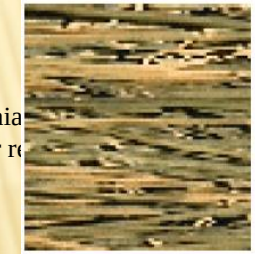
CSI/CUNY-2017



Welcome

Digital Image Processing

LECTURE 7- TEXTURE

Enjoy the class !

Sos Agaian

Distinguished Professor

Dept. of Computer Science, CSI/CUNY

ROADMAP TODAY

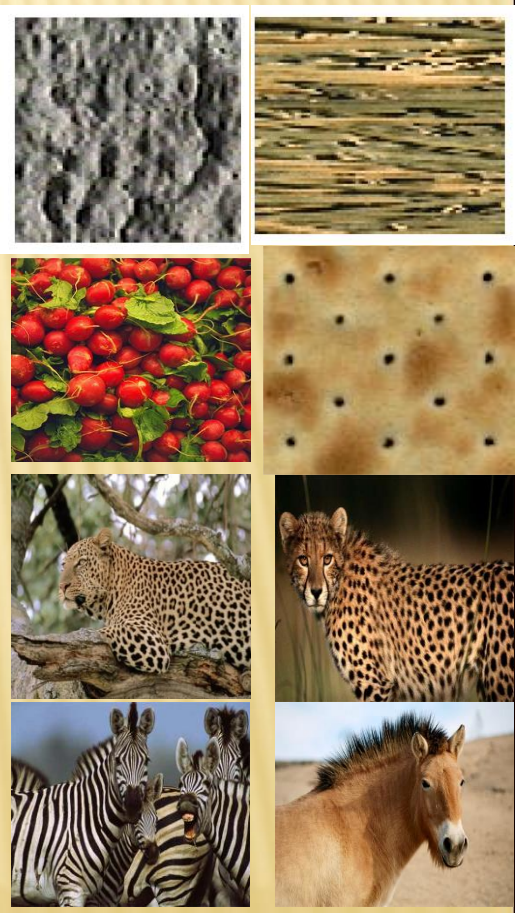
1. **Why texture?**
2. **What is texture?**
3. **Texture Related Tasks**
4. **Edge Density And Direction**

Model

1. **Capacity or Box-counting Dimension**

Statistical Texture Descriptors

1. **Histogram and Features**
2. **Local Binary Patterns**
3. **First Order Statistics**
4. **Grey Level Co-Occurrence Matrix**
5. **Haralick Features**



TODAY'S LEARNING GOALS

Understand what texture is, and the difference between first order and second order measures

Understand the

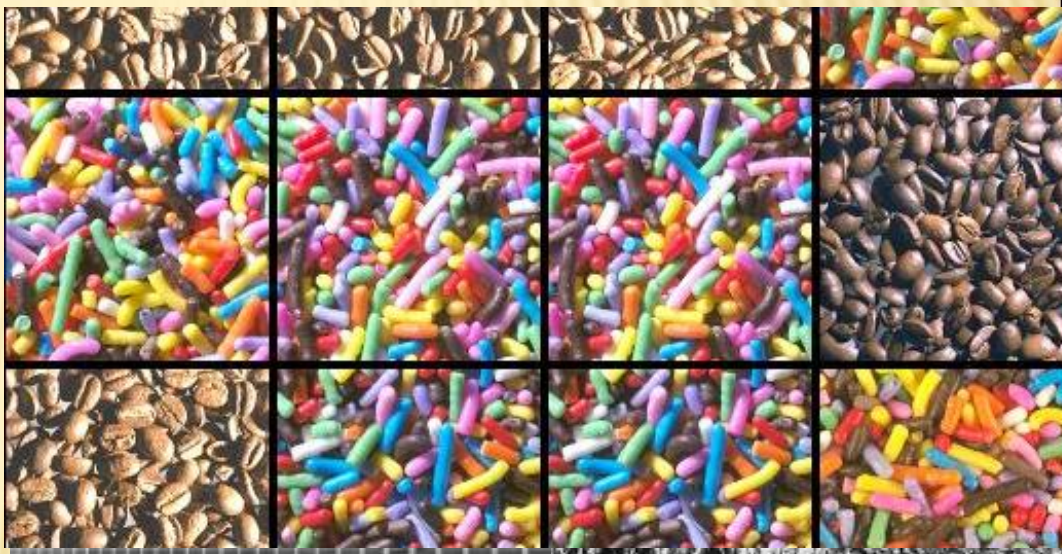
1. **Edge Density And Direction**
2. **Capacity or Box-counting Dimension**
3. **Histogram and Features**
4. **Local Binary Patterns**
5. **First Order Statistics**
6. **Grey Level Co-Occurrence Matrix**

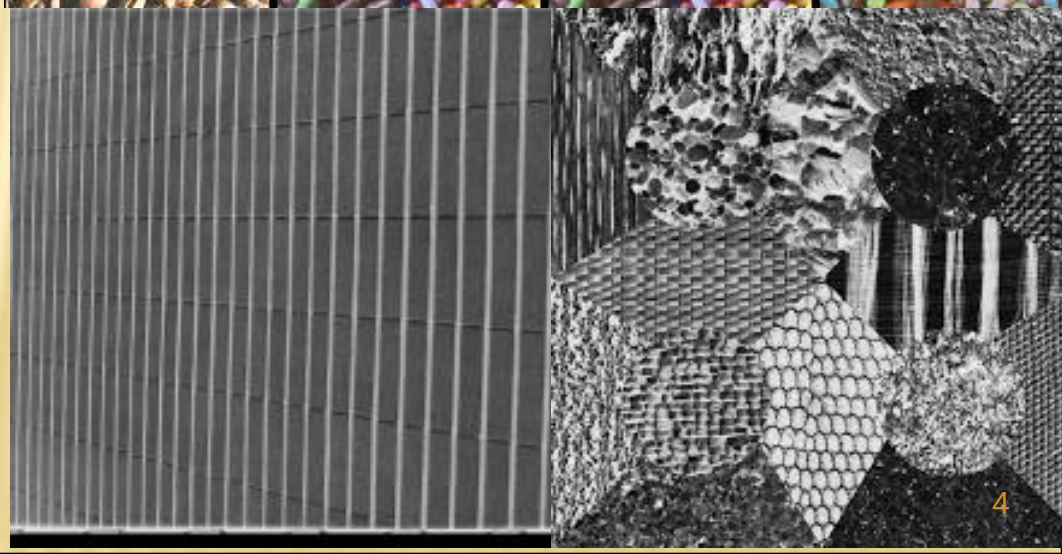
, and be able to describe algorithm

Understand how we go from an image to a GLCM feature image

Three Common Texture Sources

1. The Brodatz Album
2. MIT Vision Texture Database – no longer maintained Dec 2002
3. MeasTex Image Texture Database





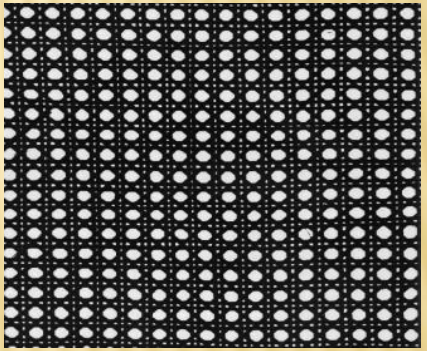
What is texture?

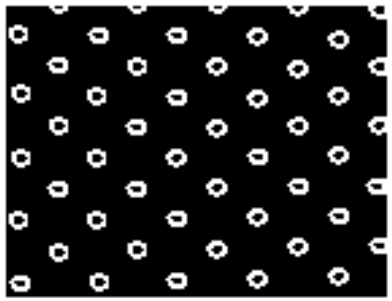
There is no accurate definition.

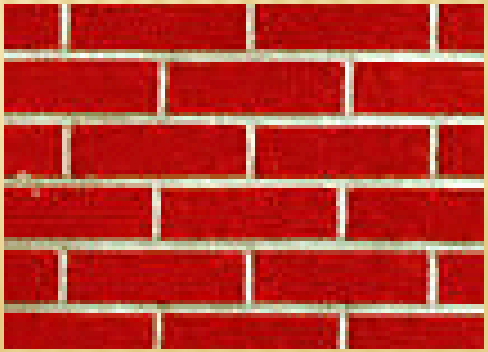
A measure of the variation of the intensity of a surface, quantifying properties such as smoothness, coarseness and regularity.

Source: *The Free On-line Dictionary of Computing, © 1993-2004 Denis Howe*

- Spatial arrangement of intensities within an image
- Similar Repeating Structures
- Degree of Randomness





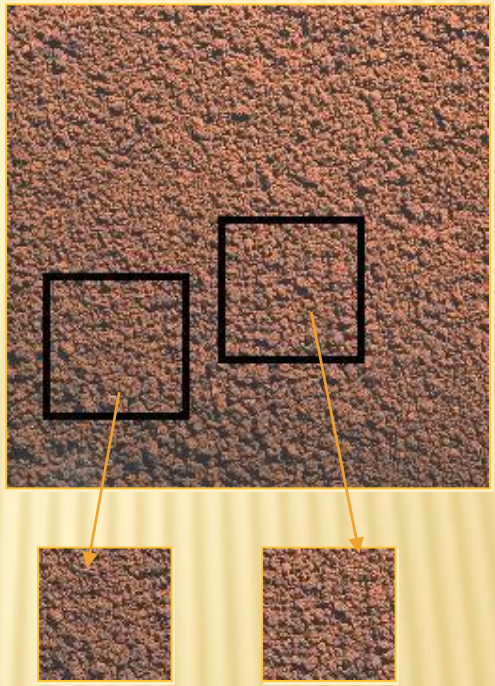


A texture is defined as an image composed by a high number of similar elements located in different positions of the image

Illustrative Example: What Is Texture?

Texture: pattern that
"looks the same"
at all locations, or

Images containing repeating patterns



May be structured or random

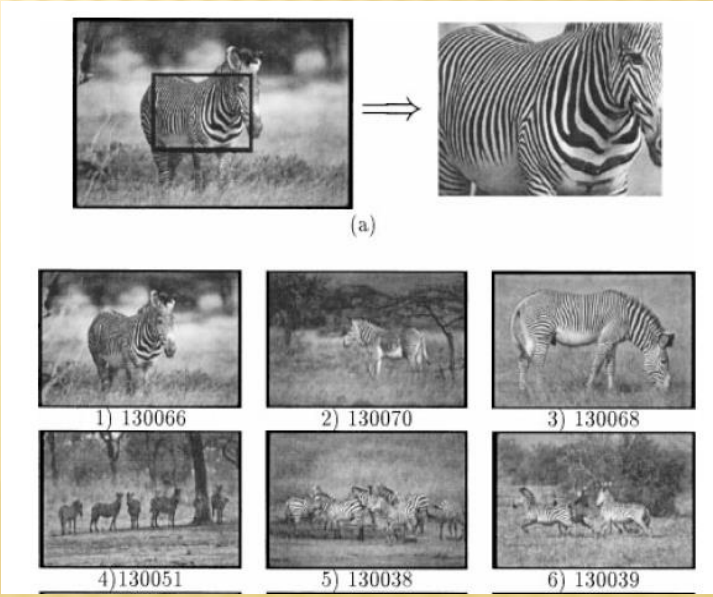
Definition: Texel

Texel = texture element, the fundamental unit of texture space. Can be defined in a strict geometrical sense, or statistically.

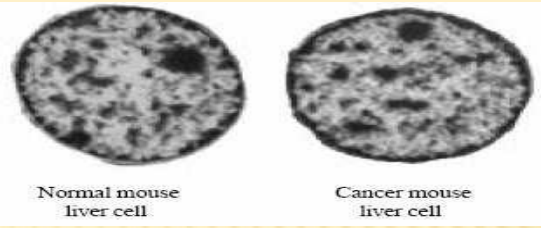
Note that you can define texels in any image scale, and that the best image scale for analysis is problem dependent.

Why Texture? Motivations and Applications

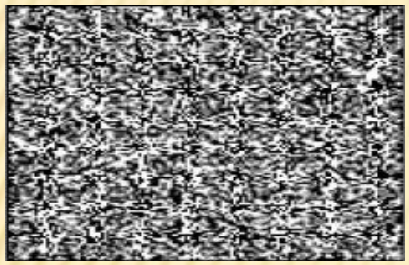
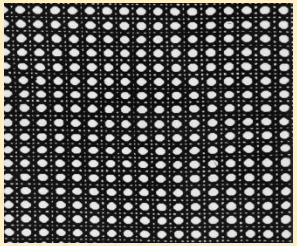
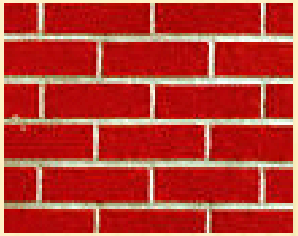
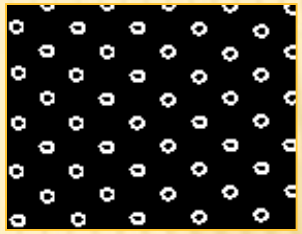
- Detect regions with textures
- Classify using texture
- Segmentation – dividing areas up by similar textures
- Synthesis – given a sample of the texture, generate random images with the same texture.
- Compression Image
- Restoration and Editing
- Image Retrieval
- Rendering Life-like Animations
  - Application to satellite images,
  - Application to medical images
  - Application to food processing industry
- Describes contents of real world images, i.e., clouds, fabrics, surfaces, wood, stone
- Texture descriptor provides measures of properties such as smoothness, coarseness, and regularity

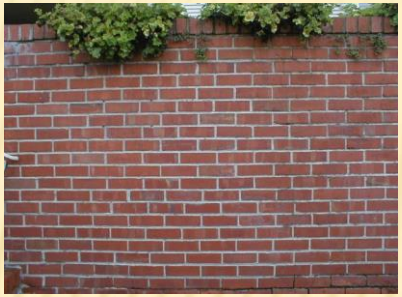


TEXTURE IS EVERYWHERE: FROM SKIN TO SCENE IMAGES

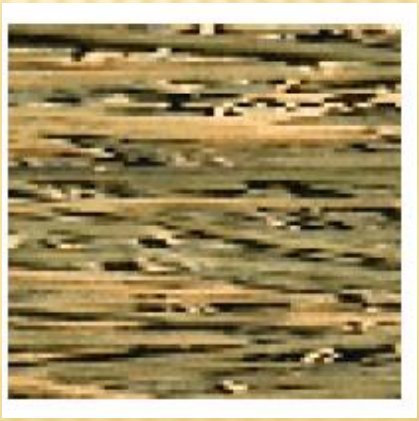
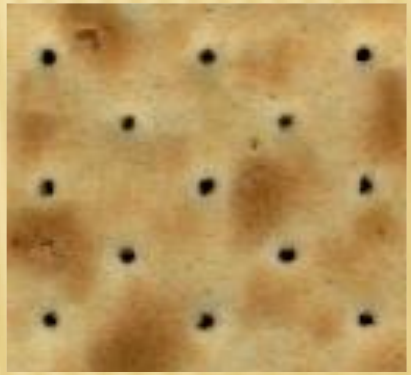


random



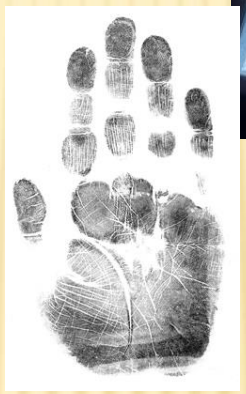




What defines a texture?

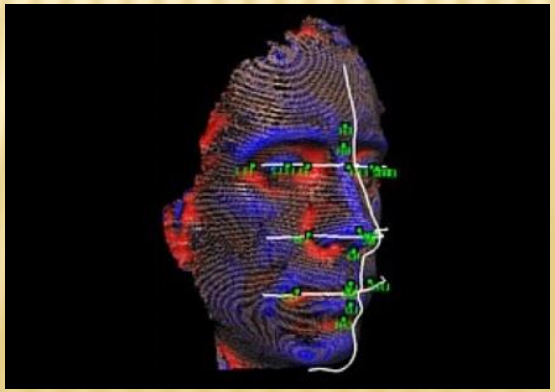
Why Texture? Motivations and Application

Food processing industry

Biometrics analysis (fingerprint, iris or retina, etc.)



Medical image analysis



Global information system (GIS) (for land, etc. analysis)

TEXTURE-RELATED TASKS

Shape from Texture

- Estimate surface orientation or shape from image texture

Texture Classification

- Concerned with identifying a given textured region from a set of texture classes. Statistical methods are extensively used. (e.g. GLCM, contrast, entropy, homogeneity)

Texture Segmentation:

given image with many textured areas, determine boundaries, or

Synthesis

- Generate new texture patches given some examples

Pattern recognition:

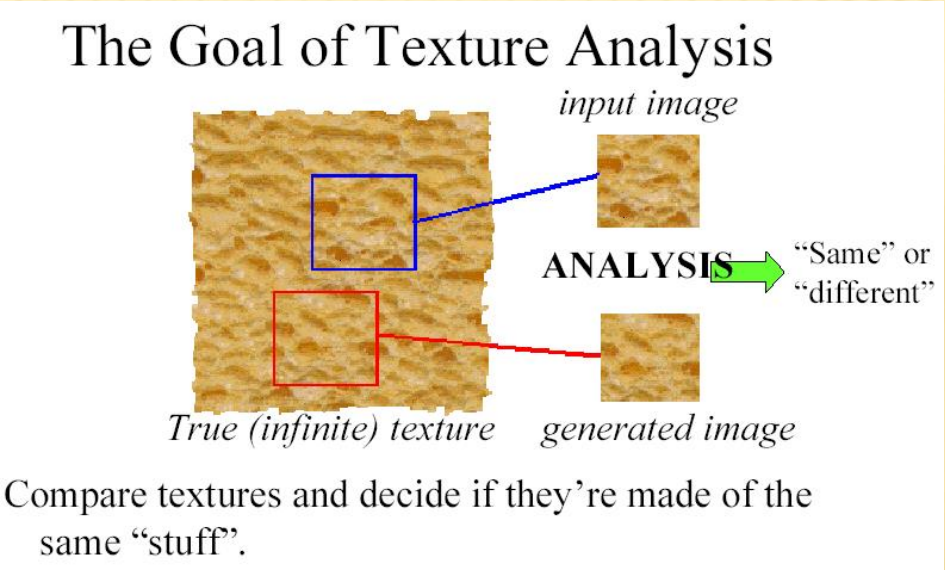
given texture region, determine the class the region belongs to

Generative model:

given textured region, determine a description or model for it

Why Analyze Texture? Texture Analysis

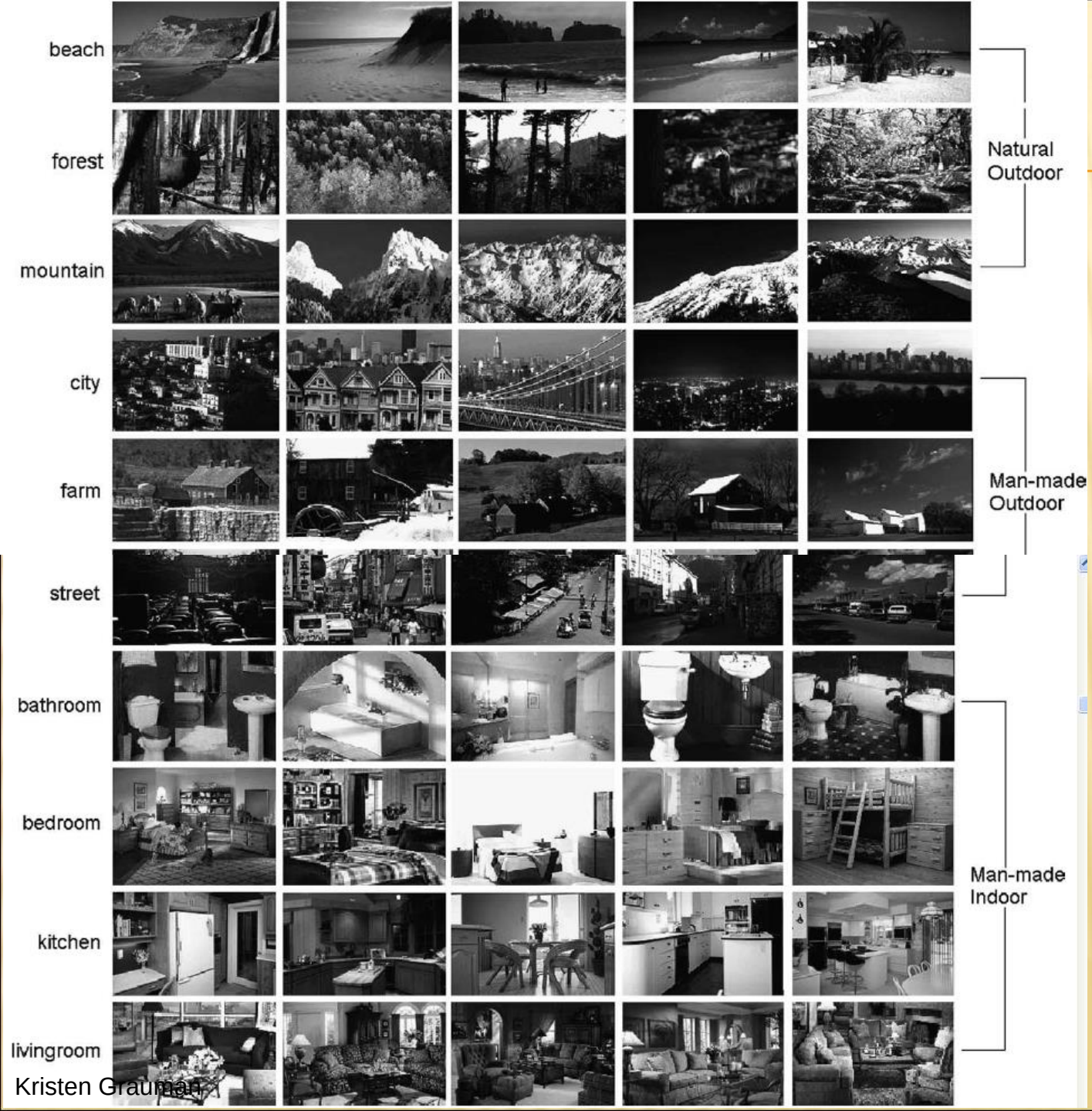
In general, *texture analysis* seeks to derive a general, efficient and compact quantitative description of textures so that various mathematical operations can be used to alter, compare and transform textures.



Why Analyze Texture? Classifying Materials, "Stuff"

Texture classification assigns a given texture to some texture classes

Figure by Varma & Zisserman



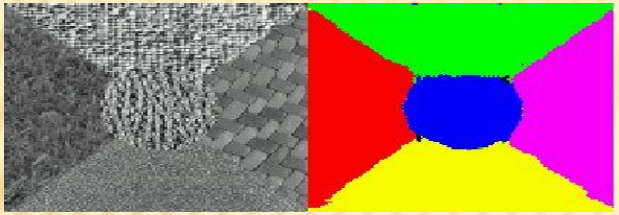
Why Analyze Texture?

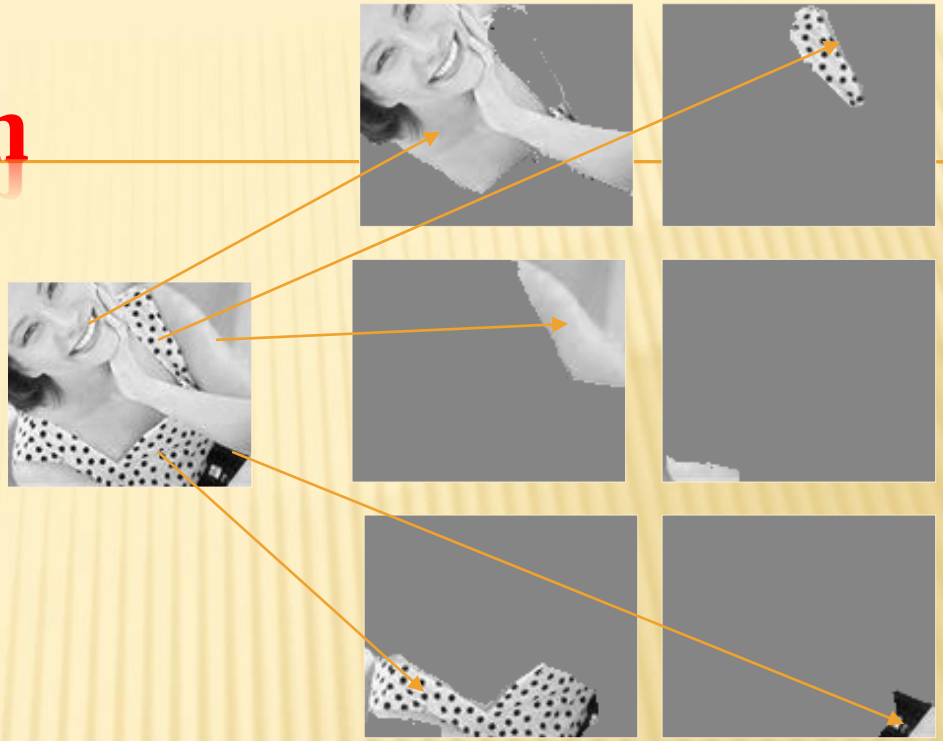
Characterizing scene categories by texture

L. W. Renninger and J. Malik. When is scene identification just texture recognition? Vision Research 44 (2004) 2301–2311

Texture Segmentation

Texture segmentation partitions an image into a set of disjoint regions based on texture properties, so that each region is homogeneous with respect to certain texture characteristics





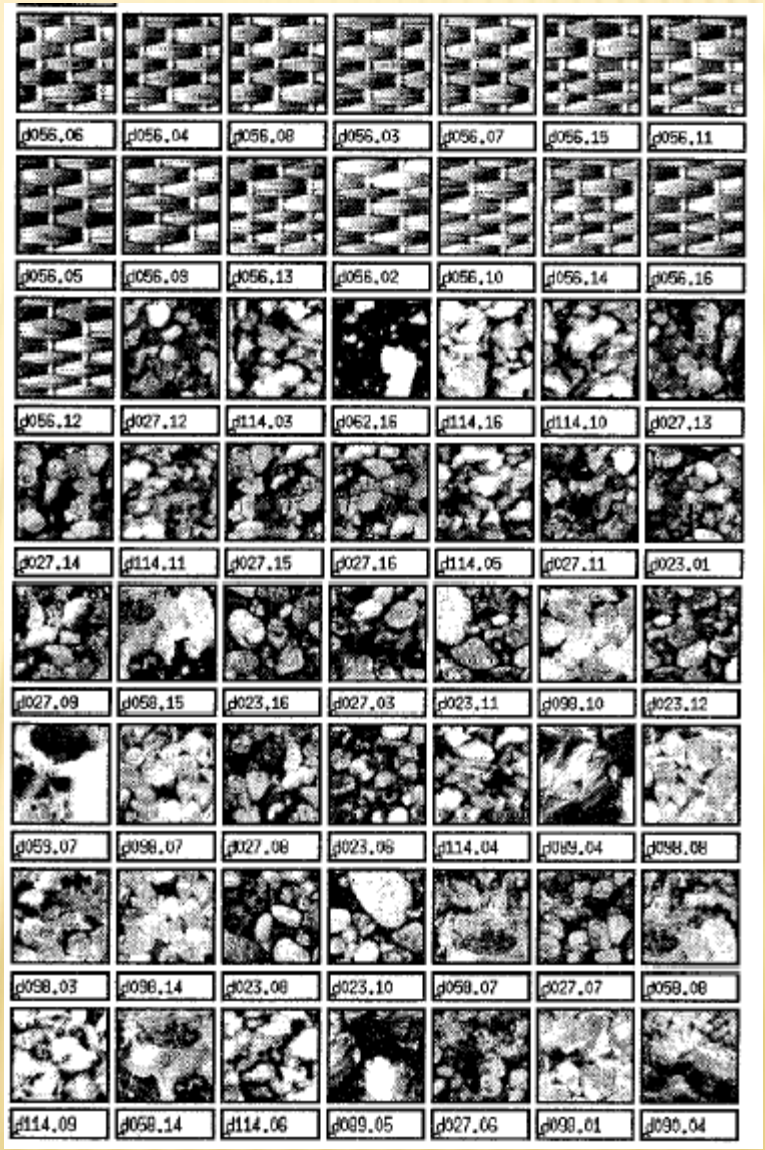
Segmenting aerial imagery by textures

Texture segmentation is very useful in a variety of applications of pattern recognition and machine leaning.

For example, in remote sensing and GIS analysis, texture segmentation could be applied to detect landscape change from an aerial photo.

Application: Texture-based Image Matching

Query image



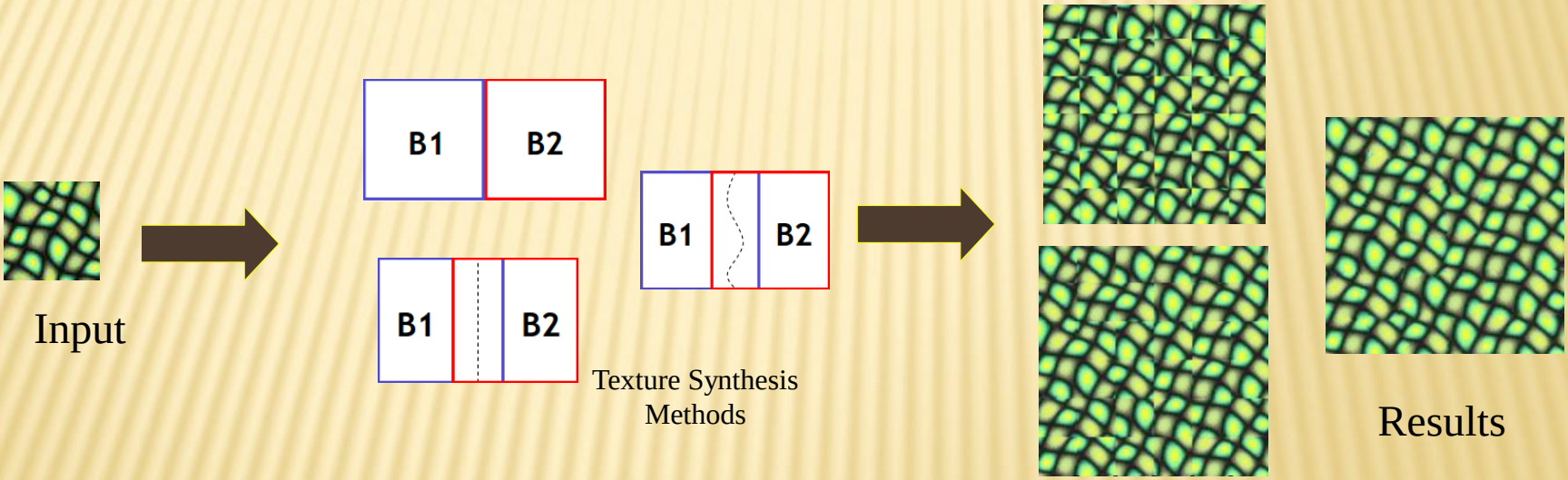
Ordered list of best matches

Decreasing response vector similarity

from Forsyth & Ponce

Why Texture? Texture Synthesis

Goal: Given a sample texture, to generate a new image having the same visual textural appearance, or construction of large regions of texture from given example images



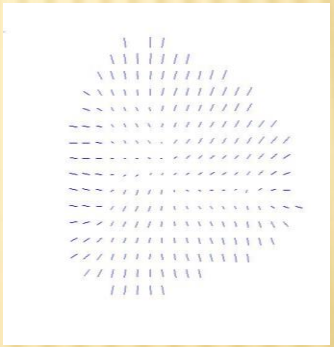
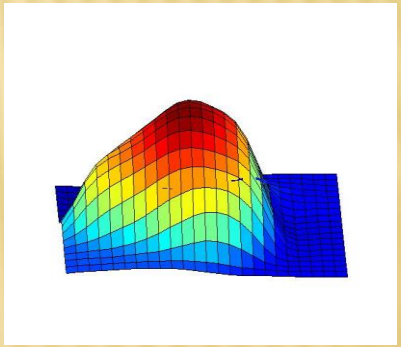
Many applications: computer graphics , image editing, video synthesis virtual environments, hole-filling, texturing surfaces.

SHAPE FROM TEXTURE

Use deformation of a text from point to point to estimate shape, or estimate surface orientation or shape from image texture

- Recover 3D information from 2D image

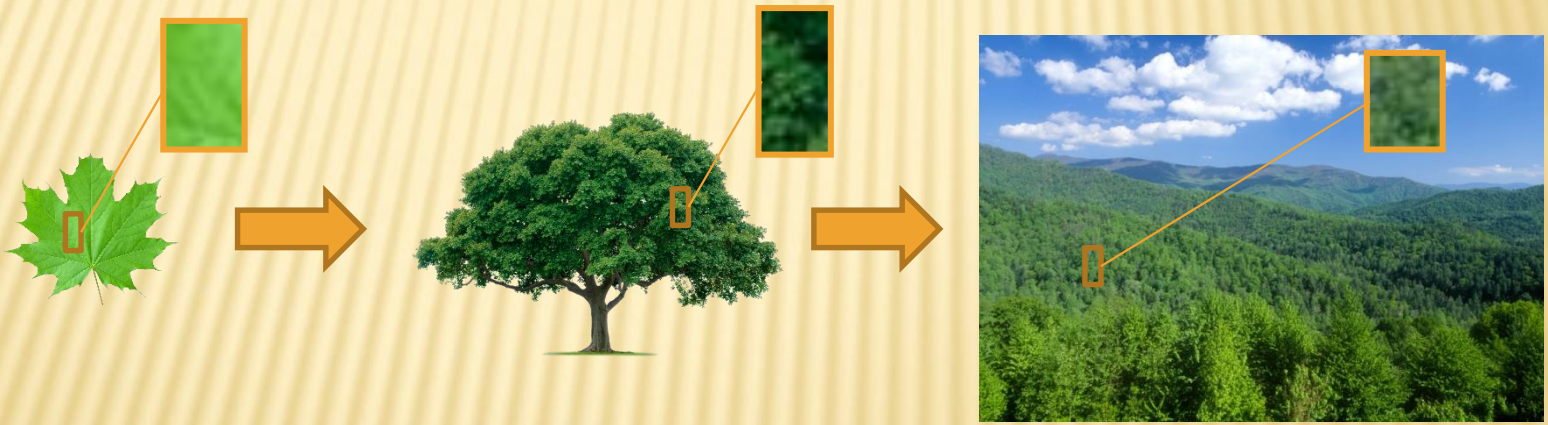


Pics from A. Loh

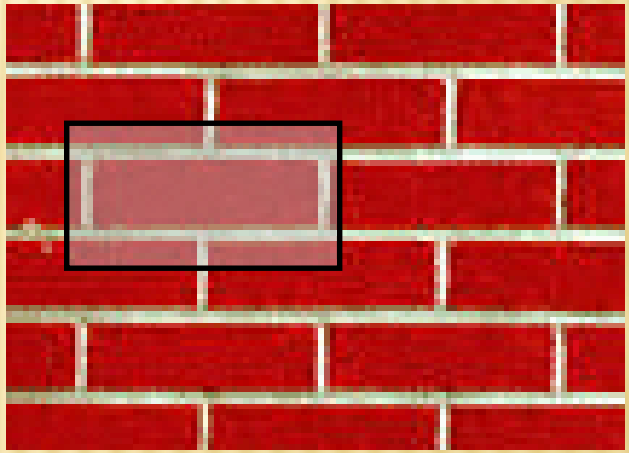
OBJECTS VS. TEXTURE

Depending on the scale, the same thing can be presented as an object or a texture



PERIODIC TEXTURE ANALYSIS

- Assumes the entirety of the image is periodic
- No statistical variation
- Too limited of a model

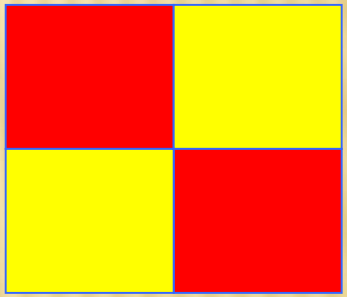


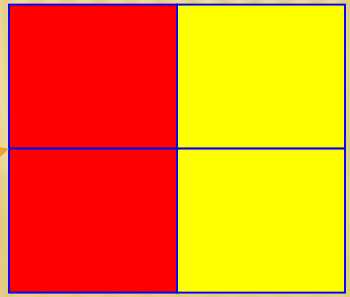
CHALLENGING ISSUES

Variance feature computed in window of size 3x3

Variance feature computed in window of size 15x15

- Rotation and scale invariance (3D)
- Segmentation/extraction of texture regions from images
- Texture in noise
- Scale impacts the choice of texels, and vice versa
- Texture is a spatial property, 1D histograms cannot be used to accurately characterize complex textures alone
  - Every pixel is independent and everything happens at a tiny scale
- Example – The two distinct textures below will produce the same histogram





Computing Texture Images

- Select a window size and select a texture feature
- for each pixel (i,j) in the image:
  - Center the window at pixel (i,j)
  - Compute the texture feature

  Assign the computed value to the center pixel (i,j) in a new image of the same size

  - Pixels close to the image border can be handled in the same manner as for filtering/convolution

Two Edge-based Texture Measures: 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

1. edginess per unit area

$$F_{edgeness} = \frac{\left|\{p \mid Mag(p) \geq T\}\right|}{N}$$

where N is the size of the unit area, T is a threshold
Note: edginess measures the busyness, but not the orientation of the texture

2. edge magnitude and direction histograms

Fmagdir = (Hmagnitude, Hdirection)

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

Capacity or Box-counting Dimension

The Box-Counting Dimension was introduced by Kolmogorov in the 1930 and it was called entropy dimension, or Kolmogorov entropy. Sometimes it is called logarithmic density [K. Falcone, Fractal Geometry].

Box counting is a method of gathering data for analyzing complex patterns by breaking a dataset, object, image, etc. into smaller and smaller pieces, typically "box"-shaped, and analyzing the pieces at each smaller scale (https://en.wikipedia.org/wiki/Box_counting)

Why do we care about the fractal dimension? How is this useful?

The Box-Counting Dimension has been used for characterization of complex and irregular shapes

The fractal dimension (*FD*) can be used as a measure

1. for morphological complexity in biological systems
2. for calculating fractal dimension of urban form based on remote sensing image (It can describe the fractal characteristic of the city,

http://ieeexplore.ieee.org/document/4241514/,
http://www.tandfonline.com/doi/pdf/10.1007/s11806-009-0096-1)

BOX COUNTING METHOD FOR BINARY IMAGE

For a set A, $N_n(A)$ = number of boxes of side $1/2^n$ which interser the set A:

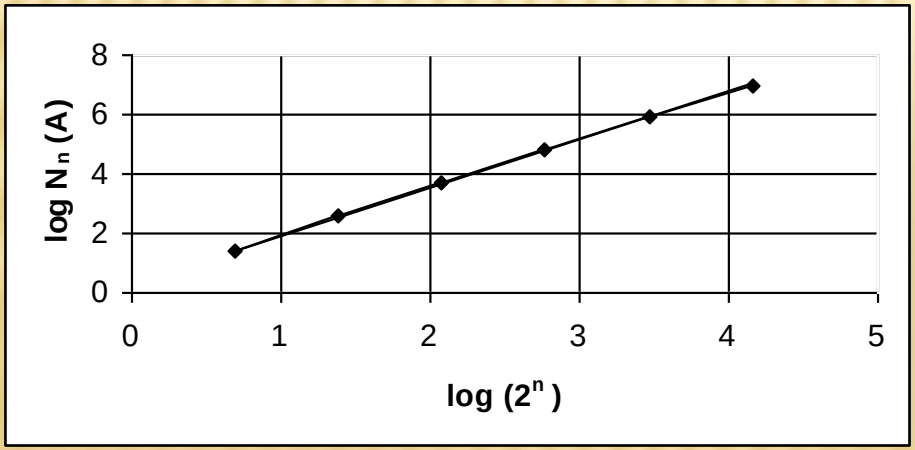
$$D = \lim_{n\to\infty} \log N_n(A) / \log 2^n$$

Or,

$$D = 1 - \lim_{n\to\infty} \log N_n(A) / \log 2^n$$

n	N_n (A)	2^n	log N_n (A)	log 2^n
1	4	2	1,386	0,693
2	12	4	2,484	1,386
3	36	8	3,583	2,079
4	108	16	4,682	2,772
5	324	32	5,780	3,465
6	972	64	6,879	4,158

Challenges: To find fractal dimension, it is not easy to choose the sizes of the boxes for complex images like the radiographic images



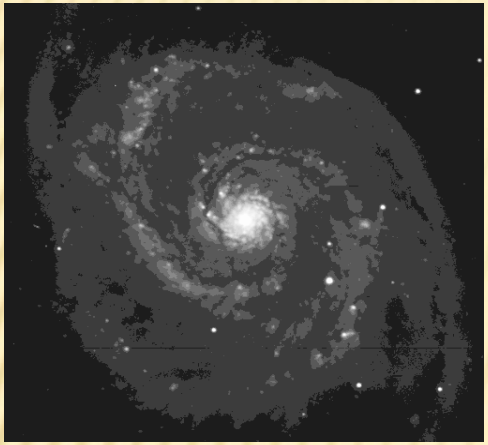
The slope of the line equals the dimension, and it is defined as the amount of change along the Y-axis, divided by the amount of change along the X-axis

The General Process of the Box-counting Method

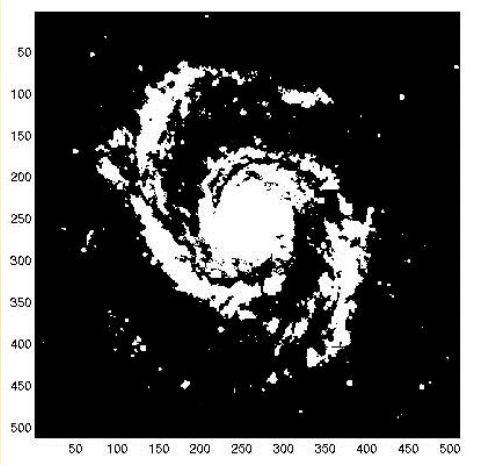
is as follows.

1. cover the research object by a rectangle. Certainly, the rectangle is non-empty. The number of non-empty grids is 1.
2. divide the side of the rectangle into two parts, and get 4 grids. The size of the grid is 1/2. Cover them on the object, and the number of the non-empty grids we obtain is N （½
3. quarter the side of the rectangle, and get 16 grids. The size of the grid is 1/22. Cover them on the object, and the number of the non-empty grids we obtain is N (1/22).
4. Continue this process until the side of the rectangle is divided into 2n parts. Moreover, we will get 2n×2n grids.
5. Cover them on the object, and the number of the non-empty grids we obtain will be N(1/2n). Here, 1/2, 1/22, …,1/2n is the size of grid scale δ, and n is seen as the rank of division.
6. Generally, we choose n=9. Thus, we can get nine pairs of data about this size and the corresponding number of the non-empty grids.
7. The absolute value of the slope of the fit line is just the fractal dimension of the research object (Slope D of the linear portion of function $\log N(r) = D(\log(1/2)) + \log k$ is assumed to be the box (fractal) dimension and its k intercept)

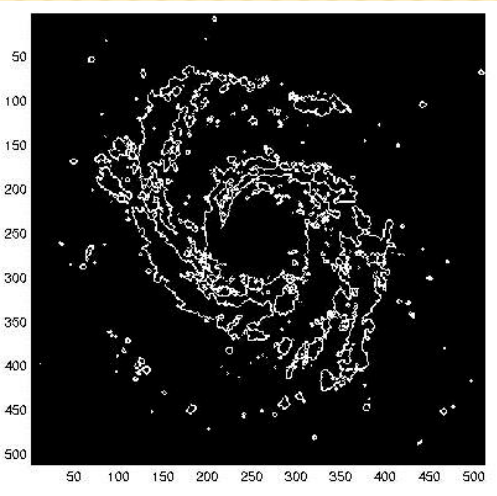
Illustrative Example



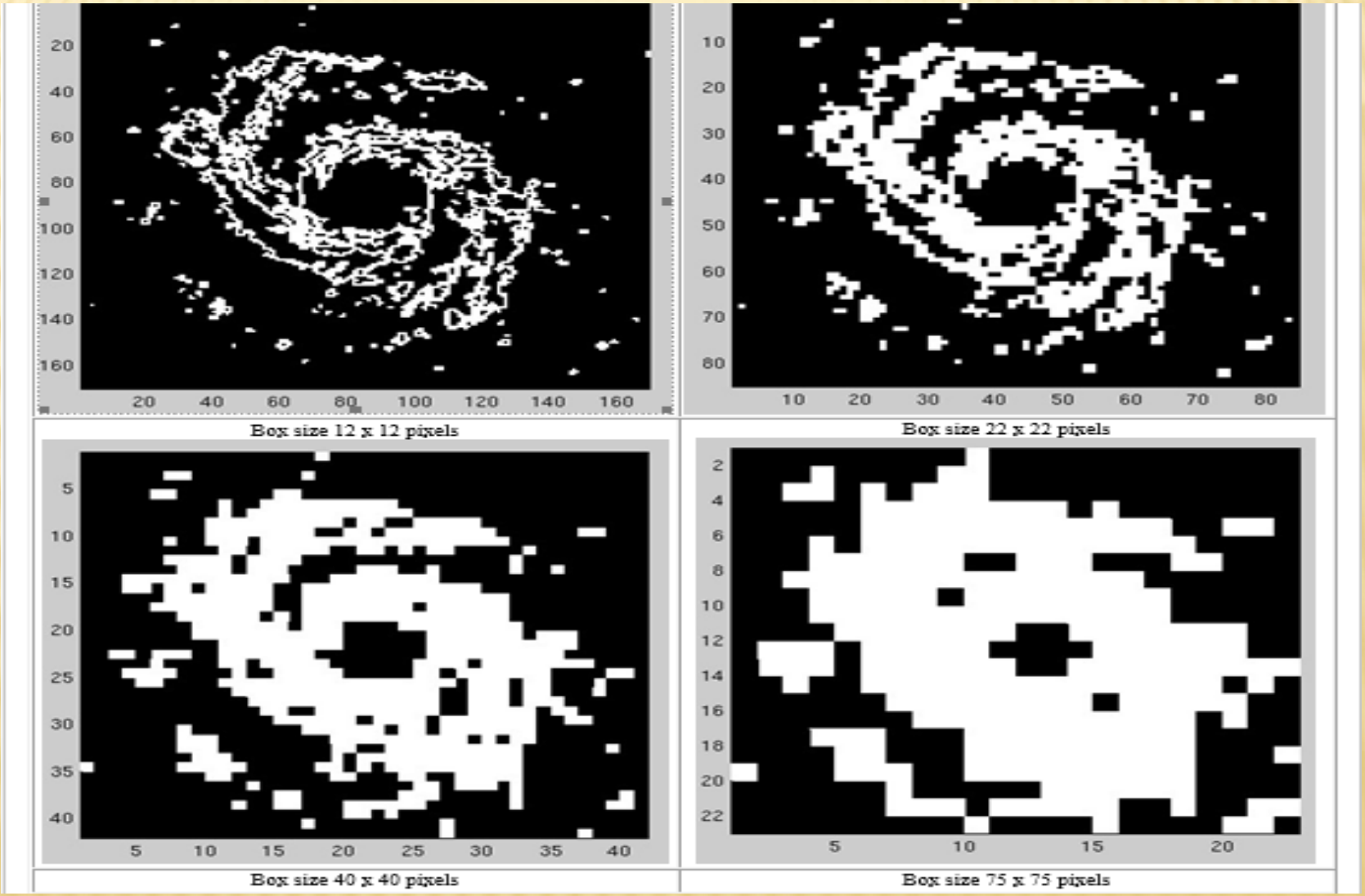
- Next, we produce a binary version of the image. All pixels above a chosen brightness are set to one, the rest are set to zero.



Then we break the image up into boxes of a given size and count how many of those boxes contain the contour. We then repeat this process with several different box sizes.



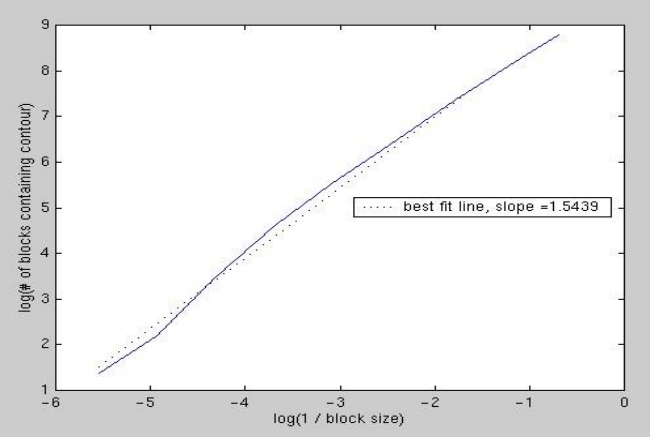
Illustrative Example:BOX COUNTING



The galaxy image at the top of the page produced the following results

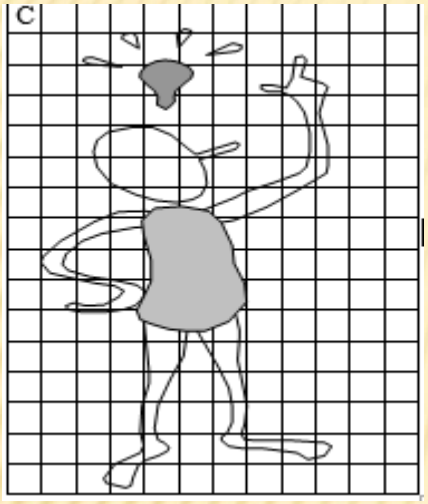
Box size (in pixels)	number of boxes containing the contour
2	6544
3	3897
6	1562
12	591
22	250
40	101
75	32
138	9
256	4

Next we plot the log of the box size vs. the numbers we counted for that box size.



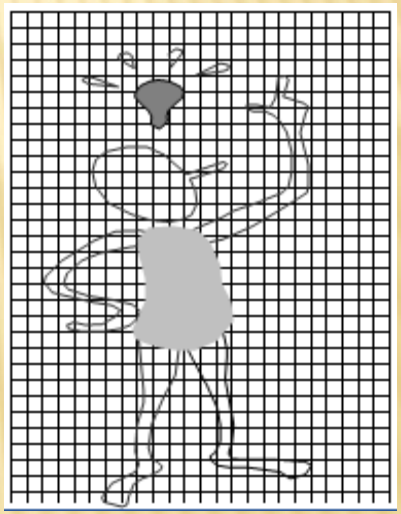
The slope of the resulting plot gives us the fractal dimension of the image, in this case 1.5439

BOX COUNTING: LET'S START WITH FOLLOWING EXAMPLE



Let put the image onto a grid with mesh size s=1/2 , and count the number of grid boxes N which contain some of the image(it is clear that the N depends on the size s, so we write N(s)).

Now, we reduce the mesh size by factor s_1, $s_1 < s$, let for example $s_1 = s/2$, and count the number of grid boxes $N(s_1)$ which contain some of the image.



The Box-Counting Dimension is calculated by

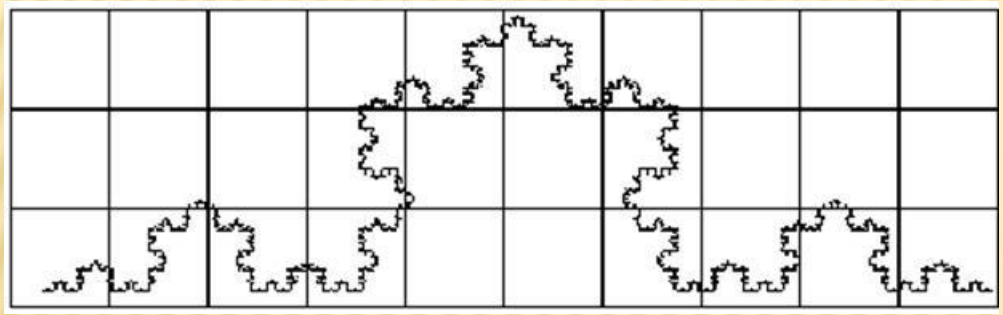
$$D_B = [\text{Log}N(s) - \text{Log}N(s_1)] / [\text{Log}s - \text{Log}s_1]$$

Remark: The problem with the self similarity dimension is that it is useful only for a class of the self-similar sets. But what we have to do with the structure, which is not self-similar.

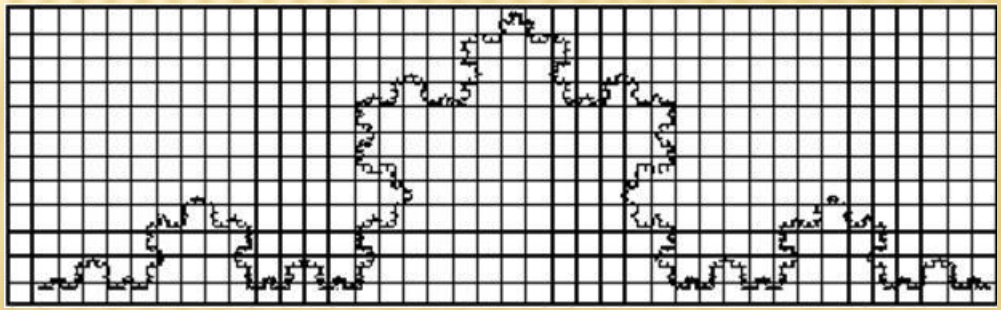
Box Counting Method – Fractal Dimension Calculation

- In order to calculate fractal's dimension you begin by covering the fractal image area with different grid sizes. Then count the number of grid blocks containing part of the fractal in them.

S=1/3



S=1/12



Box Counting Method – Fractal Dimension Calculation

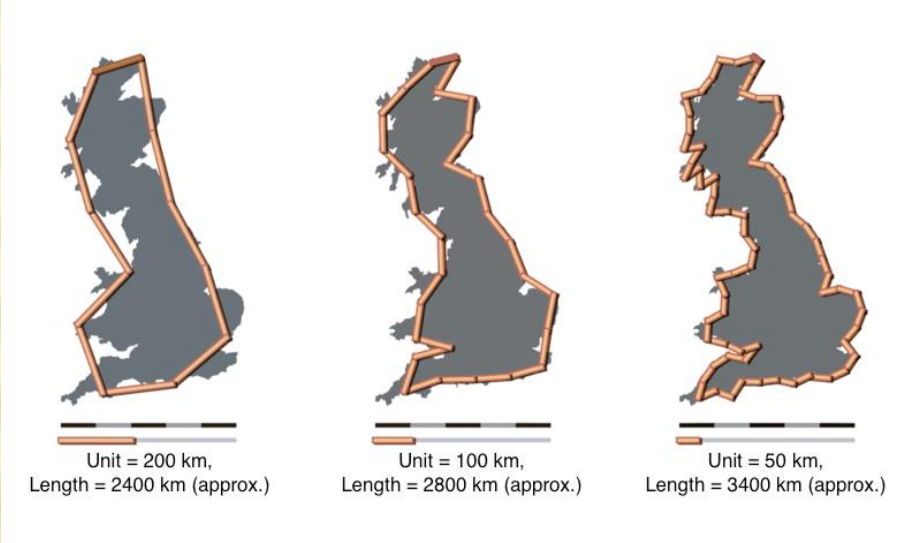
- After counting a sufficient amount of grid sizes we calculate the Fractal dimension using the formula: $D_{box} = \frac{\log n(a) - \log n(b)}{\log(1/s_a) - \log(1/s_b)}$
- In the previous example, Where n() is the number of grid blocks containing a part of the fractal and 1/S is the grid scale.
- $D_{grid\frac{1}{3}x\frac{1}{12}} = \frac{log105 - log18}{log12 - log3} \approx 1.27$ or $D_{grid\frac{1}{3}x\frac{1}{6}} = \frac{log48 - log18}{log6 - log3} \approx 1.19$

Which if we cont. to more measurements will all average to 1.26 which is the calculated dimension of the Koch Fractal.

How To Solve The Coastline Paradox?

"How long is the coast of Britain,"

the answer is that it depends on how closely you look at it, or how long your measuring stick is



If we measure the length of the west coast of Britain with a large ruler, we get a certain value for the length of the coastline.

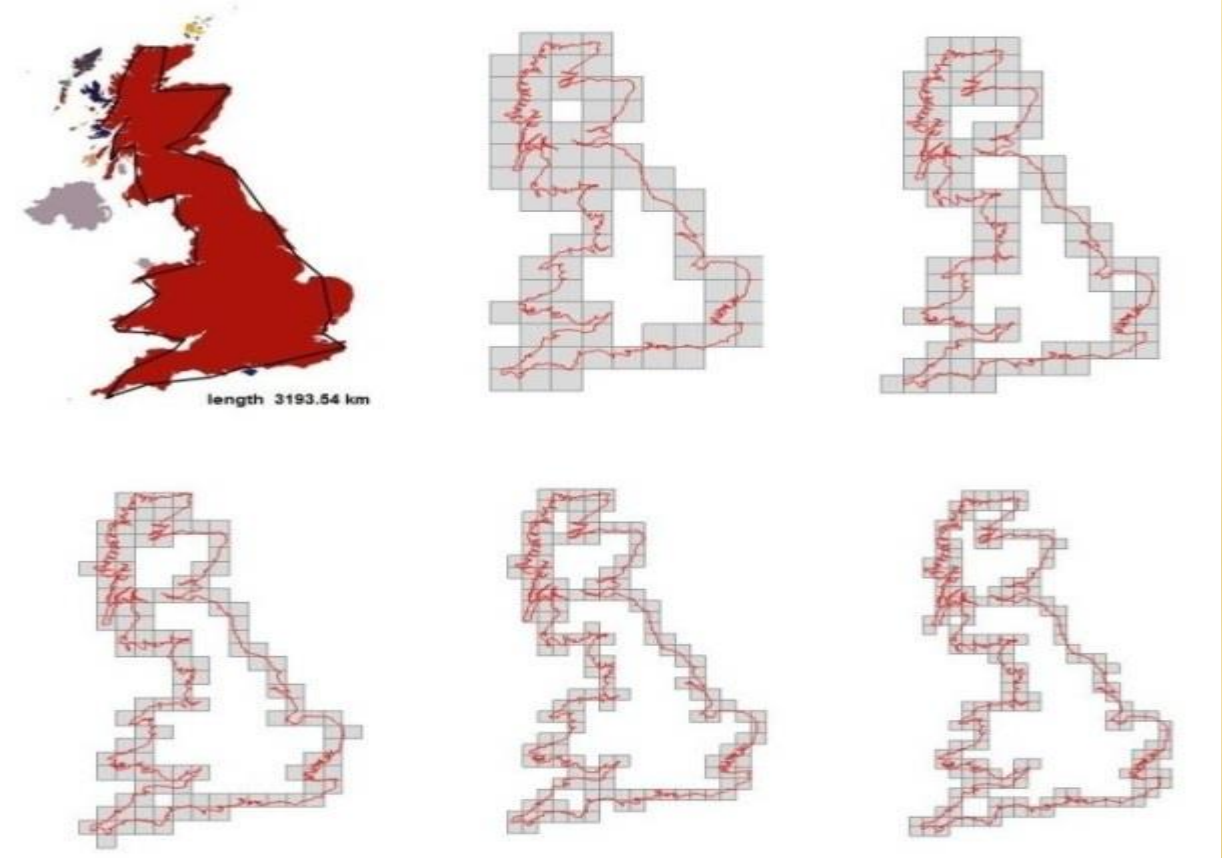
If we now measure it again with a smaller ruler, we catch more of the smaller bays and peninsulas that we missed before, and so the coastline is longer.

Conclusion: The coastline measurement problem. In short form, the length of the coast line is a function of the size of measurement one employs.

The measured value depends on the resolution used to do the measurement

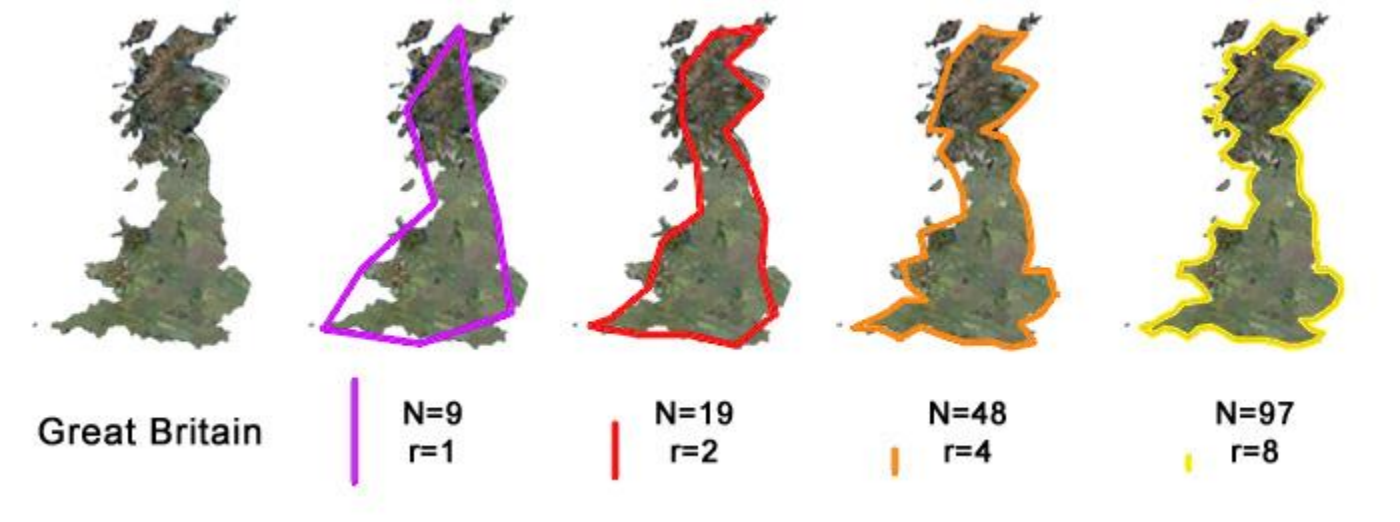
How long is the coast of Staten Island ?

Here, the first steps of the box-counting procedure about England's coastline are represented.

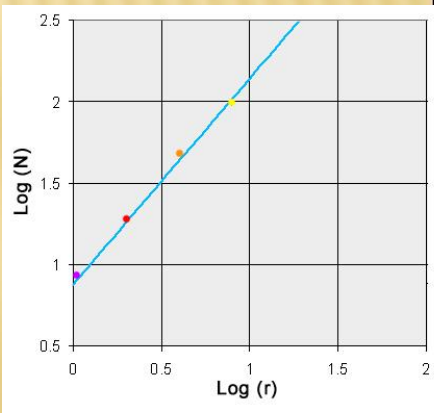


http://demonstrations.wolfram.com/BoxCountingTheDimensionOfCoastlines/

Coastline Measurement



When we use a large ruler (**r=1**, a small magnification factor), we get a very poor approximation, shown in purple, and a value for the perimeter of **N=9**. As the ruler length shrinks, the magnification **r** increases, and the value of the perimeter **N** increases. We are interested in the *rate* at which the perimeter changes as a function of the ruler length.



the coastline of Britain found it's dimension to be **1.25**

The Coast of Britain (second drfinition)

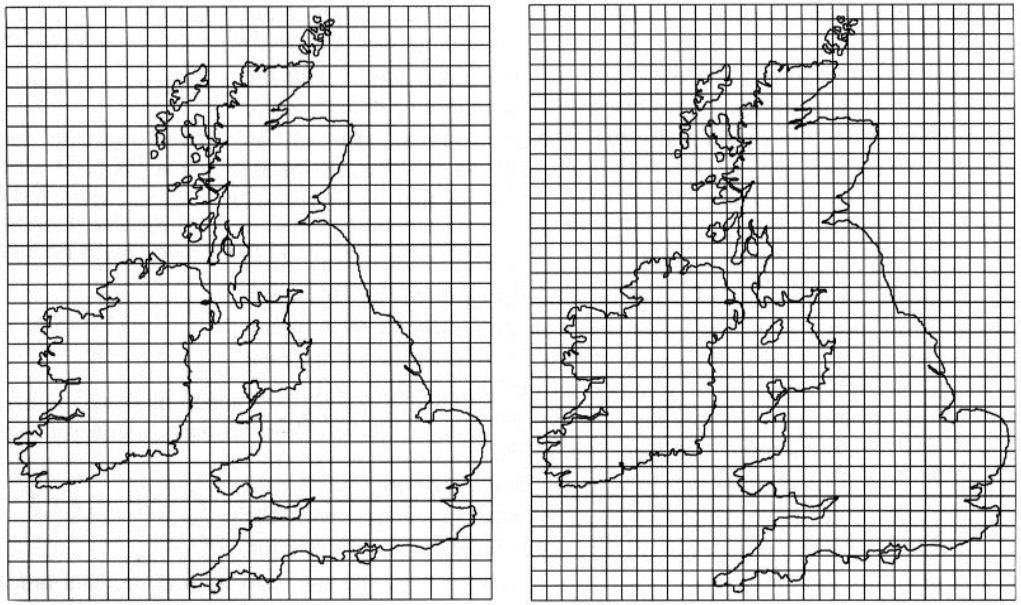


Figure 4.32 : Count all boxes that intersect (or even touch) the coastline of Great Britain, including Ireland.

Box-Count

The wild structure is box-counted using two grids. The slope of the line is $\log(52/19)/\log 2 \approx 1.45$.

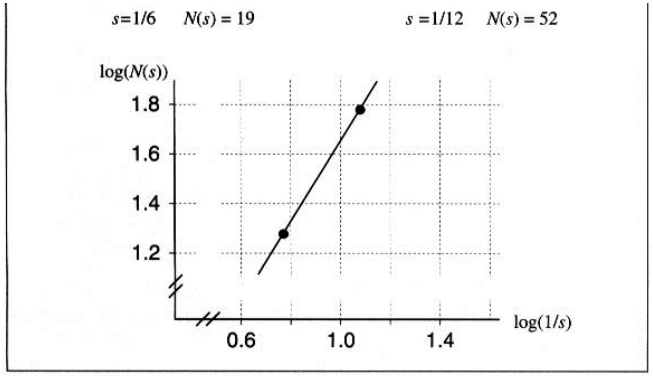
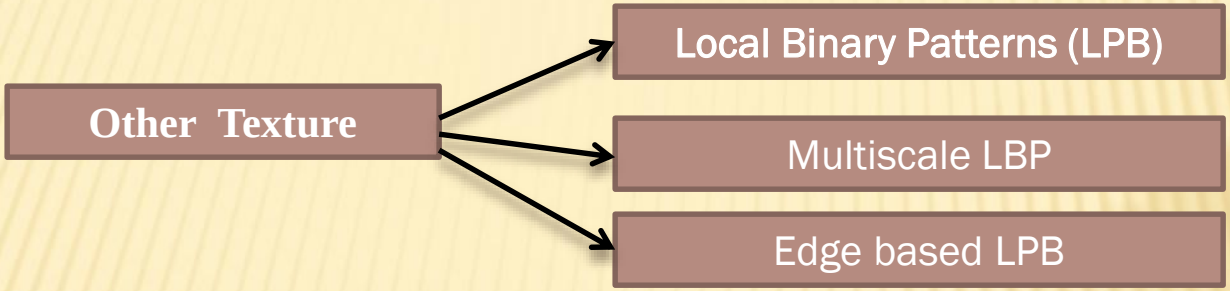


Figure 4.30

Texture: Local Binary Patterns



- **Local Texture Feature: Local Binary Patterns** (LBP) **:** LBP is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

Important properties::

- computational simplicity, which makes it possible to analyze images in challenging real-time settings.
- various real-world applications.
- LBP is invariant to any monotonic gray level change . For example, by illumination variations.

For a bibliography of LBP-related research, see
http://www.ee.oulu.fi/research/imag/texture/.

Local Binary Patterns (LBP)

For each PIXEL of an image, a BINARY CODE is produced

- to make a new matrix with the new value (binary to decimal value).

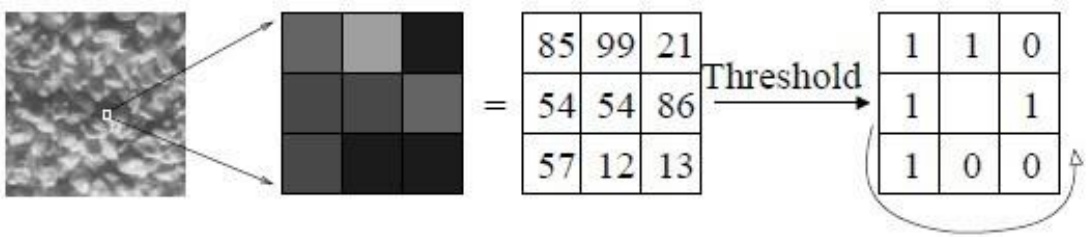
It was originally defined for 3x3 neighborhoods, giving 8 bit codes based on the 8 pixels around the central one. Formally, the LBP operator takes the form

$$LBP(I_C) = \sum_{i=0}^{7} s(I_i - I_C)2^i$$

where in this case i runs over the 8 neighbors of the central pixel c
Ic and in are the gray level values at c and i

$$s = \begin{cases} 1 & if \quad I_i - I_C > 0 \\ 0 & if \quad I_i - I_C < 0 \end{cases}$$

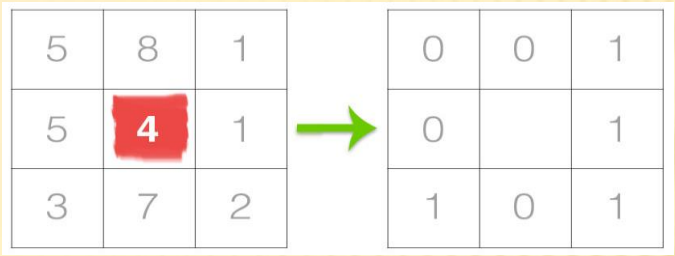
The LBP encoding process is illustrated in



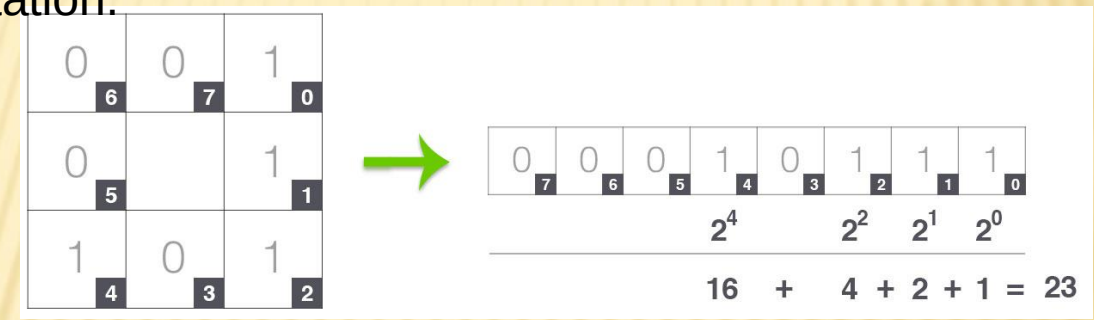
$$LBP(I_C) = \sum_{i=0}^{7} s(I_i - I_C)2^i$$

Fig. 1. The basic LBP operator.

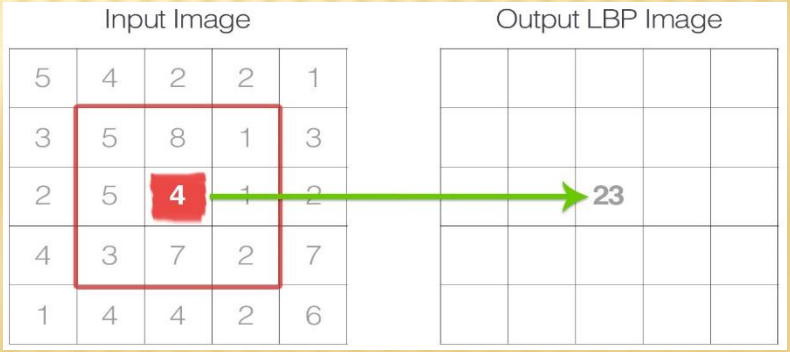
Illustrative Example-1: let's take a look at the original LBP descriptor which operates on a fixed *3 x 3* neighborhood of pixels just like this:



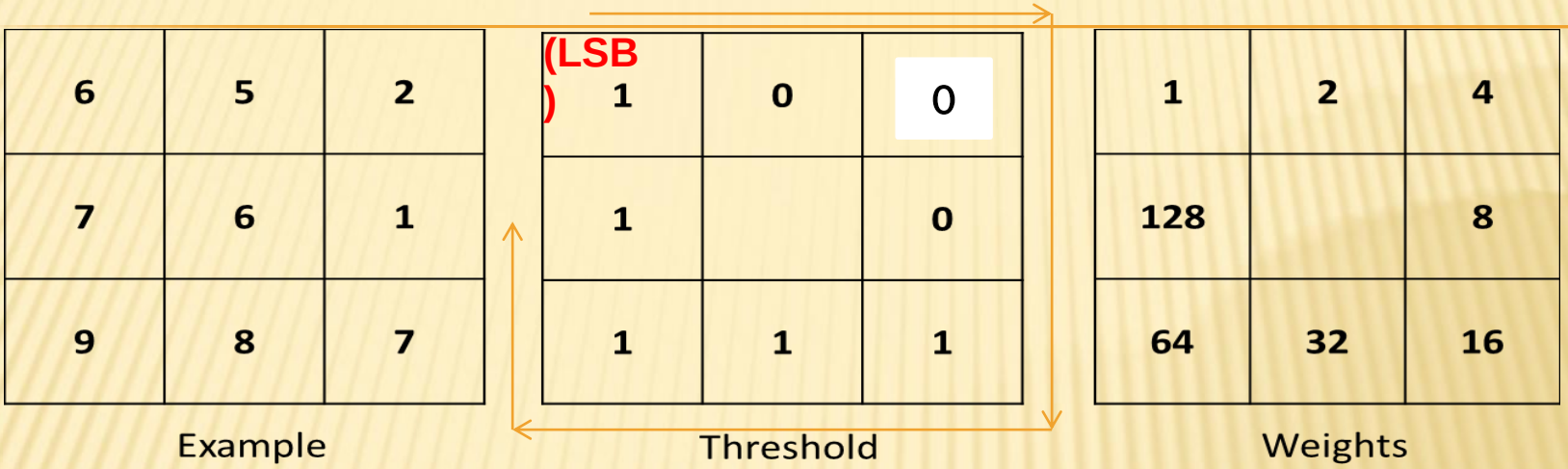
Step 2. take the 8-bit binary neighborhood of the center pixel and convert it into a decimal representation.



Step 3. store the calculated LBP value in an output array with the same width and height as the original image



Illustrative Example-2

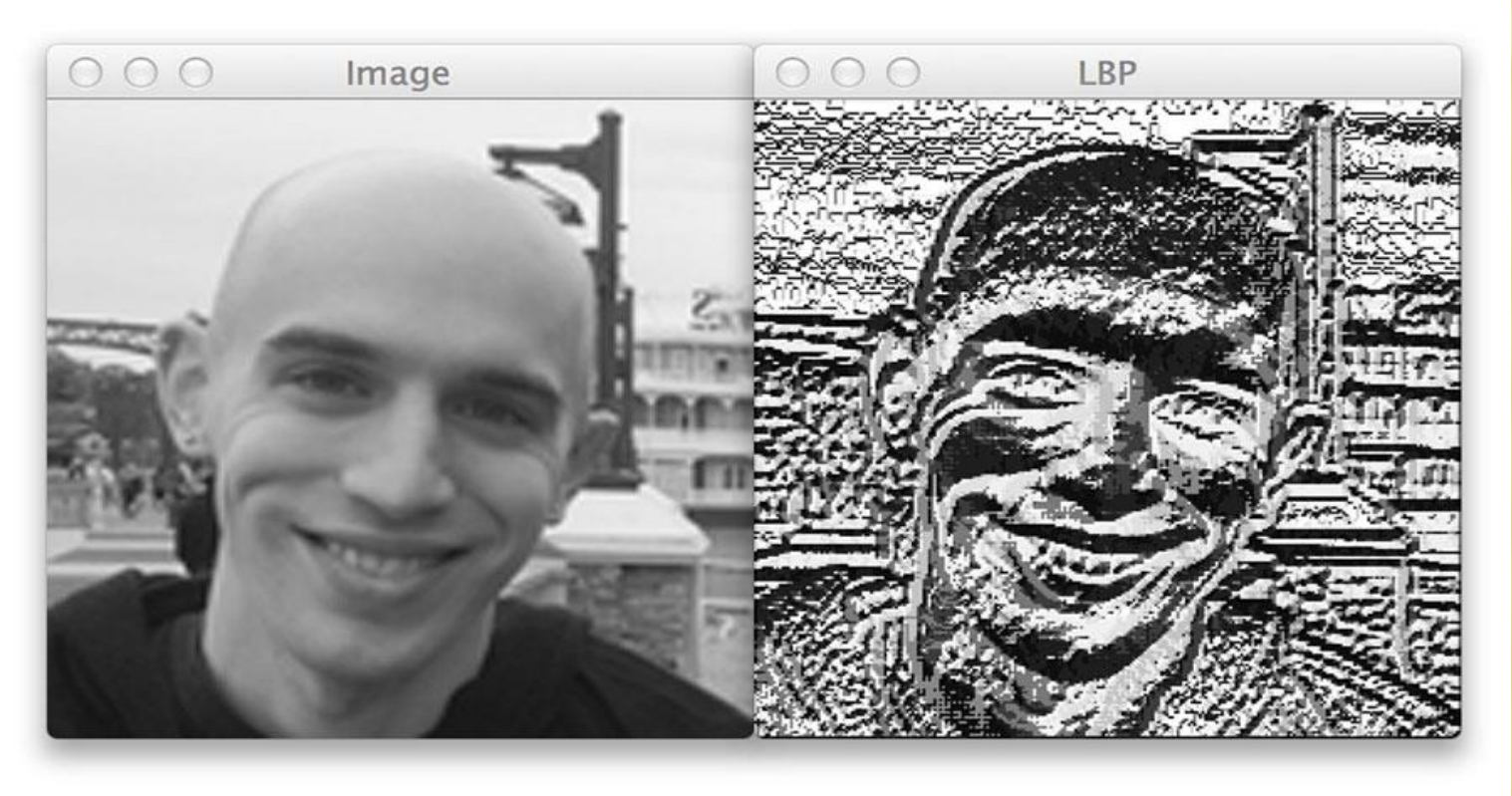


Binary Pattern:	1 (MSB)	1	1	1	0	0	0	1 (LSB)

Code/Weight (2^p):	1×2^7	1×2^6	1×2^5	1×2^4	0×2^3	0×2^2	0×2^1	1×2^0
	= 128	= 64	= 32	= 16	= 0	= 0	= 0	= 1

LBP:	1 + 0 + 0 + 0 + 16 + 32 + 64 + 128 = 241

Here is an example of computing and visualizing a full LBP 2D array:



An example of computing the LBP representation *(right)* from the original input image *(left)*.

http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

```
# import the necessary packages
from skimage import feature
import numpy as np

class LocalBinaryPatterns:
def __init__(self, numPoints, radius):
# store the number of points and radius
self.numPoints = numPoints
self.radius = radius

def describe(self, image, eps=1e-7):
# compute the Local Binary Pattern representation
# of the image, and then use the LBP representation
# to build the histogram of patterns
lbp = feature.local_binary_pattern(image, self.numPoints,
self.radius, method="uniform")
(hist, _) = np.histogram(lbp.ravel(),
bins=np.arange(0, self.numPoints + 3),
range=(0, self.numPoints + 2))

# normalize the histogram
hist = hist.astype("float")
hist /= (hist.sum() + eps)

# return the histogram of Local Binary Patterns
return hist
```

ILLUSTRATIVE APPLICATION

$$LBP(I_C) = \sum_{i=0}^{7} s(I_i - I_C)2^i \qquad s = \begin{cases} 1 & if \quad I_i - I_C > T \\ 0 & if \quad I_i - I_C < T \end{cases}$$

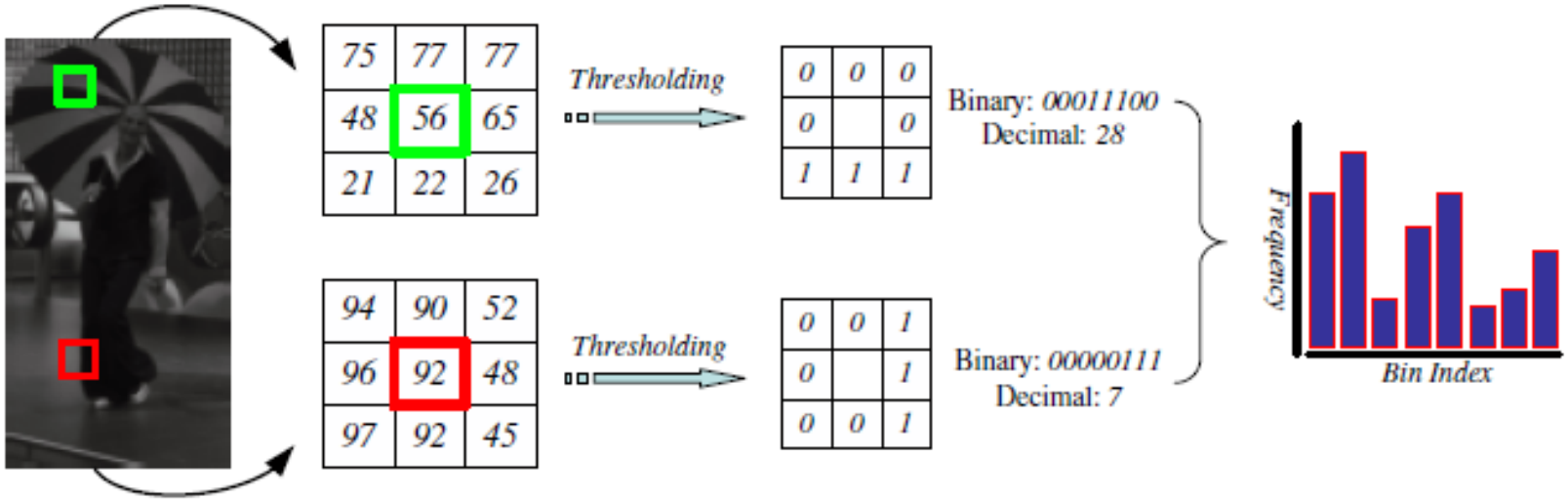


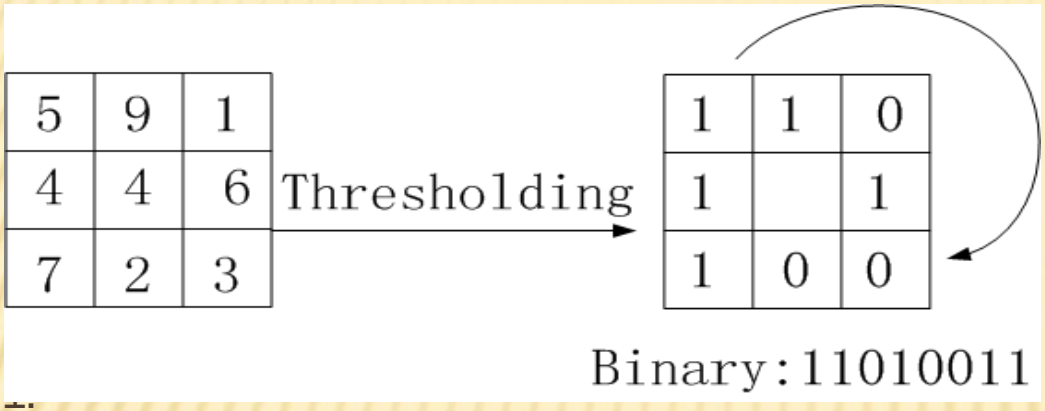
Figure 1. Illustration of LBP. Typically the binary codes obtained by local thresholding are transformed into decimal codes. Note that in this example we use a threshold of 30, which is slightly different from the original LBP. See text for more details.

Chi square statistic similarity measure is usually used for calculate the similarity of two histograms. It is defined as

$$X(p,q) = \sum_{i=0} \frac{(p_i - q_i)^2}{(p_i + q_i)}$$

where p, q is two image descriptors (histogram vectors).

ADVANCED LBP



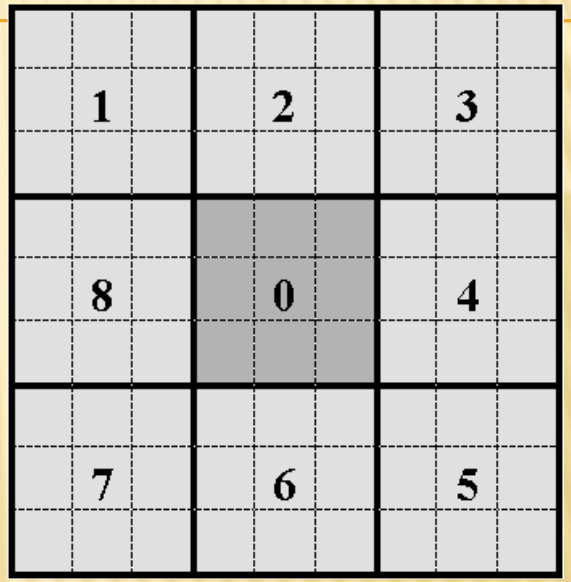


Fig.2.

- (a) The basic LBP operator. (b) The 9×9 MB-LBP operator. In each sub-region,average sum of image intensity is computed. These average sums are then thresholded by that of the center block. MB-LBP is then obtained




MB-LBP filtered images of a face image; (b) filtered by 3 ×3 MB-LBP (c) filtered by 9 × 9 MB-LBP; (d) filtered by 15 × 15 MB-LBP.

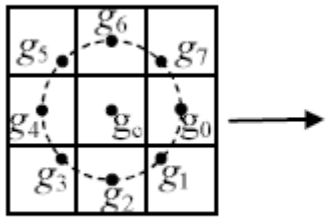
CS-LBP(CENTER-SYMMETRIC LOCAL BINARY PATTERN)

- CS-LBP operator:

$$\text{CS-LBP}_{p,r,t} = \sum_{i=0}^{N/2-1} S(|g_i - g_{i+(N/2)}|)2^i,$$

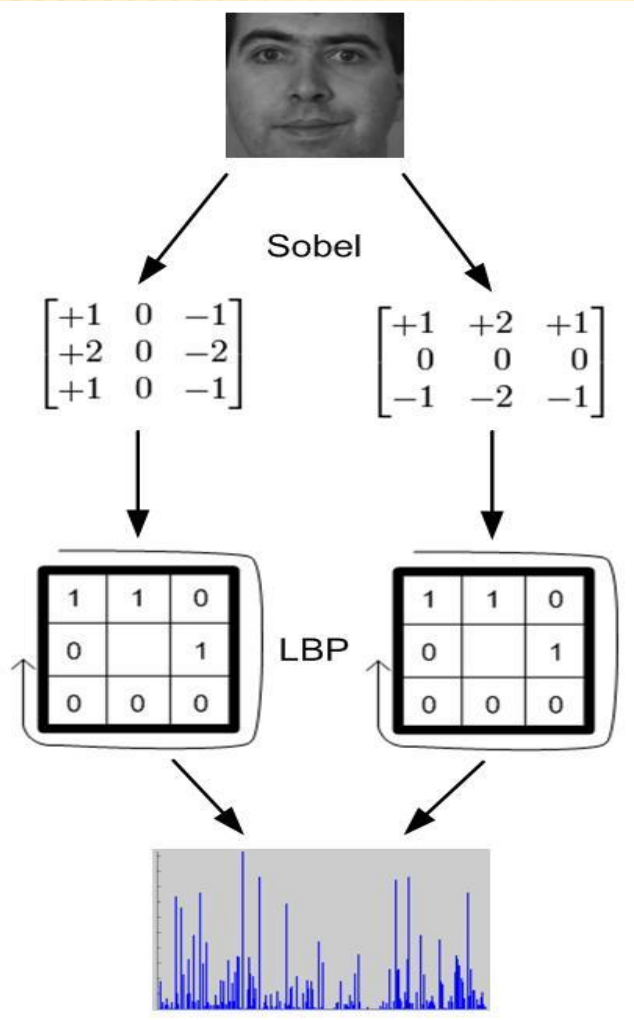
$$S(x) = \begin{cases} 1 & \text{if } x \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

Illustration:



$$\text{CS-LBP}_{8,1} = S(|g_0 - g_4|)2^0 + S(|g_1 - g_5|)2^1 + S(|g_2 - g_6|)2^2 + S(|g_3 - g_7|)2^3$$

Sobel-LBP For Face Representation



Sobel-LBP is an extension of existing Local Binary Pattern (LBP), for object representation and recognition.

The face image is filtered by Sobel operator to enhance the edge information.

Sobel-LBP feature distributions are then extracted and concatenated into a spatial histogram to be used as a face descriptor.

Experimental results indicated that Sobel-LBP provides a significantly better performance than LBP under various conditions

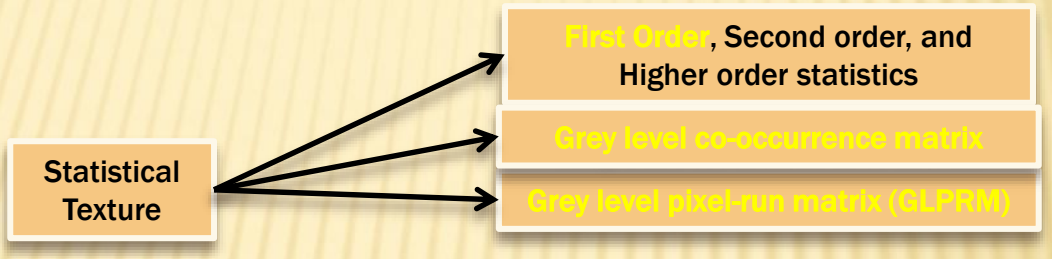
https://maxwell.ict.griffith.edu.au/cvipl/projects.html

First Order Statistics

- Statistical descriptors of grey level distribution – One Pixel
  - Mean Grey Value
  - Deviation of Grey Values
  - Coefficient of Variation
- Generally Too sensitive to factors other than identity of surface
- Problems
  - Does not Model Spatial Relationship
  - Not discriminatory enough

Statistical Texture Descriptors

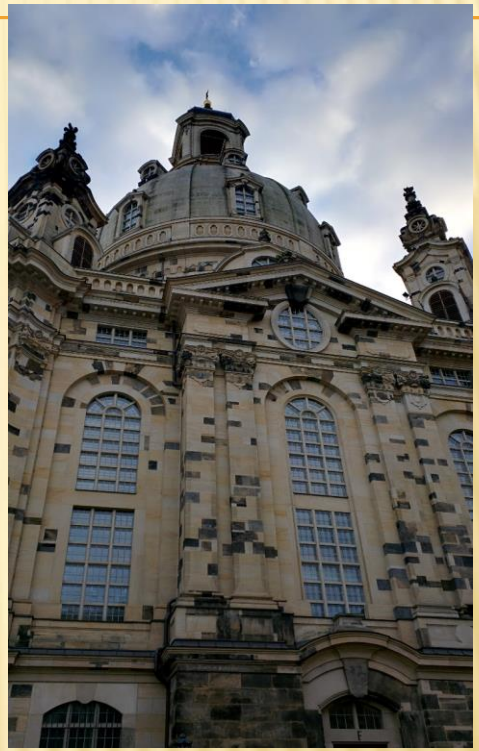
- Better suited to pseudorandom, natural textures



Most of the GLCM texture develop by Robert Haralick and co-authors in the 1970s.

FIRST ORDER STATISTICS

- Histogram of Intensity – L number of intensity levels
  - $p(z_i), i = 0, 1, 2, \ldots, L-1$
- Central Moment of Order n
  - $\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n \, p(z_i)$
- Mean - Average Intensity
  - $\mu = \sum_{i=0}^{L-1} z_i \, p(z_i)$
- Variance – measure of contrast
  - $\sigma^2(z) = \mu_2(z)$
- R – a measure of smoothness
  - $R = 1 - \frac{1}{1+\sigma^2}$
- Skewness – the lopsided nature of a histogram
  - $\gamma_1 = \frac{1}{\sigma^2} \sum_{i=0}^{L-1} (i - \mu)^3 \, p(z_i)$
- Kurtosis – measure of how flat or peaky nature of a histogram
  - $\gamma_2 = \frac{1}{\sigma^4} \sum_{i=0}^{L-1} (i - \mu)^3 \, p(z_i) - \varepsilon$



Note that: Variance measures region "roughness“
R is close to 0 for homogenous areas
R tends to 1 as σ2, ”roughness”, increase
order statistics does not describe geometry or context

2D CASE:

- Energy

$$e_i = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} I_i^2(x,y)$$

Entropy

$$Entropy_i = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} I_i(x,y) \log I_i(x,y)$$

- Kurtosis

$$k = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} [I_i(x,y) - \mu]^4}{M \times N \times \sigma^4} - 3$$

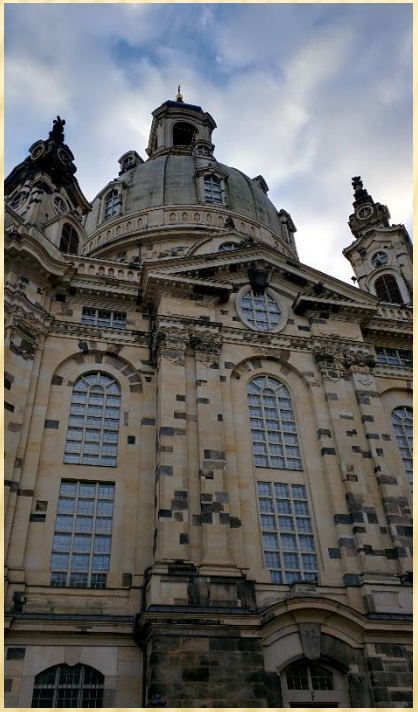
- Skew

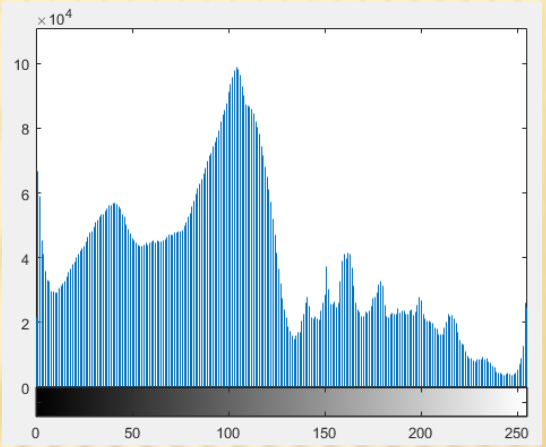
$$Skew = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} [I_i(x,y) - \mu]^3}{M \times N \times \sigma^3}$$

- Variance

$$\sigma^2 = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} [I_i(x,y) - \mu]^2}{M \times N}$$

First Order Statistic Example

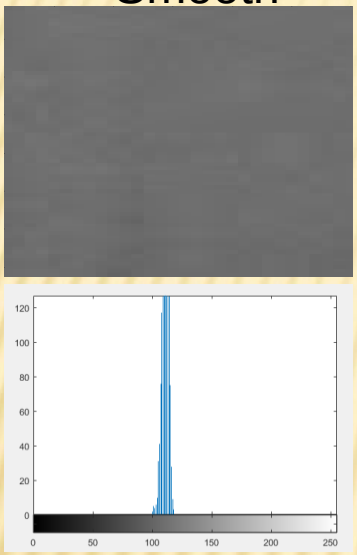




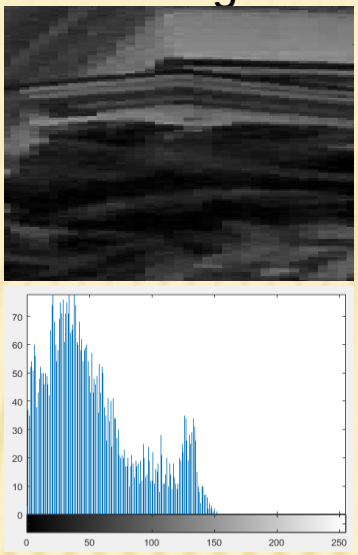
Statistic	Value
Average	100.27
Deviation	59
Smoothness	.000287
Skewness	.3222
Kurtosis	2.3913
Entropy	7.7307

First Order Statistics Example

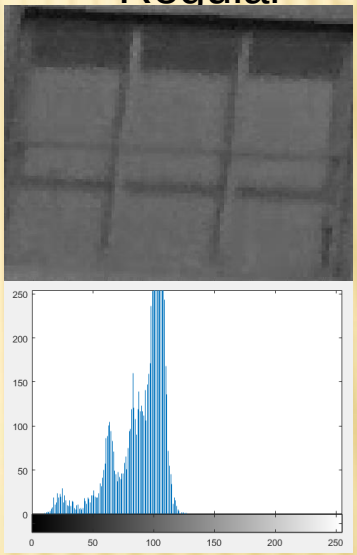
Smooth



Rough



Regular



Statistic	Overall	Smooth	Rough	Regular
Average	100.27	110.9	52.88	91.17
Deviation	59	2.376	37.23	20.17
Smoothness	.000287	.1504	.00072	.0025
Skewness	.3222	-0.0035	0.0110	-0.0665
Kurtosis	2.3913	.0404	0.0427	.2327
Entropy	7.7307	3.233	6.9265	5.767

GREY LEVEL CO-OCCURRENCE MATRICES (GLCM)

- Measures of texture computed using histograms suffer from the limitation that they carry no information regarding the relative position of the pixels with respect to each other.
- One way to bring this type of information into the texture analysis process is to consider not only the distribution of the intensities but also the positions of pixels with equal or nearly equal intensity values.

- The idea behind GLCM is to describe the texture as a matrix of "pair gray level probabilities".

Limitation:

- The texture filter functions, described in Texture Analysis cannot provide information about shape, that is, the spatial relationships of pixels in an image

GREY LEVEL CO-OCCURRENCE MATRICES (GLCM)

- It is a matrix of *frequencies* at which → two pixels, separated by a certain vector, occur in the image.
- A matrix of frequencies at which two pixels, separated by a certain vector occur in the image

$$P_d(i,j) = \sum_{p=1}^{N}\sum_{q=1}^{M}\begin{cases}1, if\ I(p,q)=i,\ \ I(p+\Delta x, q+\Delta y)=j\\ 0, otherwise\end{cases}$$

Where i,j are pixel values ((i,j)value of the co-occurrence matrix gives the number of times in the image that the i-th and j-th pixel values occur in the relation given by the offset), p,q are the spatial position in the image I,

$(\Delta x, \Delta y)$ - separation vector (define the spatial relation for which this matrix is calculated, for example (1,2) could indicate "one down, two right)

$I(p,q)$ - intensity of a pixel at (p,q)

Note: for the same image with different displacement vector , it will yield different gray-level co-occurrence matrix, which characterizes the texture homogeneity of different special distribution and orientations.

A 2D histogram that preserves intensity and spatial information

COMPUTATION OF CO-OCCURRENCE MATRIX

- Co-occurrence matrices can also be parameterized in terms of a distance, d, and an angle, θ , instead of an offset $(\Delta x, \Delta y)$
- d → Relative **distance** between the pixel pair (measured in pixel number. e.g., 1, 2, …)

θ → Relative **orientation** / rotational angle. (e.g., 0º, 45º, 90º, 135º, …)

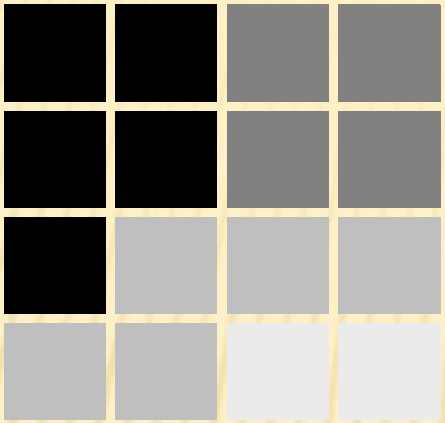
Computation Of Co-occurrence Matrix

GLCM TEXTURE: A TUTORIAL
V. 3.0 March 2017 Replaces V. 2.8
Of August, 2005 V. 3.0 Incorporates
All Corrections Up To V. 2.8.

Mryka Hall-beyer, Ph.D. Department
Of Geography University Of Calgary
Calgary, Alberta T2N 1N4 Canada

Test Image matrix

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3



Find the **number of co-occurrences** of pixel *i* to the neighboring pixel value *j*

In the illustration below, the neighbor pixel is chosen to be the one to the east (right) of each reference pixel. This can also be expressed as a (1,0) relation:

1 pixel in the x direction, 0 pixels in the y direction.

Each pixel within the window becomes the reference pixel in turn, starting in the upper left corner and proceeding to the lower right.

Note: Pixels along the right edge have no right hand neighbor, so they are not used for this count.

Computation of Co-occurrence Matrix

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Test Image matrix

i/j	0	1	2	3
0	(0,0)	(0,1)	(0,2)	(0,3)
1	(1,0)	1,1)	(1,2)	(1,3)
2	(2,0)	(2,1)	(2,2)	(2,3)
3	(3,0)	(3,1)	(3,2)	(3,3)

i/j	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

The top left cell will be filled with the number of times the combination 0,0 occurs, i.e. how many times within the image area a pixel with grey level 0 (neighbor pixel) falls to the right of another pixel with grey level 0 (reference pixel). Each cell is read in this pattern with appropriate changes in numbers.

Illustrative Example Co-occurrence Matrices

- Distance =2
- Angle = 0°

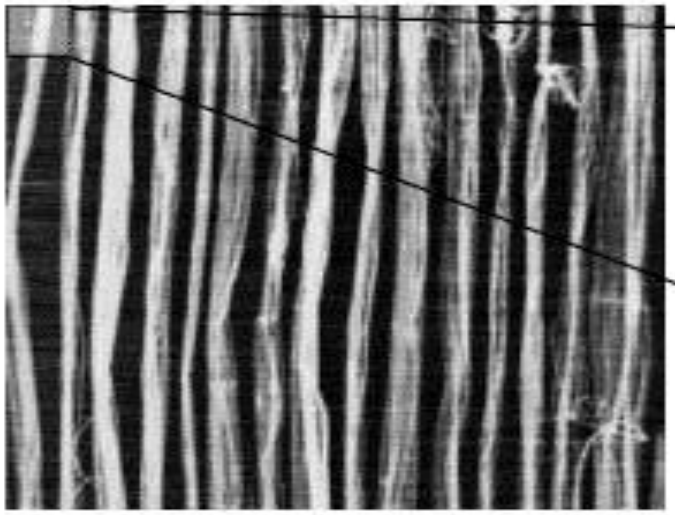
- Original Image

1	6	3	4	2
5	6	5	1	5
2	4	4	3	5
4	3	6	2	2
1	3	2	1	1

- Co-occurrence Matrix

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	0	1	0	0
3	1	2	0	0	0	0
4	0	0	1	0	0	1
5	1	1	0	1	0	0
6	1	1	0	1	0	0

Grey Level Co-occurrence Matrix Example



original image

image pixel values

211	191	71	53	48	44	43	44
212	191	76	60	52	52	53	52
212	180	75	75	75	76	76	75
212	140	53	46	49	49	46	44
211	115	53	44	44	43	43	46
211	115	54	44	44	46	37	43
211	123	55	52	52	52	44	44
211	141	76	74	75	75	71	70

quantisation 4 levels

3	2	1	0	0	0	0	0
3	2	1	0	0	0	0	0
3	2	1	1	1	1	1	1
3	2	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
3	2	1	1	1	1	1	1

quantised image

vector – length 1 pixel, direction horizontal

28	0	0	0
5	10	0	0
1	4	0	0
0	3	5	0

co-occurrence matrix

Summary of Steps in the Image

1.Pick up a window size

2. Place window in first position over top left of the image.

3. For the pixels within this window in this position, count the pixels, transform and normalize the GLCM

4. Calculate the texture measure of your choice (see below)

5. Move the window over one pixel, and repeat steps 3 and 4.

6. Continue covering all possible window positions until the texture image is complete.

Expressing the GLCM as a Probability

- Is it more likely to find a horizontal combination of, say, 2,2 in the original image, or is 2,3 more likely? Looking at the horizontal GLCM shows that the combination 2,2 occurs 6 times out of the 24 horizontal combinations of pixels in the image (12 eastern + 12 western). In other words, 6 is the entry in the horizontal GLCM in the third column (reference pixel value 2) and third row (neighbour pixel value 2). The simplest definition of the probability of a given outcome is
- "the number of times this outcome occurs, divided by the total number of possible outcomes."
- Using this definition, we can calculate: the combination (2,2) occurs in 6 cells out of 24, for a probability of 6/24 = 1/4 or 0.250. The probability of (2,3) is 1/24 or .042.

GREY LEVEL CO-OCCURRENCE MATRIX EXAMPLE

- d = (1,0)

Quantized Image				
1	1	5	6	8
2	3	5	7	1
4	5	7	1	2
8	5	1	2	5

Grey Level Co-Occurrence Matrix								
	0	2	1	0	0	0	0	0
	0	0	0	1	1	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	2	0	1	0
	0	0	0	0	0	0	1	0
	2	0	0	0	0	0	0	0
	1	0	0	1	0	0	0	0

GREY LEVEL CO-OCCURRENCE MATRIX EXAMPLE

- d = (0,1)

Quantized Image				
1	1	5	6	8
2	3	5	7	1
4	5	7	1	2
8	5	1	2	5

Grey Level Co-Occurrence Matrix								
	1	2	0	0	1	0	0	0
	0	0	1	0	1	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	0	0
	1	0	0	0	0	1	2	0
	0	0	0	0	0	0	0	1
	2	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0

GREY LEVEL CO-OCCURRENCE MATRIX HANDS ON EXAMPLE

- d = (1,1)

Quantized Image				
1	1	5	6	8
2	3	5	7	1
4	5	7	1	2
8	5	1	2	5

Grey Level Co-Occurrence Matrix								

GREY LEVEL CO-OCCURRENCE MATRIX HANDS ON EXAMPLE ANSWERS

- d = (1,1)

Quantized Image				
1	1	5	6	8
2	3	5	7	1
4	5	7	1	2
8	5	1	2	5

Grey Level Co-Occurrence Matrix								
	0	0	1	0	2	0	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	1	0
	0	0	0	0	1	0	0	0
	2	0	0	0	0	0	1	0
	1	0	0	0	0	0	0	0
	0	2	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

Properties Of The GLC Matrix

1. It is square: The reference pixels have the same range of possible values as the neighbor pixels, so the values along the top are identical to the values along the side.

2. It has the same number of rows and columns as the quantization level of the image:

- The test image is 2-bit has four (22) grey level values (0,1,2 and 3).
- Eight bit data has 256 possible values (28), so would yield a 256 x 256 square matrix, with 65,536 cells.
- 16 bit data would give a matrix of size 65536 x 65536 = 429,496,720 cells!

Symmetrical Around the Diagonal GLCM

How to make the matrix symmetrical ?

- The transpose of the eastern (1,0) matrix:

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

2	0	0	0
2	2	0	0
1	0	3	0
0	0	1	1

Add the original matrix to its transpose

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

+

2	0	0	0
2	2	0	0
1	0	3	0
0	0	1	1

=

4	2	1	0
2	4	0	0
1	0	6	0
0	0	1	2

Grey Level Co-occurrence Matrix Normalized

- Normalized by dividing each entry by the total number of pixel pairs
- $N(i,j) = \frac{P_d(i,j)}{\sum_i \sum_j P_d(i,j)}$
- Normalizes the co-occurrence values between 0 and 1. At this point the values can be considered probabilities.

Normalized GLCM

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

:

4/24	2/24	1/24	0/24
2/24	4/24	0/24	0/24
1/24	0/24	6/24	1/24
0/24	0/24	1/24	2/24

=

.16	.08	.04	0
.08	.16	0	0
.04	0	.25	.04
0	0	1	2

Summary of steps in creating a symmetrical normalized GLCM:

- 1. Create a framework matrix taking into account the bit depth
- 2. Decide on the spatial relation between the reference and neighbor pixel
- 3. Count the occurrences and fill in the framework matrix
- 4. Add the matrix to its transpose to make it symmetrical
- 5. Normalize the matrix to conceptually turn it into probabilities

Grey Level Co-occurrence Matrix Limitations

- Too many parameters
- Computationally expensive
- Not suitable for coarse textures
- Susceptible to noise

Haralick Texture Features

- Measures involving multiple pixels
  - Joint Difference Histogram – difference between adjacent pixels
- Haralick suggested a set of 14 textural features that can be calculated from the co-occurrence matrix that contain image textural characteristics
- Haralick texture features describe the correlation in intensity of pixels that are next to each other in space
- Temporal texture features describe the correlation in intensity pixel in the same position in images next to each other over time

Descriptive Statistics Of The GLCM Texture Measures

Below texture measures uses equations similar to those for common descriptive statistics, such as mean or standard deviation (or variance). However, all are calculated using the entries in the GLCM, not the original pixel values

The GLCM Mean is not simply the average of all the original pixel values in the image window.

The pixel value is weighted not by its frequency of occurrence by itself (as in a "regular" or familiar mean equation) but by its frequency of its occurrence in combination with a certain neighbor pixel value.

+

$$\mu_i = \sum_{i=0}^{N-1} iP_{i,j}, \quad \mu_j = \sum_{i=0}^{N-1} jP_{i,j}$$

The GLCM Mean for the horizontal GLCM is different from that for the vertical GLCM because the combinations of pixels are different in the two cases.

For the symmetrical GLCM, where each pixel in the window is counted once as a reference and once as a neighbor, the two values are identical.

Descriptive Statistics Of The GLCM Texture Measures

$$\mu_i = \sum_{i=0}^{N-1} iP_{i,j}, \quad \mu_j = \sum_{i=0}^{N-1} jP_{i,j}$$

Note:

- For the symmetrical GLCM, where each pixel in the window is counted once as a reference and once as a neighbor, the two values are identical.

- The summation is from 0 to (N-1), not from 1 to N. Since the first cell in the upper left of the GLCM is numbered (0,0), not (1,1), the i value (0) of this cell is the same as the value of the reference pixel (0)

- The Pij value is the probability value from the GLCM, i.e. how many times that reference value occurs in a specific combination with a neighbour pixel. It is not a measure of how many times the reference pixel occurs, period, which would be the "regular" first-order mean for the original window. First-order mean is a tool for smoothing an image to remove random noise, and also of systematically degrading the spatial resolution. •

HARALICK TEXTURE FEATURES OVERVIEW

- Calculated from Grey Level Co-Occurrence Matrices:
  - Energy – Number of repeated pairs
  - Entropy – Measure of chaos
  - Contrast – local variance of an image
  - Correlation – Measure of chaos
  - Maximum Probability – Uses the maximum value of a matrix
  - Homogeneity – The similarity of surrounding pixels

Co-occurrence Matrices

- Features based on co-occurrence matrix

$$\text{Energy/Uniform} = \sum_i \sum_j N_d^2[i, j]$$

$$\text{Entropy} = -\sum_i \sum_j N_d[i, j] \log_2 N_d[i, j]$$

$$\text{Constrast} = \sum_i \sum_j (i-j)^2 N_d[i, j]$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{N_d[i, j]}{1+|i-j|}$$

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

where μ_i, μ_j are the mean and σ_i, σ_i are the standard deviations of the row and column sums

Haralick Texture Features Windowing

- Algorithms for texture analysis are applied to an image in a series of window sizes *w*, each centered on a pixel (i,j)
- The value of the resulting statistical measure are assigned to the same position (i,j) in the new image

Haralick Texture Features Energy

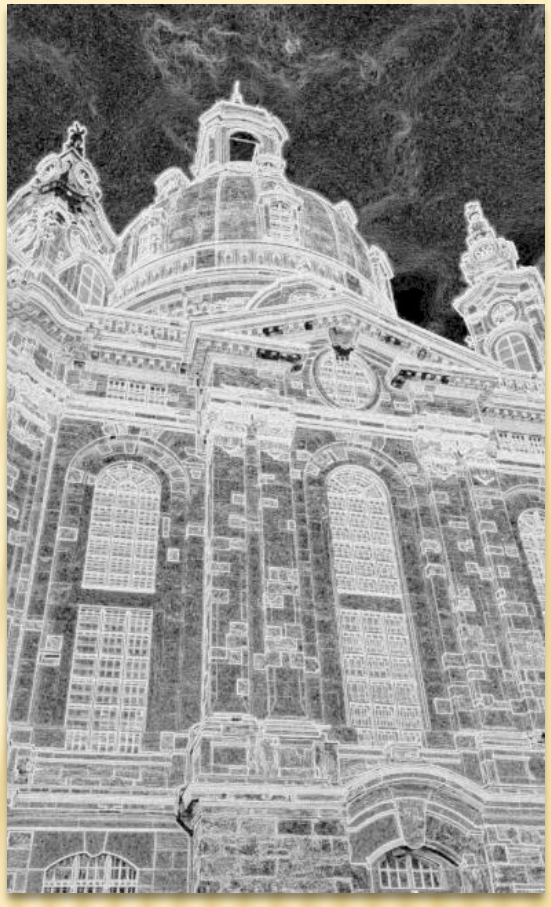
- Use local kernels to detect texture type
- Is minimal with all elements are equal
- Texture Energy Measure is produced after convolution with specific kernels:
  - $L_e = \sum_{i=1}^{m} \sum_{i=1}^{n} |P_d(i,j)|$
- If *n* kernels are applied, the result is an *n*-dimensional feature vector at each pixel of the region

Haralick Texture Features: Entropy

- **Measures randomness of intensity distribution**

$$C_e = -\sum_i \sum_j P_d(i,j) \ln P_d(i,j$$

- **Entropy is highest when all entries in $P_d(i,j)$ are of similar magnitude, and small when the entries in $P_d(i,j)$ are unequal.**



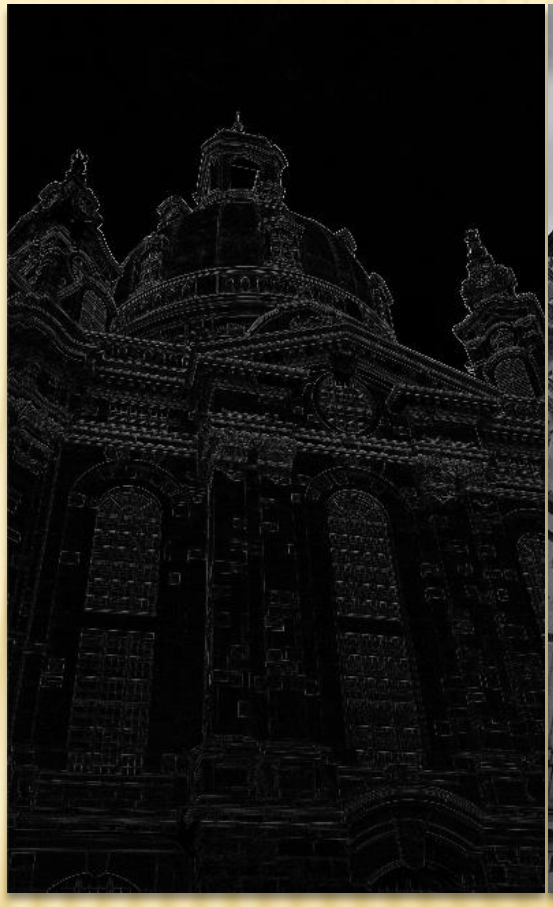
Since ln(0) is undefined, assume that 0 * ln(0) = 0:

Haralick Texture Features
Contrast

- Measure of the local variations present in an image
  - $C(k,n) =$

$$\sum_j \sum_i (i-j)^k P_d(i,j)^n$$

  - Typically k=2, n=1
- If there is a large amount of variation in an image the $P_d(i,j)$ will be concentrated away from the main diagonal and contrast will



HARALICK TEXTURE FEATURES CORRELATION

- **Measure of image linearity**
  - $C_c = \frac{\sum_i \sum_j [ijP_d(i,j)] - \mu_i \mu_j}{\sigma_i \sigma_j}$
- Correlation will be high if an image contains a considerable amount of linear structure

- **A high Correlation texture means high predictability of pixel relationships.**
- **Pixels are usually more highly correlated with pixels nearby than with more distant pixels (Spatial Autocorrelation).**
- **So, smaller window sizes will usually have a higher Correlation value than larger windows**

HARALICK TEXTURE FEATURES MAXIMUM PROBABILITY

- The largest entry in the matrix, which corresponds to the strongest response. This could be the maximum in any matrices or the maximum overall
  - $C_m = max[P_d(i,j)]$

HARALICK TEXTURE FEATURES HOMOGENEITY

- A measure of the range of grey levels
  - $C_h = \sum_i \sum_j \frac{P_d(i,j)}{1+|i-j|}$
- A homogeneous scene will contain only a few grey levels

GLCM FEATURE IMAGE EXAMPLES, W= 15

GLCM contrast is negative correlated with IDM

GLCM entropy

GLCM variance

GLCM IDM

GLCM ASM

GLCM correlation

Haralick Texture Features Computation

1) *Quantize the image data*: quantize a given image into a specified number of discrete gray levels, known as **Quantization**.
2. *Create the GLCM:* It will be a square matrix N x N in size where N is the **Number of levels** specified under **Quantization.**

Steps for matrix creation are:

a) Let s be the sample under consideration for the calculation.
b) Let W be the set of samples surrounding sample s which fall within a window centered upon sample s of the size specified under Window Size.

3. Create the GLCM: It will be a square matrix N x N in size where N is the **Number of levels** specified under **Quantization.**

Steps for matrix creation are:

a) Let s be the sample under consideration for the calculation.
c) Let W be the set of samples surrounding sample s which fall within a window centered upon sample s of the size specified under Window Size.
d) Define each element i, j of the GLCM of sample present in set W, as the number of times two samples of intensities i and j occur in specified **Spatial relationship**.
e) The sum of all the elements i, j of the GLCM will be the total number of times the specified spatial relationship occurs in W.
e) Make the GLCM symmetric:
   i. Make a transposed copy of the GLCM.
   ii. Add this copy to the GLCM itself.
f) Normalize the GLCM:
   - Divide each element by the sum of all elements.
   - The elements of the GLCM may now be considered probabilities of finding the relationship *i, j (or j, i)* in W.

3) *Calculate the selected **Feature***. This calculation uses only the values in the GLCM. For e.g.

- Energy,
- Entropy,
- Contrast,
- Homogeneity,
- Correlation,
- Shade or
- Prominence

4) The sample s in the resulting image is replaced by the value of this calculated feature.

Summary

- Texture is a useful property that is often indicative of materials, appearance cues
- **Texture representations** attempt to summarize repeating patterns of local structure
- **Filter banks** useful to measure redundant variety of structures in local neighborhood
  - Feature spaces can be multi-dimensional
- Neighborhood statistics can be exploited to "sample" or **synthesize** new texture regions
  - Example-based technique

Further Reading

- A good tutorial on texture:http://www.fp.ucalgary.ca/mhallbey/tutorial.htm
- Trygve Randen, and John Håkon Husøy, Filtering for Texture Classification:A Comparative Study, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21, NO. 4, APRIL 1999
- Selvan, S. and S. Ramakrishnan, "SVD-based modeling for image texture classification using wavelet transformation," IEEE Trans. Image Proc. **16**(11), 2688–2696 (2007
- Mellor, M., B.-W. Hong and M. Brady, "Locally rotation, contrast, and scale invariant descriptors for texture analysis," IEEE Trans. Pattern Anal. Mach. In tell. **30**(1), 52–61 (2008).
- Liu, G.-H. and J.-Y. Yang, "Image retrieval based on the texton co-occurrence matrix," Pattern Recog. Lett. **41**(12), 3521–3527 (2008).
- Liu, G. and Y. Yu, "Radon representation-based feature descriptor for texture classification," IEEE Trans. Image Process. **18**(5), 921–928 (2009).
- Choi, W.-P.et al., "Simplified Gabor wavelets for human face recognition," Pattern Recogn. **41**(3), 1186–1199 (2008).
- Zheng, C., D.-W. Sun and L. Zheng, "Recent applications of image texture for evaluation of food qualities-a review," Trends Food Sci. Tech. **17**(3), 113–128 (2006).

THANK YOU

QUESTIONS

ANSWERS

Appendix 1: Alternative Formulation

- The fractal dimension is usually fractional and increases with the object complexity (i.e. complex fractals reveal more measurable detail when magnified than less complex ones) and thus it is a formal means of quantifying shape complexity.
- An alternative way of thinking about the fractal dimension is as a measure of space filling: complex objects fill more space than simpler ones, bearing in mind that fractals are not just complex, but complex at many scales.

is computed using an alternative formulation

G. Landini , **Fractals in microscopy**

Authors

1.Oral Pathology Unit, School of Dentistry, College of Medical and Dental Sciences, University of Birmingham, Birmingham, U.K.

Appendix 1: Box-counting Method- Alternative Formulation

- Thus, to calculate accurate for asymmetrical neuronal images, we propose another modification of the BC method: *each image should be analyzed for symmetry*. The axis of rotation should be constructed and the image should be rotated by four angles ($45° + k\pi/2$, where $k = 0, 1, 2,$ and 3). In each position, apparent should be recorded and final (or precise) will be the mean of these values. Figures 3(c) and 3(d) illustrate this procedure for type 1 and type 2 asymmetrical neurons from the monkey dentate nucleus.

Box-Counting Method of 2D Neuronal Image: Method Modification and Quantitative Analysis Demonstrated on Images from the Monkey and Human Brain, Computational and Mathematical Methods in Medicine, Volume 2017 (2017), Article ID 8967902, 9 pages

Show how images at different sizes, resolutions, and rotation angles could influence in the magnitude of the box dimension

The Morphology of the Brain Neurons: Box-Counting Method in Quantitative Analysis of 2D Image

Lung cancer—a fractal viewpoint (*Nature Reviews Clinical Oncology* **12**, 664–675 (2015)**)**

Appendix 2: ADVANCED LBP: LOCAL MIN-MAX BINARY PATTERN

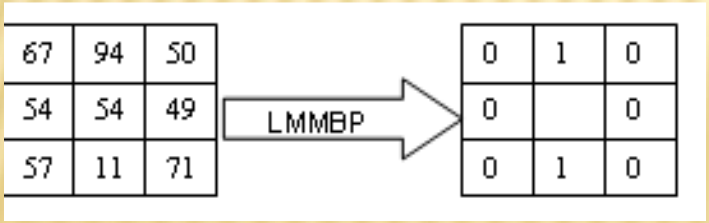
Let g_c, g_1, g_2…,g_8 be the pixel values of a 3*3local region where g_c is the value of the central pixel and g_1, g_2, …, g_8 are the pixel values of its 8 neighborhood.

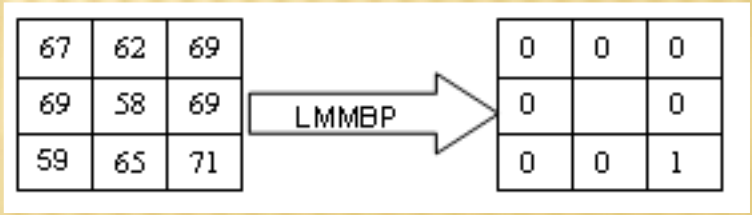
The calculation of LMin-MaxBP is described as; Find Min_j and Max_j from the eight neighbor positions. Where Min_j is the first index of least minimum or equal gray value of the neighbors g_i compared to g_c. Max_j is the first index of highest maximum or equal gray value of the neighbor's g_i compared to g_c.

The Local Min-Max Binary Pattern is described as

$$LBP(I_C) = \sum_{i=0}^{7} MinMax(i_c, i_i)2^i$$

$$MinMax(u,v) = 1 \quad if \quad \begin{cases} \text{v is the first occurred least minimum or equal to u} \\ \text{among the gray values } \leq \text{u; or} \\ \text{v is the first occurred highest maximum or equal to u} \\ \text{among the gray values } \geq \text{u} \end{cases}$$

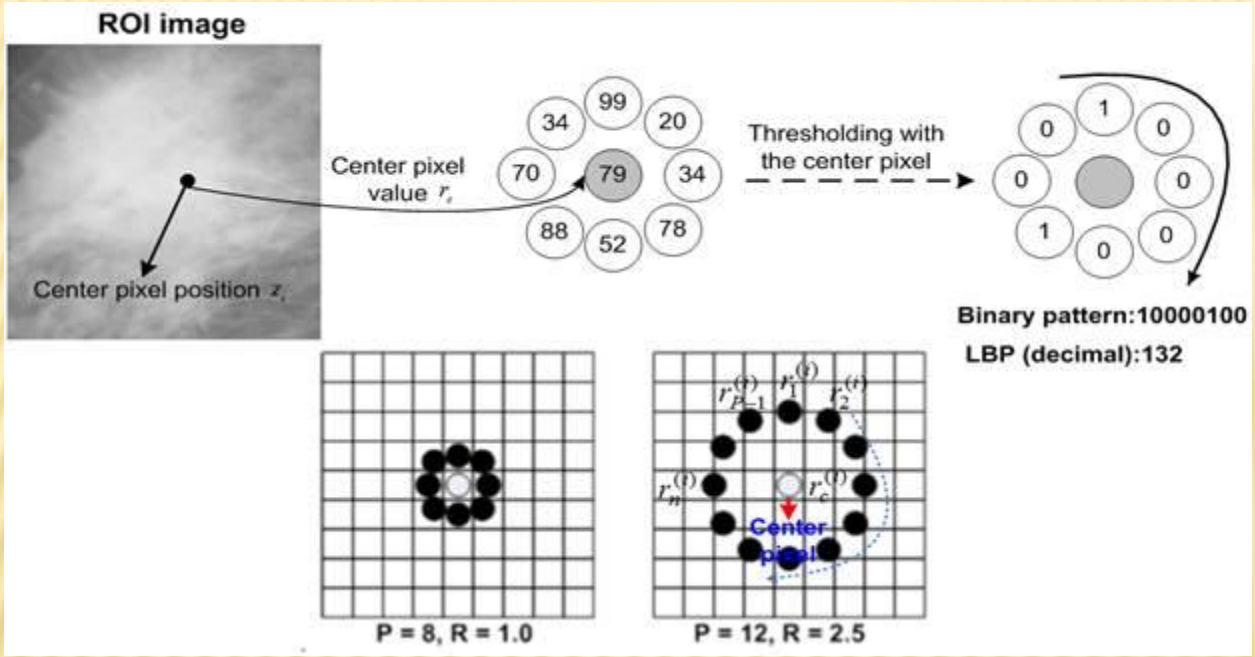




LMin-MaxBP: 01000100; Weight: 34

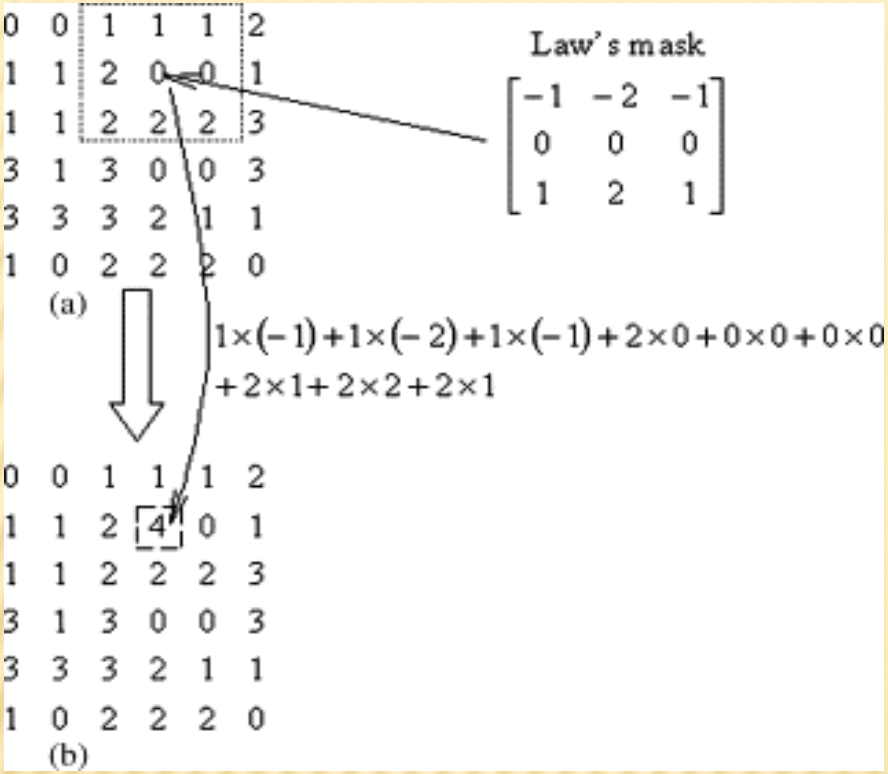
LMin-MaxBP: 01000100; Weight: 16

Appendix 3: Breast Cancer Screening Using Mammography Using **LBP Texture Feature**



"Multiresolution Local Binary Pattern texture analysis for false positive reduction in computerized detection of breast masses on mammograms," Jae Young Choi, Dae Hoe Kim, Seon Hyeong Choi, and Yong Man Ro, SPIE Medical Imaging, February 2012, San Diego, California (USA)

APPLYING ONE OF LAW'S MASK TO AN IMAGE



These two masks are also called the Sobel operator

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The size parameters of the mask in Eq.

$$F(x,y)=\sum_{k=-m}^{m}\sum_{l=-n}^{n}M(K,L)I(x+k,Y+l)$$

m and *n* are both 3. (a) Initial image; (b) after applying the mask on one pixel in (a).

Appendix 4: Creation Of 2D Law's Texture Masks

- Creation of 2D Masks: From these one-dimensional convolution kernels, we can generate 25 different two-dimensional convolution kernels by convolving a vertical 1-D kernel with a horizontal 1-D kernel. As an example, the L5E5 kernel is found by convolving a vertical L5 kernel with a horizontal E5 kernel. Of the 25 two-dimensional convolution kernels that we can generate from the one-dimensional kernels above, 24 of them are zero-sum; the L5L5 kernel is not. A listing of all 5x5 kernel names is given below:
- L5L5 E5L5 S5L5 W5L5 R5L5 L5E5 E5E5 S5E5 W5E5 R5E5 L5S5 E5S5 S5S5 W5S5 R5S5 L5W5 E5W5 S5W5 W5W5 R5W5 L5R5 E5R5 S5R5 W5R5 R5R5

- Example:

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

E5

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

LAW'S TEXTURE ENERGY MEASURES

Another example :

$$L_5^T \times S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

9D feature vector for pixel : Subtract mean neighborhood intensity from pixel
Dot product 16 5x5 masks with neighborhood . 9 features defined as follows

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

Law's texture energy measures are easy to apply and give good results for most texture types. However, co-occurrence matrices are more flexible; for example, they can be scaled to account for coarse-grained textures.

LAW'S TEXTURE ENERGY MEASURES

$$L_5^T \times E_5 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

$$L5 = [1 \quad 4 \quad 6 \quad 4 \quad 1]$$

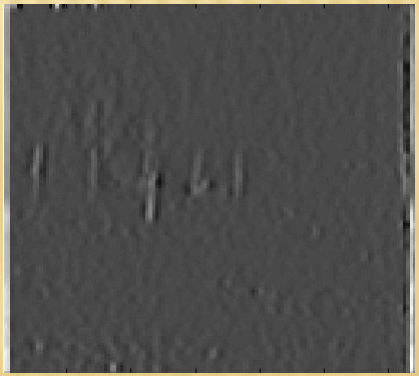
$$E5 = [-1 \quad -2 \quad 0 \quad 2 \quad 1]$$

Input Image

$L_5^T \times E_5$

Output Image

$L_5^T \times S_5$



Output Image

$$L_5^T \times S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

$$S5 = [-1 \quad 0 \quad 2 \quad 0 \quad -1]$$

LAW'S TEXTURE ENERGY MEASURES

$$L_5^T \times E_5 = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -4 & -8 & 0 & 8 & 4 \\ -6 & -12 & 0 & 12 & 6 \\ -4 & -8 & 0 & 8 & 4 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

$$L5 = [1 \quad 4 \quad 6 \quad 4 \quad 1]$$

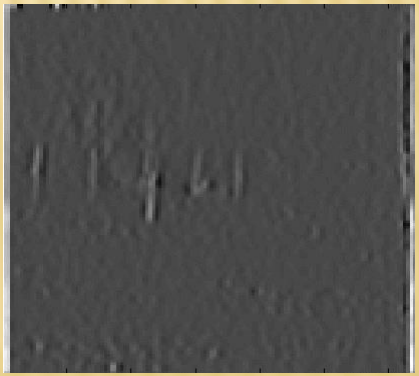
$$E5 = [-1 \quad -2 \quad 0 \quad 2 \quad 1]$$

Input Image

$L_5^T \times E_5$

Output Image

$L_5^T \times S_5$

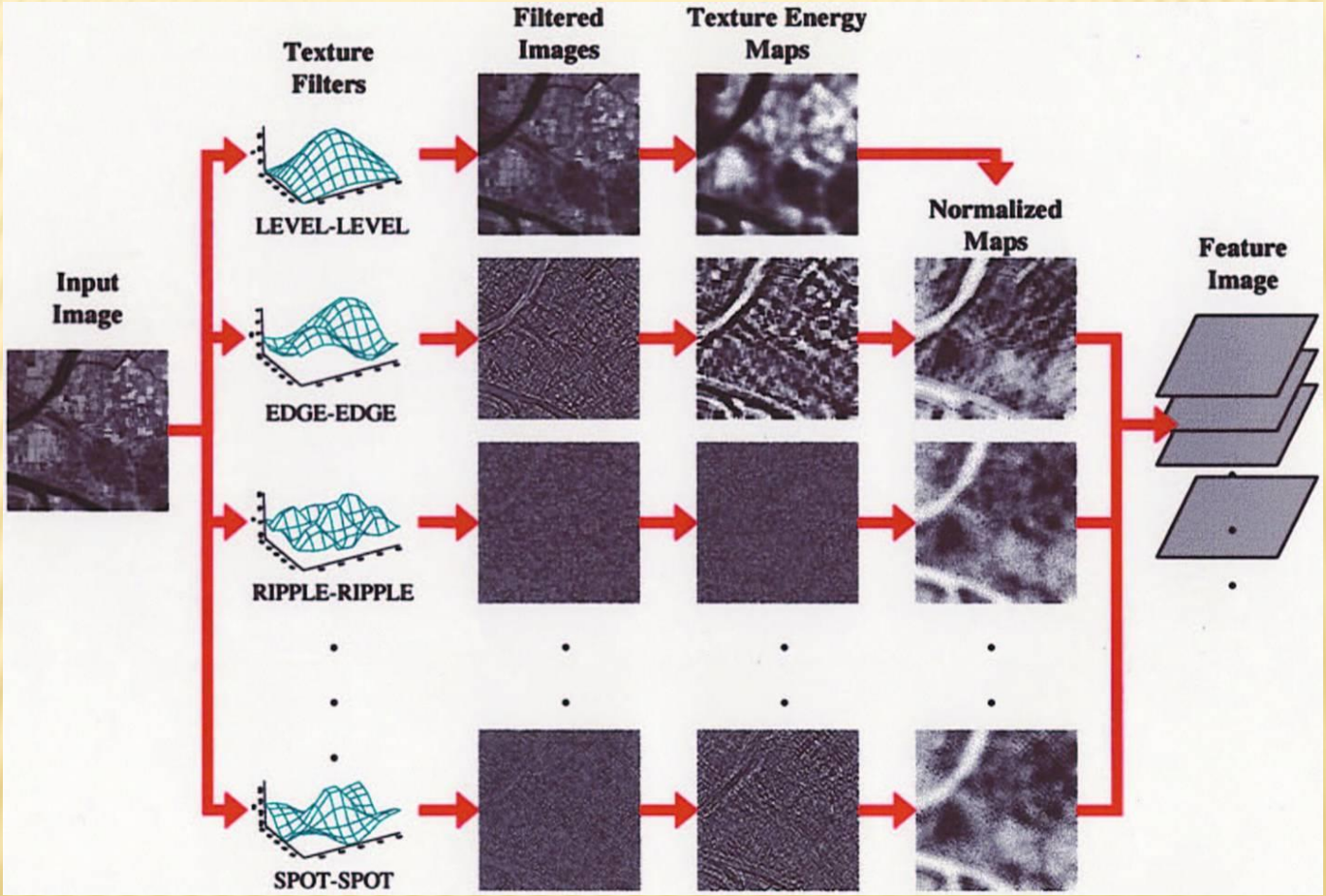


Output Image

$$L_5^T \times S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

$$S5 = [-1 \quad 0 \quad 2 \quad 0 \quad -1]$$

LAW'S PROCESS



```
        //从上到下插值放大并相加，即得blend图像结果
        Mat currentImg = resultHighestLevel;
        for (int l=levels-1; l>=0; l--) {
            Mat up;

            pyrUp(currentImg, up, resultLapPyr[l].size());
            currentImg = up + resultLapPyr[l];
        }
        return currentImg;
    }

    void blendLapPyrs() {
        //获得每层金字塔中直接用左右两图Laplacian变换拼成的图像resultLapPyr
        resultHighestLevel = leftHighestLevel.mul(maskGaussianPyramid.back()) +
            rightHighestLevel.mul(Scalar(1.0,1.0,1.0) - maskGaussianPyramid.back());
        for (int l=0; l<levels; l++) {
            Mat A = leftLapPyr[l].mul(maskGaussianPyramid[l]);
            Mat antiMask = Scalar(1.0,1.0,1.0) - maskGaussianPyramid[l];
            Mat B = rightLapPyr[l].mul(antiMask);
            Mat_<Vec3f> blendedLevel = A + B;

            resultLapPyr.push_back(blendedLevel);
        }
    }

public:
    LaplacianBlending(const Mat_<Vec3f>& _left, const Mat_<Vec3f>& _right, const Mat_<float>& _blendMask, int _levels)://construct function, used in LaplacianBlending
      left(_left),right(_right),blendMask(_blendMask),levels(_levels)
      {
            assert(_left.size() == _right.size());
            assert(_left.size() == _blendMask.size());
            buildPyramids();  //construct Laplacian Pyramid and Gaussian Pyramid
            blendLapPyrs();   //blend left & right Pyramids into one Pyramid
        };

        Mat_<Vec3f> blend() {
            return reconstructImgFromLapPyramid();//reconstruct Image from Laplacian Pyramid
        }
};

Mat_<Vec3f> LaplacianBlend(const Mat_<Vec3f>& l, const Mat_<Vec3f>& r, const Mat_<float>& m) {
    LaplacianBlending lb(l,r,m,4);
    return lb.blend();
}

int main() {
```

Appendix 5: Grey Level Pixel-run Matrix Texture Features

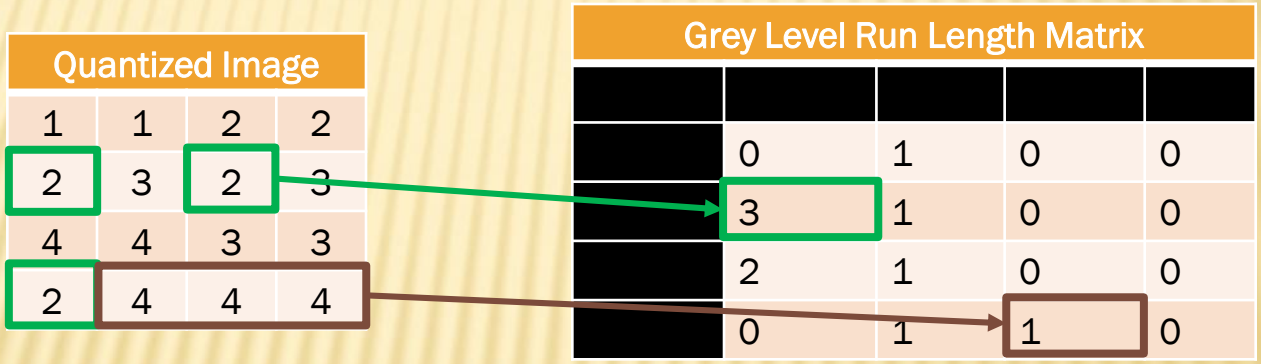
- Average pixel-Run Length (ARL)
  - $ARL = \frac{\sum_n nQ(n)}{\sum n}$
- Standard Deviation of pixel-Run Length (SDRL)
  - $SDRL = \frac{\sum_n (Q(n) - ARL)}{\sum n}$
- Variance of pixel-Run Length (VRL)
  - $VRL = \sum |Q(n) - Q(n-1)|$
- Third Moment of pixel-Run Length (TMRL)

GREY LEVEL RUN LENGTH MATRIX OVERVIEW

- Quantifies runs of the same grey level in the image
- Direction θ of pixel-run is defined similar to that in the GLCM method
- The procedure of constructing the pixel-runs is as follows:
  - each pixel row of image at direction θ is scanned
  - the first pixel of the row is set to be the first pixel-run value
  - The run continues as long as the same grey value as the first pixel is encountered
  - then the next pixel value in the row is scanned starting a new run
  - This procedure is repeated until the scanning of the whole row is completed and a new row is started

GREY LEVEL RUN LENGTH MATRIX EXAMPLE

- d = (1,0)



GREY LEVEL PIXEL-RUN MATRIX HANDS ON EXAMPLE

- **d = (0,1)**

Grey Level Image					
0	0	1	1	1	2
1	1	2	0	0	1
1	1	2	2	2	3
3	1	3	0	0	3
3	3	3	2	1	1
1	0	2	2	2	0

Grey Level Run Length Matrix				
i\j	1	2	3	4
0				
1				
2				
3				

GREY LEVEL PIXEL-RUN MATRIX HANDS ON EXAMPLE

- **d = (0,1)**

Grey Level Image					
0	0	1	1	1	2
1	1	2	0	0	1
1	1	2	2	2	3
3	1	3	0	0	3
3	3	3	2	1	1
1	0	2	2	2	0

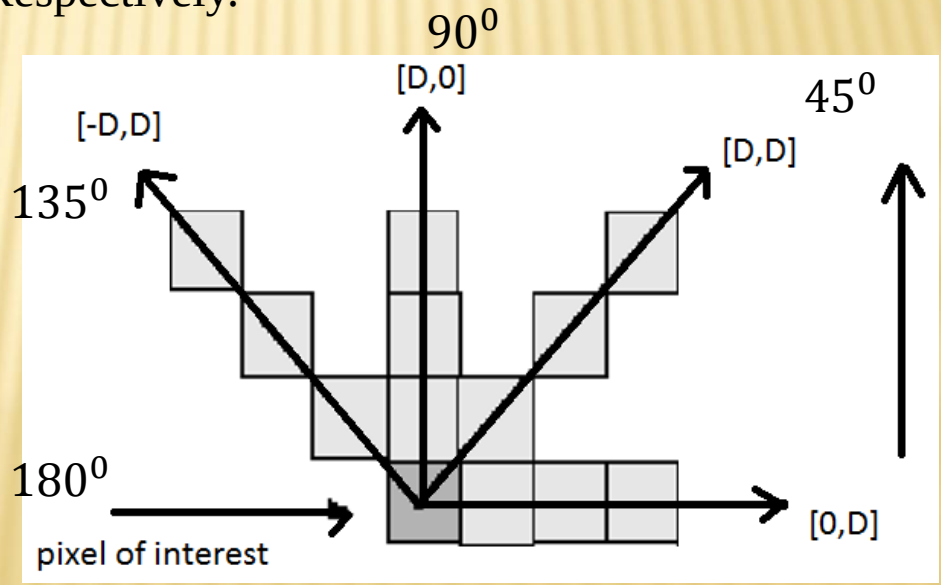
Grey Level Run Length Matrix				
i\j	1	2	3	4
0	8	0	0	0
1	8	2	0	0
2	5	1	0	0
3	1	3	0	0

Appendix 6: **GLCM Directions Of Analysis**

- Horizontal (0^0)
- Vertical (90^0)
- Diagonal:
- Bottom left to top right (-45^0)
- Top left to bottom right (-135^0)

„Denoted P_0, P_{45}, P_{90}, & P_{135} Respectively.
Ex. P_0 (i, j)

GLCM direction analysis:



- GLCM of an image is computed using a displacement vector d, defined by its **radius δ** and **orientation θ**.

- Consider a 4x4 image represented by figure 1a with four gray-tone values 0 through 3. A generalized GLCM for that image is shown in figure 1b where #(i,j) stands for number of times i and j have been neighbors satisfying the condition stated by displacement vector d.

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

1a. Test image

Gray tone	0	1	2	3
0	#(0,0)	#(0,1)	#(0,2)	#(0,3)
1	#(1,0)	#(1,1)	#(1,2)	#(1,3)
2	#(2,0)	#(2,1)	#(2,2)	#(2,3)
3	#(3,0)	#(3,1)	#(3,2)	#(3,3)

1b. General form of GLCM

- The four GLCM for angles equal to 0°, 45°, 90° and 135° and radius equal to 1 are shown in figure 2 a-d.

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

GLCM for δ=1 & θ=0^0

6	0	2	0
0	4	2	0
2	2	2	2
0	0	2	0

GLCM for δ=1 & θ=90^0

4	1	0	0
1	2	2	0
0	2	4	1
0	0	1	0

GLCM for δ=1 & θ=45^0

2	1	3	0
1	2	1	0
3	1	0	2
0	0	2	0

GLCM for δ=1 & θ=135^0

❖ Choice of radius δ

- δ values ranging should be ranging from 1, 2 to 10, but the best result is for δ = 1 and 2.
- Applying large displacement value to a fine texture would yield a GLCM that does not capture detailed textural information.
- As a pixel is more likely to be correlated to other closely located pixel than the one located far away, the above consideration is correct.

❖ Choice of angle θ

- Every pixel has eight neighboring pixels allowing eight choices for θ, which are 0°, 45°, 90°, 135°, 180°, 225°, 270° or 315°.

- According to the definition of GLCM, the co-occurring pairs obtained by choosing θ equal to 0° would be similar to those obtained by choosing θ equal to 180°. This concept extends to 0°,45°, 90°and 135° as well. Hence, one has four choices to select the value of θ.

2D case: ENERGY

- Also called **Uniformity or Angular second moment.**
- M**easures the textural uniformity that is pixel pair repetitions**.
- Detects disorders in textures.
- Energy reaches a maximum value equal to one.

$$Energy = \sum_i \sum_j {p_{ij}}^2$$

Entropy

- **Measures the disorder or complexity of an image**.
- The entropy is large when the image is not texturally uniform.
- Complex textures tend to have high entropy.
- Entropy is strongly, but inversely correlated to energy.

- $Entropy(ent) = -\sum_i \sum_j p_{ij} \log_2 p_{ij}$

2D case:

Contrast

- **Measures the spatial frequency of an image and is difference moment of GLCM**.
- It is the difference between the highest and the lowest values of a contiguous set of pixels.
- It measures the amount of local variations present in the image.

- Contrast(con)=$\sum_i \sum_j (i-j)^2 p_{ij}$

Homogeneity

- Also called as **Inverse Difference Moment**.
- **Measures image homogeneity as it assumes larger values for smaller gray tone differences in pair elements**.
- It is more sensitive to the presence of near diagonal elements in the GLCM.
- It has maximum value when all elements in the image are same.
- Homogeneity decreases **if** contrast increases while energy is kept constant.
- Homogeneity(hom) = $\sum_i \sum_j \frac{1}{1+(i-j)^2} p_{ij}$

Homogeneity

- Also called as **Inverse Difference Moment**.
- **Measures image homogeneity as it assumes larger values for smaller gray tone differences in pair elements**.
- It is more sensitive to the presence of near diagonal elements in the GLCM.
- It has maximum value when all elements in the image are same.
- Homogeneity decreases **if** contrast increases while energy is kept constant.
- Homogeneity(hom) = $\sum_i \sum_j \frac{1}{1+(i-j)^2} p_{ij}$

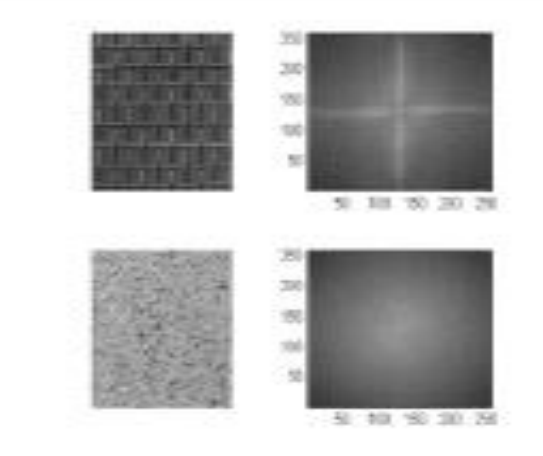
Variance

- **This statistic is a measure of heterogeneity and is strongly correlated to first order statistical variable such as standard deviation**.
- Variance increases when the gray level values differ from their mean.

- Variance(var)= $\sum_i \sum_j (i-\mu)^2 p_{ij}$

where μ is the mean of p_{ij}

FOURIER ANALYSIS



- The Fourier spectra give direction and frequency for periodic or near periodic 2D patterns
- Local FFT in windows
- Concentrated power → regularity
- High frequency power → fine texture
  Directionality → directional texture