Advanced Image Segmentation

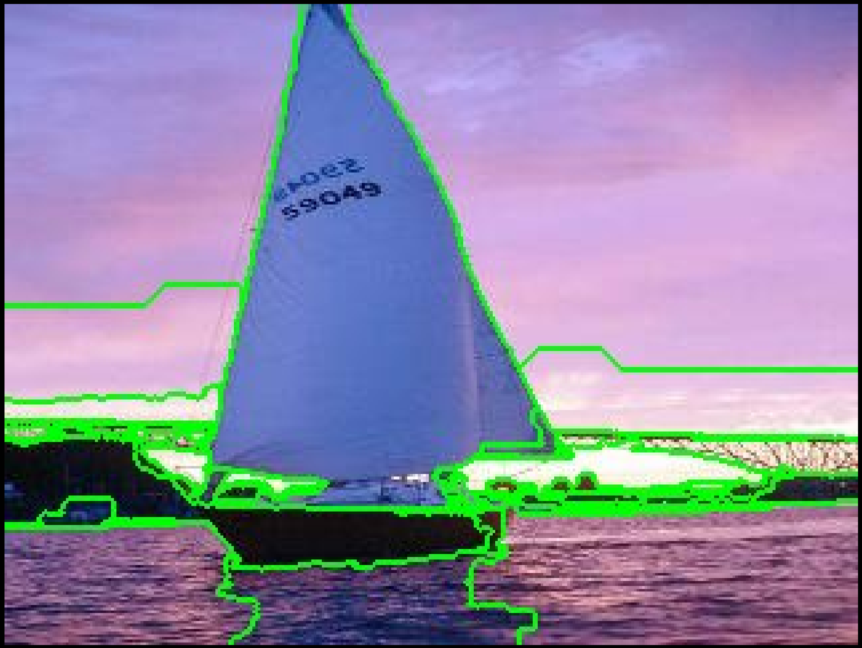
Image segmentation is the operation of partitioning an image into a collection of connected sets of pixels.

1. into regions, which usually cover the image

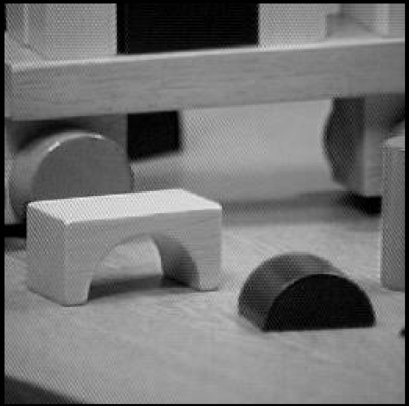
2. into linear structures, such as
   - line segments
   - curve segments

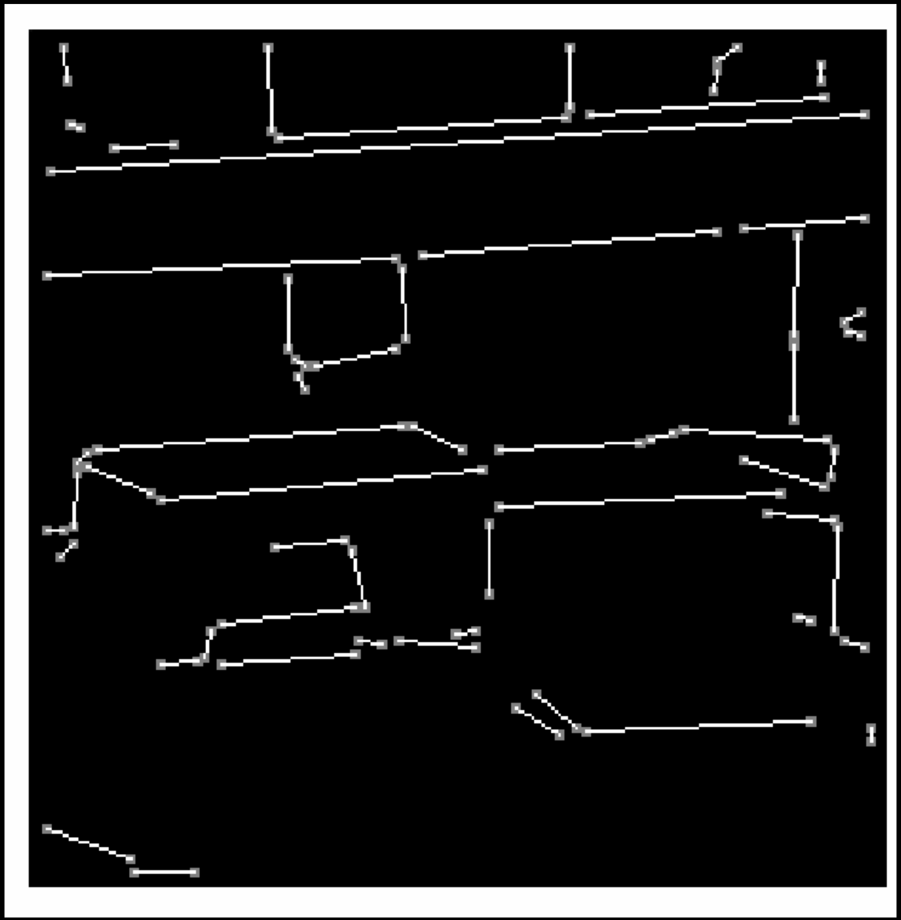
3. into 2D shapes, such as
   - circles
   - ellipses
   - ribbons (long, symmetric regions)

Example 1: Regions

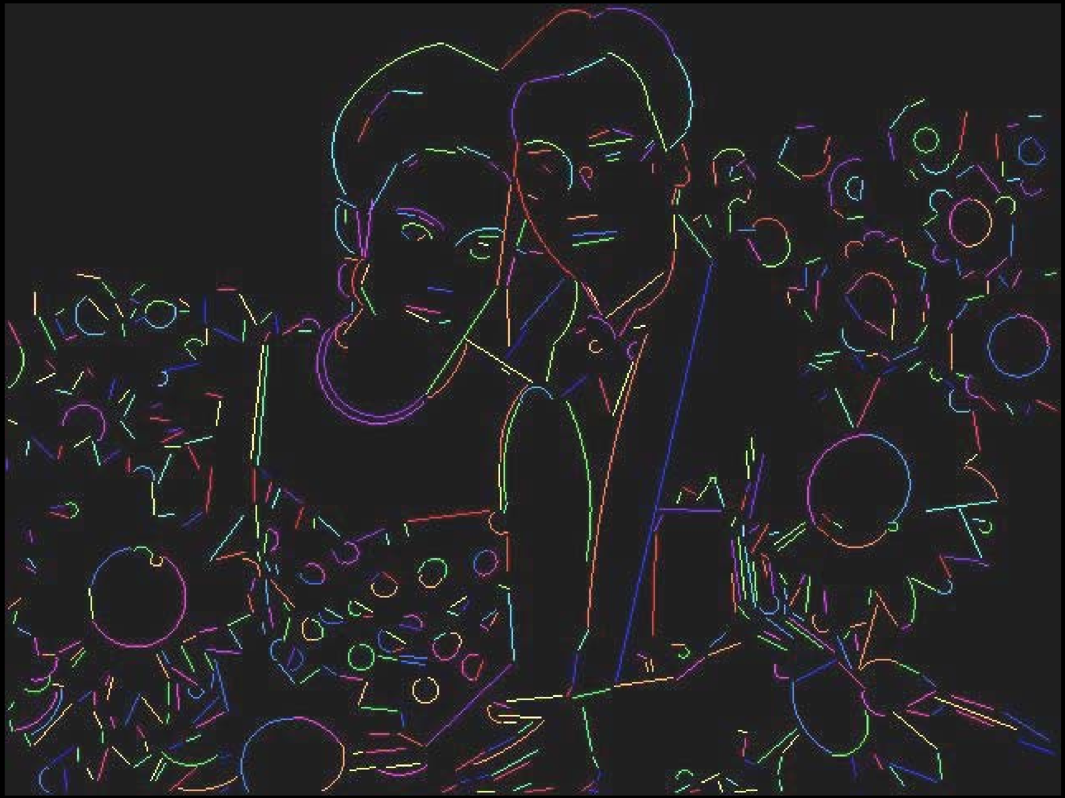


Example 2: Straight Lines





Example 3:
Lines and Circular Arcs



Region Segmentation: Segmentation Criteria

From Pavlidis

A segmentation is a partition of an image I into a set of regions S satisfying:

1. $\cup$ Si = S — Partition covers the whole image.
2. Si $\cap$ Sj = ϕ, i $\neq$ j — No regions intersect.
3. $\forall$ Si, P(Si) = true — Homogeneity predicate is satisfied by each region.
4. P(Si $\cup$ Sj) = false, i $\neq$ j, Si adjacent Sj — Union of adjacent regions does not satisfy it.

So

So all we have to do is define and implement the similarity predicate.

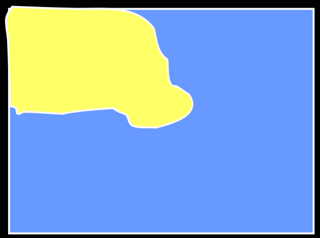
But, what do we want to be similar in each region?

Is there any property that will cause the regions to be meaningful objects?

Main Methods of Region Segmentation

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

Region Growing



Region growing techniques start with one pixel of a potential region and try to grow it by adding adjacent pixels till the pixels being compared are too disimilar.

- The first pixel selected can be just the first unlabeled pixel in the image or a set of seed pixels can be chosen from the image.

- Usually a statistical test is used to decide which pixels can be added to a region.

The RGGROW Algorithm

- Let R be the N pixel region so far and P be a neighboring pixel with gray tone y.

- Define the mean $\bar{X}$ and scatter S^2 (sample variance) by

$$\bar{X} = 1/N \sum_{(r,c) \in R} I(r,c)$$

$$S^2 = 1/N \sum_{(r,c) \in R} (I(r,c) - \bar{X})^2$$

The RGGROW Statistical Test

The T statistic is defined by

$$T = \left[\frac{(N-1) * N}{(N+1)} (y - \bar{X})^2 / S^2 \right]^{1/2}$$

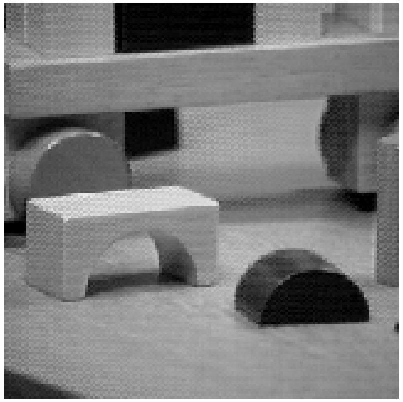
It has a T_{N-1} distribution if all the pixels in R and the test pixel y are independent and identically distributed normals (IID assumption) .

Decision and Update

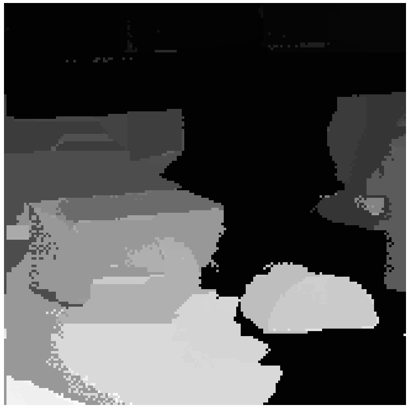
- For the T distribution, statistical tables give us the probability $Pr(T \leq t)$ for a given degrees of freedom and a confidence level. From this, pick suitable threshold t.
- If the computed $T \leq t$ for desired confidence level, add y to region R and update $\overline{X}$ and S^2.
- If T is too high, the value y is not likely to have arisen from the population of pixels in R. Start a new region.

RGGROW Example

image



segmentation



Not so great and
it's order dependent.

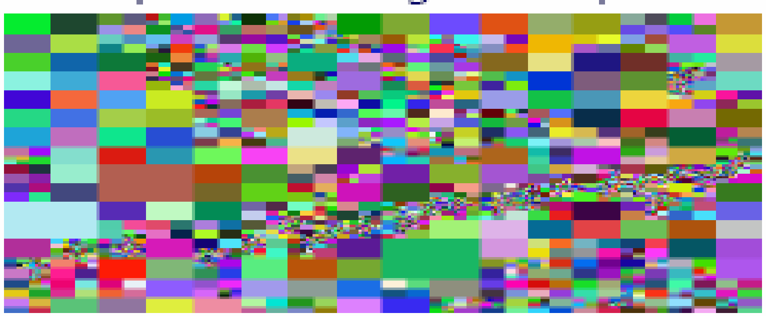
Split and Merge

1. Start with the whole image
2. If the variance is too high, break into quadrants
3. Merge any adjacent regions that are similar enough.
4. Repeat Steps 2 and 3, iteratively till no more splitting or merging occur

Idea: Good
Results: Blocky

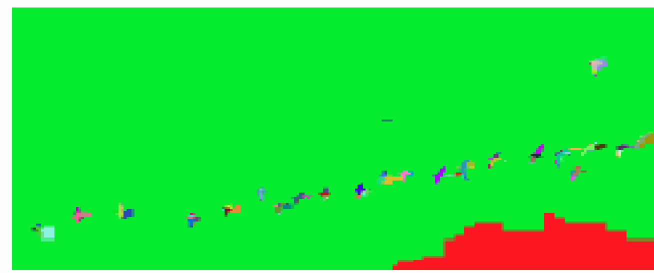
Split and merge example

Split and merge example



4087 regions

Split and merge example



135 regions

Clustering

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

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

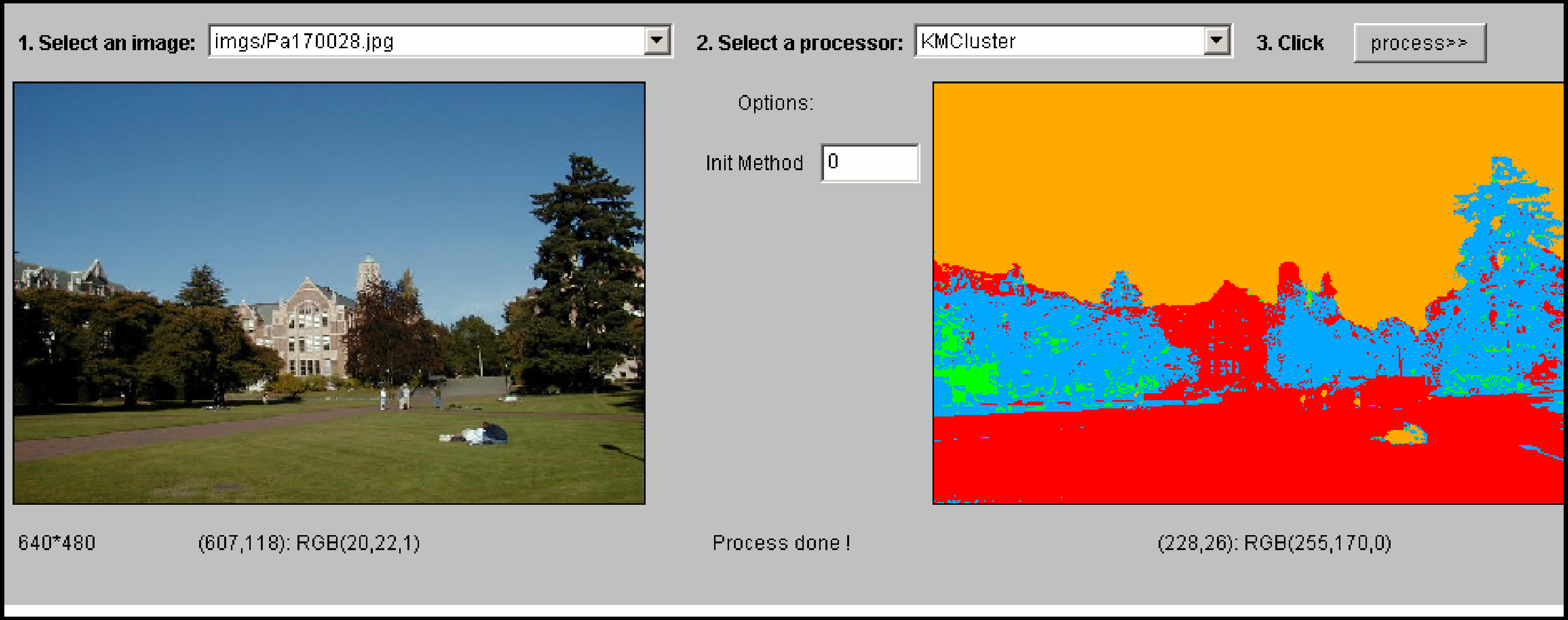
- Out of all possible partitions into K clusters, choose the one that minimizes D.

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

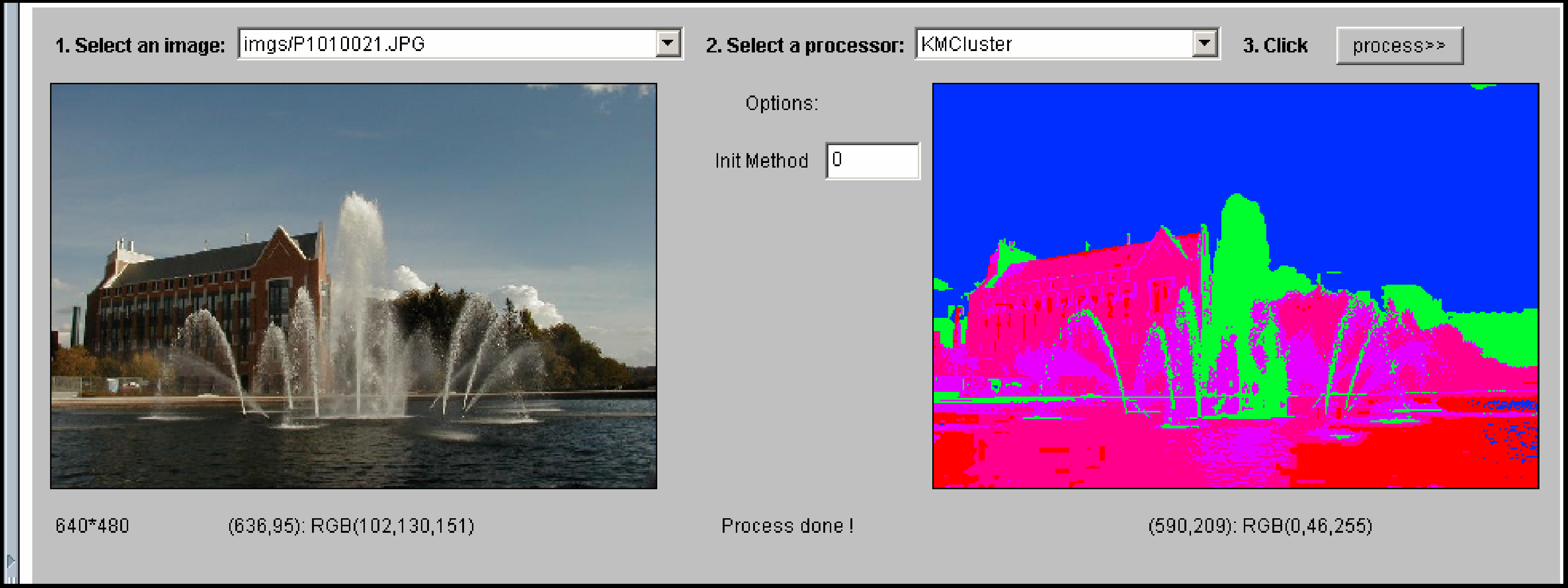
Some Clustering Methods

- K-means Clustering and Variants
- Histogram-Based Clustering and Recursive Variant
- Graph-Theoretic Clustering
- EM Clustering

K-Means Example 1



K-Means Example 2



K-means Variants

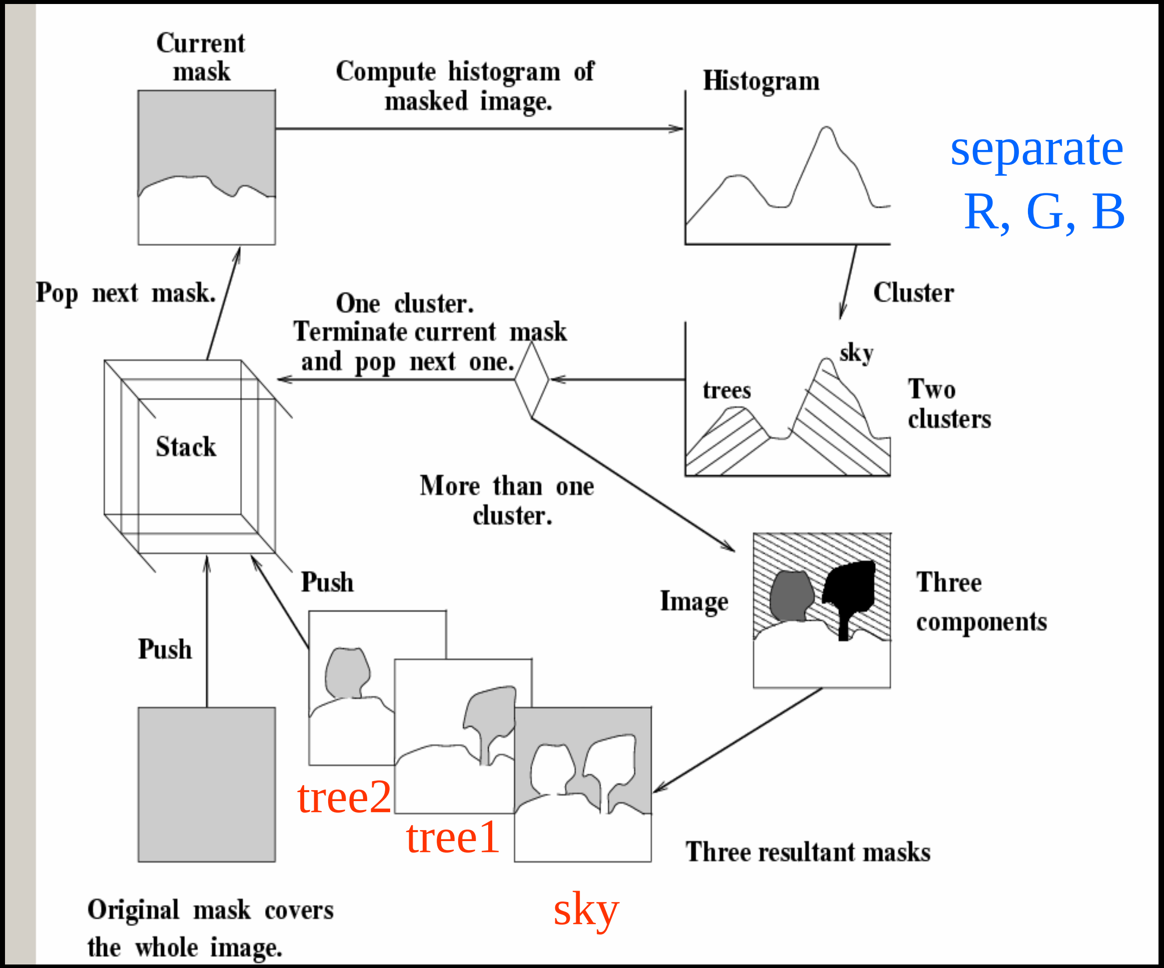
- Different ways to initialize the means
- Different stopping criteria
- Dynamic methods for determining the right number of clusters (K) for a given image

Ohlander's Recursive Histogram-Based Clustering

- color images of real indoor and outdoor scenes
- starts with the whole image
- selects the R, G, or B histogram with largest peak and finds clusters from that histogram
- converts to regions on the image and creates masks for each
- pushes each mask onto a stack for further clustering

Ohlander's Method

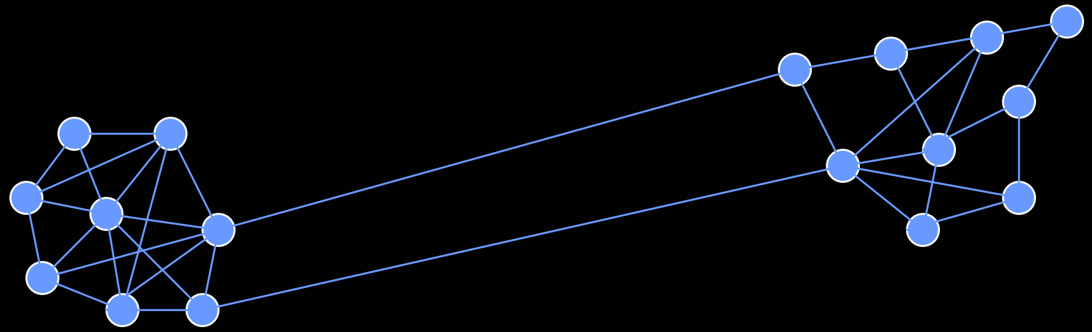
Ohta suggested using (R+G+B)/3, (R-B)/2 and (2G-R-B)/4 instead of (R, G, B).



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.

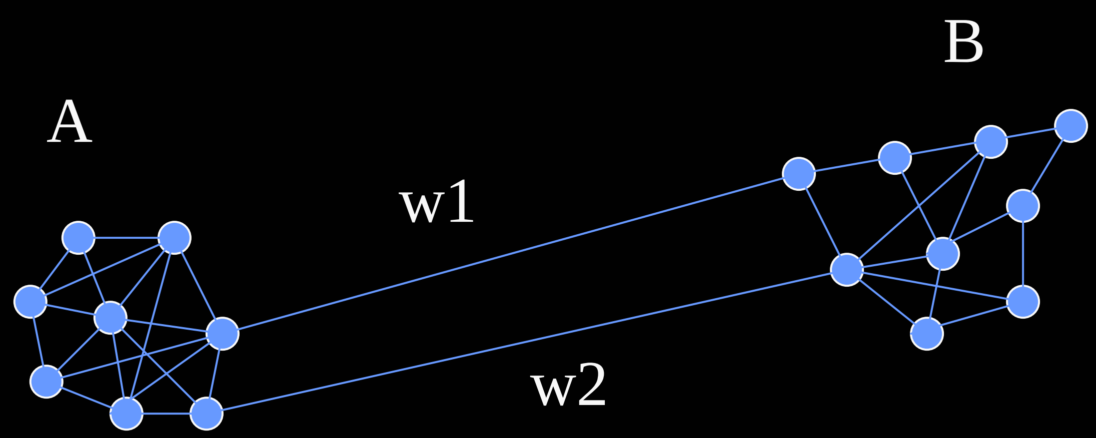
- Graph G can be broken into 2 disjoint graphs with node sets A and B by removing edges that connect these sets.

- 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.

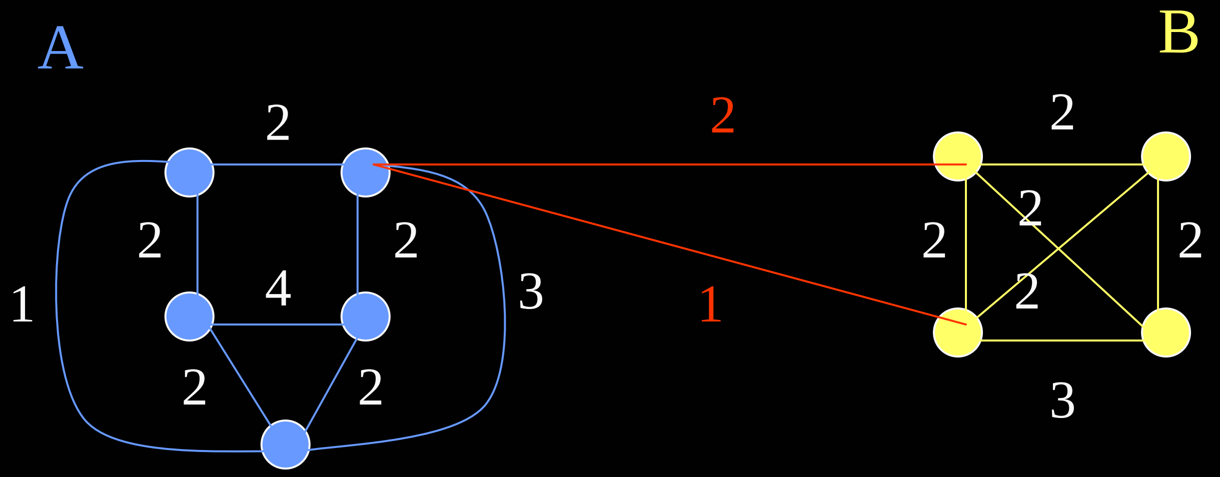
$$\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A, B)}{\mathrm{asso}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{asso}(B,V)}$$

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



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

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:

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

- 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, IEEE Conf. Computer Vision and Pattern Recognition(CVPR), 1997

How Shi used the procedure

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

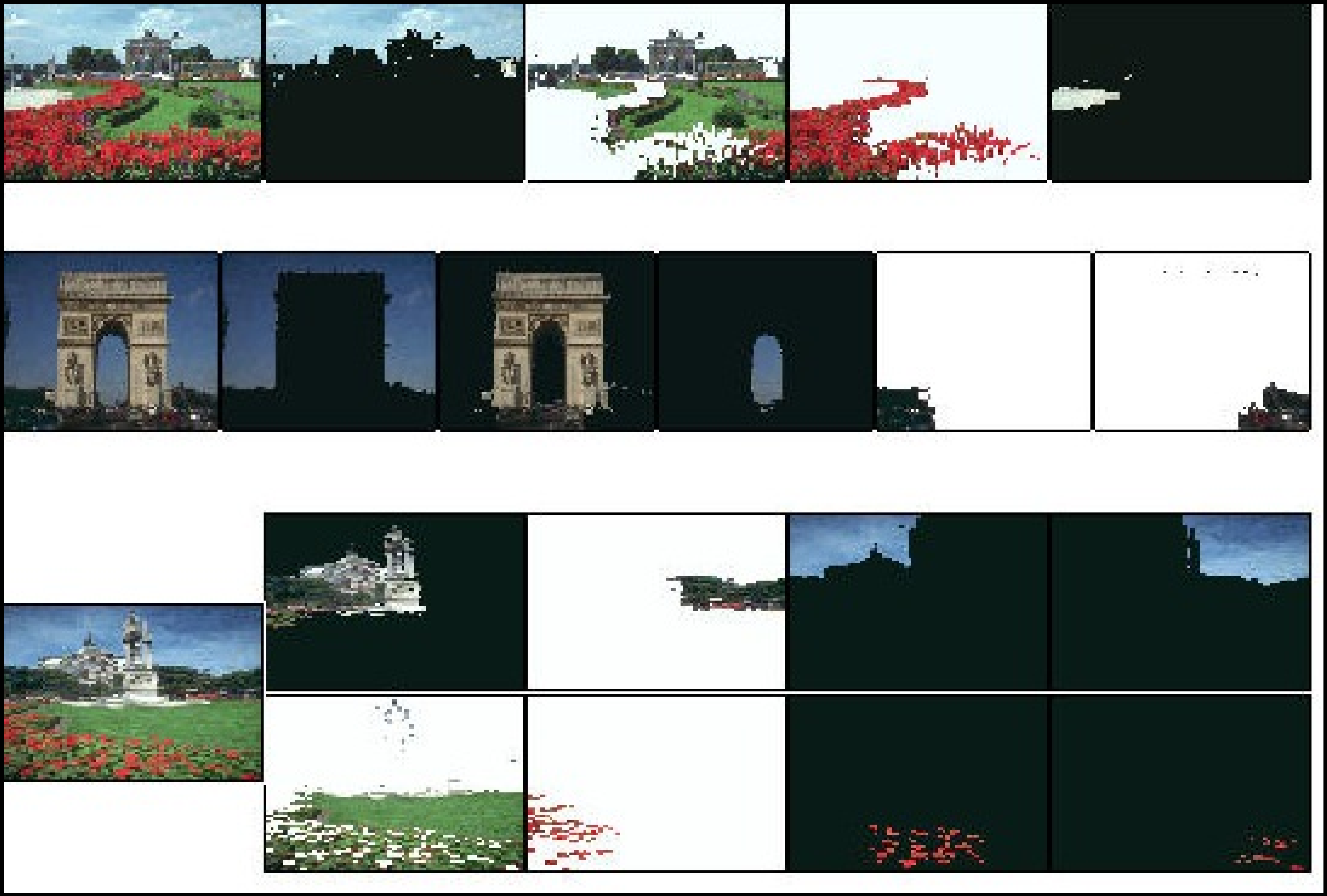
$$w(i,j) = e^{\|F(i)-F(j)\|_2 / \sigma I} * \begin{cases} e^{\|X(i)-X(j)\|_2 / \sigma X} & \text{if } \|X(i)-X(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

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-2.cs.cmu.edu/~jshi



EM Algorithm and its Applications

Prepared by Yi Li

Department of Computer Science and Engineering

University of Washington

From K-means to EM is from discrete to probabilistic

K-means revisited

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

1. Set ic (iteration count) to 1

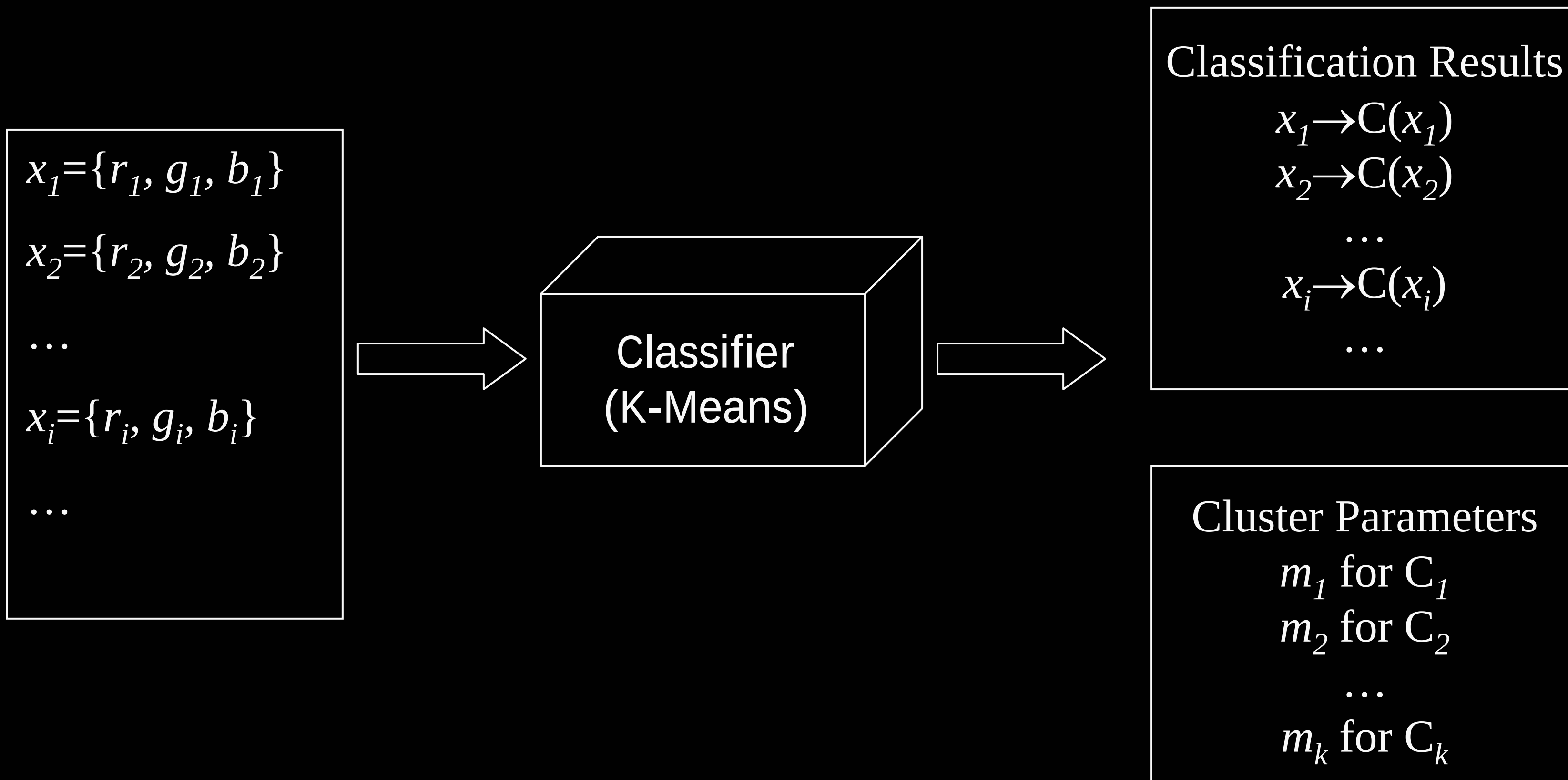
2. Choose randomly a set of K means $m_1(1), \ldots, m_K(1)$.

3. For each vector x_i, compute $D(x_i, m_k(ic)), k=1,\ldots K$ and assign x_i to the cluster C_j with nearest mean.

4. Increment ic by 1, update the means to get $m_1(ic),\ldots,m_K(ic)$.

5. Repeat steps 3 and 4 until $C_k(ic) = C_k(ic+1)$ for all k.

K-Means Classifier



K-Means 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

m_1 for C_1

m_2 for C_2

…

m_k for C_k

Classification Results

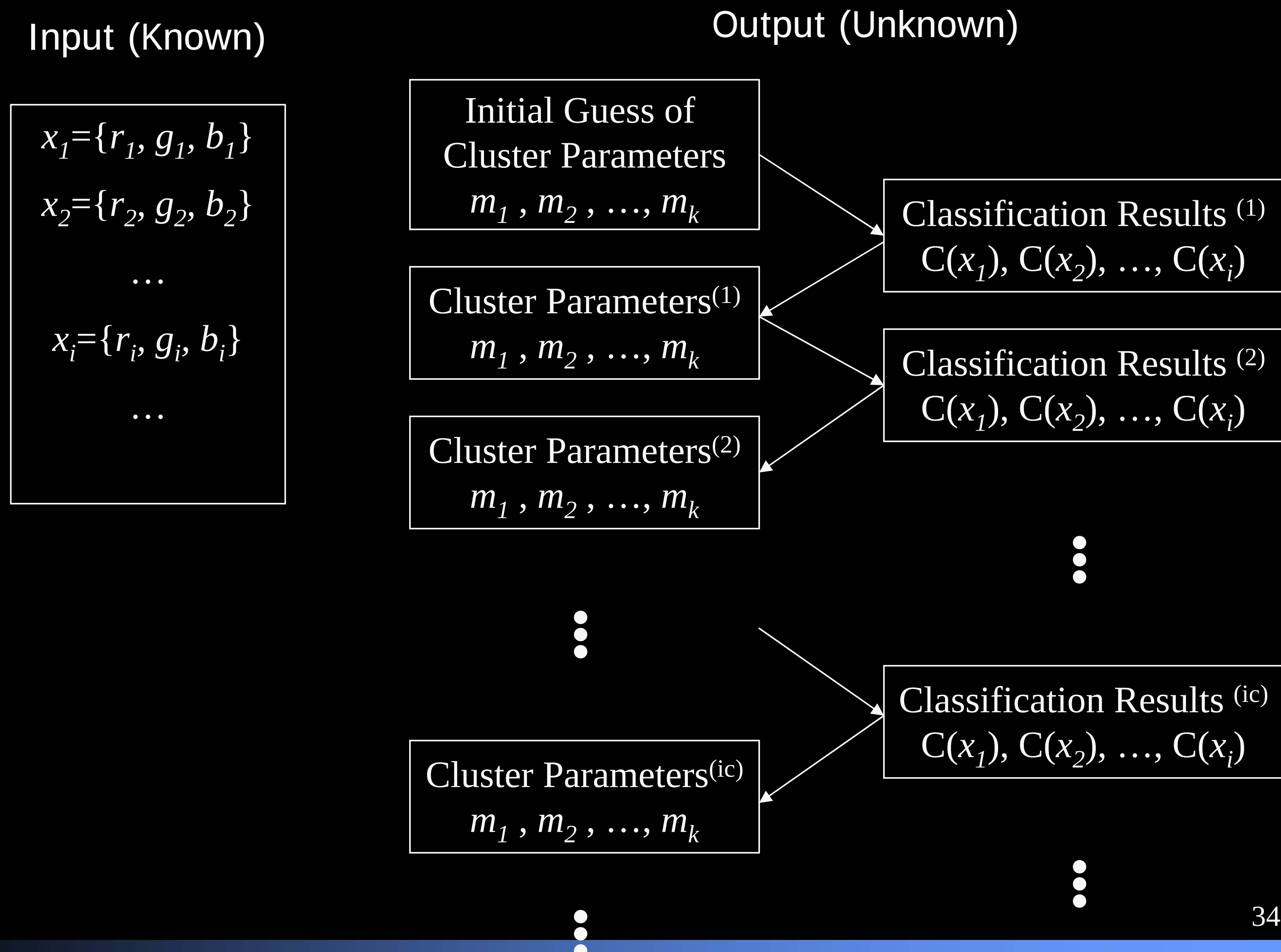
$x_1 \rightarrow C(x_1)$

$x_2 \rightarrow C(x_2)$

…

$x_i \rightarrow C(x_i)$

…


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 (Cont.)

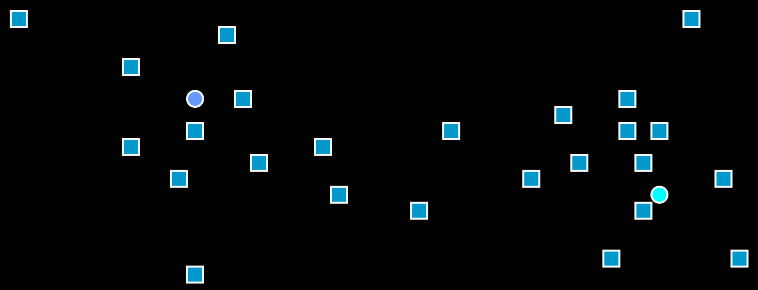
- Boot Step:
  - Initialize K clusters: $C_1, \ldots, C_K$

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

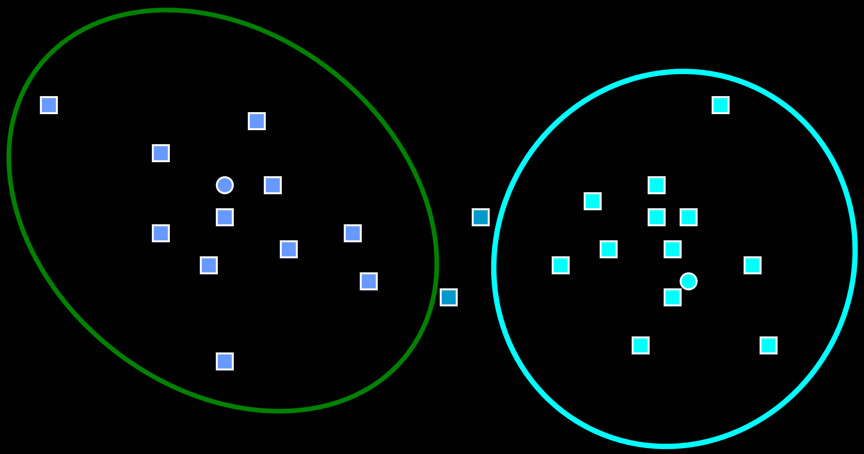
    $$x_i \Longrightarrow C(x_i)$$
  - Re-estimate the cluster parameters

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

K-Means Example



K-Means Example



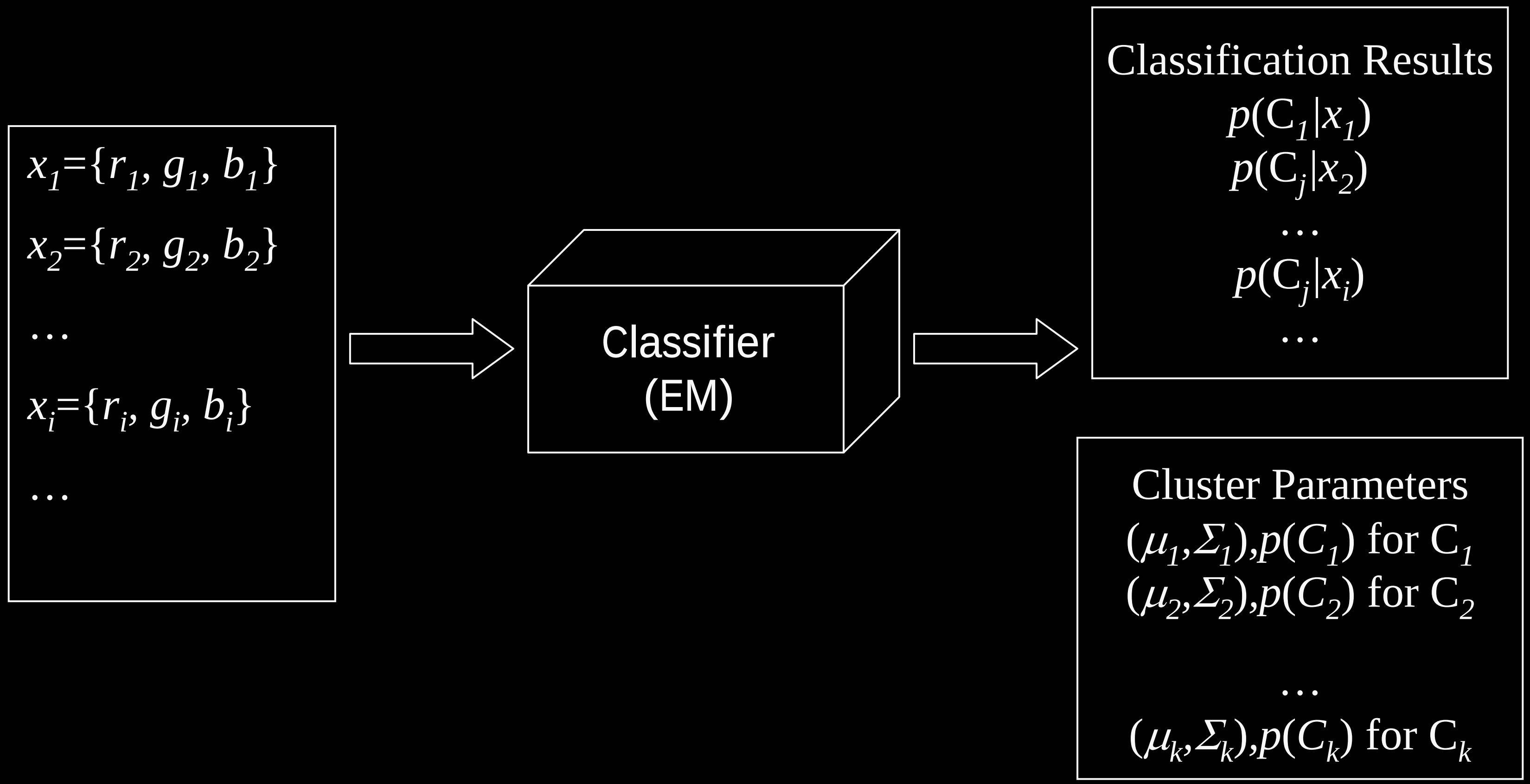
K-Means → EM

- <u>Boot Step</u>:
  - Initialize K clusters: $C_1, \ldots, C_K$

    (μ_j, Σ_j) and $P(C_j)$ for each cluster j.
- <u>Iteration Step</u>:
  - 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, …, 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 using EM

- Step 1: Feature Extraction
- Step 2: Image Segmentation using EM

Symbols

- 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