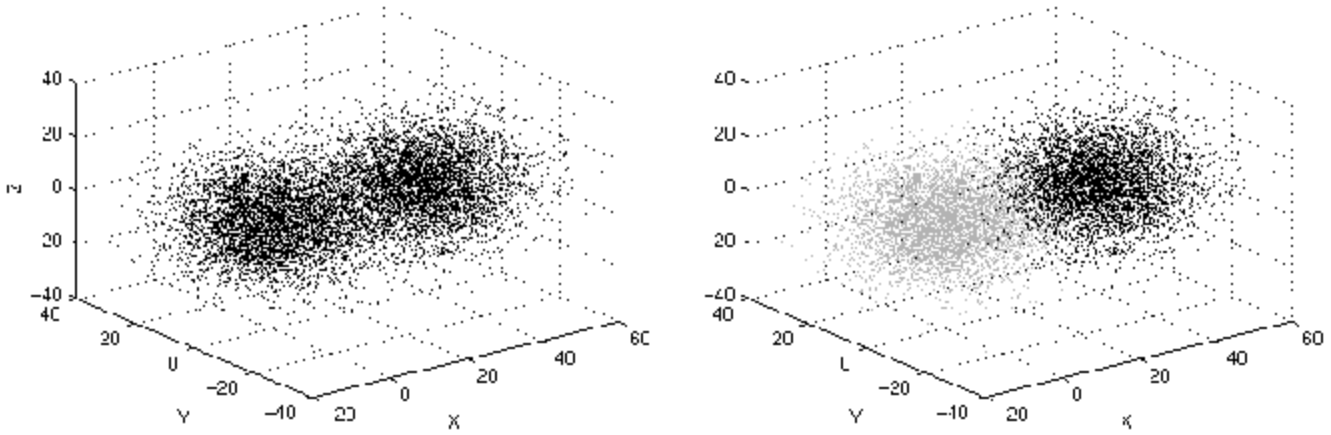
Attraction basin : the region for which all trajectories lead to the same mode



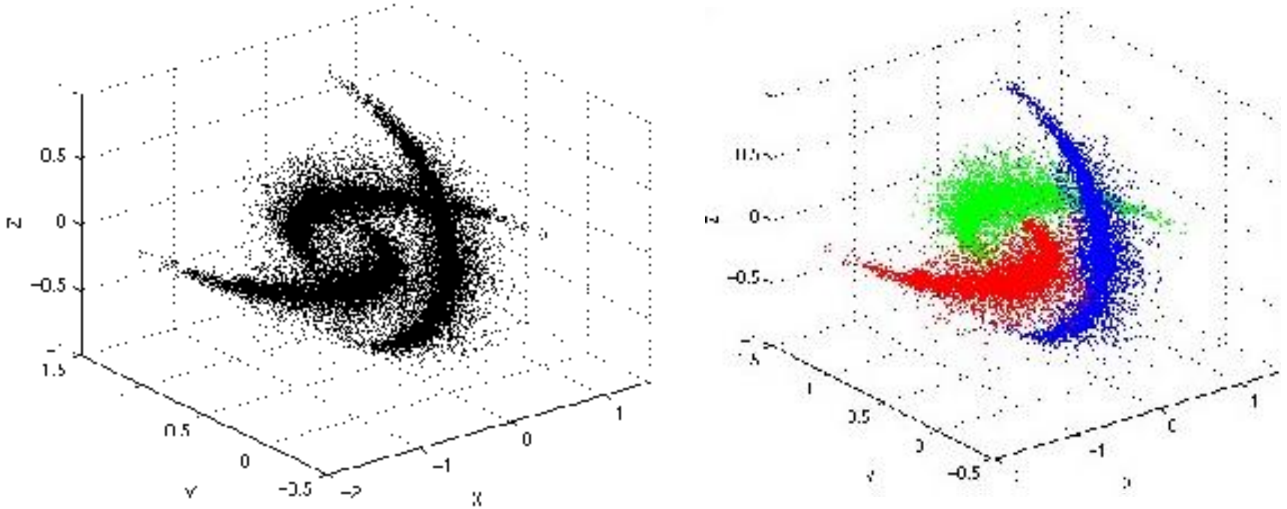
Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer

Clustering

Synthetic Examples



Simple Modal Structures



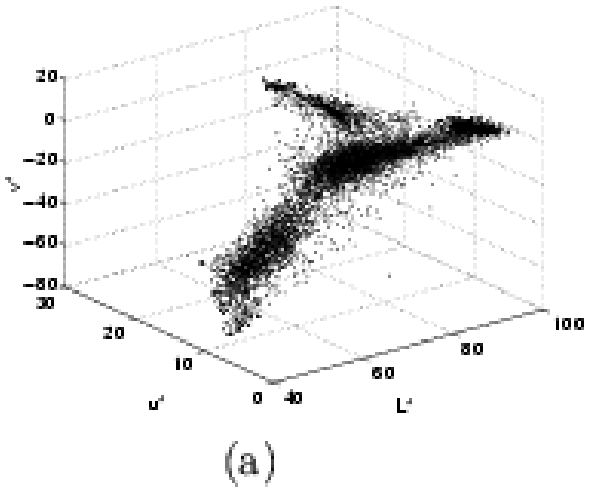
Complex Modal Structures

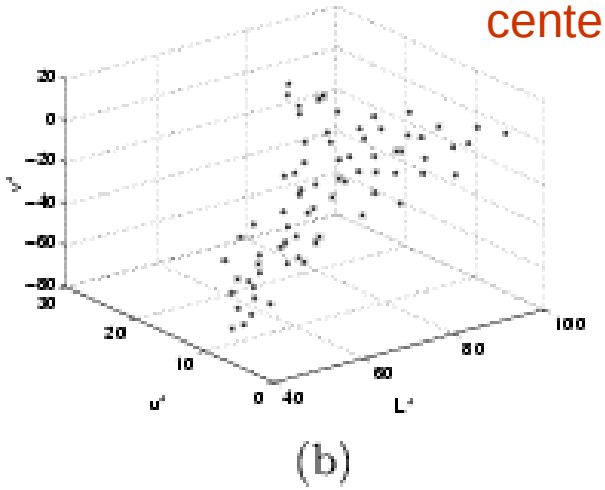
Clustering

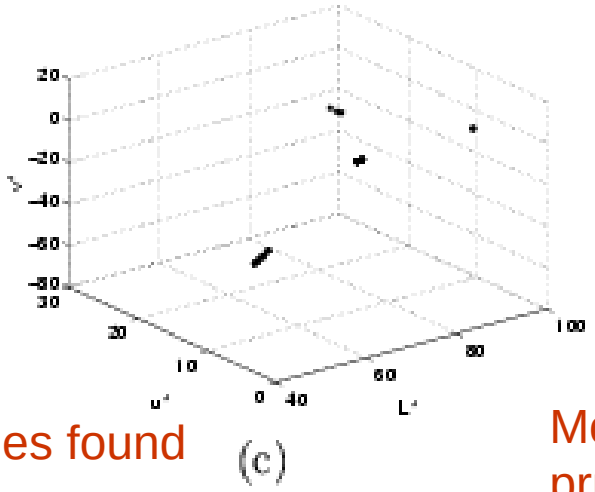
Real Example

Feature space:
L*u*v representation

Initial window centers

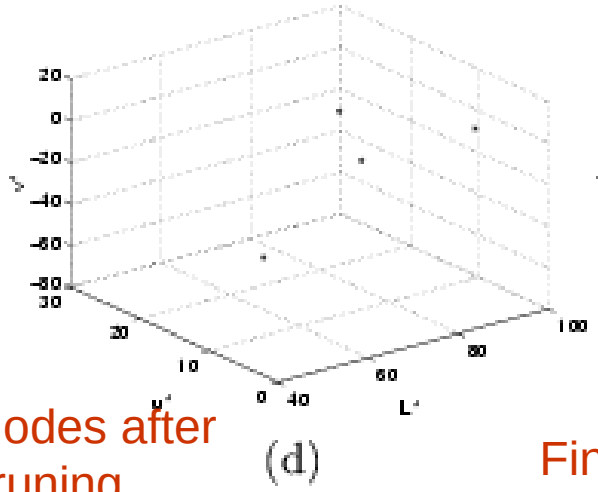


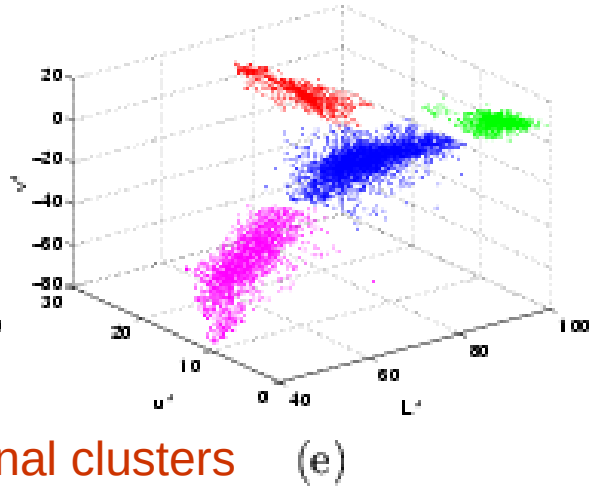




Modes found

Modes after pruning

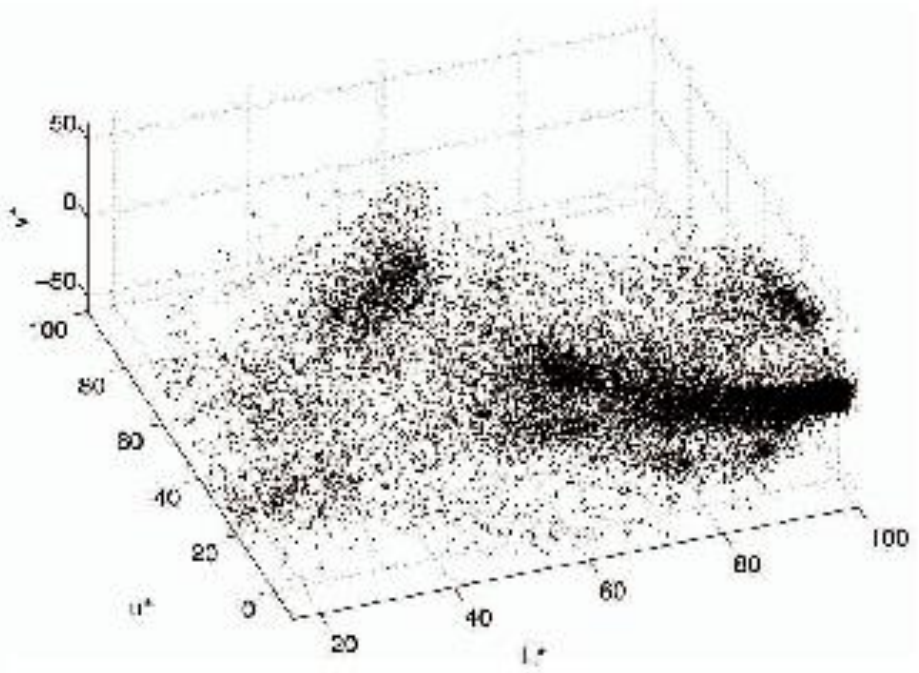




Final clusters

Clustering

Real Example

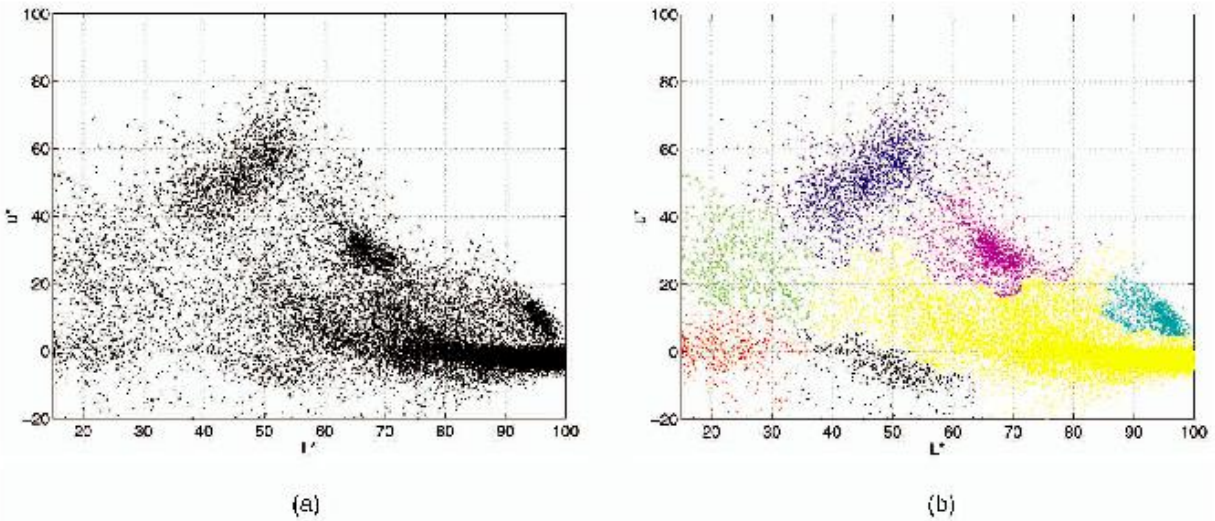


L*u*v space representation

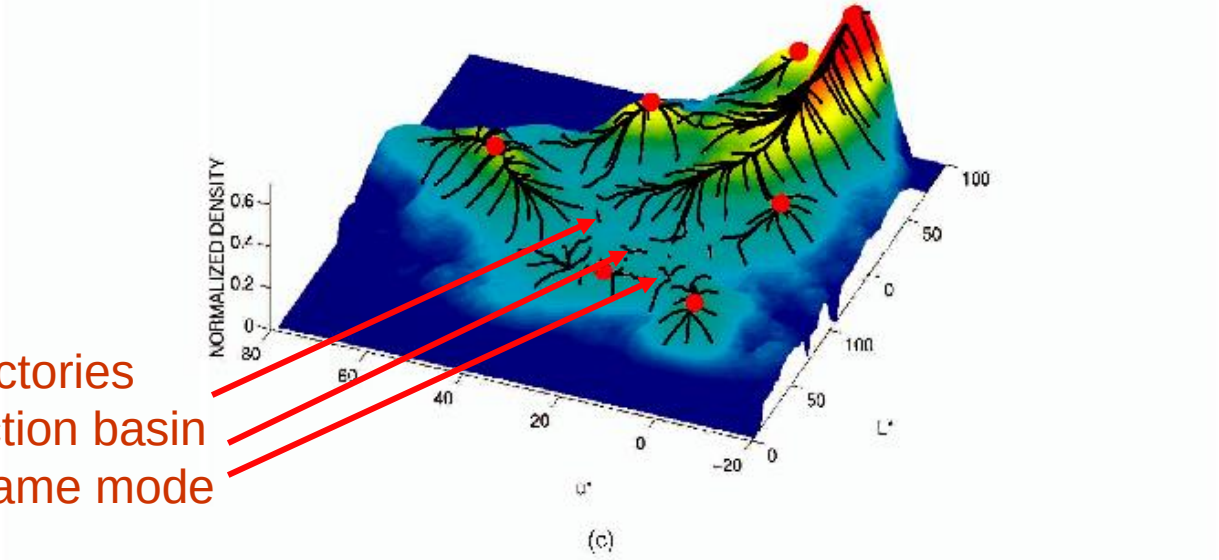
Clustering

Real Example

2D (L*u) space representation



Final clusters



Not all trajectories in the attraction basin reach the same mode

Discontinuity Preserving Smoothing

Feature space : Joint domain = spatial coordinates + color space

$$K(\mathbf{x}) = C \cdot k_s\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|\right) \cdot k_r\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|\right)$$

Meaning : treat the image as data points in the spatial and gray level domain

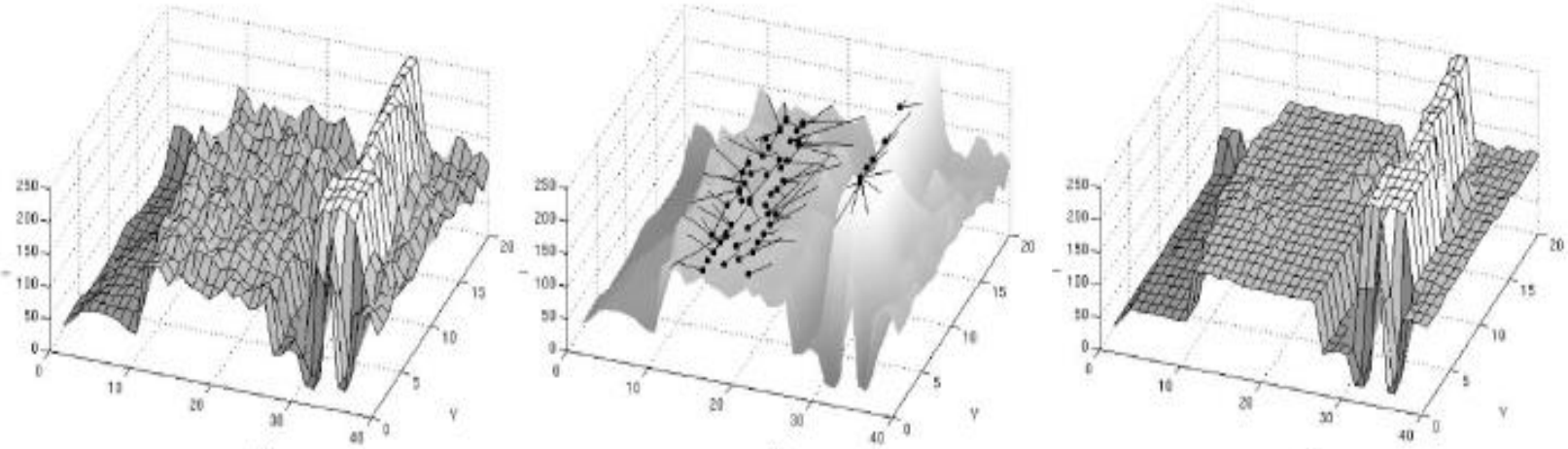
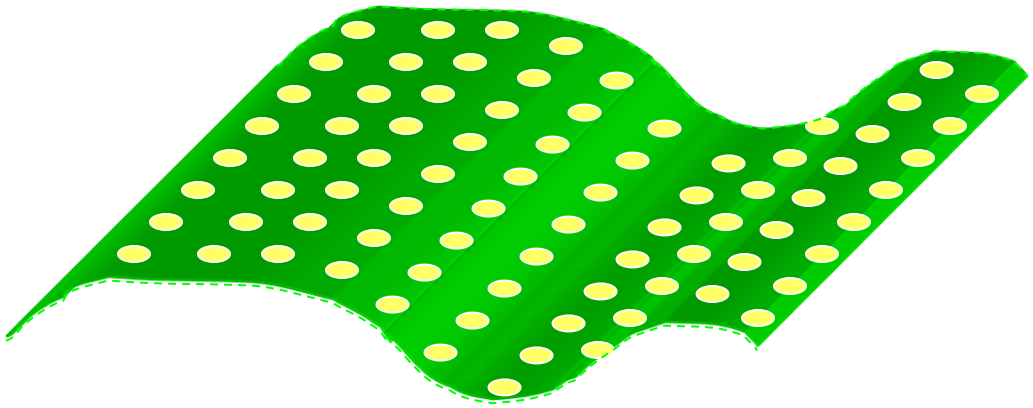


Image Data (slice)

Mean Shift vectors

Smoothing result

Discontinuity Preserving Smoothing



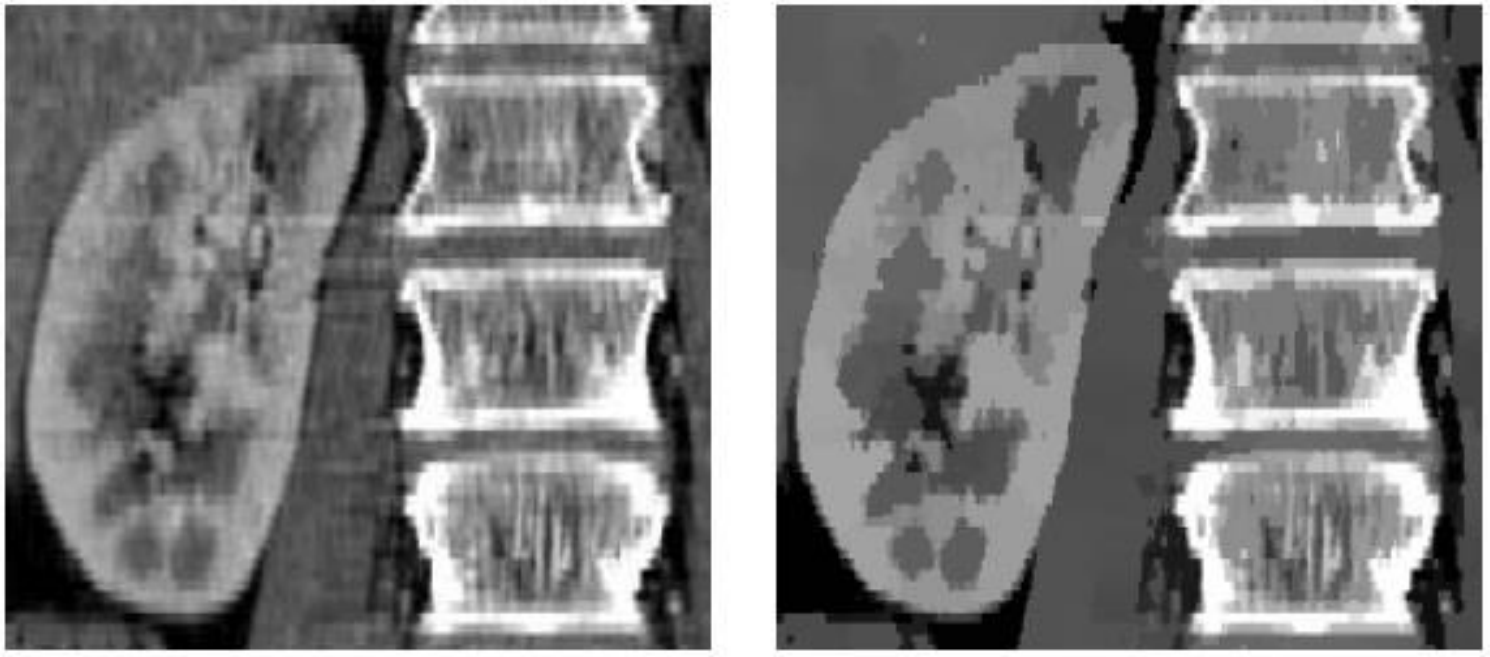
The image gray levels…

… can be viewed as data points in the *x, y, z* space (joined spatial And color space)

Algorithm 7.2: Mean shift discontinuity preserving filtering

1. For each image pixel $\mathbf{x}_i$, initialize step $j = 1$ and $\mathbf{y}_{i,1} = \mathbf{x}_i$.
2. Compute $\mathbf{y}_{i,j+1}$ as given in equation (7.12) until convergence $\mathbf{y}_{i,\text{con}}$.
3. The filtered pixel values are defined as $\mathbf{z}_i = (\mathbf{x}_i^s, \mathbf{y}_{i,\text{con}}^r)$, i.e., the value of the filtered pixel at the location $\mathbf{x}_i^s$ is assigned the image value of the pixel of convergence $\mathbf{y}_{i,\text{con}}^r$.

Mean-shift Filtering Example

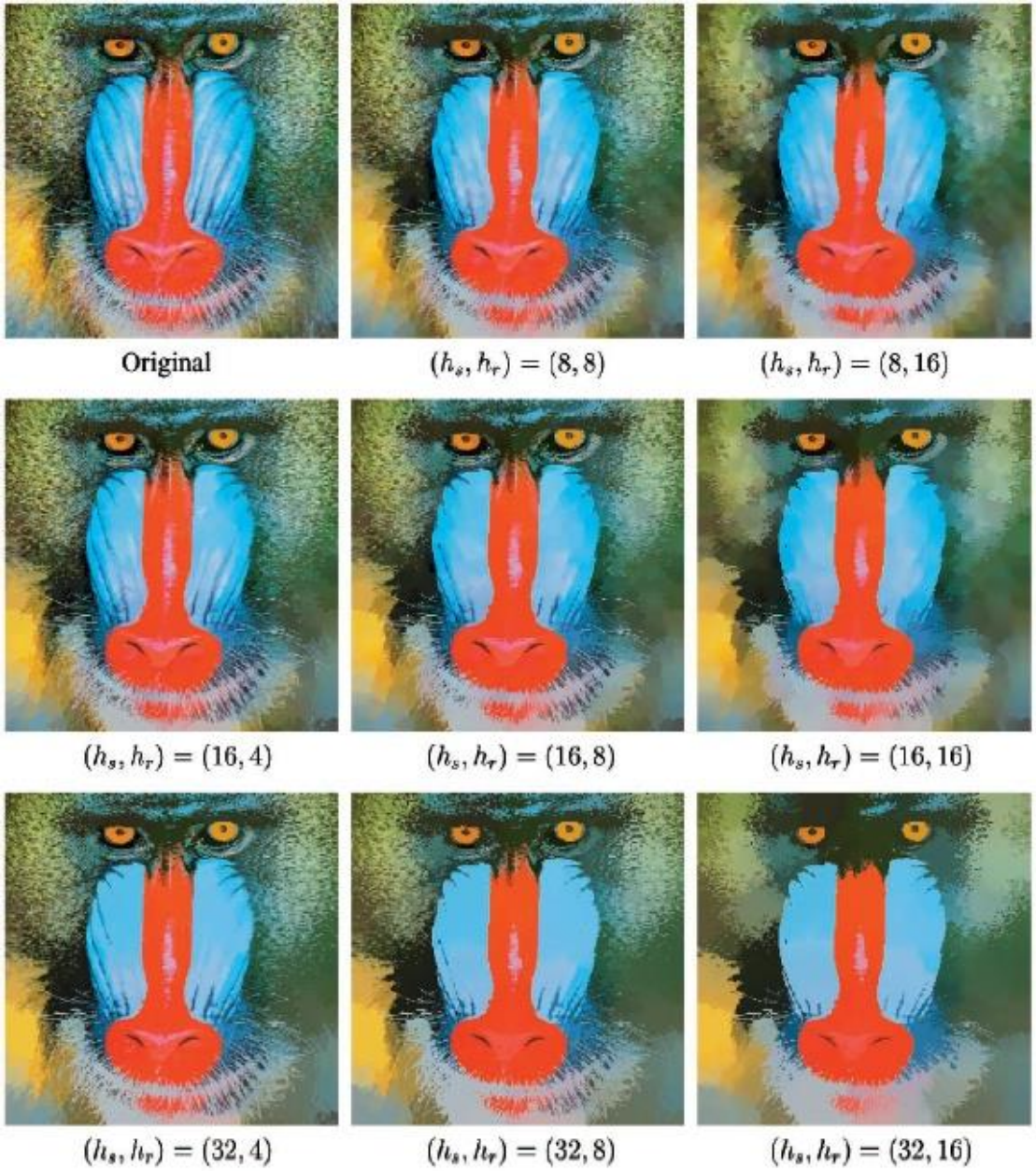


(a) (b)

Figure 7.4: Meanshift filtering. (a) Original X-ray computed tomography image of human kidney and spine. (b) Filtered image. *Courtesy of R. Beichel, Graz University of Technology.*

Discontinuity Preserving Smoothing

The effect of window size in spatial and range spaces



Discontinuity Preserving Smoothing

Example

Segmentation

Algorithm:

- Run Filtering (*discontinuity preserving smoothing*)
- Cluster the clusters which are closer than window size

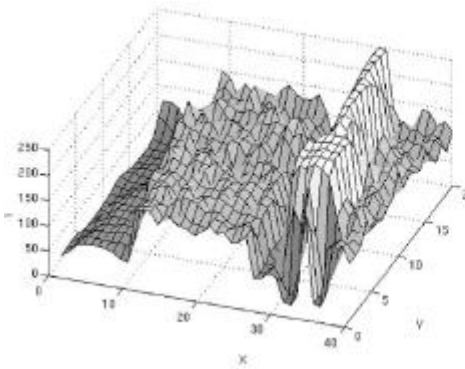
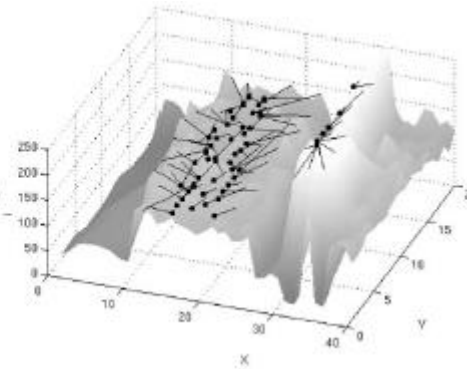
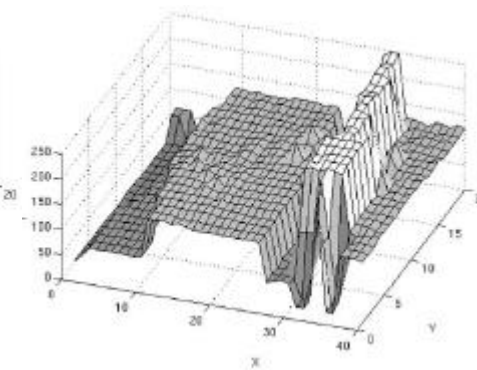


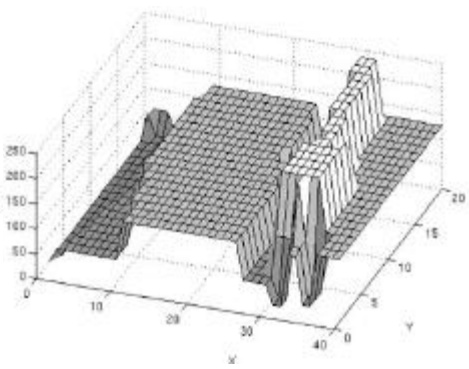
Image Data (slice)



Mean Shift vectors



Smoothing result



Segmentation result

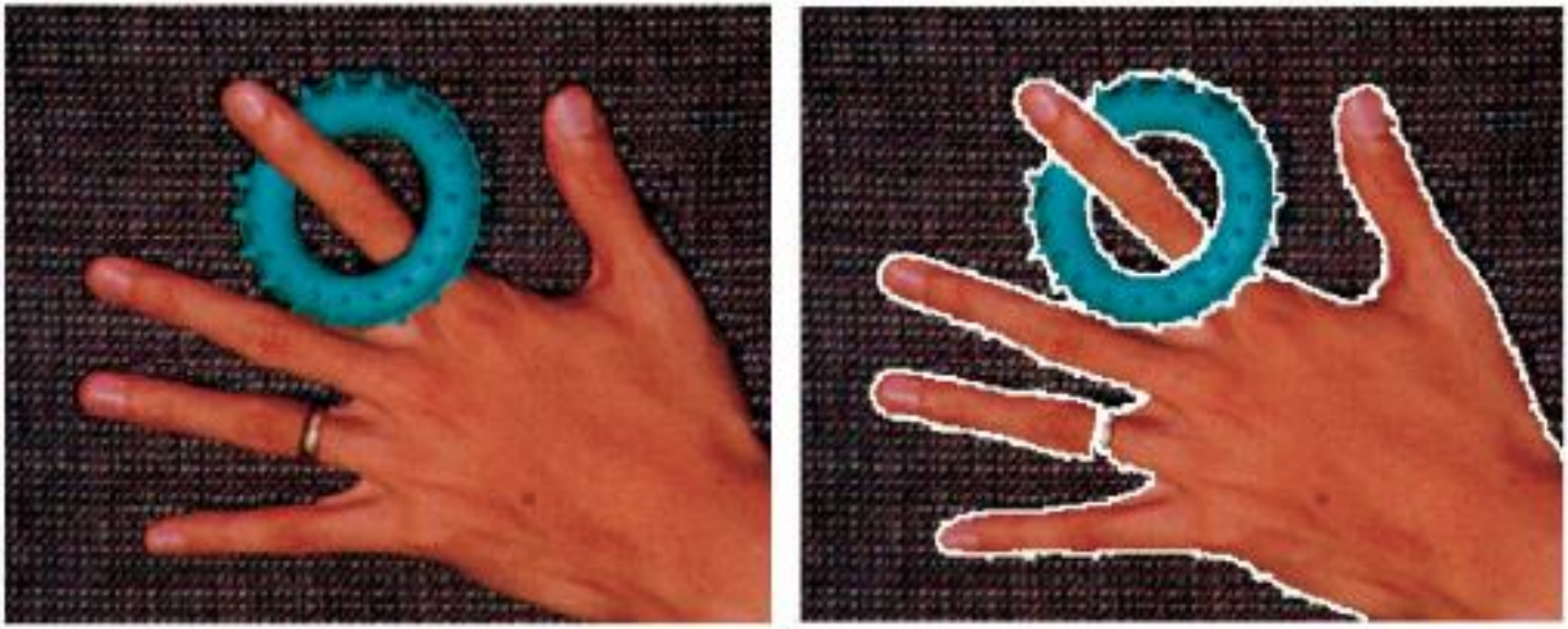
Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer
http://www.caip.rutgers.edu/~comanici

Algorithm 7.3: Mean shift image segmentation

1. Employ the *mean shift discontinuity preserving filtering* and store all information about the d-dimensional convergence points $\mathbf{y}_{i,\text{con}}$.
2. Determine the clusters $\{C_p\}_{p=1,...,m}$ by grouping all $\mathbf{z}_i$, which are closer than h_s in the spatial domain and h_r in the range domain. In other words, merge the *basins of attraction* of these convergence points.
3. Assign $L_i = \{p | \mathbf{z}_i \in \mathbf{C}_p\}$ for each pixel $i = 1, ..., n$.
4. If desired, eliminate regions smaller than P pixels as described in Algorithm 6.22.

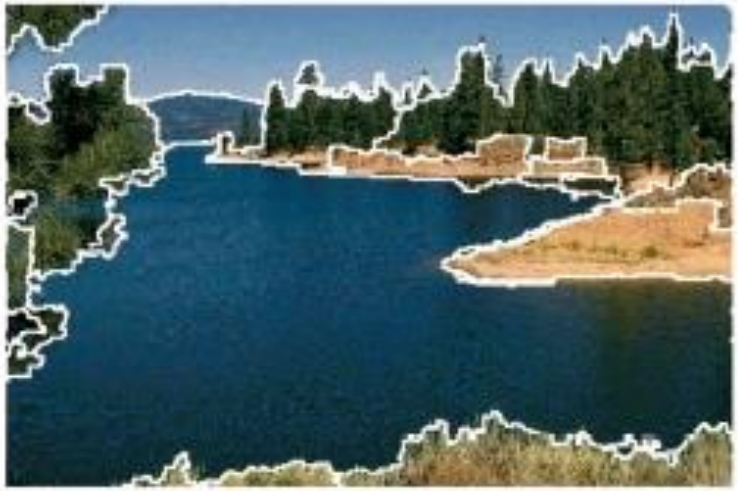
Segmentation

Example



Segmentation

Example



Segmentation

Example

Graph-Based Image Segmentation

- Represent tokens using a weighted graph.
  - affinity matrix
- Cut up this graph to get subgraphs with strong interior links
- Normalized Cuts

Normalized-Cuts Algorithm

Basic idea:

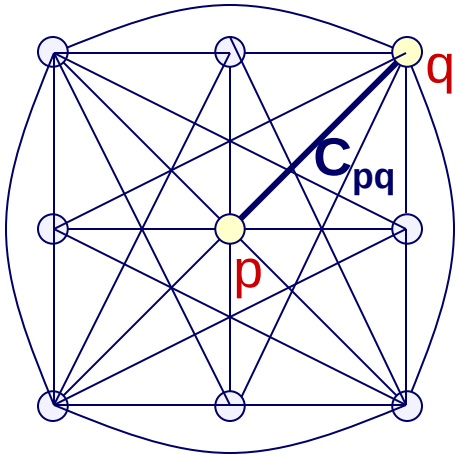
Propose a new graph-theoretic criterion for measuring the goodness of an image partition

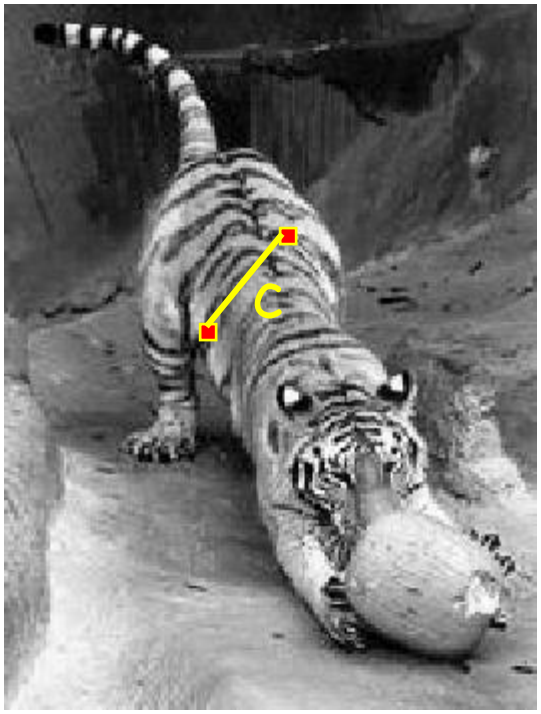
The minimization of this criterion can be formulated/approximated as a generalized eigenvalue problem.

Two questions:

1. What is a precise criterion for a good partition?
2. How can such a partition be computed efficiently?

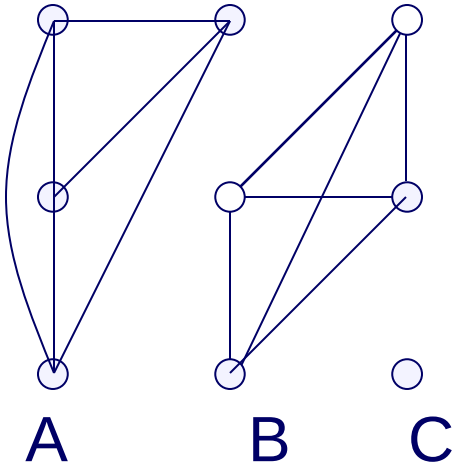
Images as graphs



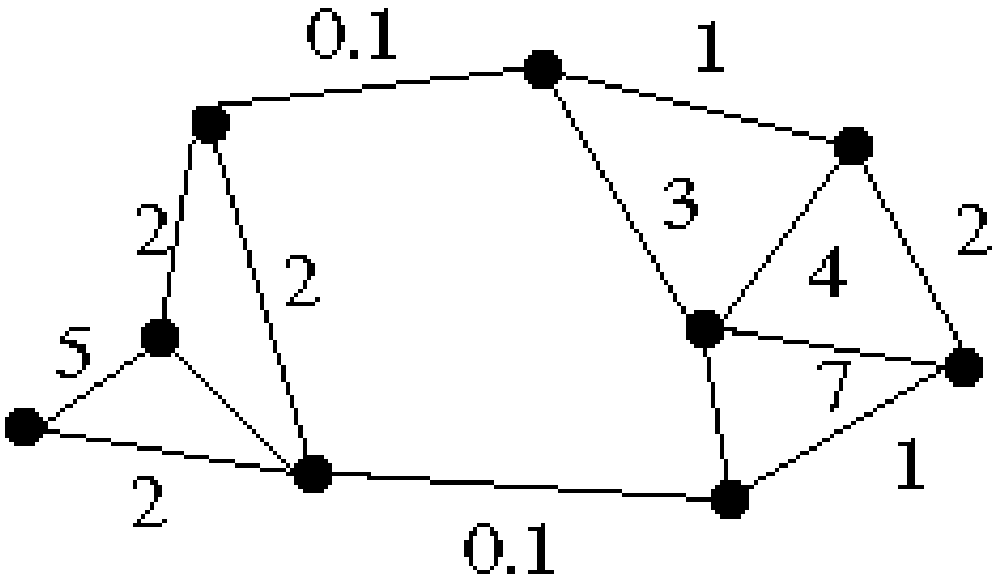


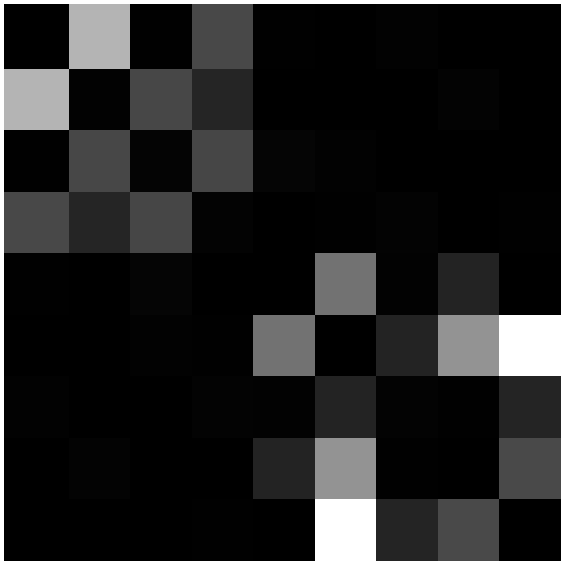
- *Fully-connected* graph
  - node for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - cost $\mathbf{c_{pq}}$ for each link
    - $\mathbf{c_{pq}}$ measures *dissimilarity*
      - dissimilarity: difference in color and position

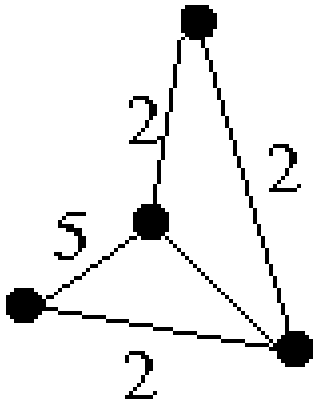
Segmentation by Graph Cuts

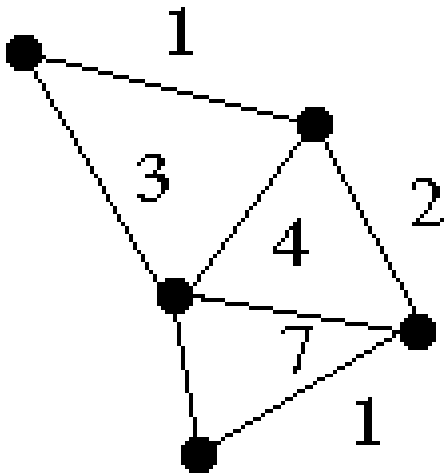


- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have high cost
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments


0.1
1
2
2
3
4
2
5
7
2
1
0.1




2
2
5
2


1
3
2
4
7
1

Measuring Affinity

Intensity

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

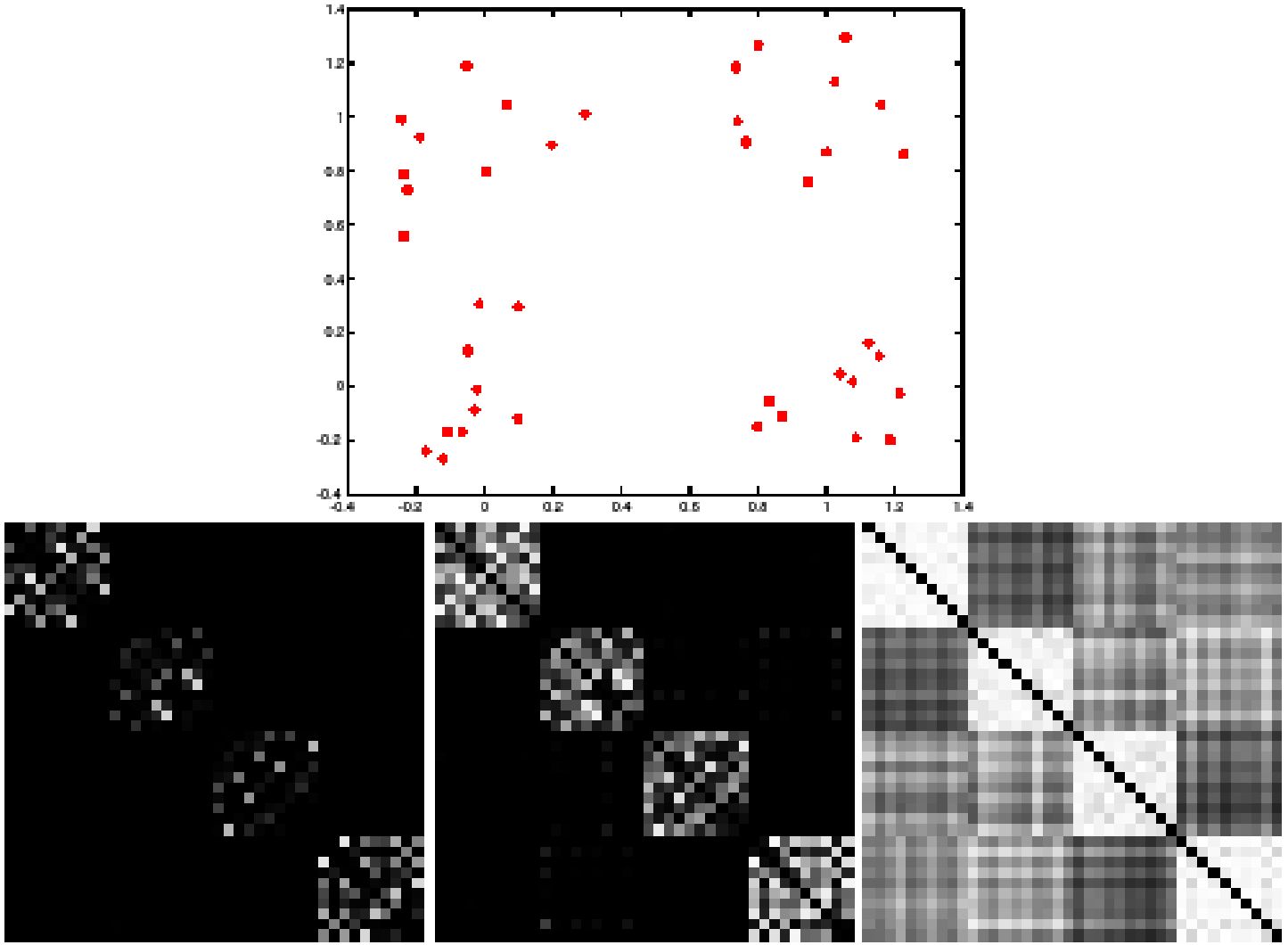
Distance

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

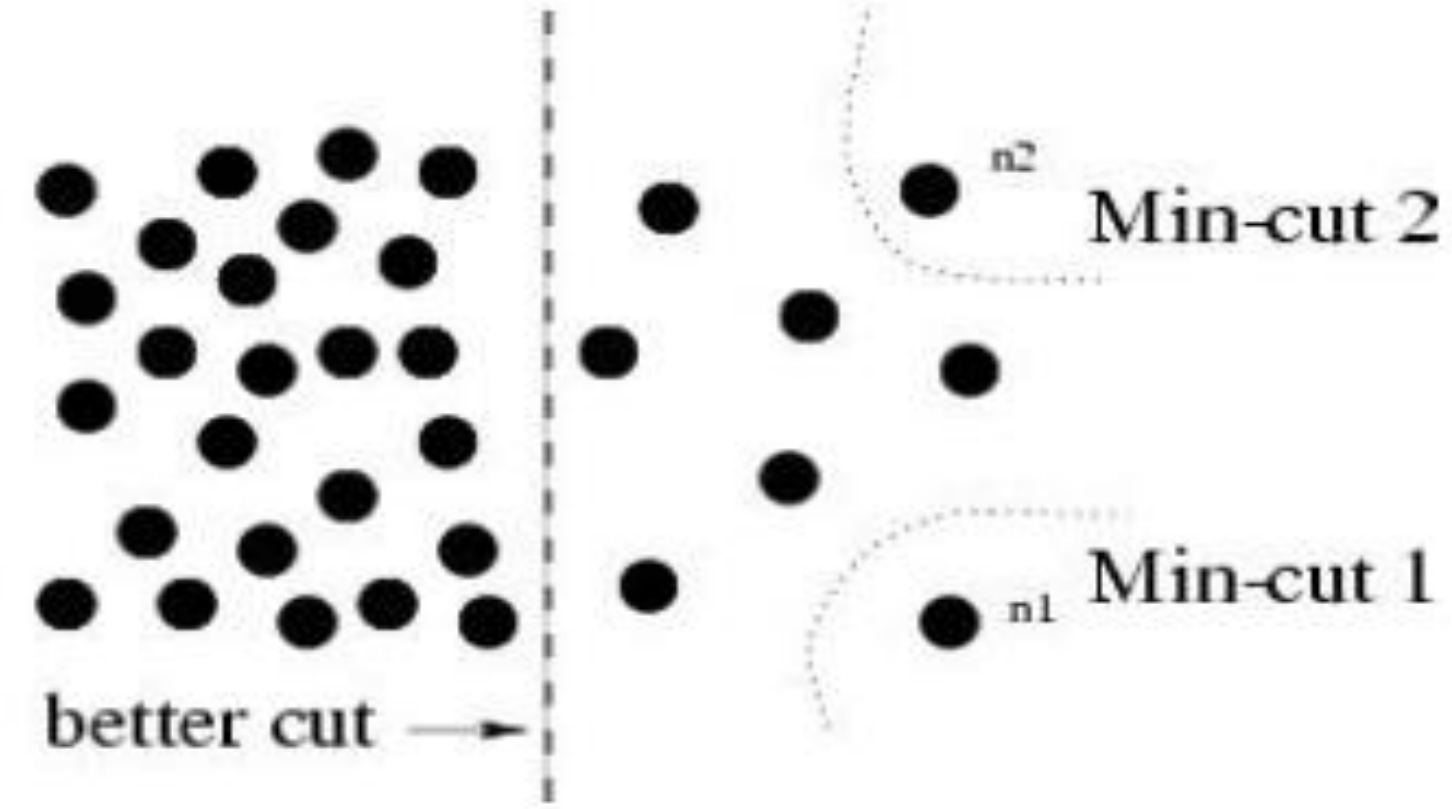
Texture

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

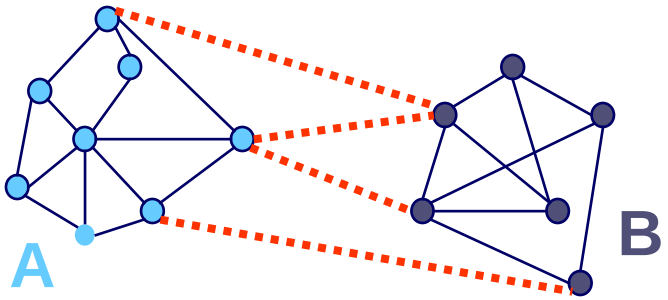
Scale affects affinity



Problem with Minimum Cut



Cuts in a graph



- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut: $cut(A,B) = \sum_{p \in A, q \in B} c_{p,q}$
- Normalized Cut
  - a cut penalizes large segments
  - fix by normalizing for size of segments

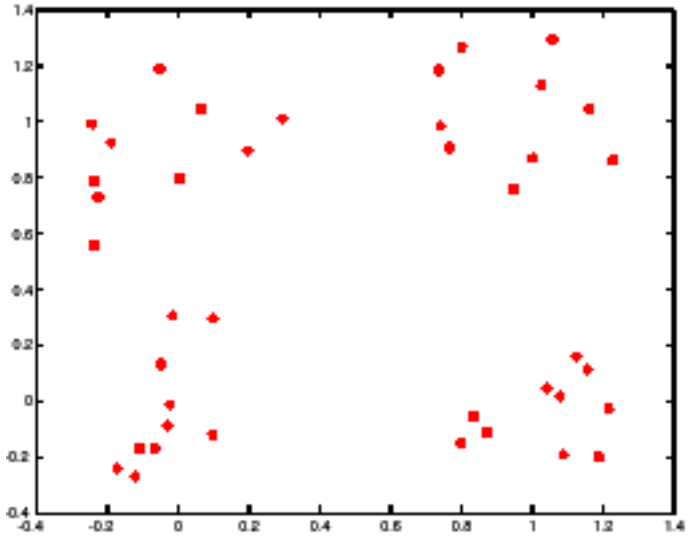
$$Ncut(A,B) = cut(A,B)\left[\frac{1}{\sum_{p \in A} c_{p,q}} + \frac{1}{\sum_{q \in B} c_{p,q}}\right]$$

Volume(A) = sum of costs of all edges that touch A

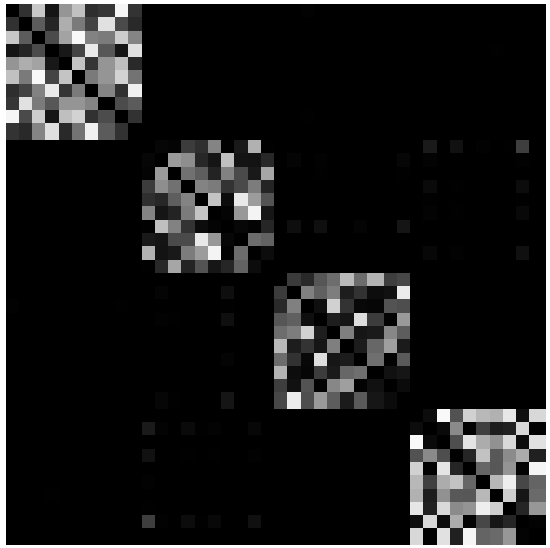
Eigenvectors and cuts

- Simplest idea: we want a vector a 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 $a^T A a$
- But need the constraint $a^T a = 1$
- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

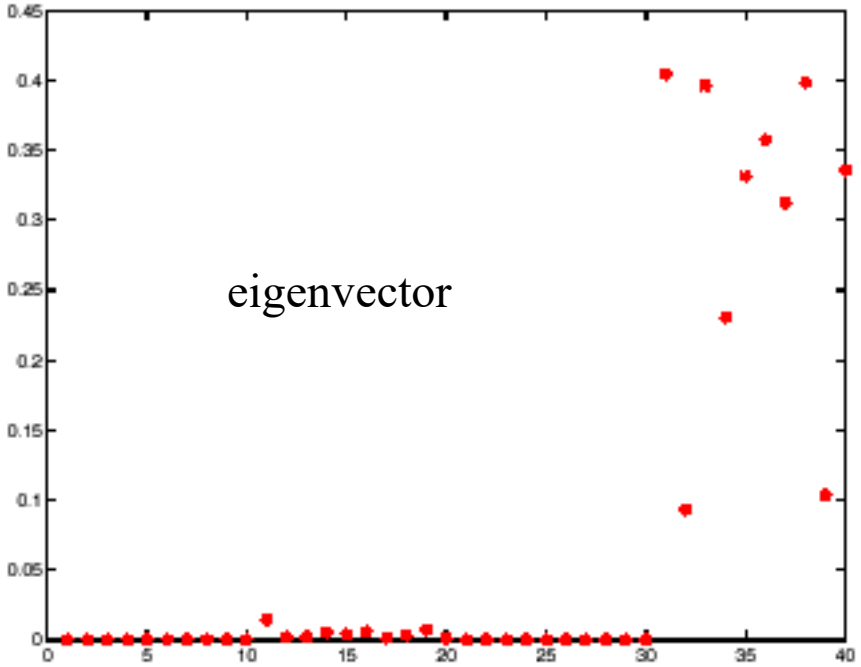
Example Eigenvector



points



matrix



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

Normalized Cuts

- Current criterion evaluates within cluster similarity, but not across cluster difference
- Instead, we'd like to maximize the within-cluster similarity compared to the across cluster difference
- Write graph as V, one cluster as A and the other as B
- Maximize $$\left(\frac{assoc(A,A)}{assoc(A,V)}\right)+\left(\frac{assoc(B,B)}{assoc(B,V)}\right)$$
- i.e. construct A & B such that their within cluster similarity is high compared to their association with the rest of the graph

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

Normalized cuts

- Write a vector y whose elements are 1 if item is in A, -b if it's in B
- Write the matrix of the graph as W, and the matrix which has the row sums of W on its diagonal as D, 1 is the vector with all ones.
- Criterion becomes $$\min_y \left(\frac{y^T (D - W) y}{y^T D y} \right)$$
- and we have a constraint $$y^T D 1 = 0$$
- This is hard to do, because y's values are quantized

Normalized cuts

- Instead, solve the generalized eigenvalue problem

$$\min_y \left(y^T (D - W) y \right) \text{ subject to } \left(y^T D y = 1 \right)$$

- which gives

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

- Now 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

Normalize Cut in Matrix Form

$\mathbf{W}$ is the cost matrix : $\mathbf{W}(i, j) = w_{i,j}$;

$\mathbf{D}$ is the sum of costs from node i : $\mathbf{D}(i,i) = \sum_j \mathbf{W}(i, j)$;

After lots of math, we get:

$$Ncut(A,B) = \frac{\mathbf{y}^{\mathbf{T}}(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^{\mathbf{T}}\mathbf{D}\mathbf{y}}, \quad \text{with } \mathbf{y}_i \in \{1, -b\}, \mathbf{y}^T\mathbf{D}1 = 0.$$

- Solution given by "generalized" eigenvalue problem:

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

- Solved by converting to standard eigenvalue problem:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z}, \quad where\ \mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$$

- optimal solution corresponds to second smallest eigenvector
- for more details, see
  - J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR 1997

Recursive two-way Ncut grouping algorithm

1. Given an image or image sequence, set up a weighted graph **G=(V, E)** and set the weight on the edge connection two nodes to be a measure of the similarity between the two nodes.
2. Solve $(\mathbf{D}-\mathbf{W})\mathbf{x}=\lambda\mathbf{D}\mathbf{x}$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.

   Decide if the current partition should be subdivided by checking the stability of the cut, and make sure Ncut is below the pre-specified value

Simultaneous K-way cut

Use all of the top eigenvectors to simultaneously obtain a K-way partition.

1. A simple clustering method, such as k-means, is used to obtain an over-segmentation of the image into k' groups
2. (a) Greedy pruning: iteratively merge two segments as a time until only k segments are left, minimizing the k-way Ncut criterion

$$Ncut_k = \frac{cut(A_1, V - A_1)}{assoc(A_1, V)} + \frac{cut(A_2, V - A_2)}{assoc(A_2, V)} + \ldots + \frac{cut(A_k, V - A_k)}{assoc(A_k, V)}$$

(b) Global recursive cut

From the initial segments, build a condensed graph. Based on this graph, recursively bipartition according to Ncut criterion, either with the generalized eigenvector system or with exhaustive search in discrete domain.

Example: brightness image

1. Using the brightness value of the pixels and their spatial location as the weight:

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

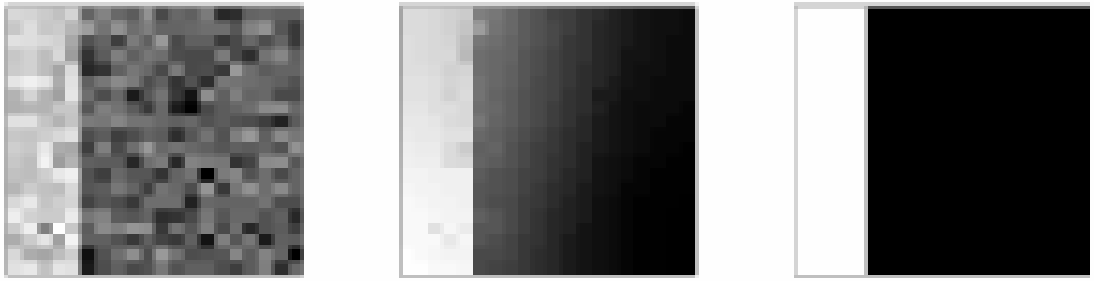
2. Can represent different type of segmentation

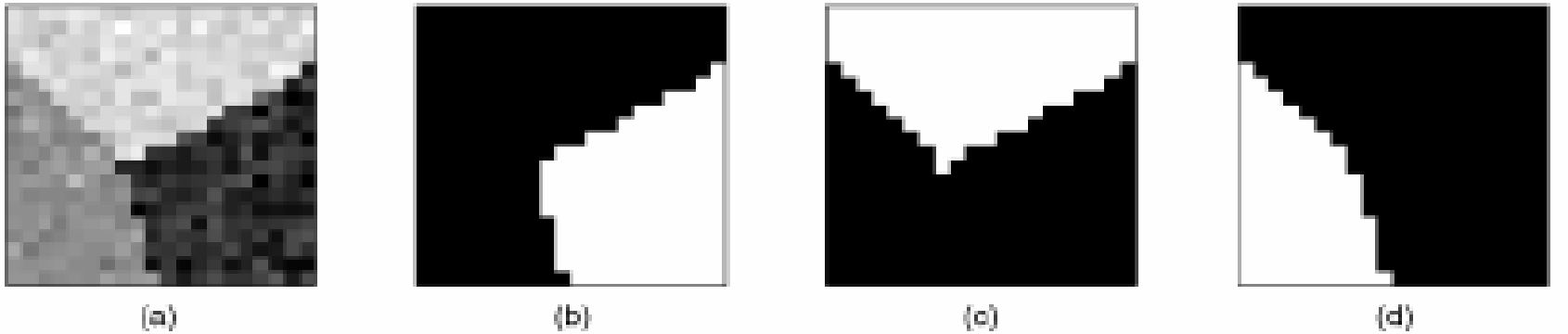
3. Solve for the eigenvectors for the second smallest eigenvector:

$$D^{\frac{1}{2}}(D-W)D^{\frac{1}{2}}x = \lambda x$$

4. Somehow choose a splitting point to discretize the eigenvector to get the bipartition

Experiments on Synthetic Images





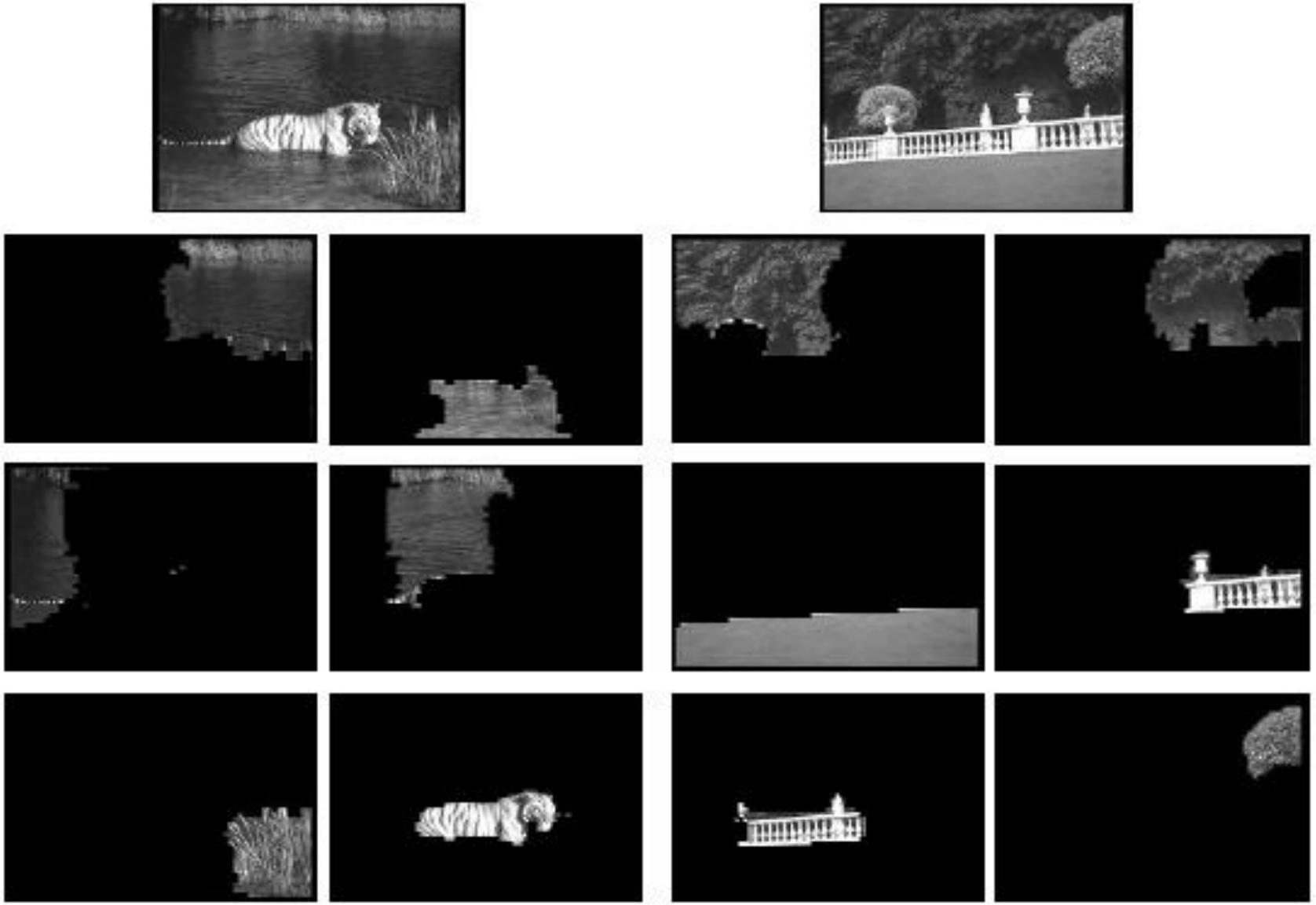
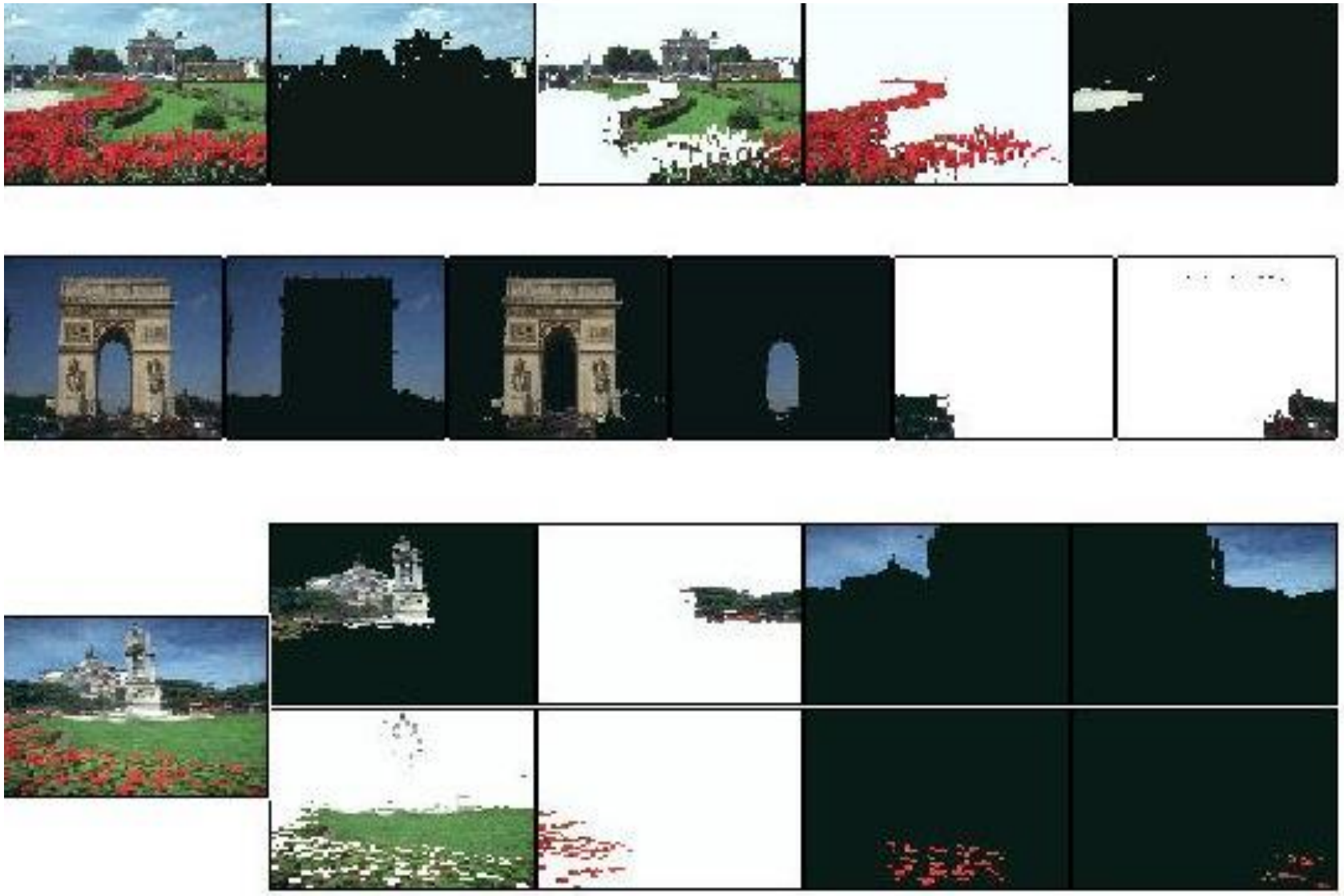


Figure from “Image and video segmentation: the normalised cut framework”, by Shi and Malik, copyright IEEE, 1998

Color Image Segmentation



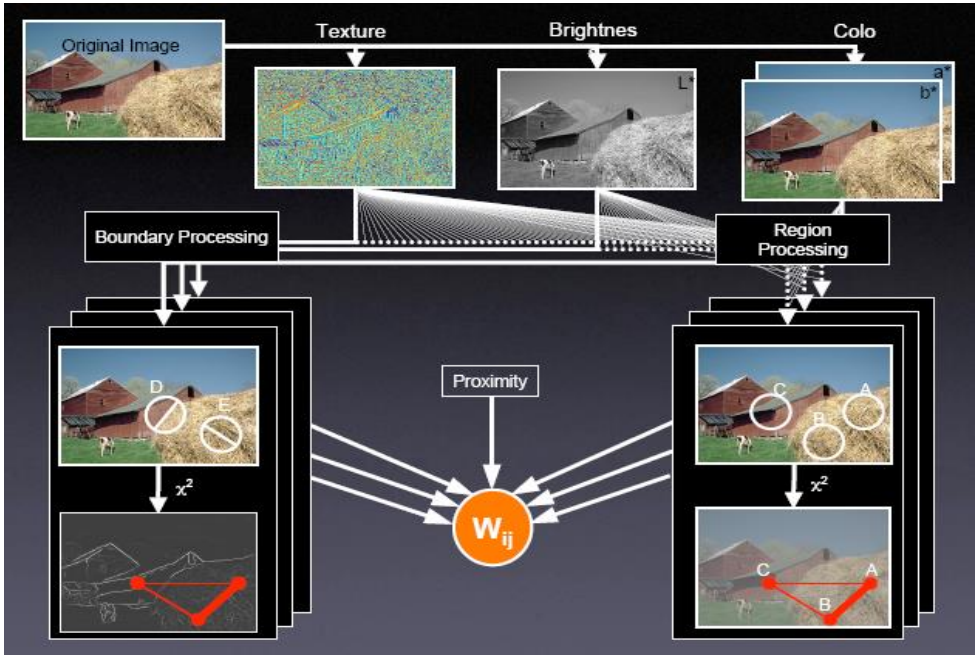
N-Cuts Segmentation

N-Cuts can also handle:

- Texture segmentation
- Motion segmentation

but it needs appropriate similarity definition

General problem of defining feature similarity incorporating many cues/features:



Summary for Normalized Cut Segmentation

- Normalized cut presents a new optimality criterion for partitioning a graph into clusters.
- Ncut is normalized measure of disassociation and minimizing it is equivalent to maximizing association.
- We solve an approximate version of the MinNcut problem by converting it into a generalized eigenvector problem.
- Defining an appropriate similarity measure is very crucial for the graph-based image segmentation.

Conclusion

- Image segmentation is a mid-level vision problem which is required in many application systems.
- K-means
- Mean-Shift
- N-Cuts
- Image segmentation, clustering, unsupervised learning
- Semantic image segmentation
  - Deep learning