Unit 5 :
Image Segmentation

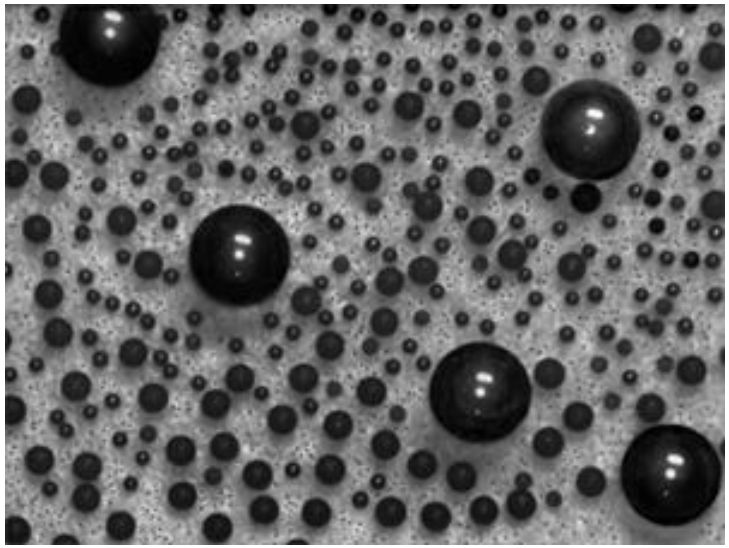
K-means Clustering

Mean Shift Segmentation

Active Appearance Model

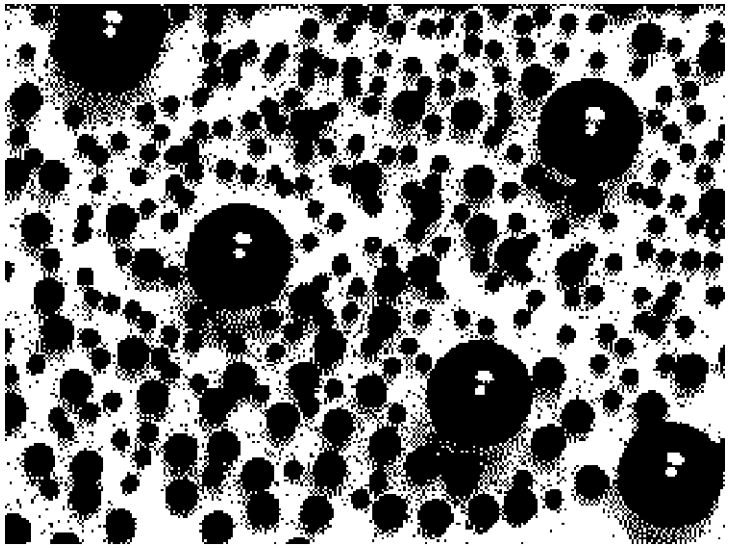
Histogram-based segmentation

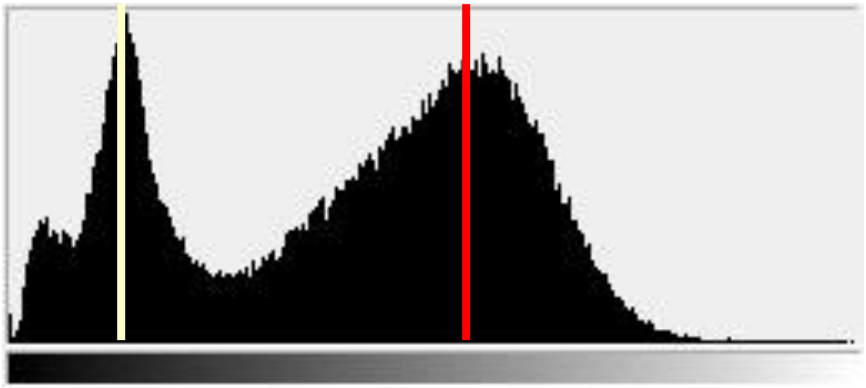
- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color



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

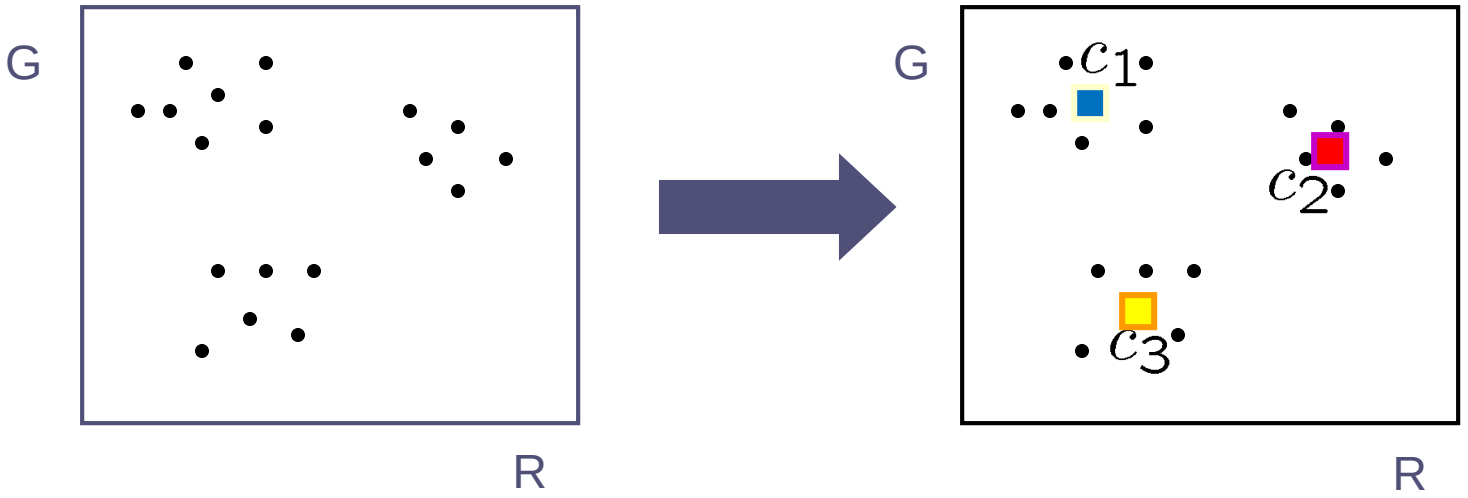




Here's what it looks like if we use two colors

Clustering

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



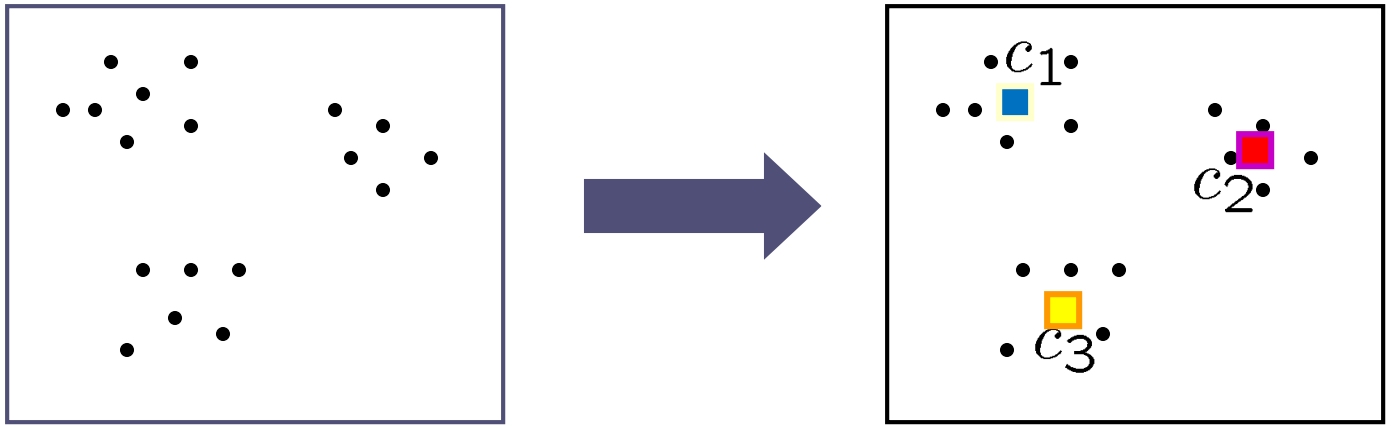
Objective

- 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?

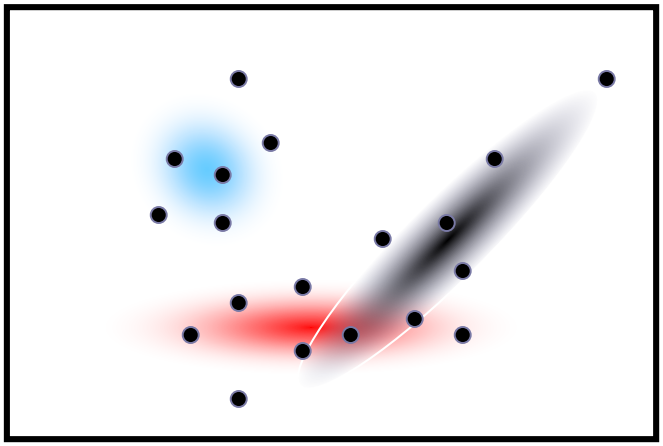
1. Randomly choose first center.
2. Pick new center with prob. proportional to: $\|p - c_i\|^2$
   (contribution of *p* to total error)
3. Repeat until *k* centers.

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



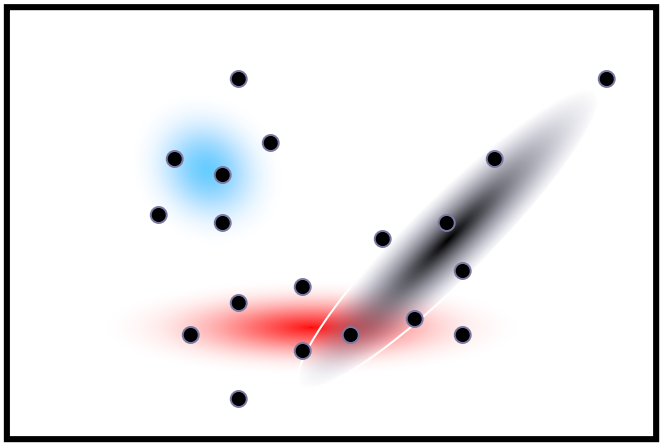
- One generative model is a mixture of Gaussians (MOG)
  - K Gaussian blobs with means $\boldsymbol{\mu_b}$ covariance matrices $\mathbf{V_b}$, dimension d
    - blob *b* defined by $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - blob *b* is selected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

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

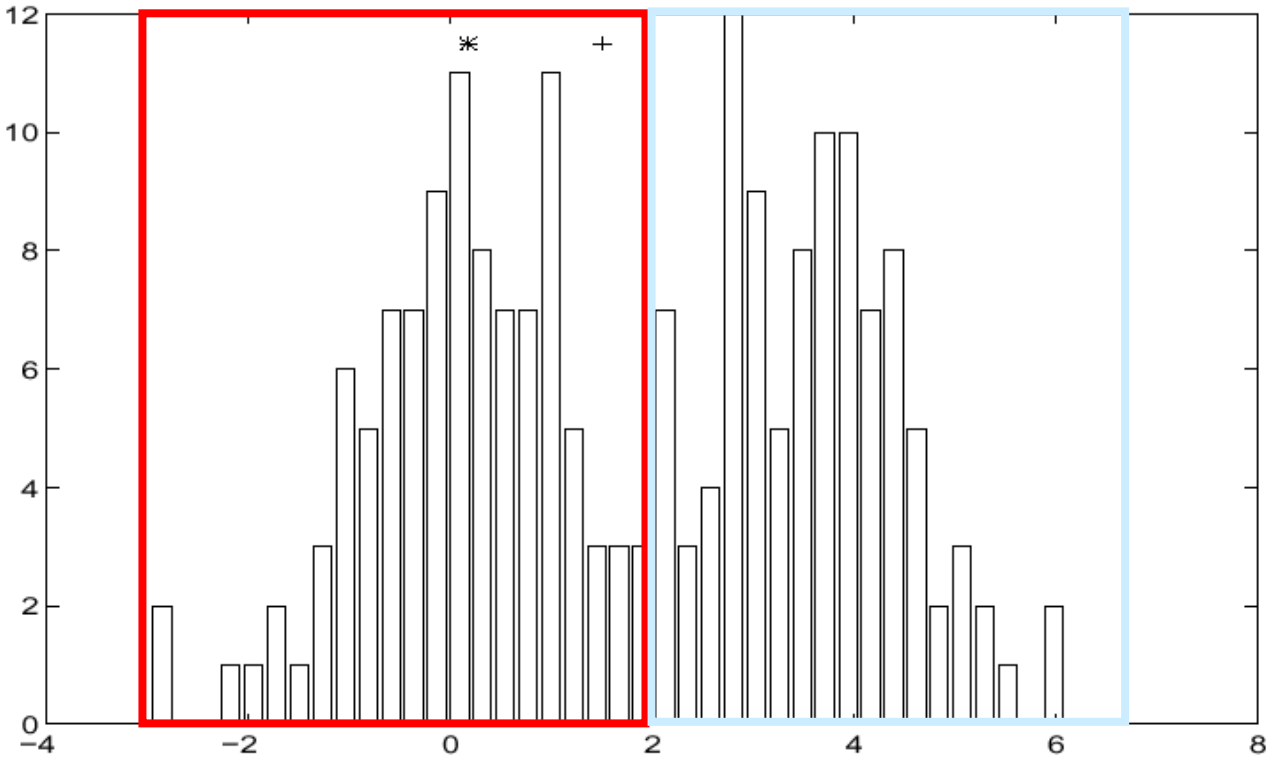
  - covariance of blob b

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

Applications of EM

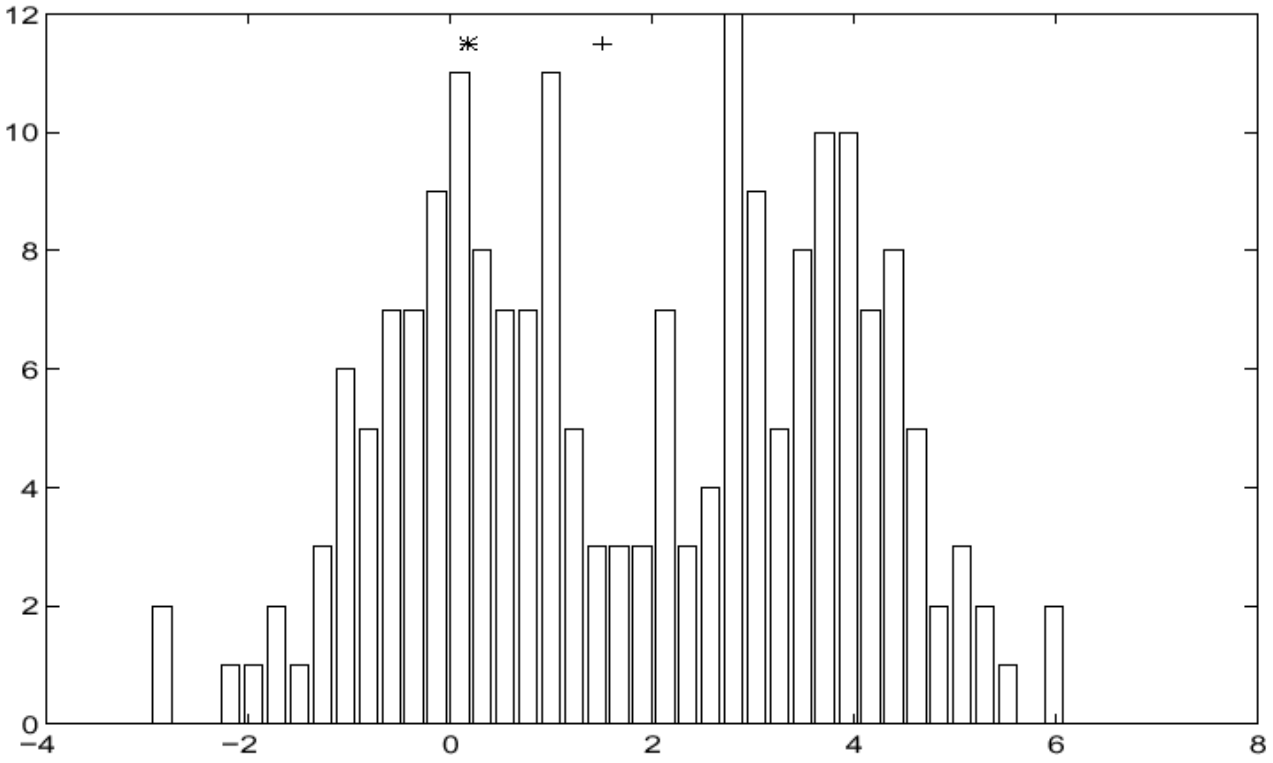
- Turns out this is useful for all sorts of problems
  - any clustering problem
  - any model estimation problem
  - missing data problems
  - finding outliers
  - segmentation problems
    - segmentation based on color
    - segmentation based on motion
    - foreground/background separation
  - ...

Finding Modes in a Histogram



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

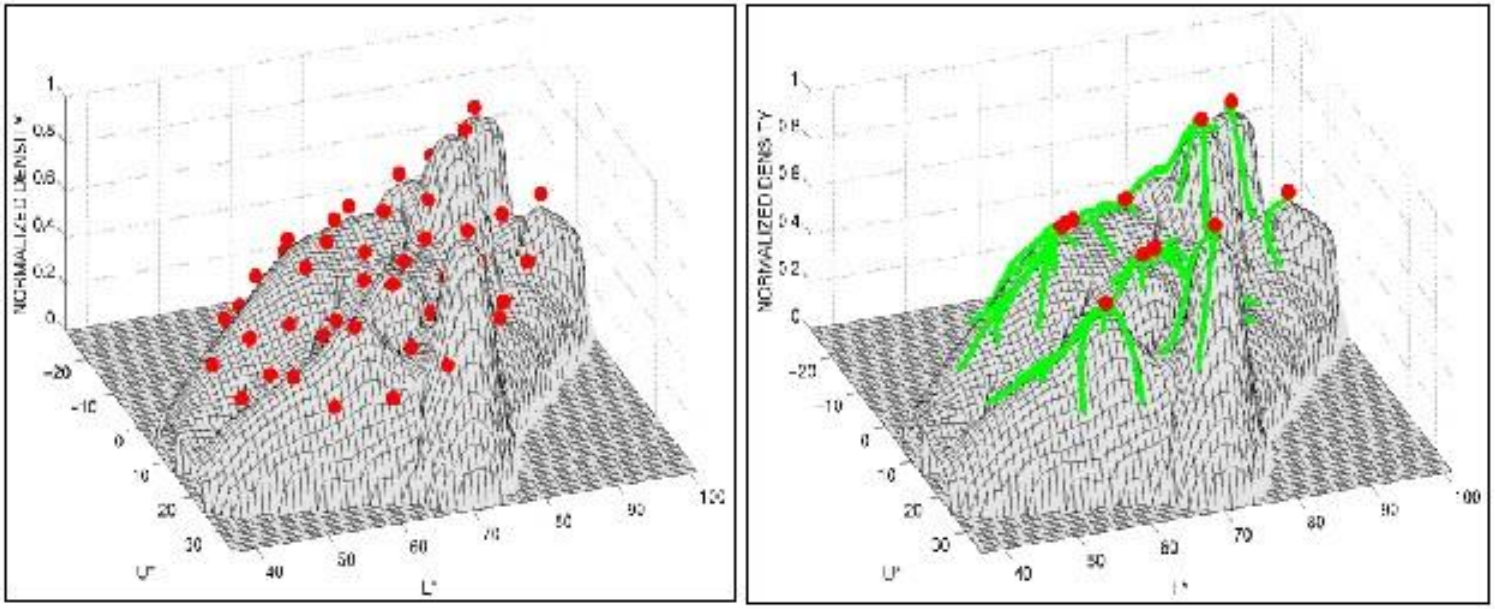
Mean Shift [Comaniciu & Meer]



- 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

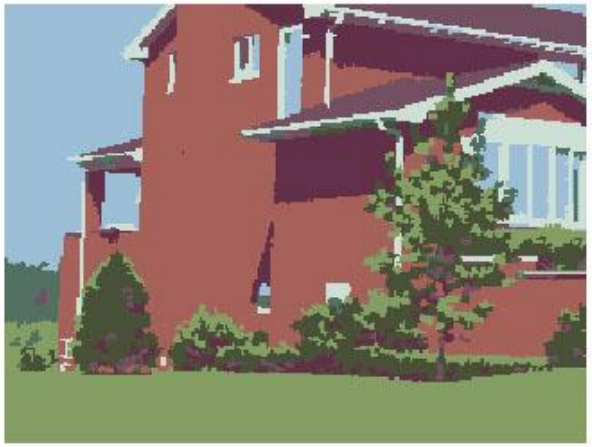
- Approach
  - Initialize a window around each point
  - See where it shifts—this determines which segment it's in
  - Multiple points will shift to the same segment

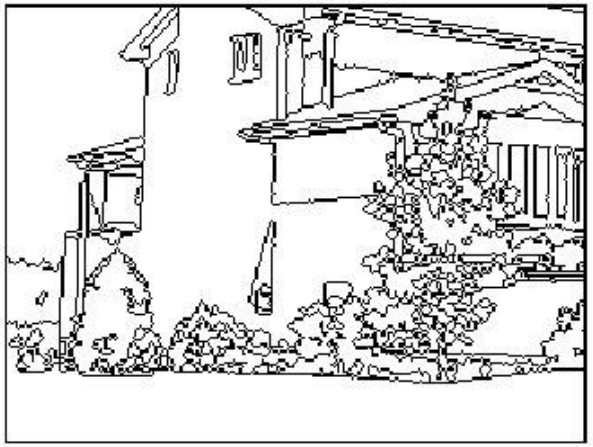


Mean shift trajectories

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



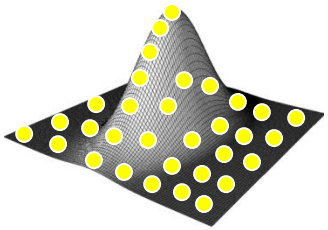


http://www.caip.rutgers.edu/~comanici/segm_images.html

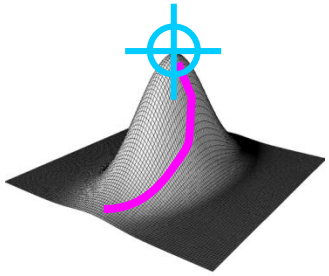
What is Mean Shift ?

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

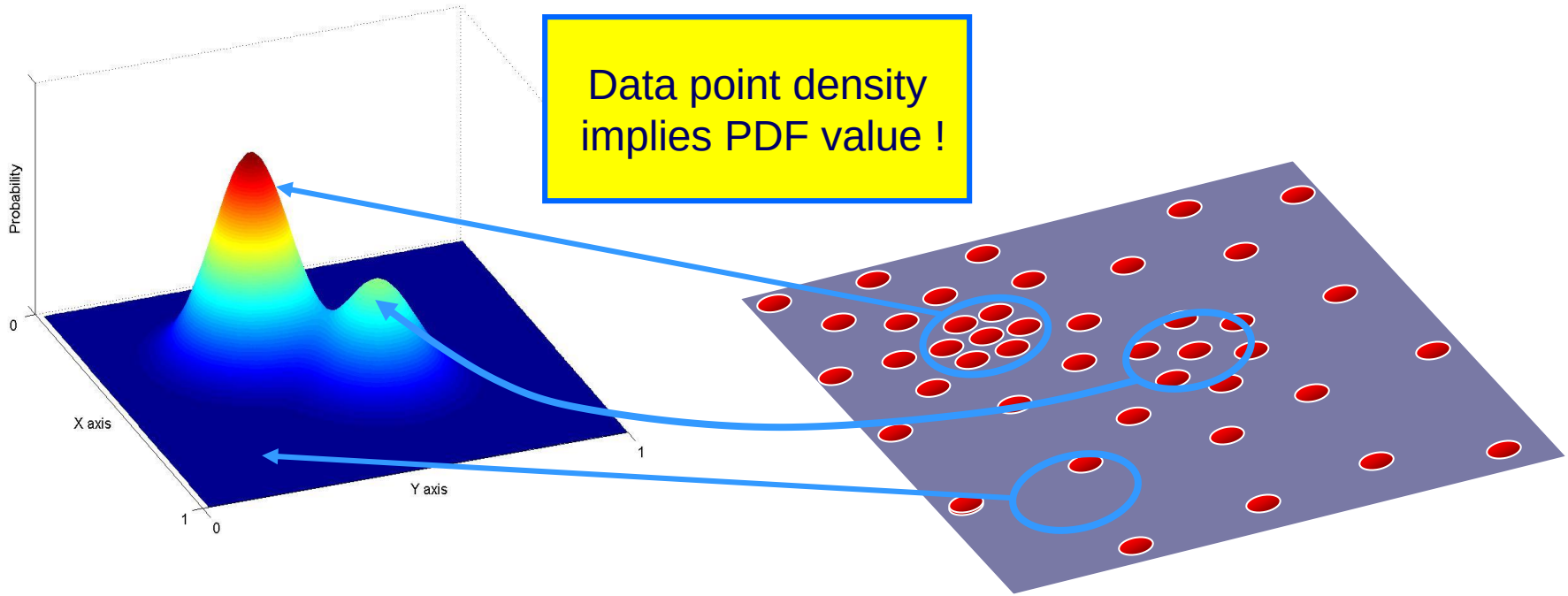


Non-parametric
Density **GRADIENT** Estimation
(Mean Shift)



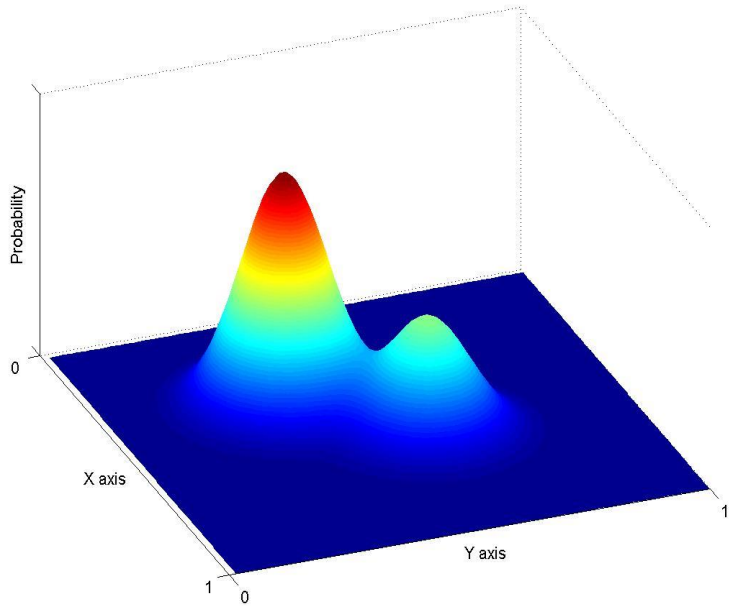
Non-Parametric Density Estimation

Assumption : The data points are sampled from an underlying PDF

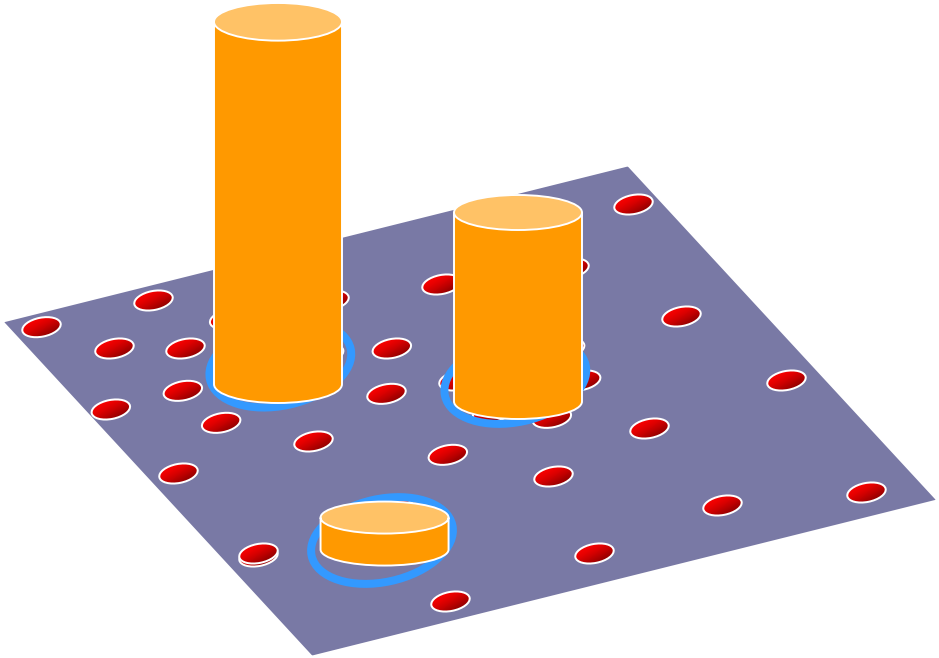


Assumed Underlying PDF **Real Data Samples**

Non-Parametric Density Estimation

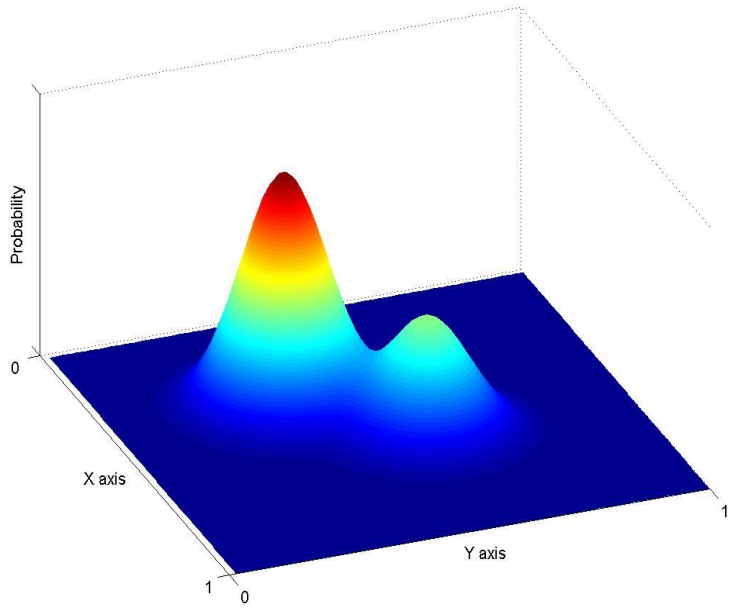


Assumed Underlying PDF

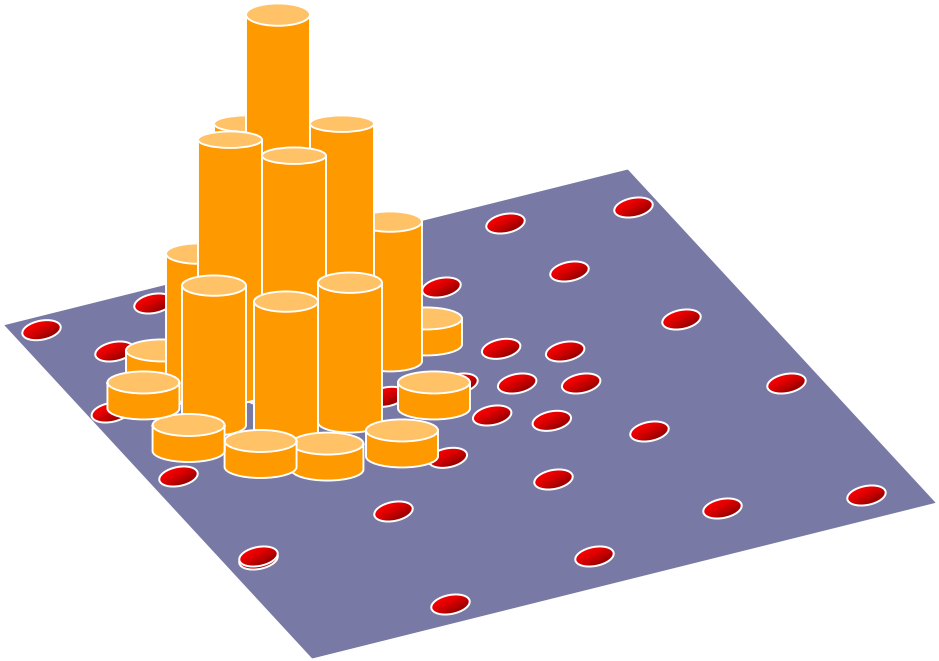


Real Data Samples

Non-Parametric Density Estimation



Assumed Underlying PDF

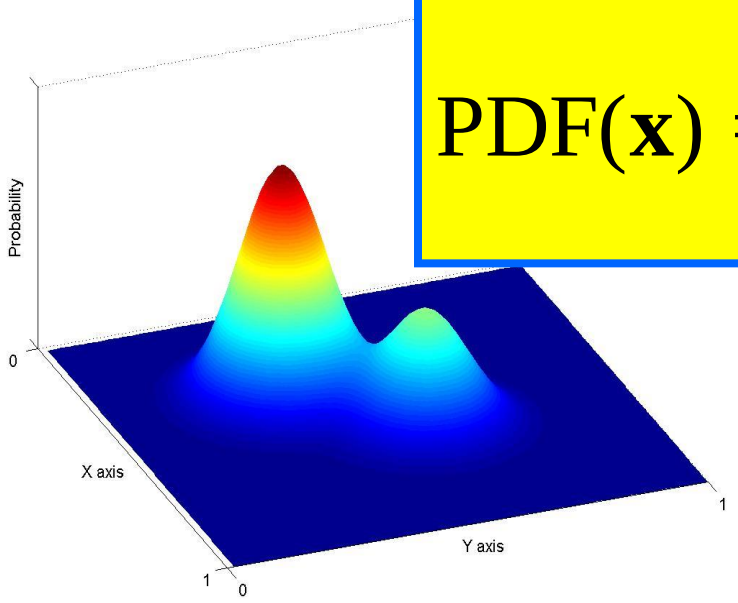


Real Data Samples

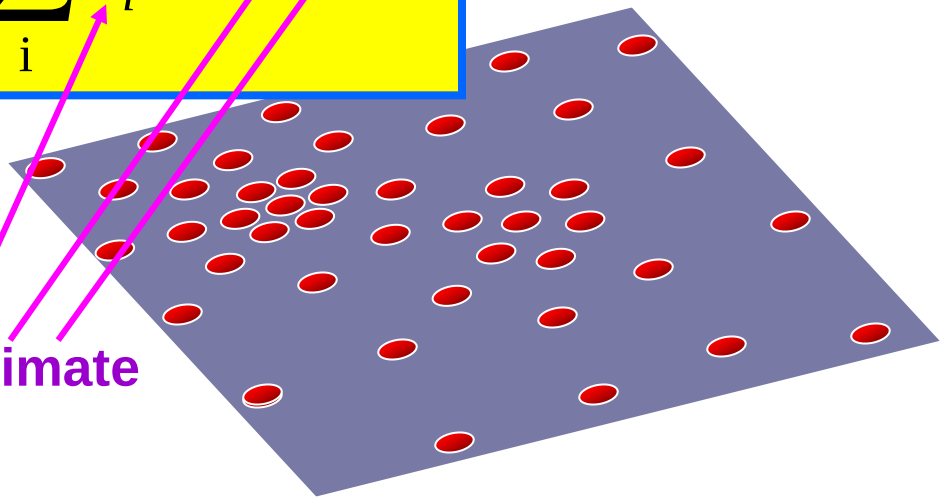
Parametric Density Estimation

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

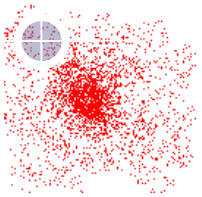
Real Data Samples

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:

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

Same function on each dimension

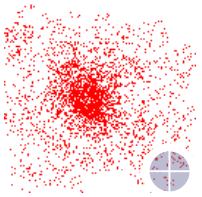
Function of vector length only

Kernel Density Estimation

Various Kernels

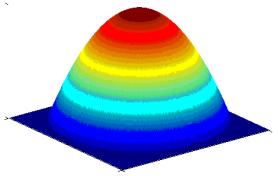
$$P(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)$$

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

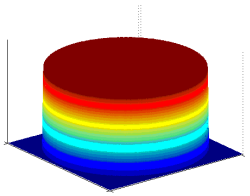


Examples:

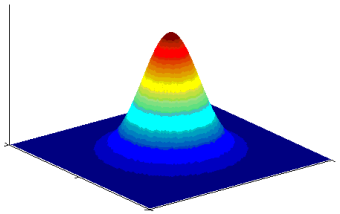
- Epanechnikov Kernel $K_E(\mathbf{x}) = \begin{cases} c\left(1 - \|\mathbf{x}\|^2\right) & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$



- 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 :

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

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

Computing The Mean Shift

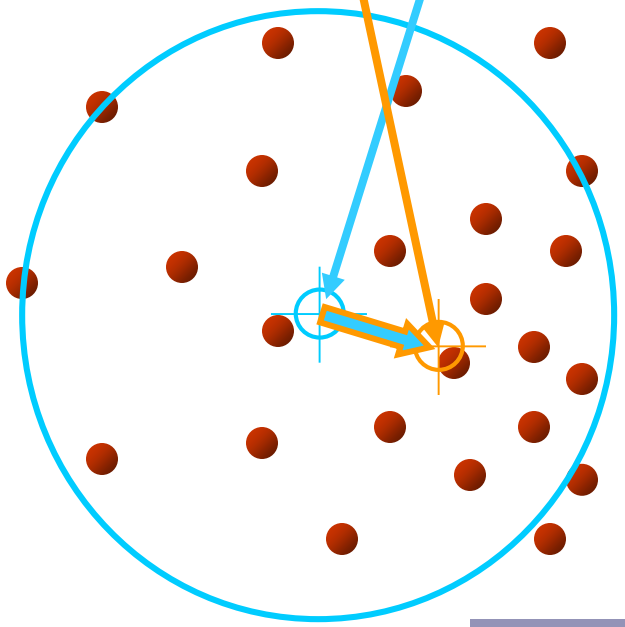
$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{c}{n}\left[\sum_{i=1}^{n} g_i\right]\mathbf{g}\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_i}{\sum_{i=1}^{n} g_i} - \mathbf{x}\right]$$

Yet another Kernel density estimation !

Simple Mean Shift procedure:

- Compute mean shift vector

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



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

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

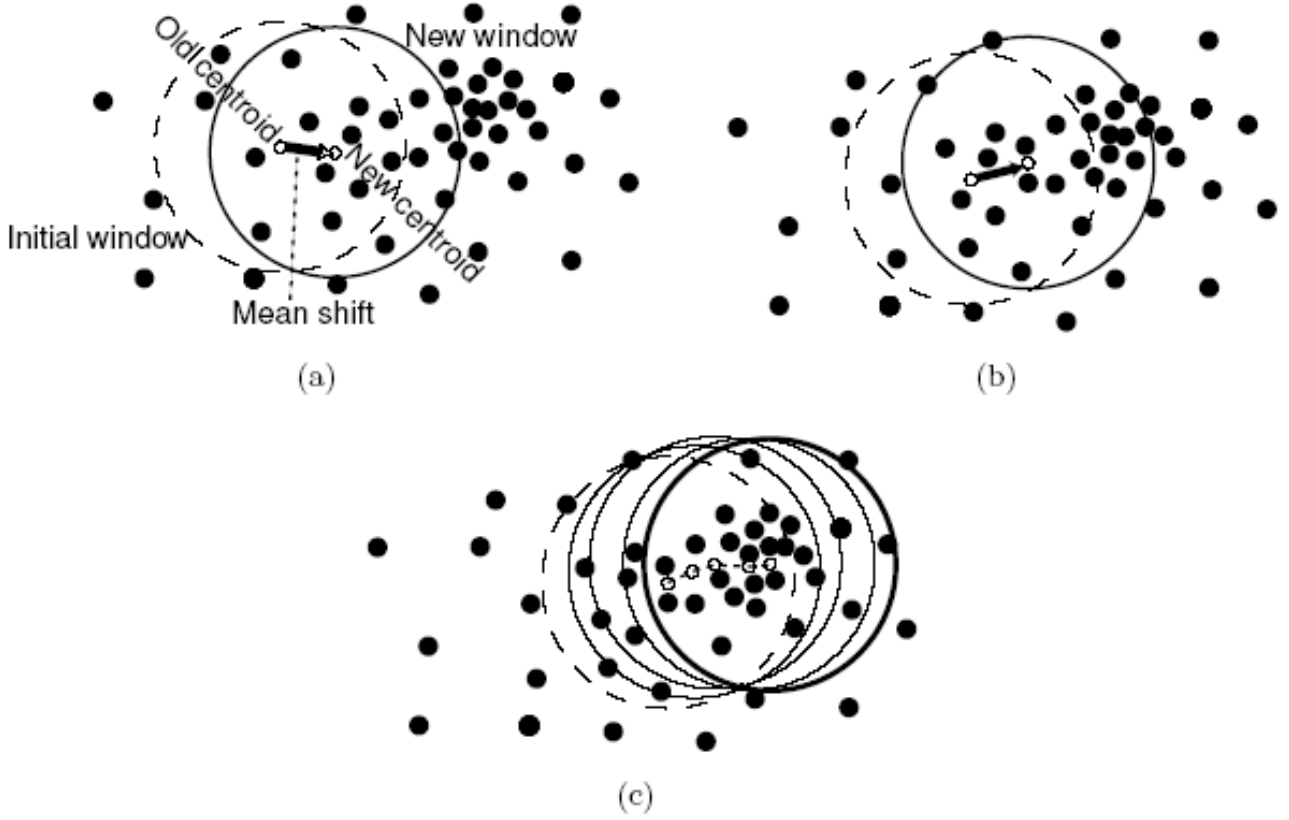
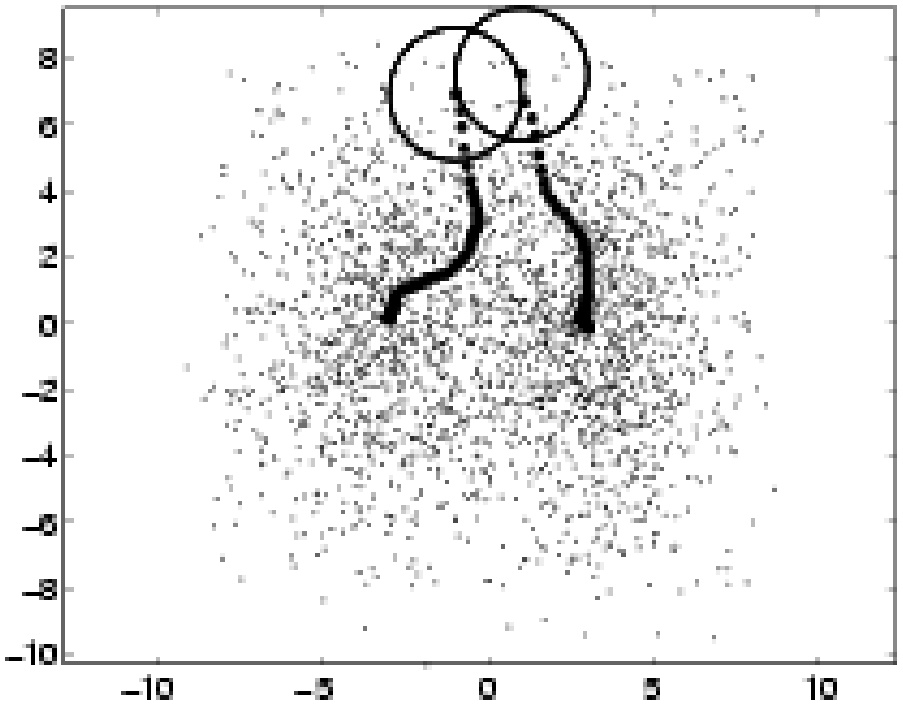


Figure 7.1: Principle of the mean shift procedure. The most dense region of data is identified in an iterative process. (a) The initial region of interest is randomly positioned over data and its centroid is determined. The new region is moved to the location of the identified centroid. The vector determining 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

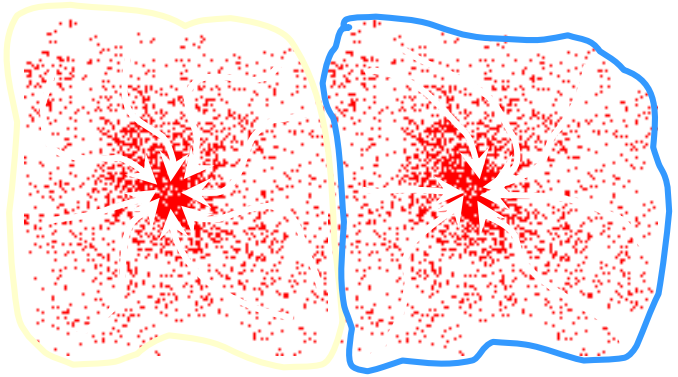
Mean Shift Applications

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

Clustering

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

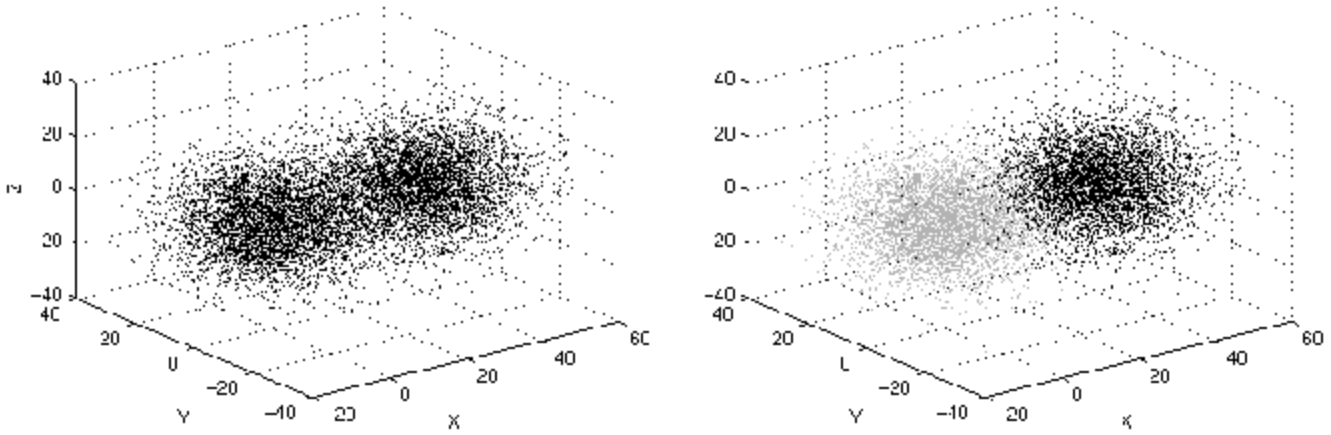
Attraction basin : the region for which all trajectories lead to the same mode



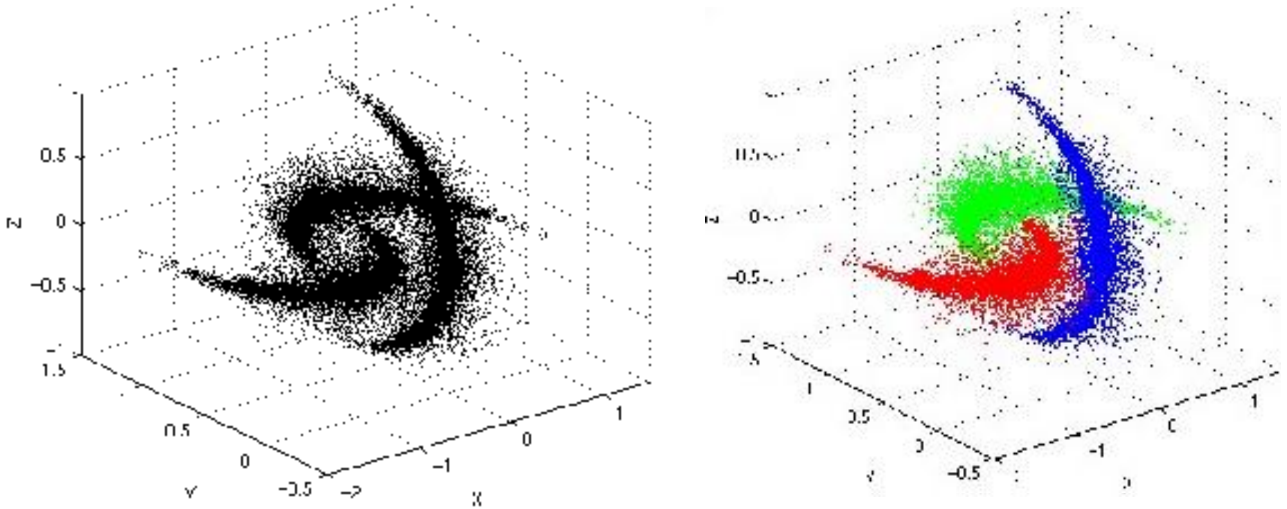
Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer

Clustering

Synthetic Examples



Simple Modal Structures



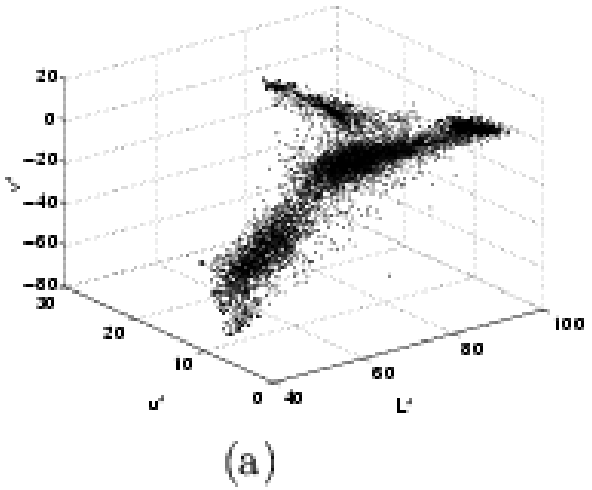
Complex Modal Structures

Clustering

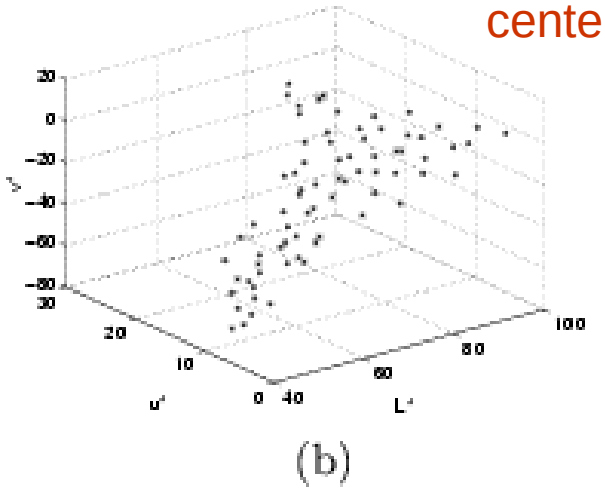
Real Example

Feature space:
L*u*v representation

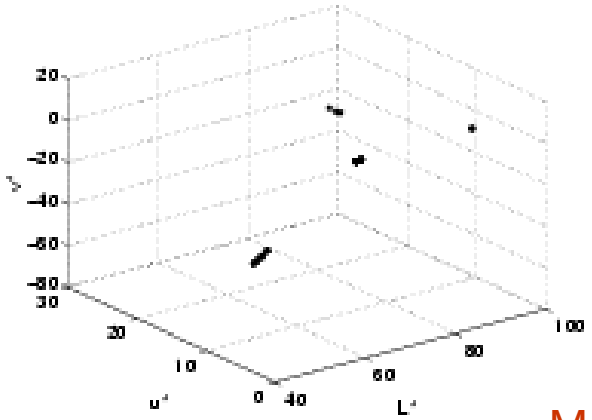
Initial window centers



(a)

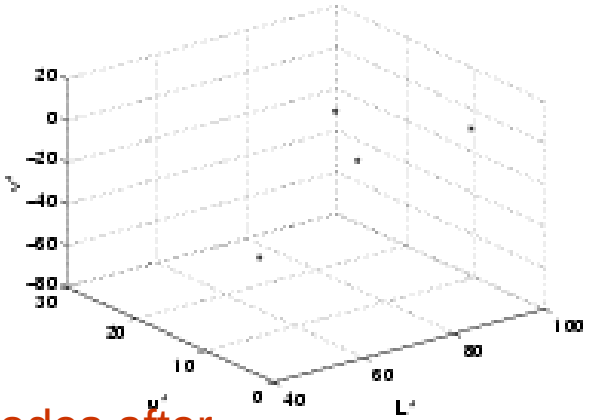


(b)



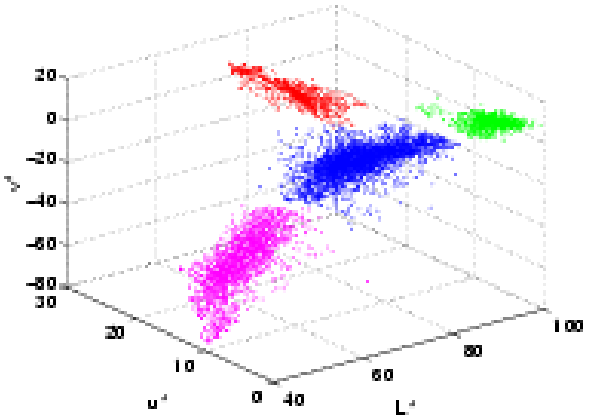
Modes found

(c)



Modes after pruning

(d)

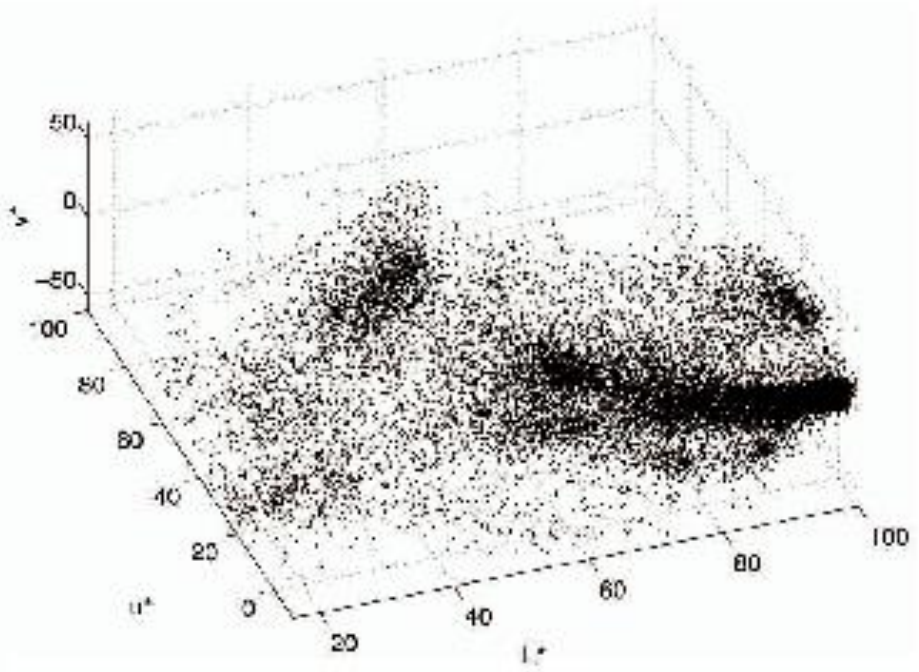


Final clusters

(e)

Clustering

Real Example

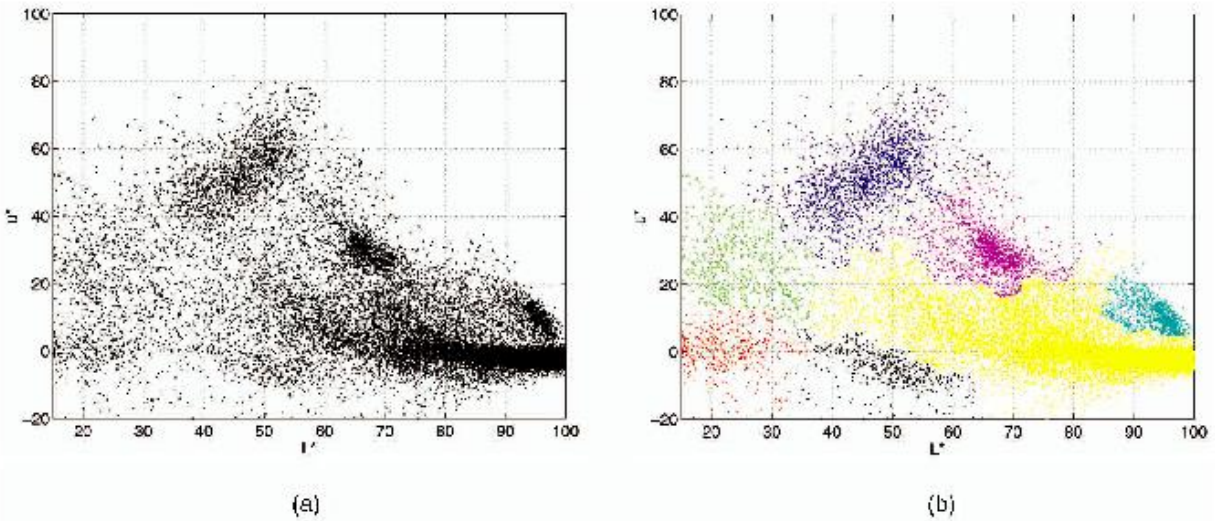


L*u*v space representation

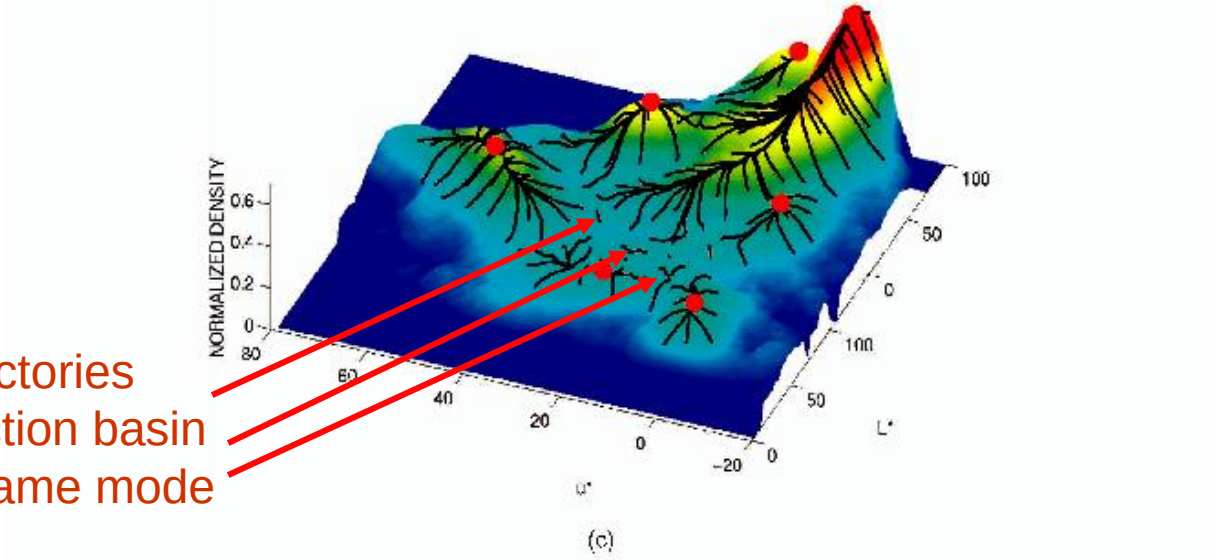
Clustering

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

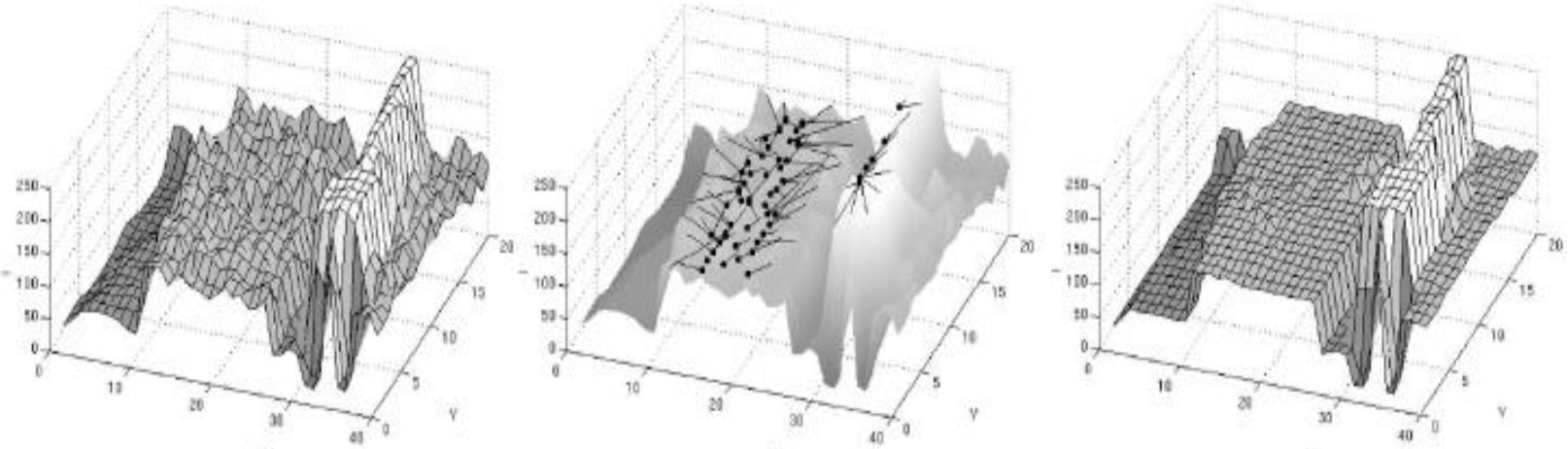
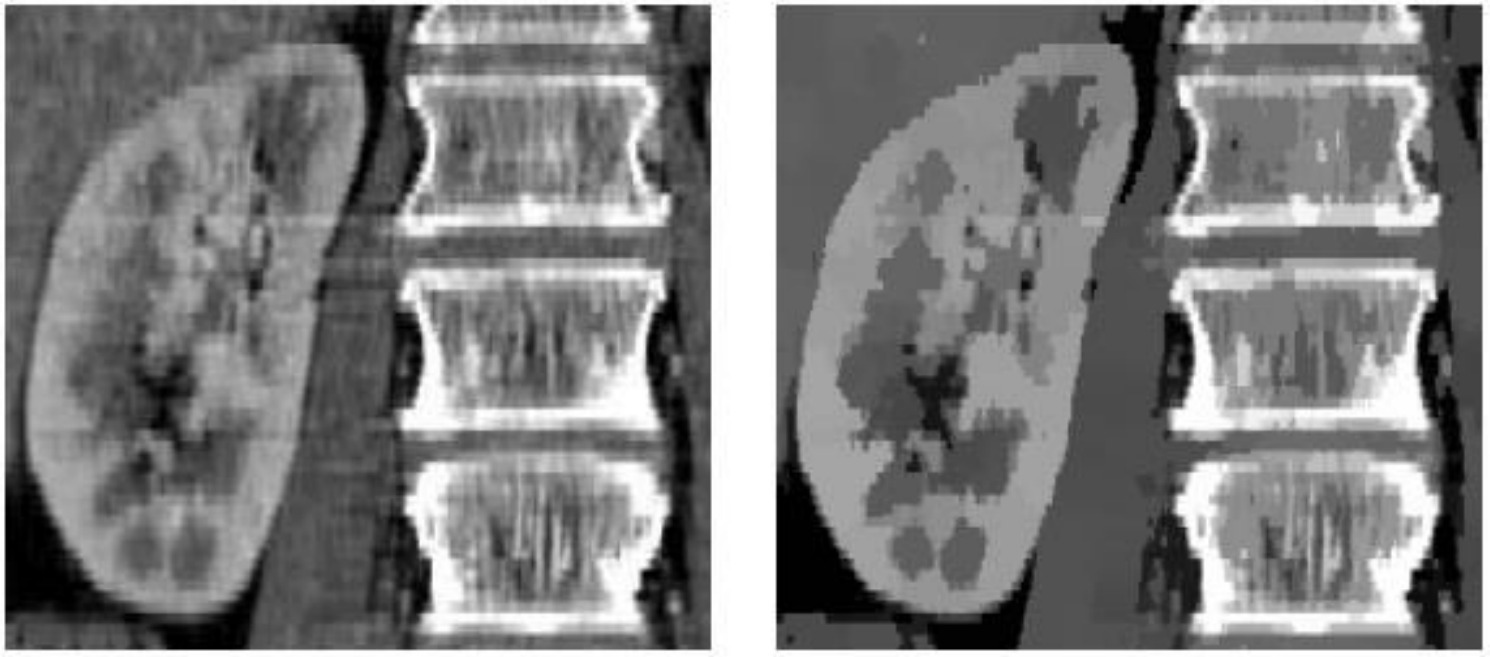


Image Data (slice)

Mean Shift vectors

Smoothing result

Mean-shift Filtering Example

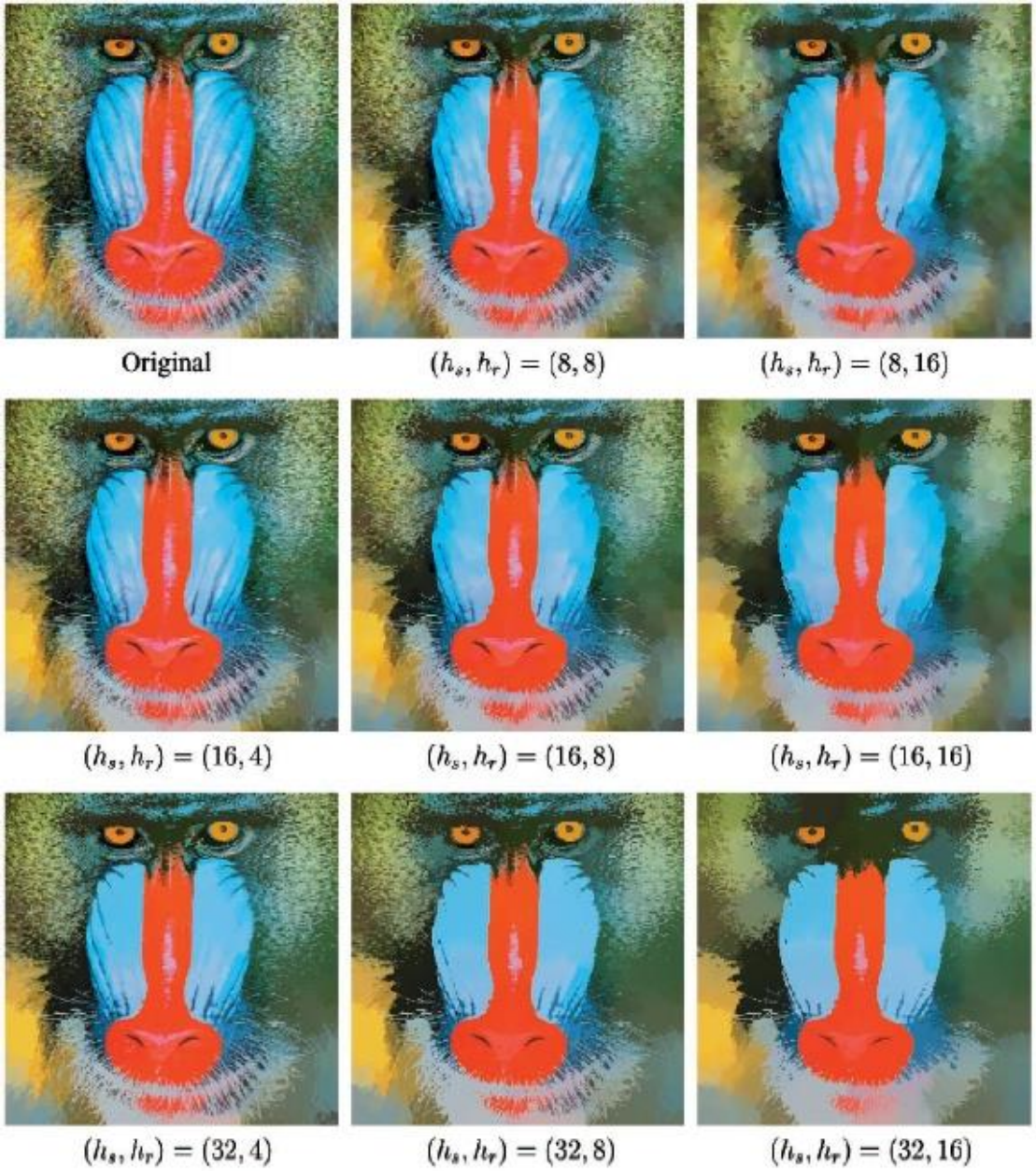


(a) (b)

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

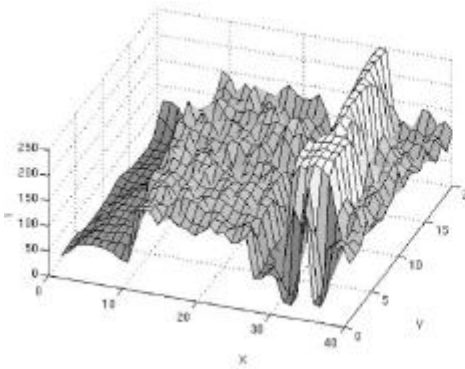
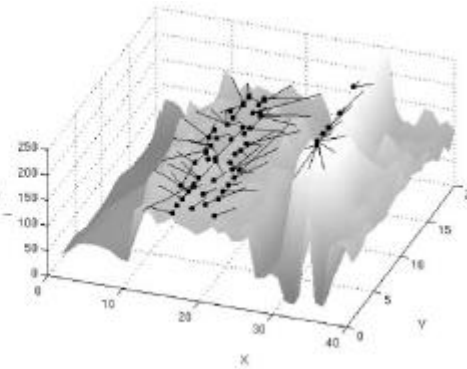
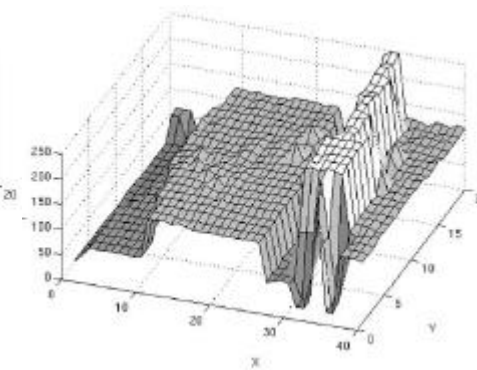


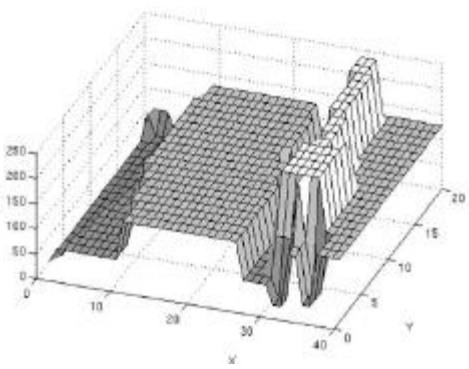
Image Data (slice)



Mean Shift vectors



Smoothing result



Segmentation result

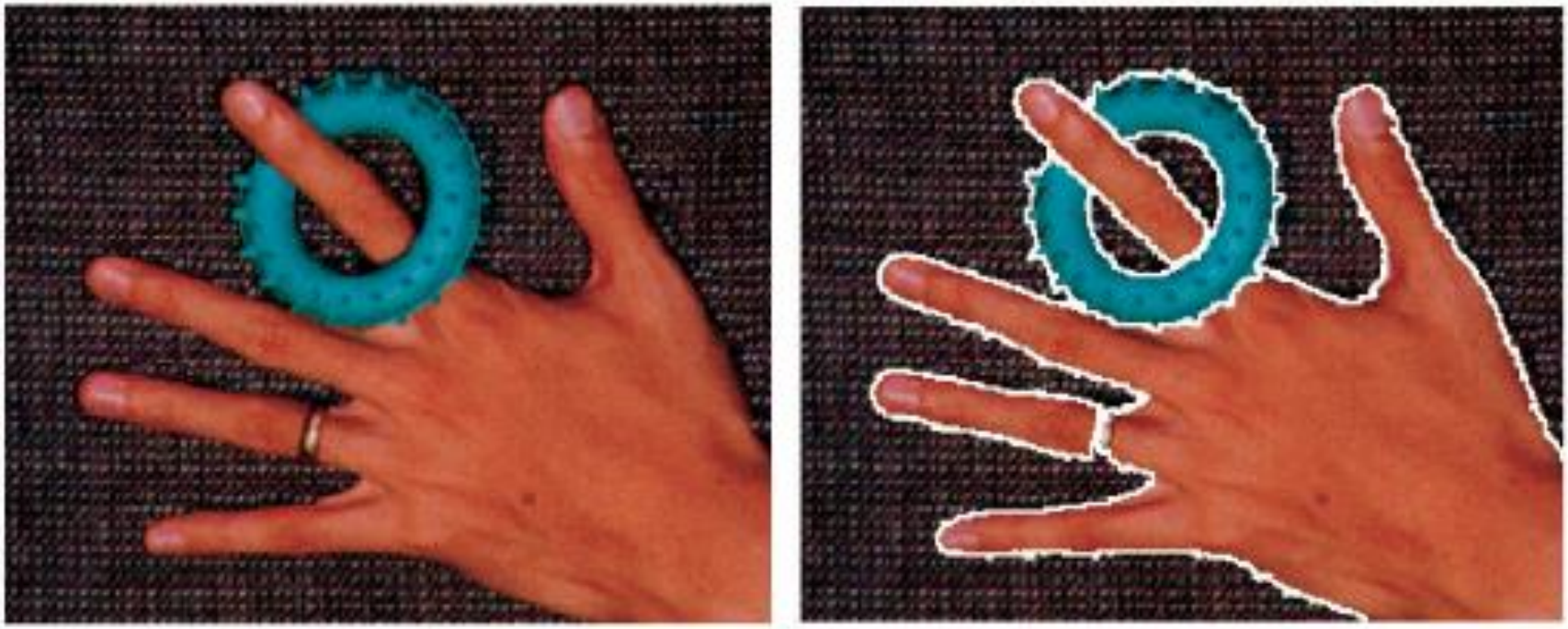
Mean Shift : A robust Approach Toward Feature Space Analysis, by Comaniciu, Meer
http://www.caip.rutgers.edu/~comanici

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.

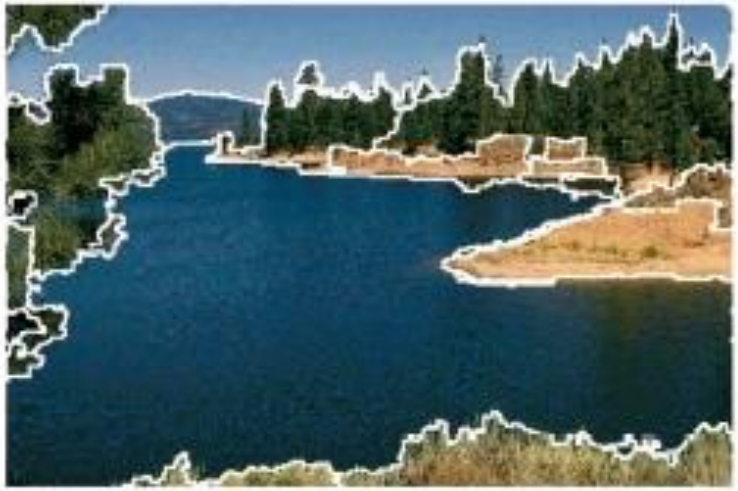
Segmentation

Example



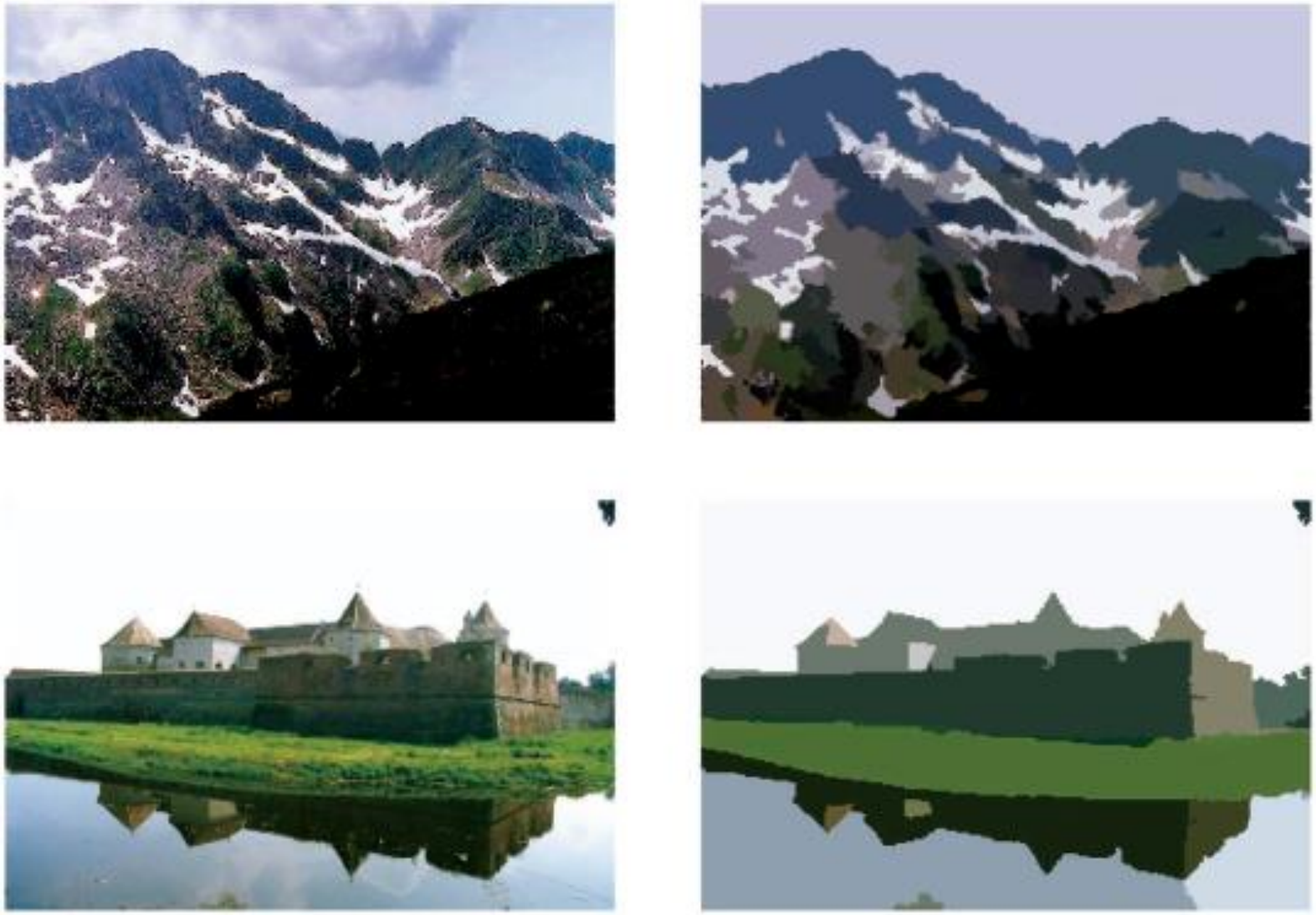
Segmentation

Example



Segmentation

Example



Shape Priors

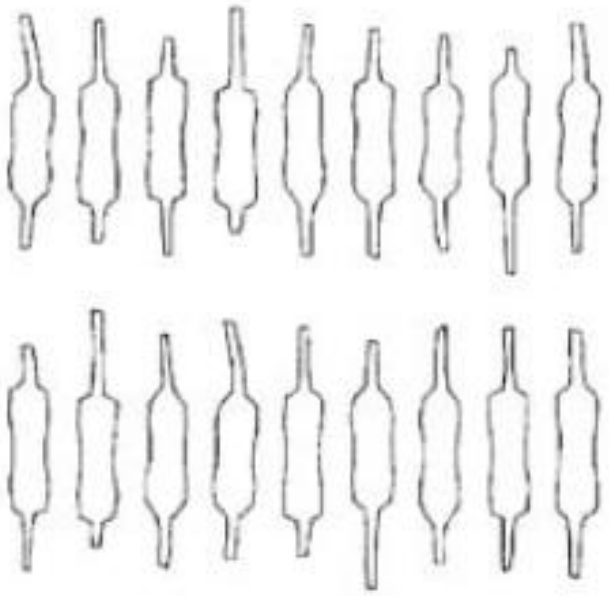
- Snakes sometimes exhibit too many degrees of freedom, making it more likely to be trapped in local minima during their evolution.
- One solution to this problem is to control the snake with fewer degrees of freedom through the use of B-spline approximation, i.e. B-snake

$$\boldsymbol{f}(\boldsymbol{s}) = \sum_{k} B_k(\boldsymbol{s})\boldsymbol{x}_k$$

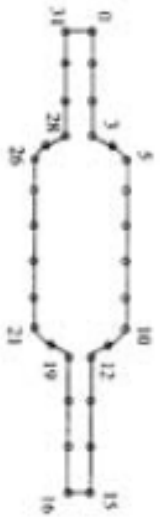
$$\boldsymbol{F} = \boldsymbol{B}\boldsymbol{X}$$

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}^T(0) \\ \vdots \\ \boldsymbol{f}^T(N) \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} B_0(s_0) & \dots & B_K(s_0) \\ \vdots & \ddots & \vdots \\ B_0(s_N) & \dots & B_K(s_N) \end{bmatrix}, \text{ and } \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^T(0) \\ \vdots \\ \boldsymbol{x}^T(K) \end{bmatrix}$$

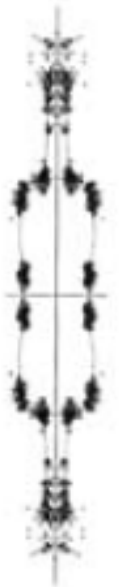
Point Distribution Model



Resistor shapes



Control points



$$x'_k = sRx_k + t$$

Distribution of point locations (after alignment)

One potential way of describing this distribution would be by the location $\bar{x}_k$ and 2D covariance C_k of each individual point x_k. These could then be turned into a quadratic penalty (prior energy) on the point location,

$$E_{\text{loc}}(x_k) = \frac{1}{2}(x_k - \bar{x}_k)^T C_k^{-1}(x_k - \bar{x}_k). \tag{5.13}$$

Shape Prior Constraint

- Covariance matrix C is determined from the training data

$$C = \frac{1}{P} \sum_{p} (\boldsymbol{x}_p - \bar{\boldsymbol{x}})(\boldsymbol{x}_p - \bar{\boldsymbol{x}})^T$$

- Using eigenvalue analysis, i.e. Principal Component Analysis (PCA), the covariance matrix can be written as

$$\boldsymbol{C} = \boldsymbol{\Phi}\ \mathrm{diag}(\lambda_0 \ldots \lambda_{K-1})\ \boldsymbol{\Phi}^T.$$

- The resulting point distribution model can be written as

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \hat{\boldsymbol{\Phi}}\ \boldsymbol{b}$$

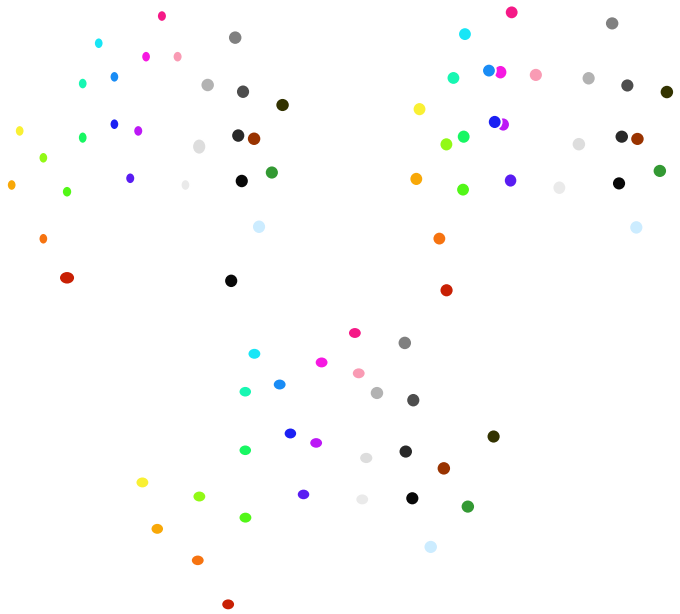
- Quadratic penalty for shape constraint

$$E_{\mathrm{shape}} = \frac{1}{2} \boldsymbol{b}^T\ \mathrm{diag}(\lambda_0 \ldots \lambda_{M-1})\ \boldsymbol{b} = \sum_{m} b_m^2 / 2\lambda_m$$

Deformable Models

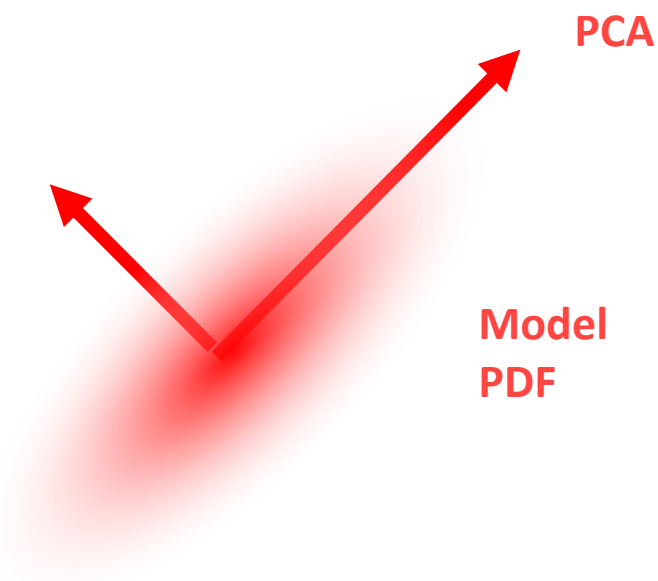
- So far segmentation methods have not exploited knowledge of shape.
- In shape based methods, the segmentation problem is again formulated as an energy-minimization problem. However, a curve evolves in the image until it reaches the lowest energy state instead of a MRF.
- External and internal forces deform the shape control the evolution of segmentation.

Point Distribution Models (PDMs)
Statistical Shape Models (SSMs)



Shape vector:

$$\mathbf{x} = (x_1, y_1, x_2, \ldots, x_{n_s}, y_{n_s})$$



Aligning Training Data

Algorithm 10.5: Approximate alignment of similar training shapes

1. In a pairwise fashion, rotate, scale, and align each $\mathbf{x}^i$ with $\mathbf{x}^1$, for $i = 2, 3, \ldots, M$ to give the set $\{\mathbf{x}^1, \hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M\}$.
2. Calculate the mean of the transformed shapes (the details of this procedure are outlined in Section 10.3).
3. Rotate, scale, and align the mean shape to align to $\mathbf{x}^1$.
4. Rotate, scale, and align $\hat{\mathbf{x}}^2, \hat{\mathbf{x}}^3, \ldots, \hat{\mathbf{x}}^M$ to match to the adjusted mean.
5. If the mean has not converged, go to step 2.

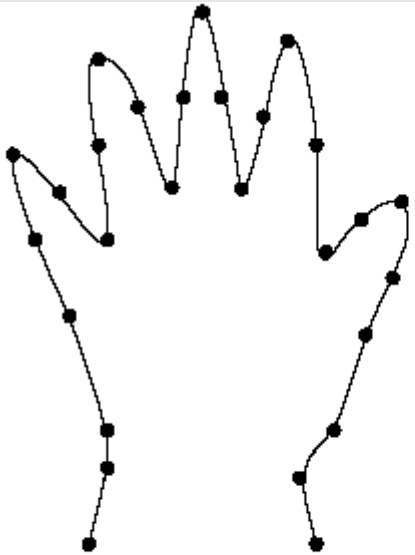


Figure 10.7: A contour representing a hand, with possible landmark points marked.

[1]Autostitch is freely available in demonstration form at http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html.

PDM Model Construction

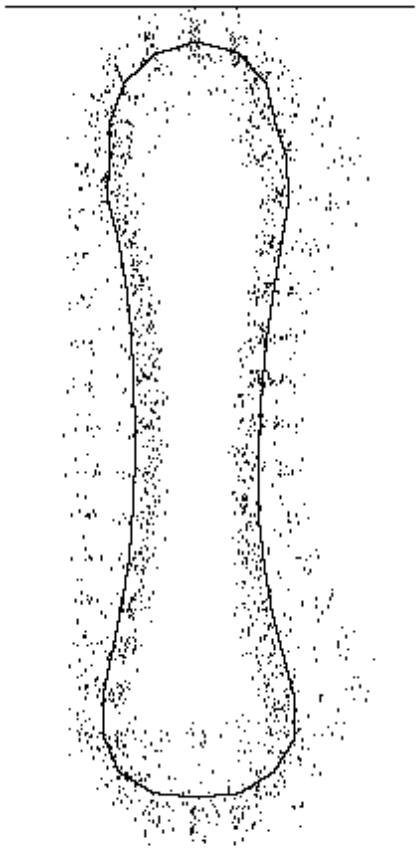


Figure 10.8: PDM of a metacarpal. Dots mark the possible positions of landmarks, and the line denotes the mean shape. *Courtesy of N. D. Efford, School of Computer Studies, University of Leeds.*

0. Given a set of shape vectors x^i
1. Align all shape vectors
2. Compute the mean shape $\bar{\mathbf{x}}$
3. Subtract each shape vector from the mean shape and obtain dx^i
4. Form the covariance matrix S from dx^i
5. Compute the SVD for covariance matrix S
6. Use the eigenvectors corresponding to the dominant eigenvalues to form a linear shape subspace $\{p^1, p^2, \ldots, p^t\}$

Thus, a shape can be approximated by this PDM model

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{Pb}$$

Principal Components

Index i	$\lambda_i/\lambda_{\text{total}}$ [%]	Cumulative total
1	63.3	63.3
2	10.8	74.1
3	9.5	83.6
4	3.4	87.1
5	2.9	90.0
6	2.5	92.5
7	1.7	94.2
8	1.2	95.4
9	0.7	96.1
10	0.6	96.7
11	0.5	97.2
12	0.4	97.6
13	0.3	97.9
14	0.3	98.2
15	0.3	98.5
16	0.2	98.7

Table 10.1: Relative contributions to total data variance for the first 16 principal components.

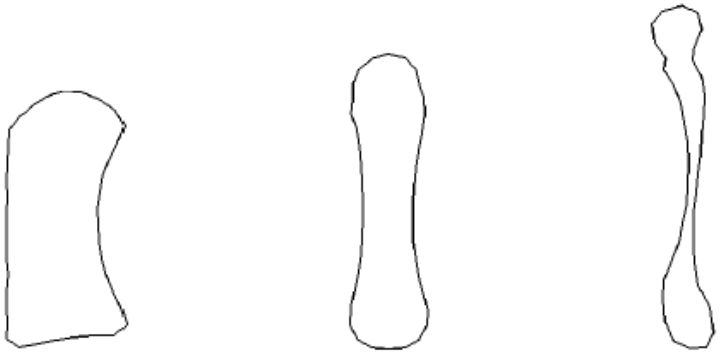


Figure 10.9: The first mode of variation. Left to right: $-2.5\sqrt{\lambda_1}$, mean shape, $2.5\sqrt{\lambda_1}$. *Courtesy of N.D. Efford, School of Computer Studies, University of Leeds.*

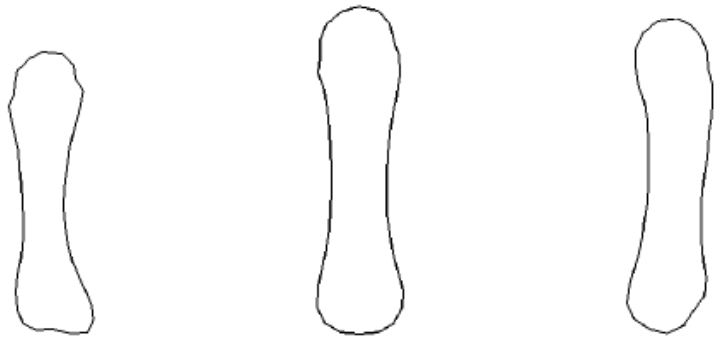


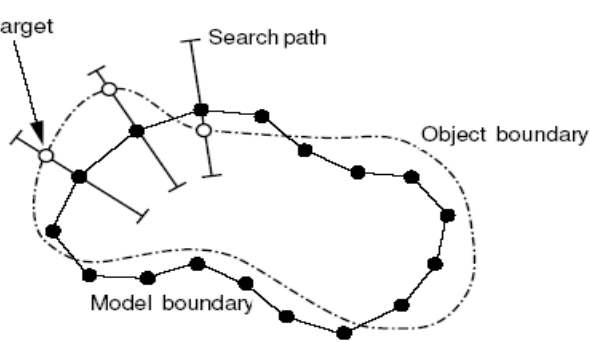
Figure 10.10: The third mode of variation. Left to right: $-2.5\sqrt{\lambda_3}$, mean shape, $2.5\sqrt{\lambda_3}$. *Courtesy of N.D. Efford, School of Computer Studies, University of Leeds.*

Algorithm 10.6: Fitting an ASM

1. Initialize an approximate fit to image data; this may be done in any suitable way but is likely to depend on geometric constraints provided by the application, together with crude image properties. This gives in local (model) co-ordinates a shape description
$$\hat{\mathbf{x}} = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N)\,.$$
2. At each landmark point, inspect the boundary normal close to the boundary, and locate the pixel of highest intensity gradient; mark this as the best target position to which to move this landmark point. This is illustrated in Figure 10.11. If there is no clear new target, the landmark is left where it is.

   We derive thereby a desired displacement vector
3. Adjust the pose parameters to provide the best fit to the target points of the current landmarks.

   There are various ways of doing this, but Algorithm 10.5 provides one approach; a quicker approximation, which is adequate since the iteration will seek out a good solution in time, is given in [Cootes and Taylor, 1992].
4. Determine the displacement vector $\delta\widetilde{\mathbf{x}}$ that adjusts the model in the new pose to the target points (details follow the end of the algorithm).



Fitting ASM (II)

5. Determine the model adjustment $\delta \mathbf{b}_t$ that best approximates $\delta \widetilde{\mathbf{x}}$. From equation (10.5) we have

$$\widetilde{\mathbf{x}} \approx \bar{\mathbf{x}} + P_t \, \mathbf{b}_t$$

and we seek $\mathrm{d}b_t$ such that

$$\widetilde{\mathbf{x}} + \delta\widetilde{\mathbf{x}} = \bar{\mathbf{x}} + P_t(\mathbf{b}_t + \delta \mathbf{b}_t) \, .$$

Hence

$$\delta\widetilde{\mathbf{x}} \approx P_t \, \delta \mathbf{b}_t \, .$$

With the properties of eigen-matrices, we can deduce

$$\delta \mathbf{b}_t = P_t^T \, \delta\widetilde{\mathbf{x}}$$

as the best approximation. Note that since the modes of variation $t+1$, $t+2$, ..., are discounted, this is necessarily only an approximation. Note also that we can at this stage prevent components of the vector $\mathbf{b}_t$ from growing in magnitude beyond any limits we may set by limiting them as we see appropriate—that is, should this equation generate a component deemed to be too large in magnitude, it would be set to the appropriate limit. Thus the re-fitted model will (probably) not match the targets precisely.

6. Iterate from step 2 until changes become negligible.

Example of ASM Fitting

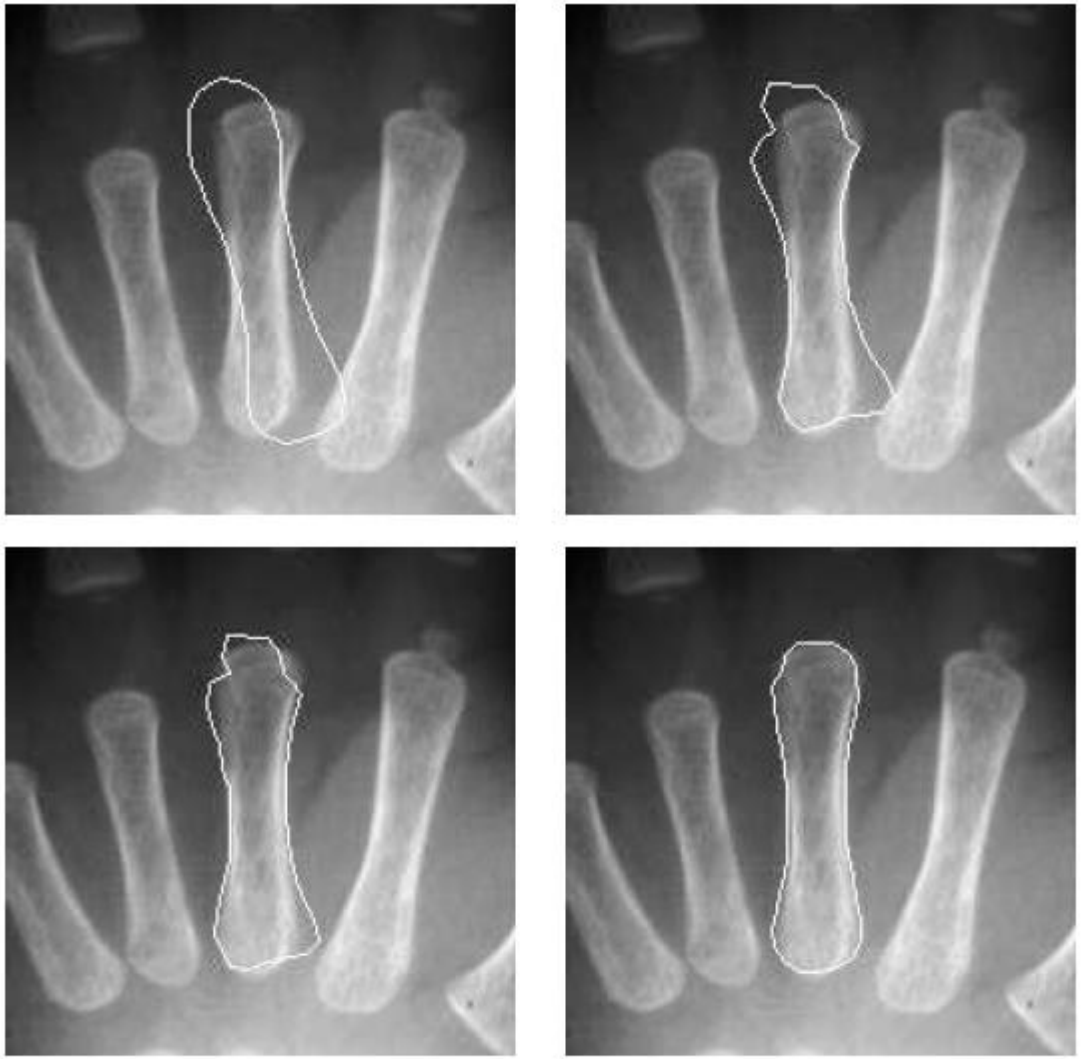


Figure 10.12: Fitting an ASM to a metacarpal; various stages of convergence—initialization, 3, 6, and 10 iterations. *Courtesy of N. D. Efford, School of Computer Studies, University of Leeds.*

Active Appearance Model (AAM)

Algorithm 10.7: AAM construction

1. Compute an ASM and approximate each shape sample as a linear combination of eigenvectors, where $\mathbf{b}_s = P_s^T(\mathbf{x} - \bar{\mathbf{x}})$ represents the sample shape parameters (equation 10.3).
2. Warp each image to the mean shape using a linear or non-linear image interpolation.
3. Normalize each image to the average intensity and unit variance $\overline{\mathbf{g}}$.
4. Perform a PCA on the normalized intensity images.
5. Express each intensity sample as a linear combination of eigenvectors, where $\mathbf{b}_g = P_g^T(\mathbf{g} - \bar{\mathbf{g}})$ represents the sample gray-level parameters.
6. Concatenate the shape coefficient vectors $\mathbf{b}_s$ and gray-level intensity coefficient vectors $\mathbf{b}_g$ in the following manner

$$\mathbf{b} = \begin{bmatrix} W\,\mathbf{b}_s \\ \mathbf{b}_g \end{bmatrix} = \begin{bmatrix} W\,P_s^T(\mathbf{x} - \bar{\mathbf{x}}) \\ P_g^T(\mathbf{g} - \bar{\mathbf{g}}) \end{bmatrix}, \tag{10.6}$$

   where W is a diagonal weighting matrix that relates the different units of shape and gray-level intensity coefficients.

7. Apply PCA to the sample set of all $\mathbf{b}$ vectors, yielding the model

$$\mathbf{b} = Q\mathbf{c}\,, \tag{10.7}$$

   where Q is a matrix consisting of eigenvectors (from equation 10.6) and $\mathbf{c}$ are the resulting model coefficients characterizing how the model instance differs from the mean shape and mean appearance. In other words, with the zero vector $\mathbf{c} = 0$, the modeled instance corresponds to the mean shape and appearance.

AAM Face Modeling

Figure 10.15: Appearance model of Dr. Tim Cootes' face. The middle panel shows the mean shape and appearance, the left and right panels show varying shape and appearance resulting from varying of the parameter $\mathbf{c}$ along one of the modes of combined shape and gray-level appearance (equation 10.7).

AAM Matching

Algorithm 10.8: Active Appearance Model matching

1. Place an appearance model roughly on the object of interest using the parameters $\mathbf{c}$, $\mathbf{t}$, and $\mathbf{u}$ and compute the difference image $\mathbf{g}_s - \mathbf{g}_m$. $g_s = T\, u^{-1}(g_{im})$
2. Compute the RMS of the difference image, $E(\mathbf{r}) = \|\mathbf{r}\|^2$.
3. Compute the model corrections $\delta\mathbf{p}$ as derived above from the residual (equations 10.15). $\delta p = -R\,(g_s - g_m)$
4. Set $k = 1$.
5. Compute new model parameters as $\mathbf{c} := \mathbf{c} - k\delta\mathbf{c}$, $\mathbf{t} := \mathbf{t} - k\delta\mathbf{t}$, and $\mathbf{u} := \mathbf{u} - k\delta\mathbf{u}$.
6. Based on these new parameters, recompute $\mathbf{g}_s - \mathbf{g}_m$ and recalculate the RMS.
7. If the RMS is less than E, accept these parameters and go to step 3.
8. Else set k to 1.5, 0.5, 0.25, etc. and go to step 5. Repeat steps 5–8 until the error cannot be reduced any further.

Summary

- Image segmentation as a data clustering problem
- K-means
- probabilistic clustering
- Mean shift
- Shape prior
- PDM, ASM, AAM
- Semantic segmentation