

# Color, Texture and Segmentation

ECE P 596

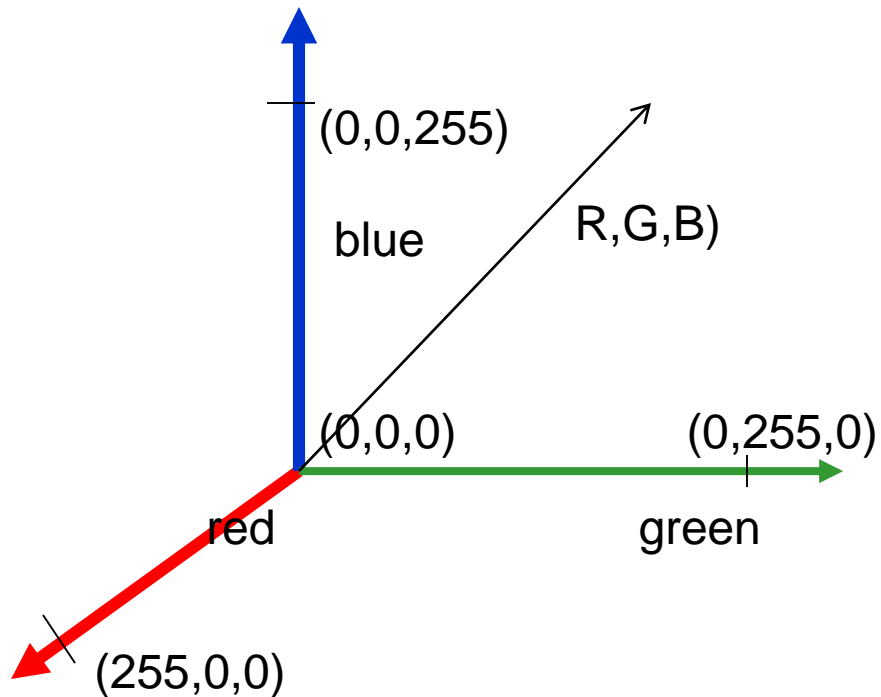
Linda Shapiro

# Color Spaces

- RGB                      standard for cameras
- HSI/HSV                hue, saturation, intensity
- CIE  $L^*a^*b$         intensity plus 2 color channels
- YIQ                      color TVs, Y is intensity
- and more

# RGB Color Space

Absolute



Normalized

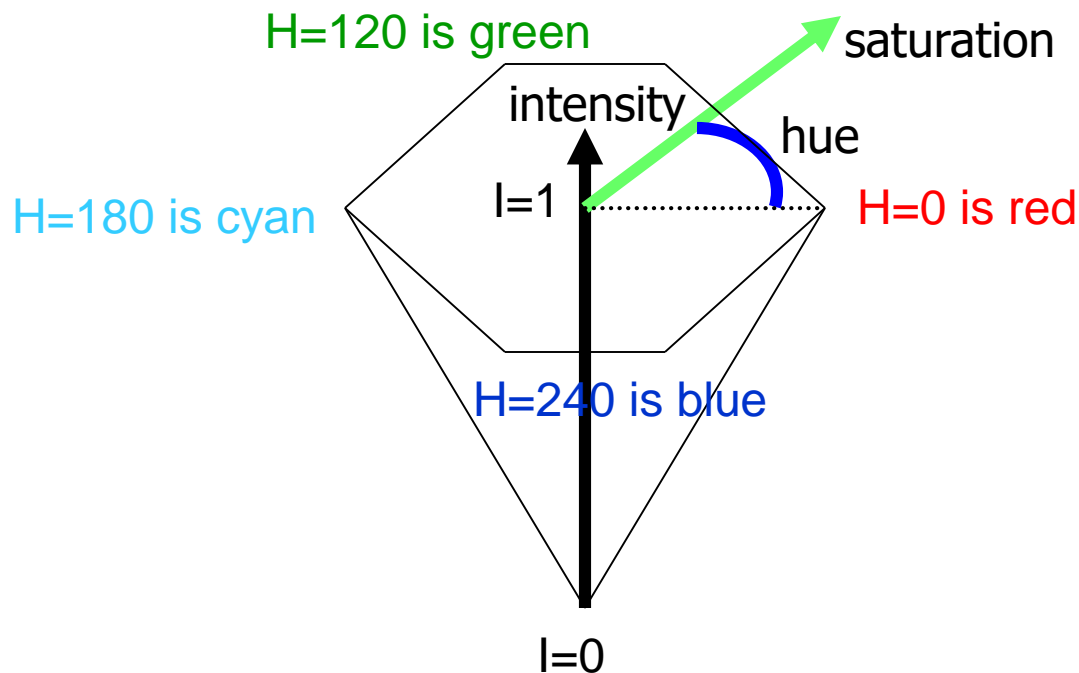
Normalized red  $r = R/(R+G+B)$

Normalized green  $g = G/(R+G+B)$

Normalized blue  $b = B/(R+G+B)$

# Color hexagon for HSI (HSV)

- Hue is encoded as an angle (0 to  $2\pi$ ).
- Saturation is the distance to the vertical axis (0 to 1).
- Intensity is the height along the vertical axis (0 to 1).



# Conversion from RGB to YIQ

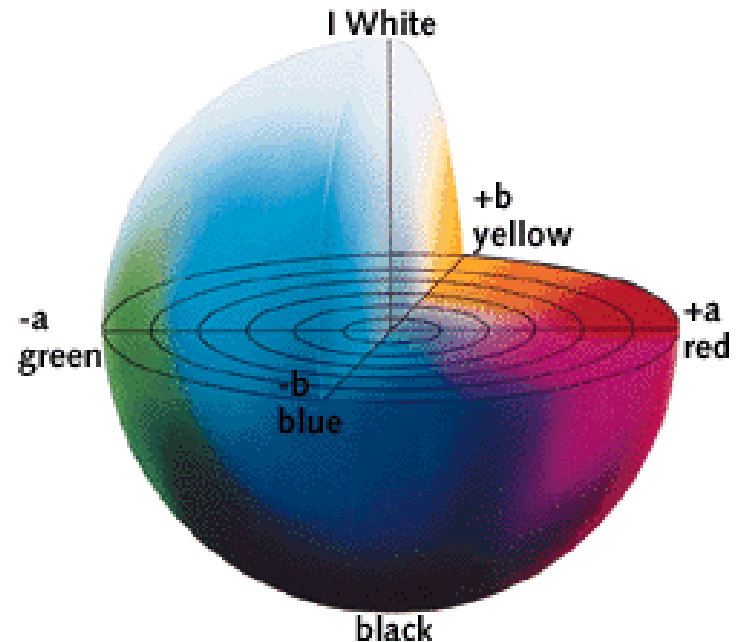
An approximate linear transformation from RGB to YIQ:

$$\begin{array}{lll} \text{luminance } Y & = & 0.30R + 0.59G + 0.11B \\ \text{R - cyan } I & = & 0.60R - 0.28G - 0.32B \\ \text{magenta - green } Q & = & 0.21R - 0.52G + 0.31B \end{array}$$

We often use this for **color to gray-tone conversion**.

# CIELAB, Lab, $L^*a^*b$

- One luminance channel (L) and two color channels (a and b).
- In this model, the color differences which you perceive correspond to Euclidian distances in CIELab.
- The a axis extends from green (-a) to red (+a) and the b axis from blue (-b) to yellow (+b). The brightness (L) increases from the bottom to the top of the three-dimensional model.



# Histograms

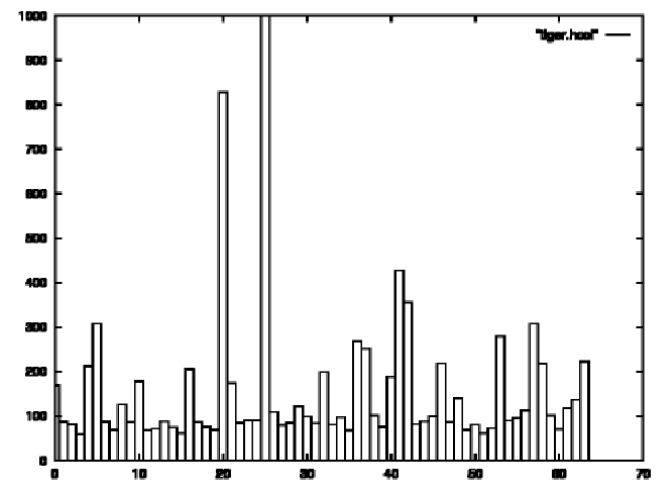
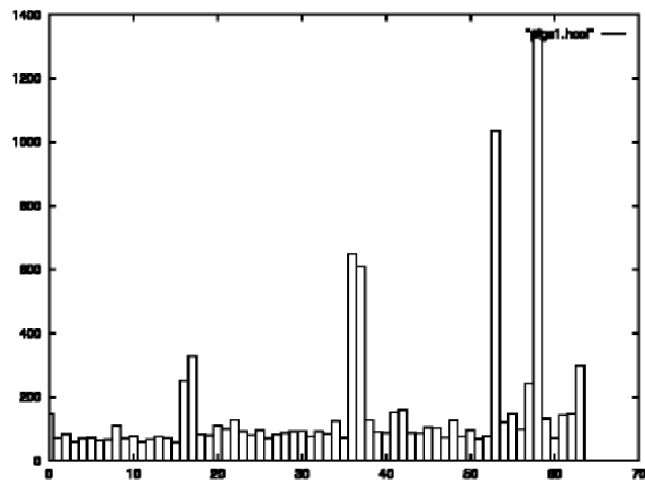
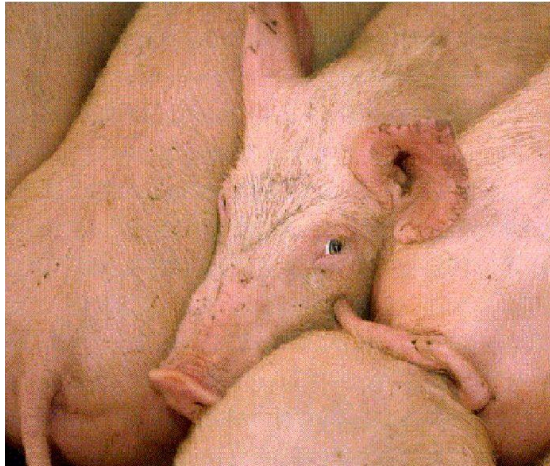
- A histogram of a gray-tone image is an array  $H[*]$  of bins, one for each gray tone.
- $H[i]$  gives the count of how many pixels of an image have gray tone  $i$ .
- $P[i]$  (the normalized histogram) gives the percentage of pixels that have gray tone  $i$ .

# Color histograms can represent an image

- Histogram is fast and easy to compute.
- Size can easily be normalized so that different image histograms can be compared.
- Can match color histograms for database query or classification.



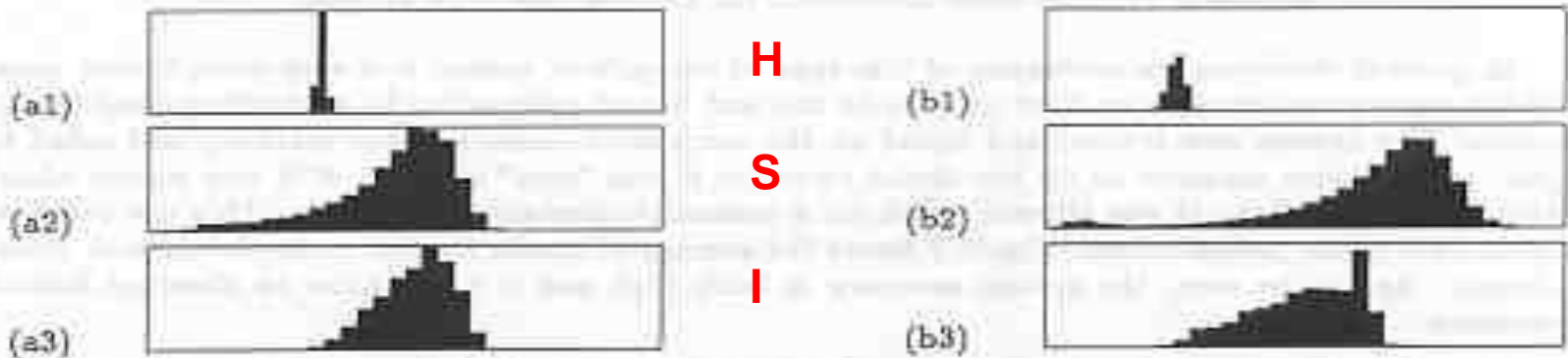
# Histograms of two color images



# How to make a color histogram

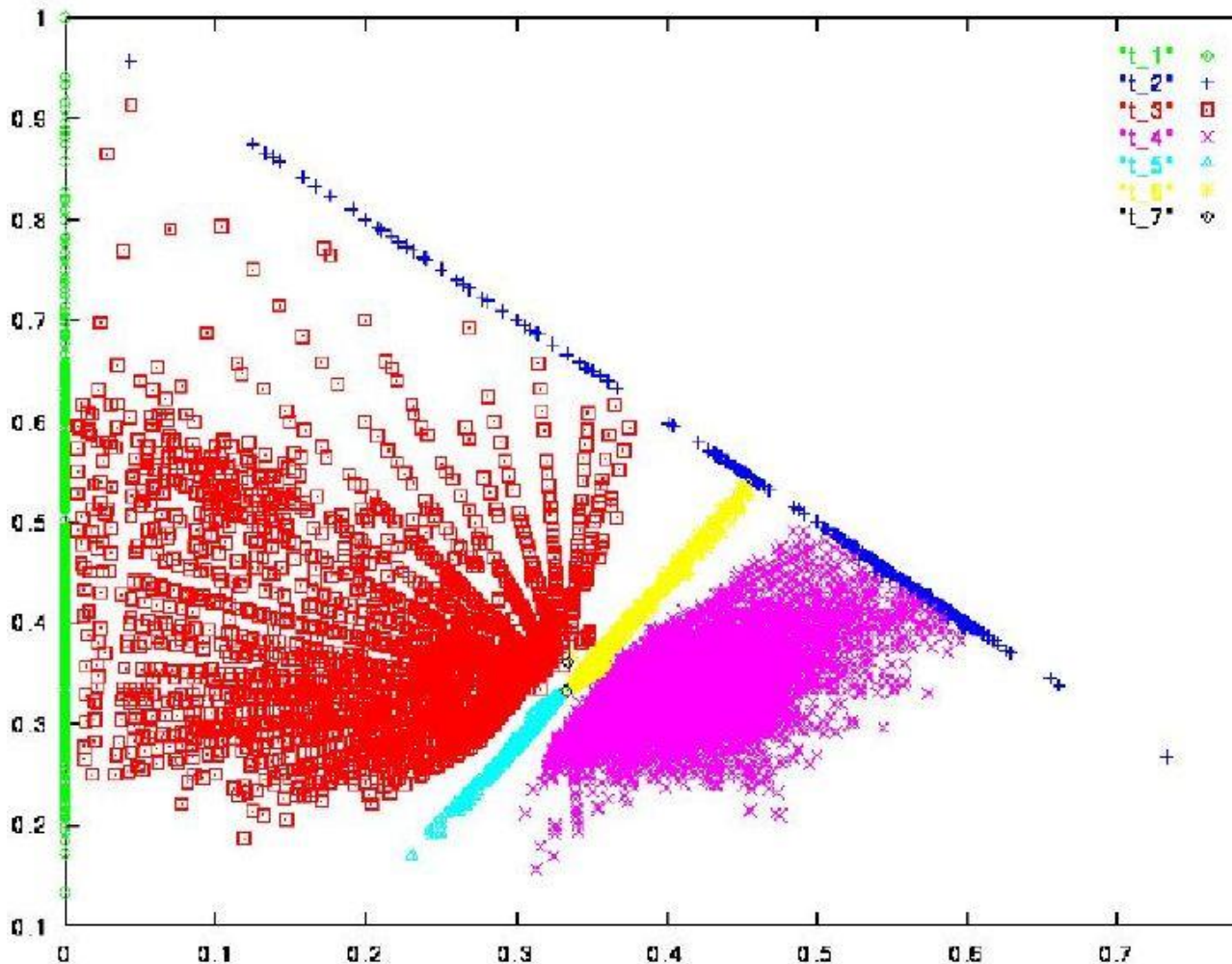
- Make a single 3D histogram.
- Make 3 histograms and concatenate them
- Create a single pseudo color between 0 and 255 by using 3 bits of R, 3 bits of G and 2 bits of B (which bits?)
- Use normalized color space and 2D histograms.

# Apples versus Oranges



Separate HSI histograms for apples (left) and oranges (right) used by IBM's VeggieVision for recognizing produce at the grocery store checkout station (see Ch 16).

# Skin color in RGB space (shown as normalized red vs normalized green)



Purple region shows skin color samples from several people. Blue and yellow regions show skin in shadow or behind a beard.

# Finding a face in video frame



- (left) input video frame
- (center) pixels classified according to RGB space
- (right) largest connected component with aspect similar to a face (all work contributed by Vera Bakic)

# Swain and Ballard's Histogram Matching for Color Object Recognition (IJCV Vol 7, No. 1, 1991)

Opponent Encoding:

- $wb = R + G + B$
- $rg = R - G$
- $by = 2B - R - G$

Histograms:  $8 \times 16 \times 16 = 2048$  bins

Intersection of image histogram and model histogram:

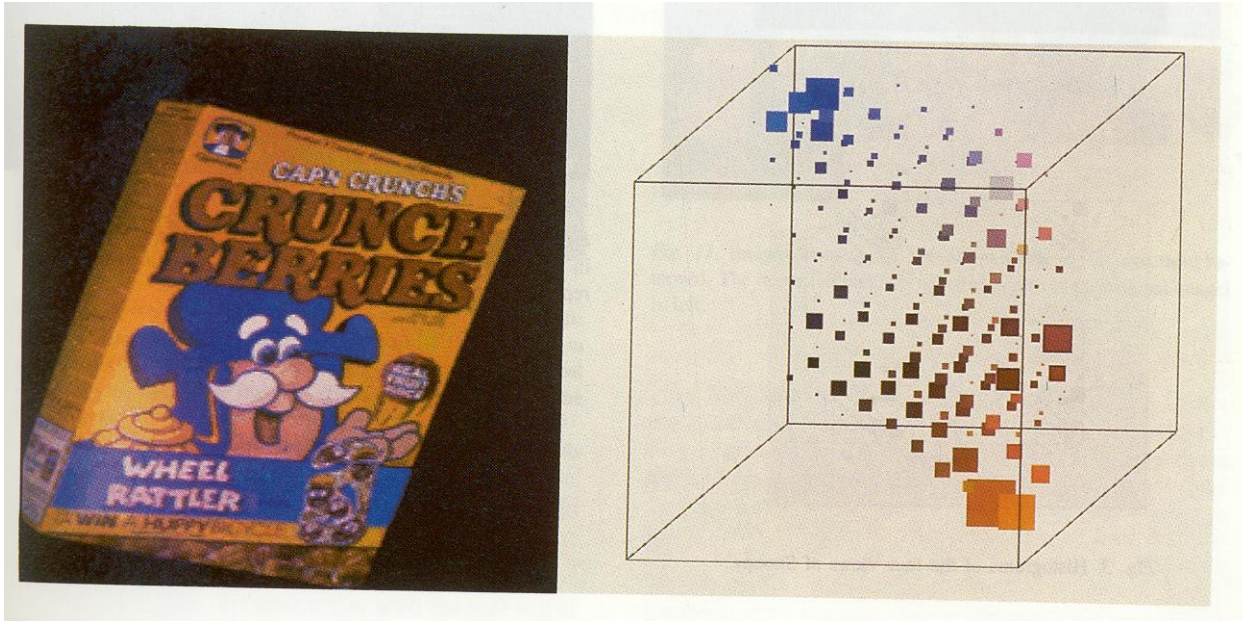
$$\text{intersection}(h(I), h(M)) = \sum_{j=1}^{\text{numbins}} \min\{h(I)[j], h(M)[j]\}$$

Match score is the **normalized** intersection:

$$\text{match}(h(I), h(M)) = \text{intersection}(h(I), h(M)) / \sum_{j=1}^{\text{numbins}} h(M)[j]$$

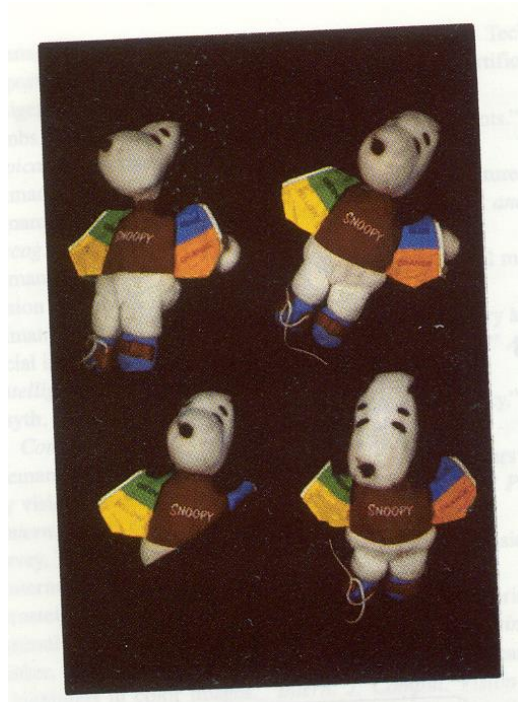


(from Swain and Ballard)

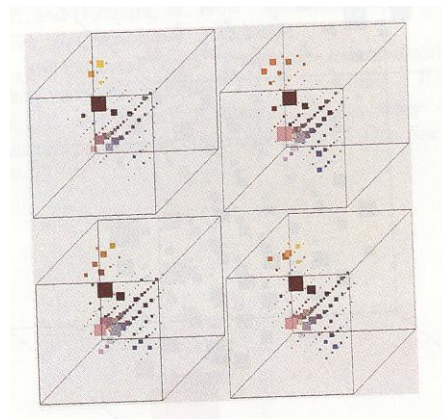


cereal box image

3D color histogram

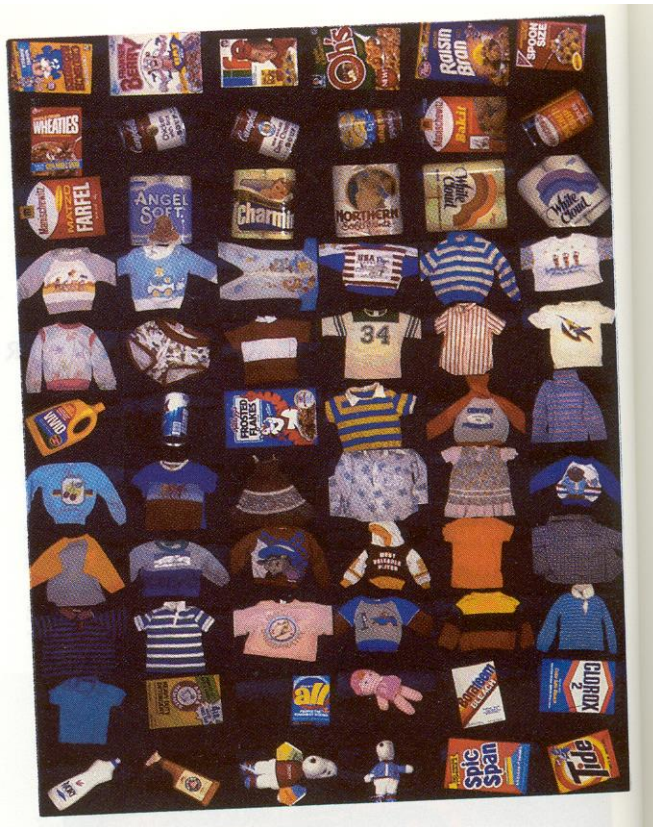


Four views of Snoopy



Histograms





The 66 models objects



Some test objects

# Swain and Ballard Results

Results were surprisingly good.

At their highest resolution (128 x 90), average match percentile (with and without occlusion) was 99.9.

This translates to 29 objects matching best with their true models and 3 others matching second best with their true models.

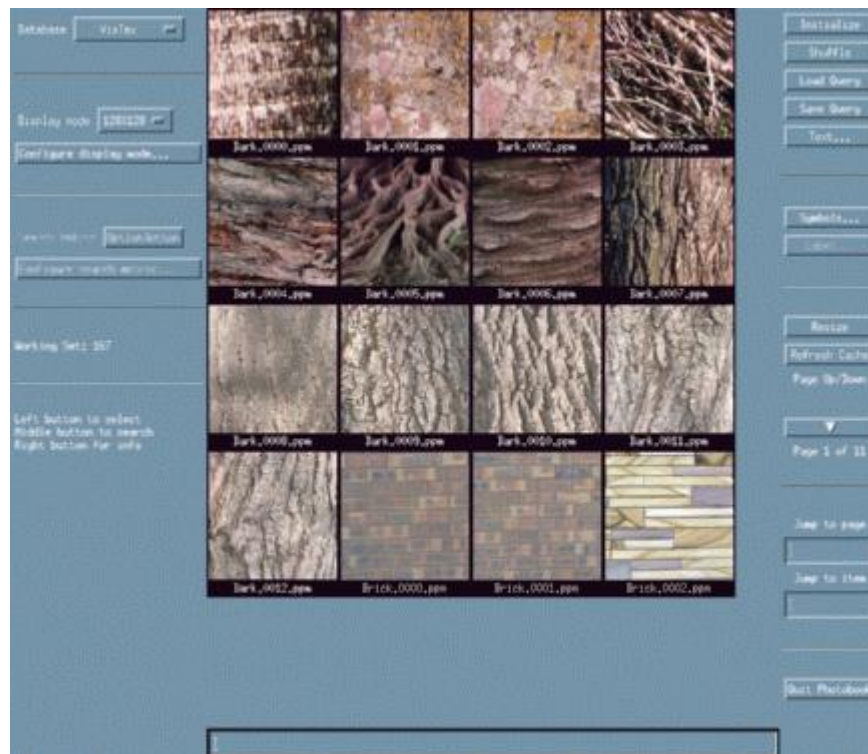
At resolution 16 X 11, they still got decent results (15 6 4) in one experiment; (23 5 3) in another.

# Uses

- Although this is an extremely simple technique, it became the **basis for many content-based image retrieval systems** and works surprisingly well, both alone, and in conjunction with other techniques.

# Texture

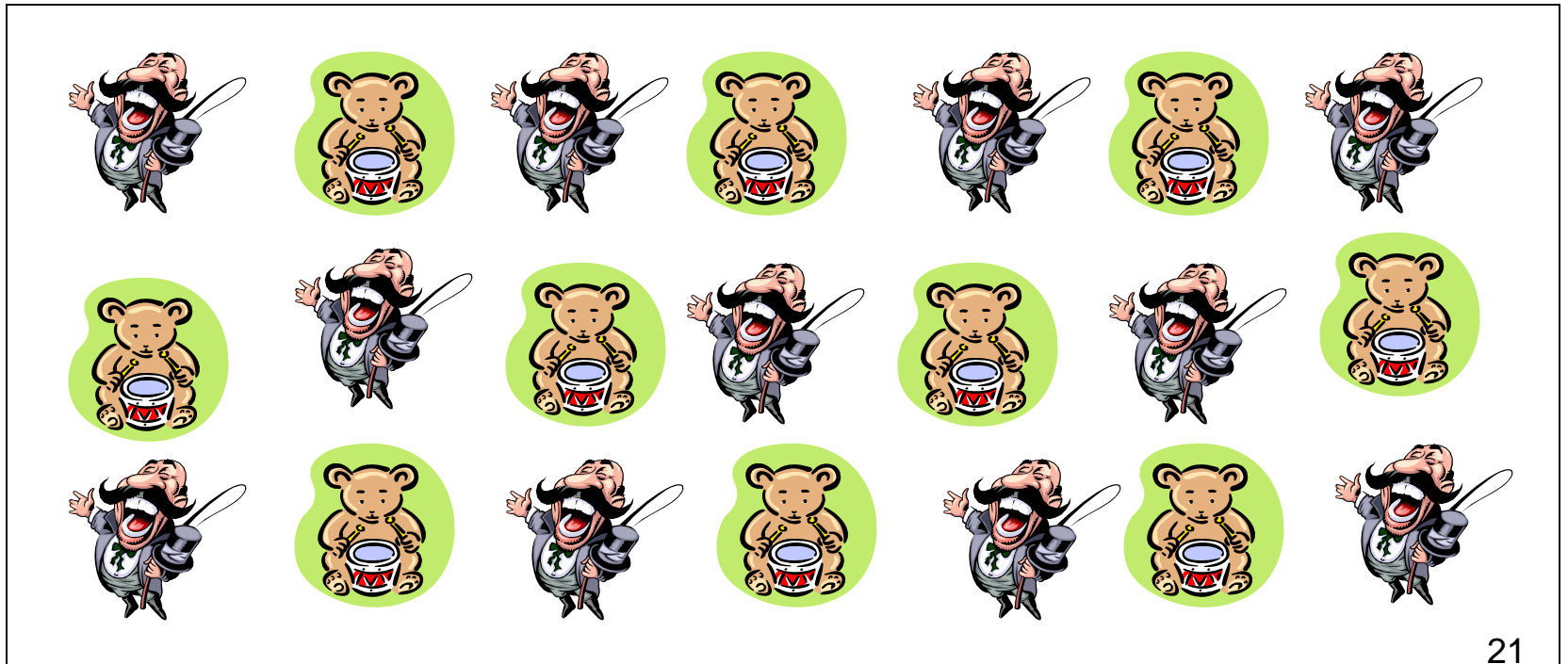
- Color is well defined.
- But what *is* texture?



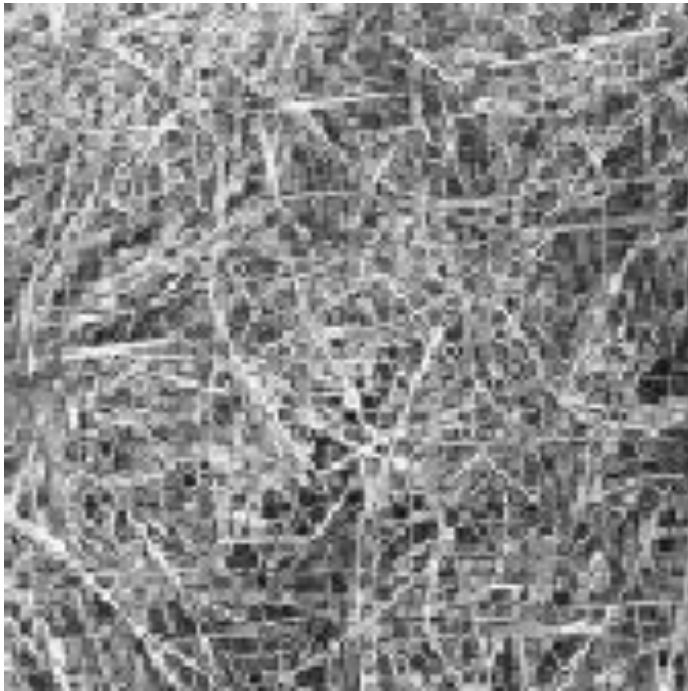
# Structural Texture

**Texture is a description of the spatial arrangement of color or intensities in an image or a selected region of an image.**

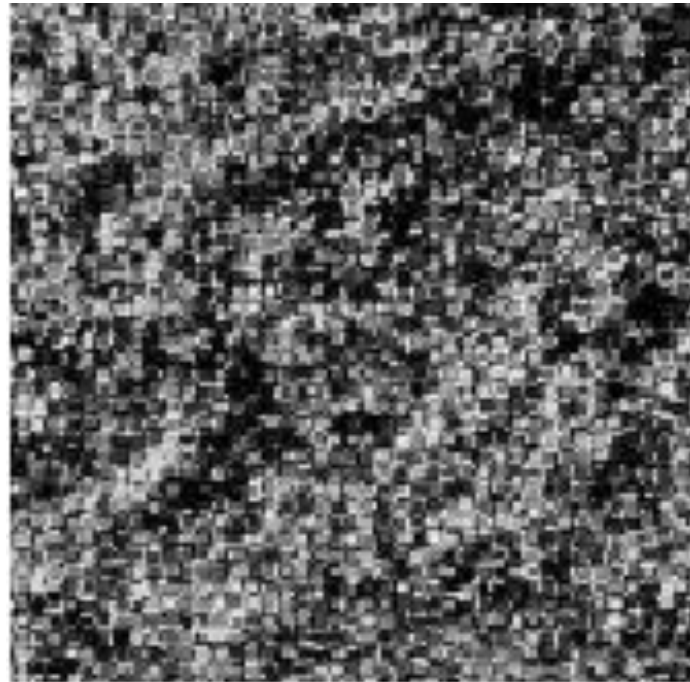
**Structural approach: a set of texels in some regular or repeated pattern**



# Natural Textures from VisTex



**grass**



**leaves**

**What/Where are the texels?**



# The Case for Statistical Texture

- Segmenting out texels is difficult or impossible in real images.
- Numeric quantities or statistics that describe a texture can be computed from the gray tones (or colors) alone.
- This approach is less intuitive, but is computationally efficient.
- It can be used for both classification and segmentation.

# Some Simple Statistical Texture Measures

## 1. Edge Density and Direction

- Use an edge detector as the first step in texture analysis.
- The number of edge pixels in a fixed-size region tells us how busy that region is.
- The directions of the edges also help characterize the texture



# Two Edge-based Texture Measures

1. edgeness per unit area

$$\mathbf{F_{edgeness}} = |\{ \mathbf{p} \mid \mathbf{gradient\_magnitude(p)} \geq \mathbf{threshold} \}| / \mathbf{N}$$

where N is the size of the unit area

2. edge magnitude and direction histograms

$$\mathbf{F_{magdir}} = ( \mathbf{H_{magnitude}}, \mathbf{H_{direction}} )$$

where these are the normalized histograms of gradient magnitudes and gradient directions, respectively.

# Example

Original Image



Frei-Chen  
Edge Image

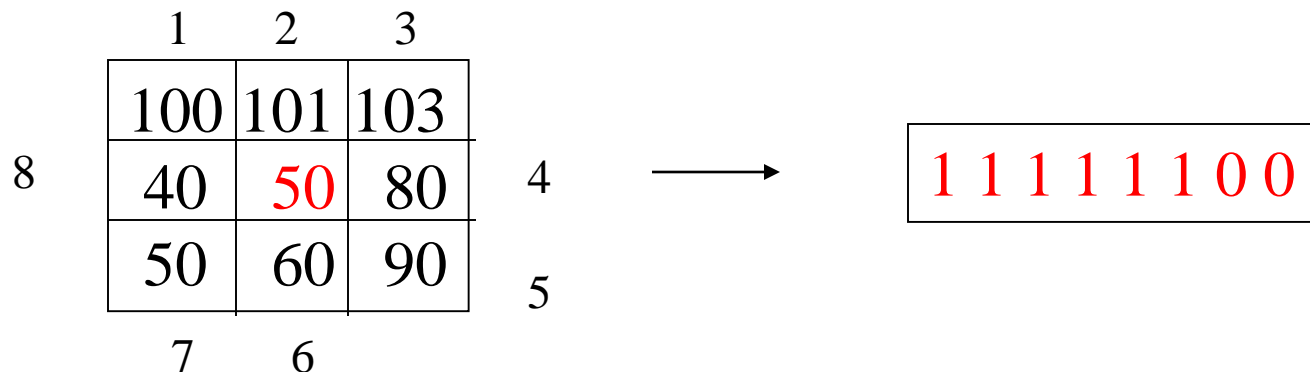


Thresholded  
Edge Image



# Local Binary Pattern Measure

- For each pixel  $p$ , create an 8-bit number  $b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8$ , where  $b_i = 0$  if neighbor  $i$  has value less than or equal to  $p$ 's value and 1 otherwise.
- Represent the texture in the image (or a region) by the histogram of these numbers.



# Example

Fids (Flexible Image Database System) is retrieving images similar to the query image using LBP texture as the texture measure and comparing their LBP histograms

## Fids demo

The screenshot displays the Fids demo interface. At the top, a query image of a garden path is shown in a red-bordered box. Below it, six similar images are displayed in a 2x3 grid. The interface includes a control bar with buttons for 'Random', 'Go', 'ZoomIn', and 'ZoomOut'. To the right of these buttons, it says 'Found 191 matches. Displaying 1 - 6'. Below the control bar, there are checkboxes for various distance measures: 'ColorHistL14x4x4', 'ColorHist8x8x8' (which is highlighted with a dashed border), 'SobelEdgeHist', 'LBPHist' (which is checked), 'fleshiness', and 'Wavelets'. To the right of these checkboxes are sliders for 'loose ... strict' settings, each with a value of 5. On the far right, there are three radio buttons for 'And', 'Or', and 'Sum' operations. At the bottom, it says 'Server Connected'.

# Example

## Fids demo

Low-level measures don't always find semantically similar images.

The screenshot shows the Fids demo interface. At the top, a grid of six images is displayed. The first image in the top row is highlighted with a red border. Below the grid, there are buttons for 'Random', 'Go', and 'ZoomIn'. To the right of these buttons, it says 'Found 119 matches. Displaying 1 - 6'. Below this, there are two columns of distance measures. The first column lists the measures: ColorHistL14x4x4, ColorHist8x8x8, SobelEdgeHist, LBPHist (checked), fleshiness, and Wavelets. The second column shows a slider for each measure, ranging from 'loose' to 'strict', with a value of 5. To the right of the sliders, there are three radio buttons: 'And' (selected), 'Or', and 'Sum'. On the far right, there are two buttons: 'Put In Cart' and 'Check Out'. Below these buttons, there is a text box that says 'A double click on an image means:' followed by two radio buttons: 'Set query / Go' (selected) and 'Zoom in'. At the bottom left, it says 'Server Connected'.

Found 119 matches. Displaying 1 - 6

distance measures    loose ... strict

Measure	Value
<input type="checkbox"/> ColorHistL14x4x4	5
<input type="checkbox"/> ColorHist8x8x8	5
<input type="checkbox"/> SobelEdgeHist	5
<input checked="" type="checkbox"/> LBPHist	5
<input type="checkbox"/> fleshiness	5
<input type="checkbox"/> Wavelets	5

And  
Or  
Sum

A double click on an image means:  
☒ Set query / Go  
☐ Zoom in

Server Connected

# What else is LBP good for?

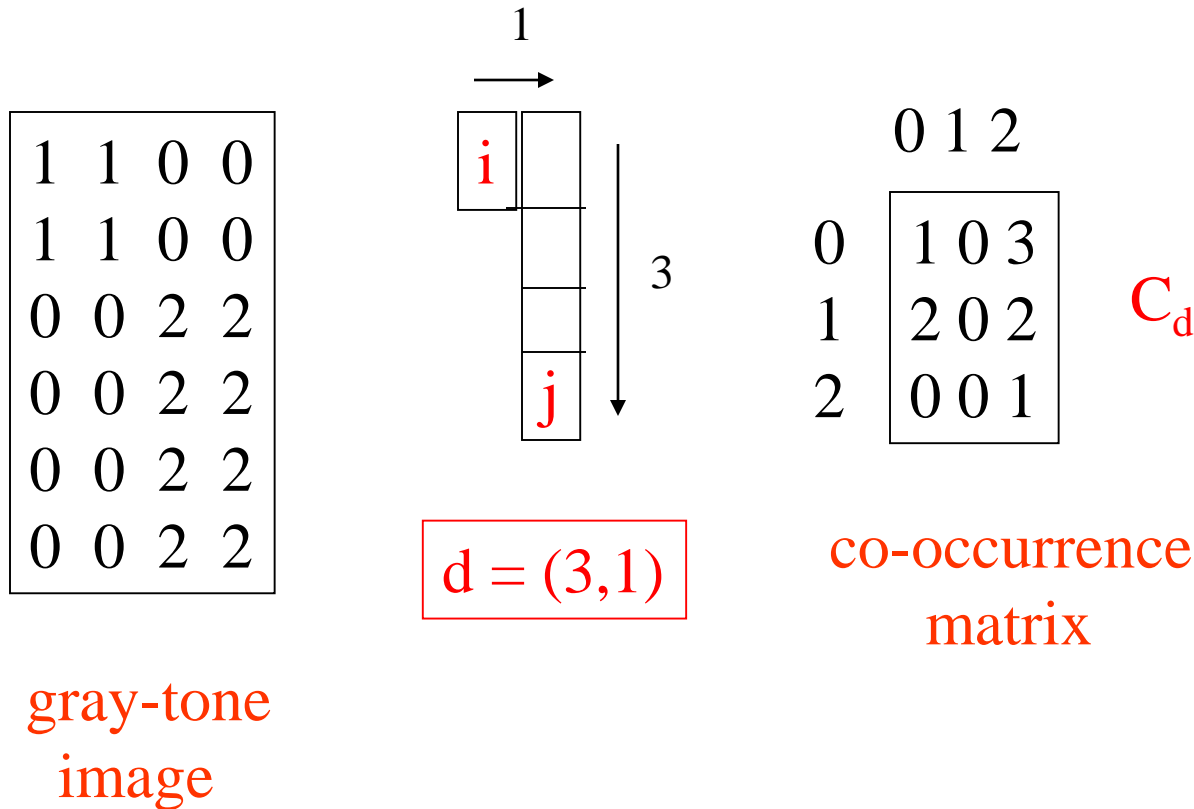
- We found it in a paper for classifying deciduous trees.
- We used it in a real system for finding cancer in Pap smears.
- We are using it to look for regions of interest in breast and melanoma biopsy slides.

# Co-occurrence Matrix Features

A co-occurrence matrix is a 2D array  $C$  in which

- Both the rows and columns represent a set of possible image values.
- $C_d(i,j)$  indicates how many times value  $i$  co-occurs with value  $j$  in a particular spatial relationship  $d$ .
- The spatial relationship is specified by a vector  $d = (dr,dc)$ .

# Co-occurrence Example



From  $C_d$  we can compute  $N_d$ , the normalized co-occurrence matrix, where each value is divided by the sum of all the values.



# Co-occurrence Features

What do these measure?

$$\text{Energy} = \sum_i \sum_j N_d^2(i, j) \quad (7.7)$$

$$\text{Entropy} = - \sum_i \sum_j N_d(i, j) \log_2 N_d(i, j) \quad (7.8)$$

$$\text{Contrast} = \sum_i \sum_j (i - j)^2 N_d(i, j) \quad (7.9)$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{N_d(i, j)}{1 + |i - j|} \quad (7.10)$$

$$\text{Correlation} = \frac{\sum_i \sum_j (i - \mu_i)(j - \mu_j) N_d(i, j)}{\sigma_i \sigma_j} \quad (7.11)$$

where  $\mu_i, \mu_j$  are the means and  $\sigma_i, \sigma_j$  are the standard deviations of the row and column sums.

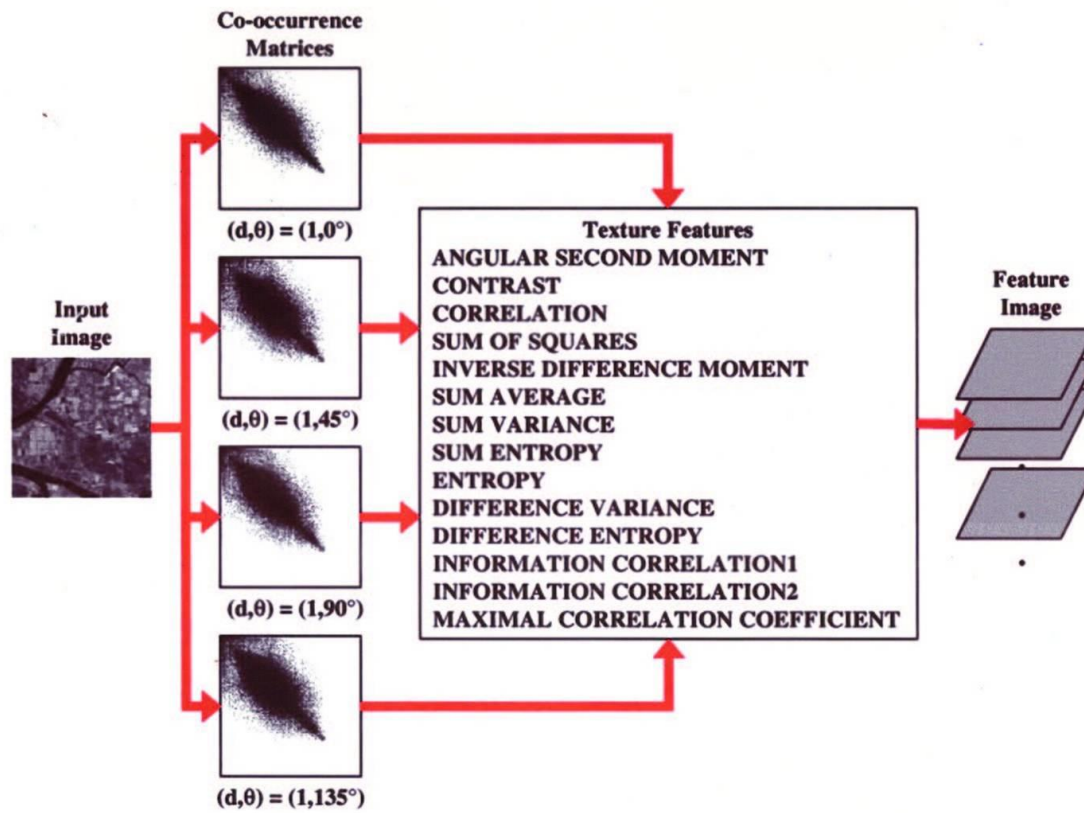
Energy measures uniformity of the normalized matrix.

# But how do you choose d?

- This is actually a critical question with **all** the statistical texture methods.
- Are the “texels” tiny, medium, large, all three ...?
- Not really a solved problem.

Zucker and Terzopoulos suggested using a  $\chi^2$  statistical test to select the value(s) of d that have the most structure for a given class of images.

# Example



# What are Co-occurrence Features used for?

- They were designed for recognizing different kinds of **land uses in satellite images**.
- They are still used heavily in geospatial domains, but they can be added on to any other calculated features.

# Laws' Texture Energy Features

- Signal-processing-based algorithms use texture filters applied to the image to create filtered images from which texture features are computed.
- The Laws Algorithm
  - **Filter** the input image using texture filters.
  - **Compute texture energy** by summing the absolute value of filtering results in local neighborhoods around each pixel.
  - **Combine features** to achieve rotational invariance.

# Law's texture masks (1)

$$\begin{array}{lll} \text{L5} & (\text{Level}) & = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \\ \text{E5} & (\text{Edge}) & = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \end{bmatrix} \\ \text{S5} & (\text{Spot}) & = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \end{bmatrix} \\ \text{R5} & (\text{Ripple}) & = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix} \end{array}$$

- (L5) (Gaussian) gives a center-weighted local average
- (E5) (gradient) responds to row or col step edges
- (S5) (LOG) detects spots
- (R5) (Gabor) detects ripples

# Law's texture masks (2)

## Creation of 2D Masks

- 1D Masks are “multiplied” to construct 2D masks:  
mask E5L5 is the “product” of E5 and L5 –

$$\begin{array}{c} \text{E5} \end{array} \begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} \times \begin{array}{c} \text{L5} \end{array} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{array}{c} \text{E5L5} \end{array} \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

# 9D feature vector for pixel

- Subtract mean neighborhood intensity from (center) pixel
- Apply 16 5x5 masks to get 16 filtered images  $F_k$ ,  $k=1$  to 16
- Produce 16 texture energy maps using 15x15 windows

$$E_k[r,c] = \sum |F_k[i,j]|$$

- Replace each distinct pair with its average map:
- 9 features (9 filtered images) defined as follows:

L5E5/E5L5

L5R5/R5L5

E5S5/S5E5

S5S5

R5R5

L5S5/S5L5

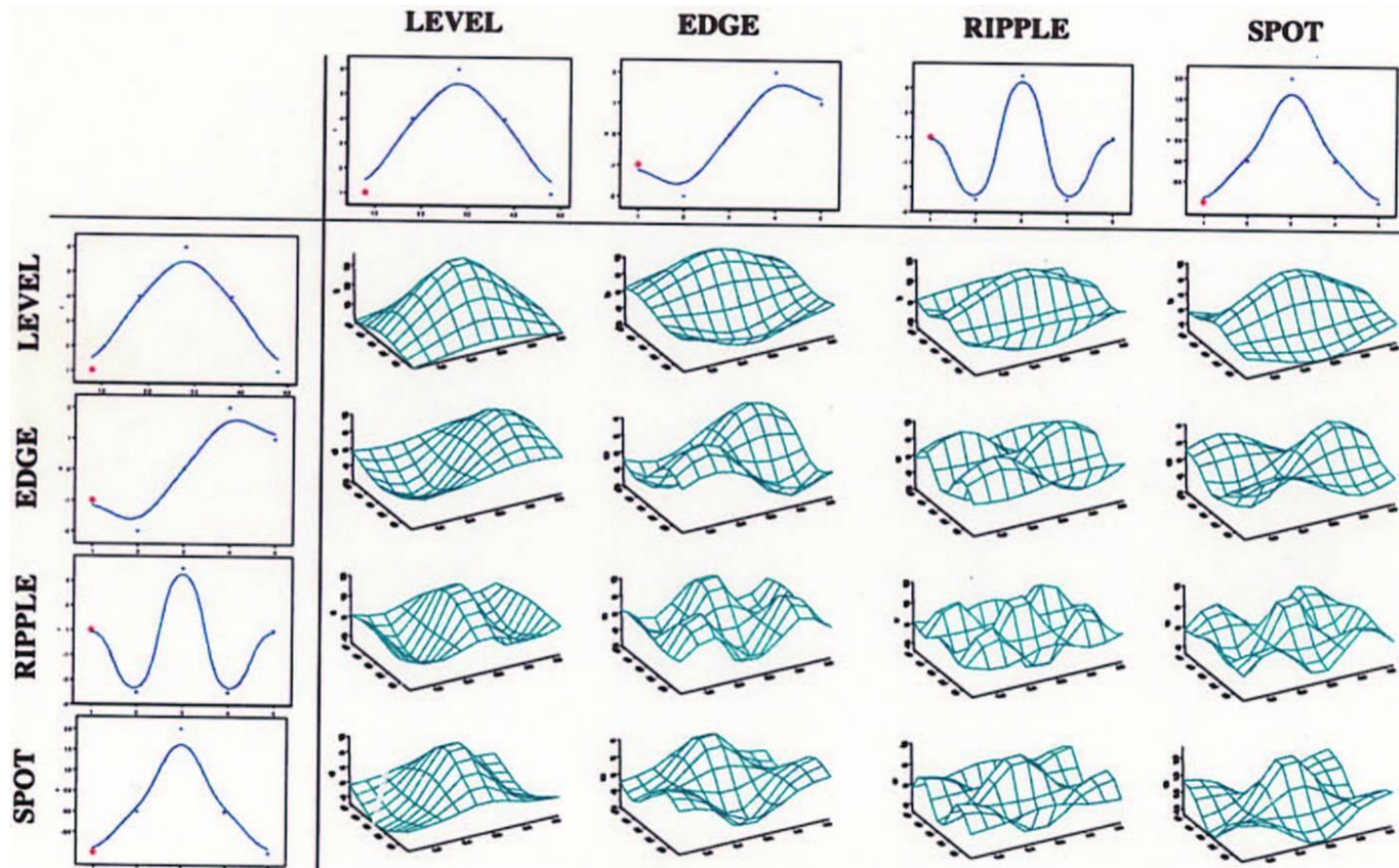
E5E5

E5R5/R5E5

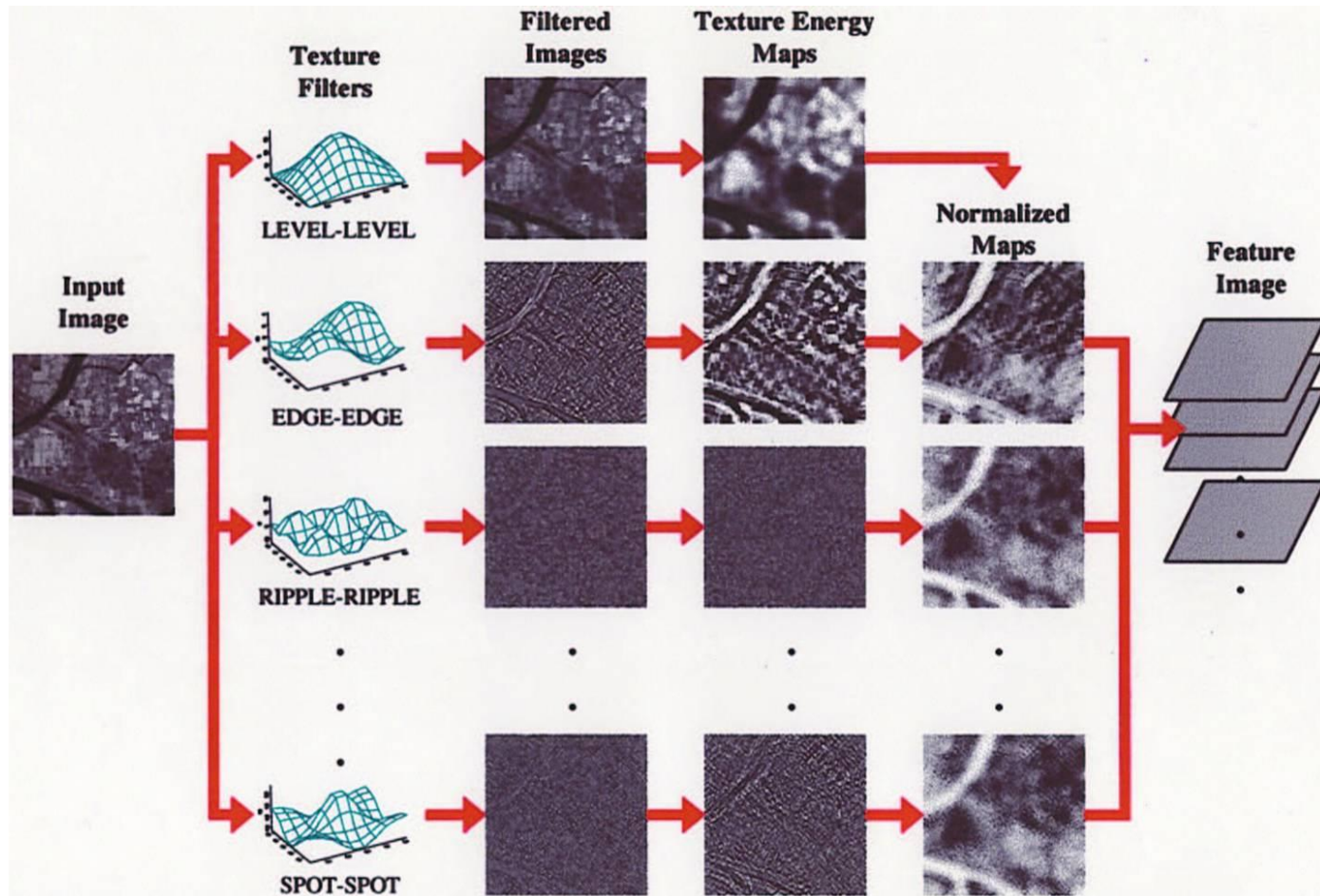
S5R5/R5S5



# Laws Filters



# Laws Process



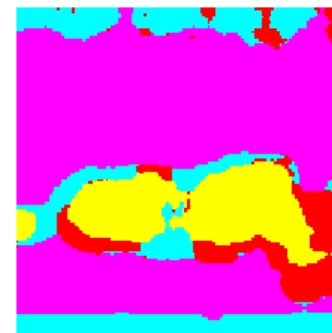
# Example: Using Laws Features to Cluster

water

tiger



(a) Original image

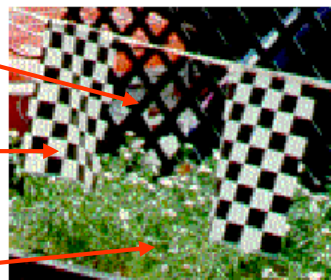


(b) Segmentation into 4 clusters

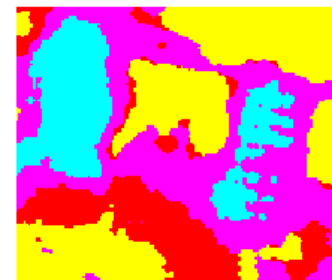
fence

flag

grass



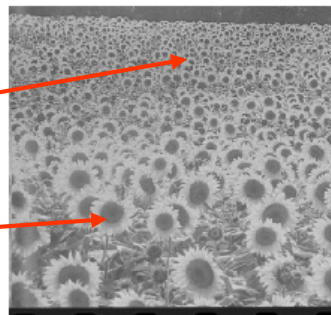
(c) Original image



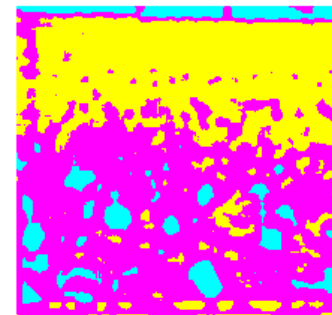
(d) Segmentation into 4 clusters

small flowers

big flowers



(e) Original image



(f) Segmentation into 3 clusters

Is there a neighborhood size problem with Laws?

# Features from sample images

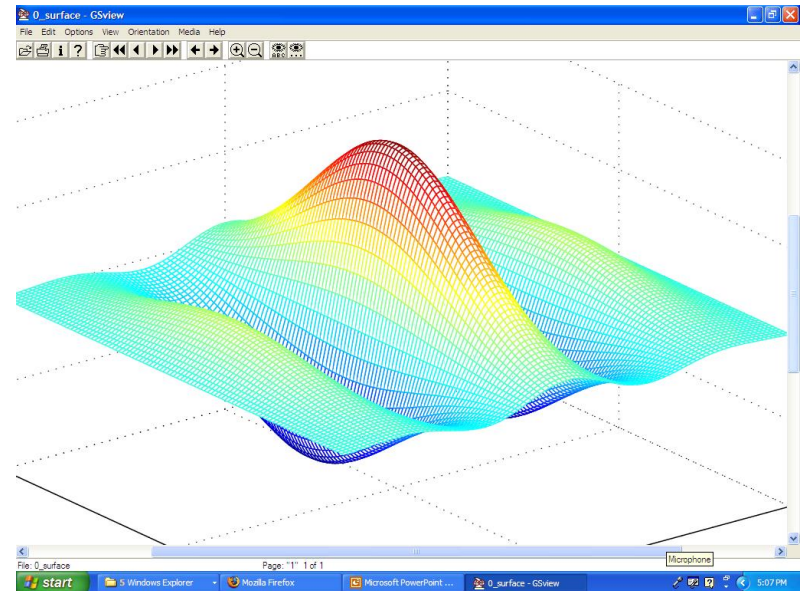
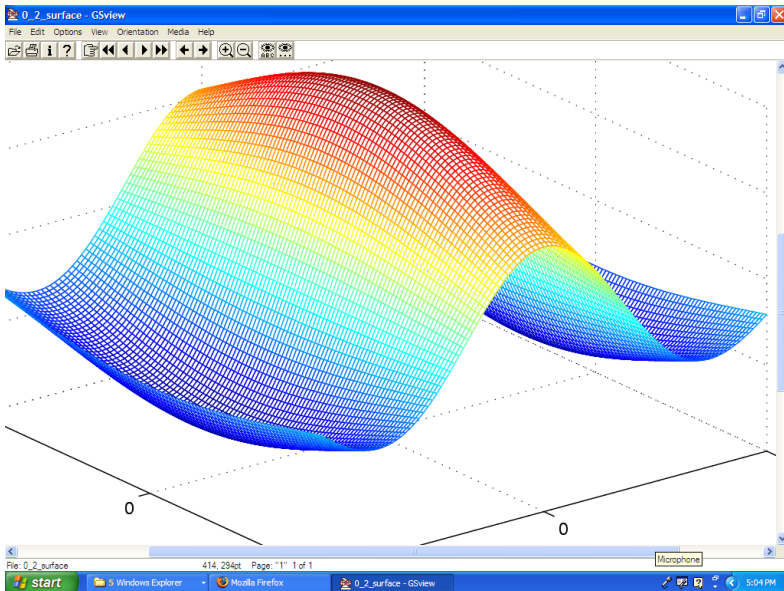
Table 7.2: Laws texture energy measures for major regions of the images of Figure 7.8.

Region	E5E5	S5S5	R5R5	E5L5	S5L5	R5L5	S5E5	R5E5	R5S5
Tiger	168.1	84.0	807.7	553.7	354.4	910.6	116.3	339.2	257.4
Water	68.5	36.9	366.8	218.7	149.3	459.4	49.6	159.1	117.3
Flags	258.1	113.0	787.7	1057.6	702.2	2056.3	182.4	611.5	350.8
Fence	189.5	80.7	624.3	701.7	377.5	803.1	120.6	297.5	215.0
Grass	206.5	103.6	1031.7	625.2	428.3	1153.6	146.0	427.5	323.6
Small flowers	114.9	48.6	289.1	402.6	241.3	484.3	73.6	158.2	109.3
Big flowers	76.7	28.8	177.1	301.5	158.4	270.0	45.6	89.7	62.9
Borders	15.3	6.4	64.4	92.3	36.3	74.5	9.3	26.1	19.5

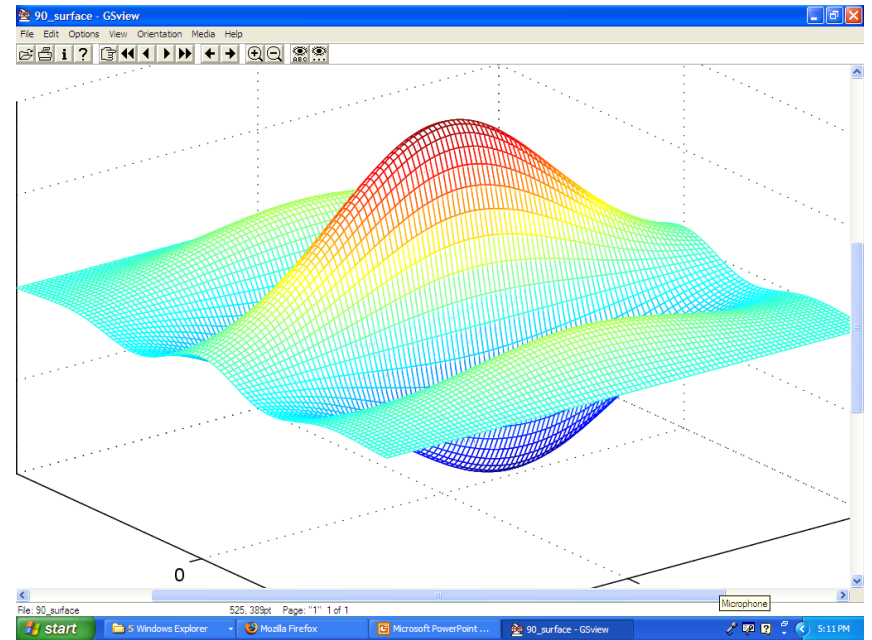
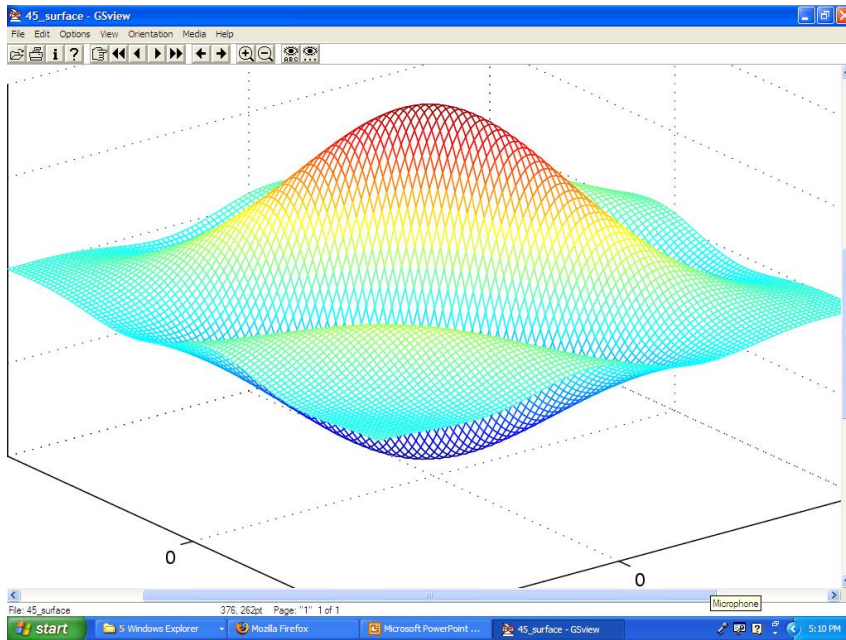


# Gabor Filters

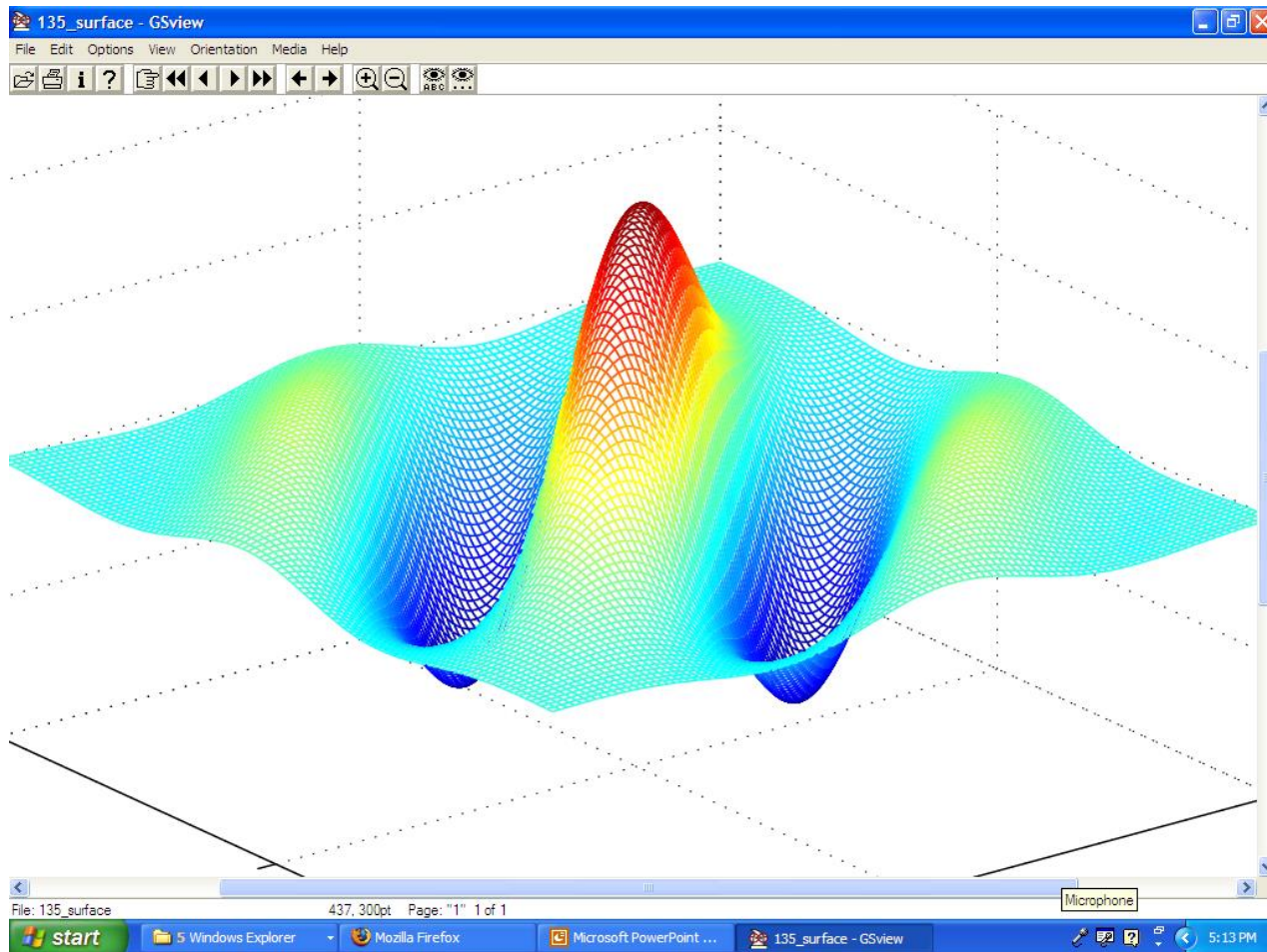
- Similar approach to Laws
- Wavelets at different frequencies and different orientations



# Gabor Filters



# Gabor Filters





# Segmentation with Color and Gabor-Filter Texture (Smeulders)





# Use of Texture

- Texture is important, but usually not as discriminative alone as color.
- The use of color and texture together can work well for both recognition and segmentation.

# Region Segmentation by Color



# Main Methods of Region Segmentation

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

# Clustering

- There are  $K$  clusters  $C_1, \dots, C_K$  with means  $m_1, \dots, m_K$ .
- The **least-squares error** is defined as

$$D = \sum_{k=1}^K \sum_{x_i \in C_k} \|x_i - m_k\|^2.$$

- 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?

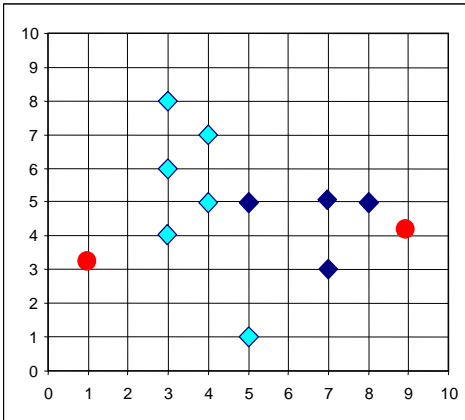
# K-Means Clustering

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  $m_1(1), \dots, m_K(1)$ .
3. For each vector  $x_i$  compute  $D(x_i, m_k(ic))$ ,  $k=1, \dots, K$  and assign  $x_i$  to the cluster  $C_j$  with nearest mean.
4. Increment ic by 1, update the means to get  $m_1(ic), \dots, m_K(ic)$ .
5. Repeat steps 3 and 4 until  $C_k(ic) = C_k(ic+1)$  for all k.

# Simple Example

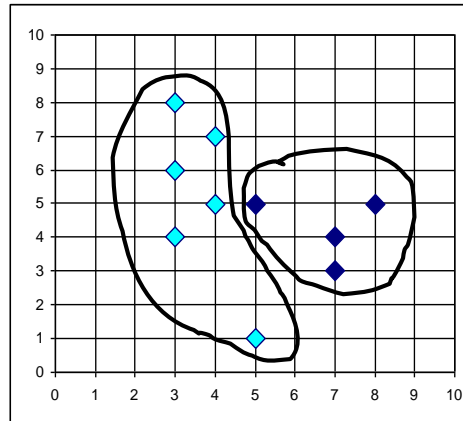
INIT.



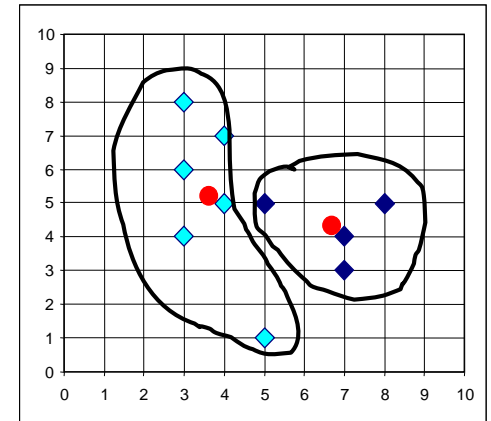
$K=2$

Arbitrarily choose  $K$  objects as initial cluster center

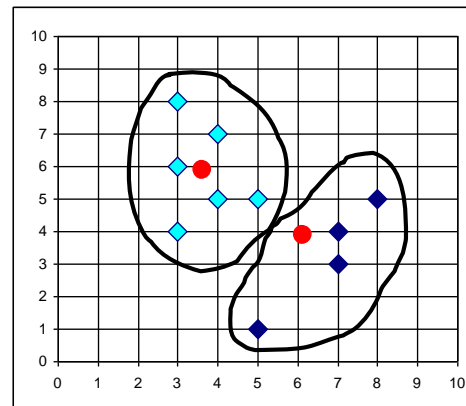
Assign each object to most similar center



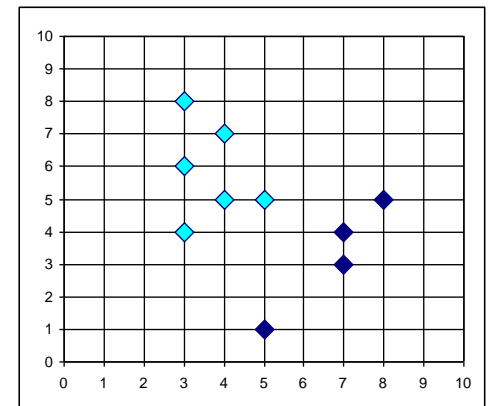
Update the cluster means



reassign



Update the cluster means



# Space for K-Means

- The example was in some arbitrary 2D space
- We don't want to cluster in that space.
- We will be clustering in **color space** (ie. RGB)
- K-means can be used to cluster in **any n-dimensional space**.

# K-Means Example 1

1. Select an image:  2. Select a processor:  3. Click

Options:  
Init Method

  
640\*480 (590,68): RGB(158,206,229)



Process done !



# K-Means Example 2

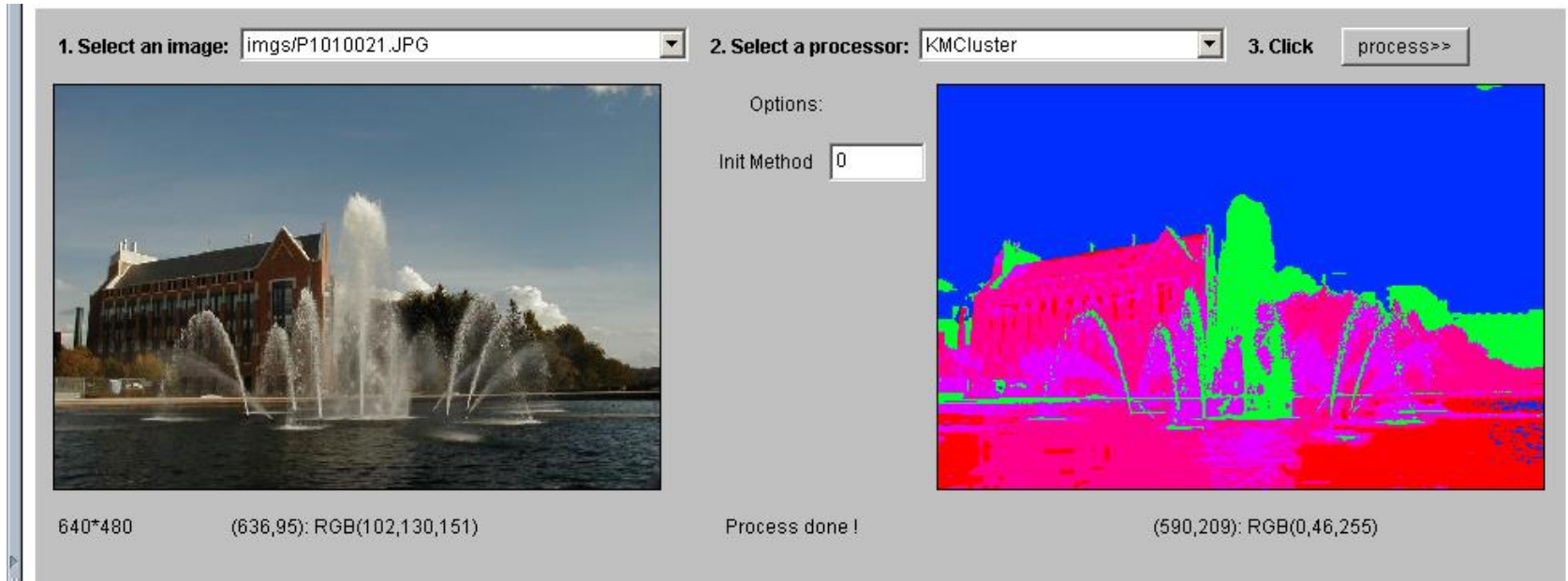
1. Select an image:  2. Select a processor:  3. Click

Options:  
Init Method

640\*480 (636,95): RGB(102,130,151)

Process done !

(590,209): RGB(0,46,255)




# K-Means Example 3

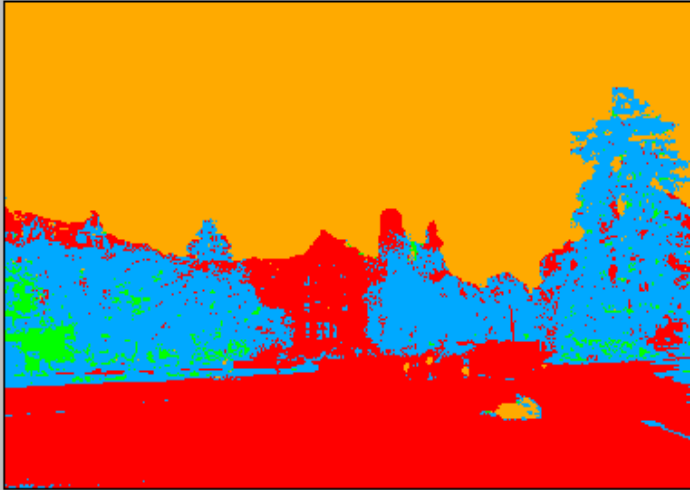
1. Select an image:  2. Select a processor:  3. Click

Options:  
Init Method

Process done !



640\*480 (607,118): RGB(20,22,1)

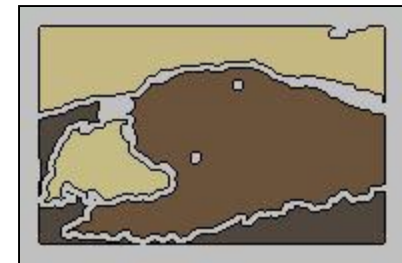
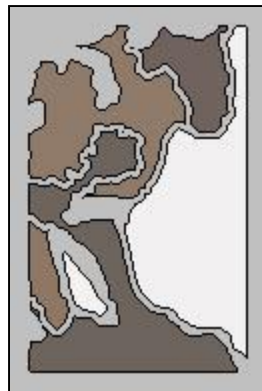
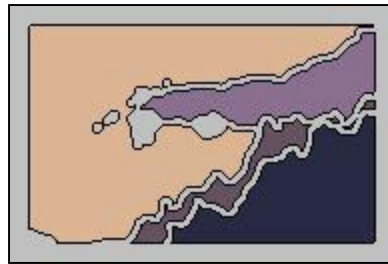


(228,26): RGB(255,170,0)

# K-means Variants

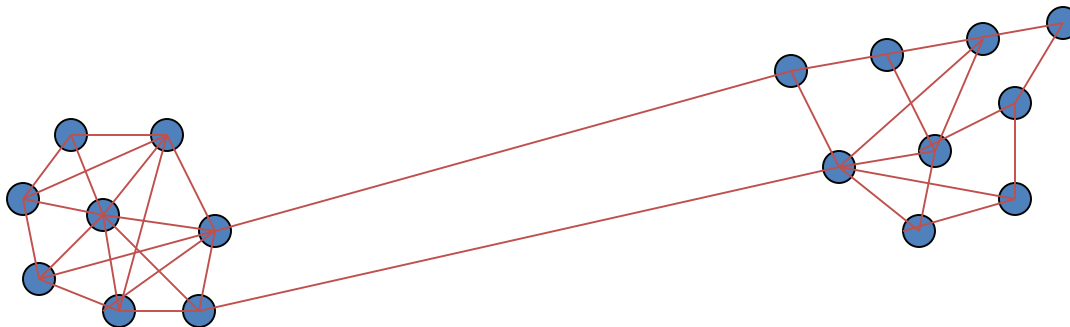
- Different ways to initialize the means
- Different stopping criteria
- Dynamic methods for determining the right number of clusters ( $K$ ) for a given image
- The EM Algorithm: a probabilistic formulation of K-means

# Blobworld: Sample Results using color, texture, and EM



# Graph-Partitioning Clustering

- 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.

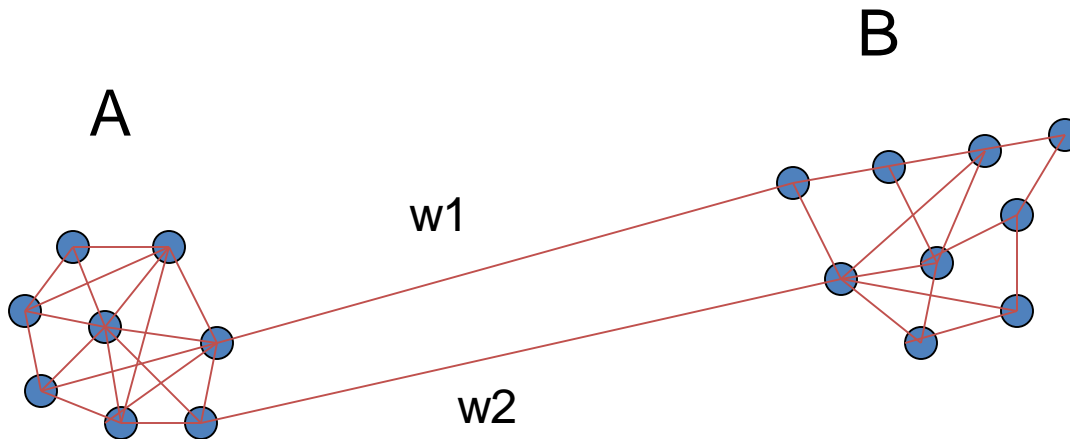


# Minimal Cuts

- 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  $\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$ .
- One way to segment  $G$  is to find the minimal cut.

# Cut(A,B)

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



# Normalized Cut

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

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

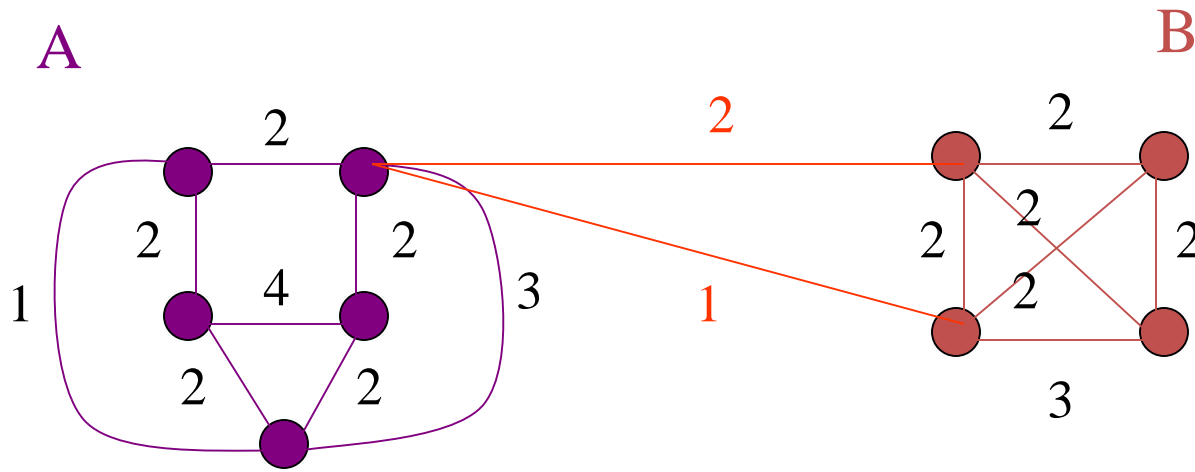
normalized  
cut

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

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



# Example Normalized Cut



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

# 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
  - $N \times N$  matrix  $D$ :  $D$  is a diagonal matrix with  $d$  on its diagonal
  - $N \times N$  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  $\lambda$

- Use the eigenvector  $y$  with second smallest eigenvalue to bipartition the graph ( $y \Rightarrow x \Rightarrow A$ )
- If further subdivision is merited, repeat recursively

# How to define the weights

Jianbo 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

