Image Segmentation

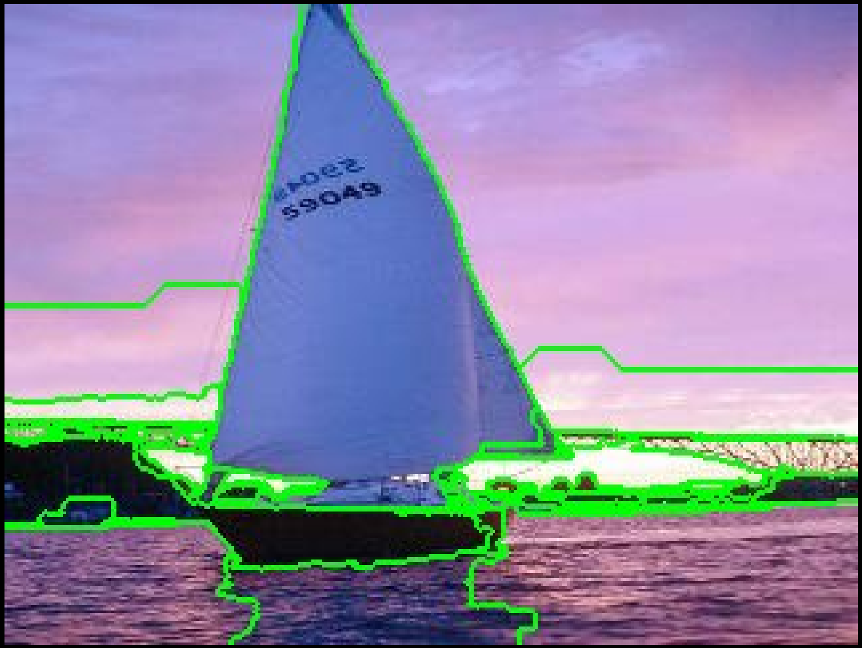
Image segmentation is the operation of partitioning an image into a collection of connected sets of pixels.

1. into regions, which usually cover the image

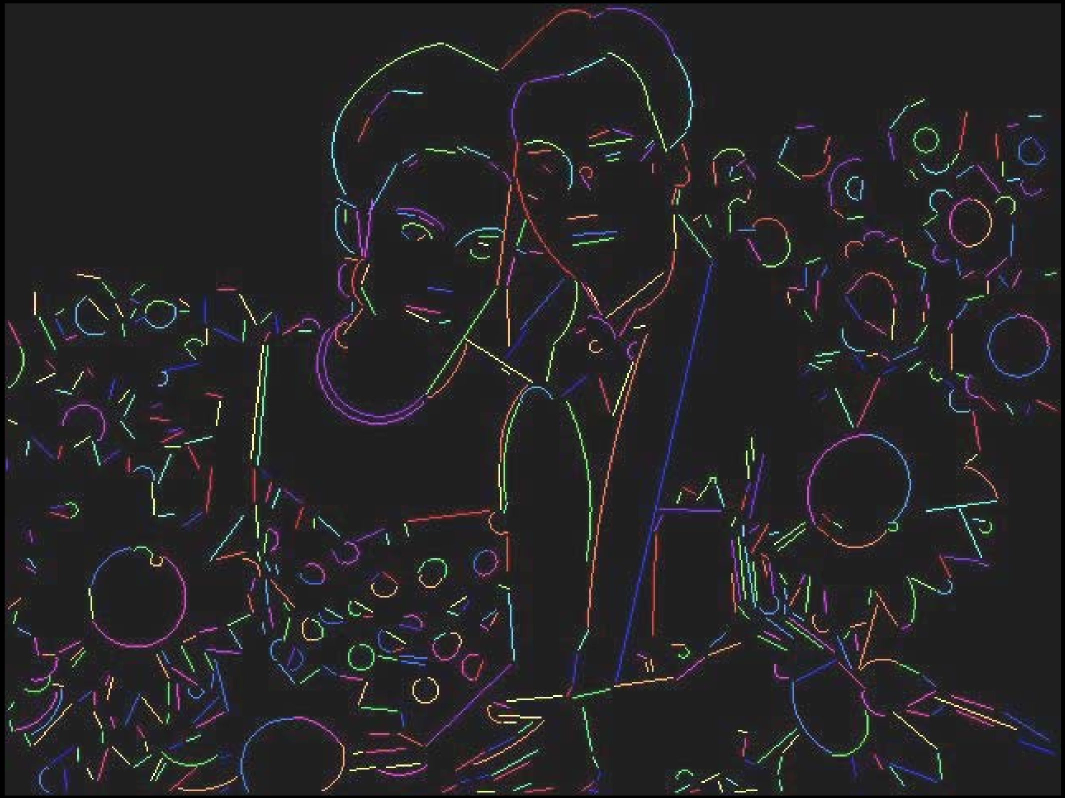
2. into linear structures, such as
    - line segments
    - curve segments

3. into 2D shapes, such as
    - circles
    - ellipses
    - ribbons (long, symmetric regions)

Example 1: Regions



Example 2:
Lines and Circular Arcs



Main Methods of Region Segmentation

1. Region Growing
2. Split and Merge
3. Clustering

Clustering

- There are K clusters C1,…, CK with means m1,…, mK.
- The **least-squares error** is defined as

$$D = \sum_{k=1}^{K} \sum_{xi \in Ck} || xi - mk ||^2 .$$

- Out of all possible partitions into K clusters, choose the one that minimizes D.

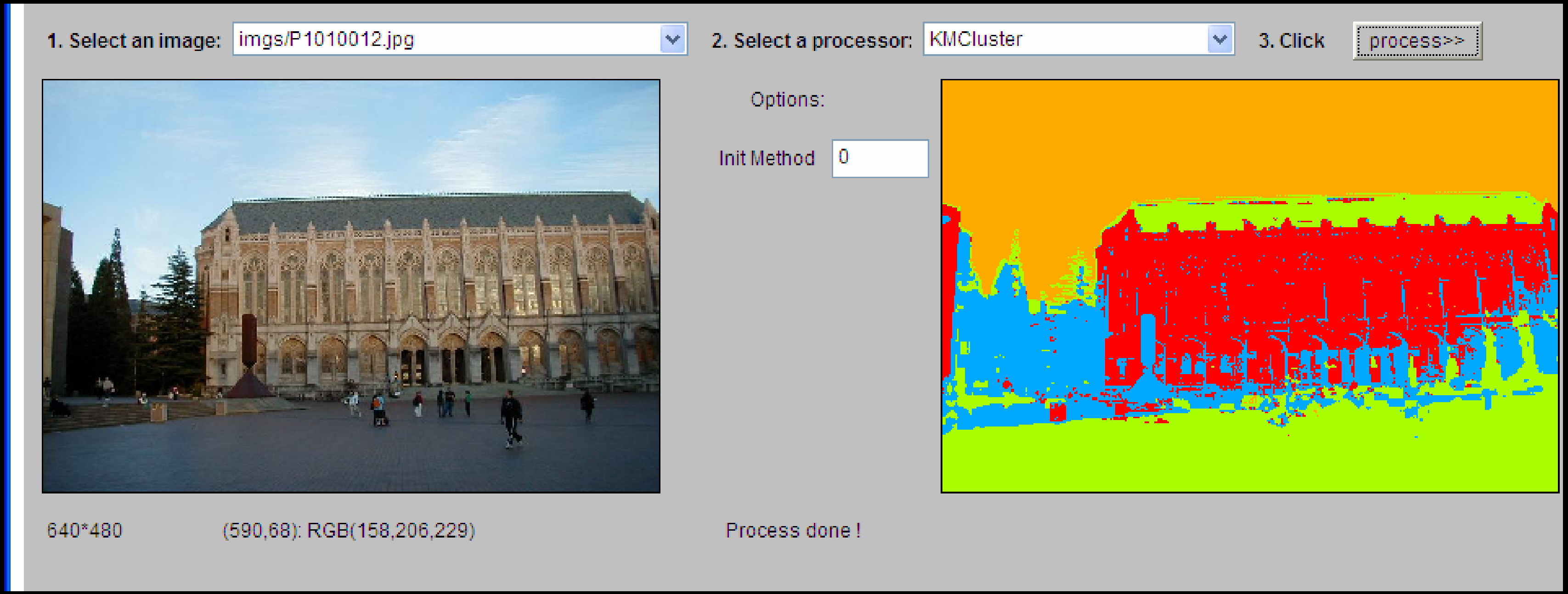
Why don't we just do this?
If we could, would we get meaningful objects?

K-Means Clustering

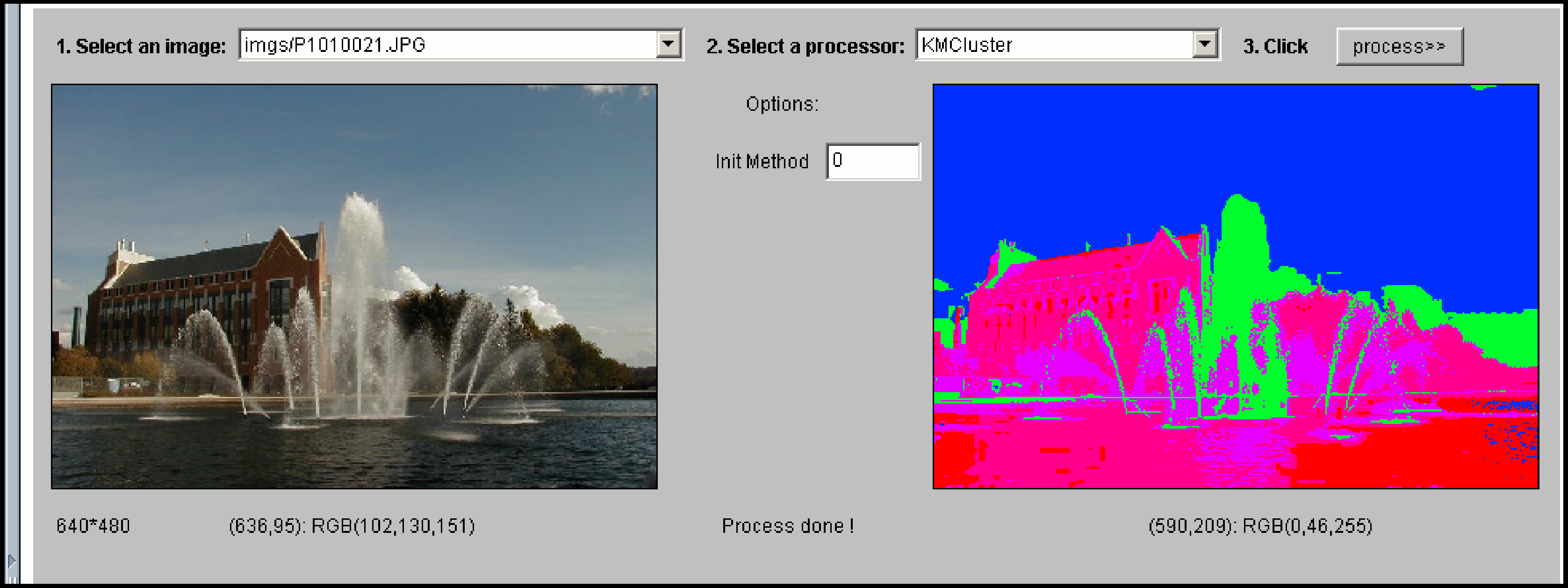
Form K-means clusters from a set of n-dimensional vectors

1. Set ic (iteration count) to 1
2. Choose randomly a set of K means m1(1), ..., mK(1).
3. For each vector xi, compute D(xi,mk(ic)), k=1,...K and assign xi to the cluster Cj with nearest mean.
4. Increment ic by 1, update the means to get m1(ic),...,mK(ic).
5. Repeat steps 3 and 4 until Ck(ic) = Ck(ic+1) for all k.

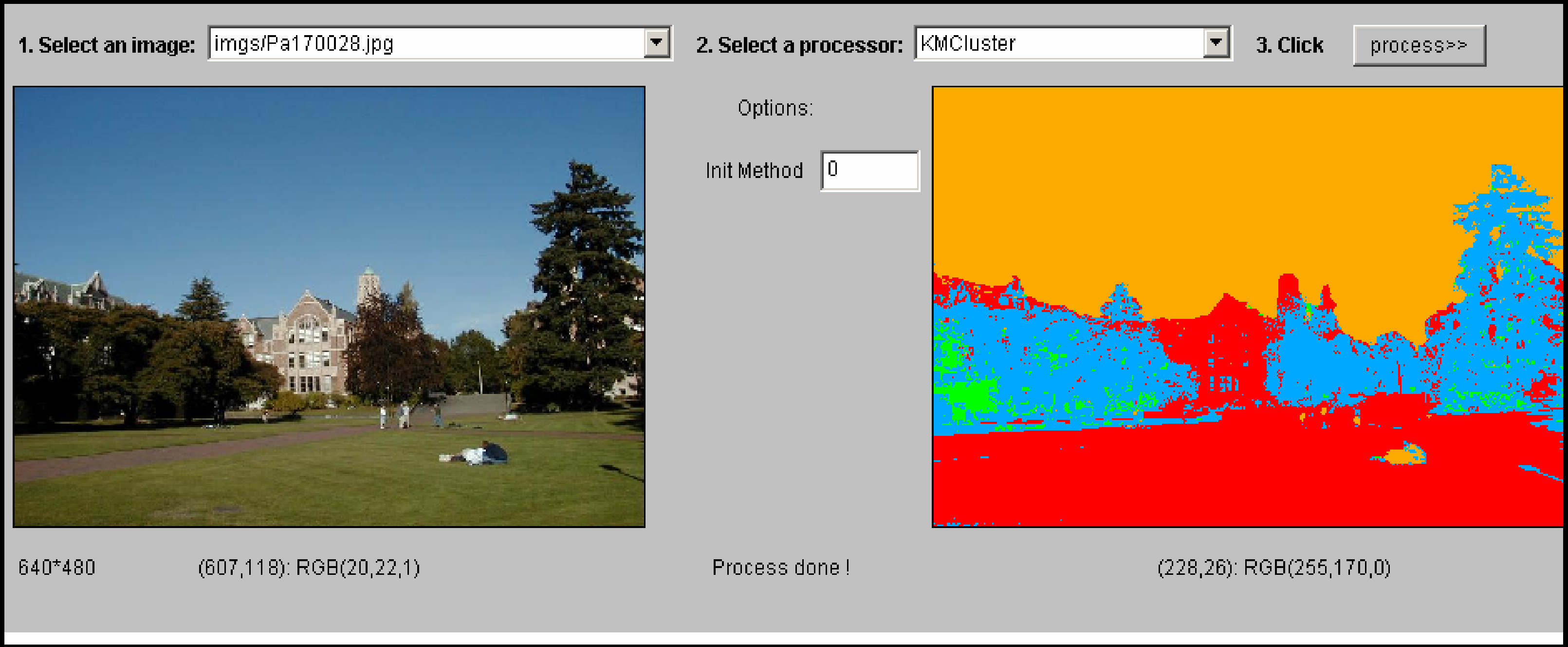
K-Means Example 1



K-Means Example 2



K-Means Example 3



K-means Variants

- Different ways to initialize the means
- Different stopping criteria
- Dynamic methods for determining the right number of clusters (K) for a given image
- The EM Algorithm: a probabilistic formulation (slides courtesy of Yi Li)

K-Means Classifier

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

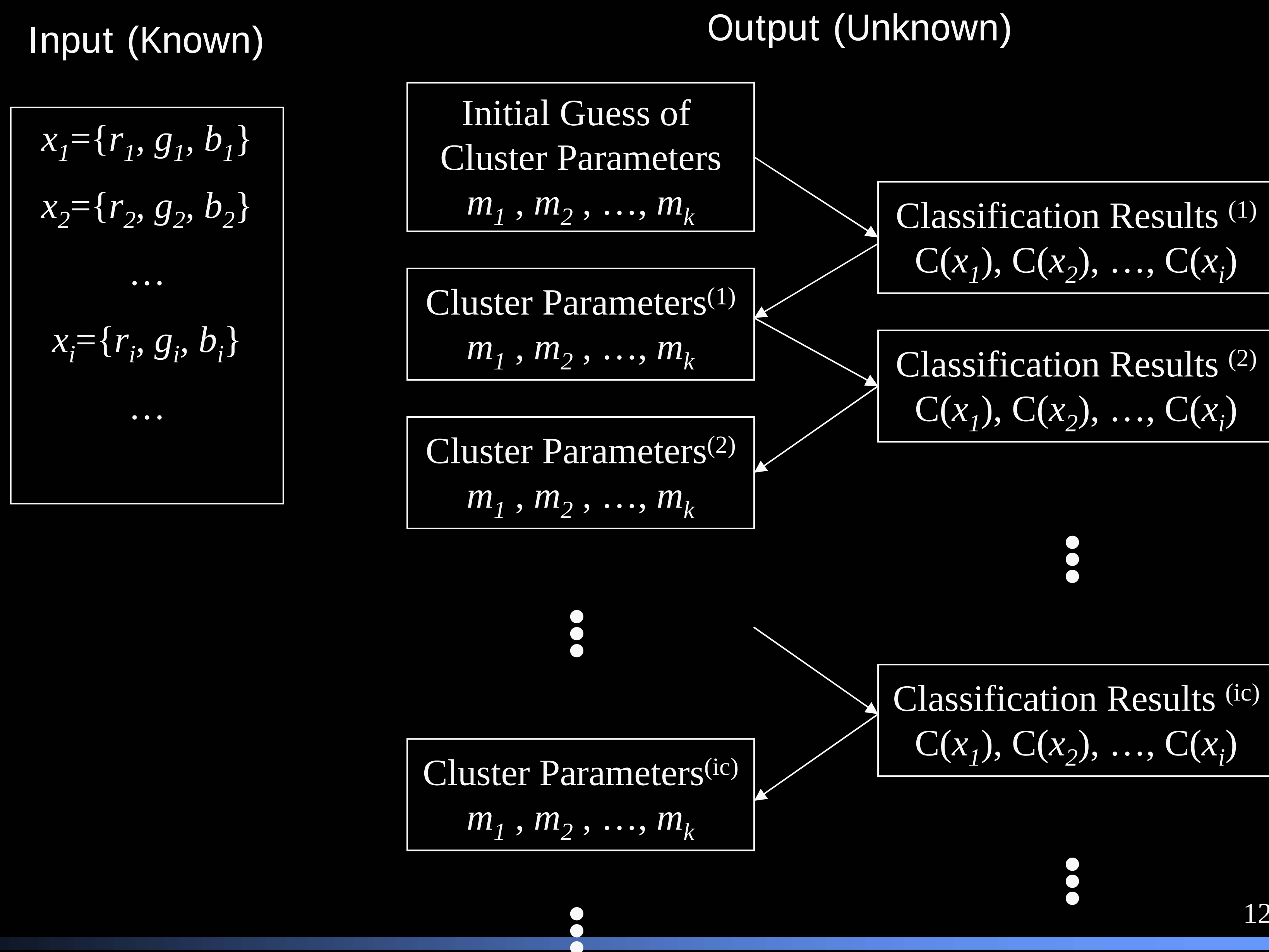
$x_i=\{r_i, g_i, b_i\}$

…

Output (Unknown)

Cluster Parameters
m_1 for C_1
m_2 for C_2
…
m_k for C_k

Classification Results
$x_1 \rightarrow C(x_1)$
$x_2 \rightarrow C(x_2)$
…
$x_i \rightarrow C(x_i)$
…



Input (Known)
Output (Unknown)
$x_1=\{r_1, g_1, b_1\}$
$x_2=\{r_2, g_2, b_2\}$
…
$x_i=\{r_i, g_i, b_i\}$
…
Initial Guess of Cluster Parameters
$m_1 , m_2 , \ldots, m_k$
Classification Results $^{(1)}$
$C(x_1), C(x_2), \ldots, C(x_i)$
Cluster Parameters$^{(1)}$
$m_1 , m_2 , \ldots, m_k$
Classification Results $^{(2)}$
$C(x_1), C(x_2), \ldots, C(x_i)$
Cluster Parameters$^{(2)}$
$m_1 , m_2 , \ldots, m_k$
Classification Results $^{(ic)}$
$C(x_1), C(x_2), \ldots, C(x_i)$
Cluster Parameters$^{(ic)}$
$m_1 , m_2 , \ldots, m_k$

K-Means

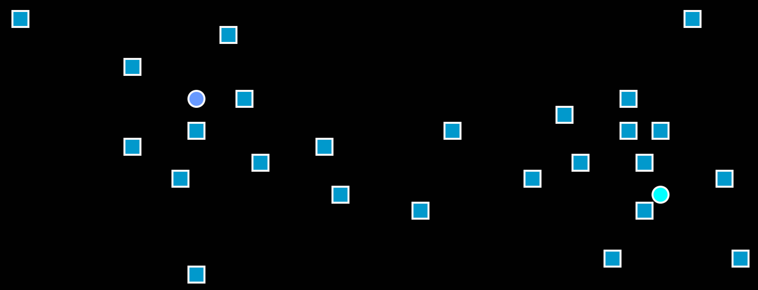
- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

    Each cluster is represented by its mean m_j
- Iteration Step:
  - Estimate the cluster for each data point

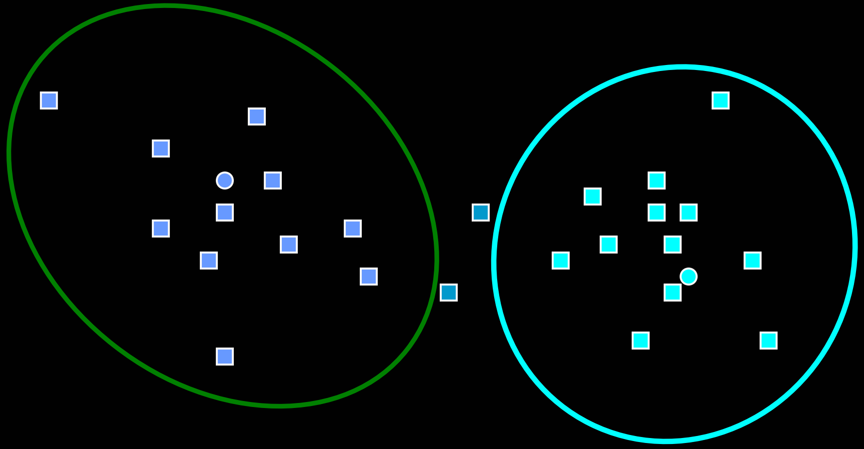
    $$x_i \Longrightarrow C(x_i)$$
  - Re-estimate the cluster parameters

    $$m_j = mean\{x_i \mid x_i \in C_j\}$$

K-Means Example



K-Means Example



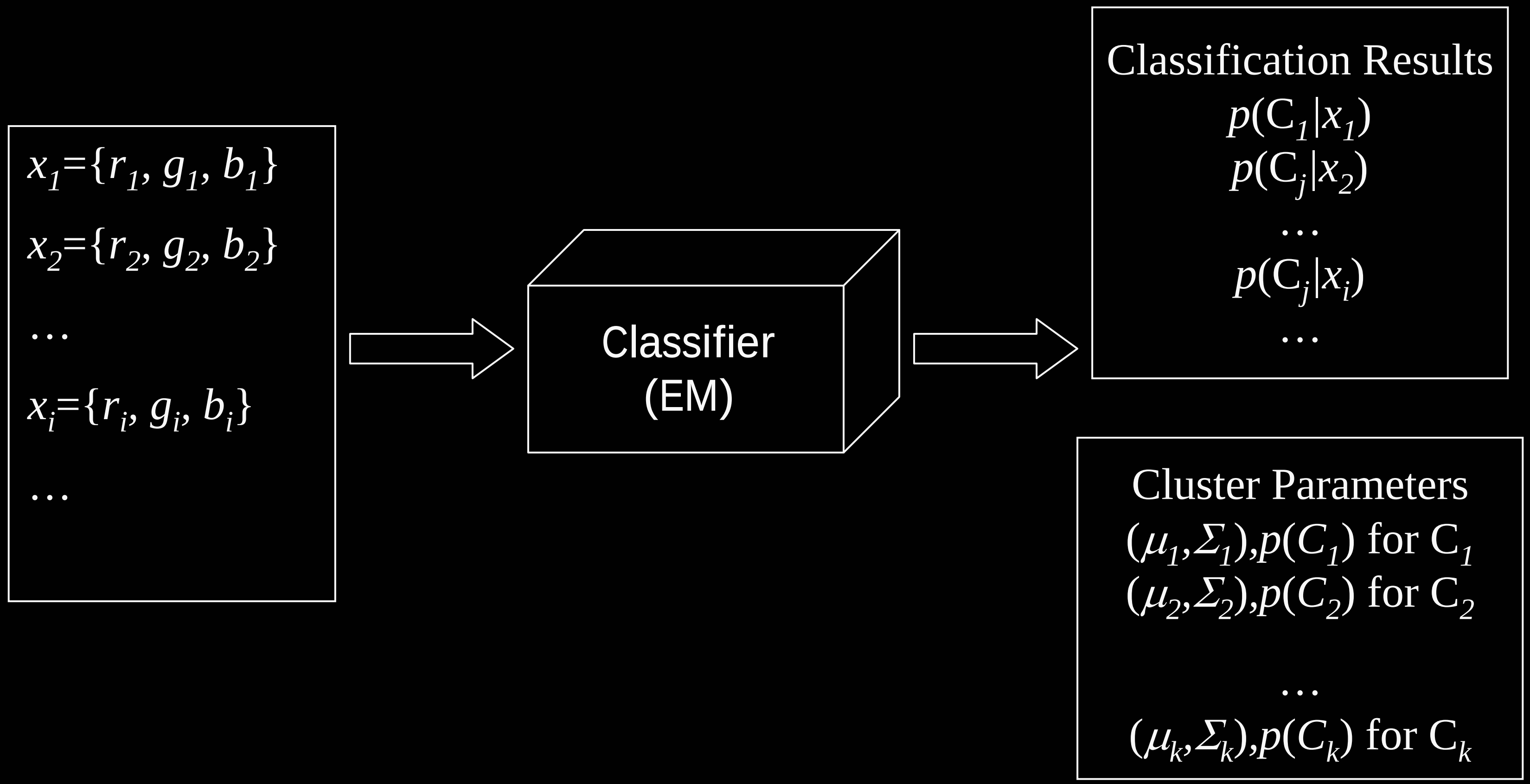
K-Means $\rightarrow$ EM

- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

    (μ_j, Σ_j) and $P(C_j)$ for each cluster j.
- Iteration Step:
  - Estimate the cluster of each data point ⇨ Expectation
  - Re-estimate the cluster parameters ⇨ Maximization

For each cluster j

EM Classifier



EM Classifier (Cont.)

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

Output (Unknown)

Cluster Parameters

(μ_1,Σ_1), $p(C_1)$ for C_1

(μ_2,Σ_2), $p(C_2)$ for C_2

…

(μ_k,Σ_k), $p(C_k)$ for C_k

Classification Results

$p(C_1|x_1)$

$p(C_j|x_2)$

…

$p(C_j|x_i)$

…

Expectation Step

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

+

Input (Estimation)

Cluster Parameters

(μ_1,Σ_1), $p(C_1)$ for C_1

(μ_2,Σ_2), $p(C_2)$ for C_2

…

(μ_k,Σ_k), $p(C_k)$ for C_k

⇒

Output

Classification Results

$p(C_1|x_1)$

$p(C_j|x_2)$

…

$p(C_j|x_i)$

…

Maximization Step

Input (Known)

$x_1=\{r_1, g_1, b_1\}$

$x_2=\{r_2, g_2, b_2\}$

…

$x_i=\{r_i, g_i, b_i\}$

…

+

Input (Estimation)

Classification Results

$p(C_1|x_1)$

$p(C_j|x_2)$

…

$p(C_j|x_i)$

…

⇨

Output

Cluster Parameters

(μ_1,Σ_1), $p(C_1)$ for C_1

(μ_2,Σ_2), $p(C_2)$ for C_2

…

(μ_k,Σ_k), $p(C_k)$ for C_k

EM Algorithm

- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

    (μ_j, Σ_j) and $P(C_j)$ for each cluster j.

- Iteration Step:
  - Expectation Step

$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

  - Maximization Step

$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

EM Demo

- Demo

  http://www.neurosci.aist.go.jp/~akaho/MixtureEM.html

- Example

  http://www-2.cs.cmu.edu/~awm/tutorials/gmm13.pdf

EM Applications

- Blobworld: Image segmentation using Expectation-Maximization and its application to image querying
- Yi’s Generative/Discriminative Learning of object classes in color images

Image Segmentation with EM

- The feature vector for pixel i is called x_i.
- There are going to be K segments; K is given.
- The j-th segment has a Gaussian distribution with parameters $\theta_j=(\mu_j,\Sigma_j)$.
- α_j*'s* are the weights (which sum to 1) of Gaussians. Θ is the collection of parameters:

  $\Theta = (\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k)$

Initialization

- Each of the K Gaussians will have parameters $\theta_j=(\mu_j,\Sigma_j)$, where
  - μ_j is the mean of the j-th Gaussian.
  - Σ_j is the covariance matrix of the j-th Gaussian.
- The covariance matrices are initialed to be the identity matrix.
- The means can be initialized by finding the average feature vectors in each of K windows in the image; this is data-driven initialization.

E-Step

$$p(j \mid x_i, \Theta) = \frac{\alpha_j f_j(x_i \mid \theta_j)}{\sum_{k=1}^{K} \alpha_k f_k(x_i \mid \theta_k)}$$

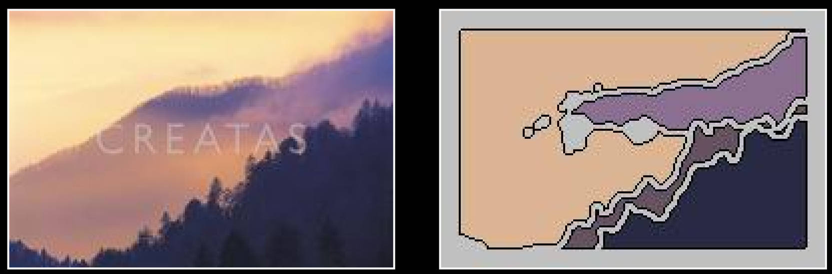
$$f_j(x \mid \theta_j) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_j \mid^{1/2}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1} (x-\mu_j)}$$

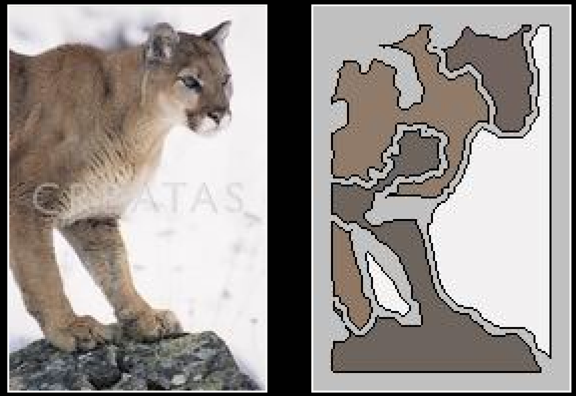
M-Step

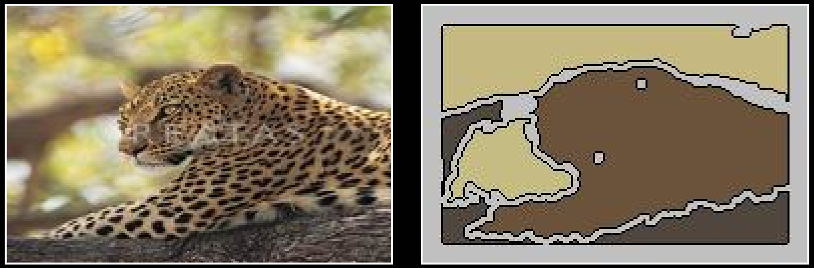
$$\mu_j^{new} = \frac{\sum_{i=1}^{N} x_i \, p(j \mid x_i, \Theta^{old})}{\sum_{i=1}^{N} p(j \mid x_i, \Theta^{old})}$$

$$\Sigma_j^{new} = \frac{\sum_{i=1}^{N} p(j \mid x_i, \Theta^{old})(x_i - \mu_j^{new})(x_i - \mu_j^{new})^T}{\sum_{i=1}^{N} p(j \mid x_i, \Theta^{old})}$$

Sample Results



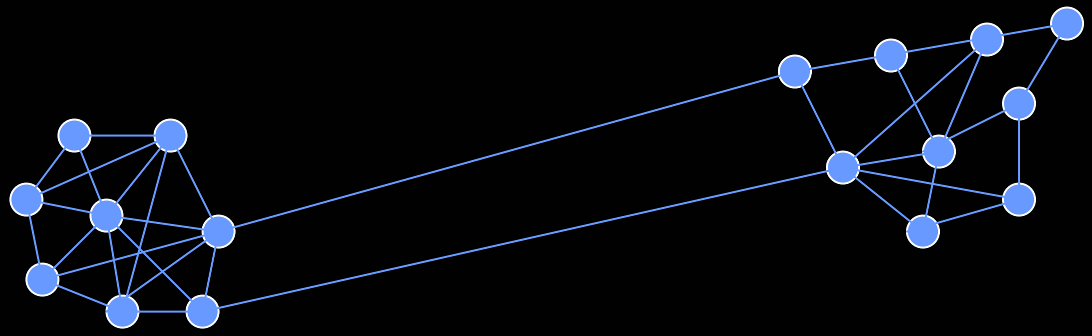




Jianbo Shi's Graph-Partitioning

- An image is represented by a graph whose nodes are pixels or small groups of pixels.

- The goal is to partition the vertices into disjoint sets so that the similarity within each set is high and across different sets is low.

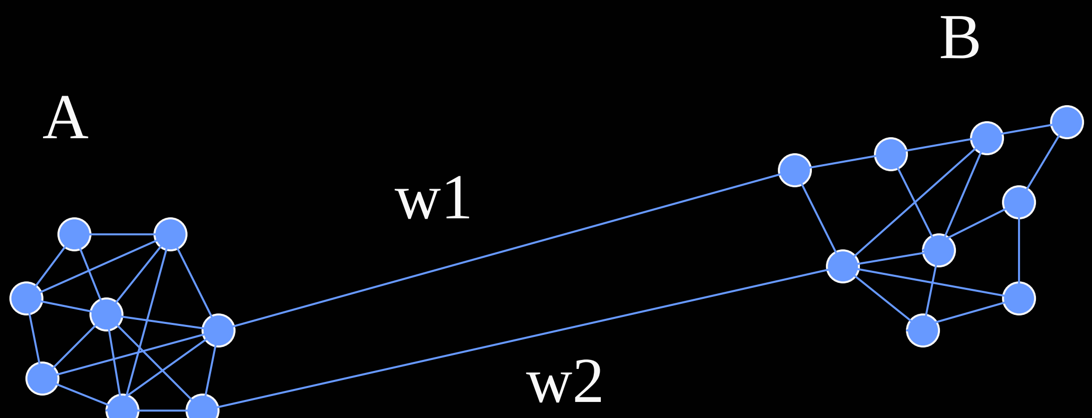


Minimal Cuts

- Let $G = (V,E)$ be a graph. Each edge (u,v) has a weight $w(u,v)$ that represents the similarity between u and v.
- Graph G can be broken into 2 disjoint graphs with node sets A and B by removing edges that connect these sets.
- Let $cut(A,B) = \sum_{u \varepsilon A,\ v \varepsilon B} w(u,v)$.
- One way to segment G is to find the minimal cut.

Cut(A,B)

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



Normalized Cut

Minimal cut favors cutting off small node groups, so Shi proposed the normalized cut.

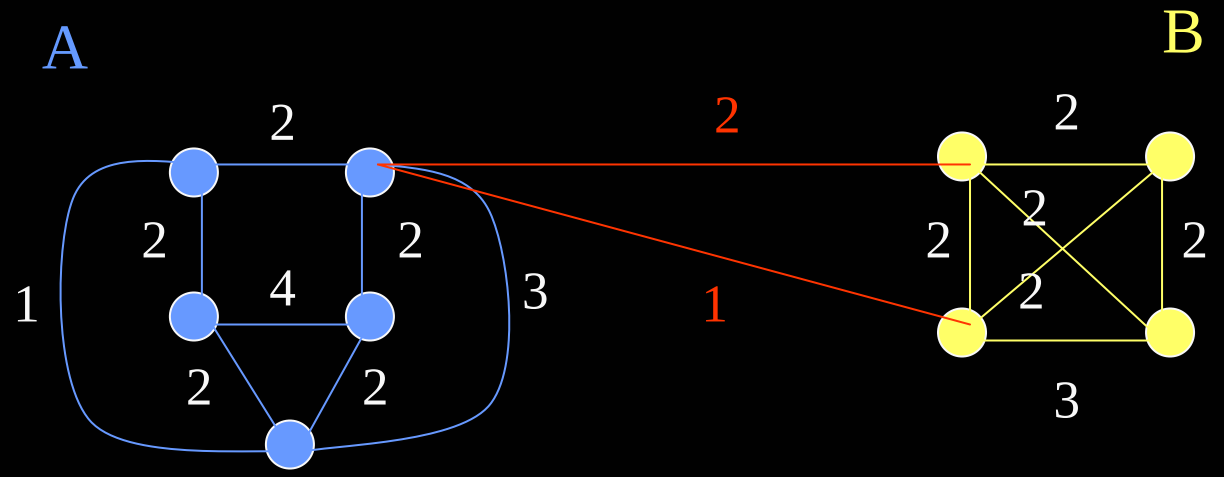
$$\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A,B)}{\mathrm{asso}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{asso}(B,V)}$$

normalized cut

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

How much is A connected to the graph as a whole.

Example Normalized Cut



$$Ncut(A,B) = \frac{3}{21} + \frac{3}{16}$$

Shi turned graph cuts into an eigenvector/eigenvalue problem.

- Set up a weighted graph G=(V,E)
  - V is the set of (N) pixels
  - E is a set of weighted edges (weight w_{ij} gives the similarity between nodes i and j)
  - Length N vector d: d_i is the sum of the weights from node i to all other nodes
  - N x N matrix D: D is a diagonal matrix with d on its diagonal
  - N x N symmetric matrix W: $W_{ij} = w_{ij}$

- Let x be a characteristic vector of a set A of nodes
  - $x_i = 1$ if node i is in a set A
  - $x_i = -1$ otherwise
- Let y be a continuous approximation to x

$$y = (1+x) - \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}(1-x).$$

- Solve the system of equations

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

  for the eigenvectors y and eigenvalues λ
- Use the eigenvector y with second smallest eigenvalue to bipartition the graph (y => x => A)
- If further subdivision is merited, repeat recursively

How Shi used the procedure

Shi defined the edge weights w(i,j) by

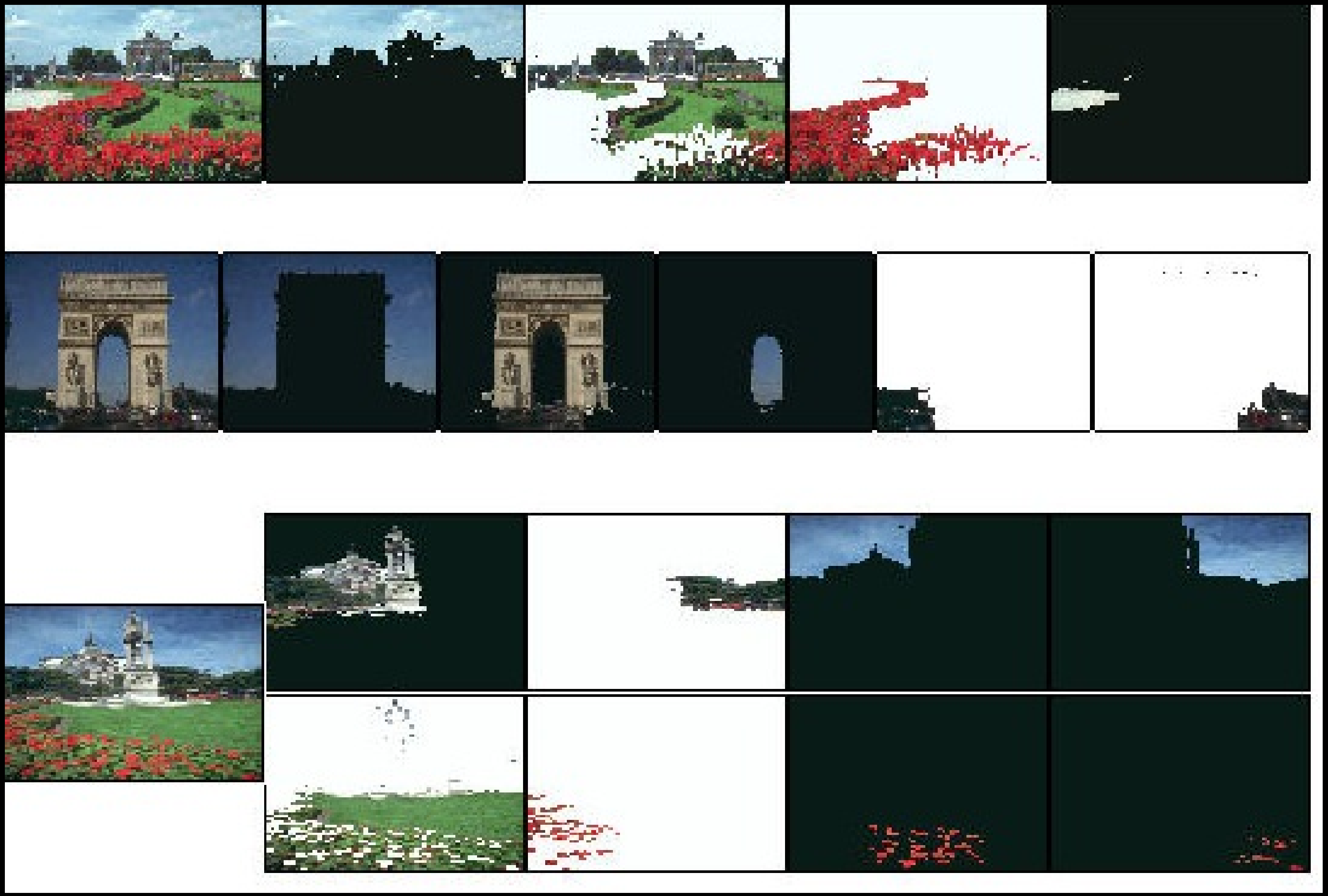
$$w(i,j) = e^{\|F(i)-F(j)\|_2 / \sigma I} * \begin{cases} e^{\|X(i)-X(j)\|_2 / \sigma X} & \text{if } \|X(i)-X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

where X(i) is the spatial location of node i
F(i) is the feature vector for node I
which can be intensity, color, texture, motion…

The formula is set up so that w(i,j) is 0 for nodes that are too far apart.

Examples of Shi Clustering

See Shi's Web Page
http://www.cis.upenn.edu/~jshi/



Problems with Graph Cuts

- Need to know when to stop
- Very Slooooow

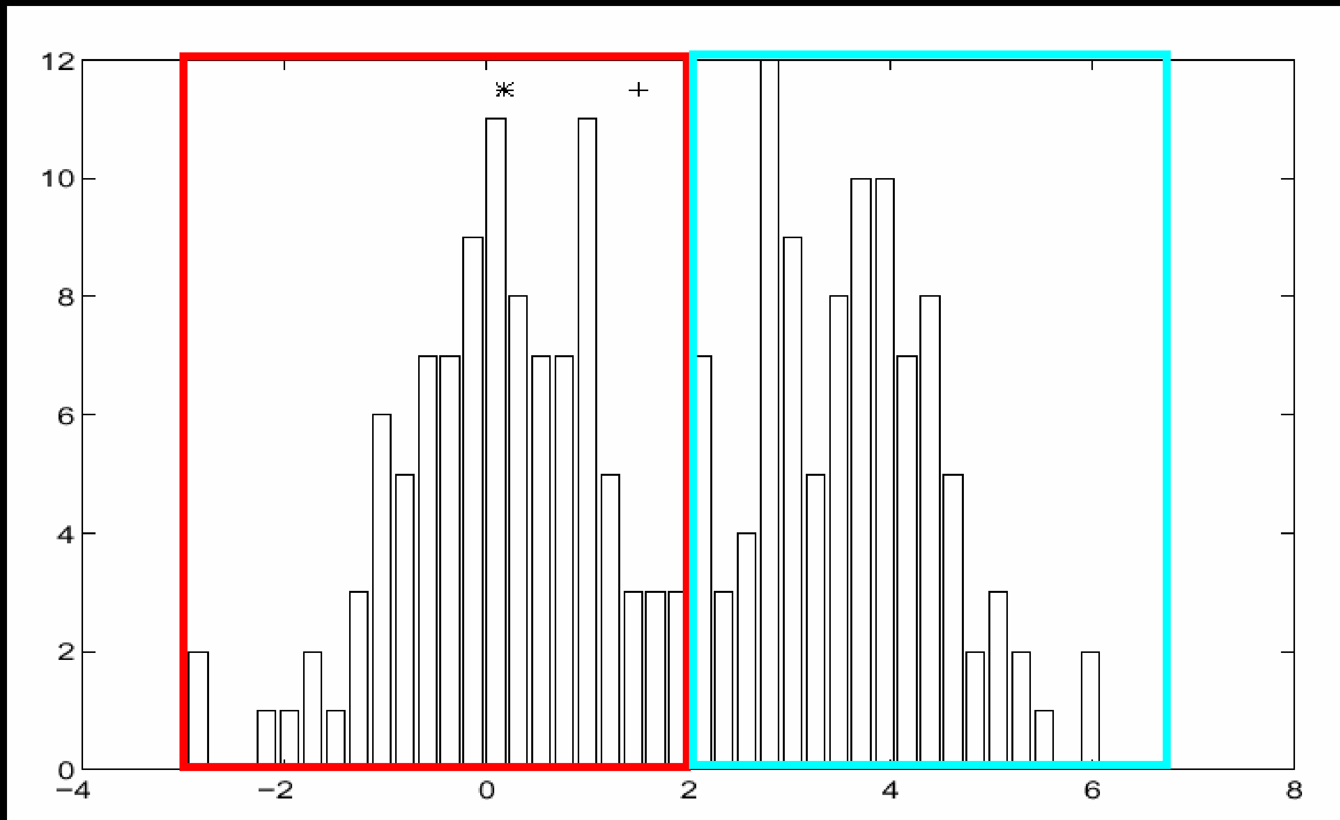
Problems with EM

- Local minima
- Need to know number of segments
- Need to choose generative model

Mean-Shift Clustering

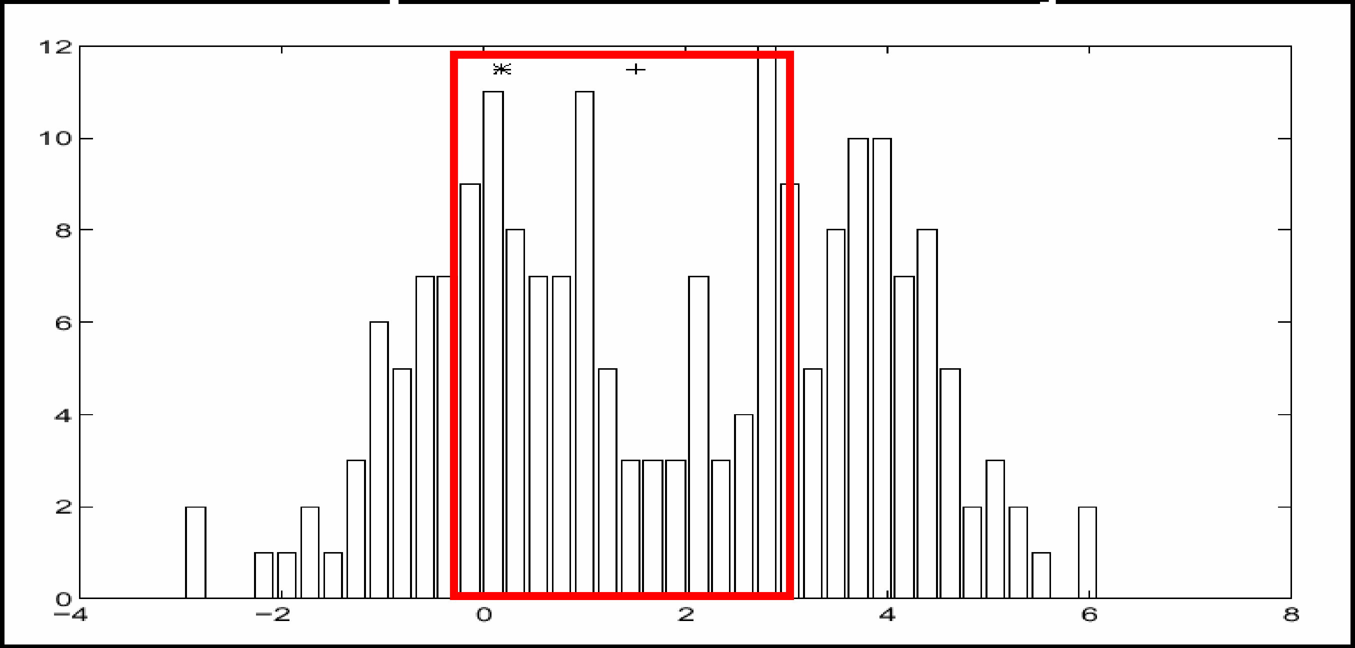
- Simple, like K-means
- But you don’t have to select K
- Statistical method
- Guaranteed to converge to a fixed number of clusters.

Finding Modes in a Histogram



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

Mean Shift [Comaniciu & Meer]

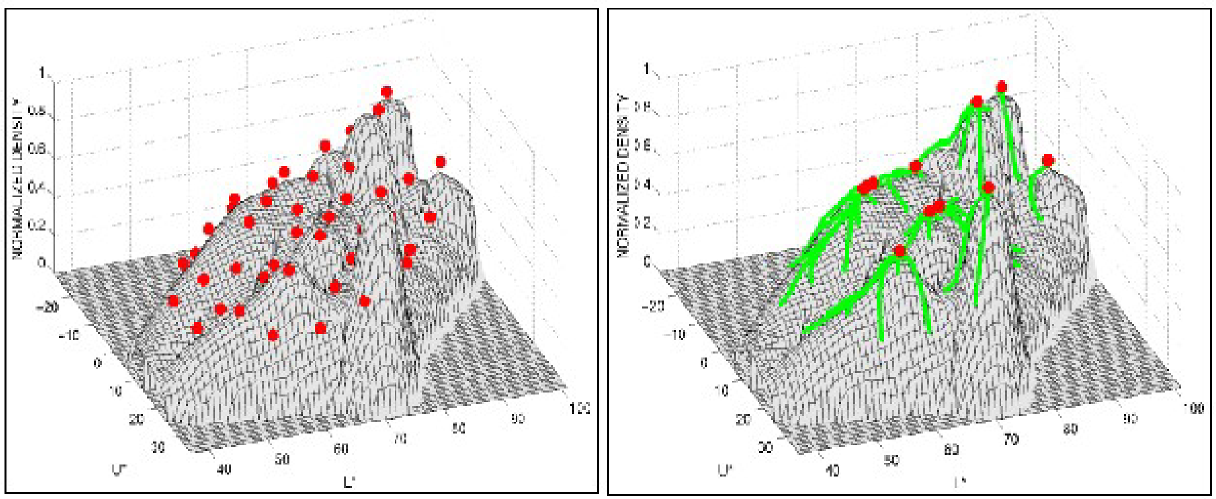


- Iterative Mode Search

1. Initialize random seed, and window W
2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} x H(x)$
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



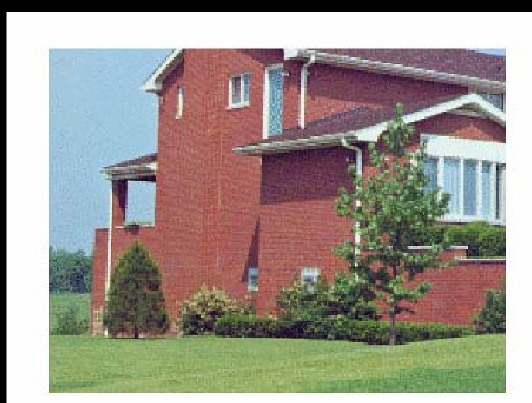
Mean shift trajectories

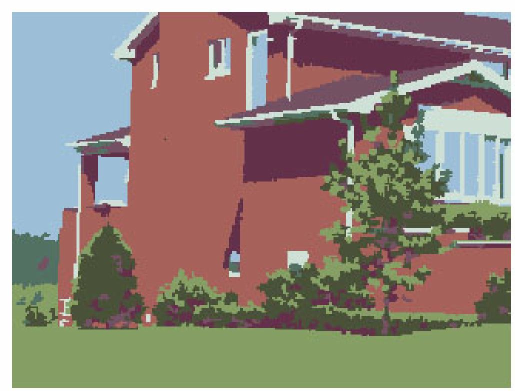
Segmentation Algorithm

- First run the mean shift procedure for each data point x and store its convergence point z.
- Link together all the z's that are closer than .5 from each other to form clusters
- Assign each point to its cluster
- Eliminate small regions

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





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

References

– Shi and Malik, "Normalized Cuts and Image Segmentation," Proc. CVPR 1997.

– Carson, Belongie, Greenspan and Malik, "Blobworld: Image Segmentation Using Expectation-Maximization and its Application to Image Querying," IEEE PAMI, Vol 24, No. 8, Aug. 2002.

– Comaniciu and Meer, "Mean shift analysis and applications," Proc. *ICCV* 1999.