Image Segmentation

Jianping Fan
Dept of Computer Science
UNC-Charlotte

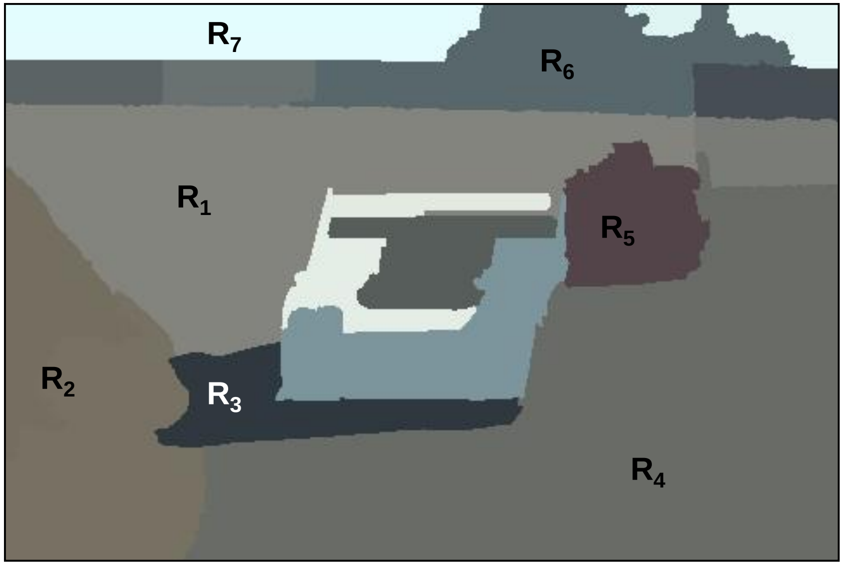
Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Outlines

- **Seeded Region Growing**
- **K-Means Clustering**
- **Graph Cut & Normalized Cut**

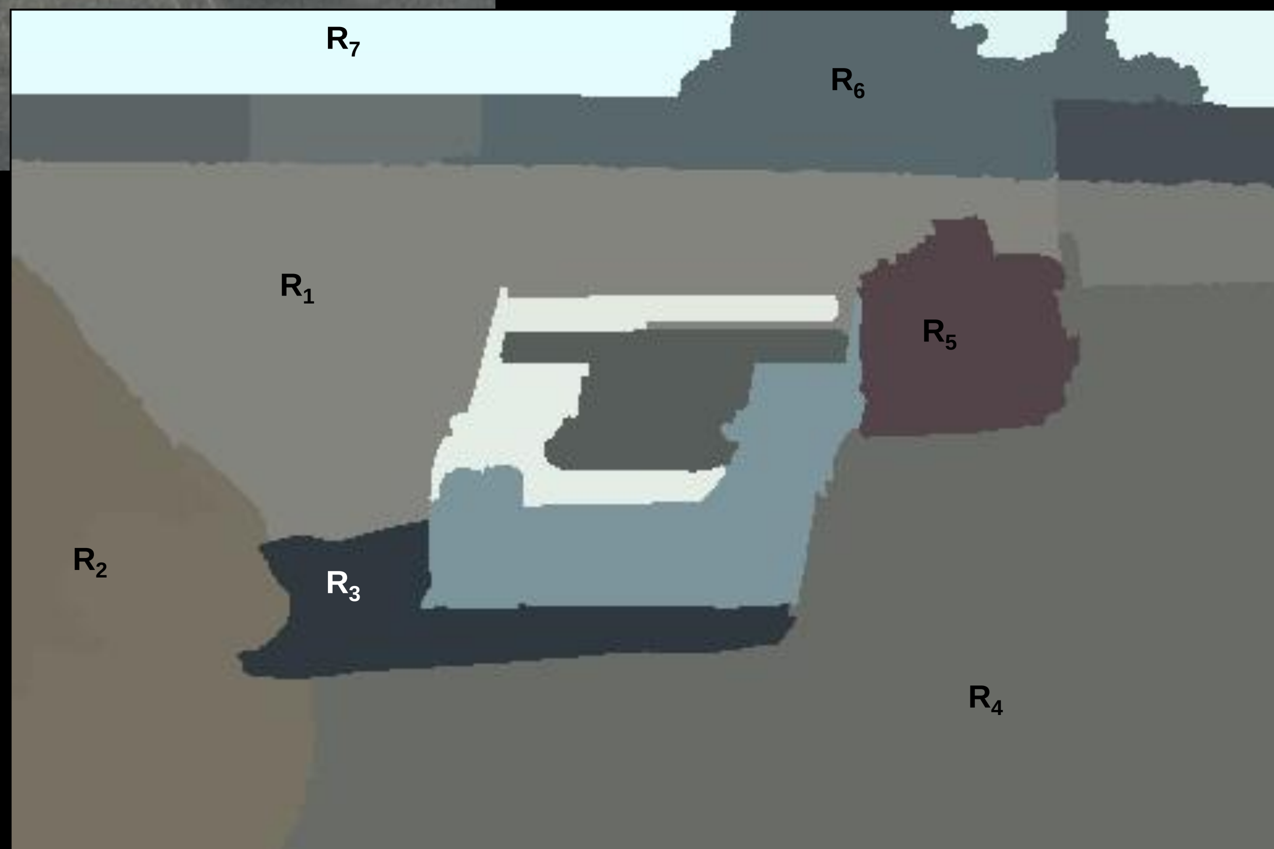
Definitions

- Based on *sets*.
- Each image R is a set of regions R_i.
  - Every pixel belongs to one region.
  - One pixel can only belong to a single region.



$$R = \bigcup_{i=1}^{S} R_i \qquad R_i \bigcap R_j = \varnothing$$


CASTROL
CASTROL
Castrol
96
33
96


R_7
R_6
R_1
R_5
R_2
R_3
R_4

Basic Formulation

Let R represent the entire image region. Segmentation partitions R into n subregions, R_1, R_2, ..., R_n, such that:

a) $\bigcup_{i=1}^{n} R_i = R$

b) R_i is a connected region, $i = 1, 2, \ldots, n$.

c) $R_i \cap R_j = \phi$ for all i and $j, i \neq j$

d) $P(R_i) = TRUE$ for $i = 1, 2, \ldots, n$.

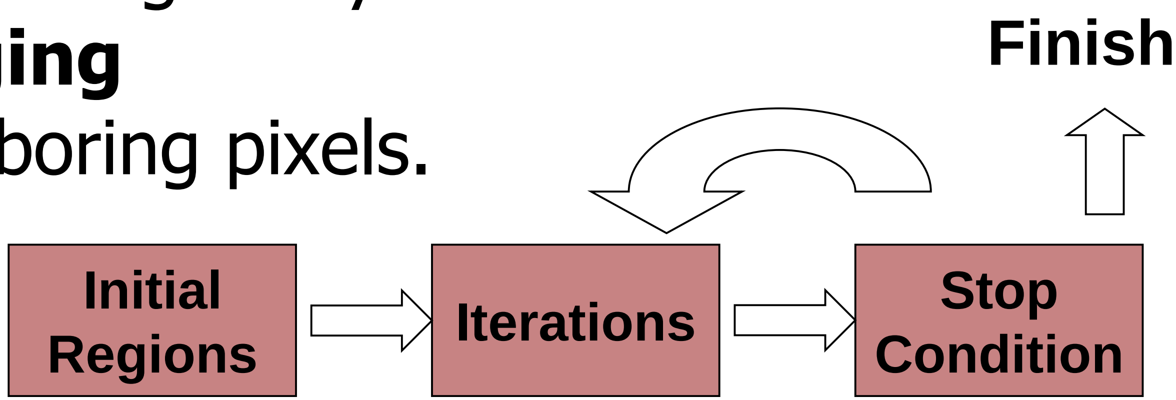
e) $P(R_i \cup R_j) = FALSE$ for $i \neq j$.

a) Every pixel must be in a region
b) Points in a region must be connected.
c) Regions must be disjoint.
d) All pixels in a region satisfy specific properties.
e) Different regions have different properties.

Region growing

- Groups pixels into larger regions.
- Starts with a **seed** region.
- **Grows** region by **merging** neighboring pixels.

- Iterative process
  - How to start?
  - How to iterate?
  - When to stop?



Similarity Criteria

- Homogeneity of regions is used as the main segmentation criterion in region growing.
  - gray level
  - color, texture
  - shape
  - model
  - etc.

Choice of criteria affects segmentation results dramatically!

Region Growing

- Inter-Pixel Similarity Calculation

  (a) Pixel Neighborhood Similarity Calculation

$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$

Region Growing

- Inter-Pixel Similarity Calculation

(1) Inter-Pixel **Similarity**

$$D[(x,y),(x-1,y-1)] = | I(x,y) - I(x-1,y-1) |$$

(2) Binary Classification

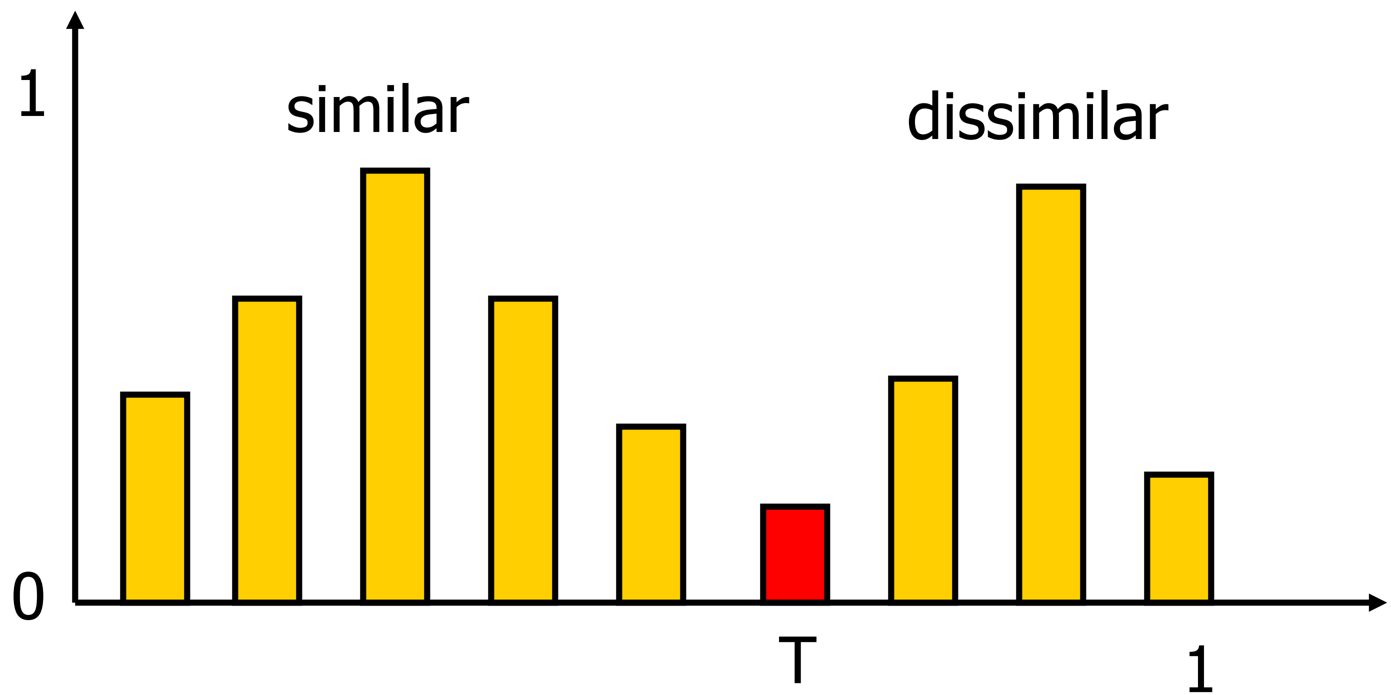
$$S(x,y) = \begin{cases} 1, similar, D[(x,y),(x-1,y-1)] < T \\ 0, dissimilar, D[(x,y),(x-1,y-1)] \geq T \end{cases}$$

Region Growing

- Threshold Determination for Decision Making

  Relationships among neighboring pixels can defined as:

  similar versus **dissimilar**



Region Growing

- Threshold Determination for Decision Making

Entropy for similar and dissimilar pixels:

$$H(\overline{T}) = \max_{T=0,1,\ldots,M} \{H_{nsc}(T) + H_{sc}(T)\}.$$

$$P_{nsc}(i) = \frac{f_i}{\sum_{h=0}^{T} f_h}, \quad 0 \leqslant i \leqslant T,$$

Region Growing

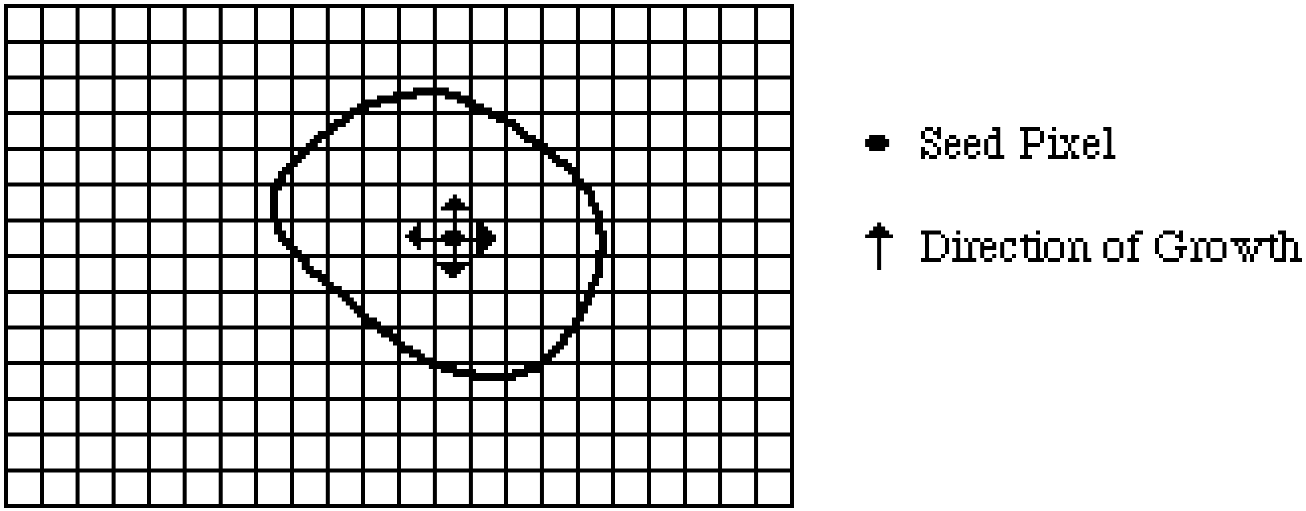
- Threshold Determination for Decision Making

$$
\begin{aligned}
H_{\text{nsc}}(T+1) &= -\sum_{i=0}^{T+1} \frac{f_i}{P_0(T+1)} \log \frac{f_i}{P_0(T+1)} \\
&= -\frac{P_0(T)}{P_1(T+1)} \sum_{i=0}^{T+1} \frac{f_i}{P_0(T)} \log\left\{ \frac{f_i}{P_0(T)} \frac{P_0(T)}{P_0(T+1)} \right\} \\
&= \frac{P_0(T)}{P_0(T+1)} H_{\text{nsc}}(T) - \frac{f_{T+1}}{P_0(T+1)} \log \frac{f_{T+1}}{P_0(T+1)} \\
&\quad - \frac{P_0(T)}{P_0(T+1)} \log \frac{P_0(T)}{P_0(T+1)},
\end{aligned}
$$

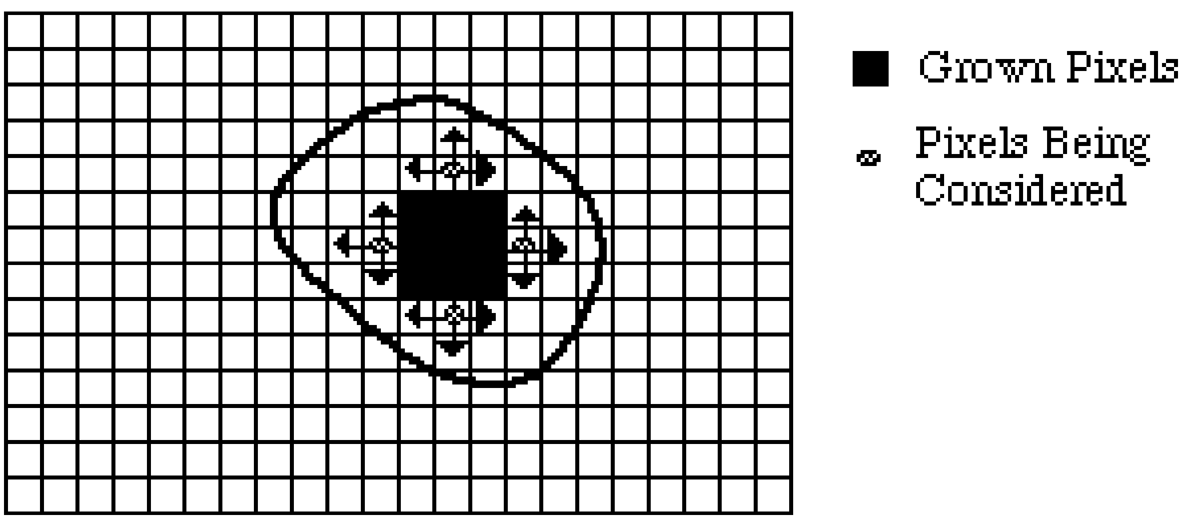
$$
\begin{aligned}
H_{\text{sc}}(T+1) &= -\sum_{i=T+2}^{M} \frac{f_i}{P_1(T+1)} \log \frac{f_i}{P_1(T+1)} \\
&= -\frac{P_1(T)}{P_1(T+1)} \sum_{i=T+2}^{M} \frac{f_i}{P_1(T)} \log\left\{ \frac{f_i}{P_1(T)} \frac{P_1(T)}{P_1(T+1)} \right\} \\
&= \frac{P_1(T)}{P_1(T+1)} H_{\text{sc}}(T) + \frac{f_{T+1}}{P_1(T+1)} \log \frac{f_{T+1}}{P_1(T+1)} \\
&\quad - \frac{P_1(T)}{P_1(T+1)} \log \frac{P_1(T)}{P_1(T+1)}.
\end{aligned}
$$

Gray-Level Criteria

- Comparing to Original Seed Pixel
  - Very sensitive to choice of **seed point**.
- Comparing to Neighbor in Region
  - Allows gradual changes in the region.
  - Can cause significant drift.
- Comparing to Region Statistics
  - Acts as a **drift dampener**.
- Other possibilities!



(a) Start of Growing a Region



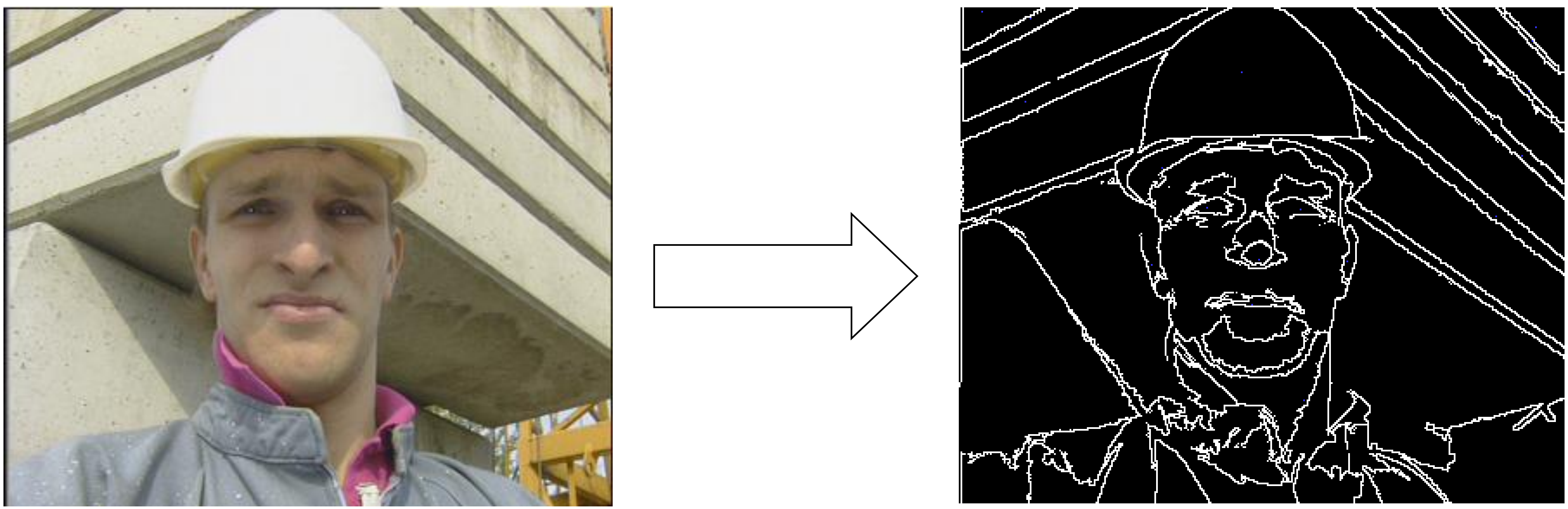
(b) Growing Process After a Few Iterations

Region merging

- Algorithm
  - Divide image into an initial set of regions.
    - One region per pixel.
  - Define a **similarity criteria** for merging regions.
  - **Merge** similar regions.
  - Repeat previous step until no more merge operations are possible.

Region Growing

- Region Growing Results



Region Growing

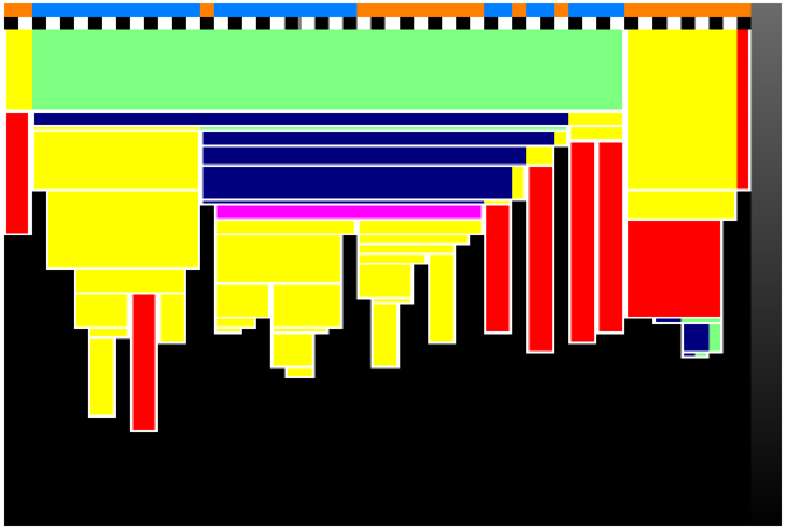
- Region Growing Results

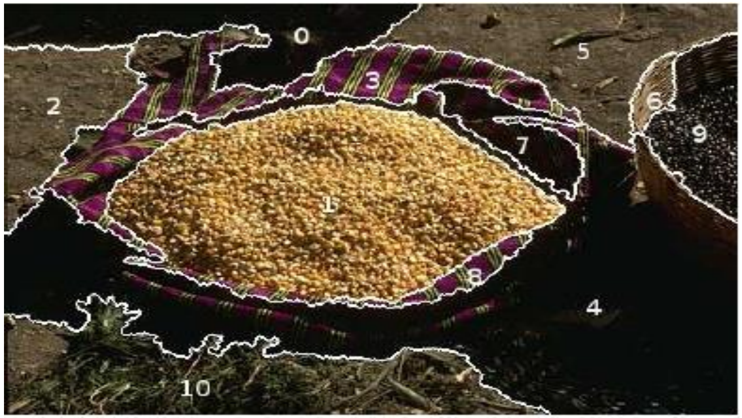
Region Growing

- **Region Growing Results**

Integrating Multi-modal Features for Region Growing

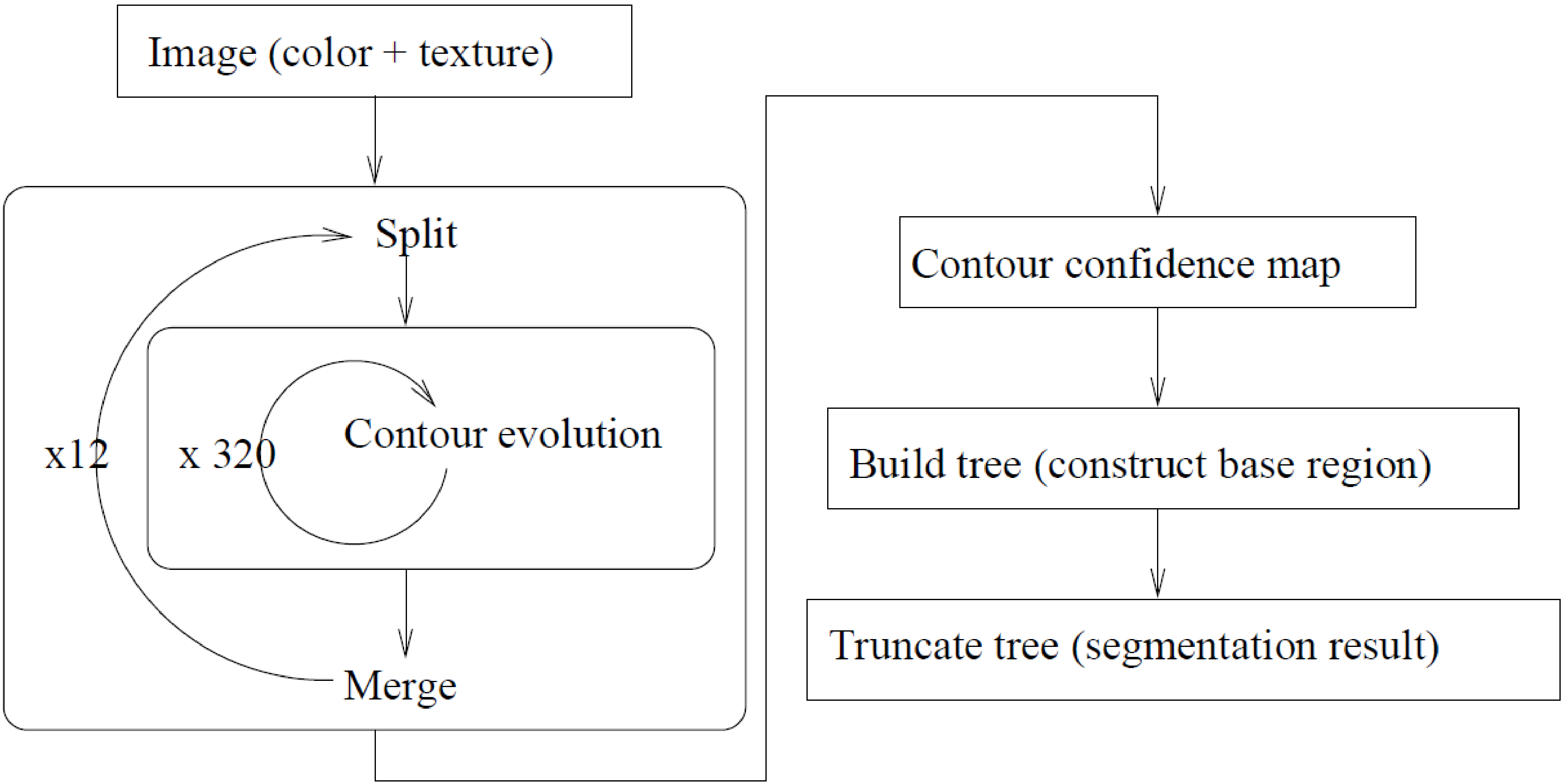
- **Image Segmentation**





Integrating Multi-modal Features for Region Growing

- **Image Segmentation**

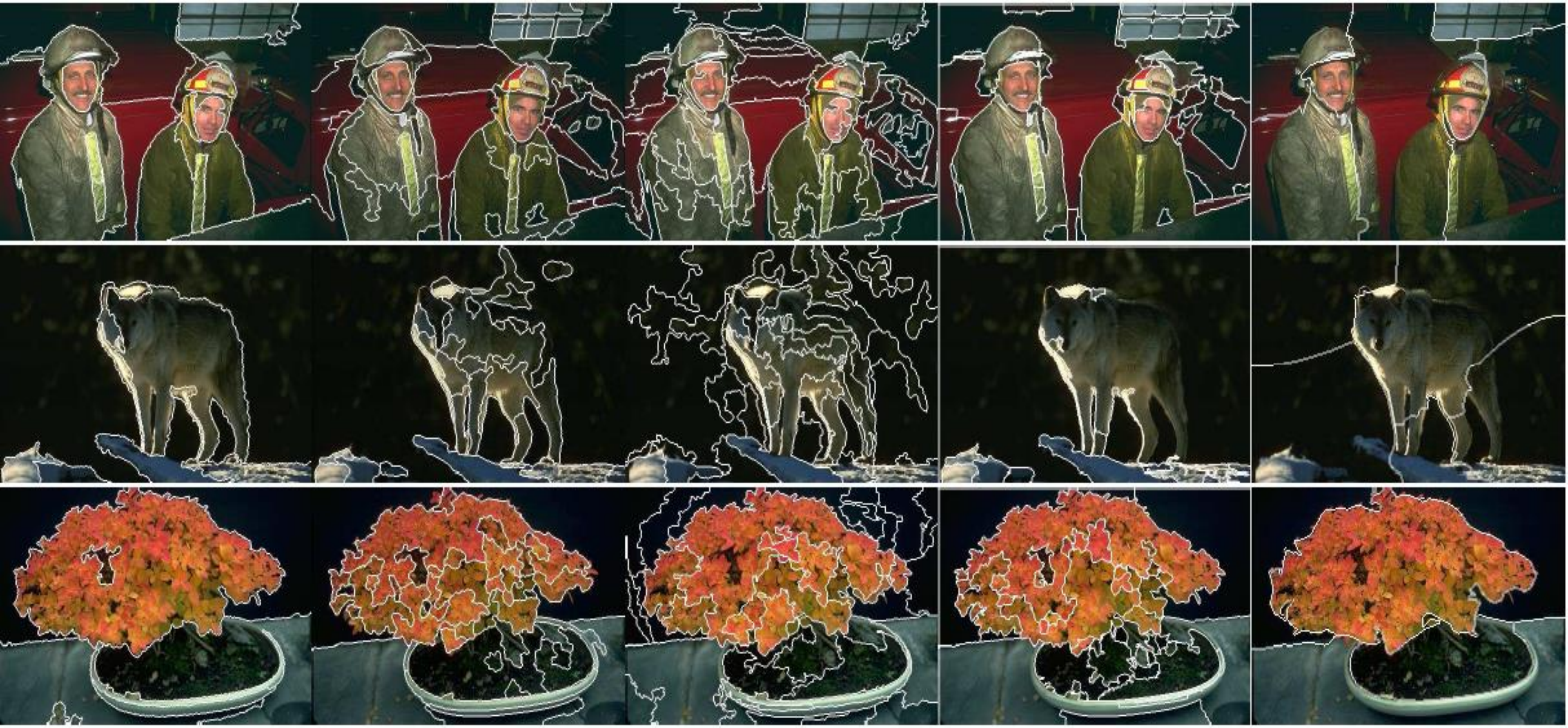


Integrating Multi-modal Features for Region Growing

- **Image Segmentation**

Integrating Multi-modal Features for Region Growing

- **Image Segmentation**



Integrating Multi-modal Features for Region Growing

- **Image Segmentation**

Integrating Multi-modal Features for Region Growing

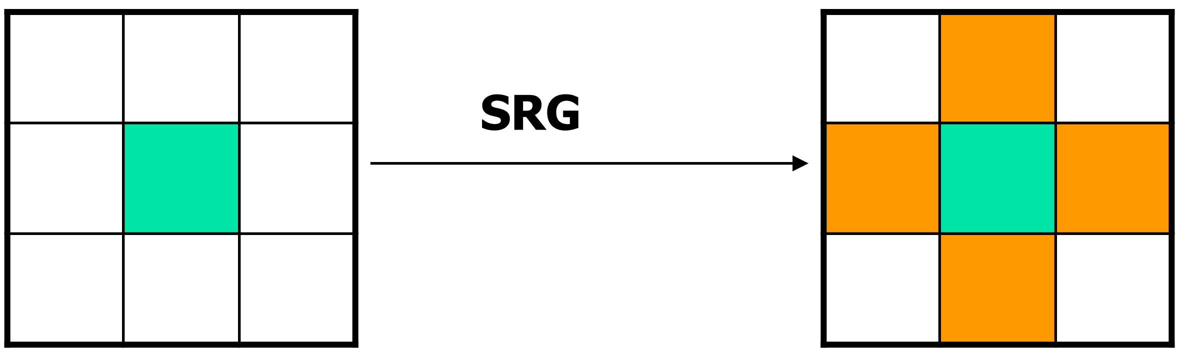
- **Observations**
  - **Only segmentation from visual information cannot support automatic image understanding & interpretation!**
  - **Image segmentation results may not make sense to human beings!**

Seeded Image Segmentation

- Color image segmentation polices:
  - Threshold
  - Boundary-based
  - Region-based
  - Hybrid techniques

Hybrid techniques

- Seeding region growing (SRG)
  - Different merging order possibility



Automatic seed selection algorithm

- Condition 1:
  - A seed pixel candidate must have the similarity higher than a threshold values.
- Condition 2:
  - A seed pixel candidate must have the maximum relative Euclidean distance to its eight neighbors less than a threshold value.

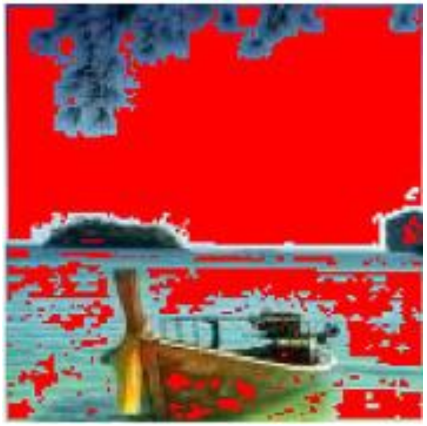
Automatic seed selection algorithm

- Calculate the maximum distance to its neighbors as : $d_{max} = \max_{i=1}^{8} d_i$
  - $$d_i = \frac{\sqrt{(Y - Y_i)^2 + (C_b - C_{b_i})^2 + (C_r - C_{r_i})^2}}{\sqrt{Y^2 + C_b^2 + C_r^2}}, i = 1...8$$
  - YC_bC_r of the pixel, $Y_iC_{b_i}C_{r_i}$ of its neighbors
  - Not on the boundary
  - 0.05

Condition 2:
A seed pixel candidate must have the maximum relative Euclidean distance to its eight neighbors less than a threshold value.

Automatic seed selection algorithm

- Connected seeds are considered as one seed.



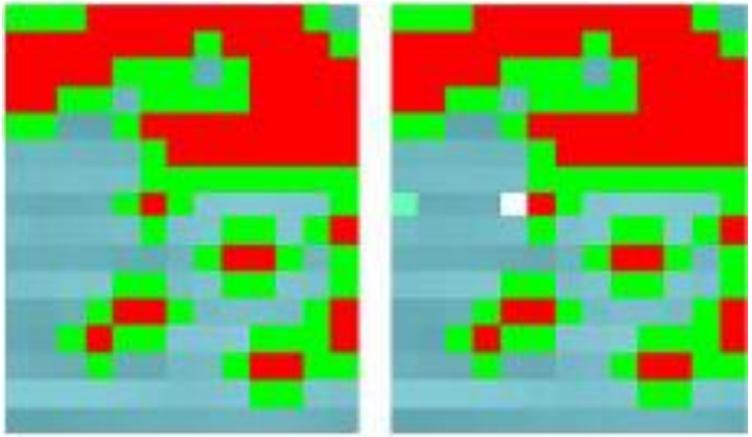
Original color image **the detected seeds are shown in red color**

Region growing

- The pixels that are unclassified and neighbors of at least one region, calculate the distance:

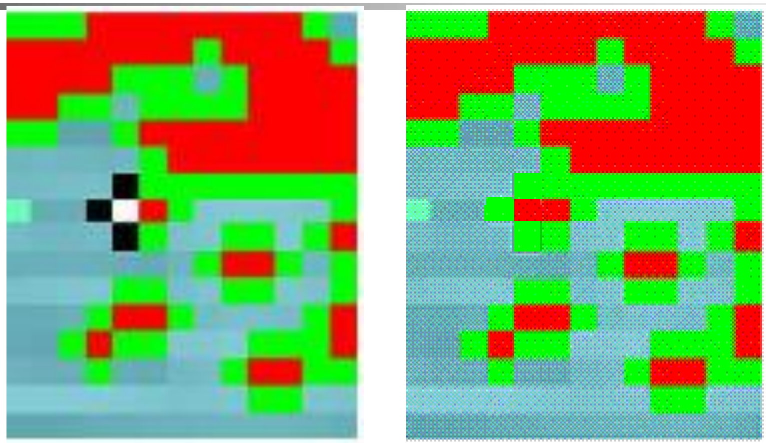
$$d_i = \frac{\sqrt{\left(Y_i - \overline{Y}\right)^2 + \left(C_{b_i} - \overline{C}_b\right)^2 + \left(C_{r_i} - \overline{C}_r\right)^2}}{\sqrt{Y_i^2 + C_{b_i}^2 + C_{r_i}^2}}$$

Region growing



1. red pixels are the seeds and the green pixels are the pixels in the sorted list T in a decreasing order of distances.
2. the white pixel is the pixel with the minimum distance to the seed regions
3. check its 4-neighbors

Region growing



1. If all labeled neighbors of p have a same label, set p to this label.
2. If the labeled neighbors of p have different labels, calculate the distances between p and all neighboring regions and classify p to the nearest region.
3. Then update the mean of this region, and add 4 neighbors of p, which are neither classified yet nor in T, to T in a decreasing order of distances.
4. Until the T is empty.

Region merging

- Consider the color different and size of regions :
  - Color different between two adjacent region Ri and Rj is defined as:

$$d\left(Ri,Rj\right)=\frac{\sqrt{\left(\overline{Y}_i-\overline{Y}_j\right)^2+\left(\overline{C}_{b_i}-\overline{C}_{b_j}\right)^2+\left(\overline{C}_{r_i}-\overline{C}_{r_j}\right)^2}}{\min\left(\sqrt{\overline{Y}_i^{\,2}+\overline{C}_{b_i}^{\,2}+\overline{C}_{r_i}^{\,2}},\sqrt{\overline{Y}_j^{\,2}+\overline{C}_{b_j}^{\,2}+\overline{C}_{r_i}^{\,2}}\right)}$$

  - Size
    - select $\frac{1}{150}$ of the total number of pixels in an image as the threshold.

Region merging

- We first examine the two regions having the smallest color different among others.
  - If $d(Ri, Rj) <$ threshold, merge the two regions and re-compute the mean of the new region.
  - We repeat the process until no region has the distance less than the threshold.
  - Threshold=0.1

Region merging

- If the size with number of pixels in a region is smaller than a threshold, the region is merged into its neighboring region with the smallest color difference.
  - This procedure is repeated until no region has size less than the threshold.

Experimental results

JSEG algorithm.

Experimental results

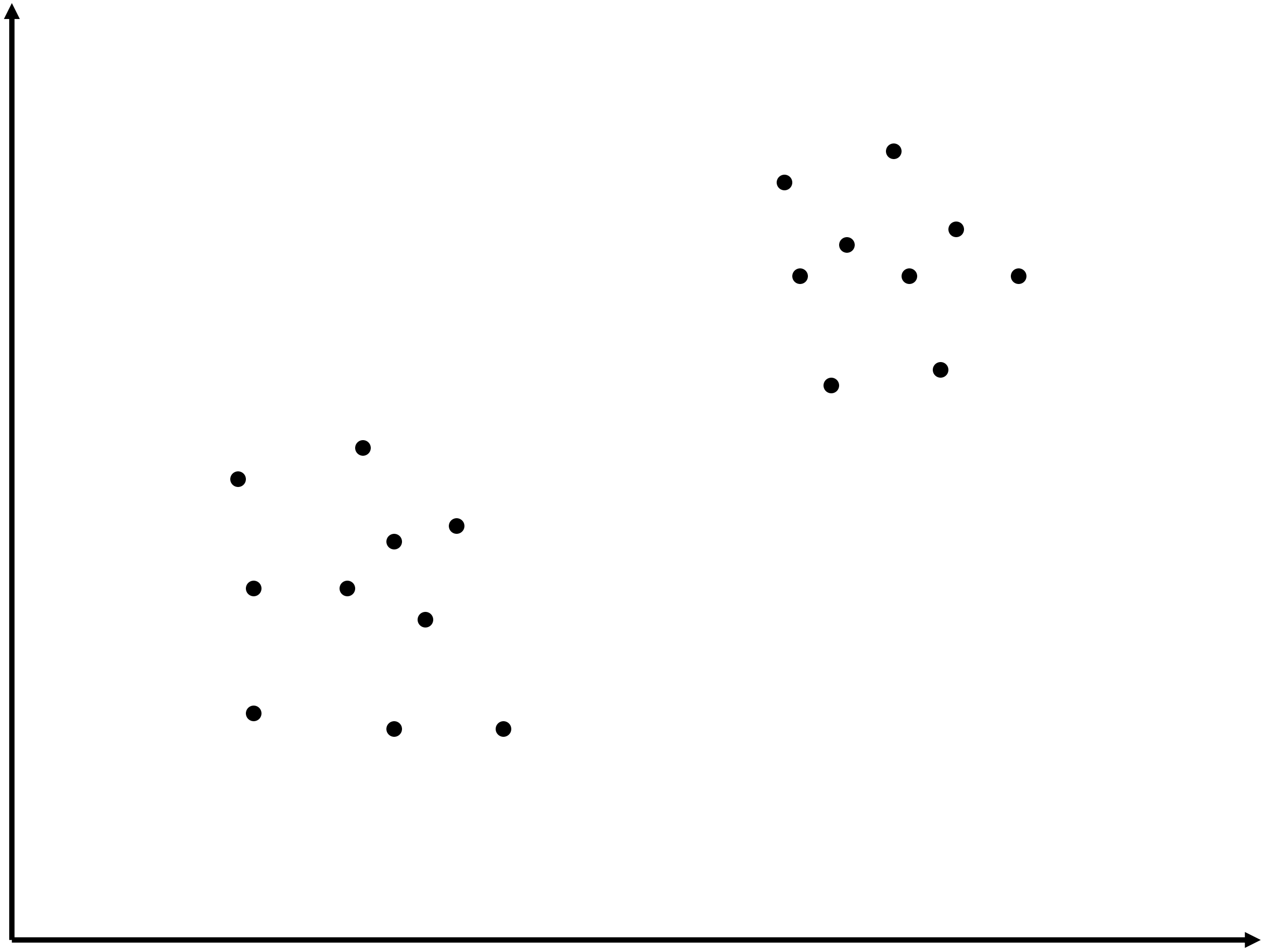
JSEG algorithm.

JSEG algorithm.

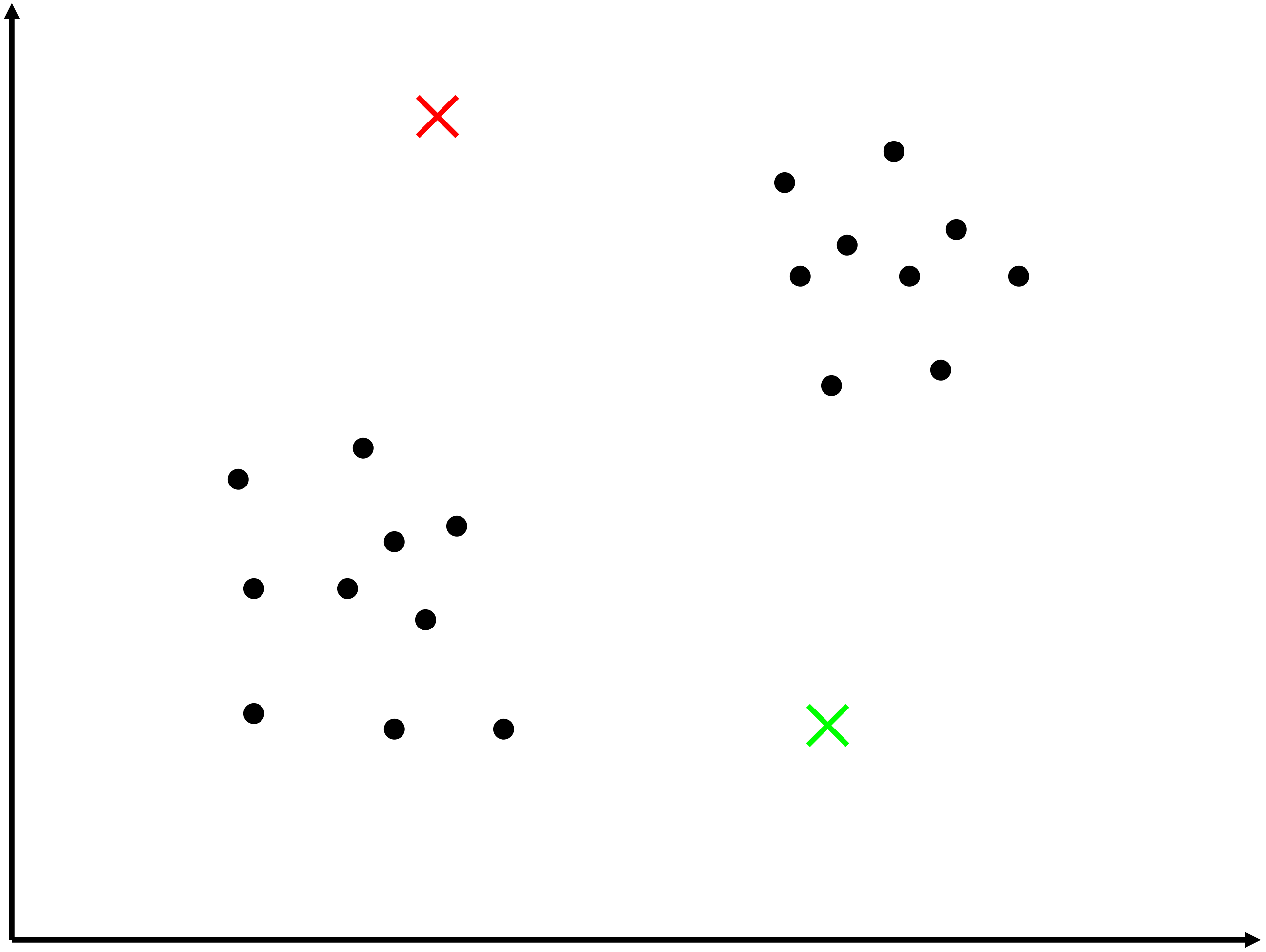
K-Means Clustering

1. **Partition the data points into K clusters randomly. Find the centroids of each cluster.**
2. **For each data point:**
   - **Calculate the distance from the data point to each cluster.**
   - **Assign the data point to the closest cluster.**
3. **Re-compute the centroid of each cluster.**
4. **Repeat steps 2 and 3 until there is no further change in the assignment of data points (or in the centroids).**

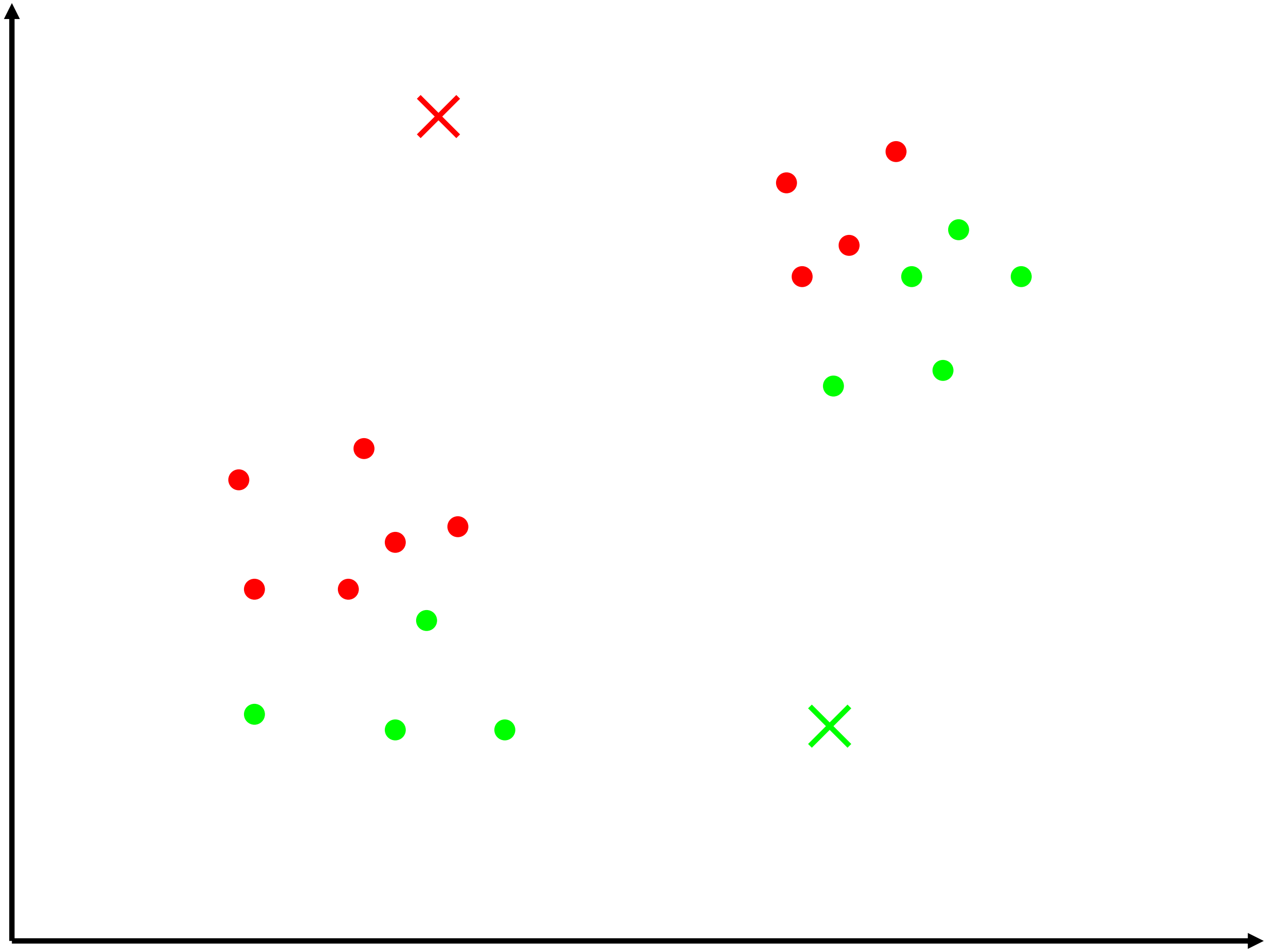
K-Means Clustering



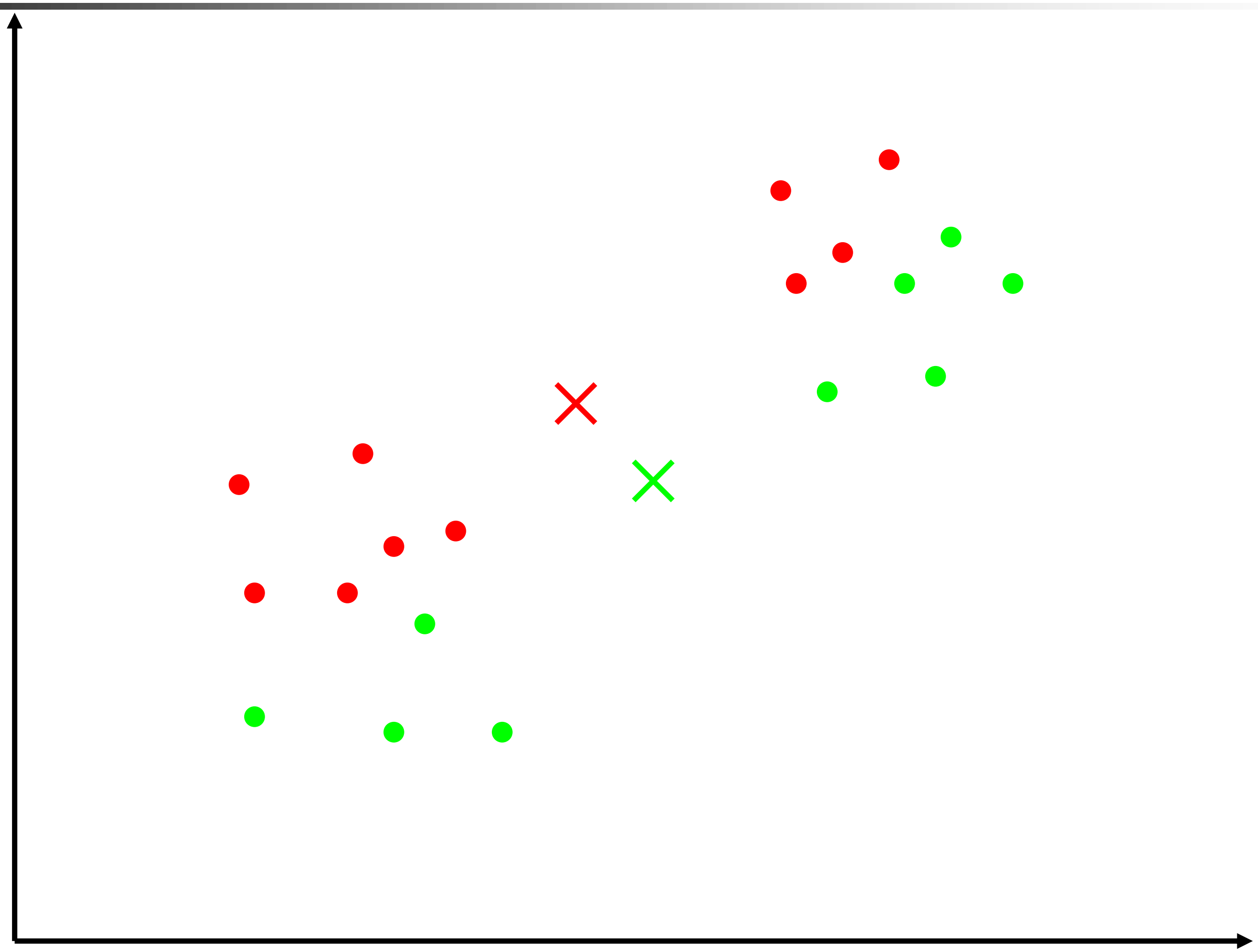
K-Means Clustering



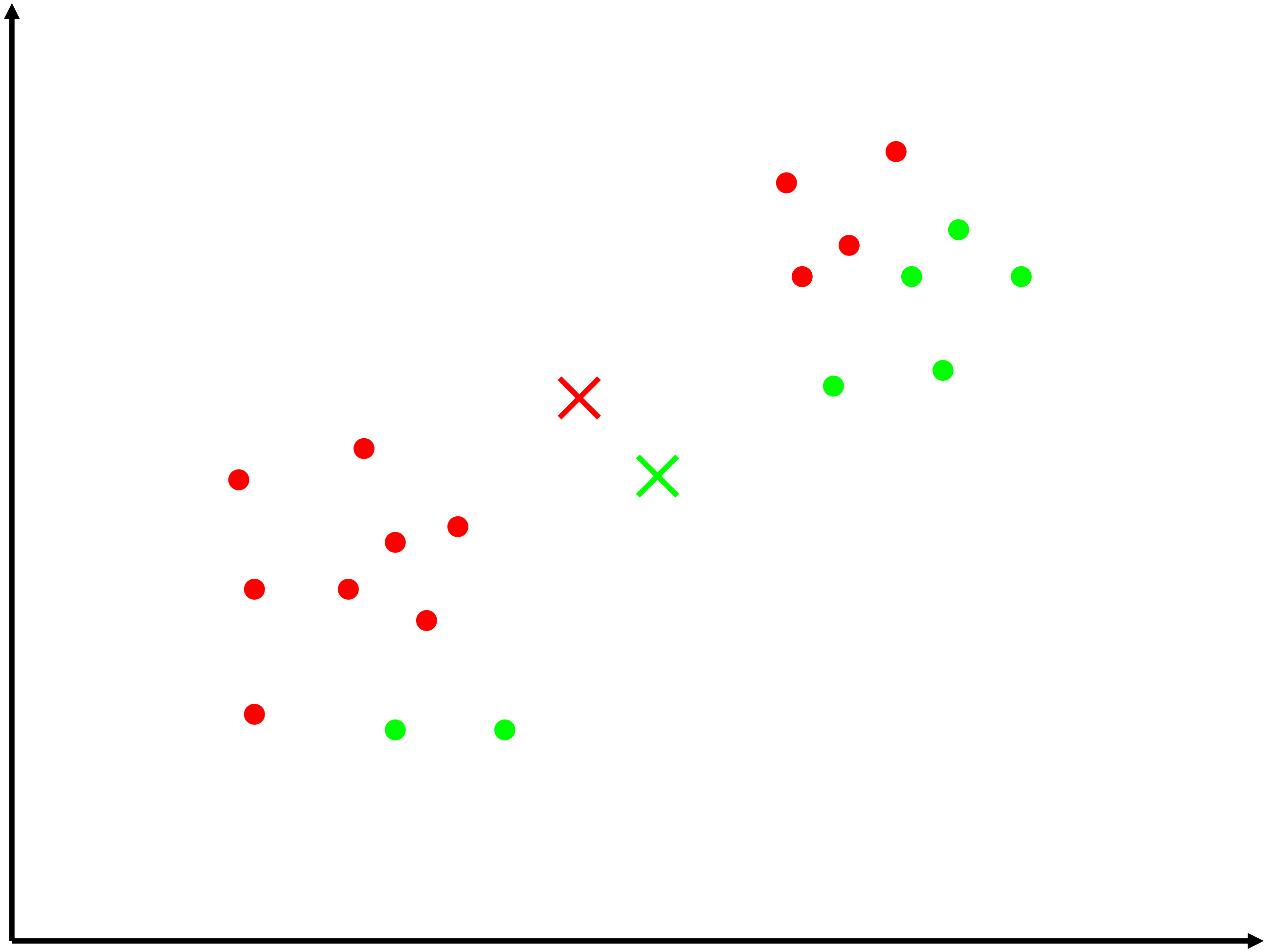
K-Means Clustering



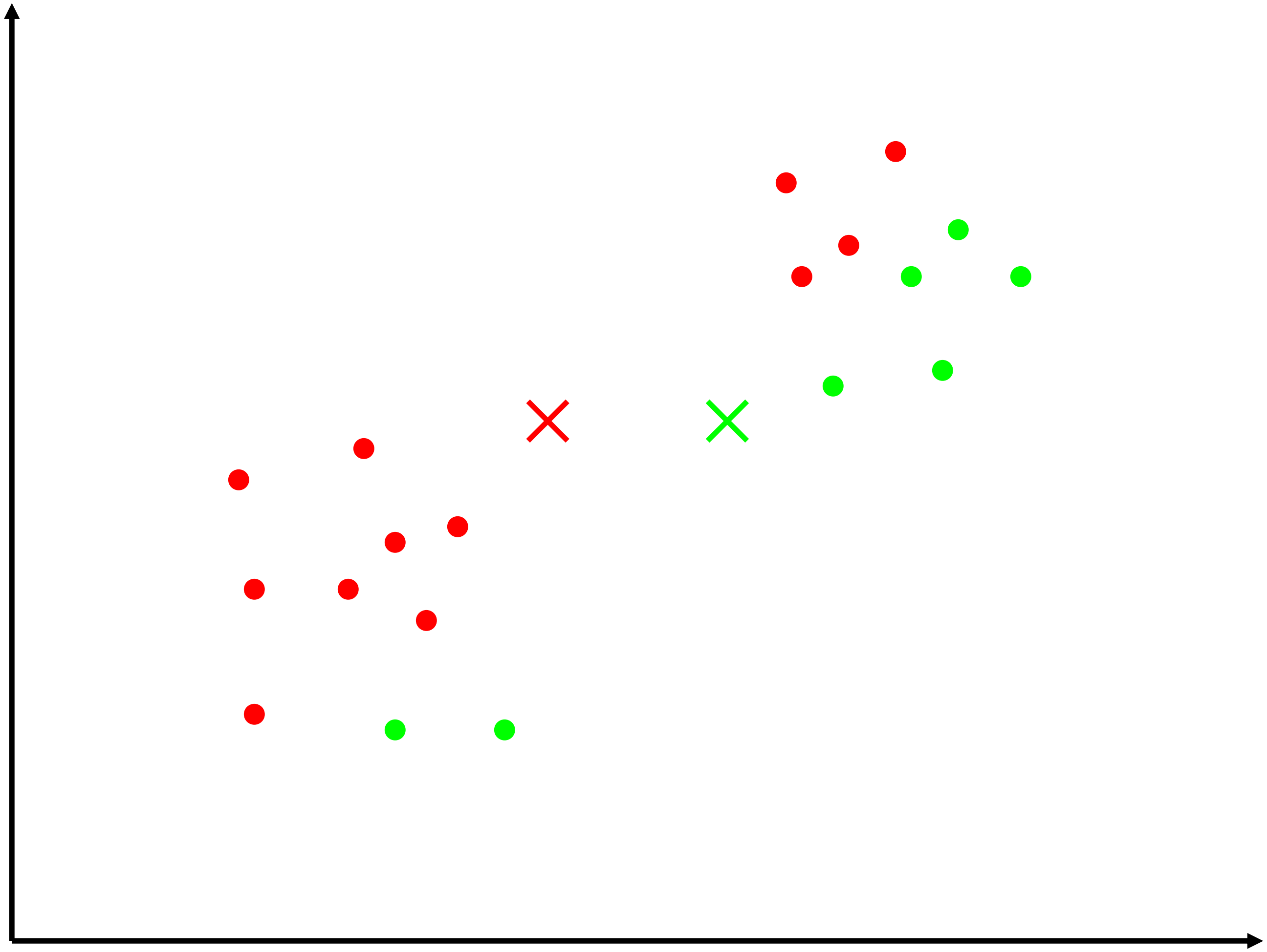
K-Means Clustering



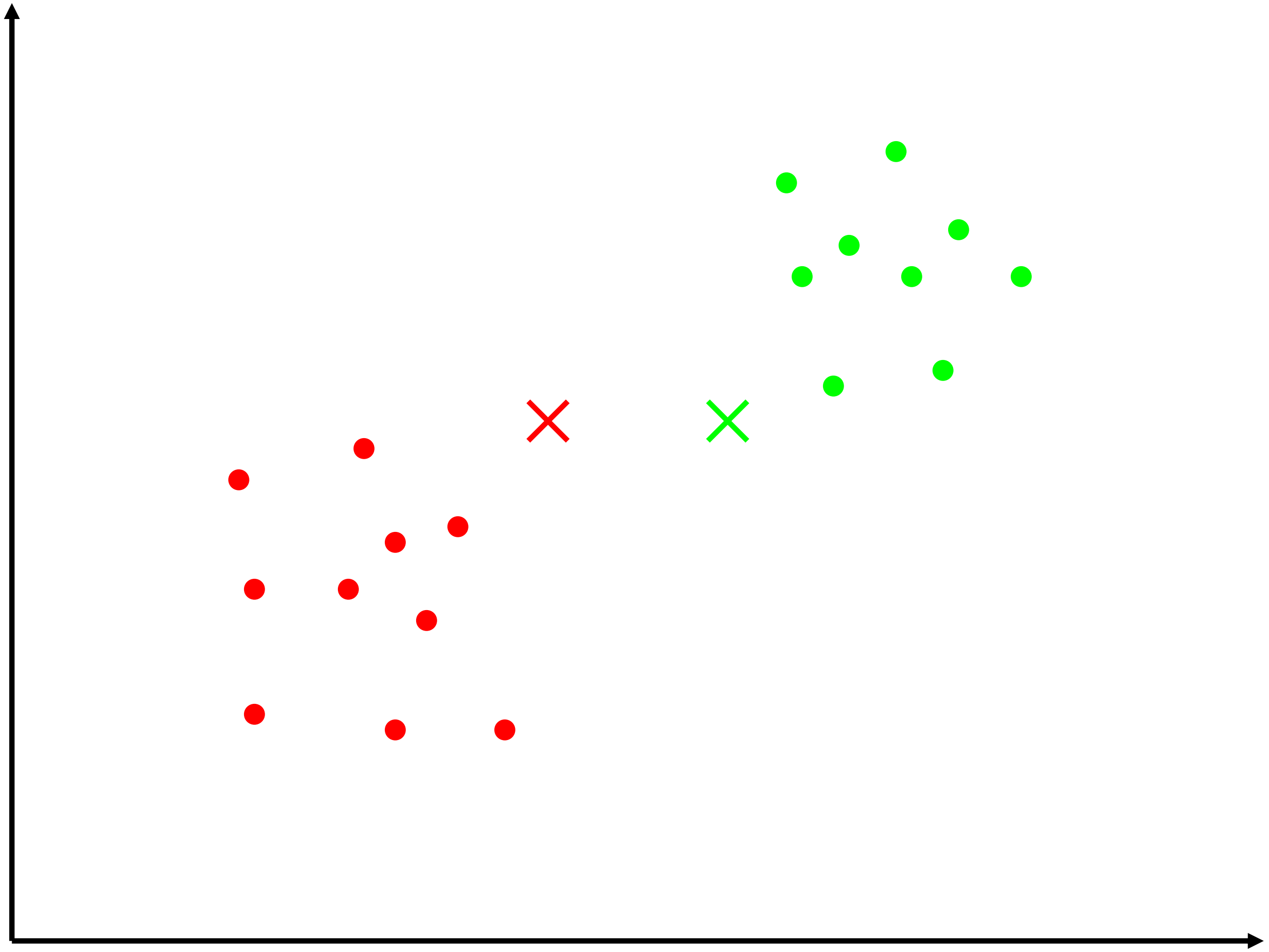
K-Means Clustering



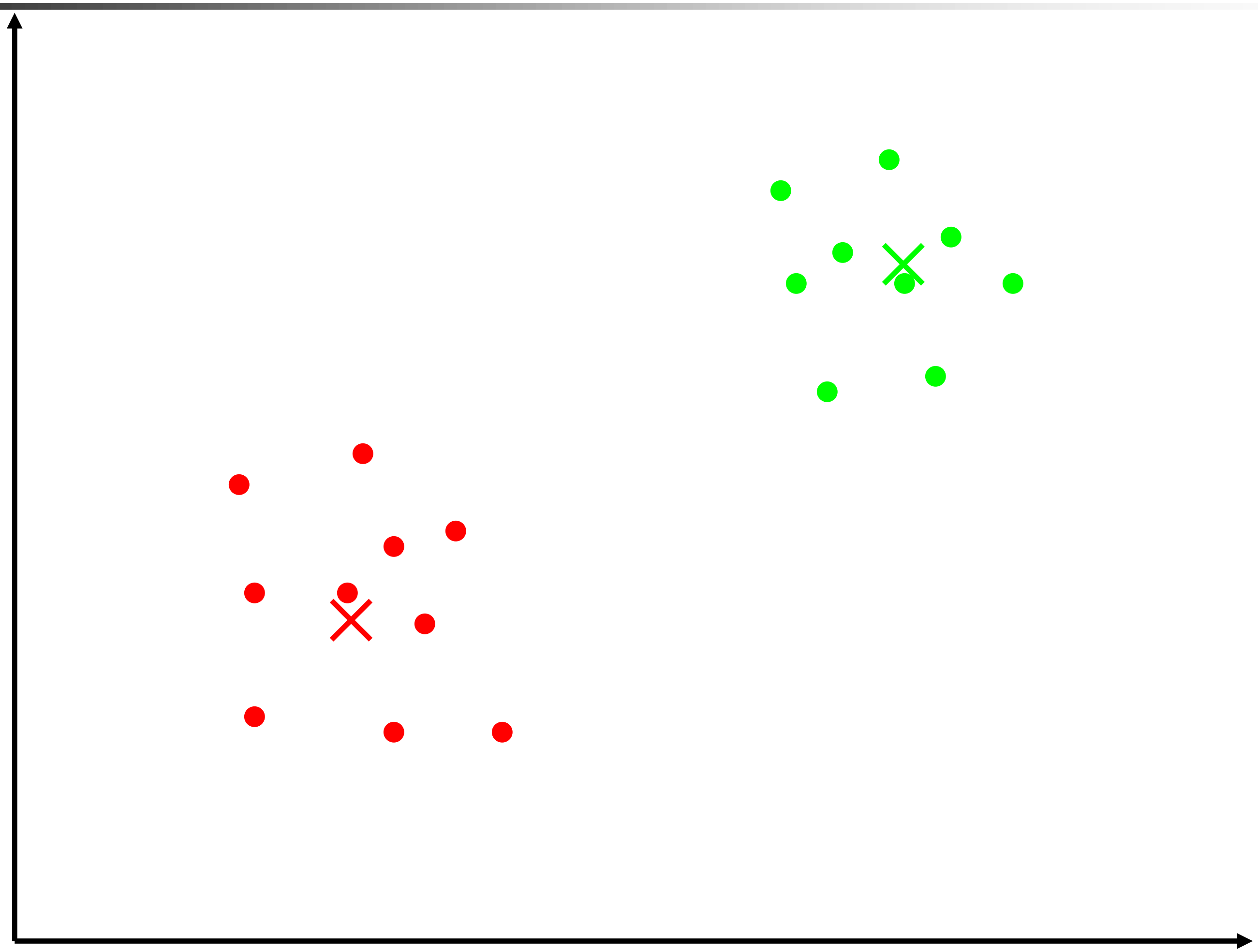
K-Means Clustering



K-Means Clustering

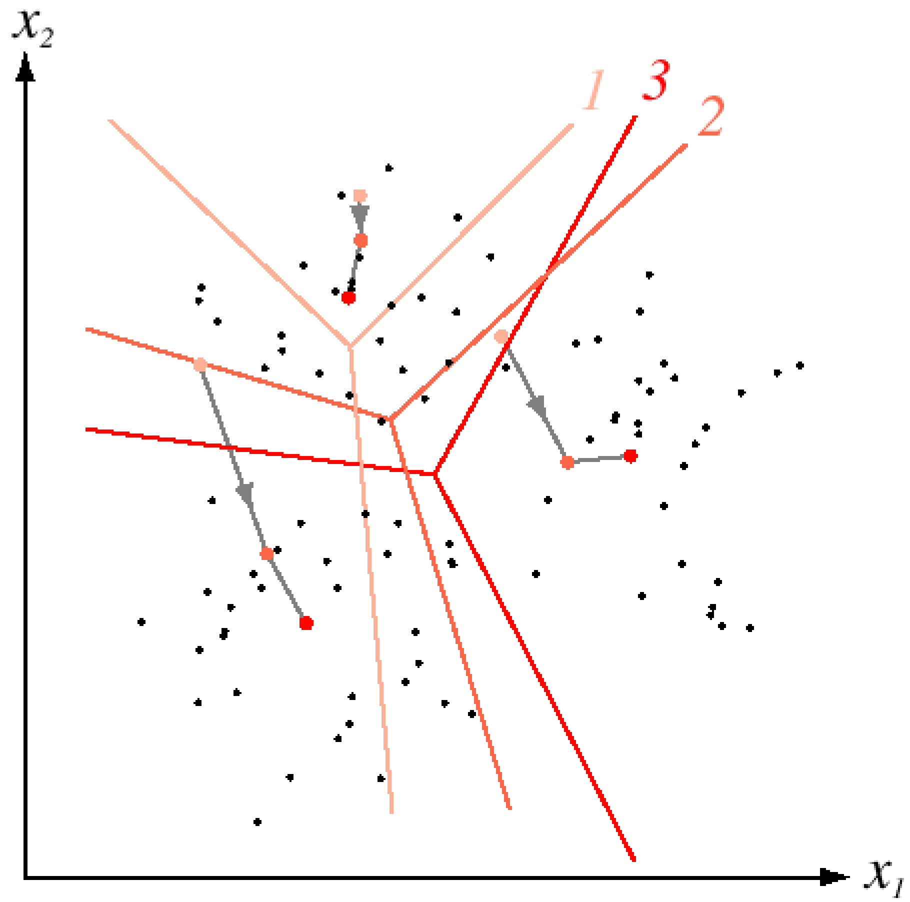


K-Means Clustering



K-Means Clustering

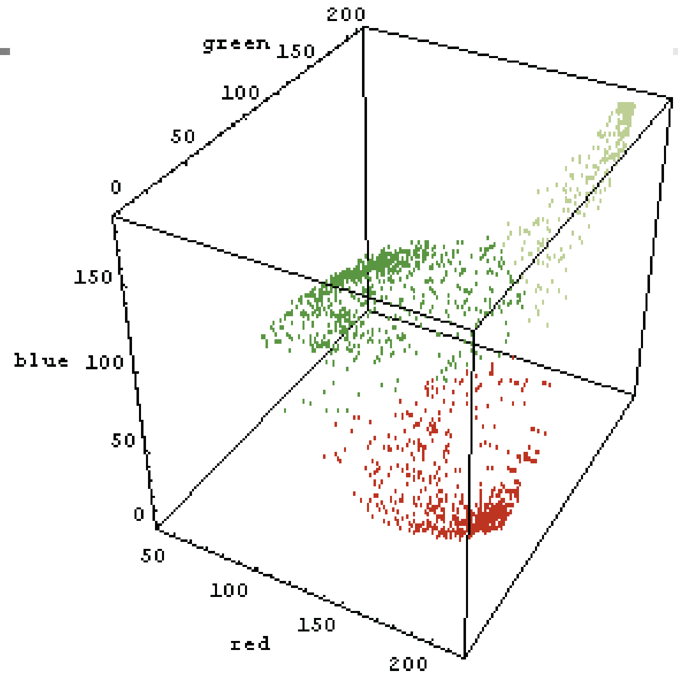
- **Example**



Duda et al.

K-Means Clustering

- **RGB vector**

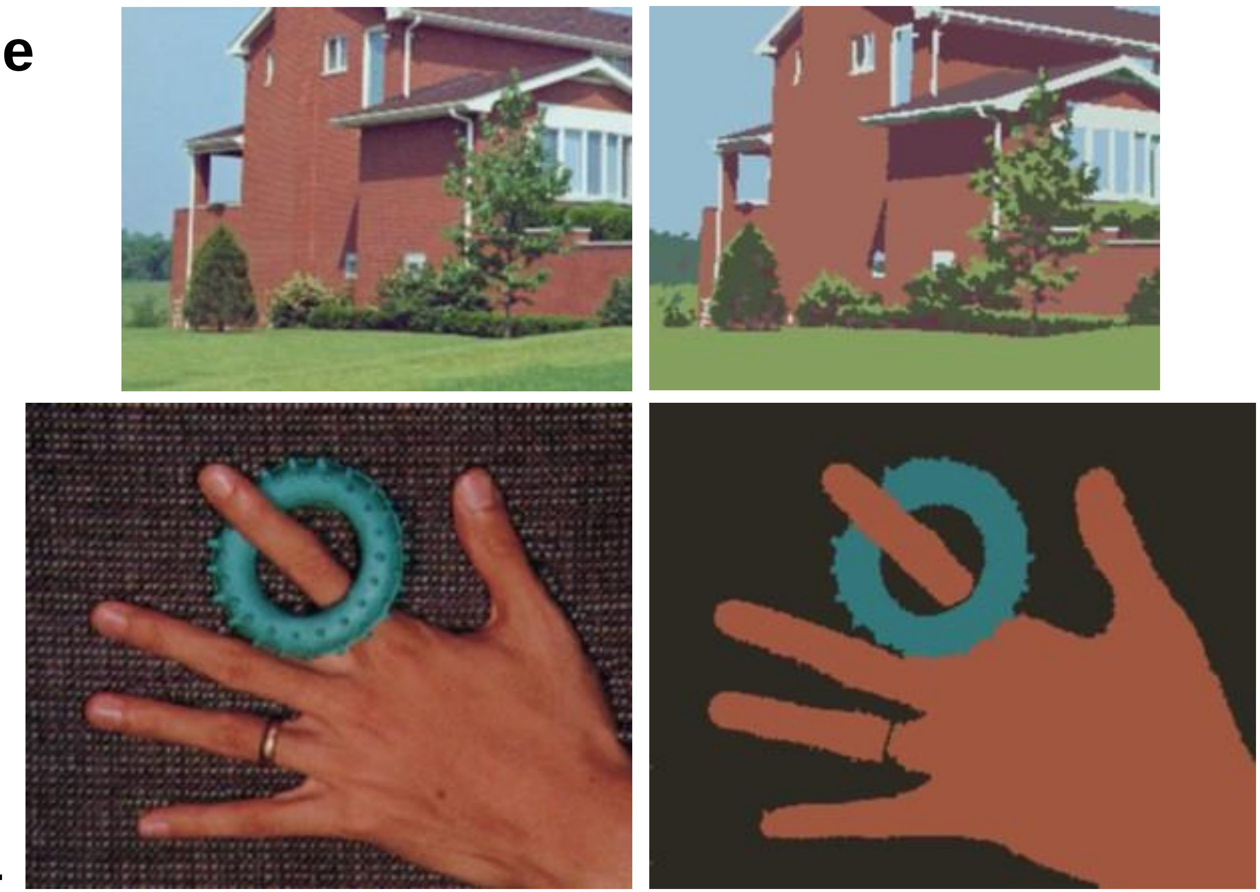


K-means clustering minimizes

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

Clustering

- **Example**



D. Comaniciu and P. Meer, *Robust Analysis of Feature Spaces: Color Image Segmentation*, 1997.

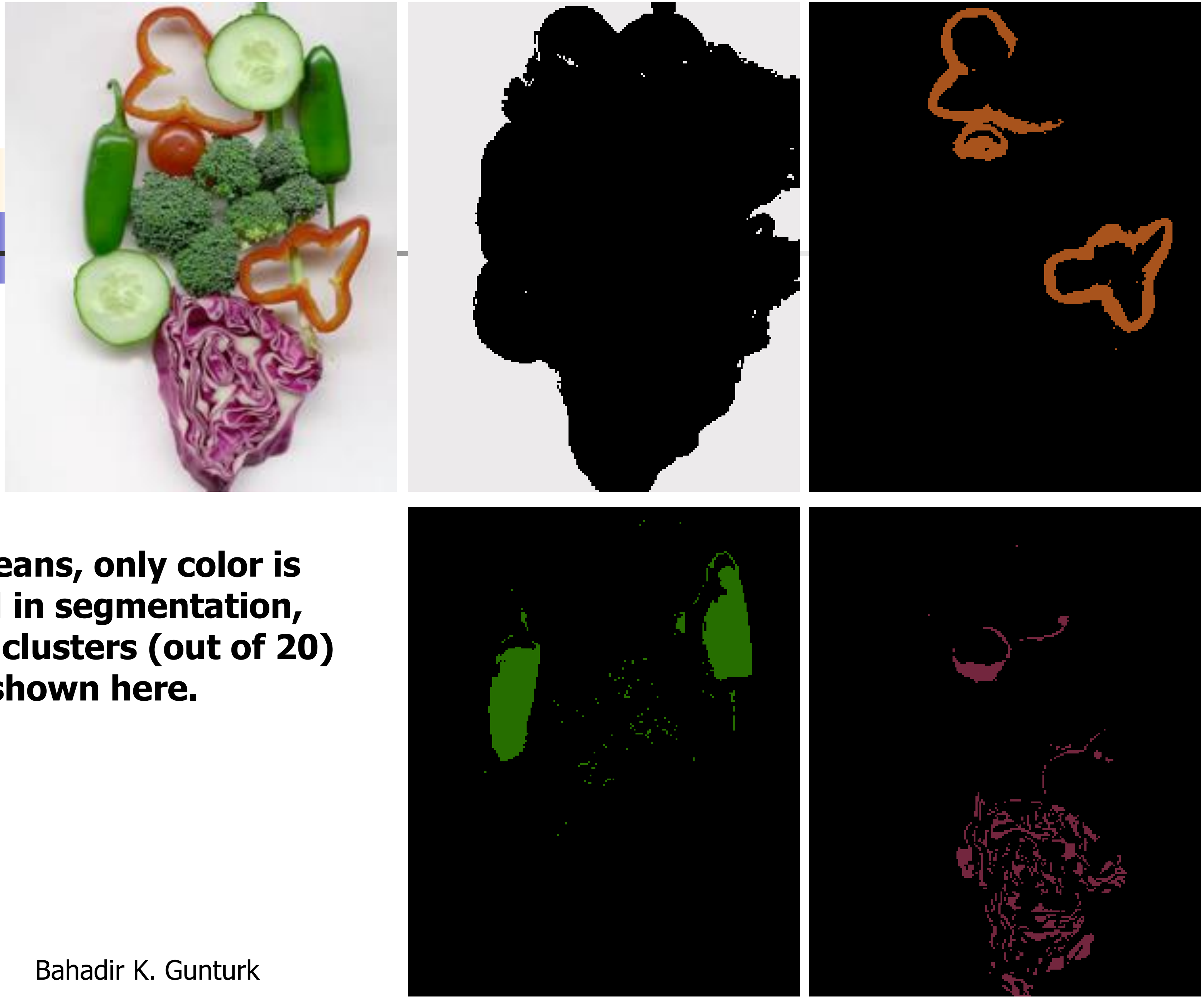
K-Means Clustering

- **Example**

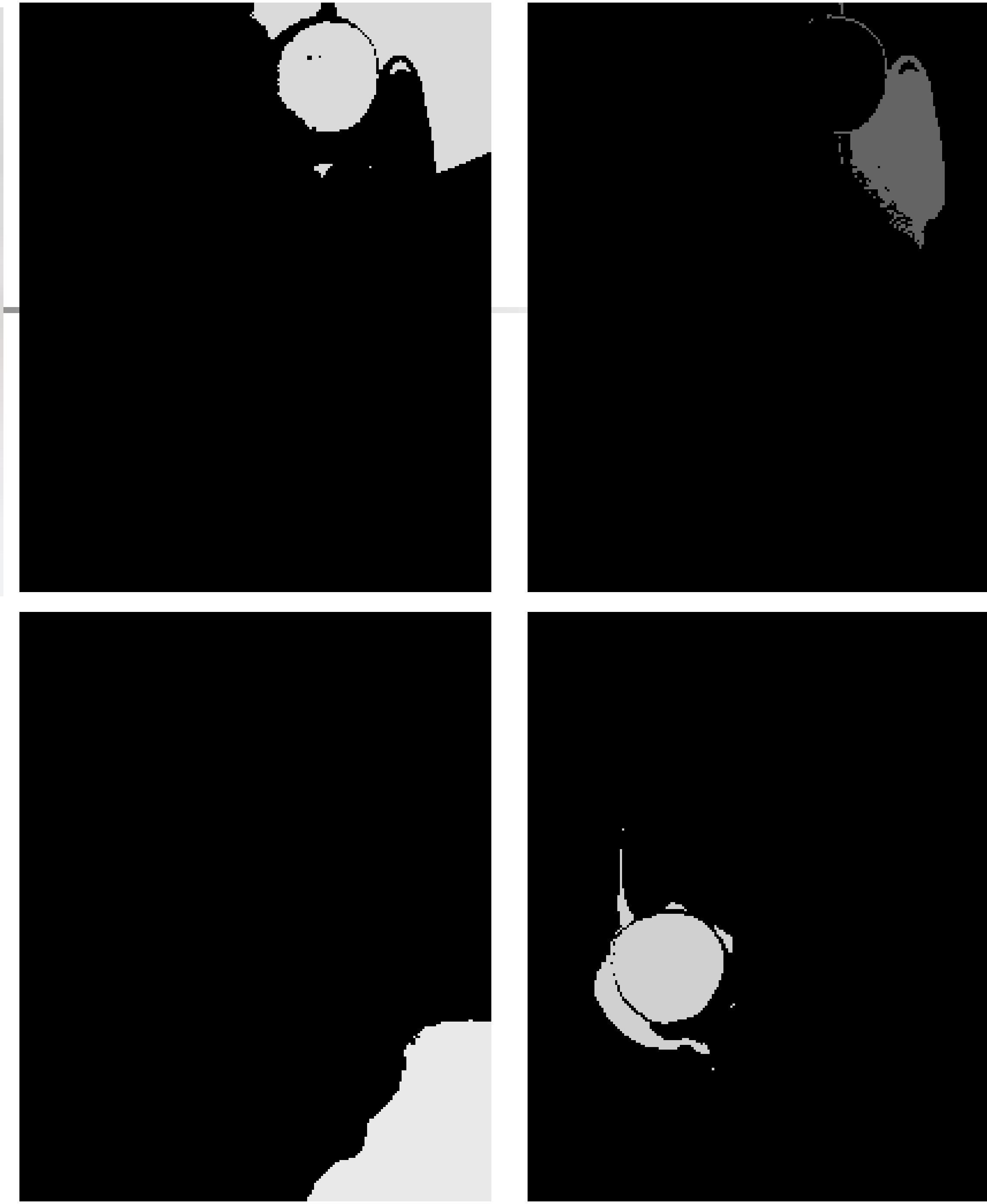
Original

K=5

K=11



K-means, only color is used in segmentation, four clusters (out of 20) are shown here.



K-means, color and position is used in segmentation, four clusters (out of 20) are shown here.

Each vector is (R,G,B,x,y).

K-Means Clustering: Axis Scaling

- **Features of different types may have different scales.**
  - **For example, pixel coordinates on a 100x100 image vs. RGB color values in the range [0,1].**
- **Problem: Features with larger scales dominate clustering.**
- **Solution: Scale the features.**

Spectral Clustering for Image Segmentation

Graph-based Image Segmentation

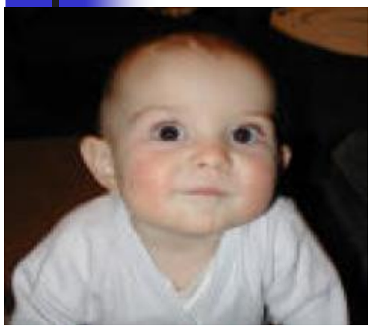
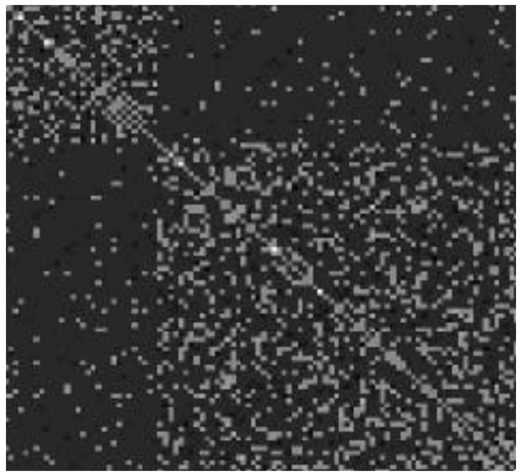


Image (I)



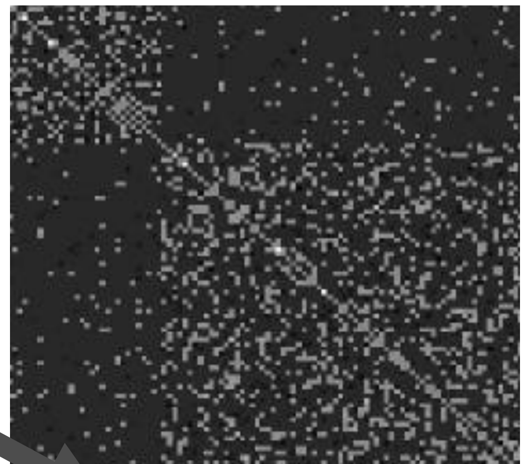
Intensity
Color
Edges
Texture

Graph Affinities
(W)

Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation

Image (I)



Graph Affinities (W)

Intensity
Color
Edges
Texture

$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation

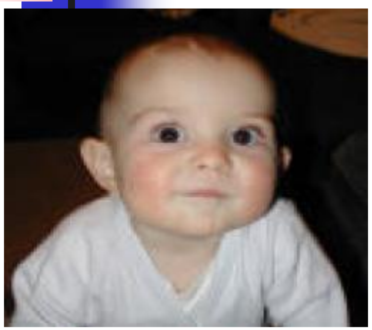
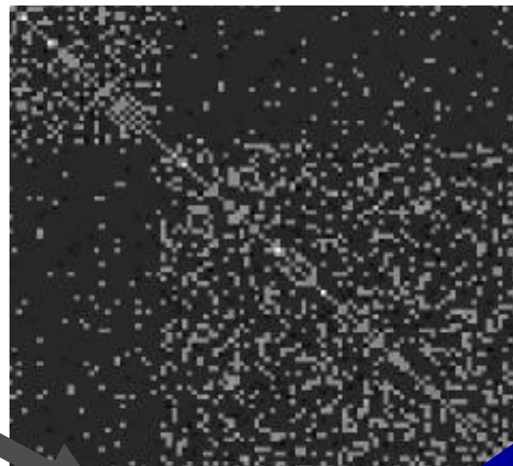


Image (I)



Graph Affinities (W)

Eigenvector X(W)

Intensity
Color
Edges
Texture

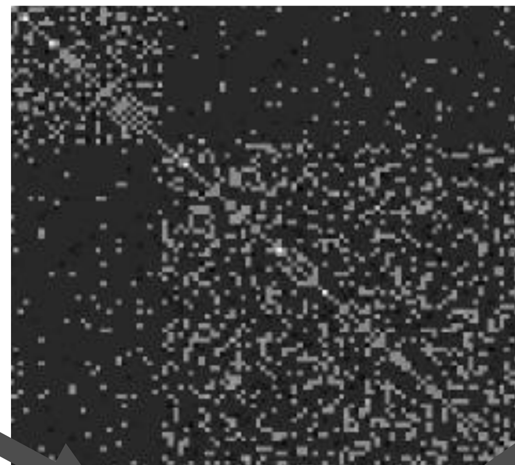
$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

$$(D - W)X = \lambda DX$$

$$X_A(i) = \begin{cases} 1 & if \quad i \in A \\ 0 & if \quad i \notin A \end{cases}$$

Graph-based Image Segmentation

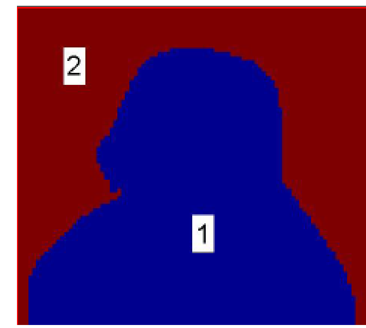
Image (I)



Graph Affinities
(W)

Intensity
Color
Edges
Texture

Eigenvector
X(W)



Discretization

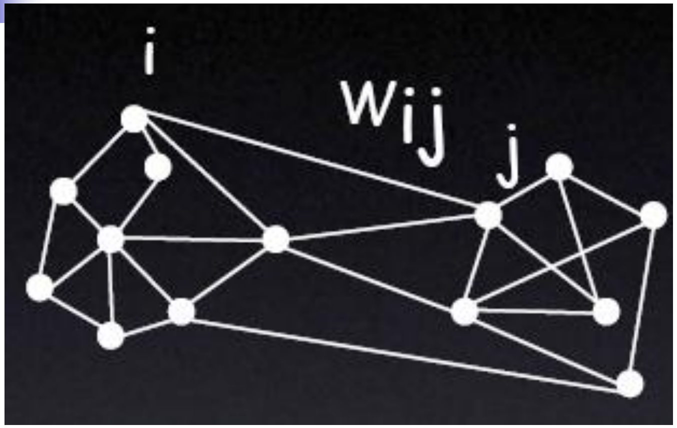
$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

$$(D - W)X = \lambda DX$$

$$X_A(i) = \begin{cases} 1 & if \quad i \in A \\ 0 & if \quad i \notin A \end{cases}$$

Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation



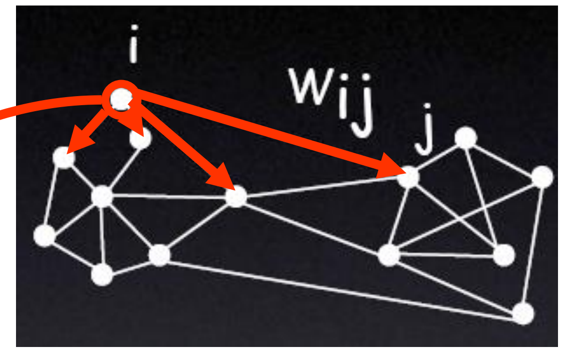
G = {V,E}

V: graph nodes
E: edges connection nodes → **Pixels**
Pixel similarity

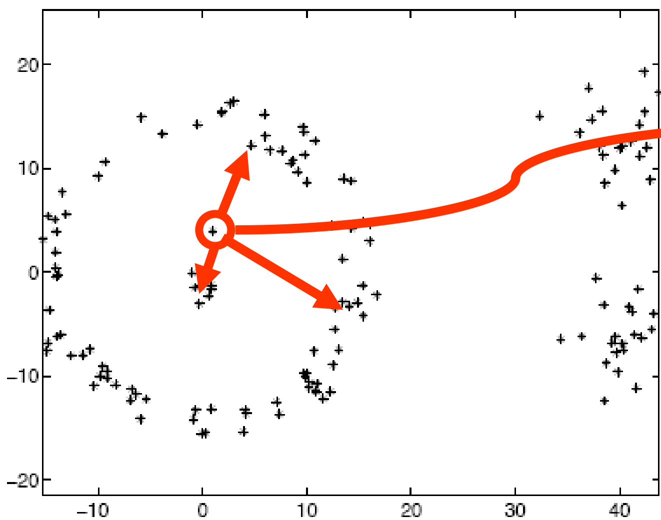
Slides from Jianbo Shi

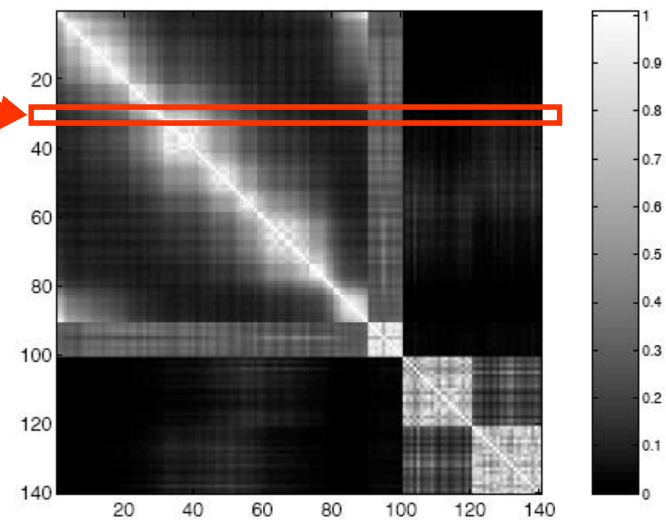
Graph terminology

- Similarity matrix: $W = \left[w_{i,j} \right]$



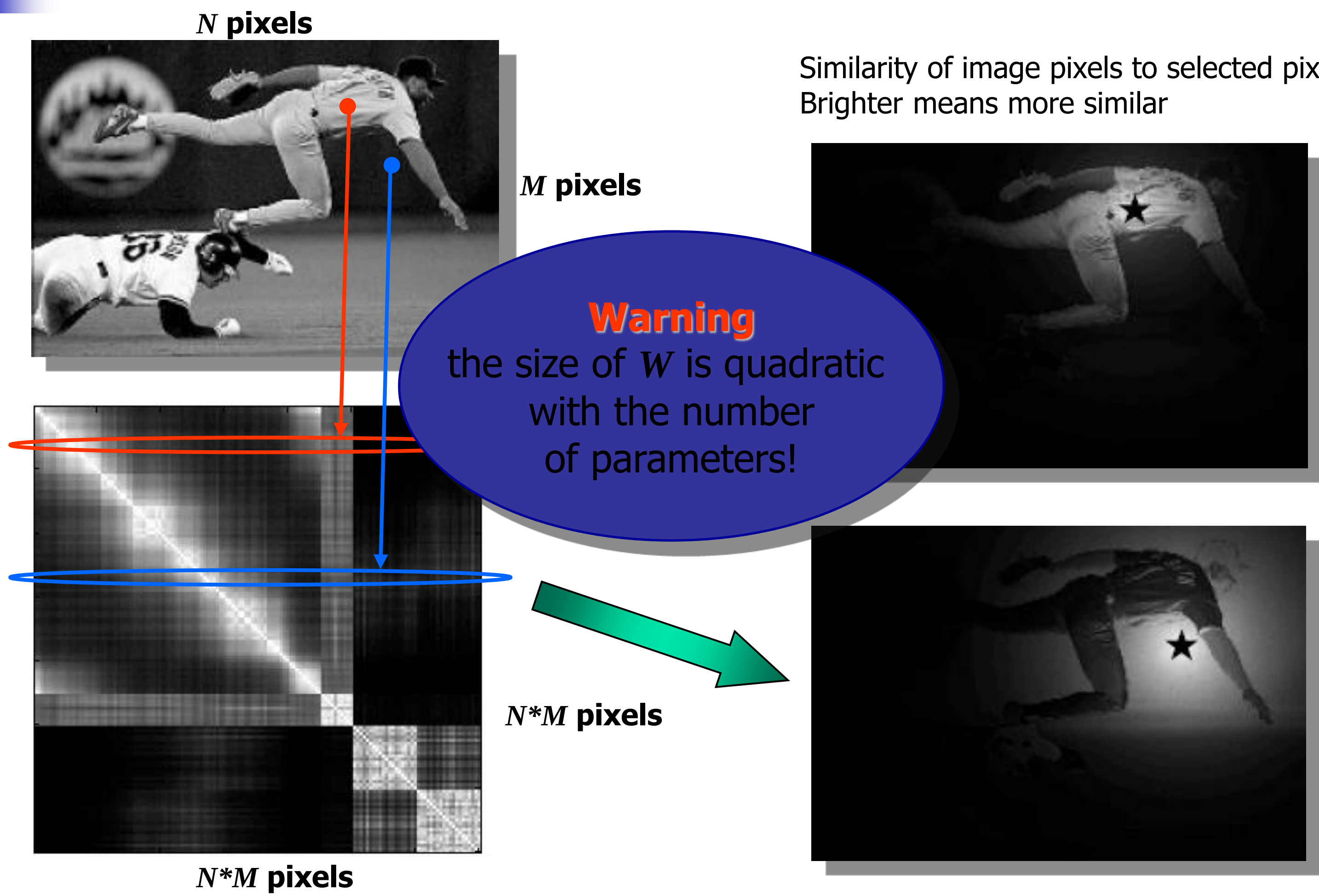
$$w_{i,j} = e^{\frac{-\left\| X_{(i)} - X_{(j)} \right\|_2^2}{\sigma_X^2}}$$





Slides from Jianbo Shi

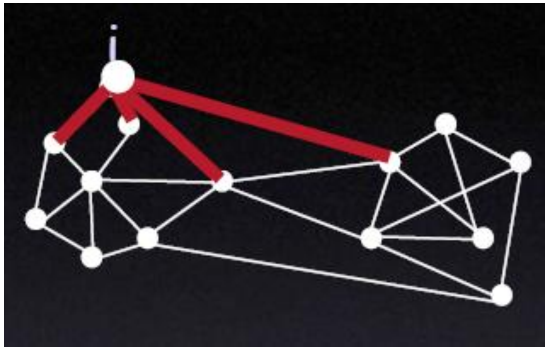
Affinity matrix

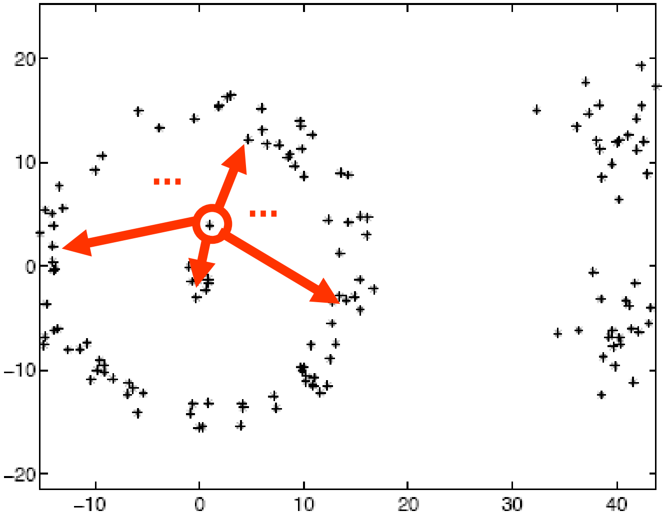


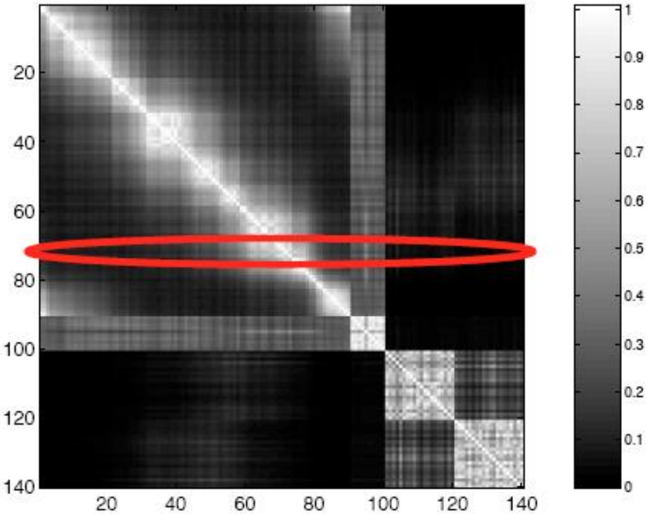
Graph terminology

- Degree of node:

$$d_i = \sum_j w_{i,j}$$





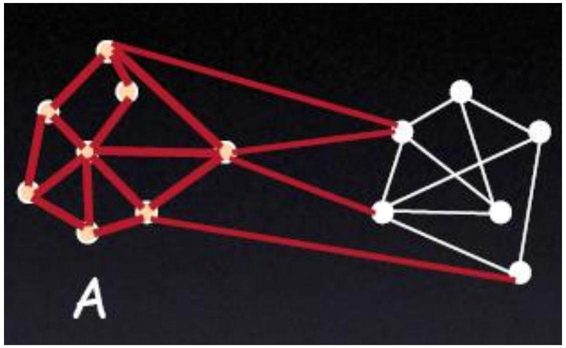


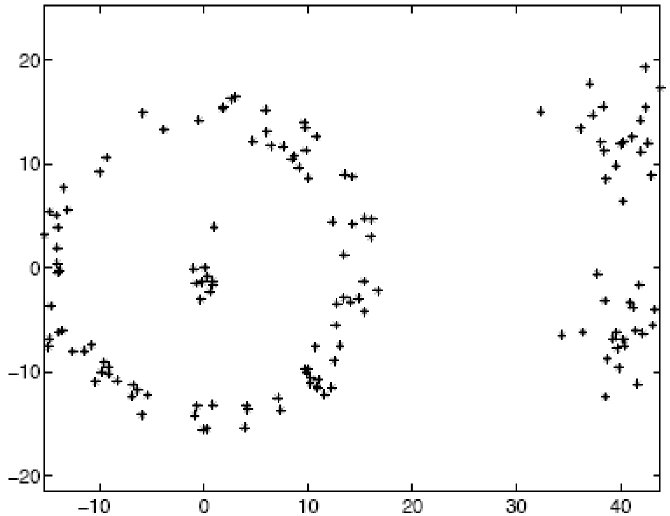
Slides from Jianbo Shi

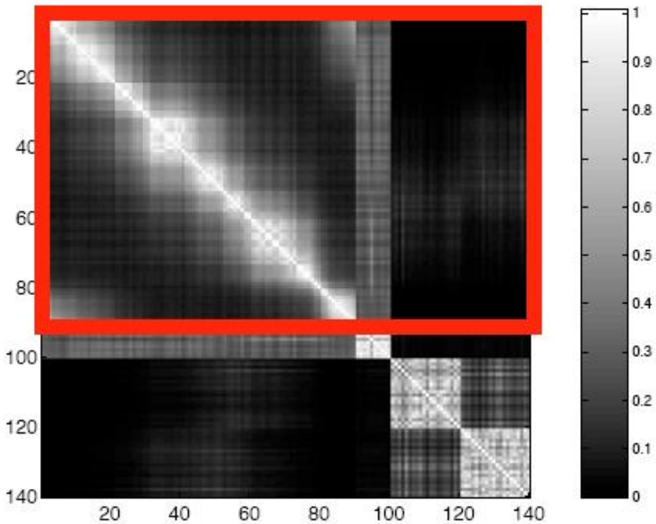
Graph terminology

- Volume of set:

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





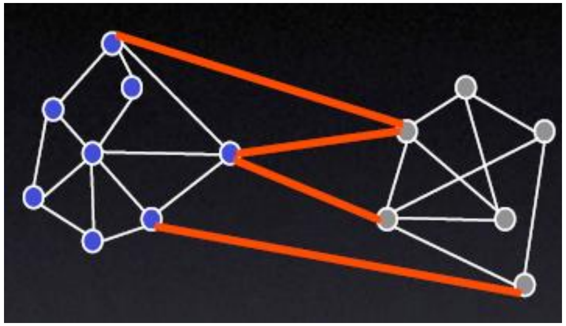


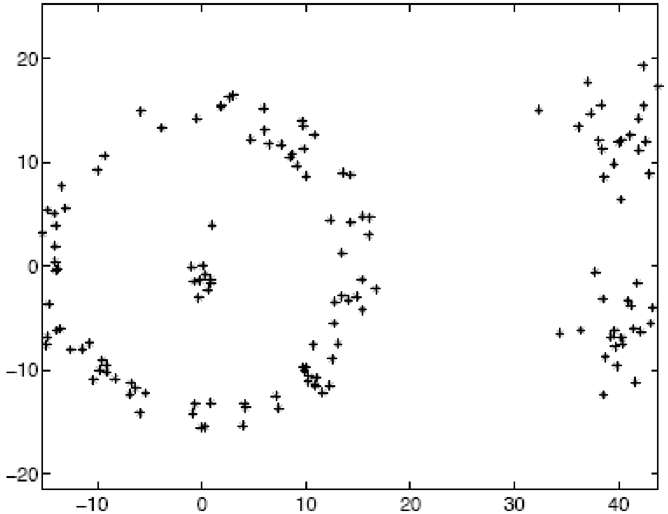
Slides from Jianbo Shi

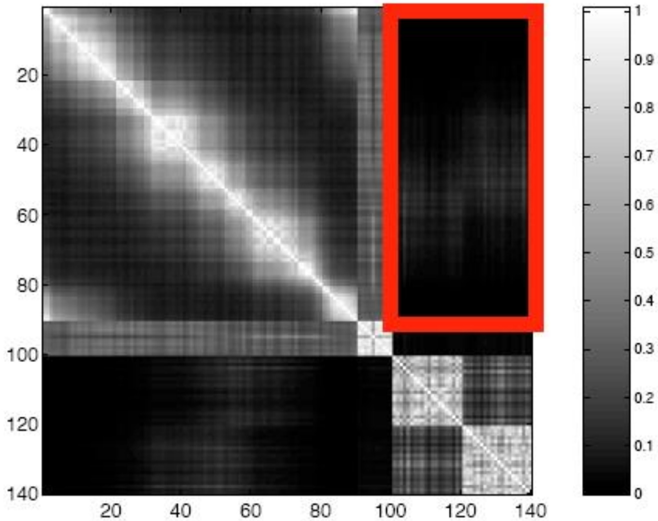
Graph terminology

- Cuts in a graph:

$$cut(A,\bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{i,j}$$







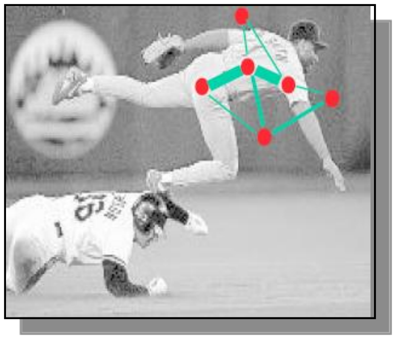
Slides from Jianbo Shi

Representation

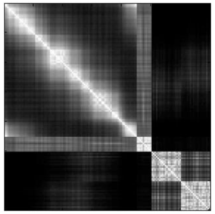
Partition matrix X:

$$X = [X_1, \ldots, X_K]$$

$$X = \begin{array}{c} \textbf{segments} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \textbf{pixels}$$



Pair-wise similarity matrix W: $W(i, j) = aff(i, j)$



Degree matrix D: $D(i,i) = \sum_j w_{i,j}$

Laplacian matrix L: $L = D - W$

Pixel similarity functions

Intensity

$$W(i,j) = e^{\frac{-\left\| I_{(i)} - I_{(j)} \right\|_2^2}{\sigma_I^2}}$$

Distance

$$W(i,j) = e^{\frac{-\left\| X_{(i)} - X_{(j)} \right\|_2^2}{\sigma_X^2}}$$

Texture

$$W(i,j) = e^{\frac{-\left\| c_{(i)} - c_{(j)} \right\|_2^2}{\sigma_c^2}}$$

Pixel similarity functions

Intensity

$$\frac{-\left\|I_{(i)}-I_{(j)}\right\|_2^2}{\sigma_I^2}$$

here c(x) is a vector of filter outputs.
A natural thing to do is to square the outputs
of a range of different filters
at different scales and orientations,
smooth the result, and rack
these into a vector.

$$e \quad \frac{\ldots_{(i)}-X_{(j)}\big\|_2^2}{\sigma_X^2}$$

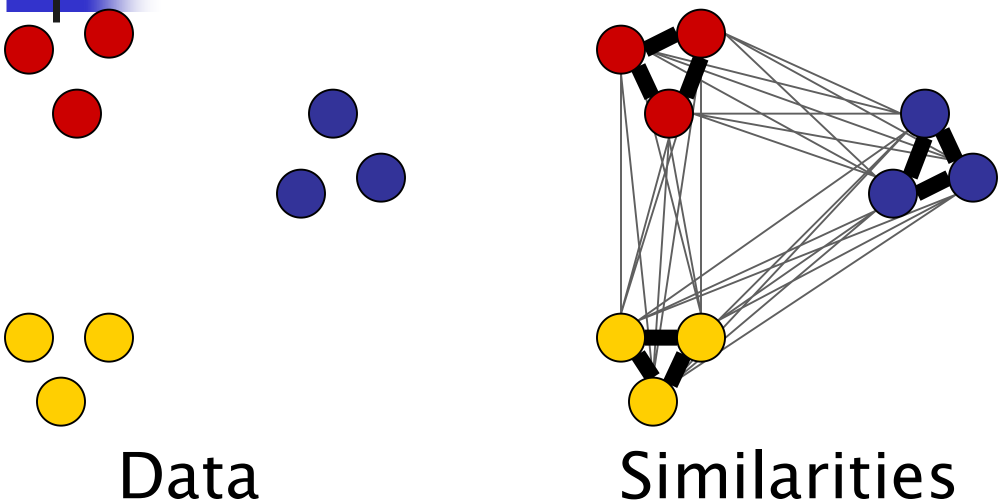
Texture

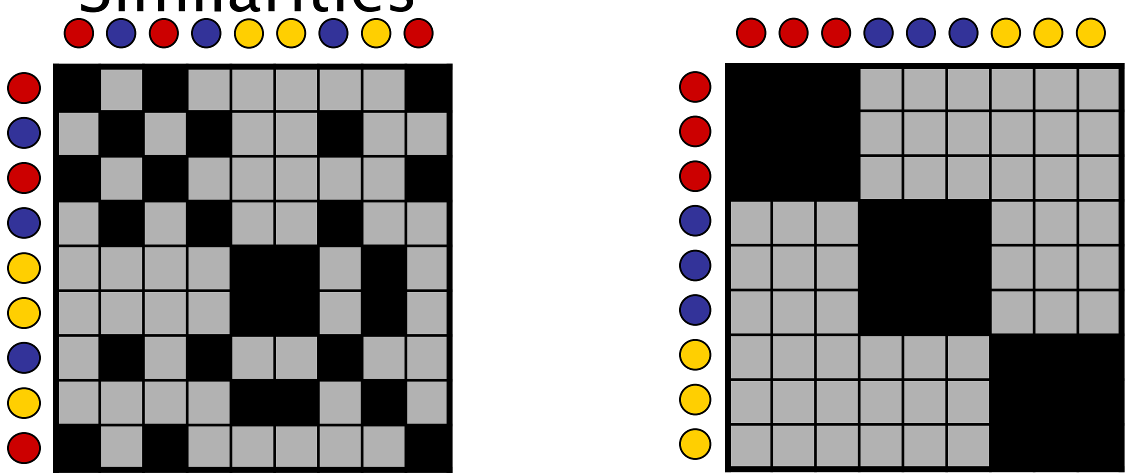
$$W(i,j)=e^{\frac{-\left\|c_{(i)}-c_{(j)}\right\|_2^2}{\sigma_c^2}}$$

Definitions

- Methods that use the spectrum of the affinity matrix to cluster are known as ***spectral clustering***.
- Normalized cuts, Average cuts, Average association make use of the eigenvectors of the affinity matrix.
- Why these methods work?

Spectral Clustering





* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Eigenvectors and blocks

- Block matrices have block eigenvectors:

$\lambda_1 = 2$ $\lambda_2 = 2$ $\lambda_3 = 0$ $\lambda_4 = 0$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \begin{bmatrix} .71 \\ .71 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ .71 \\ .71 \end{bmatrix}$$

- Near-block matrices have near-block eigenvectors:

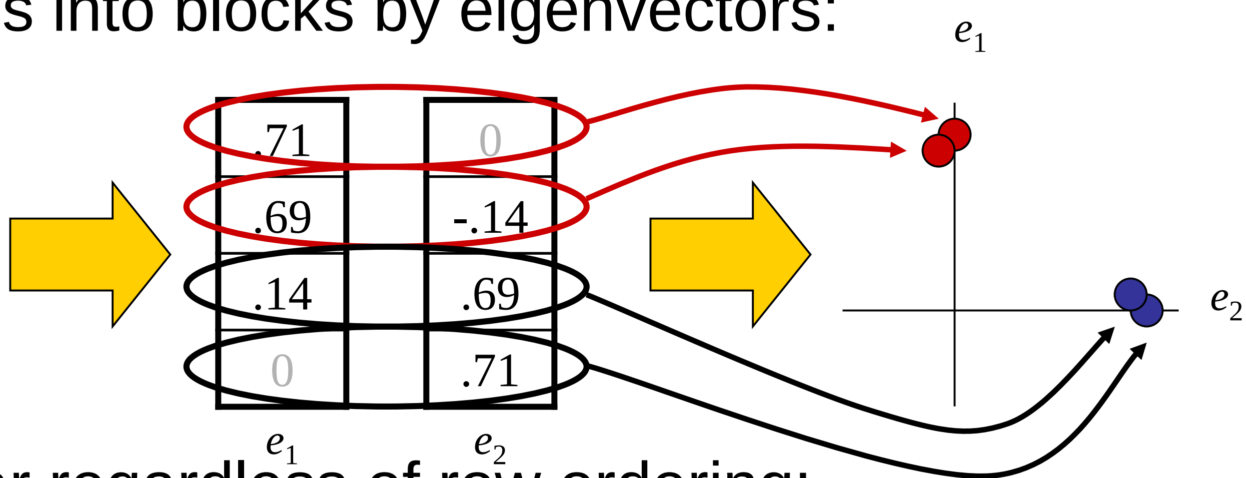
$\lambda_1 = 2.02$ $\lambda_2 = 2.02$ $\lambda_3 = -0.02$ $\lambda_4 = -0.02$

$$\begin{bmatrix} 1 & 1 & .2 & 0 \\ 1 & 1 & 0 & -.2 \\ .2 & 0 & 1 & 1 \\ 0 & -.2 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \begin{bmatrix} .71 \\ .69 \\ .14 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ -.14 \\ .69 \\ .71 \end{bmatrix}$$

Spectral Space

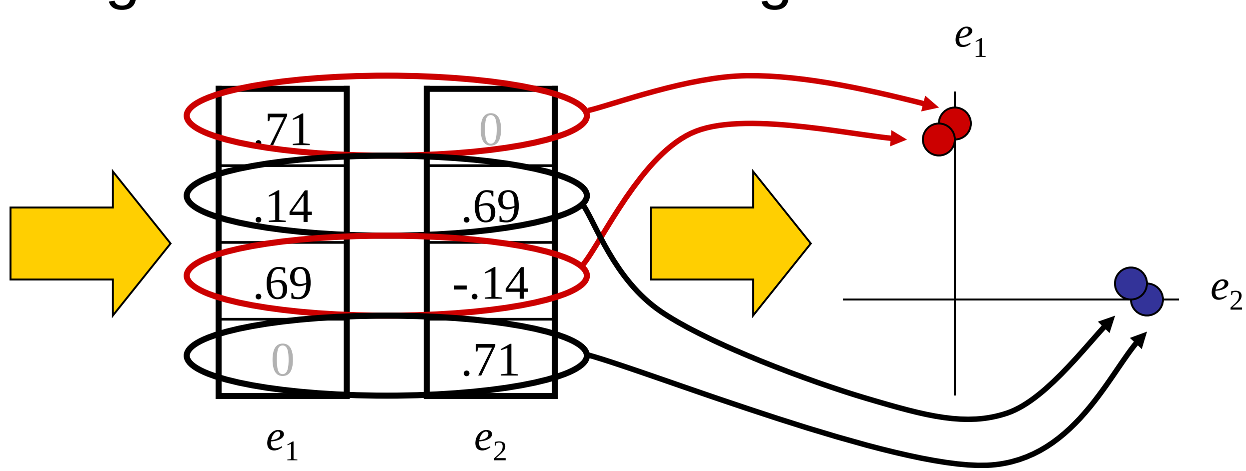
- Can put items into blocks by eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



- Clusters clear regardless of row ordering:

1	.2	1	0
.2	1	0	1
1	0	1	-.2
0	1	-.2	1



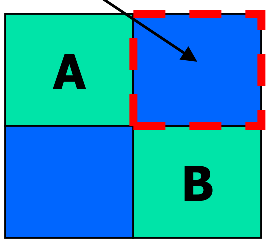
How do we extract a good cluster?

- Simplest idea: we want a vector $\boldsymbol{x}$ 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 $x^T W x$
- But need the constraint $x^T x = 1$
- This is an eigenvalue problem - choose the eigenvector of W with largest eigenvalue.

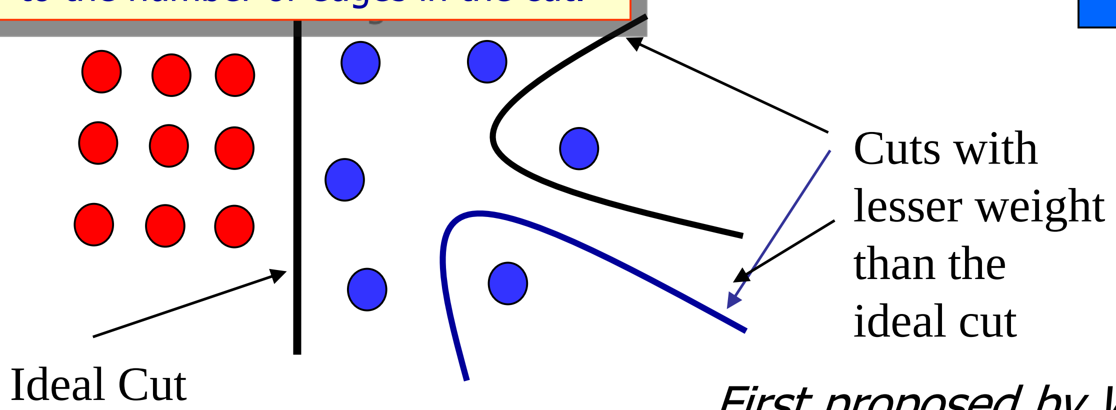
Minimum cut

- Criterion for partition:

$$\min cut(A,B) = \min_{A,B} \sum_{u \in A, v \in B} w(u,v)$$



Problem!
Weight of cut is directly proportional to the number of edges in the cut.



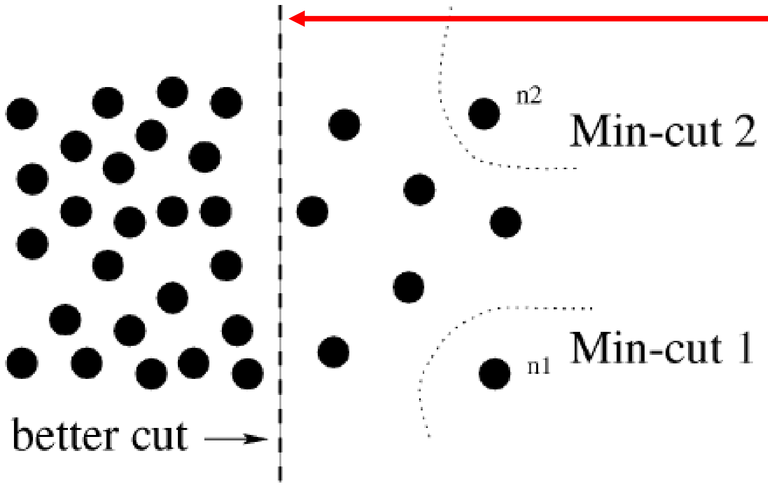
First proposed by Wu and Leahy

Normalized Cut

Normalized cut or balanced cut:

$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

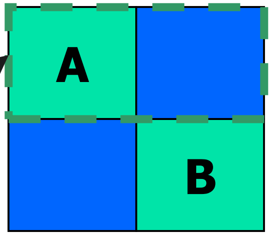
Finds better cut



Normalized Cut

- Volume of set (or association):

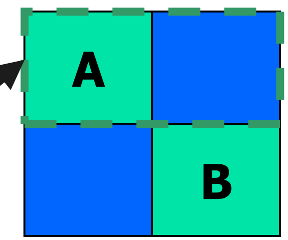
$$vol(A) = assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$



Normalized Cut

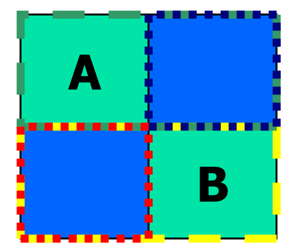
- Volume of set (or association):

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



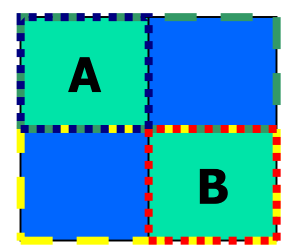
- Define normalized cut: "a fraction of the total edge connections to all the nodes in the graph":

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



- Define normalized association: "how tightly on average nodes within the cluster are connected to each other"

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$



Observations(I)

- Maximizing $Nassoc$ is the same as minimizing $Ncut$, since they are related:

$$Ncut(A,B) = 2 - Nassoc(A,B)$$

- How to minimize $Ncut$?
  - Transform $Ncut$ equation to a matricial form.
  - After simplifying:

$$D(i,i) = \sum_j W(i,j)$$

$$\min_x Ncut(x) = \min_y \frac{y^T (D - W) y}{y^T D y}$$

Subject to: $y^T D1 = 0$

Rayleigh quotient

NP-Hard!

y's values are quantized

Observations(II)

- Instead, relax into the continuous domain by solving generalized eigenvalue system:

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

- Which gives: $(D-W)y = \lambda D y$
- Note that $(D-W)1 = 0$ so, the first eigenvector is $y_0 = 1$ with eigenvalue 0.
- The second smallest eigenvector is the real valued solution to this problem!!

Algorithm

1. Define a similarity function between 2 nodes. i.e.:

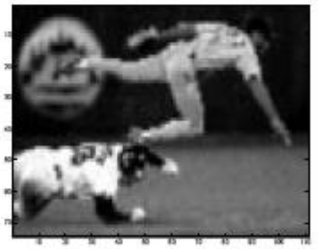
$$w_{i,j} = e^{\frac{-\left\|F_{(i)}-F_{(j)}\right\|_2^2}{\sigma_I^2} + \frac{-\left\|X_{(i)}-X_{(j)}\right\|_2^2}{\sigma_X^2}}$$

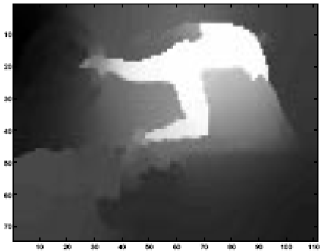
2. Compute affinity matrix (W) and degree matrix (D).
3. Solve $(D-W)y = \lambda D y$
4. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
5. Decide if re-partition current partitions.

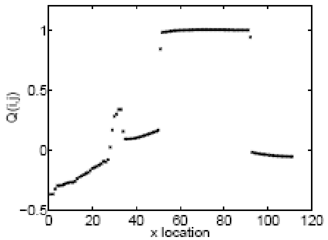
Note: since precision requirements are low, W is very sparse and only few eigenvectors are required, the eigenvectors can be extracted very fast using Lanczos algorithm.

Discretization

- Sometimes there is not a clear threshold to binarize since eigenvectors take on continuous values.





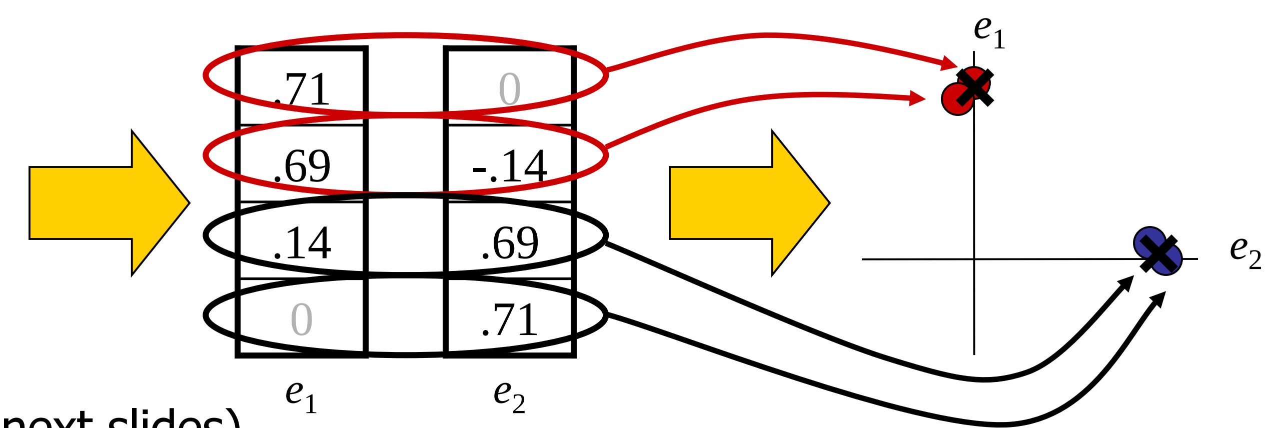


- How to choose the splitting point?
  a) Pick a constant value (0, or 0.5).
  b) Pick the median value as splitting point.
  c) Look for the splitting point that has the minimum *Ncut* value:
     1. Choose *n* possible splitting points.
     2. Compute *Ncut* value.
     3. Pick minimum.

Use *k*-eigenvectors

- Recursive 2-way *Ncut* is slow.
- We can use more eigenvectors to re-partition the graph, however:
  - Not all eigenvectors are useful for partition (**degree of smoothness**).
- Procedure: compute *k-means* with a high *k*. Then follow one of these procedures:
  a) Merge segments that minimize *k*-way *Ncut* criterion.
  b) Use the k segments and find the partitions there using exhaustive search.

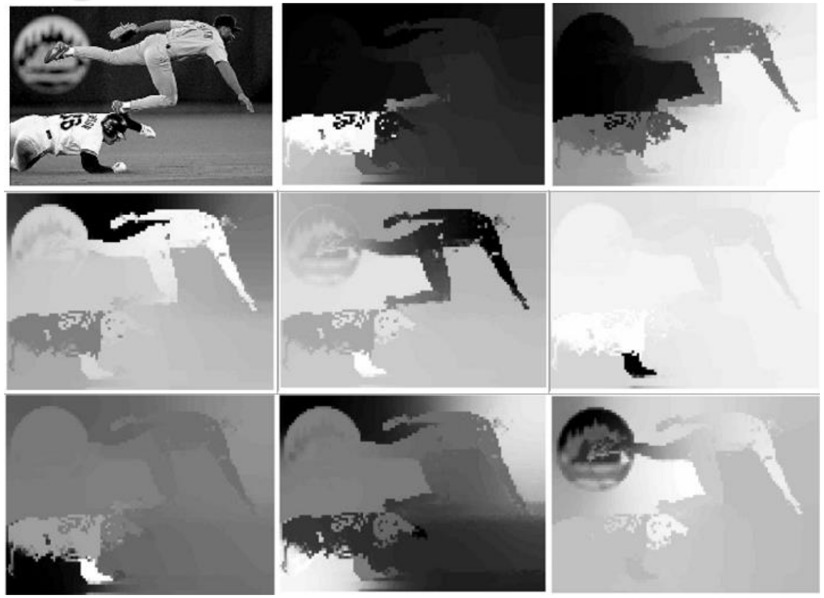
1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



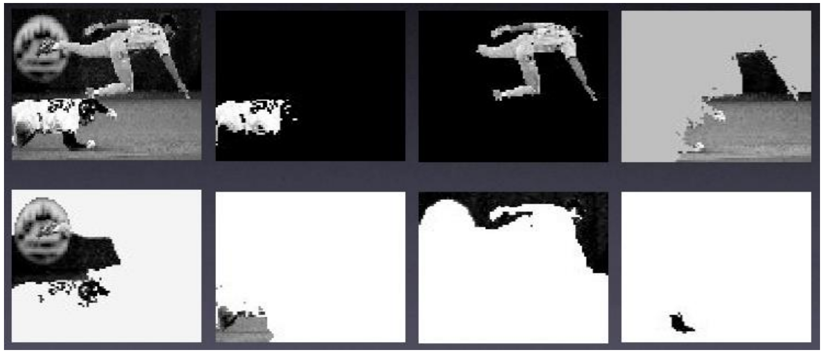
- Compute Q (next slides).

Example

Eigenvectors



Segments

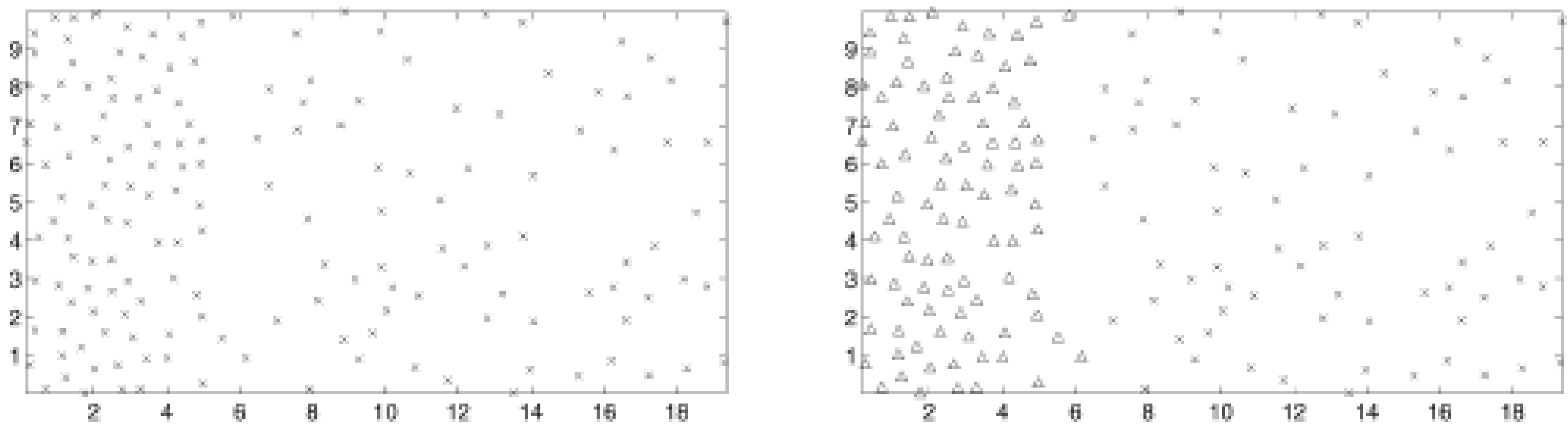


Experiments

- Define similarity: $w_{i,j} = e^{\frac{-\left\|F_{(i)}-F_{(j)}\right\|_2^2}{\sigma_I^2}+\frac{-\left\|X_{(i)}-X_{(j)}\right\|_2^2}{\sigma_X^2}}$
  - $F(i)=1$ for point sets.
  - $F(i)=I(i)$ for brightness images.
  - $F(i)=[v, v.s.\sin(h), v.s.\cos(h)]$ for HSV images.
  - $F(i)=[|I*f_1|,\cdots,|I*f_n|]$ in case of texture.

Experiments (I)

- Point set segmentation:

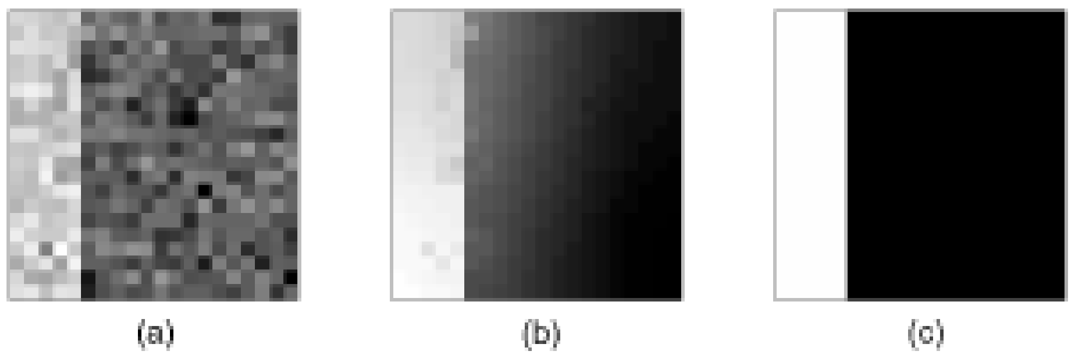


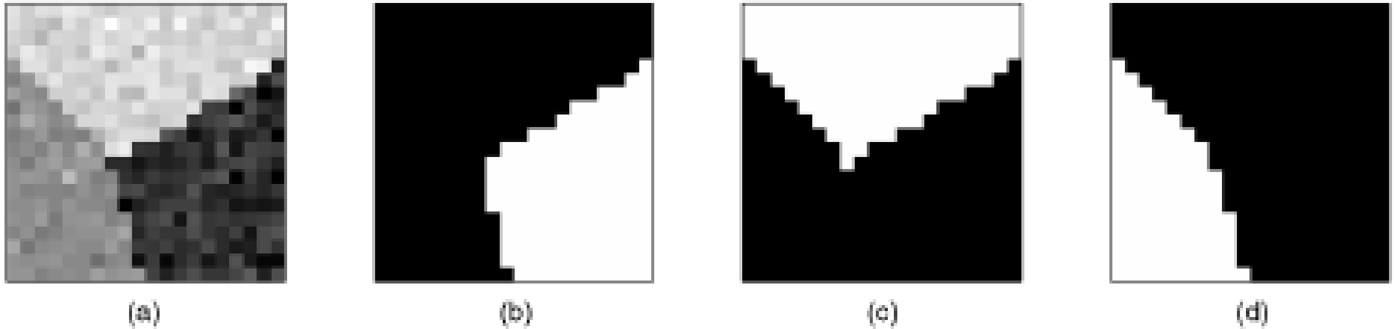
(a) (b)

(a) Pointset generated by Poisson process. (b) Segmentation results.

Experiments (II)

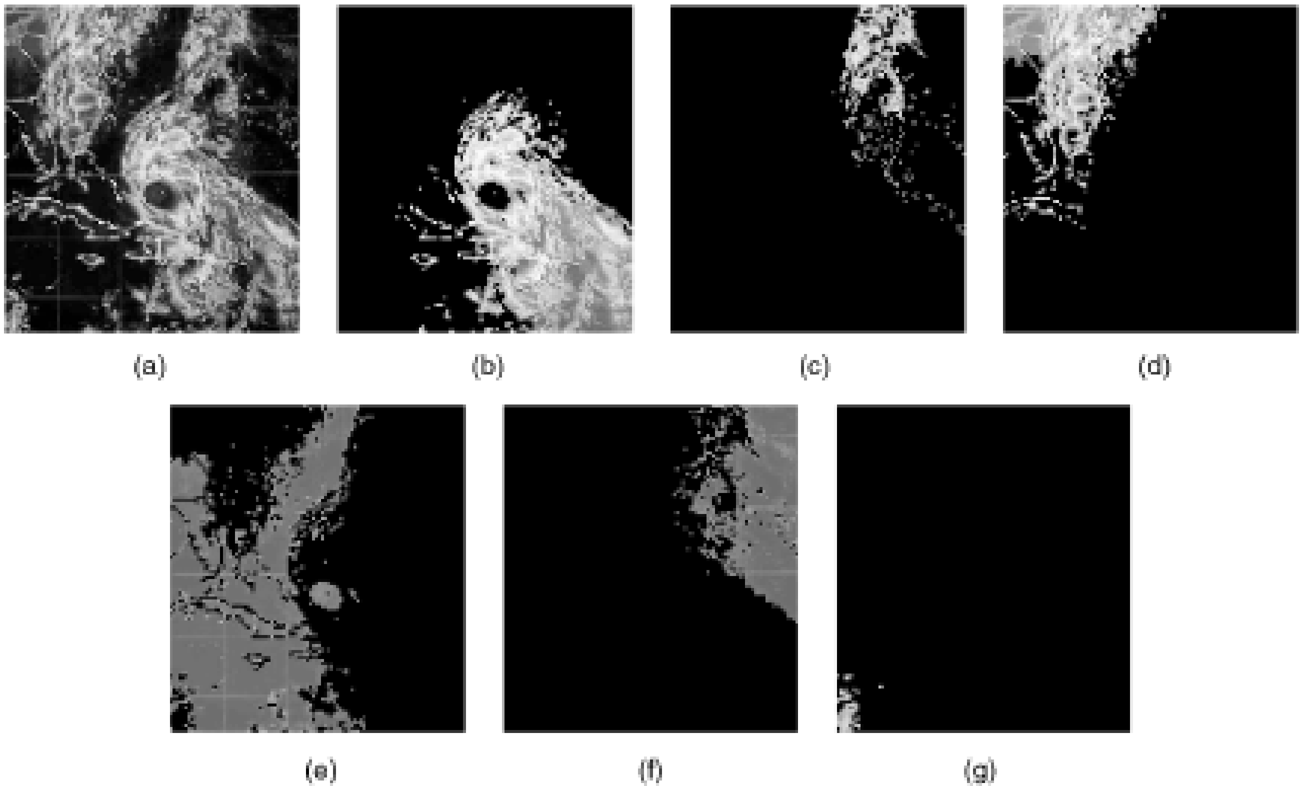
- Synthetic images:





Experiments (III)

- Weather radar:



Experiments (IV)

- Motion segmentation

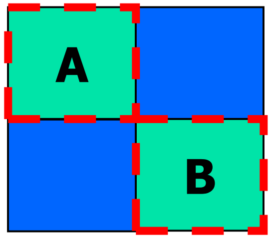
(a) (b)

(c) (d) (e) (f) (g)

Other methods

- Average association
  - Use the eigenvector of W associated to the biggest eigenvalue for partitioning.
  - Tries to maximize:

$$\frac{assoc(A,A)}{|A|}+\frac{assoc(B,B)}{|B|}$$



  - Has a bias to find tight clusters. Useful for Gaussian distributions.

Other methods

- Average cut
  - Tries to minimize:

$$\frac{cut(A,B)}{|A|} + \frac{cut(A,B)}{|B|}$$

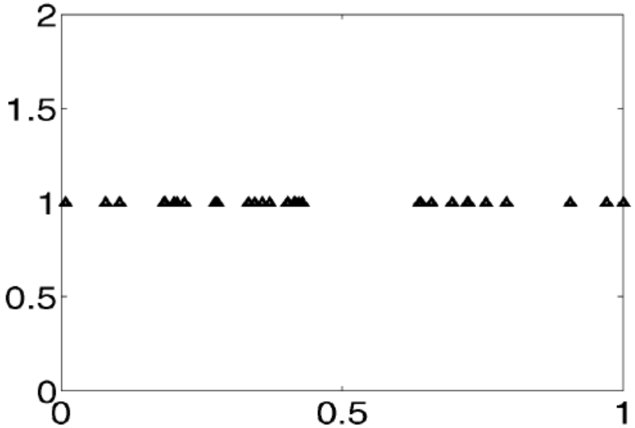
  - Very similar to normalized cuts.
  - We cannot ensure that partitions will have a a tight within-group similarity since this equation does not have the nice properties of the equation of normalized cuts.

Other methods

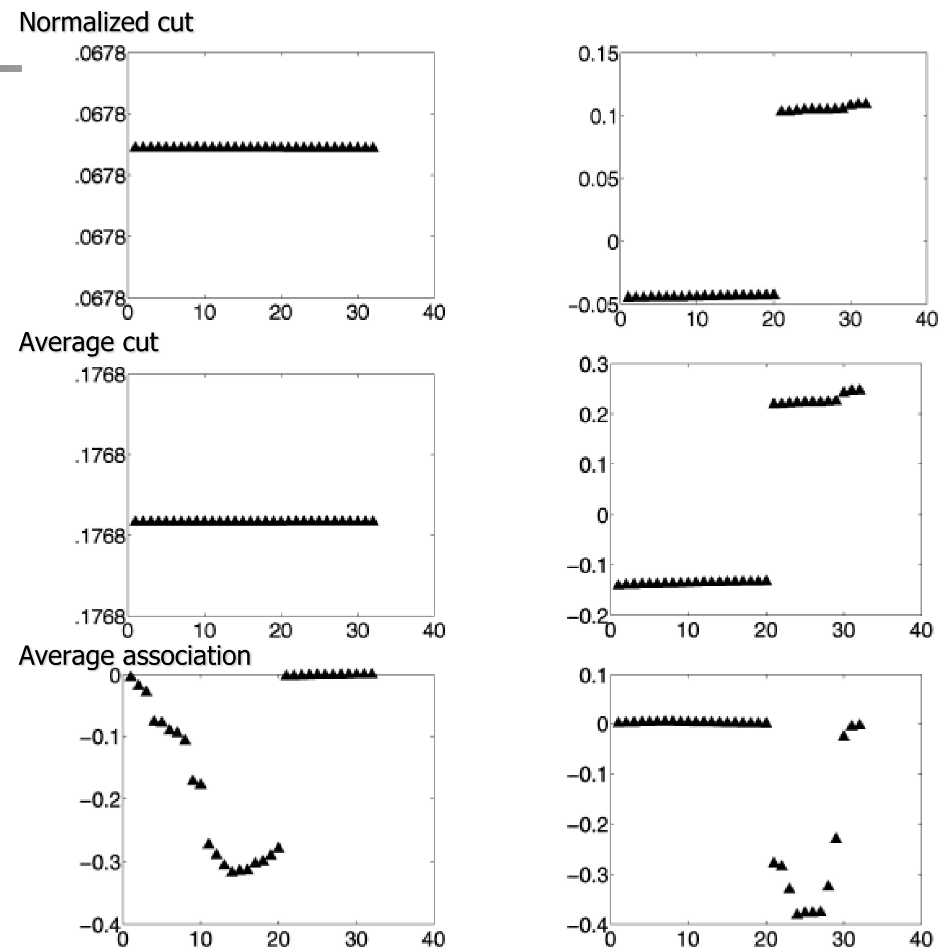
Finding clumps $\longleftrightarrow$ Finding splits

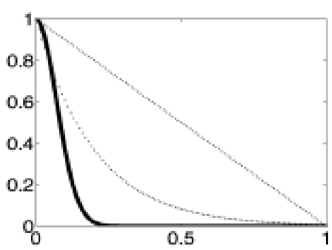
	Average association	Normalized Cut	Average cut
Discrete formulation	$\frac{asso(A,A)}{\lvert A \rvert} + \frac{asso(B,B)}{\lvert B \rvert}$	$\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}$ or $2 - \left(\frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)}\right)$	$\frac{cut(A,B)}{\lvert A \rvert} + \frac{cut(A,B)}{\lvert B \rvert}$
Continuous solution	$Wx = \bar{\lambda} x$	$(D-W)\,x = \bar{\lambda} D\,x$ or $W\,x = (1-\bar{\lambda}) D\,x$	$(D-W)\,x = \bar{\lambda}\,x$

Other methods

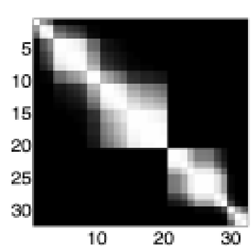


20 points are randomly distributed from 0.0 to 0.5
12 points are randomly distributed from 0.65 to 1.0



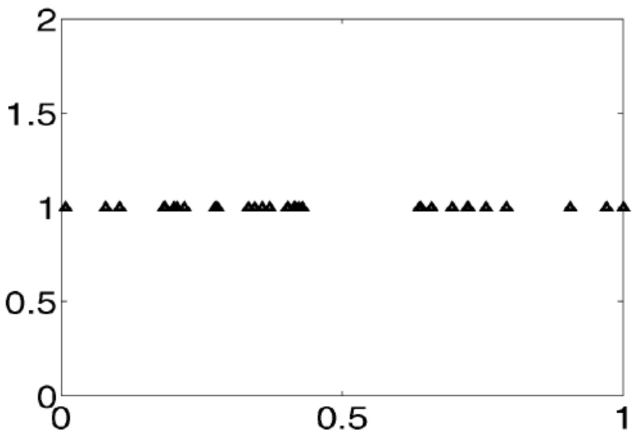


(a)

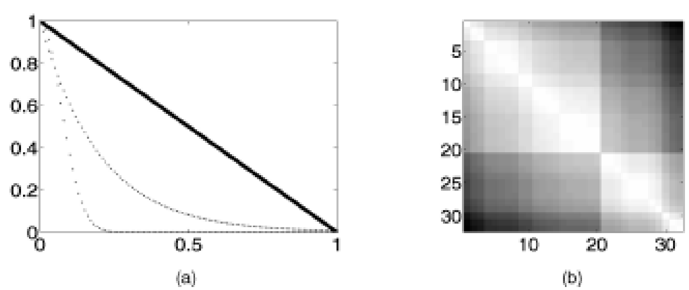


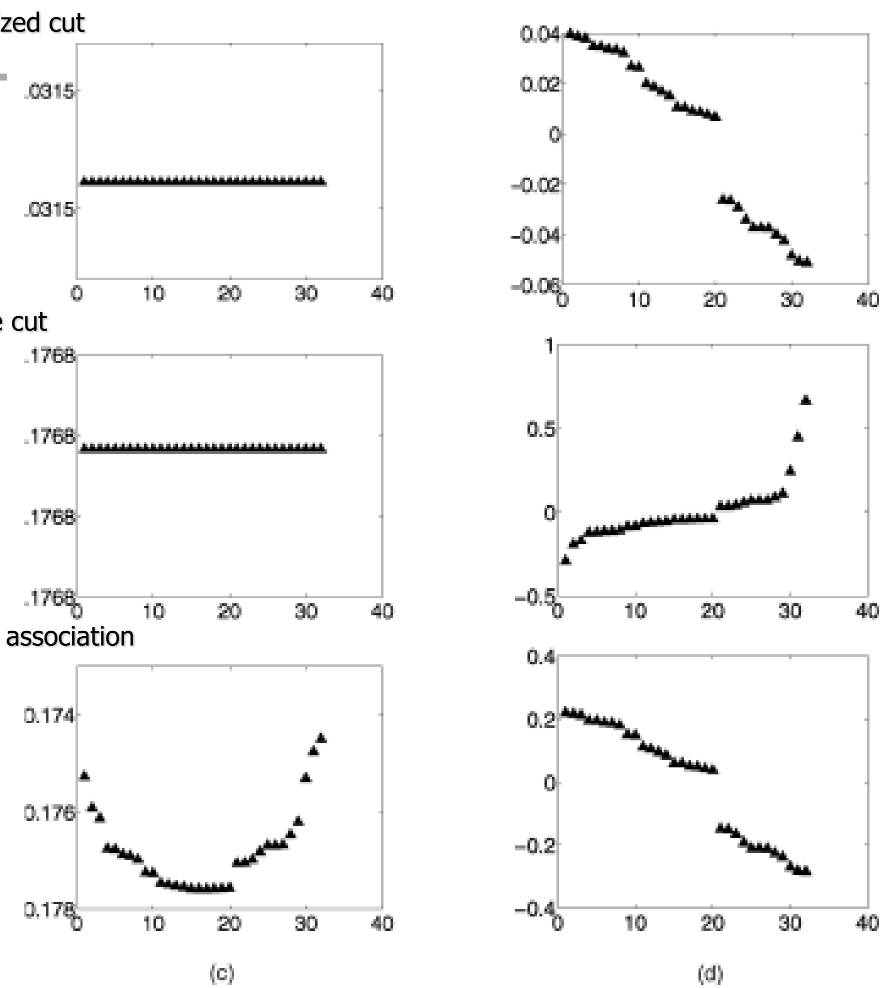
(b)

Other methods

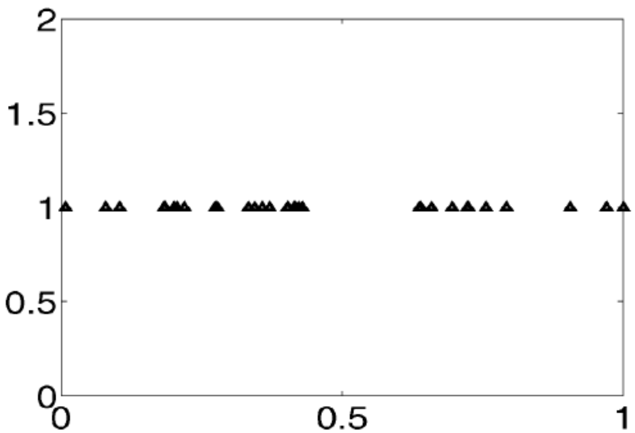


20 points are randomly distributed from 0.0 to 0.5
12 points are randomly distributed from 0.65 to 1.0

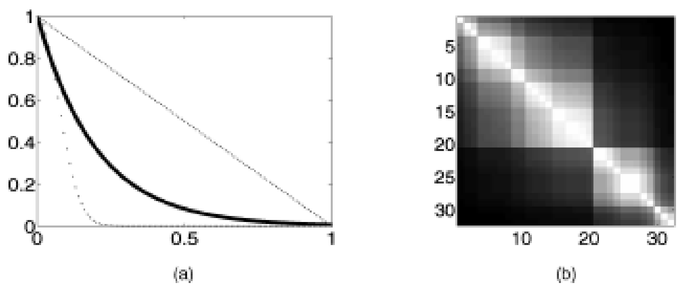


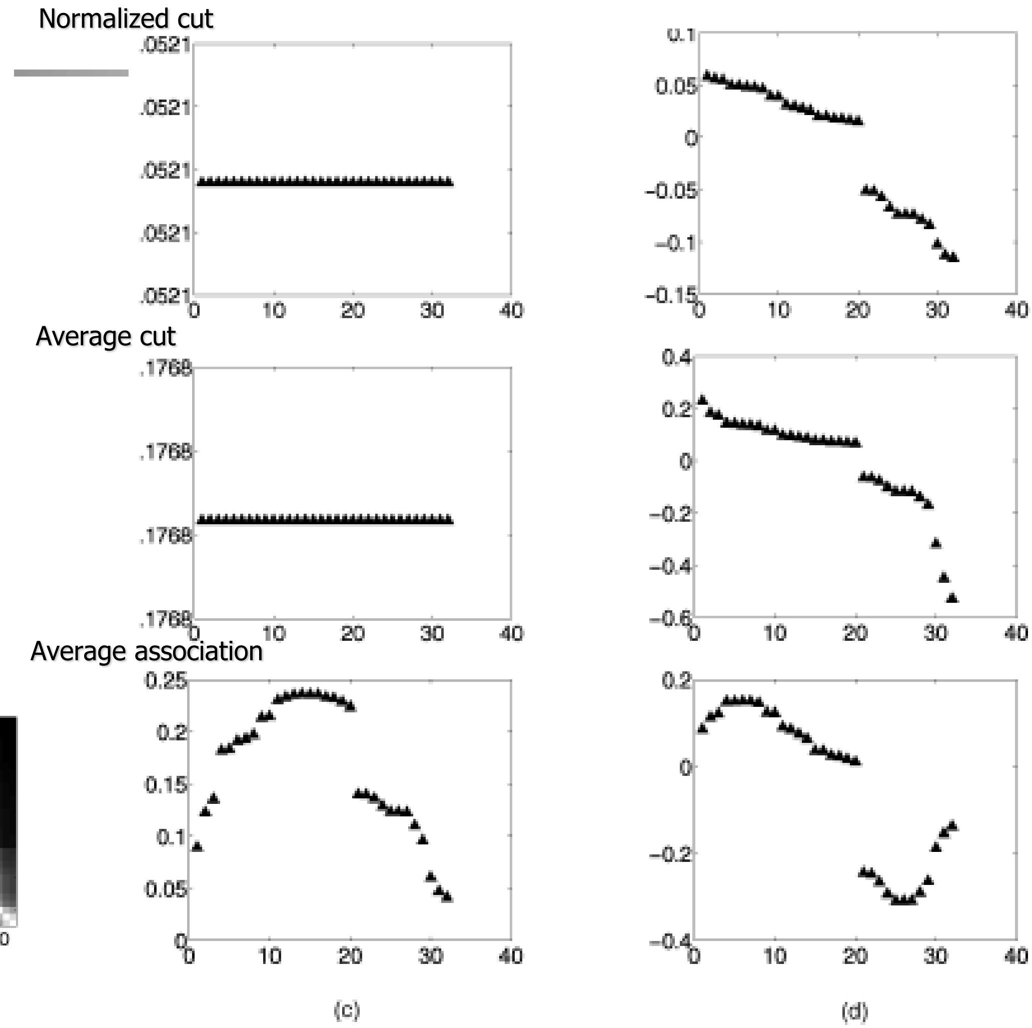


Other methods



20 points are randomly distributed from 0.0 to 0.5
12 points are randomly distributed from 0.65 to 1.0





Spectral Clustering for Image Segmentation

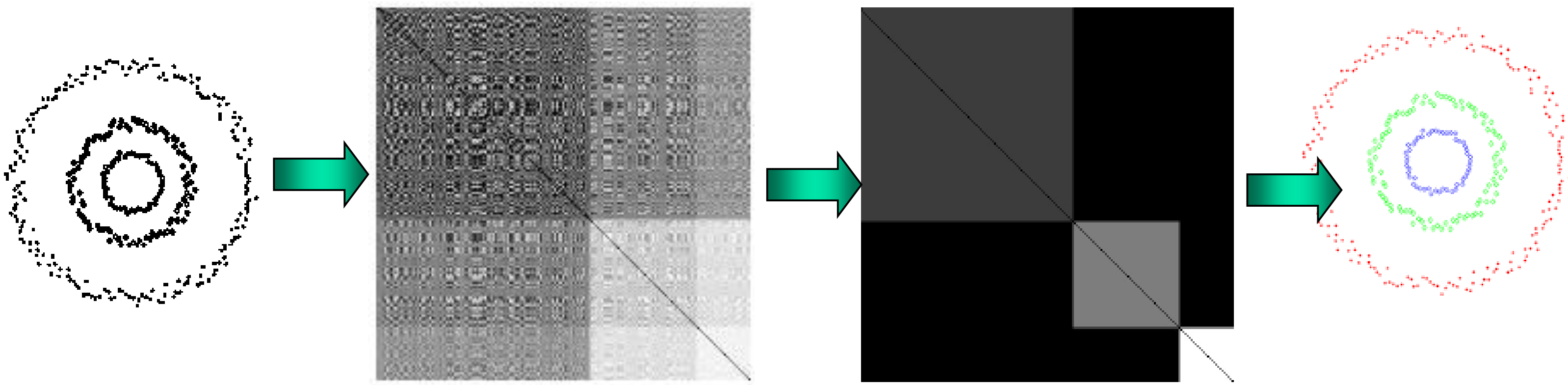
- Good news:
  - Simple and powerful methods to segment images.
  - Flexible and easy to apply to other clustering problems.
- Bad news:
  - High memory requirements (use sparse matrices).
  - Very dependant on the scale factor for a specific problem.

$$W(i,j) = e^{\frac{-\left\| X_{(i)} - X_{(j)} \right\|_2^2}{\sigma_X^2}}$$

Examples

$$w_{i,j} = e^{\frac{-\left\| X_{(i)} - X_{(j)} \right\|_2^2}{\sigma_X^2}}$$

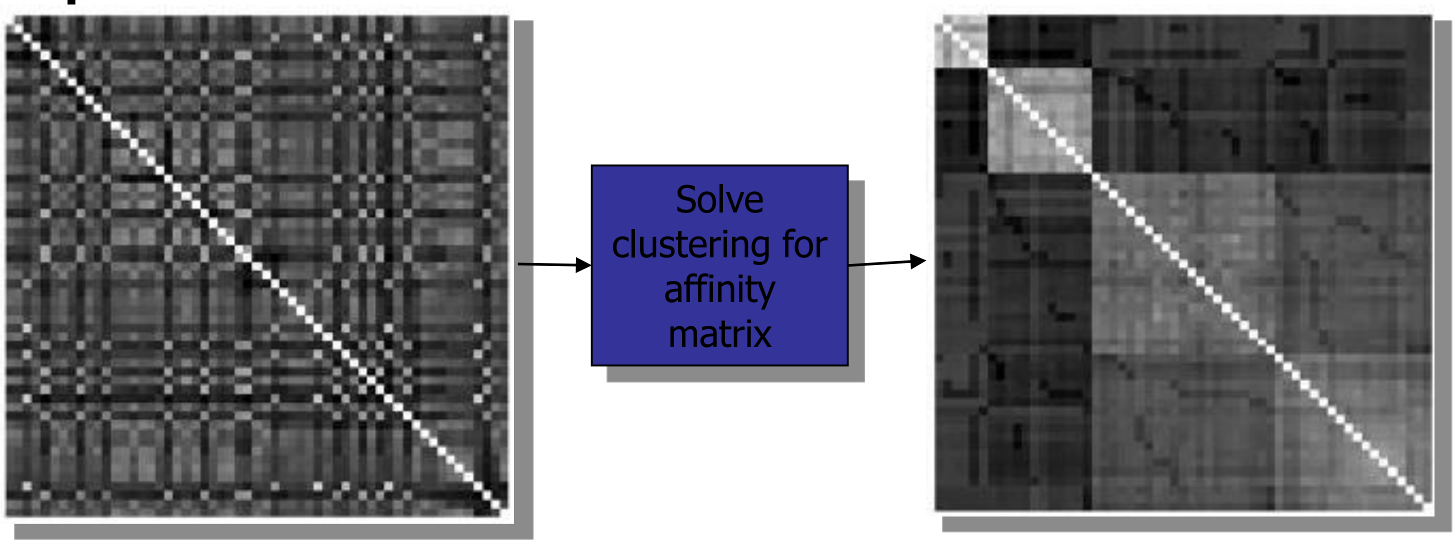
Spectral
Clutering



Images from Matthew Brand (TR-2002-42)

Spectral clustering

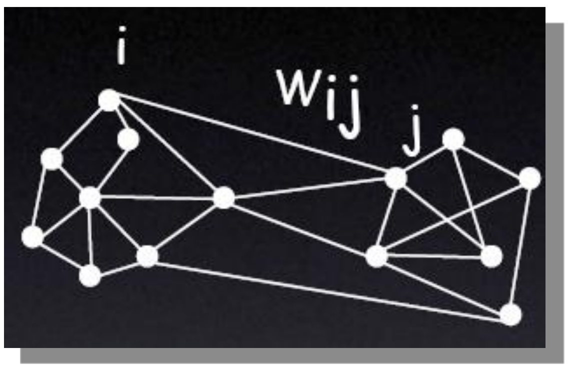
- Makes use of the spectrum of the similarity matrix of the data to cluster the points.



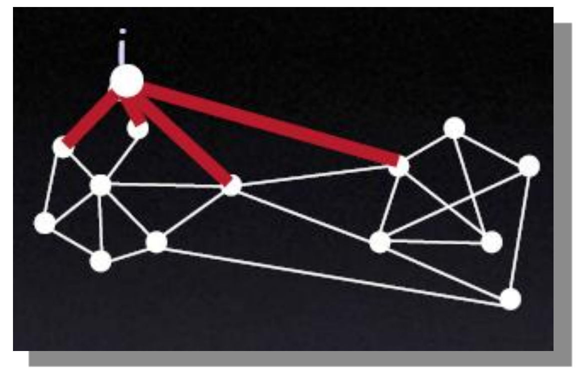
w(i,j)→ distance node i to node j

Graph terminology

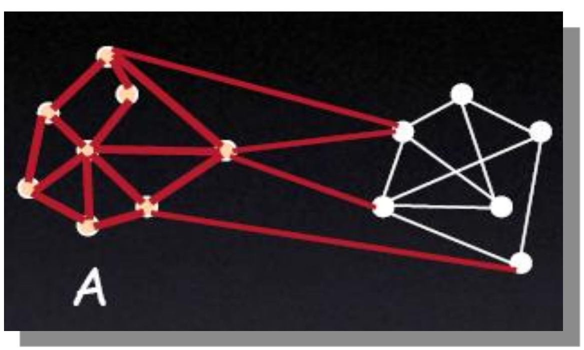
Similarity matrix: $W = \left[w_{i,j} \right]$



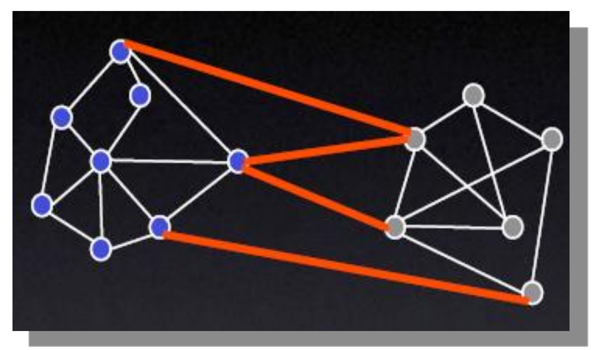
Degree of node: $d_i = \sum w_{i,j}$



Volume of set:



Graph cuts:



Normalized cuts results