EE421/521 Image Processing

Lecture 9
IMAGE SEGMENTATION

Introduction

Segmentation

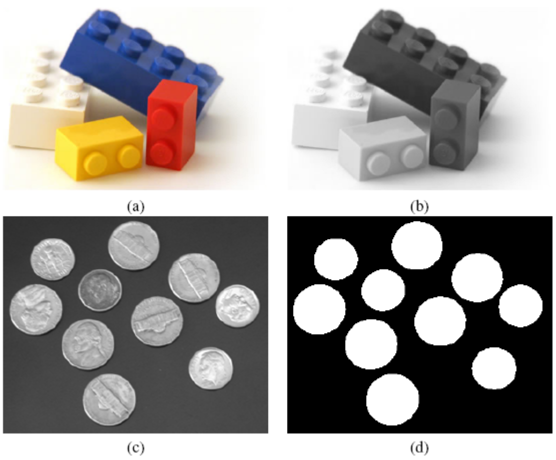
- Partitioning image into homogeneous regions with respect to some characteristic
  - Gray level, texture, color, motion, context (foreground, background)
- Useful mid-level representation of an image
- Facilitates computer vision tasks

Examples

(a) A hard image to segment due to uneven lighting, projected shadows, and occlusion among objects

(b) Grayscale of (a). Segmentation of this image is even harder (even impossible with some methods)

(c) An image easy to segment.



Applications

- Object recognition
- Image annotation
- Video summarization
- Background subtraction
- Medical Imaging
- Image compression

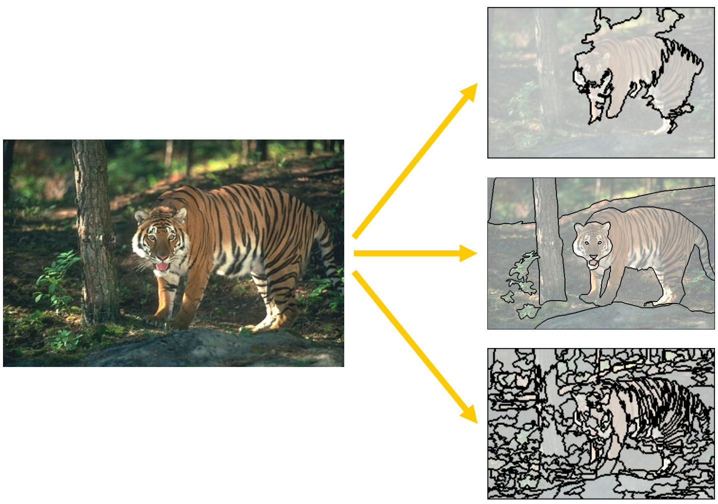
Regions and Edges

- **Regions** are found based on *SIMILARITIES* between values of adjacent pixels
  - We could "trace" regions to obtain **edges**
- **Edges** are found based on *DIFFERENCES* between values of adjacent pixels
  - We could "fill" closed contours to obtain **regions**

Strategies

- Top-down segmentation
  - Pixels belong together because they come from the same object
- Bottom-up segmentation
  - Pixels belong together because they look similar

Over & Under Segmentation



Segmentation Techniques

- Intensity-based (non-contextual): based on pixel distributions (i.e., histograms).
- Region-based (contextual): rely on adjacency and connectivity criteria between a pixel and its neighbors.
  - Region growing
  - Region splitting

9

Intensity-Based Segmentation

Intensity-Based Segmentation

- Rely on pixel statistics (histogram properties) to determine which pixels belong to “foreground” objects and which pixels should be labeled as “background.”

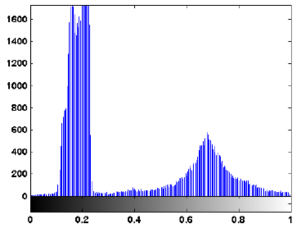
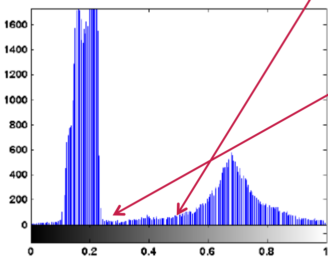


Image Thresholding

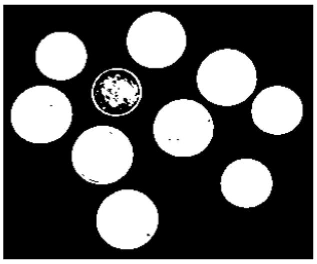
- This is usually performed by comparing each pixel intensity against a reference value (*threshold*) and replacing the pixel with a value (say 1 or 0) that means “foreground” or “background” depending on the outcome of the comparison.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{otherwise} \end{cases}$$

Global Thresholding



`T = 0.4947`



`T = 0.25`

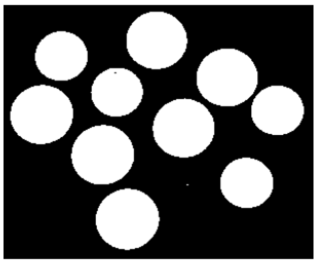
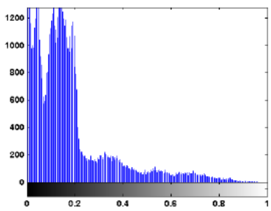
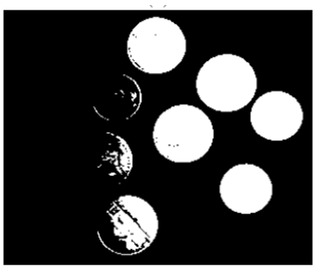


Image Thresholding and Illumination

- Even easy images may become hard to segment

Histogram lost its bimodal shape.

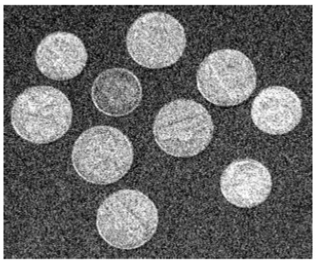




Result of segmentation with T = 0.25.

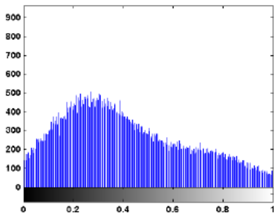
Image Thresholding and Noise

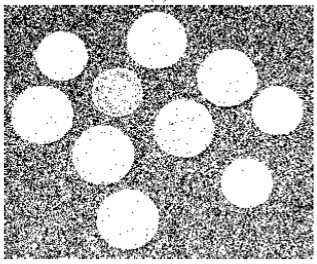
- Even easy images may become hard to segment



Gaussian noise with zero mean and variance 0.03.

Histogram lost its bimodal shape.





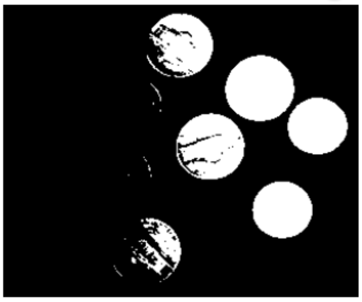
Result of segmentation with T = 0.25.

Local Thresholding

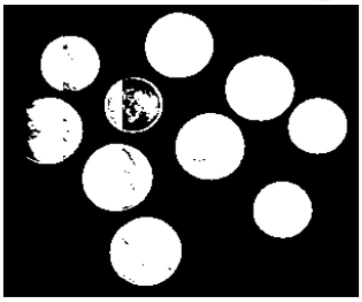
- Threshold blocks of pixels, one block at a time.
  - If the blocks that are too small: large computational cost.
  - If the blocks that are too large: results may not be substantially better than the ones obtained with global thresholding.

Local Thresholding Example

Global thresholding



Local thresholding



Limitations of Thresholding

- Operates on each image pixel independently.
  - Pixels must have similar values.
- Spatial coherency cannot be satisfied.
  - Pixels do not have to be connected.

Region-based Segmentation

Region-based Segmentation

- A pixel cannot be considered a part of an object based solely on its gray value.
- Incorporates measures of connectivity among pixels in order to decide whether those pixels belong to the same region (or object).

Region-Based Segmentation

Divide an image I into n regions R_1, R_2, ..., R_n such that:

1. $\bigcup_{i=1}^{n} R_i = I$
2. R_i is a connected region, $i = 1, 2, \cdots, n$.
3. $R_i \cap R_j = \emptyset$ for all i and j, $i \neq j$.
4. $P(R_i)$ = TRUE for $i = 1, 2, \cdots, n$.
5. $P(R_i \cup R_j)$ = FALSE for any adjacent regions R_i and R_j.

where $P(R_i)$ is a logical predicate defined over the points in set R_i and $\emptyset$ is the empty set.

Region-Based Segmentation

- Logical predicates (also called *homogeneity criteria*) include:
  - Local mean relative to global mean: the average intensity in a region is significantly different than the average gray level in the whole image.
  - Local standard deviation relative to global mean: The standard deviation of the pixel intensities in a region is less than a small percentage of the average gray level in the whole image.
  - Variance: At least a certain percentage of the pixels in a region are within two standard deviations of the local mean.

Region Growing Algorithm

```
Let f(x,y) be the input image
Define a set of regions R1, R2, ..., Rn, each consisting of a
          single seed pixel
repeat
   for i = 1 to n do
      for each pixel p at the border of Ri do
         for all neighbors of p do
            Let (x,y) be the neighbor's coordinates
            Let Mi be the mean gray level of pixels in Ri
            if the neighbor is unassigned and
                      |f(x,y) - Mi| <= Delta then
               Add neighbor to Ri
               Update Mi
            end if
         end for
      end for
   end for
until no more pixels can be assigned to regions
```


Region-Growing Example

$$P(R_i) = \begin{cases} \text{TRUE} & \text{if } |f(x,y) - \mu_i| \leq \Delta \\ \text{FALSE} & \text{otherwise} \end{cases}$$

6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

Seed pixels

6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

Results after first iteration, delta = 3

6	7	7	6	5
7	7	8	6	5
5	5	6	7	6
0	1	2	0	1
1	0	0	2	0

Results after second iteration

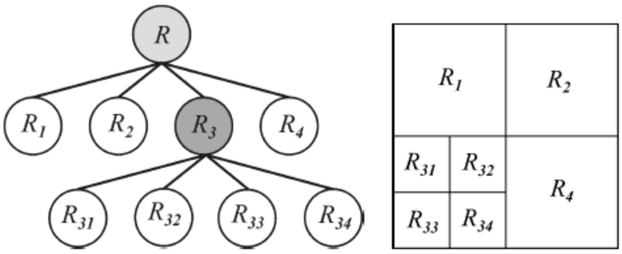
Region Growing Limitations

- Significantly different results may be obtained when switching between 4-connectivity and 8-connectivity criteria.
- Segmentation results are sensitive to the choice of logical uniformity predicate.
- The number of seeds provided by the user may not be sufficient to assign every pixel to a region, or some may belong to the same region.

Region Splitting and Merging

- Start from the entire image and partition (split) it into smaller sub-images until each resulting region is considered homogeneous by some criterion.
- Merge two or more adjacent regions into one region if they satisfy the homogeneity criterion.

Region Splitting and Merging



By Oge Marques

Region Splitting and Merging

1. Define a logical uniformity predicate $P(R_i)$.
2. Compute $P(R_i)$ for each region.
3. Split into four disjoint quadrants any region R_i for which $P(R_i)$ = FALSE.
4. Repeat steps 2 and 3 until all resulting regions satisfy the uniformity criterion, i.e., $P(R_i)$ = TRUE.
5. Merge any adjacent regions R_j and R_k for which $P(R_j \cup R_k)$ = TRUE.
6. Repeat step 5 until no further merging is possible.

By Oge Marques

Project 2.4

Segmentation

Due 05.01.2014

Problem 2.4 Segmentation via Region Growing

1. Select an image and display it.
2. Guess the number of segments in this image and select a seed pixel in each region. Show the **seed** pixels on the image.
3. Use the region growing technique to assign each pixel to one of the regions (compare a pixel's intensity with the average intensity value of the region). Employ **8- connectivity** criterion.
4. Repeat Steps 2 & 3 with two different **number of regions** and two different **threshold values** and comment on the results.

Segmentation as Clustering

Segmentation as Clustering

- Agglomerative clustering
  - Start with each point in a separate cluster
  - At each iteration, merge two of the "closest" clusters
- Divisive clustering
  - Start with all points grouped into a single cluster
  - At each iteration, split the "largest" cluster

Segmentation as Clustering

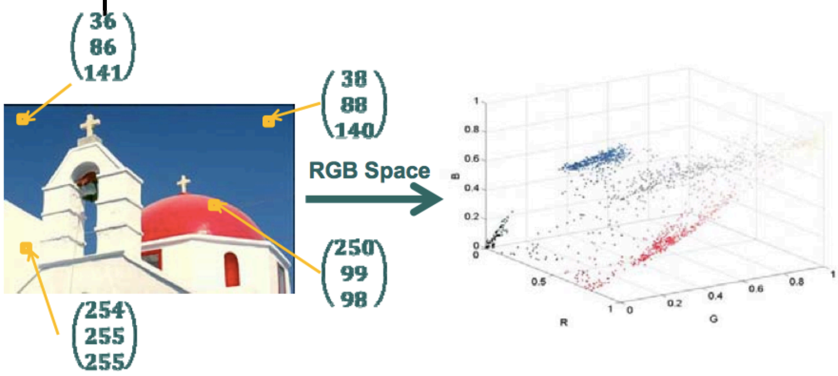
$$v_i = \begin{pmatrix} x_i \\ y_i \\ R_i \\ G_i \\ B_i \\ f_1(x_i, y_i) \\ \vdots \\ f_k(x_i, y_i) \end{pmatrix}$$

- Select a set of image features; position $\{x, y\}$, color $\{R, G, B\}$, a set of filter responses $\{f_1(x,y) \ldots f_k(x,y)\}$
- For each pixel p_i form a feature vector v_i

Features in Feature Space

- The vector represents a point in an N dimensional feature space.
- The feature vectors for similar pixels should occupy nearby locations in this feature space
- Thus, homogeneous image regions become dense clouds of feature vectors in feature space.

Example



- The feature space has 3 dimensions (RGB).
- Principal image regions generate dense clusters in feature space.

Spatial + RGB Space

- The feature space has 5 dimensions (XY-RGB).

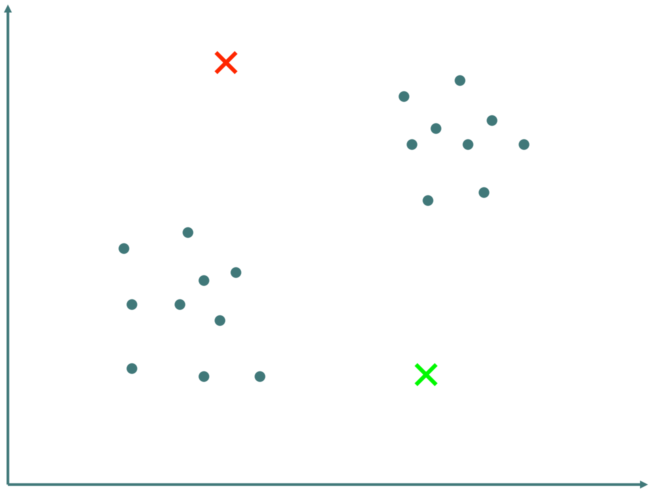
K-Means Clustering

- Compute the feature space vectors
- Randomly select K cluster centers in feature space
- Iterate until convergence
  - Assign feature vectors to the closest cluster center
  - Re-compute the cluster centers as a (weighted) mean of the feature vectors assigned to each cluster
- Label pixels according to the cluster their feature vectors belong to

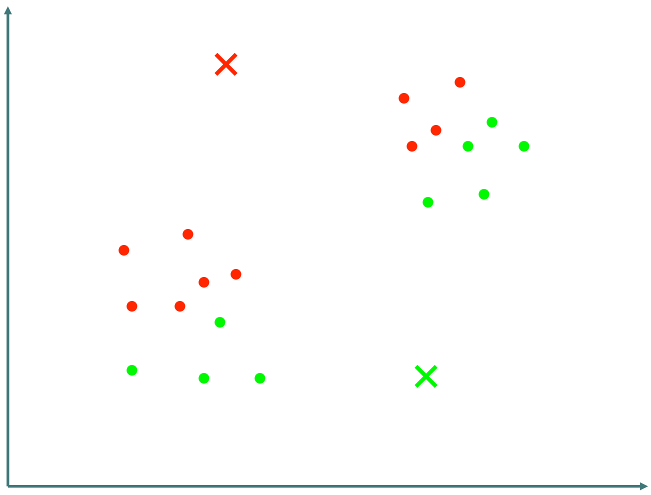
37

K-Means Clustering Sample

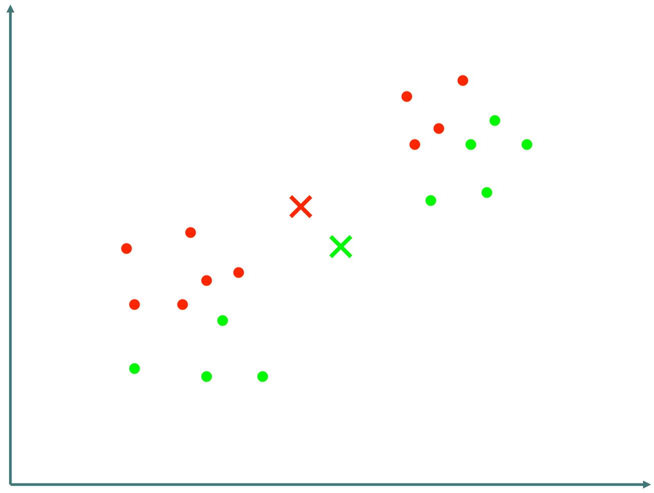
K-Means Clustering



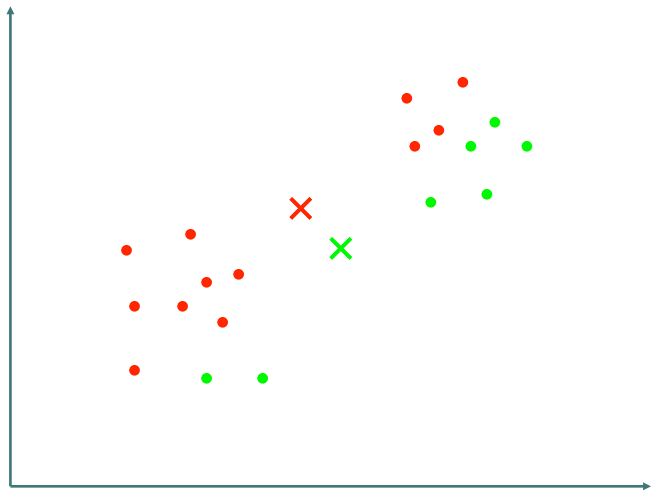
K-Means Clustering



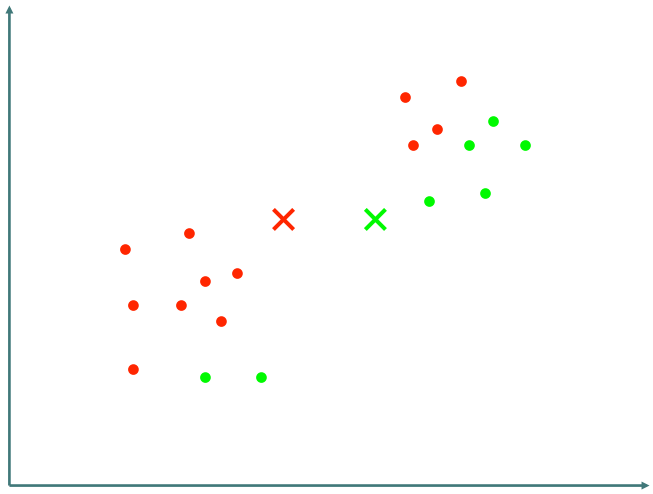
K-Means Clustering



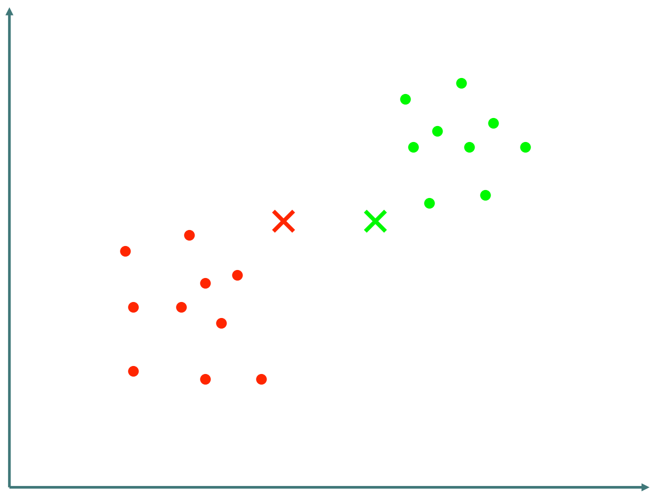
K-Means Clustering



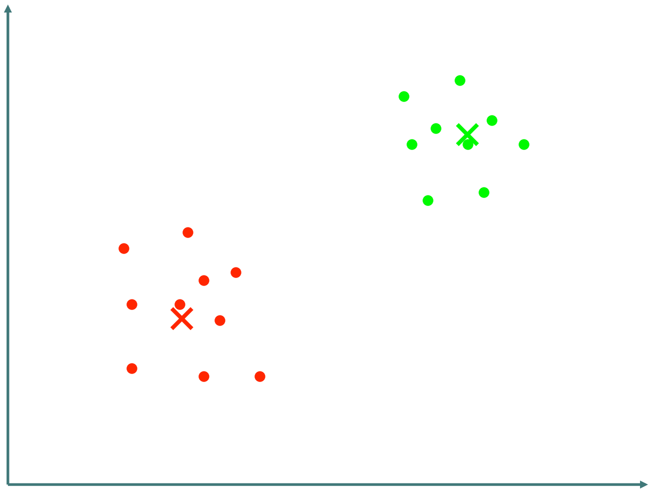
K-Means Clustering



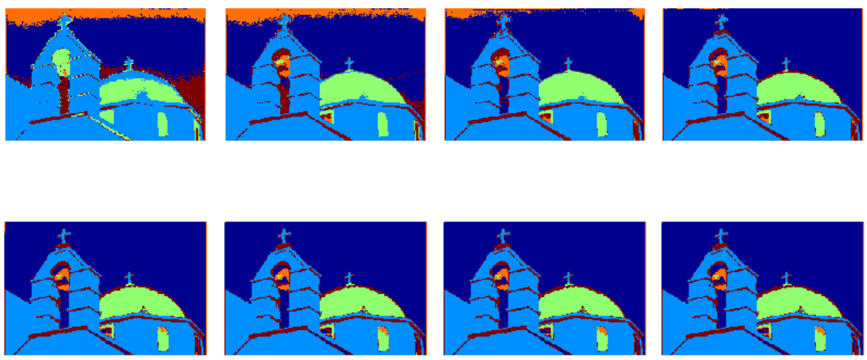
K-Means Clustering



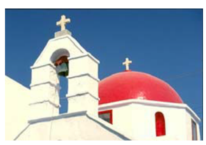
K-Means Clustering



Sample Iterations

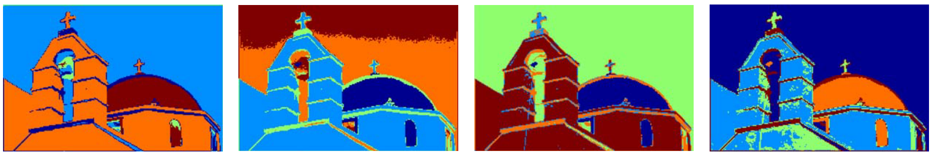


8 iterations of the K means procedure, K=5



Parameter Selection

Effect of random initialization, K=5



Effect of the choice K

K=3 K=5 K=8 K=15

K-Means pros and cons

Pros

- Simple and fast
- Converges to a local minimum of the error function

Cons

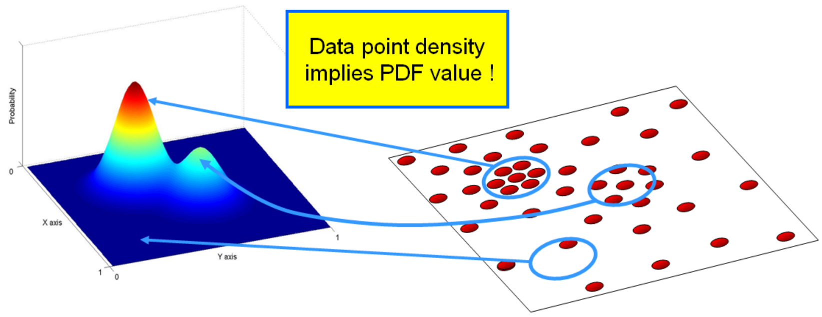
- Need to pick K
- Sensitive to initialization
- Sensitive to outliers
- Only finds “spherical” clusters

Mean Shift

- An advanced and versatile technique for clustering-based segmentation
- Finds modes in a set of data samples, manifesting an underlying probability density function (PDF) in $R{\uparrow}N$
- Seeks a mode or local maximum of density of a given distribution

Probability Density Function Estimation

Assumption : The data points are sampled from an underlying PDF



Assumed Underlying PDF **Real Data Samples**

Mean Shift

PDF Estimation

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

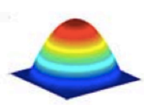
Mean Shift : Estimate not PDF but the GRADIENT

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

Various Kernels (K)

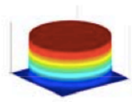
- Epanechnikov Kernel

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



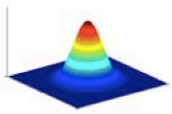
- Uniform Kernel

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



- Normal Kernel

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



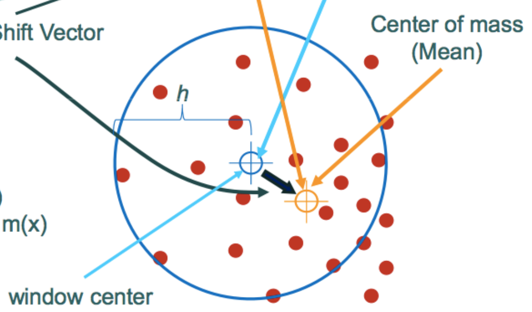
Computing The Mean Shift

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

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

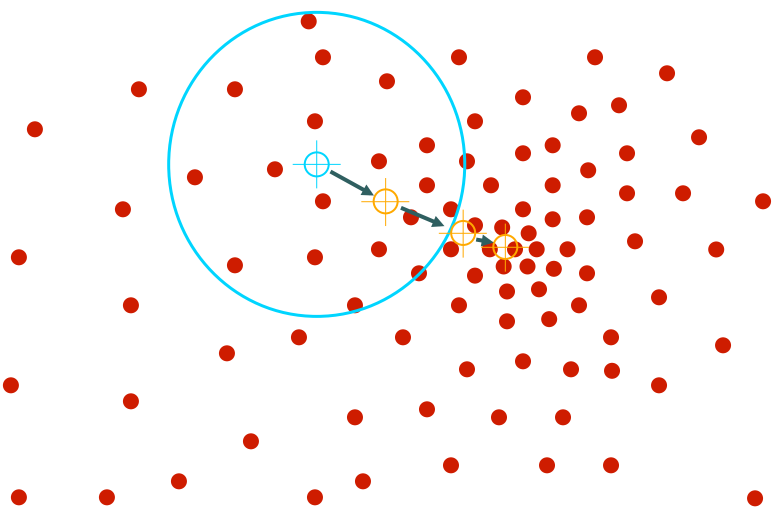
$g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$

Mean Shift Vector

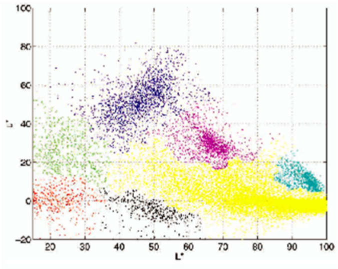


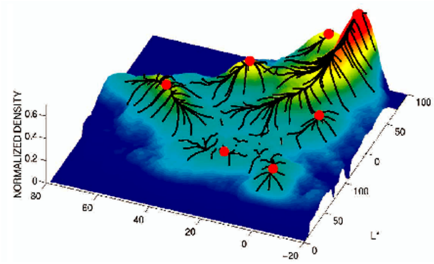
Simple Mean Shift procedure:
- Compute mean shift vector m(x)
- Translate the Kernel window by m(x)

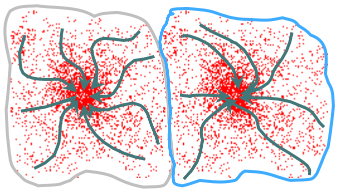
Mean Shift Iterations



Clustering

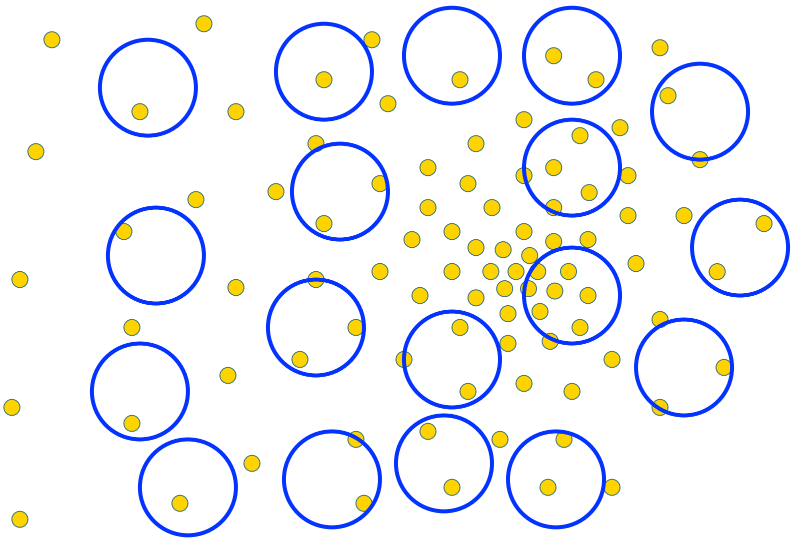






Attraction basin : the region for which all trajectories lead to the same mode

Clustering



Tessellate the space with windows

Run the procedure in parallel

Mean Shift Segmentation

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

57

Mean Shift Properties

- Automatic convergence speed – the mean shift vector size depends on the gradient itself.
- Near maxima, the steps are small and refined
- Convergence is guaranteed for infinitesimal steps only
- For Uniform Kernel, convergence is achieved in a finite number of steps
- Normal Kernel exhibits a smooth trajectory, but is slower than Uniform Kernel.

Mean Shift Segmentation Examples

Input Image | Smaller search window | Larger search window

Mean Shift Pros and Cons

Pros

- Does not assume spherical clusters
- Just a single parameter (window size)
- Finds variable number of modes
- Robust to outliers

Cons

- Output depends on window size
- Computationally expensive
- Does not scale well with dimension of feature space

Next Lecture

- NOISE FILTERING