Lecture 9: Texture Analysis

Introduction

- What does texture mean? Formal approach or precise definition of texture does not exist!
- Texture discrimination techniques are for the part ad hoc.

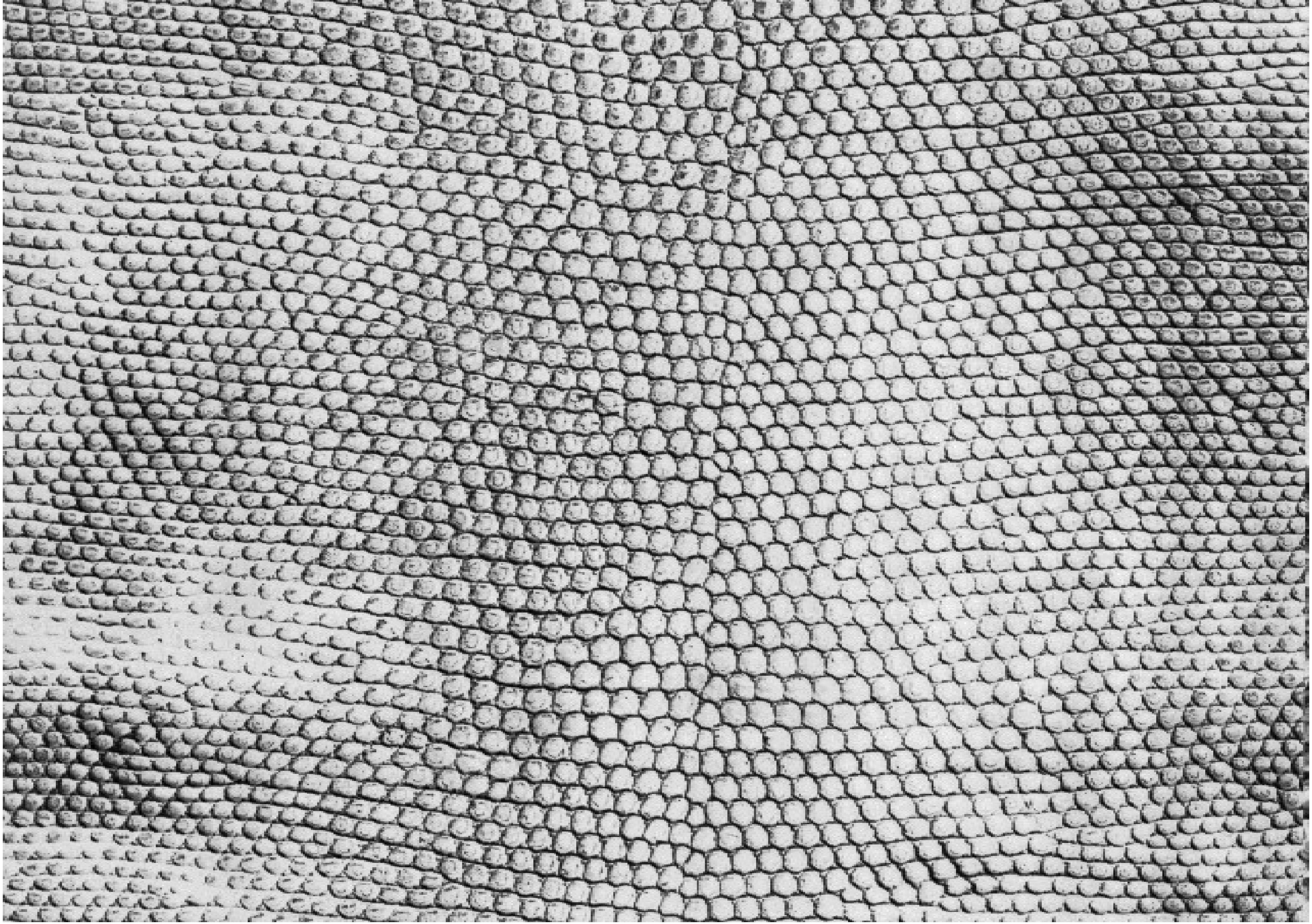
Definition of Texture

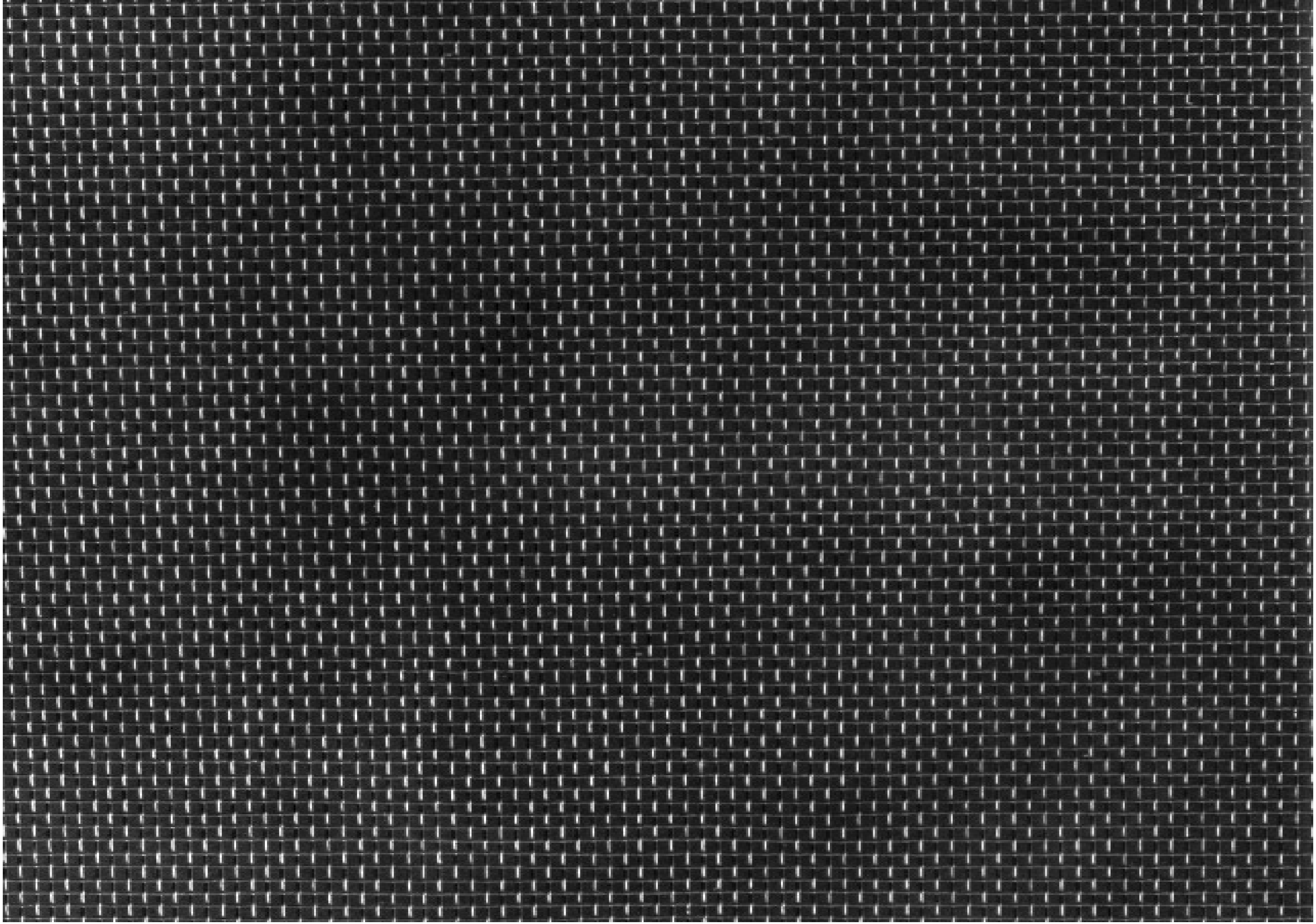
- Non-local property, characteristic of region larger than its size
- Repeating patterns of local variations in image intensity which are too fine to be distinguished as separated objects at the observed resolution

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

Definition of Texture (cont.)

- For humans, texture is the abstraction of certain statistical homogeneities from a portion of the visual field that contains a quantity of information grossly in excess of the observer’s perceptual capacity





Texture Analysis Issues

- Pattern recognition: given texture region, determine the class the region belongs to

- Generative model: given textured region, determine a description or model for it

- Texture segmentation: given image with many textured areas, determine boundaries

Image Texture Analysis

- Give a generative model and the values of its parameters, one can synthesize homogeneous image texture samples associated with the model and the given value of its parameters.

Image Texture Analysis (cont.)

- Verification: verify given image textures sample consistent with model

- Estimation: estimate values of model parameters based on observed sample examples of model-based techniques

Model-Based Techniques

- Autoregressive, moving-average, time-series models (extended to 2D)
- Markov random fields
- Mosaic models

Texture Features

- Fineness
- Coarseness
- Contrast
- Directionality
- Roughness
- Regularity
- Smoothness
- Granulation

Texture Features (cont.)

- Randomness
- Lineation
- Mottled
- Irregular
- Hummocky

Texture Analysis

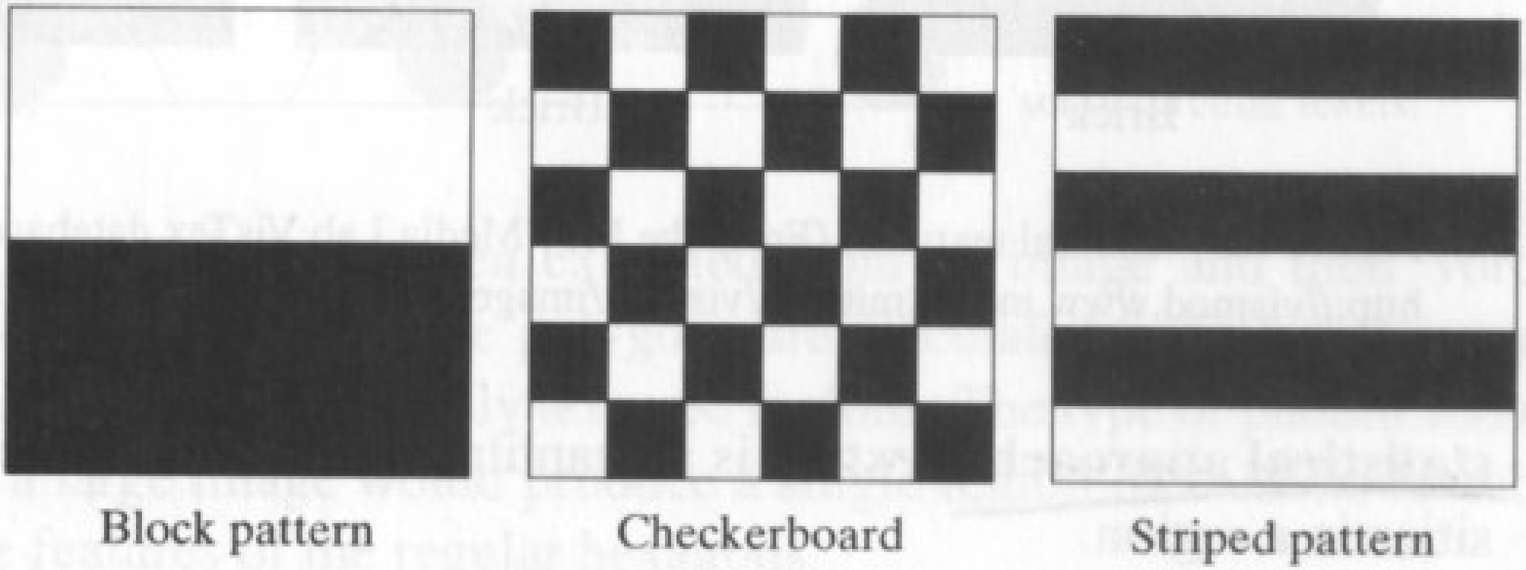


Figure 7.2 Three different textures with the same distribution of black and white.

- Structural approach
- Statistical approach

Structural approach

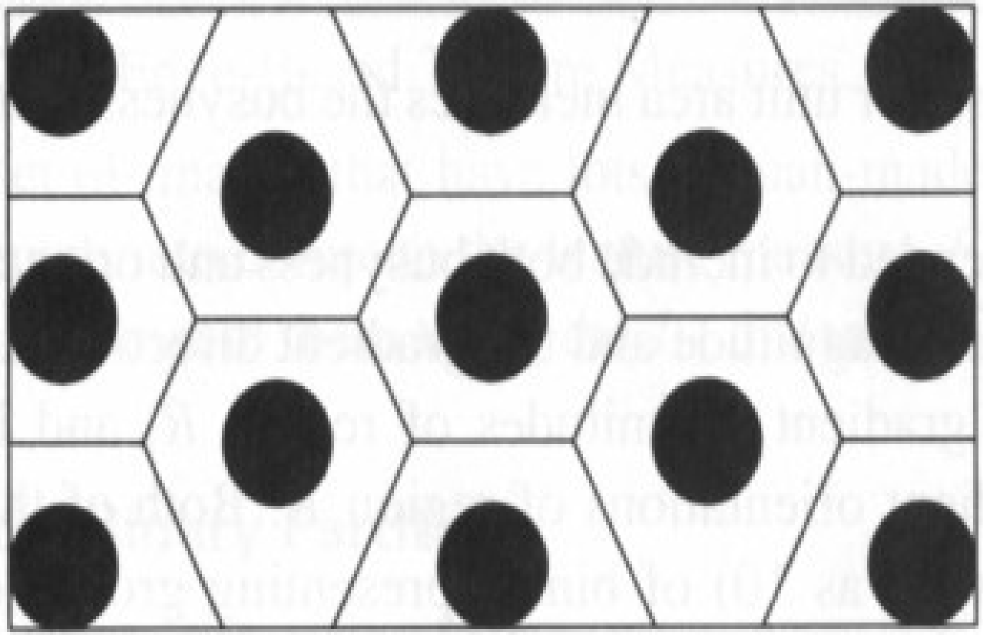
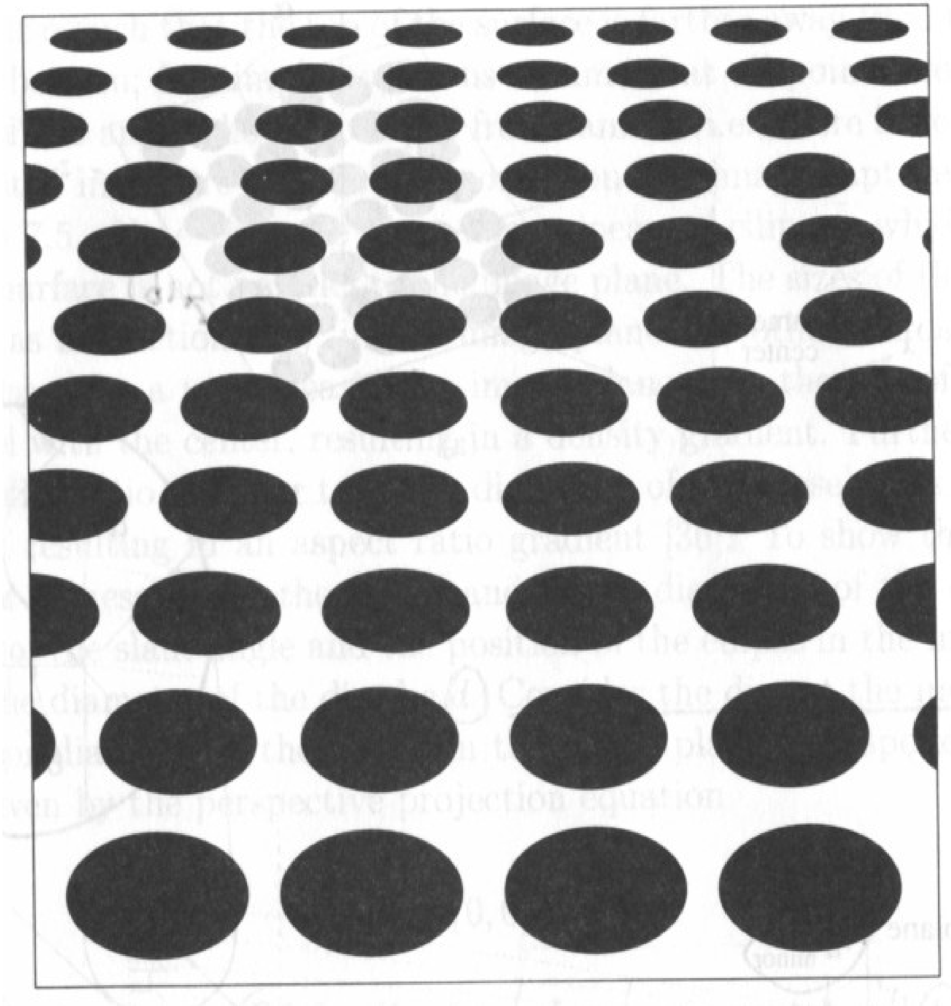


Figure 7.4 The Voronoi tesselation of a set of circular texels.

Shape from texture

1. Size
2. Shape
3. Density



Statistical approach

1. Edge density and direction

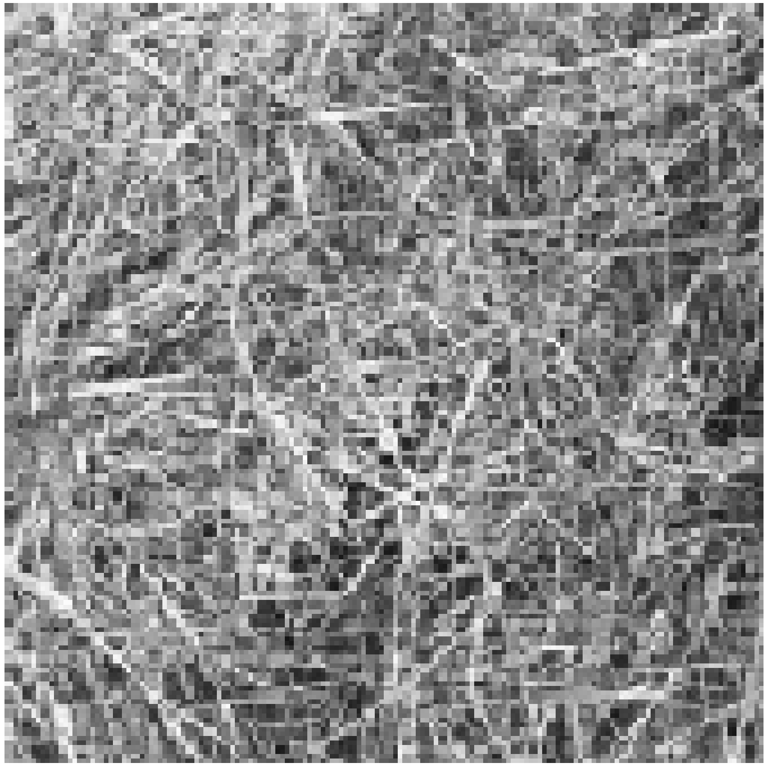
2. Local binary partition

3. Co-occurrence matrices and features

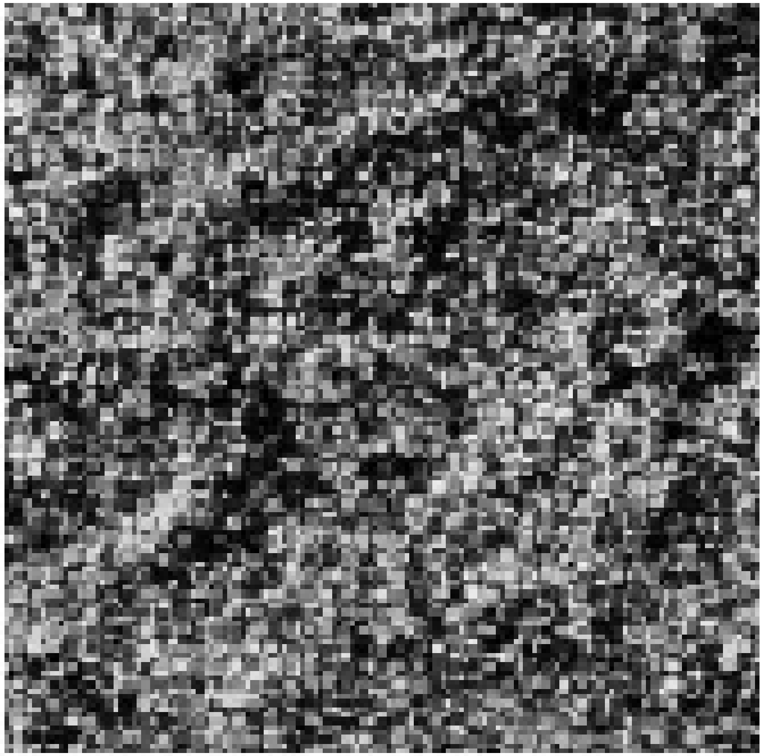
4. Laws texture energy measures

5. Autocorrelation

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.

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

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

where N is the size of the unit area

2. edge magnitude and direction histograms

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

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 partition

- For each pixel *p* in the image, the eight neighbors are examined to see if their intensity is greater than that of *p.*

- The results from the eight neighbors are used to construct an eight-digit binary number $b_1b_2b_3b_4b_5b_6b_7b_8$

- $b_i=0$ if the intensity of the *ith* neighbor is less than or equal to that of *p*
- $b_i=1$ otherwise

- A histogram of these numbers is used to represent the texture of the 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.

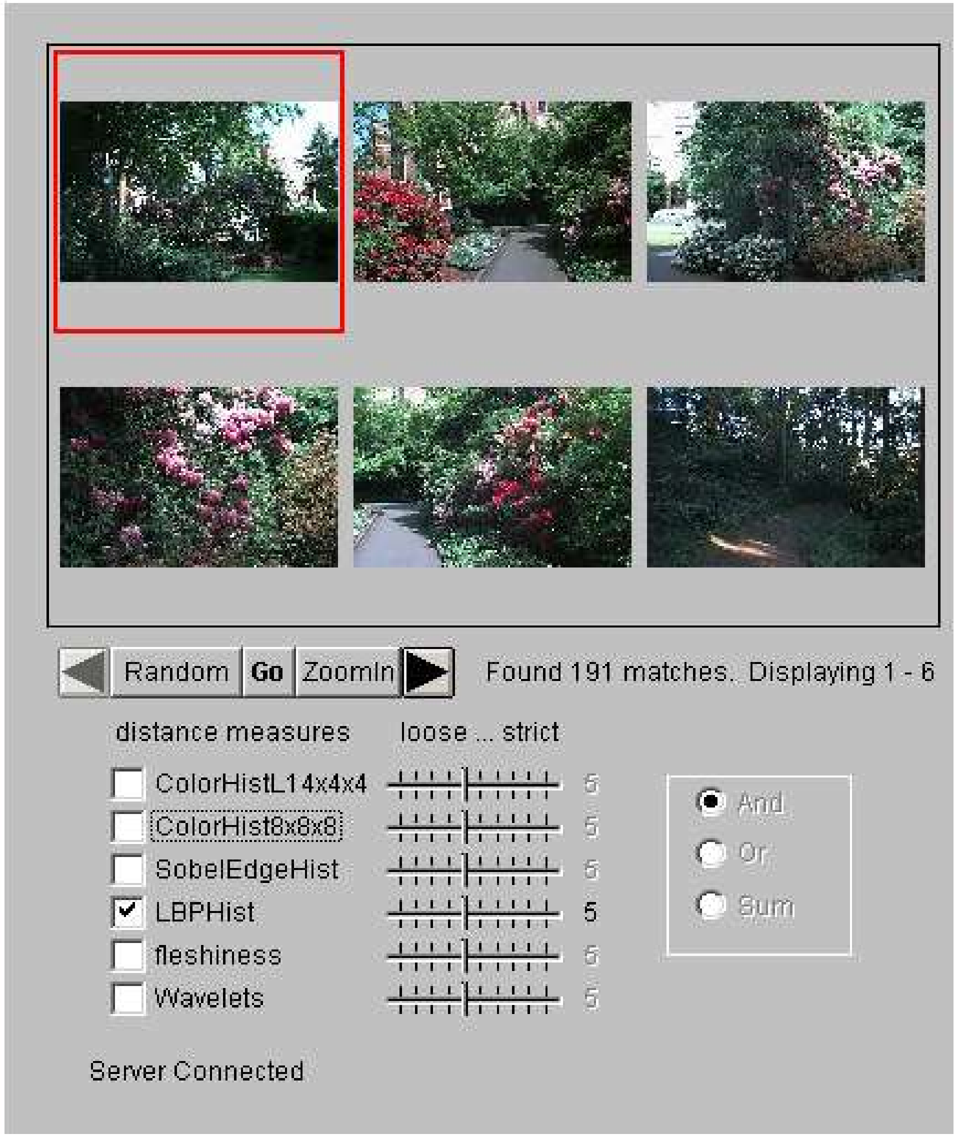
	1	2	3	
	100	101	103	
8	40	50	80	4
	50	60	90	5
	7	6		

→ 1 1 1 1 1 1 0 0

Example

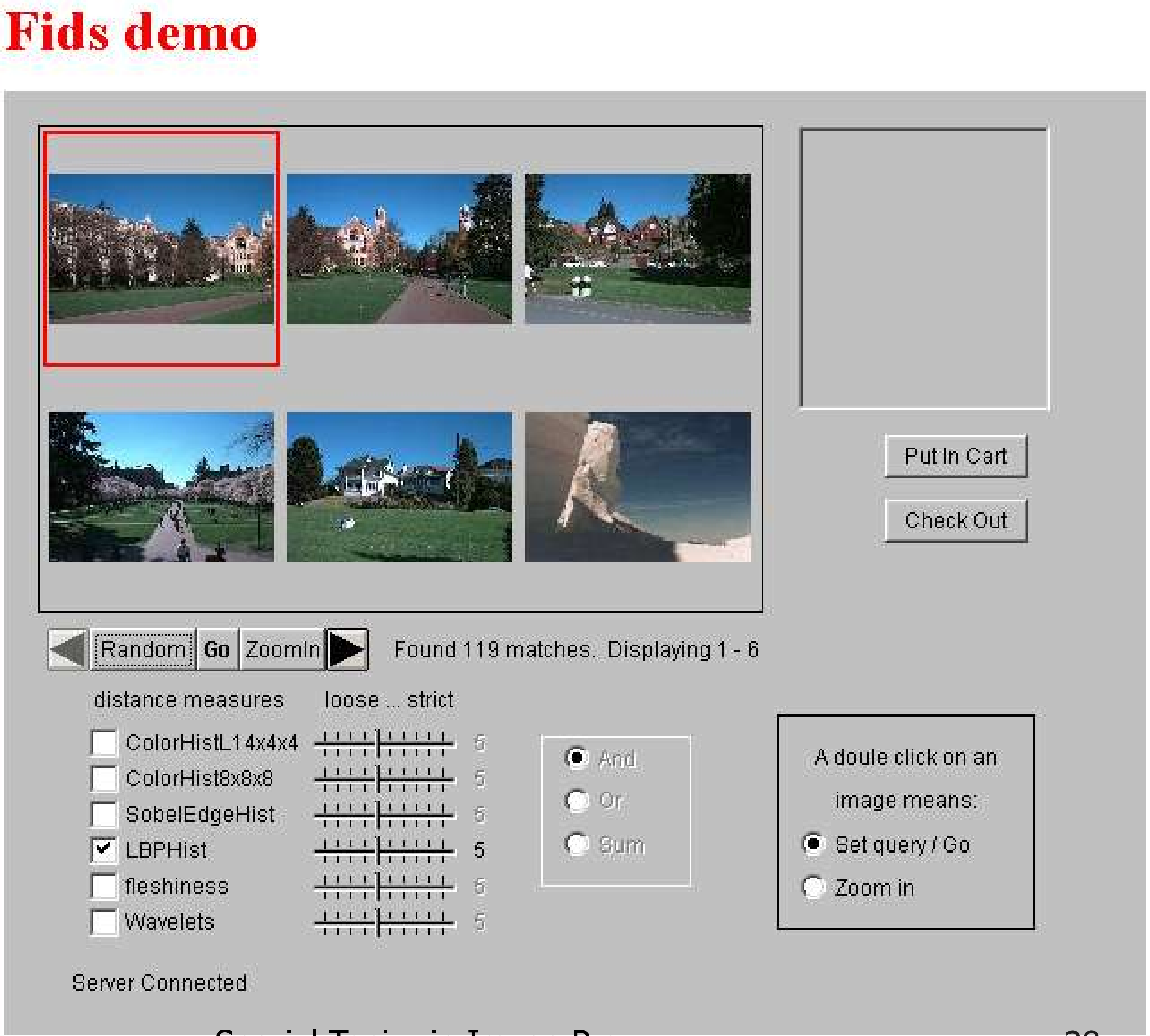
Fids (Flexible Image Database System) is retrieving images similar to the query image using LBP texture as the texture measure and comparing their LBP histograms

Fids demo



Example

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



Co-occurrence Matrix Features

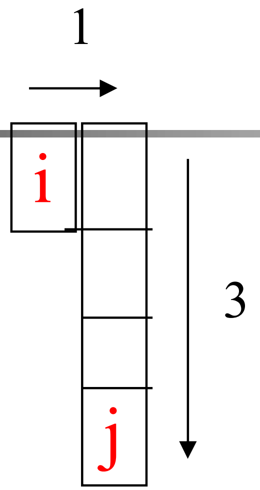
A co-occurrence matrix is a 2D array C in which

- 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

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{array}$$

gray-tone image



$\boxed{d = (3,1)}$

$$\begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 3 \\ 1 & 2 & 0 & 2 \\ 2 & 0 & 0 & 1 \\ \hline \end{array} \quad C_d$$

co-occurrence matrix

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 matrices

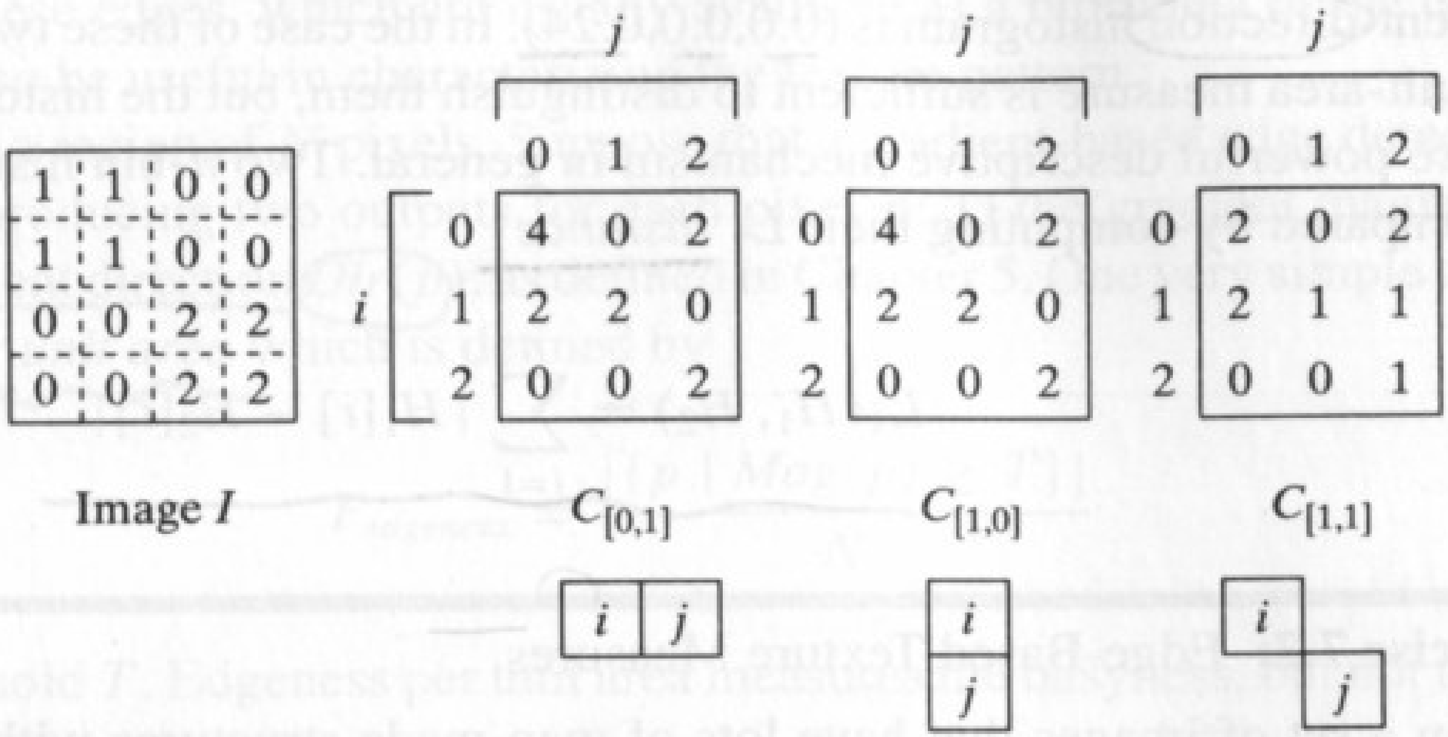


Figure 7.6 Three different co-occurrence matrices for a gray-tone image.

$$C_d[i,j] = \left|\left\{[r,c] \mid I[r,c] = i \text{ and } I[r+dr, c+dc] = j\right\}\right|$$

Co-occurrence Features

What do these measure?

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

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

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

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

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

where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column sums.

Energy measures uniformity of the normalized matrix.

But how do you choose d?

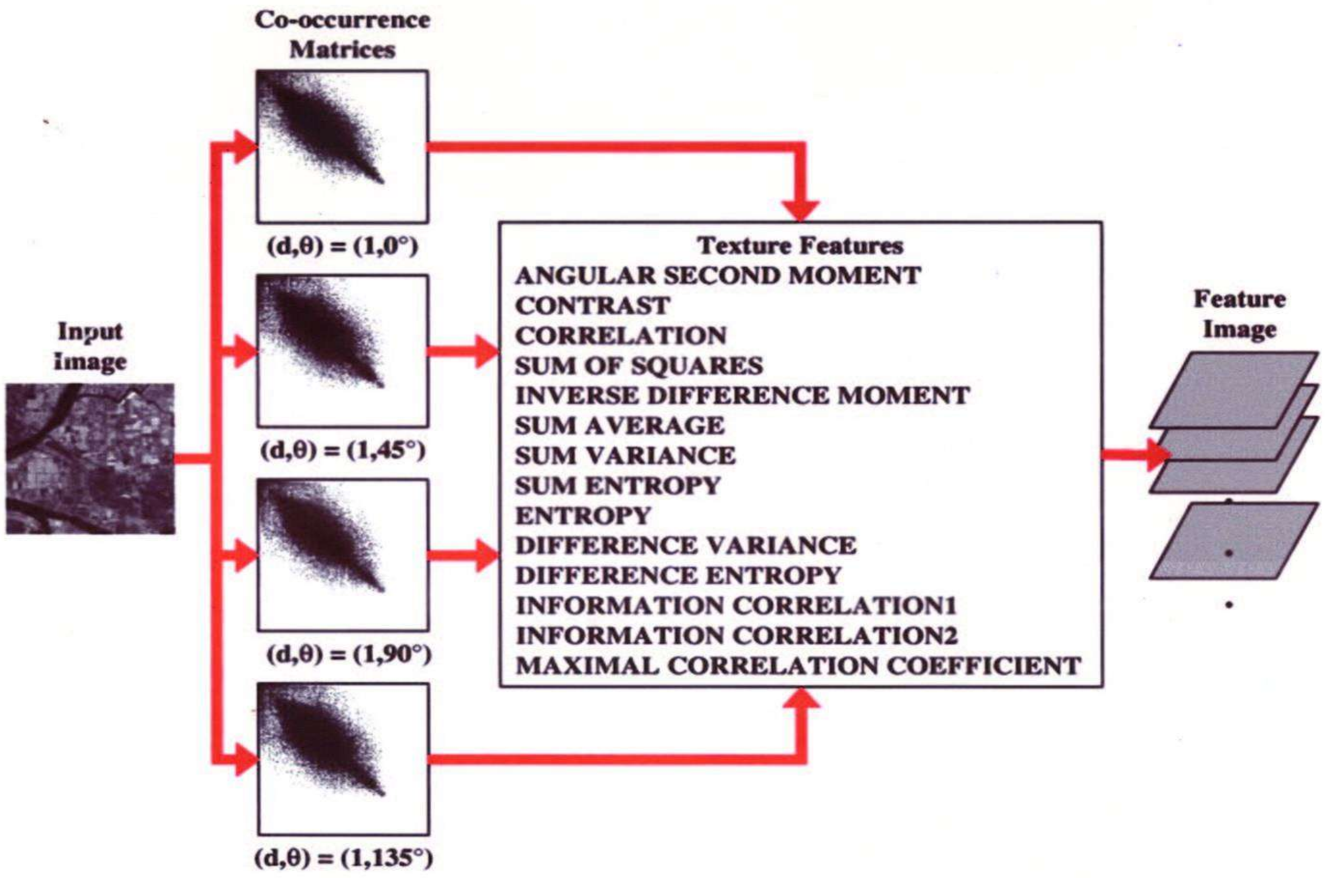
- This is actually a critical question with **all** the statistical texture methods.

- Are the “texels” tiny, medium, large, all three …?

- Not really a solved problem.

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

Example



Co-occurrence Matrices
Input Image
(d,θ) = (1,0°)
(d,θ) = (1,45°)
(d,θ) = (1,90°)
(d,θ) = (1,135°)
Texture Features
ANGULAR SECOND MOMENT
CONTRAST
CORRELATION
SUM OF SQUARES
INVERSE DIFFERENCE MOMENT
SUM AVERAGE
SUM VARIANCE
SUM ENTROPY
ENTROPY
DIFFERENCE VARIANCE
DIFFERENCE ENTROPY
INFORMATION CORRELATION1
INFORMATION CORRELATION2
MAXIMAL CORRELATION COEFFICIENT
Feature Image

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.

Laws' texture energy measures

1. Remove the effects of illumination by moving a small window around the image, and subtracting the local average from each pixel.

2. Nine different 5x5 masks are applied to the preprocessed image, producing 9 images.

3. Smooth the images using absolute mean filter.

$$E_k[r,c] = \sum_{j=c-7}^{c+7} \sum_{i=r-7}^{r+7} \left| F_k[i,j] \right|$$

4. The output is a single image with a vector of nine texture attributes at each pixel.

Laws' nine 5x5 masks

L5 (Level)	=	[1	4	6	4	1]
E5 (Edge)	=	[-1	-2	0	2	1]
S5 (Spot)	=	[-1	0	2	0	-1]
R5 (Ripple)	=	[1	-4	6	-4	1]

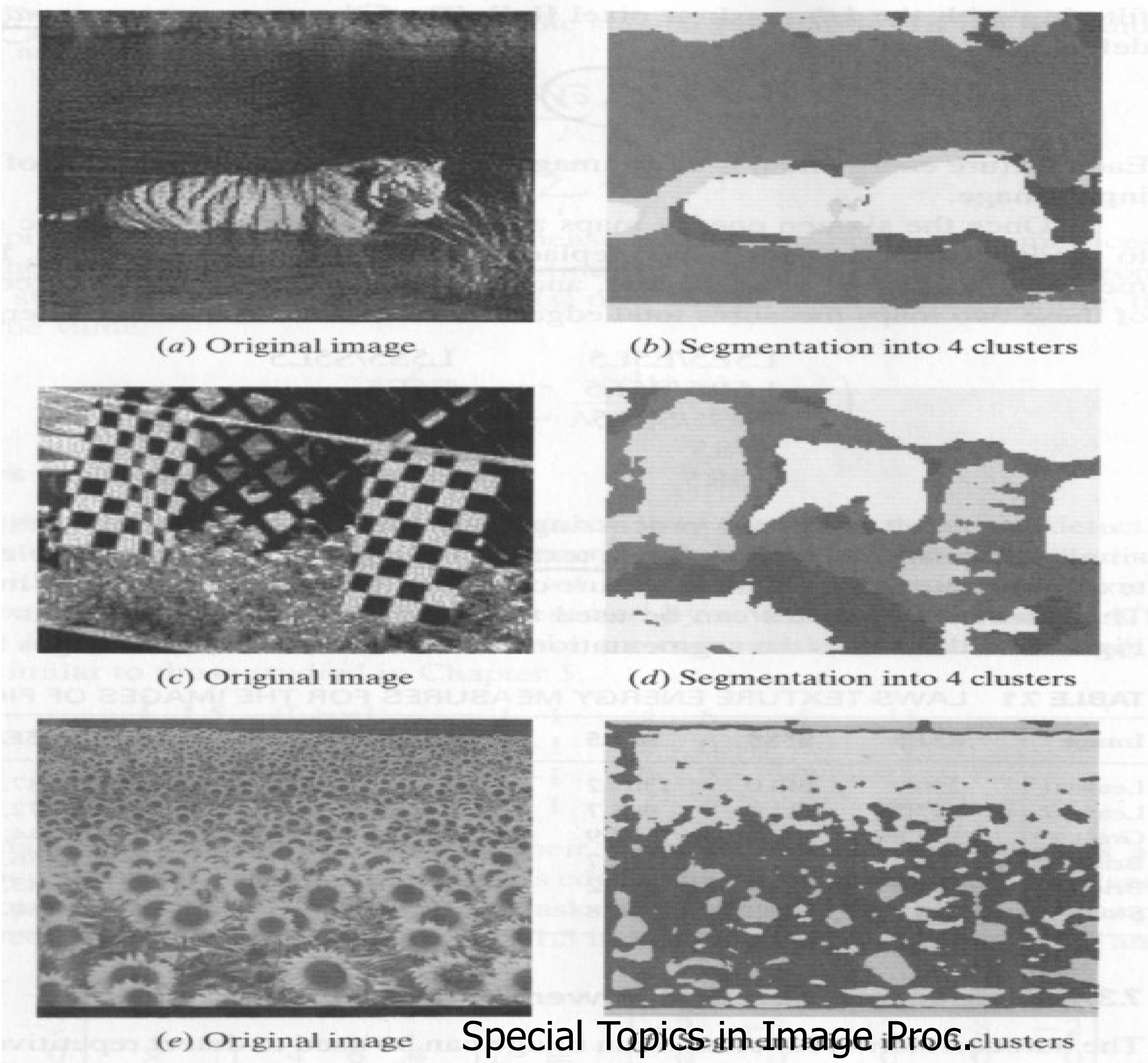
1. L5E5/E5L5
2. L5S5/S5L5
3. L5R5/R5L5
4. E5S5/S5E5
5. E5R5/R5E5
6. S5R5/R5S5
7. E5E5
8. S5S5
9. R5R5

Autocorrelation

- To describe the fineness/coarseness of the texture.

$$\rho(dr,dc)=\frac{\sum_{r=0}^{N}\sum_{c=0}^{N} I[r,c]I[r+dr,c+dc]}{\sum_{r=0}^{N}\sum_{c=0}^{N} I^2[r,c]}=\frac{I[r,c]\circ I_d[r,c]}{I[r,c]\circ I[r,c]}$$

Texture Segmentation



Laws' texture masks (1)

L5	(Level)	=	[1 4 6 4 1]
E5	(Edge)	=	[-1 -2 0 2 1]
S5	(Spot)	=	[-1 0 2 0 -1]
R5	(Ripple)	=	[1 -4 6 -4 1]

- (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

Laws' texture masks (2)

Creation of 2D Masks

- **1D Masks are "multiplied" to construct 2D masks:**

  mask E5L5 is the "product" of E5 and L5 –

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

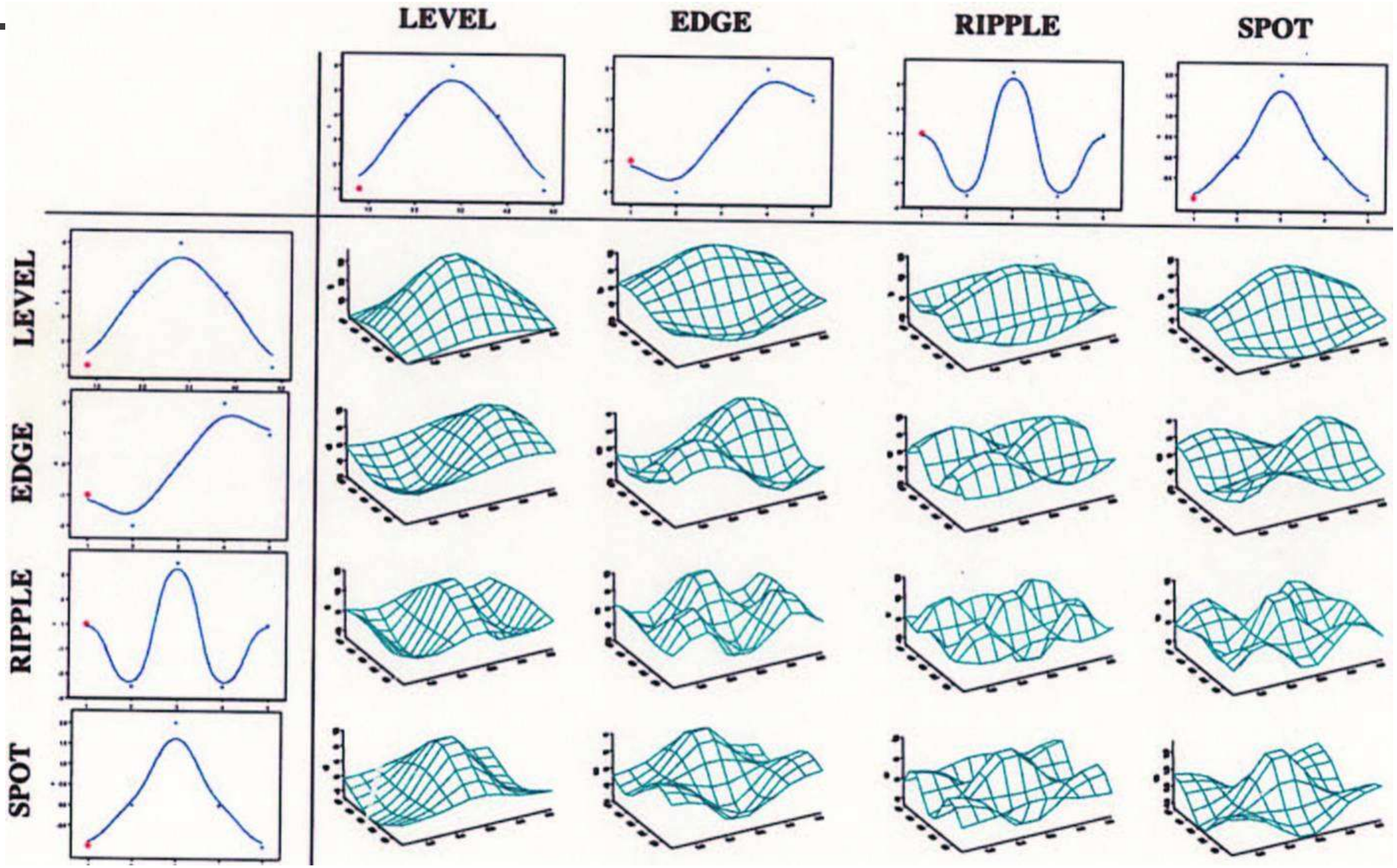
9D feature vector for pixel

- 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

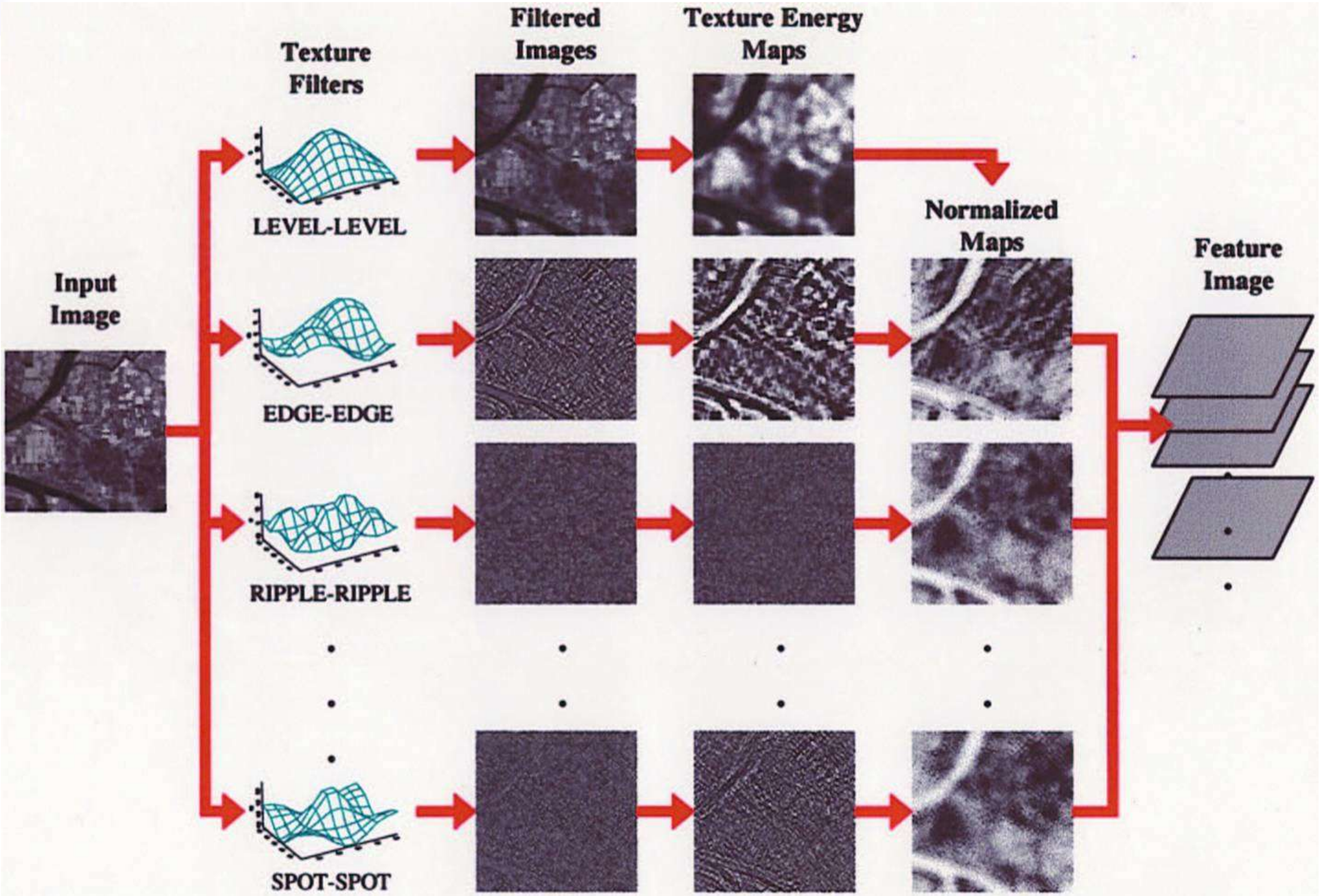
  $E_k[r,c] = \Sigma \; |F_k[i,j]|$
- Replace each distinct pair with its average map:
- 9 features (9 filtered images) defined as follows:

L5E5/E5L5	L5S5/S5L5
L5R5/R5L5	E5E5
E5S5/S5E5	E5R5/R5E5
S5S5	S5R5/R5S5
R5R5	

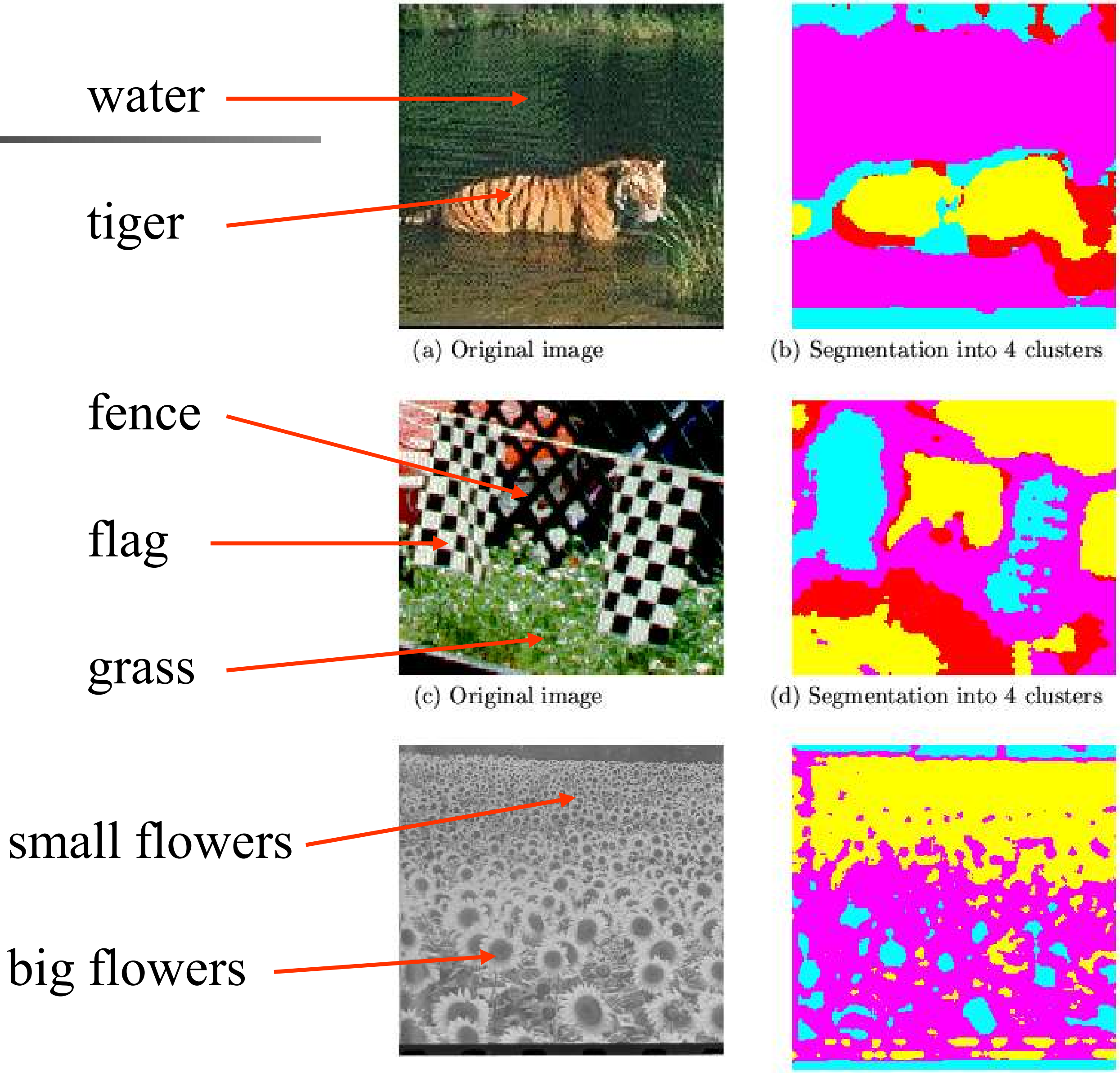
Laws' Filters



Laws' Process



Example: Using Laws' Features to Cluster



(a) Original image (b) Segmentation into 4 clusters

(c) Original image (d) Segmentation into 4 clusters

(e) Original image (f) Segmentation into 3 clusters

Is there a neighborhood size problem with Laws?

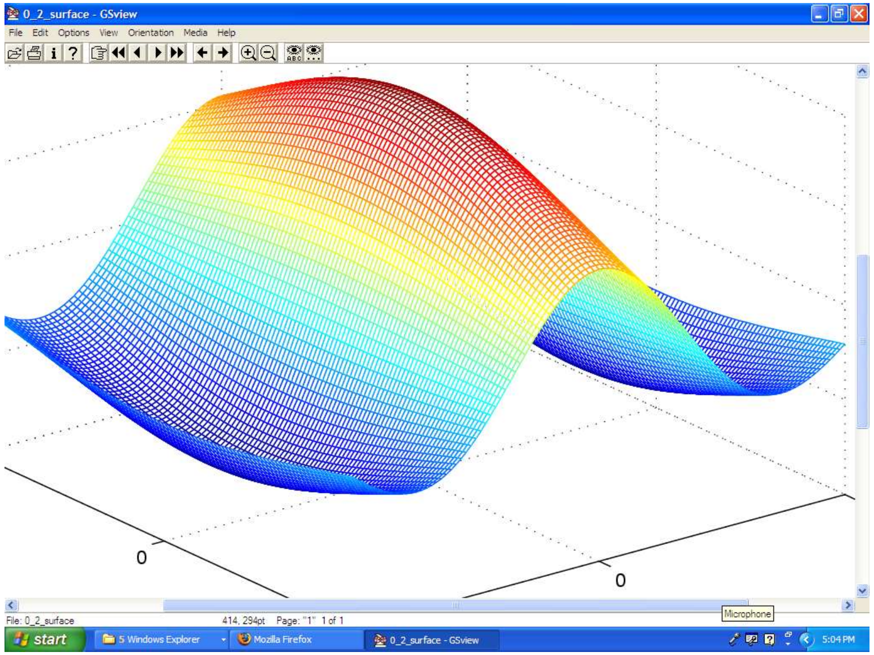
Features from sample images

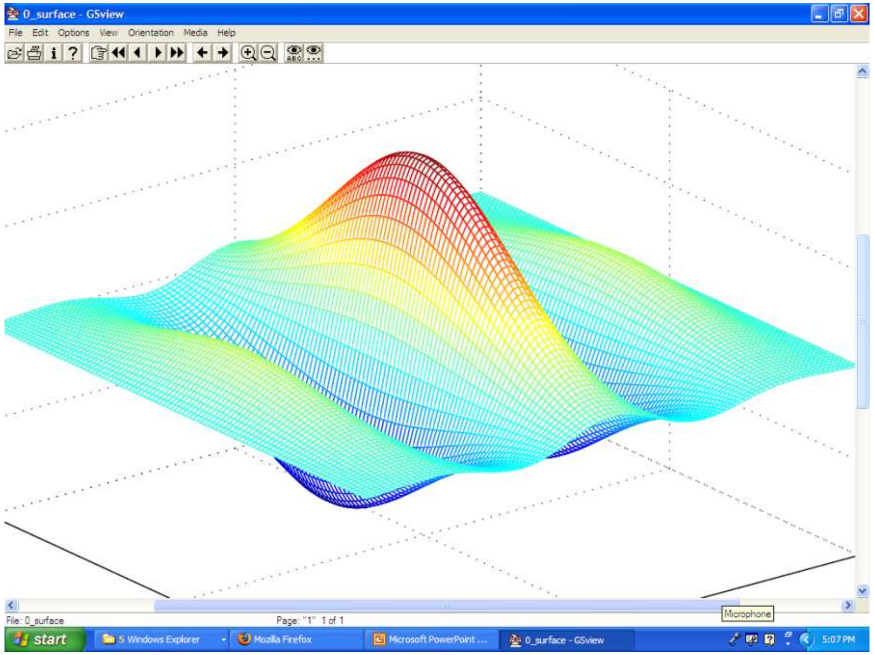
Table 7.2: Laws texture energy measures for major regions of the images of Figure 7.8.

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

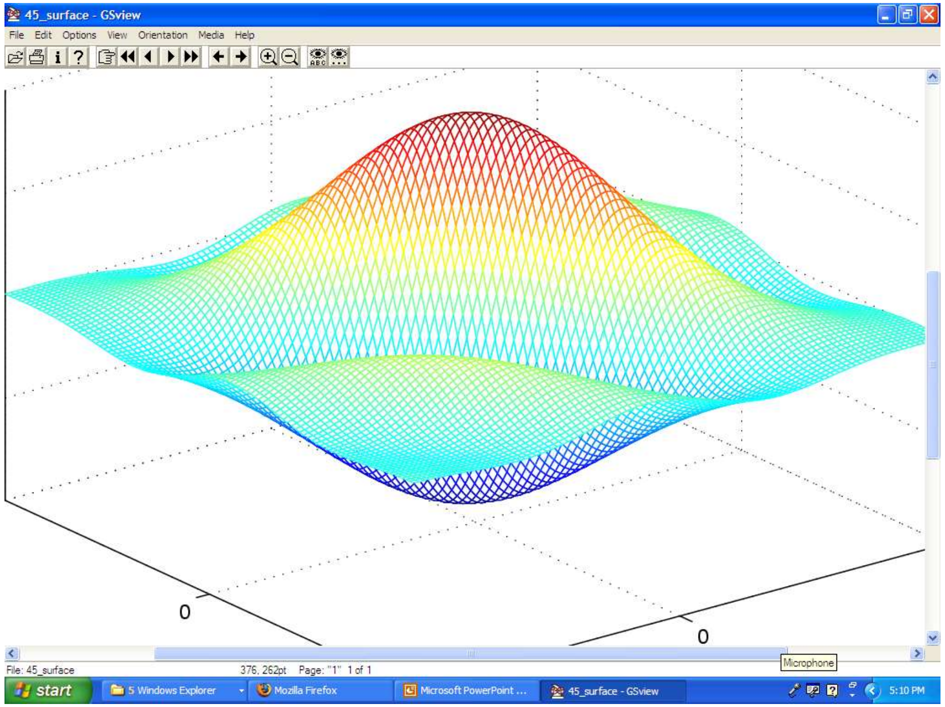
Gabor Filters

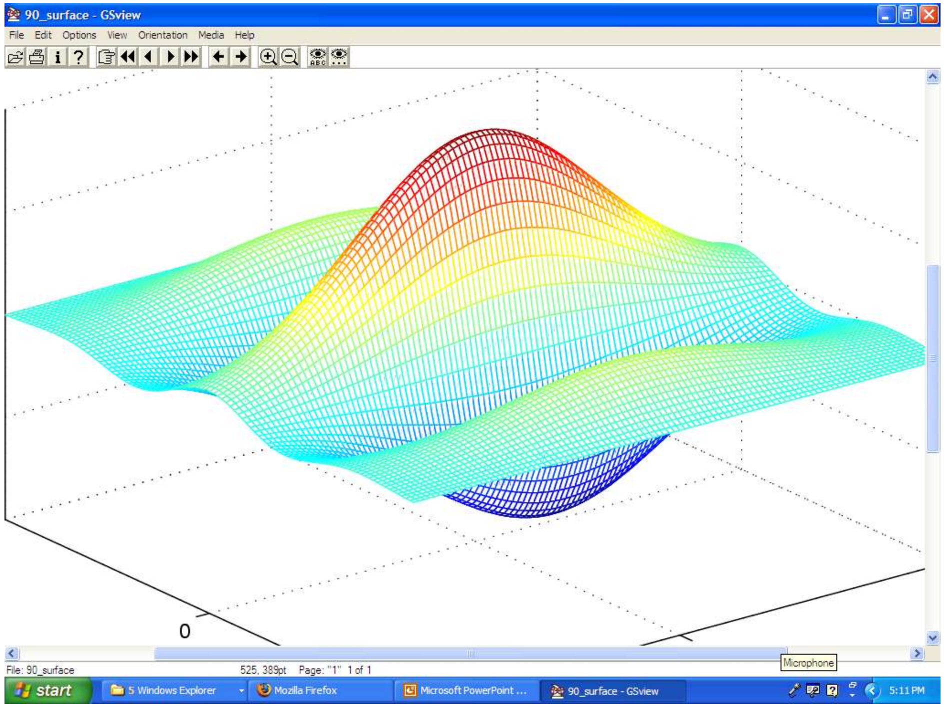
- Similar approach to Laws
- Wavelets at different frequencies and different orientations



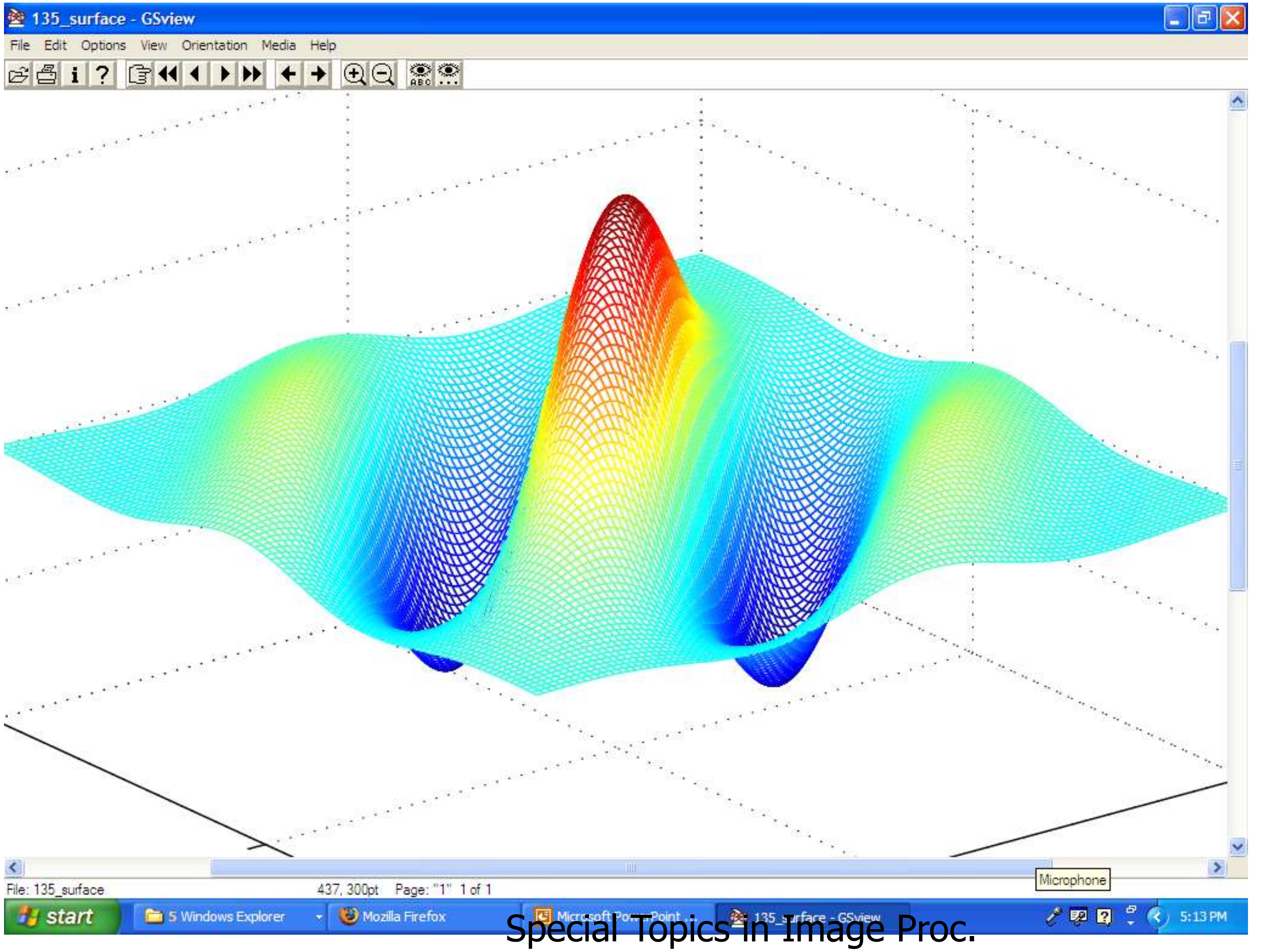


Gabor Filters

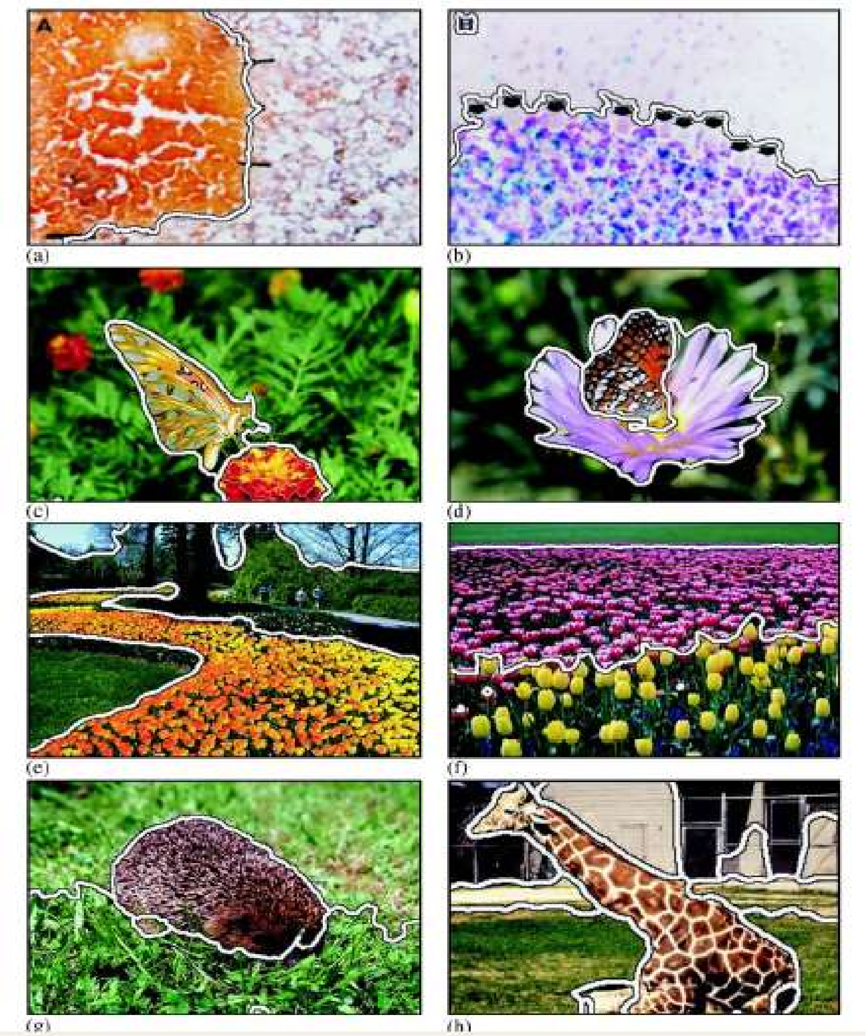




Gabor Filters



Segmentation with Color and Gabor-Filter Texture (Smeulders)



A classical texture measure:

Autocorrelation function

- Autocorrelation function can detect repetitive patterns of texels
- Also defines fineness/coarseness of the texture
- Compare the dot product (energy) of non shifted image with a shifted image

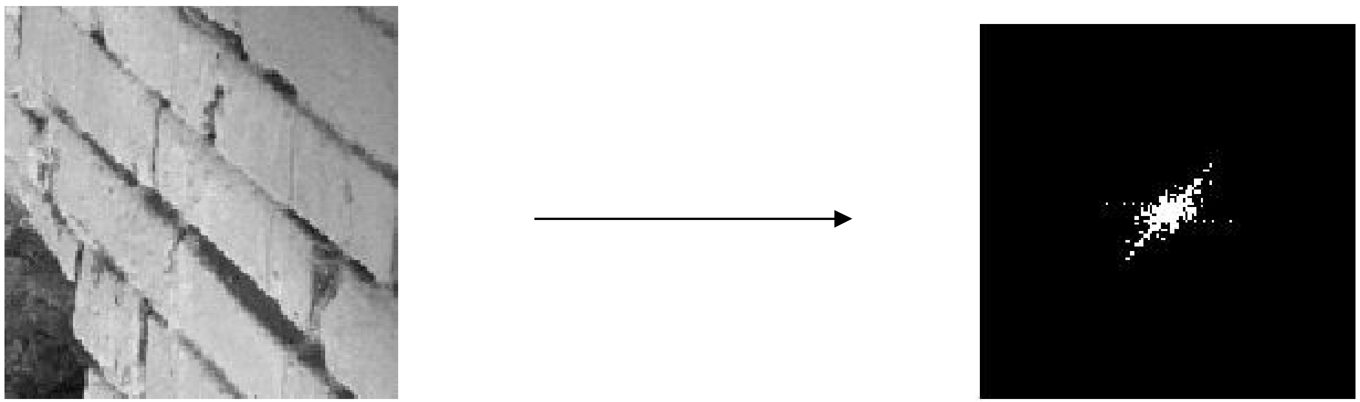
$$\rho(dr, dc) = \frac{\sum_{r=0}^{N} \sum_{c=0}^{N} I[r,c] I(r+dr, c+dc]}{\sum_{r=0}^{N} \sum_{c=0}^{N} I^2[r,c]} = \frac{I[r,c] \circ I_d[r,c]}{I[r,c] \circ I[r,c]}$$

Interpreting autocorrelation

- Coarse texture → function drops off slowly
- Fine texture → function drops off rapidly
- Can drop differently for r and c
- Regular textures → function will have peaks and valleys; peaks can repeat far away from [0, 0]
- Random textures → only peak at [0, 0]; breadth of peak gives the size of the texture

Fourier power spectrum

- High frequency power → fine texture
- Concentrated power → regularity
- Directionality → directional texture

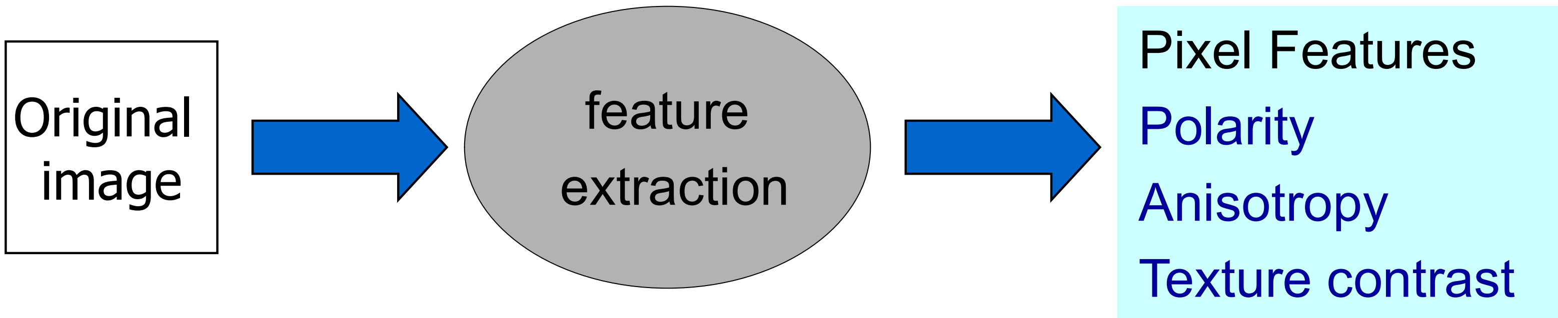


Blobworld Texture Features

- Choose the best scale instead of using fixed scale(s)
- Used successfully in color/texture segmentation in Berkeley's Blobworld project

Feature Extraction

- Input: image
- Output: pixel features
  - Color features
  - Texture features
  - Position features
- Algorithm: Select an appropriate scale for each pixel and extract features for that pixel at the selected scale



Texture Scale

- Texture is a local neighborhood property.
- Texture features computed at a wrong scale can lead to confusion.
- Texture features should be computed at a scale which is appropriate to the local structure being described.

The white rectangles show some sample texture scales from the image.

Scale Selection Terminology

- Gradient of the L* component (assuming that the image is in the L*a*b* color space) : $\nabla I \begin{bmatrix} I_x \\ I_y \end{bmatrix}$

- Symmetric Gaussian : $G_\sigma(x, y) = G_\sigma(x) * G_\sigma(y)$

- Second moment matrix: $M_\sigma(x, y) = G_\sigma(x, y) * (\nabla I)(\nabla I)^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

Notes: $G_\sigma(x, y)$ is a separable approximation to a Gaussian.

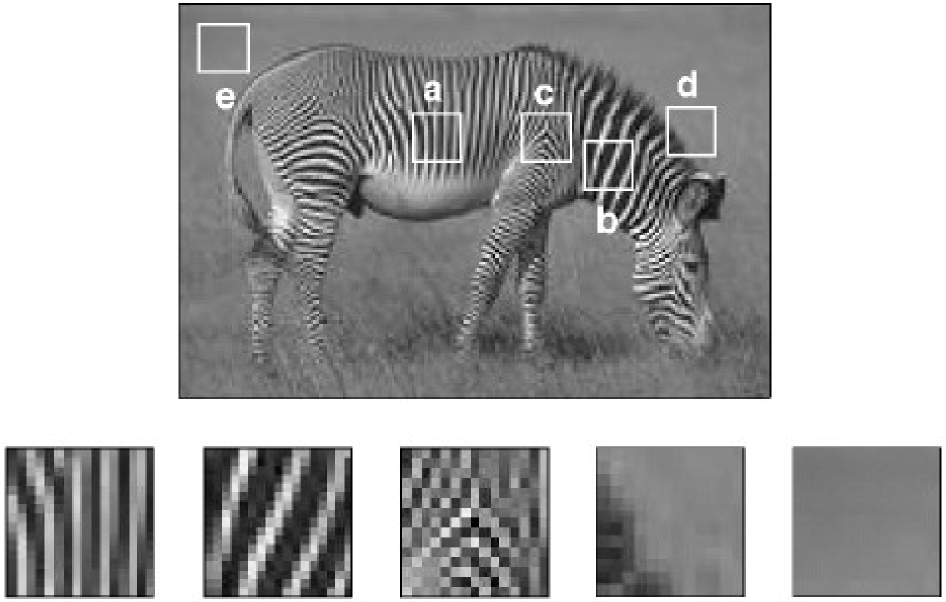
σ is the standard deviation of the Gaussian [0, .5, … 3.5].

σ controls the size of the window around each pixel [1 2 5 10 17 26 37 50].

$M_\sigma(x,y)$ is a 2X2 matrix and is computed at different scales defined by σ.

Scale Selection (continued)

- Make use of polarity (a measure of the extent to which the gradient vectors in a certain neighborhood all point in the same direction) to select the scale at which M_σ is computed



(a) (b) (c) (d) (e)

Edge: polarity is close to 1 for all scales σ
Texture: polarity varies with σ
Uniform: polarity takes on arbitrary values

Scale Selection (continued)

polarity p_σ

$$p_\sigma = \frac{|E_+ - E_-|}{E_+ + E_-}$$

$$E_+ = \sum_{x,y} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_+$$

$$E_- = \sum_{x,y} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_-$$

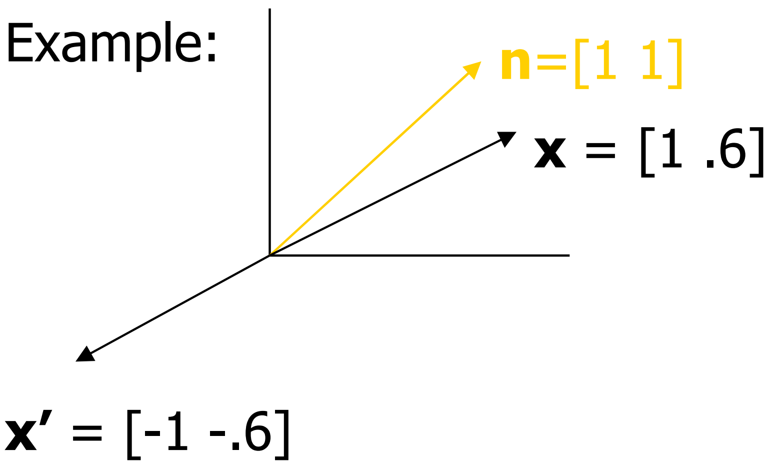
- **n** is a unit vector perpendicular to the dominant orientation.

- The notation [x]+ means x if x > 0 else 0

  The notation [x]- means x if x < 0 else 0

- We can think of E^+ and E^- as measures of how many gradient vectors in the window are on the positive side and how many are on the negative side of the dominant orientation in the window.

Example:



Scale Selection (continued)

- Texture scale selection is based on the derivative of the polarity with respect to scale σ.
- Algorithm:

1. Compute polarity at every pixel in the image for $\sigma_k = k/2$, ($k = 0,1\ldots7$).

2. Convolve each polarity image with a Gaussian with standard deviation 2k to obtain a smoothed polarity image.

3. For each pixel, the selected scale is the first value of σ for which the difference between values of polarity at successive scales is less than 2 percent.

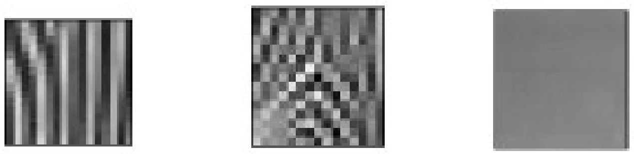
Texture Features Extraction

Extract the texture features at the selected scale

- Polarity (polarity at the selected scale) : $p = p_{\sigma*}$
- Anisotropy : $a = 1 - \lambda_2 / \lambda_1$

  λ_1 and λ_2 denote the eigenvalues of M_σ

  λ_2 / λ_1 measures the degree of orientation: when λ_1 is large compared to λ_2 the local neighborhood possesses a dominant orientation. When they are close, no dominant orientation. When they are small, the local neighborhood is constant.

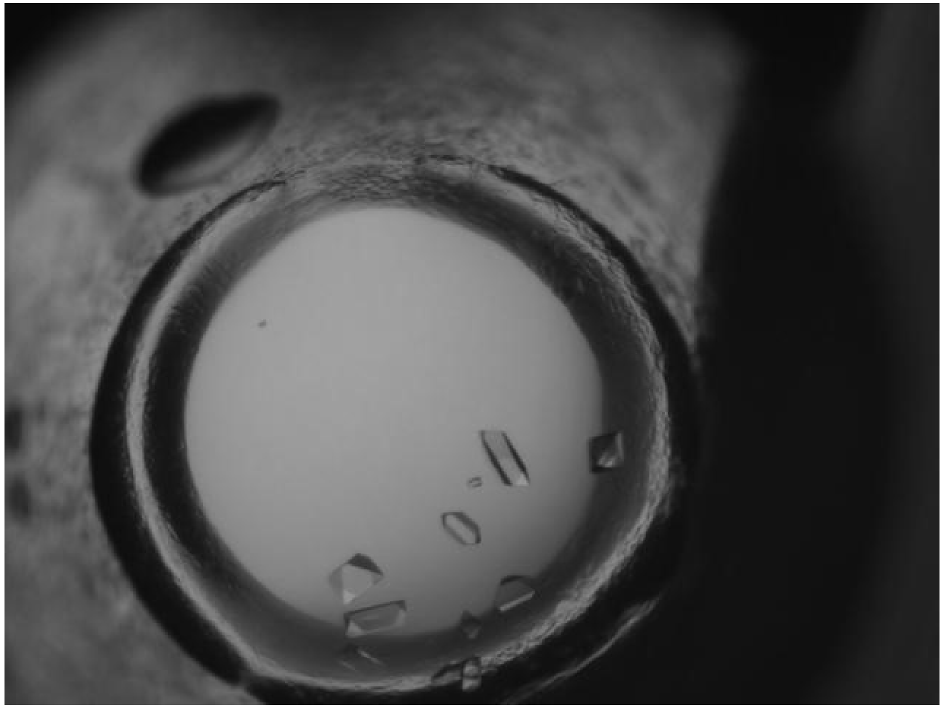


- Local Contrast: $C = 2(\lambda_1+\lambda_2)^{3/2}$

A pixel is considered homogeneous if $\lambda 1+\lambda 2$ < a local threshold

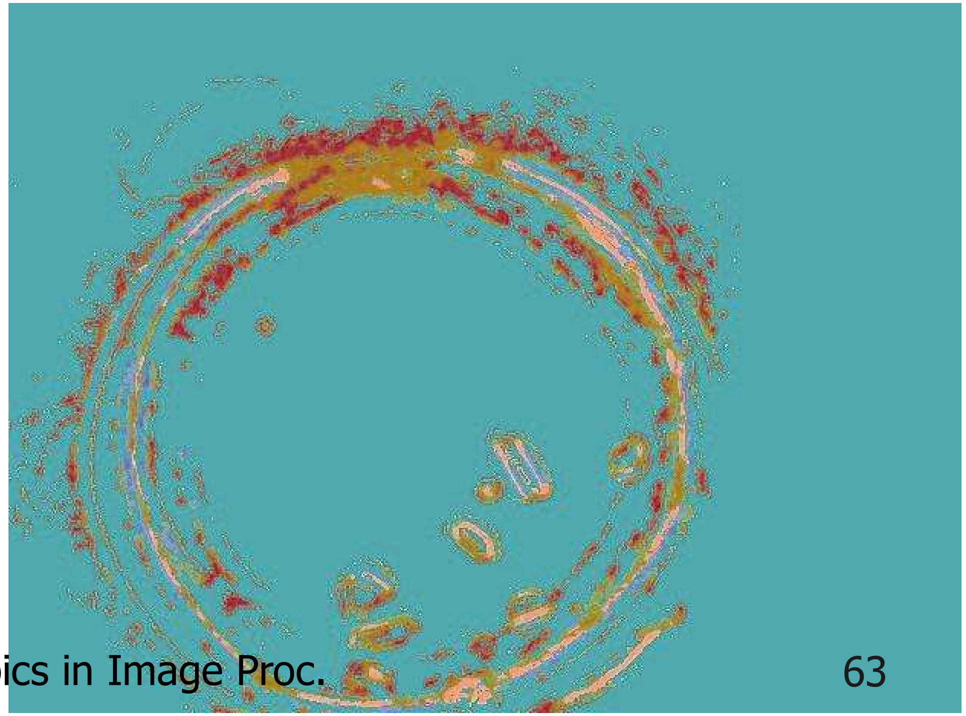
Blobworld Segmentation Using Color and Texture

Application to Protein Crystal Images

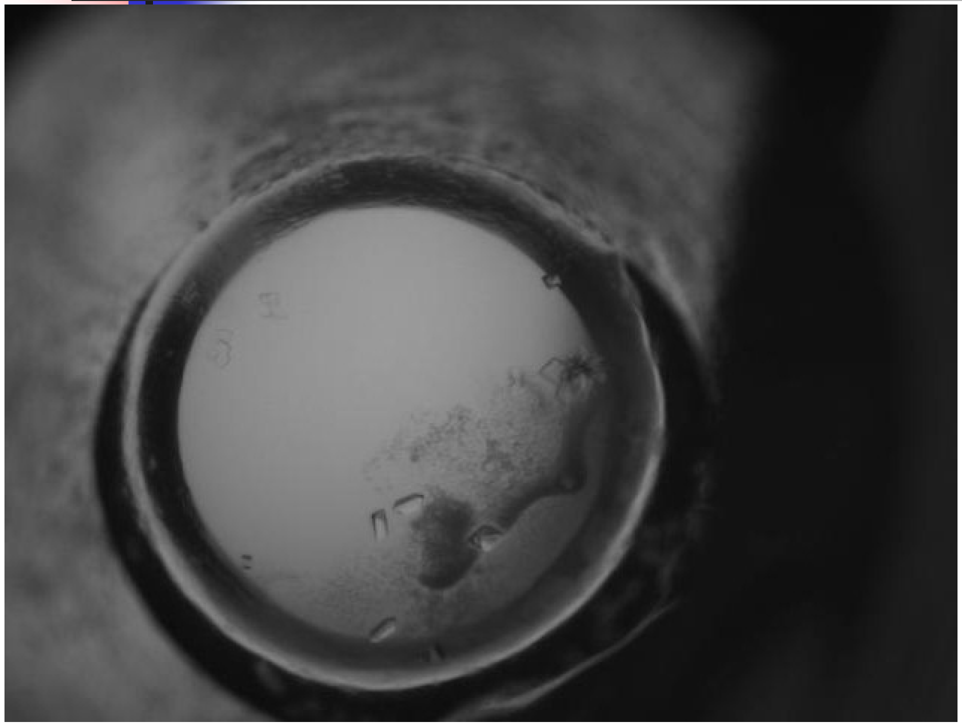


Original image in PGM (Portable Gray Map) format

- K-mean clustering result (number of clusters is equal to 10 and similarity measure is Euclidean distance)
- Different colors represent different textures



Application to Protein Crystal Images



Original image in PGM (Portable Gray Map) format

- K-mean clustering result (number of clusters is equal to 10 and similarity measure is Euclidean distance)
- Different colors represent different textures