Basic Image Processing Algorithms

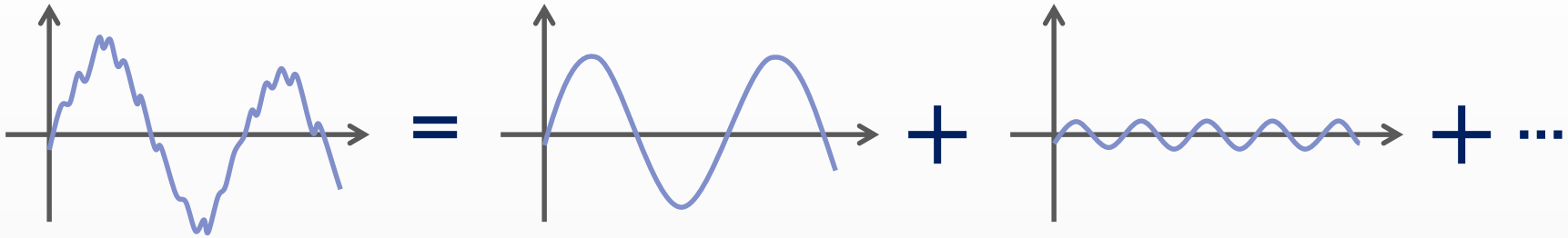
PPKE-ITK

Lecture 4.

Fourier Transformation
Image filtering in the frequency domain

What is Fourier Transform?

- A transformation maps data between (different) domains.
- The Fourier Transform changes between the representation in the **time domain** and in the **frequency domain**.
- The information is the same in both domains, only the representation is different.
- It is a reversible transform.
- It builds on the fact that any function can be represented as a weighted sum of sinusoid functions:



- If we can describe sinusoids we can describe every function.

2D Fourier transform

- Forward transform: map an image $x(n_1, n_2)$ of size $N_1 \times N_2$ from the **spatial domain** into the $X(\omega_1, \omega_2)$ **frequency domain**

$$X(\omega_1, \omega_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

  - even if the image is real the spectrum is complex due to the complex exponential factors
  - ω_1, ω_2 frequencies: continuous variables
  - $X(\omega_1, \omega_2)$ continuous Fourier transform or spectrum of the discrete image
  - Drawback: no computable representation of $X(\omega_1, \omega_2)$
  - Solution: **Discrete Fourier Transform (DFT)**: sample the continuous spectrum with equally spaced frequencies

Discrete Fourier Transform

- We sample one period of the Fourier transform in evenly spaced frequencies:

$$X(\omega_1, \omega_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

The size of the image in the spatial domain is $N_1 x N_2$

The size of the image in the frequency domain will be the same: $N_1 x N_2$

$$X(k_1, k_2) = X(\omega_1, \omega_2)\Big|_{\omega_1 = \frac{2\pi}{N_1} k_1, \omega_2 = \frac{2\pi}{N_2} k_2}$$

$$k_1 = 0,1..., N_1 - 1$$
$$k_2 = 0,1..., N_2 - 1$$

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\frac{2\pi}{N_1} k_1 n_1} e^{-j\frac{2\pi}{N_2} k_2 n_2}$$

Only one period is kept

Discrete Fourier Transform

- Forward formula: gives the description of the image in the discrete frequency domain

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\frac{2\pi}{N_1}k_1 n_1} e^{-j\frac{2\pi}{N_2}k_2 n_2}$$

- Inverse Fourier transform: maps from the discrete frequency domain back to the discrete spatial domain

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) e^{j\frac{2\pi}{N_1}k_1 n_1} e^{j\frac{2\pi}{N_2}k_2 n_2}$$

  - algorithmically it has the same structure as the forward transform,

Discrete Fourier Transform

- DFT is an exact transform, there is no transformation error.
  - Not surprising, since we use the same image size for the representation of both $x(n_1, n_2)$ and $X(\omega_1, \omega_2)$.
- Most of the properties of continuous FT hold for DFT
  - Except linear shift of FT becomes circular shift for DFT.
- DFT and inverse DFT are computable transformations
- There are fast ways to compute the DFT: ***Fast Fourier Transform***
  - If the size of the image is *N*x*N*, then the **naive implementation** requires N^4 multiplications: N^2 for each (k_1,k_2) point.
  - The **FFT** with row/column decomposition requires only $N^2 \log_2 N$ multiplications.
- **FFT makes the Fourier transformation applicable in many practical cases.**

Interpretation of Fourier coefficients

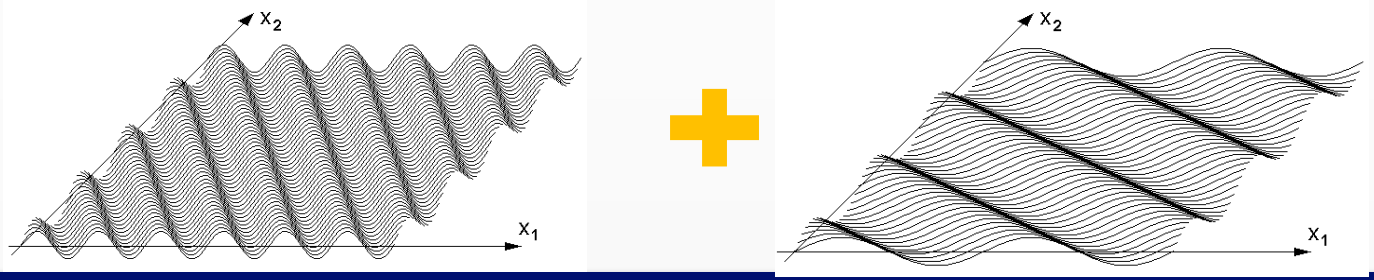
- Analogy of Fourier coefficient based representation:
  - Consider the image as a superposition of sinusoid/cosine waves with different amplitudes, frequencies and directions
  - 1D case in formulas: (F_k: signal, D_n Fourier cofficient)

$$F_k = \sum_{n=0}^{N-1} D_n \exp(j \frac{2\pi}{N} k\, n)$$

$$F_k = D_0 + D_{N/2} \cos(\pi k) + \sum_{n=1}^{N/2-1} 2\left(\mathrm{Re}(D_n) \cos\left(\frac{2\pi}{N} kn \right) - \mathrm{Im}(D_n) \sin\left(\frac{2\pi}{N} kn \right) \right)$$

D_0: real number, average of the function (image)

  - 2D case (e.g. image) visualization:

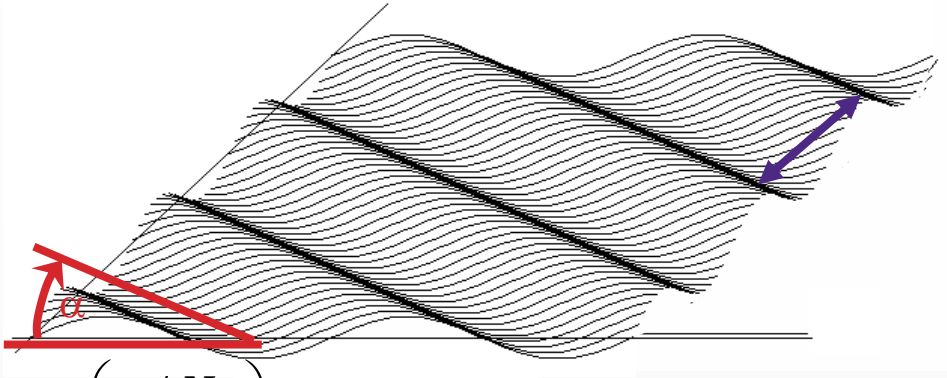


Interpretation of 2D Fourier coefficients

$$C_{mn}\exp\left(j2\pi\left(m\frac{x}{N_1}+n\frac{y}{N_2}\right)\right)+C_{-m-n}\exp\left(-j2\pi\left(m\frac{x}{N_1}+n\frac{y}{N_2}\right)\right)=$$

$$\mathrm{Re}(C_{mn})\cdot 2\cdot\cos\left(2\pi\left(m\frac{x}{N_1}+n\frac{y}{N_2}\right)\right)-\mathrm{Im}(C_{mn})\cdot 2\cdot\sin\left(2\pi\left(m\frac{x}{N_1}+n\frac{y}{N_2}\right)\right)$$

- real part of a C_{mn} Fourier coefficient is the **amplitude** of a cos-wave, while the imaginary part is the amplitude of a sinusoid wave
- **wavelength** and **orientation** of the waves are encoded in the (m,n) *position* coordinates of the coefficients in the 2D Fourier map



wavelength:

$$\frac{1}{\lambda}=\sqrt{\left(\frac{m}{N_1}\right)^2+\left(\frac{n}{N_2}\right)^2}$$

orientation:

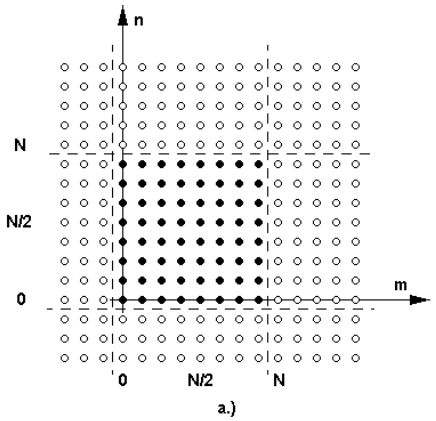
$$\alpha=\arctan\left(\frac{n/N_2}{m/N_1}\right)$$

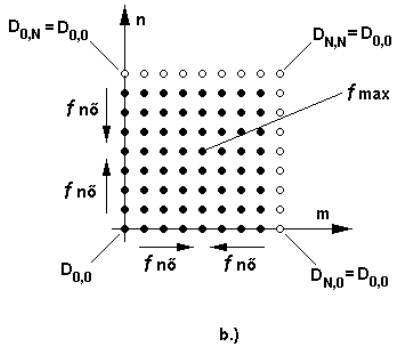
m=3 n=2

Spatial frequency:

$$f=1/\lambda$$

Illustration of the periodicity of coefficients Centered DFT: better visualization





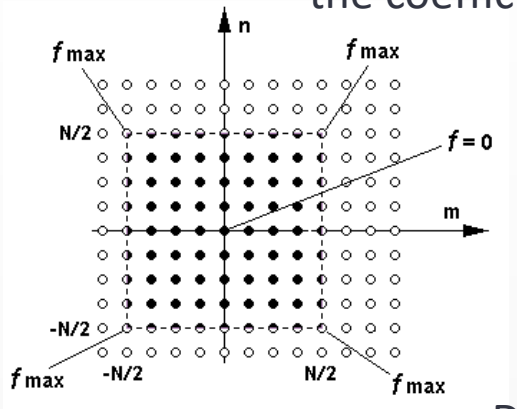
- In the center of the DFT array D_{00} is the *zero-frequency coefficient (DC compo-nent)*
- Distance from the center
  - Frequency of the corresponding sin/cos wave

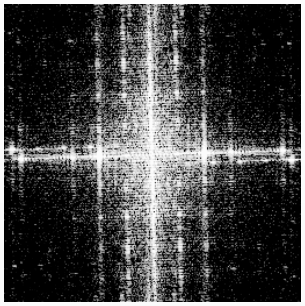
$$f = \sqrt{(m/N_1)^2 + (n/N_2)^2}$$

- Orientation
  - Direction perpendicular to the wavefront

$$\gamma = \arctan\left((n/N_2)/(m/N_1)\right)$$

Re-arrangement of the coefficient matrix



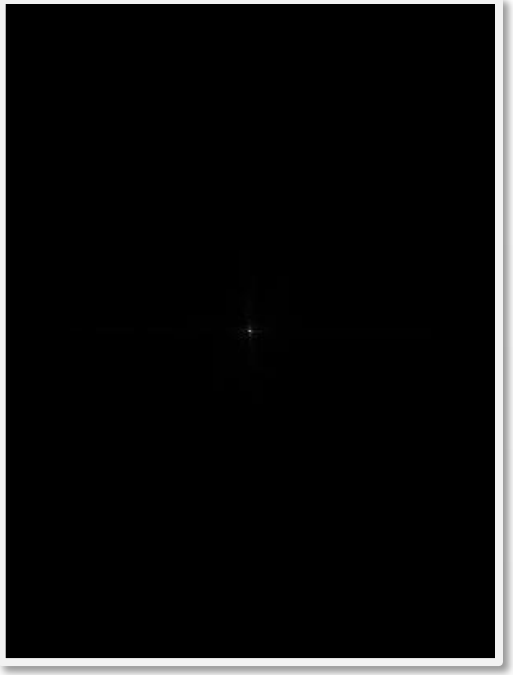


DFT abs. value image

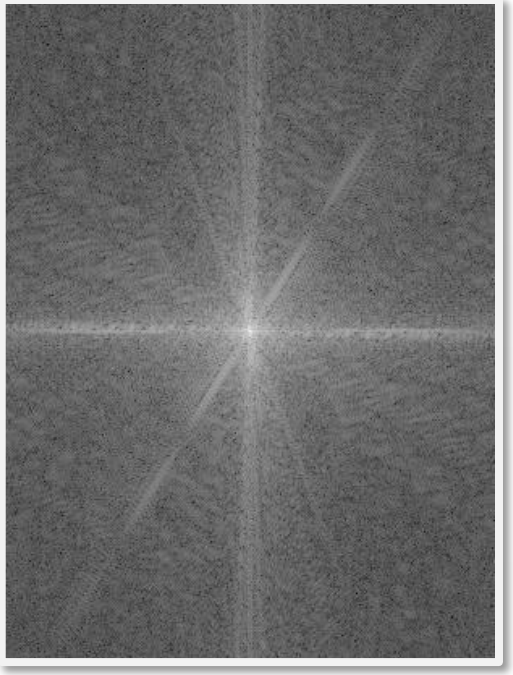
Point-wise Intensity Transformation

- **Log transformation**:
  - Commonly used to visualize the Fourier transform of an image

Original Image*



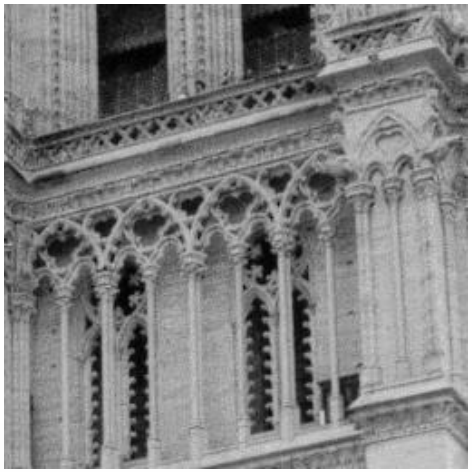
The magnitude of the DFT

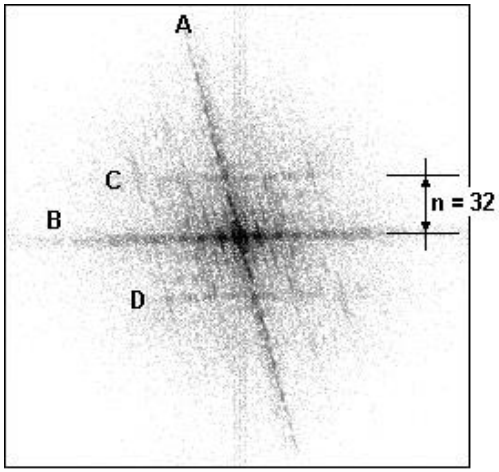


Log of the magnitude of the DFT

*Chez Mondrian by András Kertész (1926)

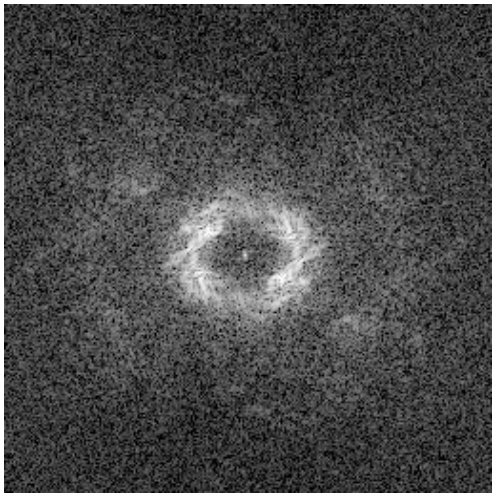
Imaged absolute values of DFT coefficients – facade of the Notre Dame Paris





- In the transformed map, directions of strong lines are <u>perpendicular</u> to the major contours in the image:
  - **A** line- horizontal ledges (párkányok)
  - **B** line- slim vertical columns.
  - **C** and **D** lines – periodic vertical patterns with the frequency „$n = \pm 32$” : decoration of the windows behind the columns

Imaged absolute values of DFT coefficients– analysing fingerprint images



- No characteristic lines in the transform
  - ← ridges of fingerprints run in any directions
- At *d* distance from the center a significant ring shaped maximum
  - ← the average spatial frequency of the fingerprint ridges is *d*-times the basic frequency f_b, and there exist no nominant directions

Filtering in the DFT space

- Low-Pass Filter (LPF):
  - Filtering out the large spatial frequencies

a) b)

Result of filtering out large frequencies. Erased all coeff. a.) above 16 f_b, b.) above 8 f_b

Slide credit ® Prof. Vladimir Székely, BME

Filtering in the DFT space

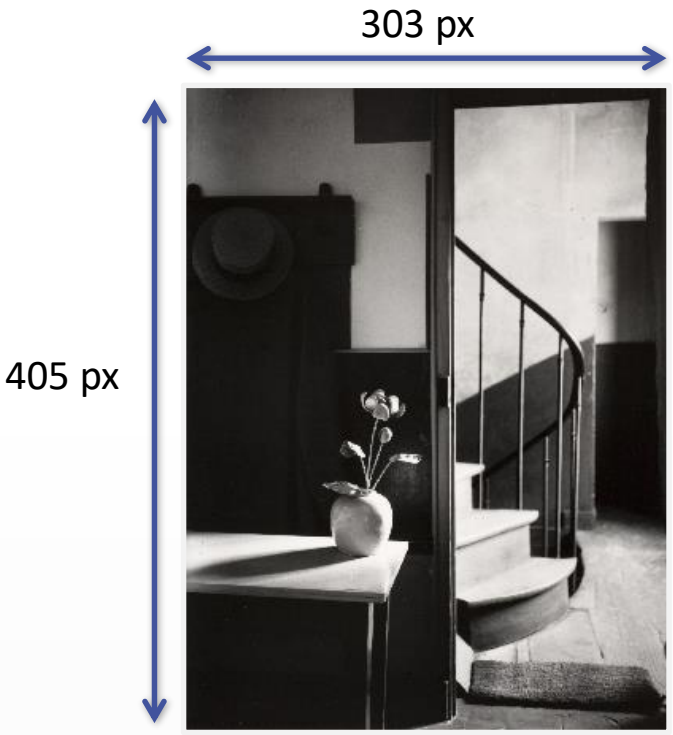
- High-Pass Filter (HPF)
  - Filtering out the low spatial frequencies

a) b)

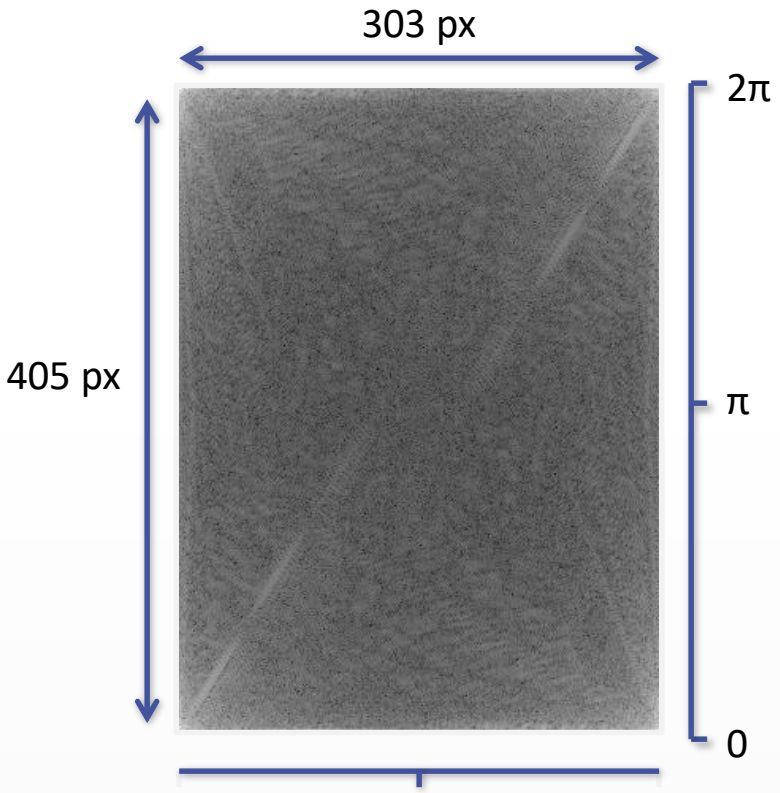
Result of filtering out large frequencies. Erased all coeff. a.) below 4 f_b, b.) below 10 f_b

Slide credit ® Prof. Vladimir Székely, BME

Discrete Fourier Transform



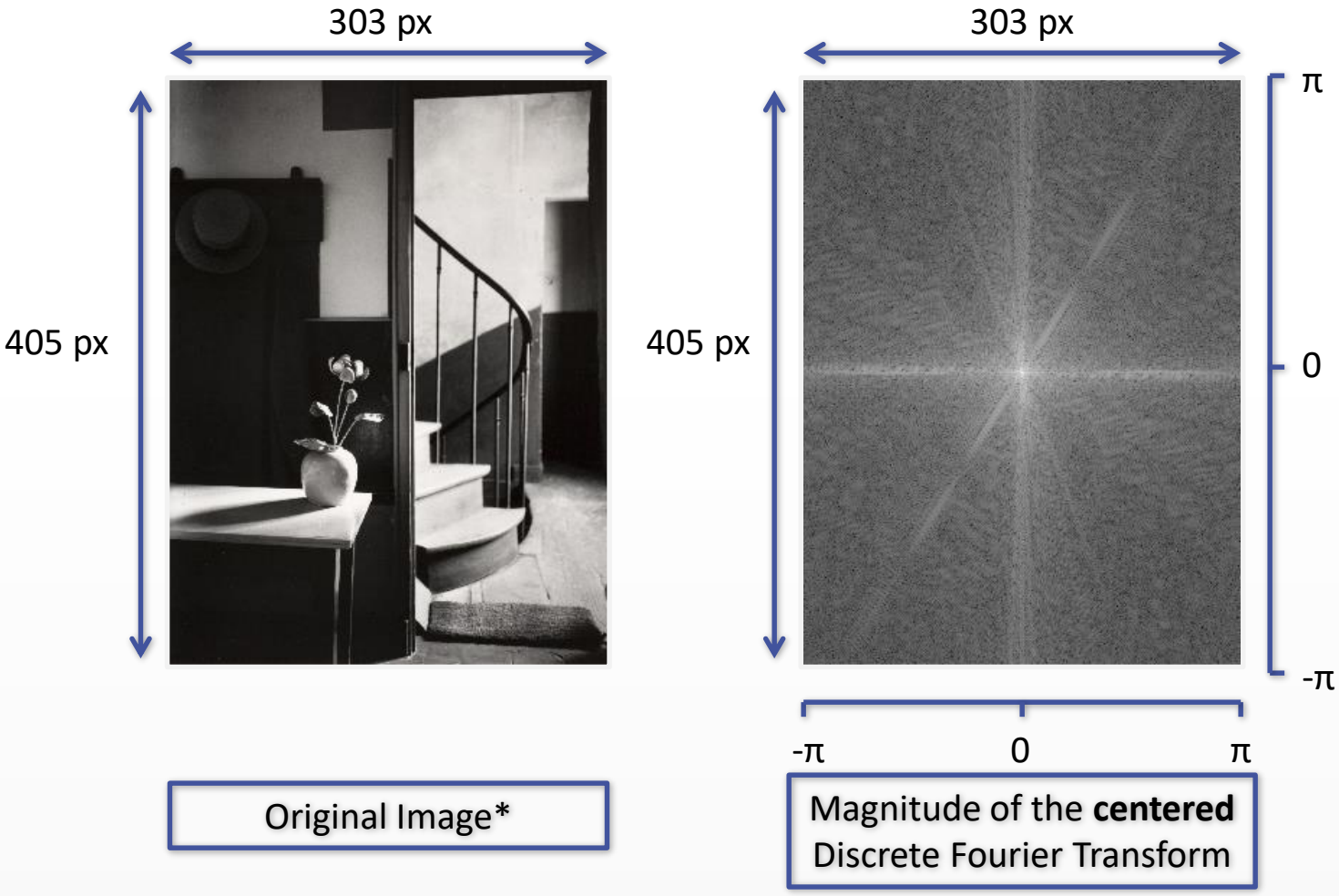
Original Image*



Magnitude of the Discrete Fourier Transform

*Chez Mondrian by András Kertész (1926)

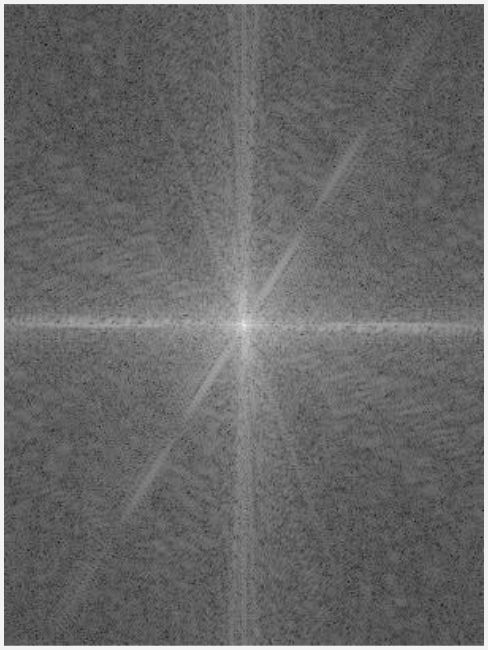
Discrete Fourier Transform



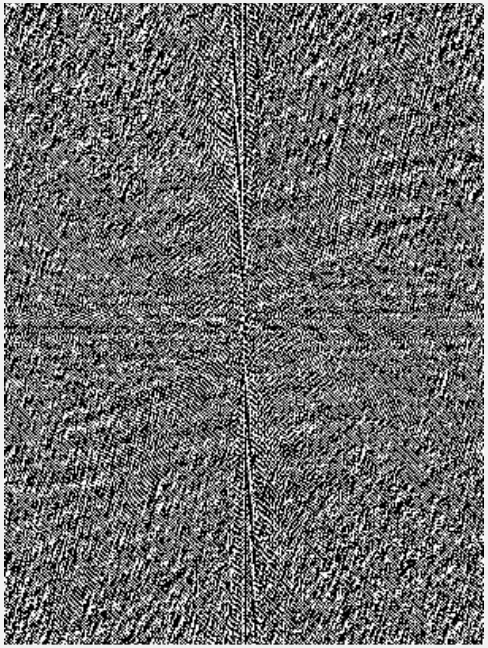
*Chez Mondrian by András Kertész (1926)

Discrete Fourier Transform

Original Image*



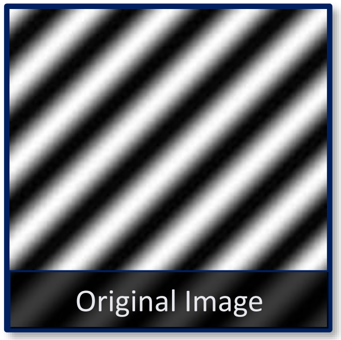
Magnitude of the **DFT**

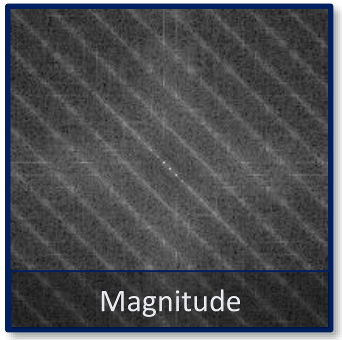


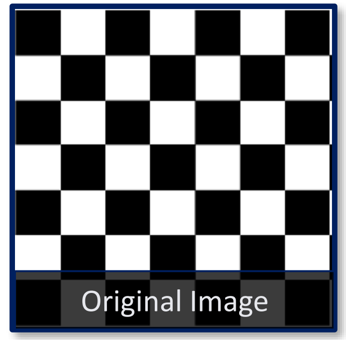
Phase of the **DFT**

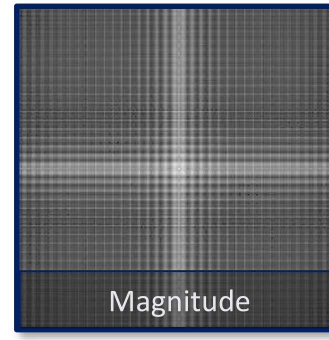
*Chez Mondrian by András Kertész (1926)

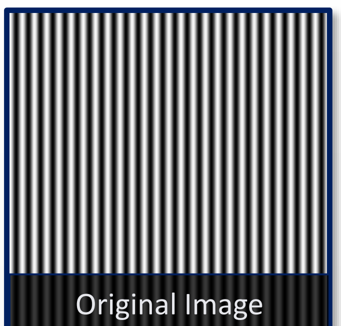
Discrete Fourier Transform - Examples

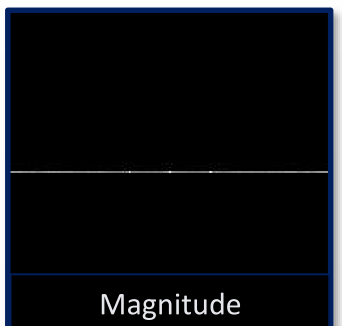

Original Image


Magnitude

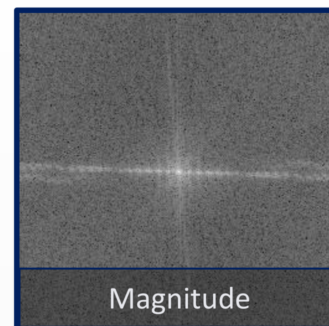

Original Image


Magnitude

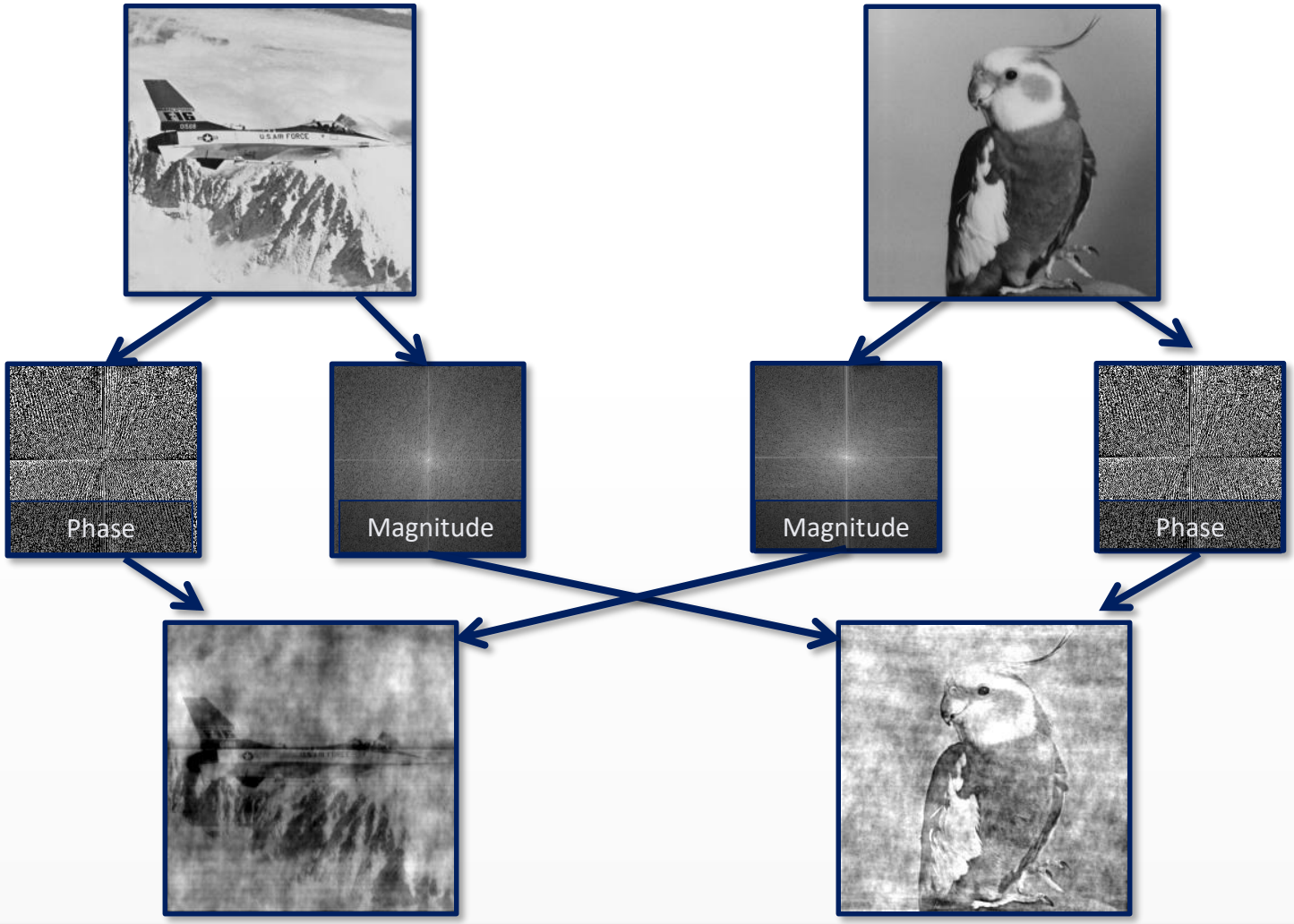

Original Image


Magnitude


Original Image


Magnitude

Discrete Fourier Transform

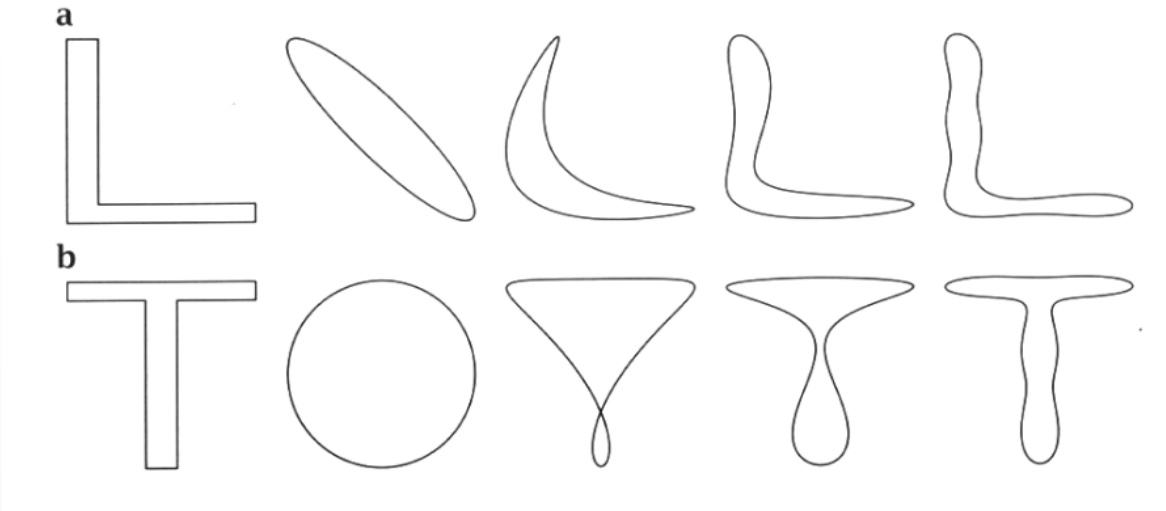


Representing curves with Fourier-descriptors

1. Complex numbers from the 2D curve points: $z_k = x_k + j \cdot y_k$
2. 1D DFT transform calculated for the complete closed curve

$$C_n = \sum_{k=0}^{K-1} z_k \exp(j \frac{2\pi}{K} n \cdot k)$$

3. Setting high frequency C_n coefficient to zero, then recovering of the approximate contour points by inverse transform



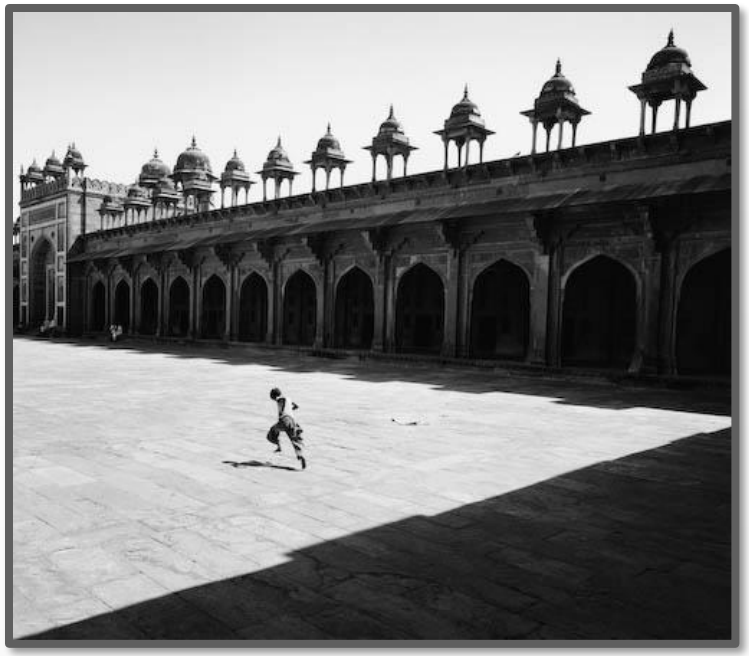
Reconstruction of letters „L” and „T” with 2, 3, 4 and 8 Fourier coefficients

Homomorphic filtering

- ***Motivation*: image** with **large dynamic range**, e.g. natural scene on brightly sunny day, recorded on a **medium** with **small dynamic range** results in image contrast significantly reduced especially in dark and bright regions

- ***Goal*:** reduce the dynamic range, increase contrast

- Example for a spatial filter also using Fourier-based steps



Homomorphic filtering

- It simultaneously ***normalizes the brightness*** across an image and ***increases contrast***.
- Assumes the following image model: the image is formed by recording the light reflected from the objects illuminated by a light source.

$$x(n_1, n_2) = i(n_1, n_2) \cdot r(n_1, n_2)$$

Illumination: slowly varying, main contributor to dynamic range

Reflectance: rapidly varying, main contributor to local contrast

- We want to reduce the illumination component, and increase the reflectance component.

Homomorphic filtering

- The main steps of homomorphic filtering:
  1. To separate the two components we first use log transformation:

$$\log\big(x(n_1,n_2)\big)=\log\big(i(n_1,n_2)\big)+\log\big(r(n_1,n_2)\big)$$

  2. Since we assume that the illumination component varies slowly and the reflectance varies rapidly, we can get the two component by using (Fourier-based) low and high pass filters:

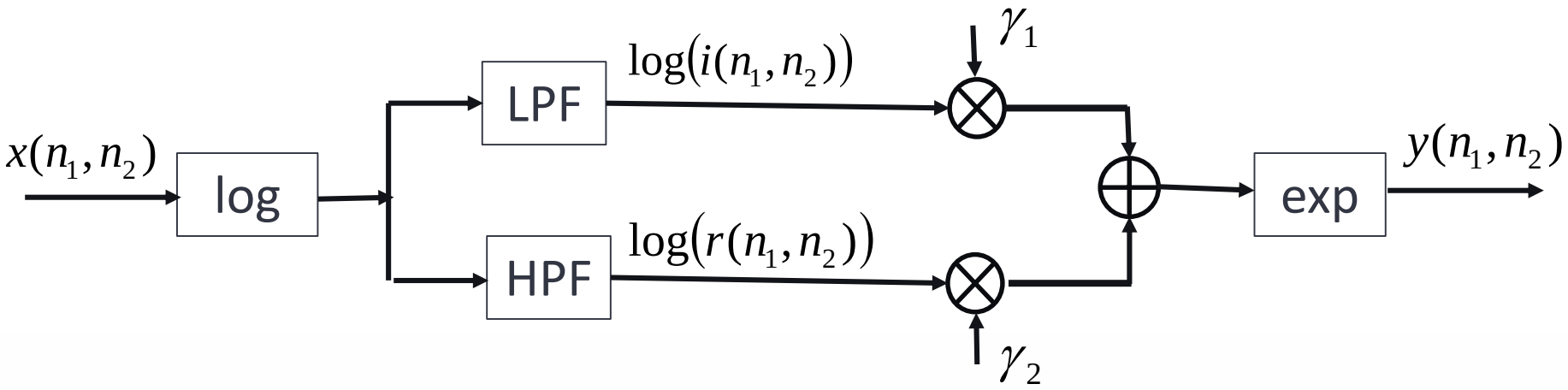
$$\log\big(i(n_1,n_2)\big)=LPF\big[\log\big(x(n_1,n_2)\big)\big]$$
$$\log\big(r(n_1,n_2)\big)=HPF\big[\log\big(x(n_1,n_2)\big)\big]$$

  3. Weight the two component:

$$\log\big(y(n_1,n_2)\big)=\gamma_1\log\big(i(n_1,n_2)\big)+\gamma_2\log\big(r(n_1,n_2)\big),\ \text{where}\ \ \gamma_1<1,\gamma_2>1$$

  4. Transform back to the original range, using the exponential transform.

Homomorphic filtering

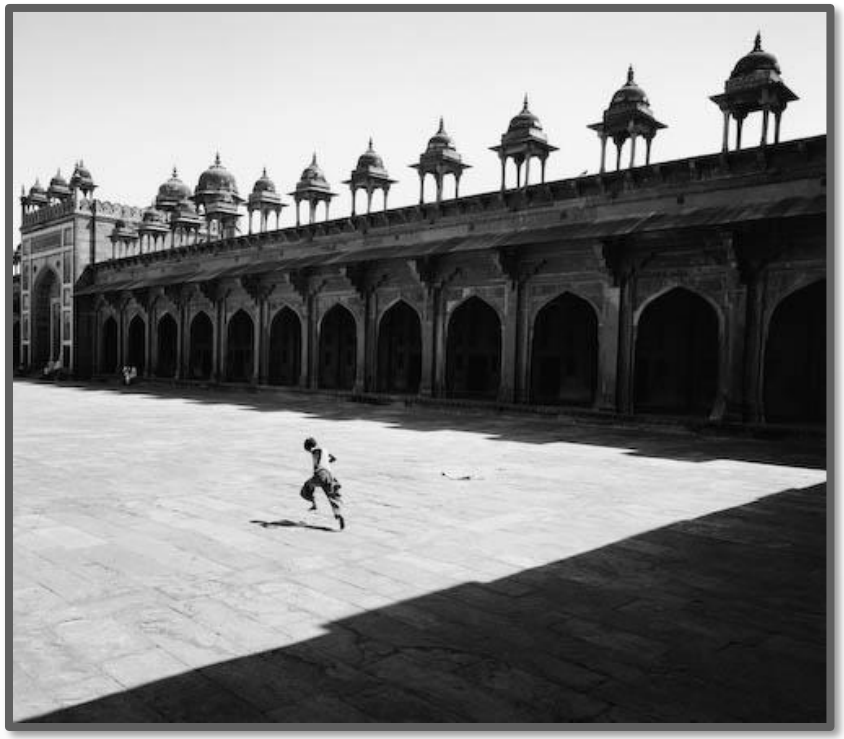


$$x(n_1, n_2) = i(n_1, n_2) r(n_1, n_2)$$

$$\log(y(n_1, n_2)) = \gamma_1 \log(i(n_1, n_2)) + \gamma_2 \log(r(n_1, n_2))$$

$$y(n_1, n_2) = [i(n_1, n_2)]^{\gamma_1} [r(n_1, n_2)]^{\gamma_2}$$

Homomorphic filtering



Original Image

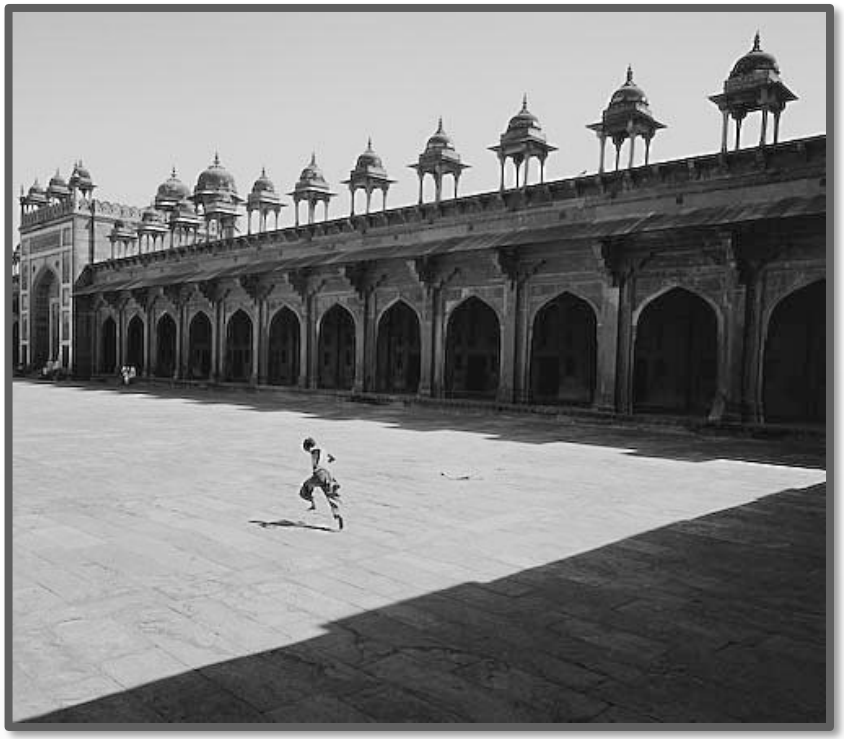


Image after homomorphic filtering

Main Sources

Fundamentals of Digital Image and Video Processing lectures by Aggelos K. Katsaggelos

Introduction to Fourier Transform (https://www.youtube.com/watch?v=1JnayXHhjlg)

Introduction to Compex Exponential Function (https://www.youtube.com/watch?v=qjT3XvS7Qno)

Texture analysis

Textures - definition

- Textures demonstrate the difference between an artificial world of objects whose surfaces are only characterized by their color and reflectivity properties to that of real world imagery
- How we can define texture: microstructure
  - certainly not: an arbitrary pattern that extends over a large image
- Basic properties
  - Small elementary pattern which is repeated periodically or quasi-periodically in space (like pattern on a wall paper)
- It is sufficient to describe:
  - Small elementary pattern
  - Repetition rules (characteristic scales)
- Types
  - Artificial (Julesz, Pratt, Gagalowic)
  - Natural (Brodatz)

Textures - definition

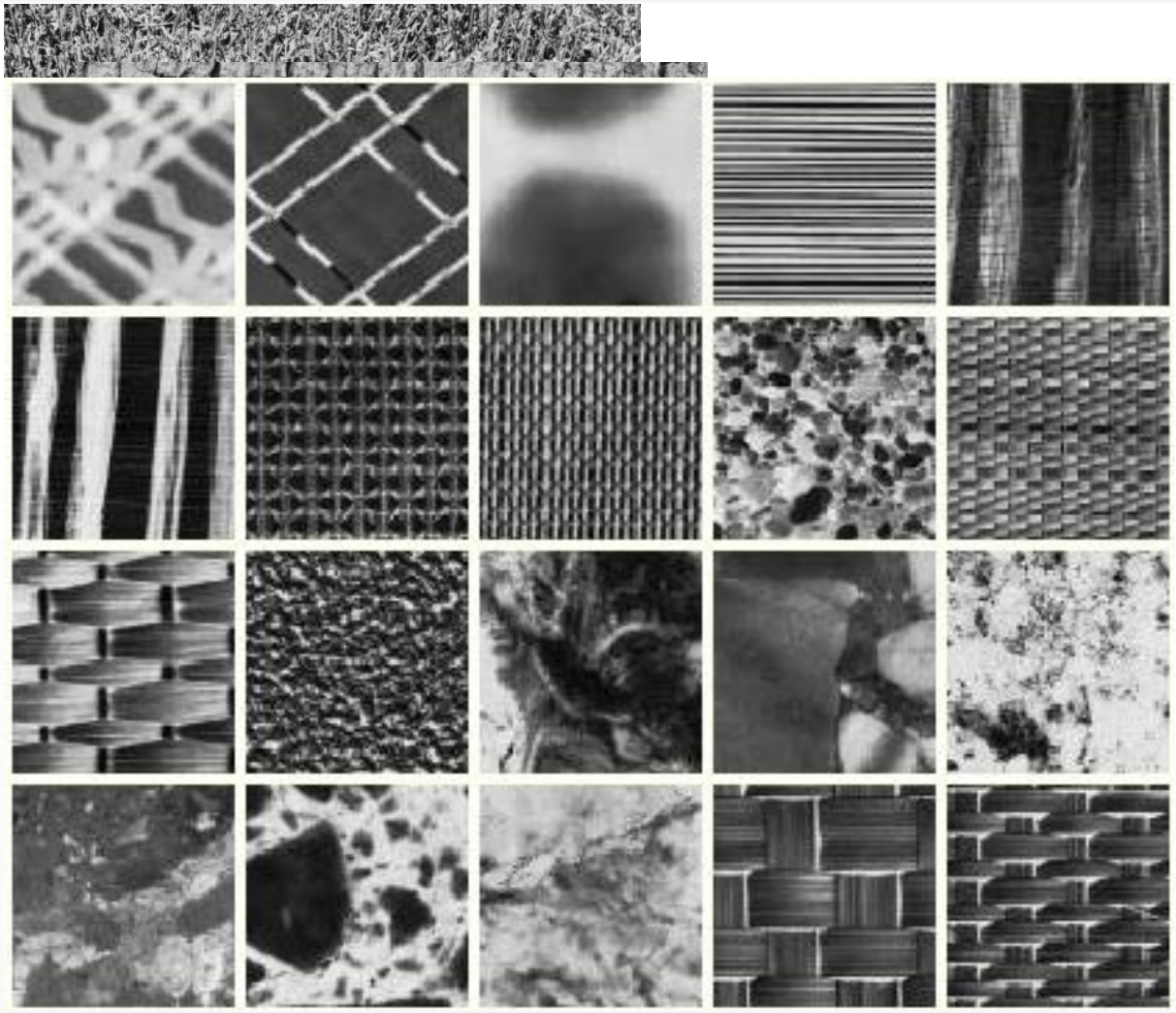
- Hawkins:
  - Some local ‚order' is repeated over a region which is large in comparison to the order's size,
  - The order consists in the nonrandom arrangement of elementary parts
  - The parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region
- Description:
  - *coarseness* ~ period of repetition
    - e.g. wool is "coarser" than silk, under the same conditions
  - fineness / rudeness, contrast, orientation, arrangement ...

Texture



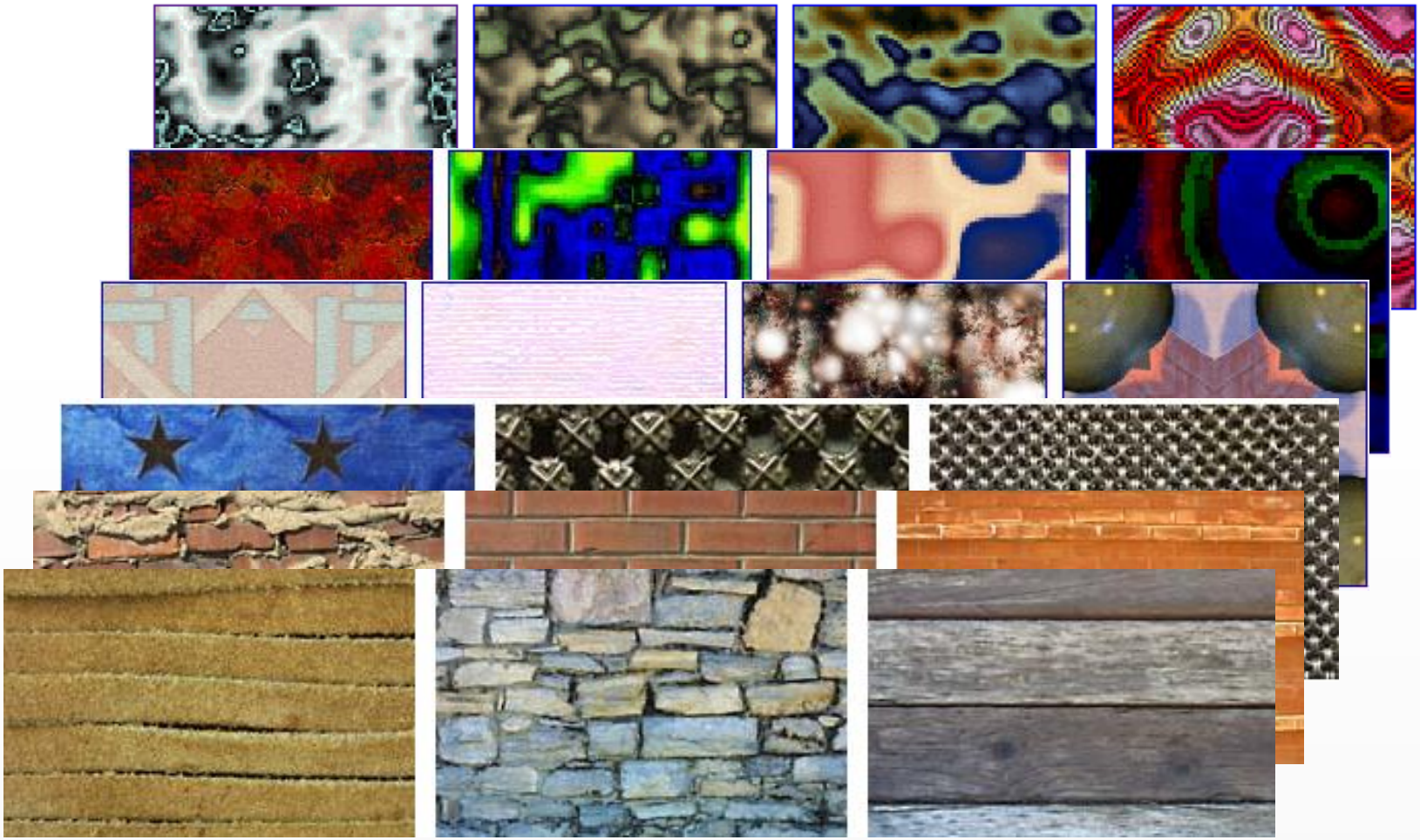
- A texture is an image that follows some statistical properties
- It has similar structures repeated over and over again

Natural textures - Brodatz



grass (fű)
bark (fakéreg)
canvas (vászon)
sand (homok)
pigskin (disznóbőr)
oxhide (marhabőr)...

Artificial textures

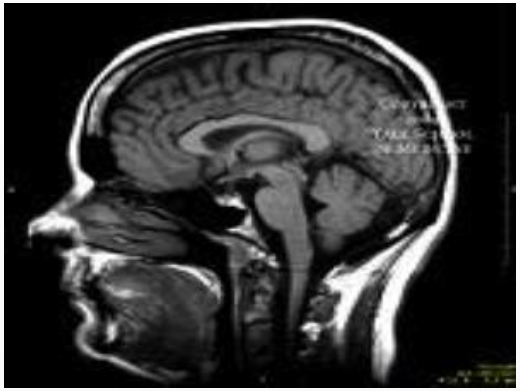


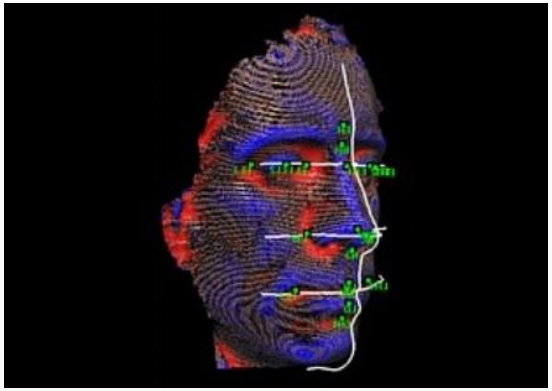
Application Areas of Texture Analysis

Food processing industry

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

Medical image analysis





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

Texture analysis:

- There are various primary issues in texture analysis:
  - *TEXTURE CLASSIFICATION*
  - *TEXTURE SEGMENTATION*
  - *SHAPE RECOVERY FROM TEXTURE, and*
  - *MODELING*.

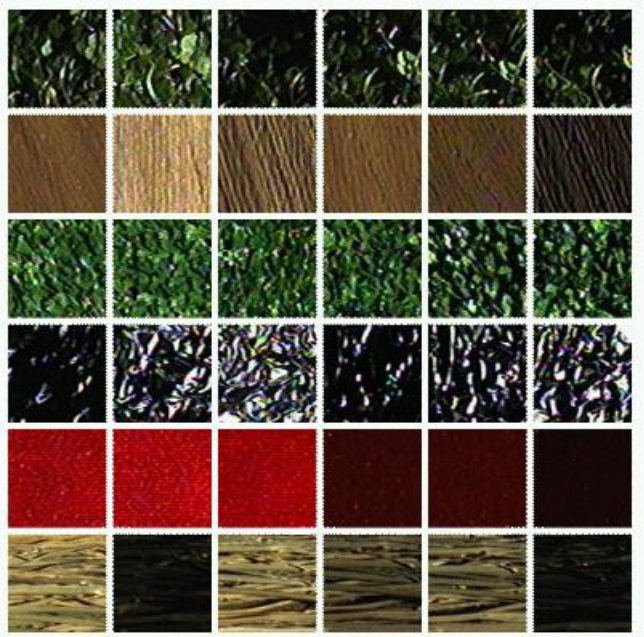
Texture classification

- In texture **classification**, the problem is **identifying** the given textured region from a given set of texture classes.
  - The texture analysis algorithms **extract distinguishing feature** from each region to facilitate classification of such patterns.

Novel image to be classified

?

=



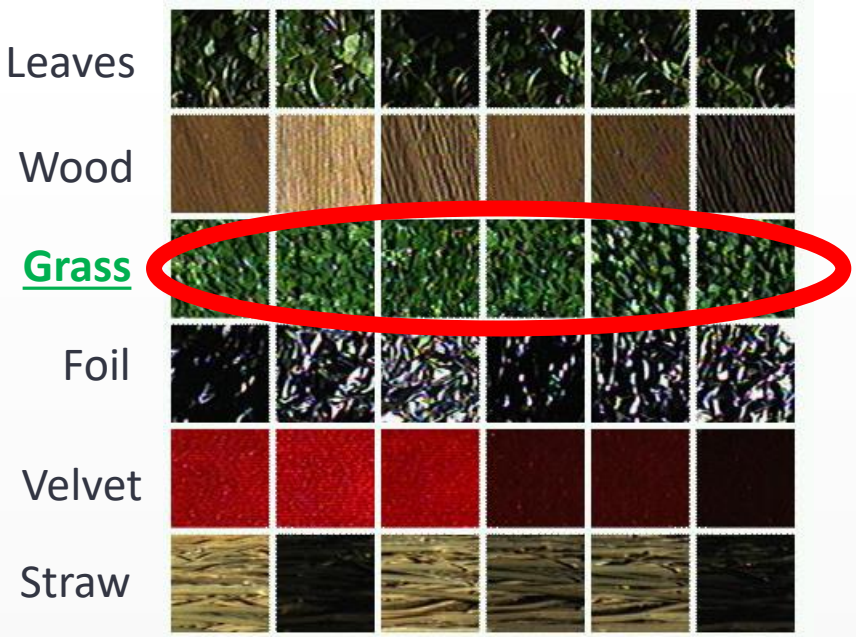
Texture classification

- In texture **classification**, the problem is **identifying** the given textured region from a given set of texture classes.
  - The texture analysis algorithms **extract distinguishing feature** from each region to facilitate classification of such patterns.

Novel image to be classified

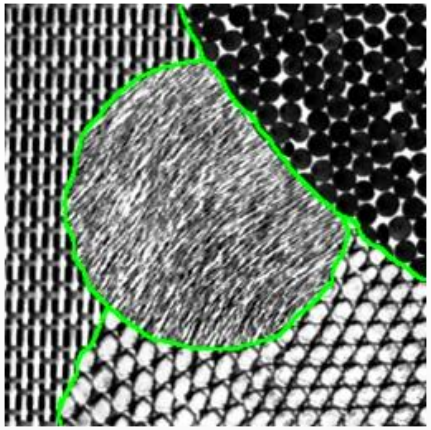
?

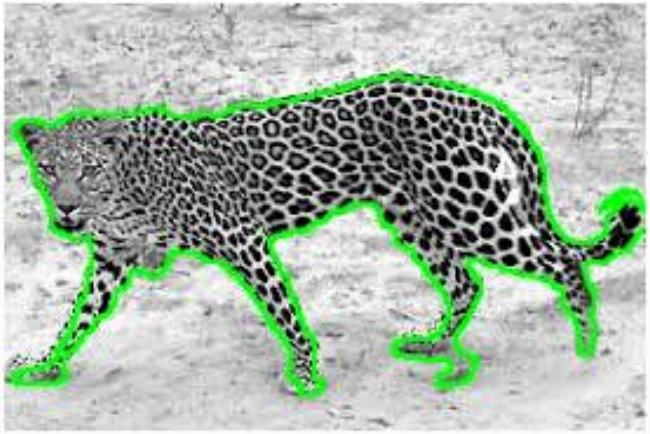
=



Texture segmentation

- Unlike texture classification, texture **segmentation** is concerned with automatically determining the **boundaries** between various textured regions in an image.
- Both reign-based methods and boundary-based methods have been attempted to segments texture images.



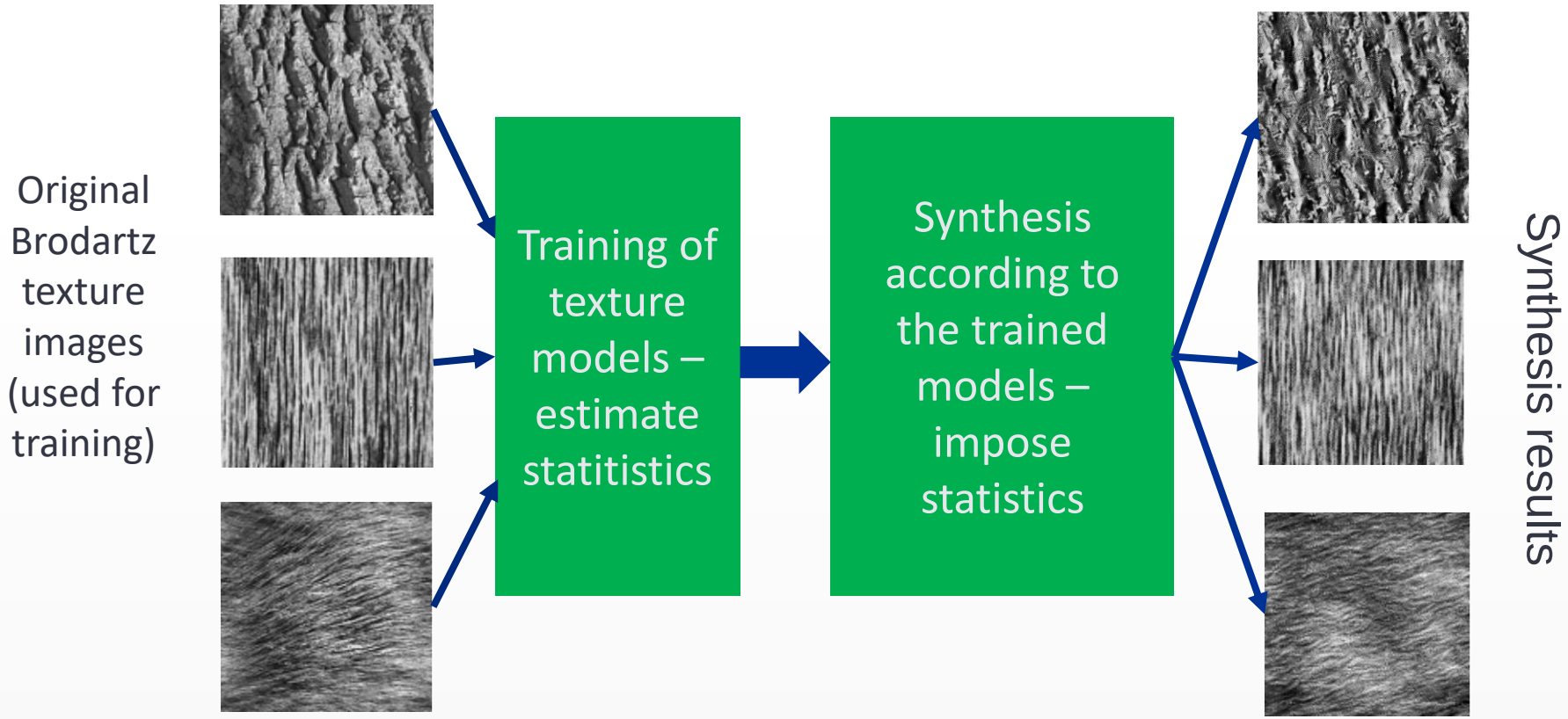


Shape recovery from texture

- Image plane variation in the texture properties, such as density, size and orientation of texture primitives, are the cues exploited by shape –from-texture algorithms.
- Quantifying the changes in the shape of texture elements is also useful to determine surface orientation.

Texture modeling

- Specify a model that clearly identifies the given pattern sample



Techniques for Texture Extraction

- There are various techniques for texture extraction. Texture feature extraction algorithms can be grouped as follows:
  - **Statistical**
  - **Geometrical**
  - **Model based**
  - **Signal Processing**

Statistical method includes

A. *Local features*
   - Grey level of central pixels,
   - Average of grey levels in window,
   - Median,
   - Standard deviation of grey levels,
   - Difference of maximum and minimum grey levels,
   - Difference between average grey level in small and large windows,
   - Kirsch feature,
   - Combine features

B. *Galloway*
   - run length matrix

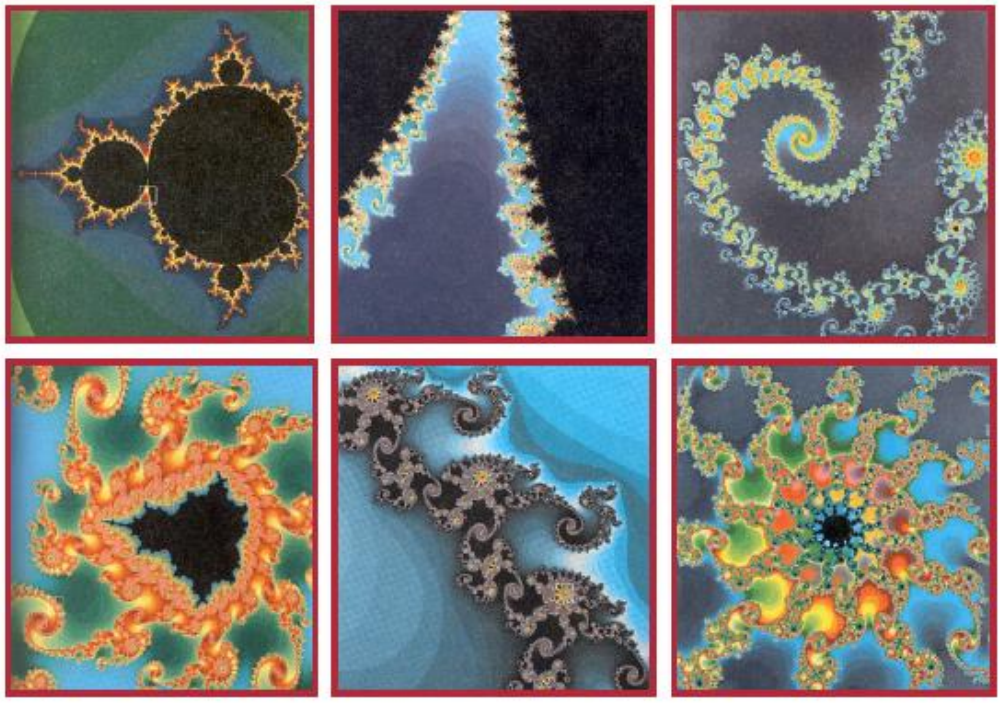
C. *Haralick*
   - co-occurrence matrix

Geometrical method includes

- First threshold images into binary images of n grey levels.
- Then calculate statistical features of connected areas.

Model based method includes

- These involve building mathematical models to describe textures:
  - **Markov random fields** (see: later – image segmentation lecture)
  - Fractals:

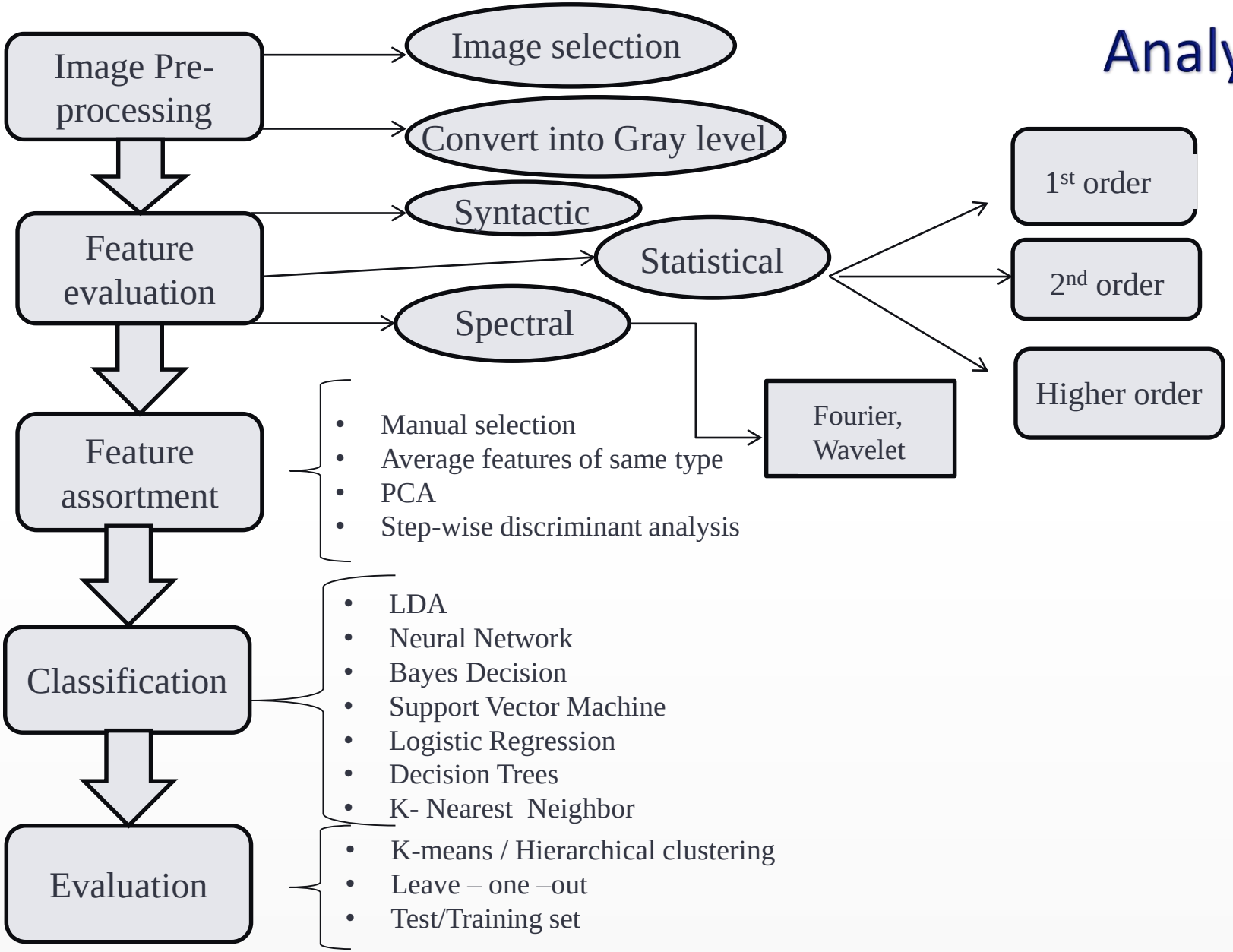


Signal processing includes

- These methods involve transforming original images using filters and calculating the energy of the transformed images.

- **Law's masks (see: today – later)**
- Laines – Daubechies wavelets
- **Fourier transform (see: today – earlier)**
- **Gabor filters (see: today – later)**

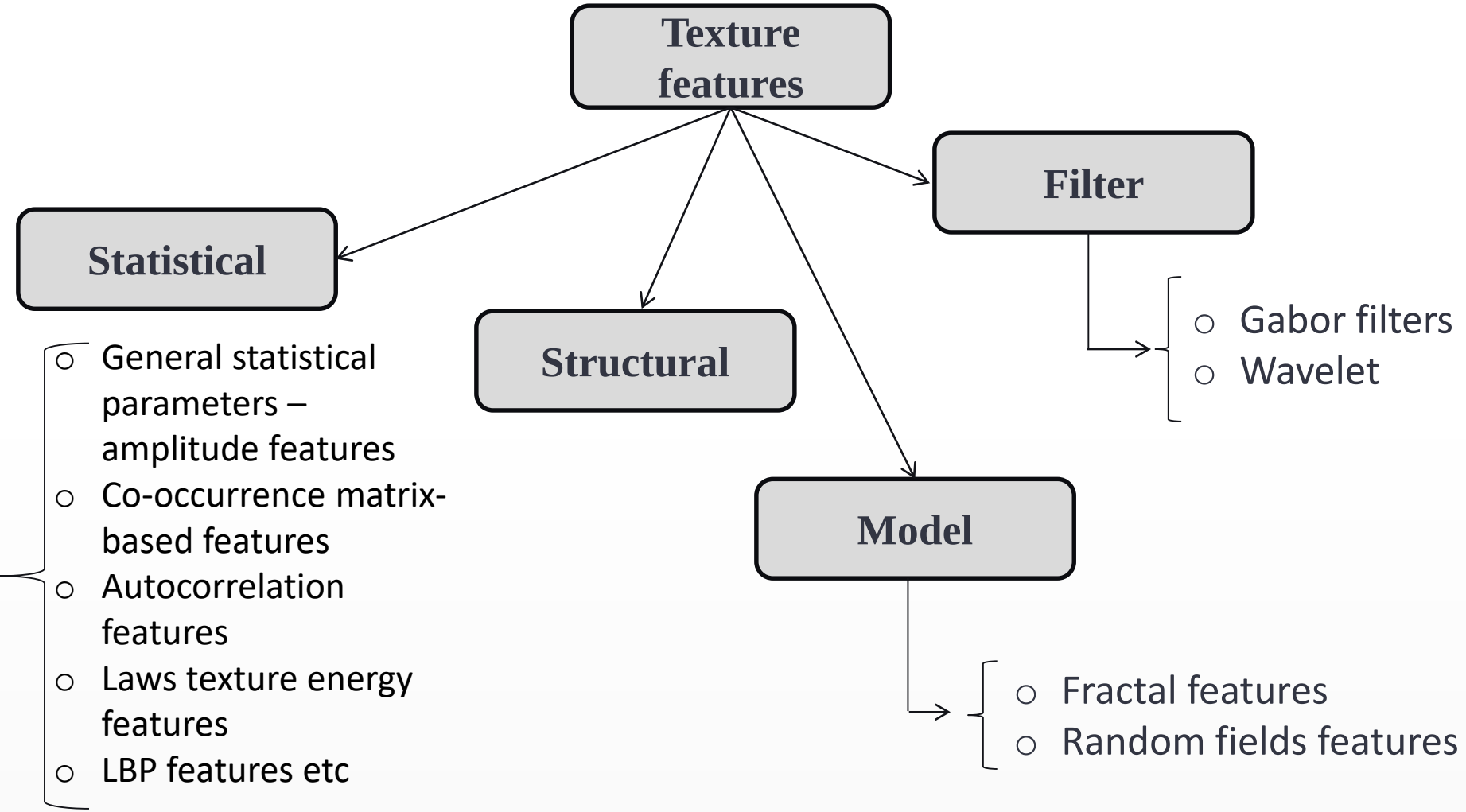
Flowchart for Texture Analysis



Discrimination vs. classification of textures

- Discrimination
  - Classification - grouping of blobs or points, and classifying them into various classes (local attributes)
  - Segmentation - separation of spots / areas (local properties + neighborhoods)
- Classification
  - Supervised approach
    - take samples of textures (statistics, metrics) then
    - examine the similarity of new textures
  - Unsupervised
    - evaluate statistics
    - sample categorization into classes

Methods for Texture Features



Statistical features
Amplitude features

- Amplitude-features
  - Mean

$$M(j,k) = \frac{1}{(2w+1)^2} \sum_{m=-w}^{w} \sum_{n=-w}^{w} F(j+m, k+n)$$

  - Deviation

$$S(j,k) = \frac{1}{(2w+1)^2} \left[\sum_{m=-w}^{w} \sum_{n=-w}^{w} \left[F(j+m, k+n) - M(j+m, k+n) \right]^2 \right]^{1/2}$$

Statistical features
Gray Level Co-occurrence Matrix (GLCM)

- Also referred as **co-occurrence distribution**.
- It is the most classical second-order statistical method for texture analysis.
- An image is composed of **pixels** each with an intensity (a specific gray level), the GLCM is a tabulation of how often different combinations of gray levels co-occur in an image or image section.
- **Gray Level Co-occurrence Matrix** (GLCM) filters operate by computing, for each filter window position, how often specific pairs of image cell values occur in neighboring cell positions (such as one cell to the right).
- The results are tabulated in a co-occurrence matrix, and specific statistical measures are computed from this matrix to produce the filtered value for the target cell

Statistical features
Gray Level Co-occurrence Matrix (GLCM)

- First order statistics example: simple histogram
  - Measures the number of different gray value occurrences of independent pixels
  - h_i: number of pixels in the image with gray value i
- Second order statistics example: GLCM
  - Measures the frequencies of joint gray value occurrences of different pixel pairs with a pre-defined spatial offset
  - $p_{ij}(\Delta x, \Delta y)$: frequency of pixel pairs with an offset $(\Delta x, \Delta y)$, where the gray value of the first pixel is i and the gray value of the second pixel is j
  - For example: Δx=1, $\Delta y = 0$, i=0, j=255: frequency of one-pixel-wide vertical black-white transitions in an image

Statistical: Co-occurrence Matrix-based Features

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

$$p_{ij}(\Delta x, \Delta y) = W \cdot Q(i, j|\Delta x, \Delta y)$$

where,

$$W = \frac{1}{(M - \Delta x)(N - \Delta y)} \qquad Q(i, j|\Delta x, \Delta y) = \sum_{n=1}^{N-\Delta y} \sum_{m=1}^{M-\Delta x} A$$

where,

$$A = \begin{cases} 1 & \text{if } f(m,n) = i \text{ and } f(m+\Delta x, n+\Delta y) = j \\ 0 & \text{otherwise} \end{cases}$$

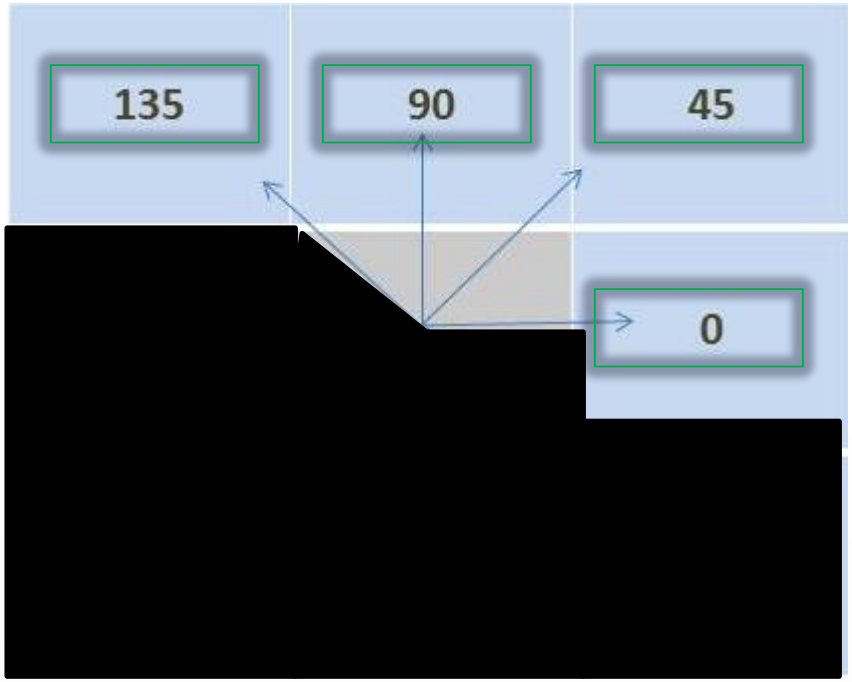
Computation of Co-occurrence Matrix

- It has size $N \times N$ (N = Number of gray-values) i.e., the rows & columns represent the set of possible pixel values.
- Polar representation of the offset:
  - two parameters d, θ (instead of $\Delta x, \Delta y$):

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, …)

8 Directions/orientations (θ) of Adjacency



we consider θ as horizontal (0°), front diagonal (45°), vertical (90°) and back diagonal (135°)

Computation of Co-occurrence Matrix

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*

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)

Pixel values: 0,1,2,3. **Thus,** $N = 4$

Thus, *size* of CM = 4x4

$d = 1$

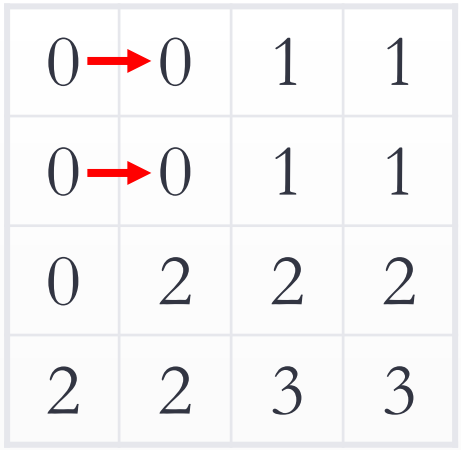
θ = horizontal (0°)

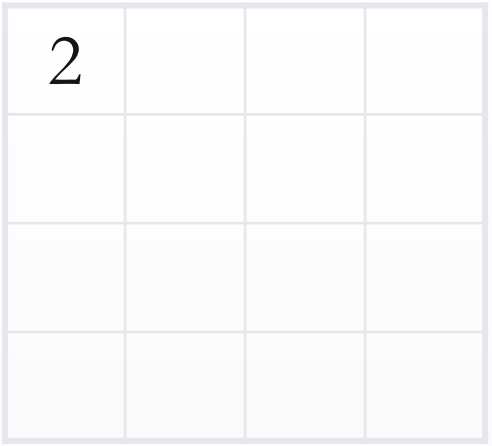
Example: Computation (contd.)

i/j	**0**
0	**#(0,0)**

$d = 1$

θ = horizontal (0°)





Example: Computation (contd.)

$d = 1$ θ = horizontal (0°)

Image

0	0→ 1		1
0	0→ 1		1
0→	2	2	2
2	2	3	3

2	**2**	**1**	0

CM for the Image

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

Example: Computation (contd.)

$d = 1$ $\theta =$ **horizontal**$(0°)$

Image

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

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

CM for the Image

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)

Example: Computation (contd.)

$d = 1$

$\theta = \textbf{vertical}(90°)$

Image

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

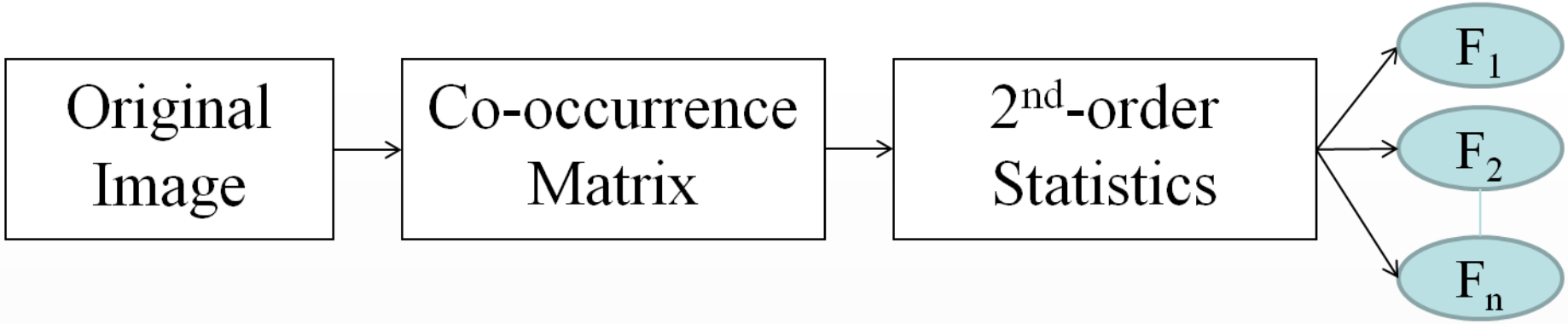
3	0	2	0
0	2	2	0
0	0	1	2
0	0	0	0

CM for the Image

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)

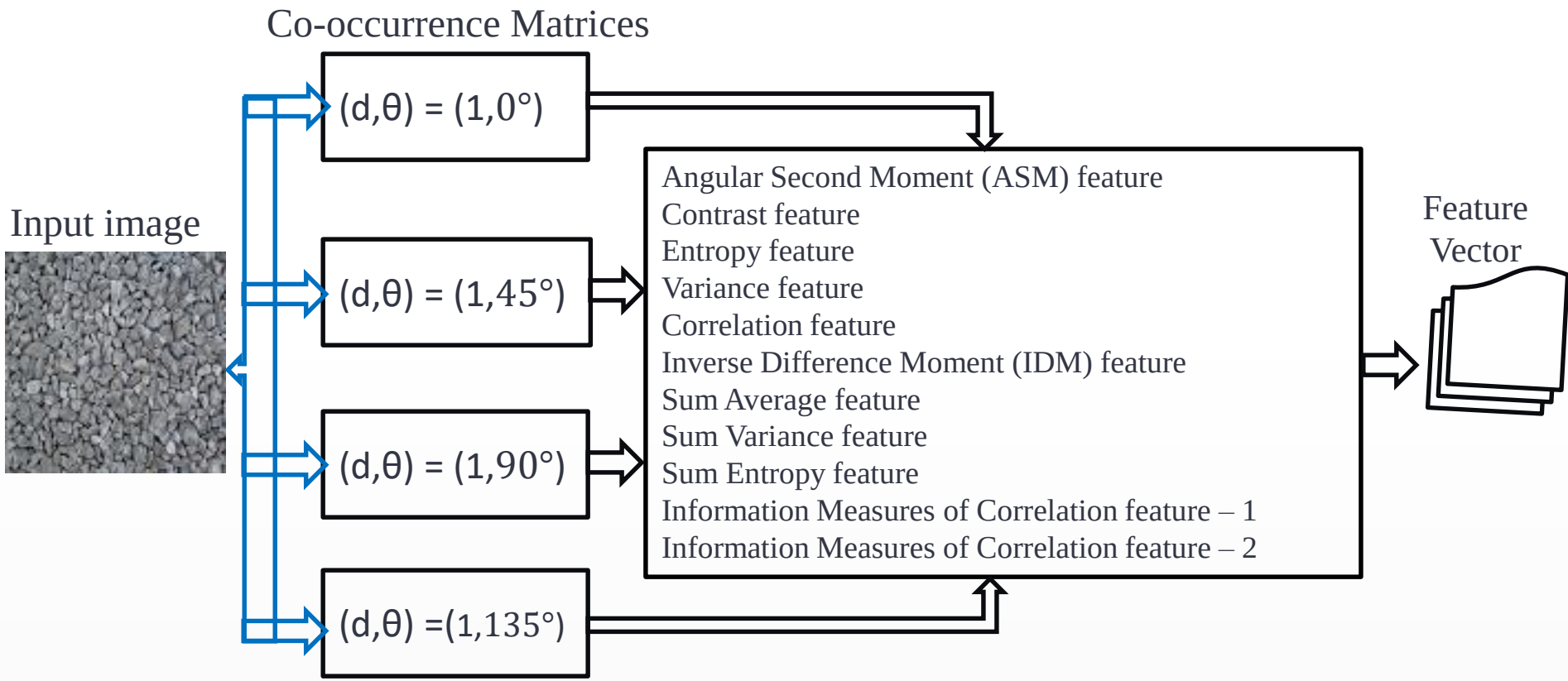
Features on co-occurrence matrix

- Co-occurrence matrices capture properties of a texture
- But they are *not directly useful* for further analysis
  (e.g., comparison of two textures)



11 Numeric features are computed from a matrix

Features on co-occurrence matrix



Energy

- Also called **Uniformity or Angular second moment**.
- **Measures 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.

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

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.

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

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) = $\sum_i \sum_j \frac{1}{1+(i-j)^2} p_{ij}$

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

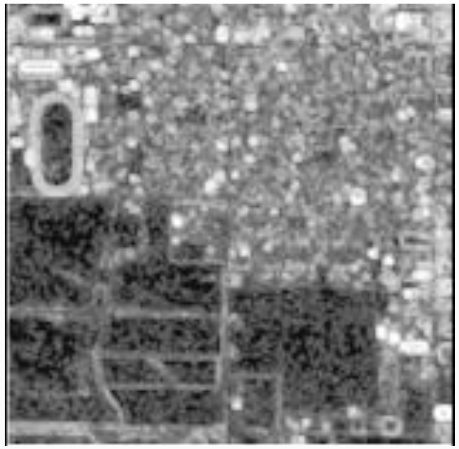
where μ is the mean of p_{ij}

GLCM Filters

- **Contrast** (Sum of Squares Variance)
  - measures gray-level contrast by using GLCM weighting factors equal to the square of the gray level difference. Thus the averaging weights are 0 for matrix position on the main diagonal and increase exponentially away from the diagonal. The filter result is 0 for areas with identical image values and is high where there are large differences in tone.

Input image

Thresholded Contrast



Contrast filter output

Features on co-occurrence matrix Examples

Feature	Comment
F2: Contrast	- Have discriminating ability. - Rotationally-variant.
F3: Entropy	- Have strong discriminating ability. - Almost rotational-invariant.
F4: Variance	- Have discriminating ability. - Rotational-invariant.
F5: Correlation	- Have strong discriminating ability. - Rotational-dependent feature.

Features on co-occurrence matrix

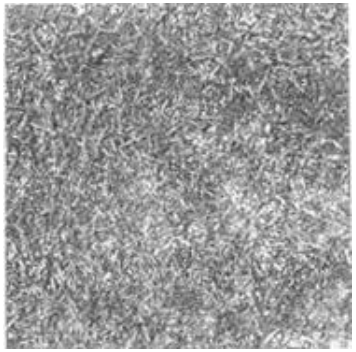
Feature	Comment
F7: Sum average	- Characteristics are similar to 'variance'/F4 - Rotational-invariant.
F10: Information Measure of Correlation–1	- It has almost similar pattern of 'sum average'/F7 but vary for various classes - Varies significantly with rotation
F11: Information Measure of Correlation–2	- It is computationally expensive compare to others. - Rotation-variant

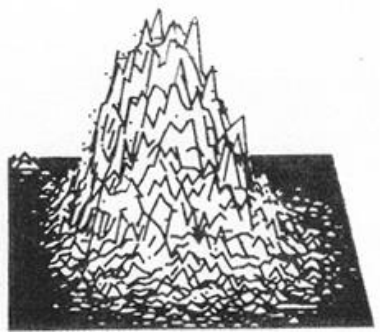
Features on co-occurrence matrix

Feature	Comment
F1: Angular Second Moment / Energy	- No distinguishing ability
F6: Inverse Different Moment	- Similar to ‘angular second moment’/F1
F8: Sum Variance	- Similar to ‘variance’/F4
F9: Sum Entropy	- Similar to ‘entropy’/F3

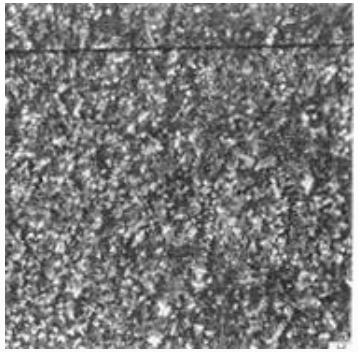
Visualization of co-occurence histograms

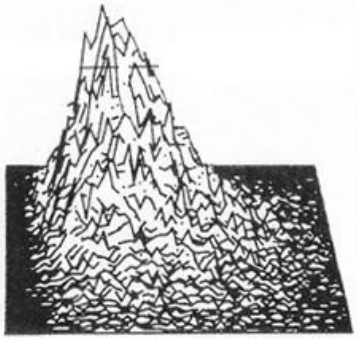
- Dependency matrices
  - co-occurrence histograms
  - calculated in a given direction, and distance
  - smoother texture implies more steady response (less dependencies)
  - Coarse texture: dominant response along the main diagonal





Grass





Ivy (borostyán)

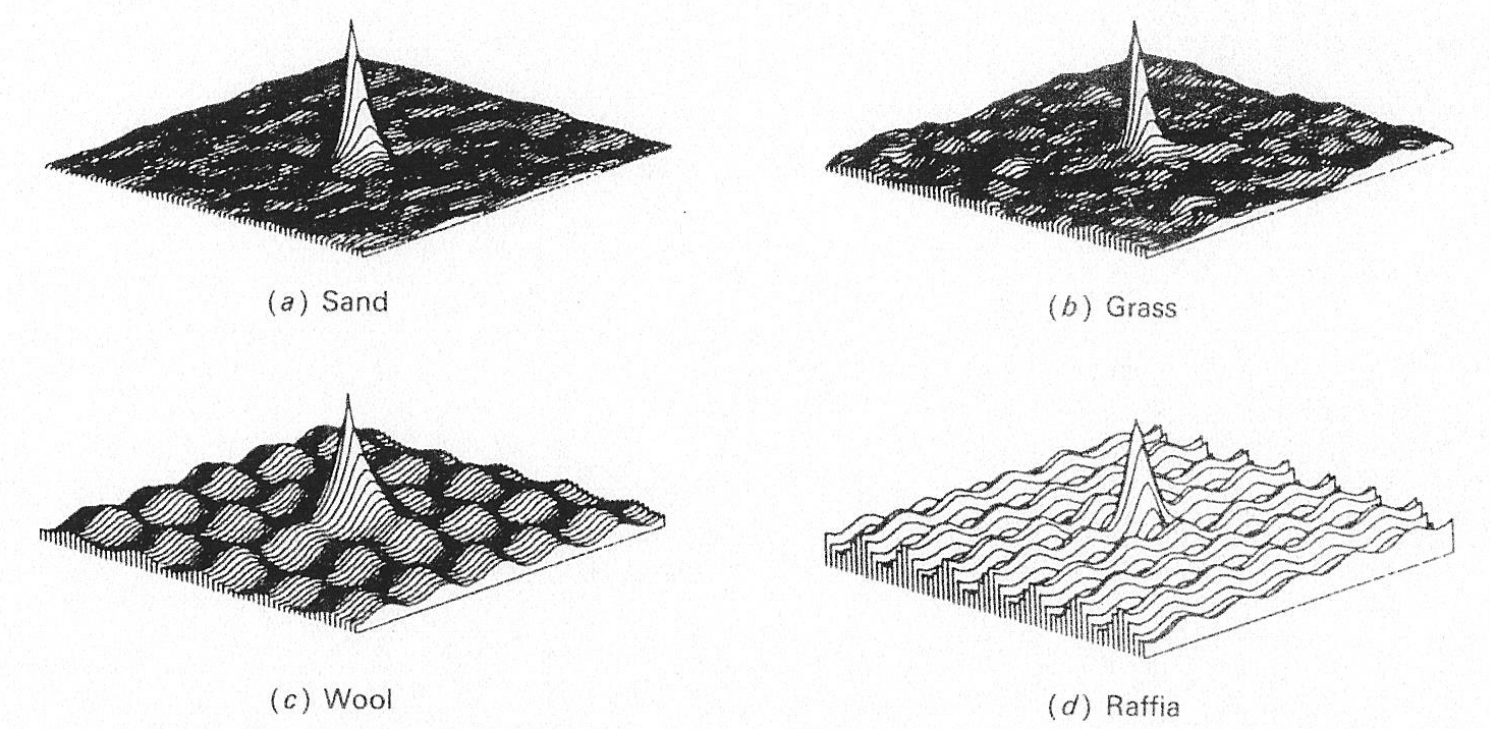
Grayscale dependency matrices for $r = 4$, $\theta = 0^{\circ}$

Statistical features
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

$$A(\Delta x, \Delta y) = \frac{\sum_{x=0}^{N} \sum_{y=0}^{N} I(x,y) I(x+\Delta x, y+\Delta y)}{\sum_{x=0}^{N} \sum_{y=0}^{N} I^2(x,y)}$$

Textures – image features

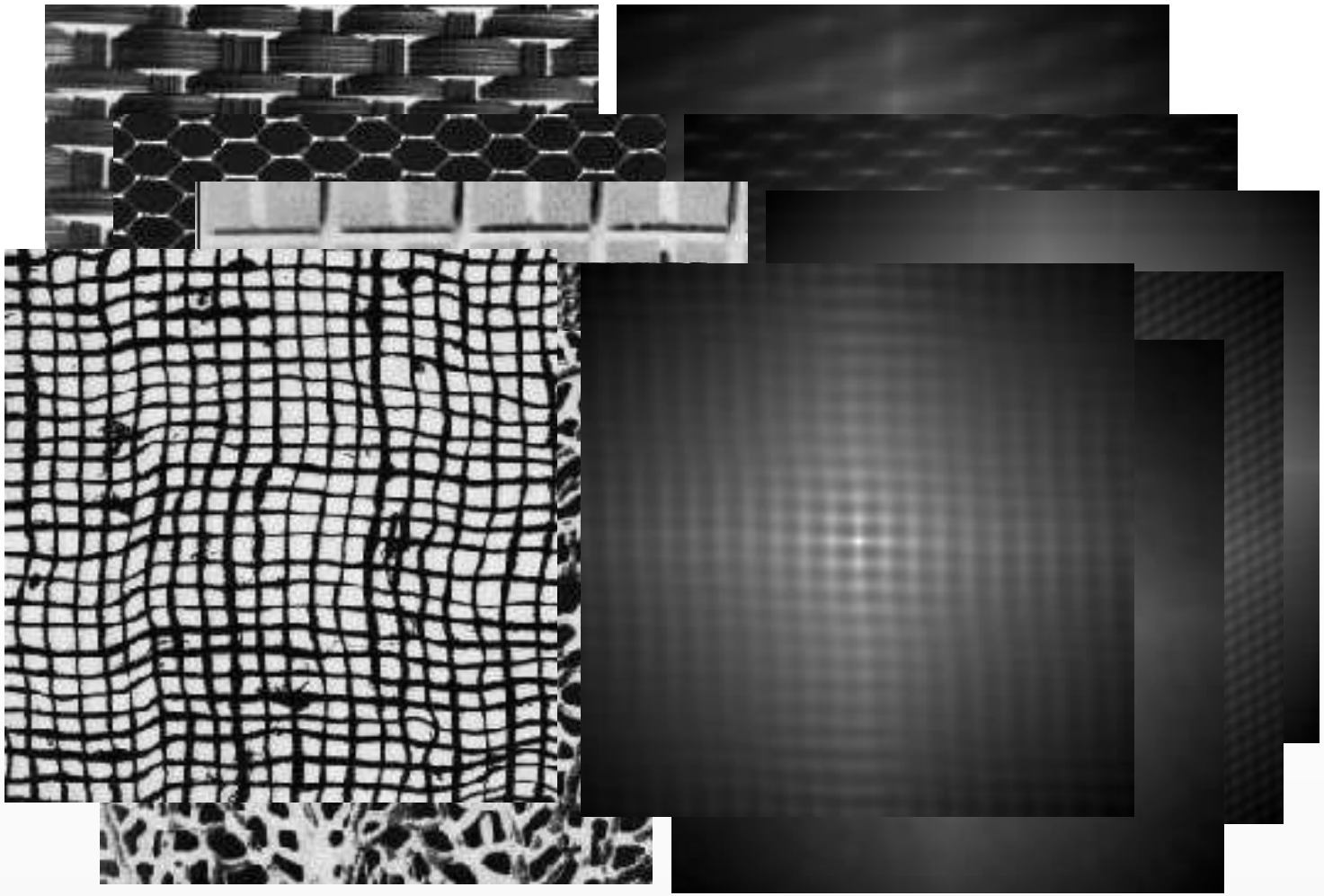


Principle: a coarse pattern texture for the same shift value shows greater autocorrelation than a finer pattern one

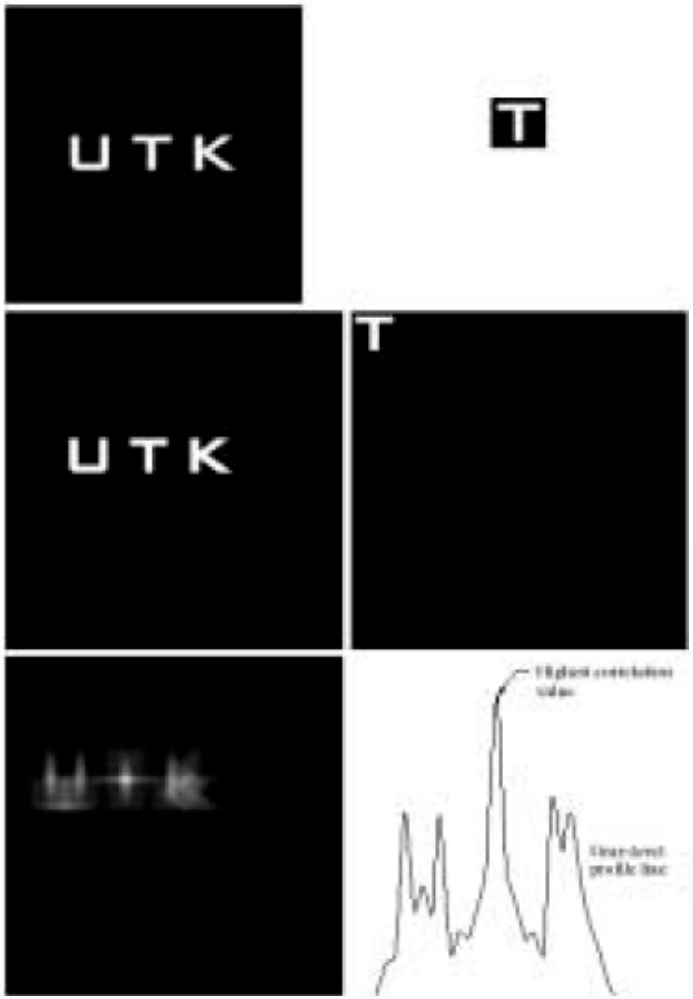
Interpreting autocorrelation

- Coarse texture → function drops off slowly
- Fine texture → function drops off rapidly
- 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

Autocorrelation



Correlation (pattern recognition)



Textures – image features

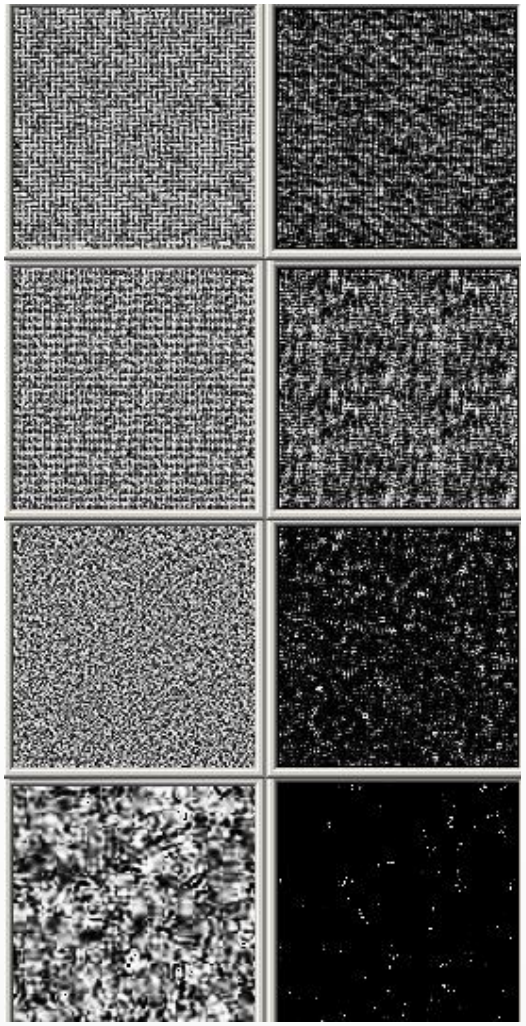
- Edge detection based procedures

E(j,k) = binary edge image obtained by some edge detector (eg. Sobel)

Use low threshold for binarization

Measure of local edge content

$$T(j,k) = \frac{1}{(2w+1)^2} \sum_{m=-w}^{w} \sum_{n=-w}^{w} E(j+m, k+n)$$



Laws filters

- Enhancing micro-structure of the texture
- Main elements:
  - Averaging
  - Edges
  - Points

$$L_3 = \frac{1}{6}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad E_3 = \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \qquad S_3 = \frac{1}{2}\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Laws filters

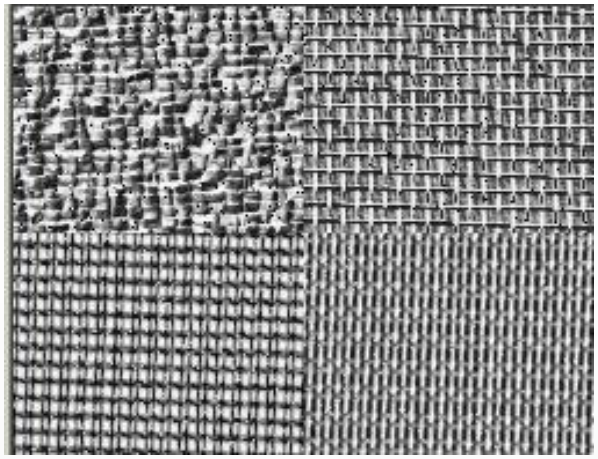
$$H_1(\,) = \frac{1}{36}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_2(\,) = \frac{1}{12}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad H_3(\,) = \frac{1}{12}\begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

$$H_4(\,) = \frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_5(\,) = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_6(\,) = \frac{1}{4}\begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

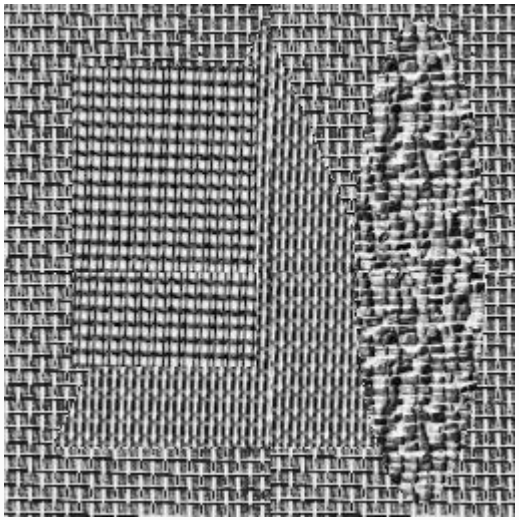
$$H_7(\,) = \frac{1}{12}\begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix} \qquad H_8(\,) = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_9(\,) = \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Texture segmentation - Laws

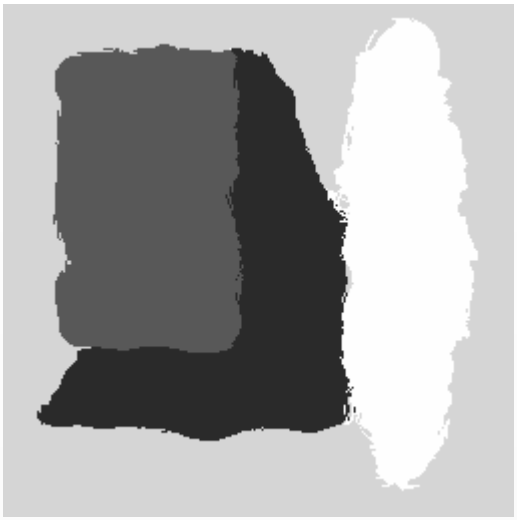
- Training step, thereafter recognizing the trained textures
- Utilizing Law matrices



Training image

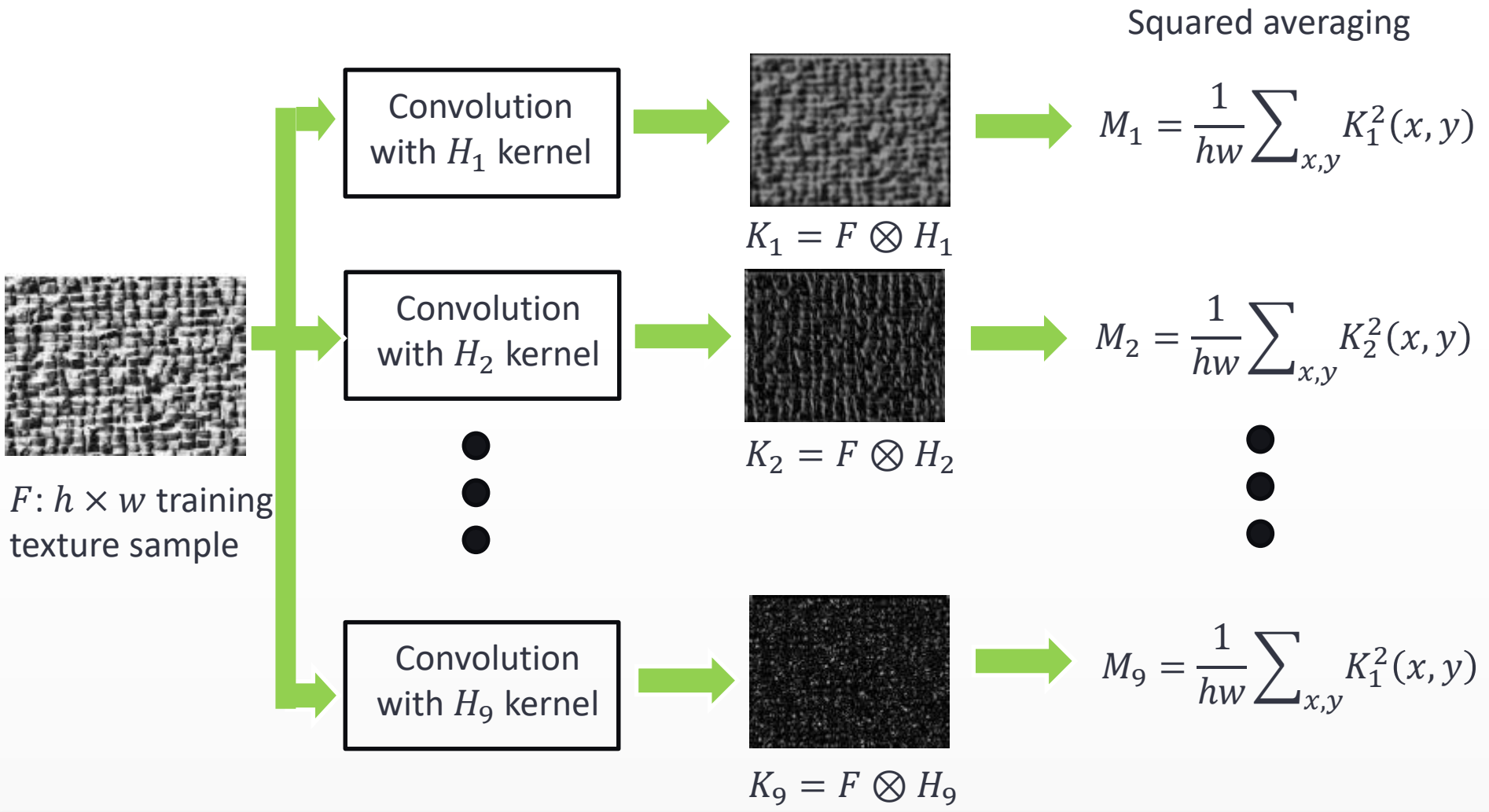


Input test image



Output image

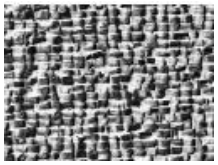
Learning a training texture model



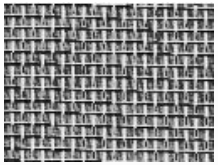
Laws filter– training phase

- Each training texture j is represented by 9 scalars: $M_1^j \ldots M_9^j$

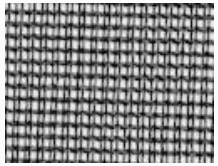
$$M_i^j = \frac{1}{hw}\sum_{x,y}\left(K_i^j(\mathrm{x},\mathrm{y})\right)^2, \text{ where } i = 1,\ldots,9, j = 1,\ldots 4, \text{ and } K_i^j = F_j \otimes K_i$$



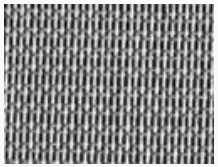
$$M_1^1, M_2^1, M_3^1, M_4^1, M_5^1, M_6^1, M_7^1, M_8^1, M_9^1$$



$$M_1^2, M_2^2, M_3^2, M_4^2, M_5^2, M_6^2, M_7^2, M_8^2, M_9^2$$



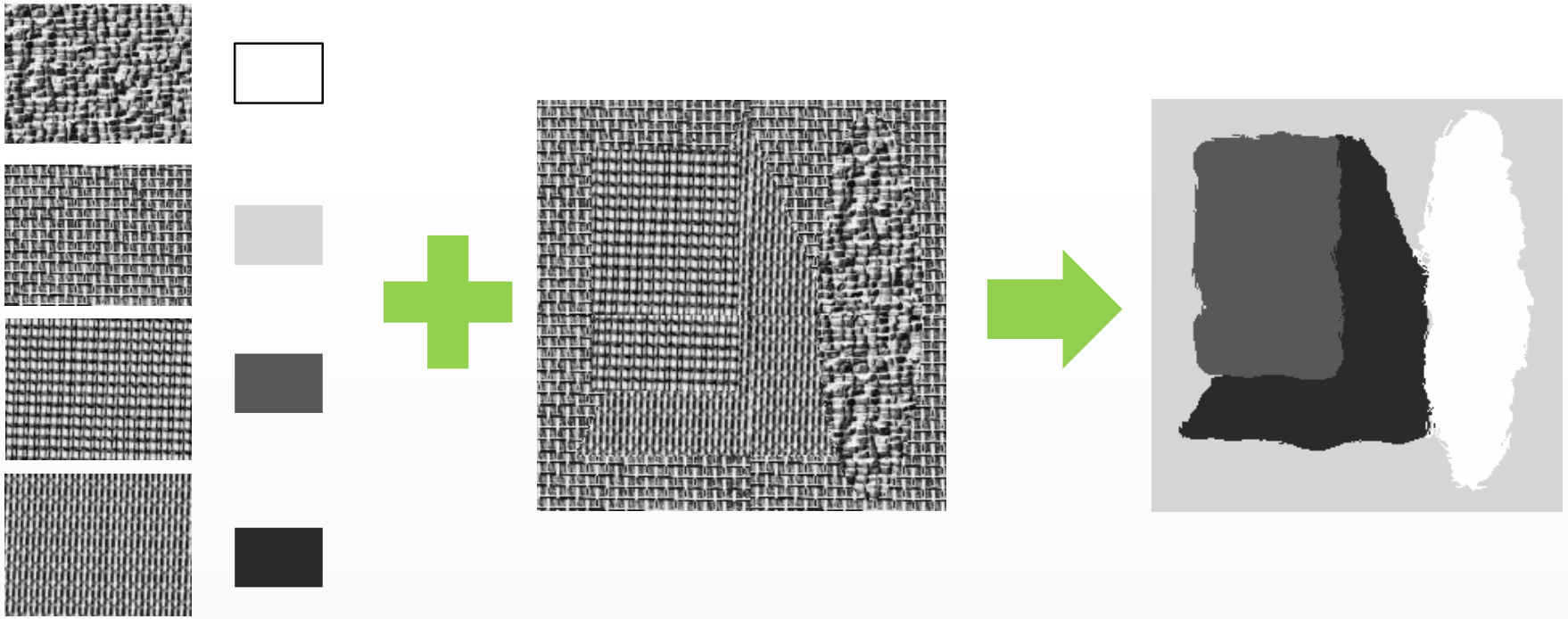
$$M_1^3, M_2^3, M_3^3, M_4^3, M_5^3, M_6^3, M_7^3, M_8^3, M_9^3$$



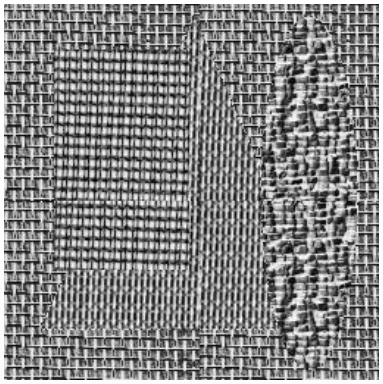
$$M_1^4, M_2^4, M_3^4, M_4^4, M_5^4, M_6^4, M_7^4, M_8^4, M_9^4$$

Laws filter– recognition phase

- Input image: consists of arbitrary regions of pre-trained textures

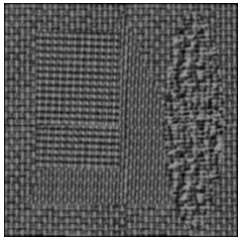


Laws filter– recognition phase



I: input image texture sample

Convolution with H_1 kernel

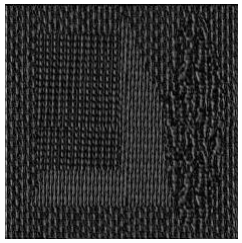


$P_1 = I \otimes H_1$

3×3 "Squared" blurring

P_1^*

Convolution with H_2 kernel

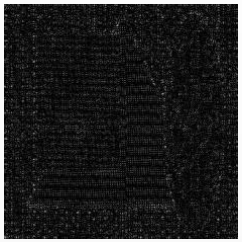


$P_2 = I \otimes H_2$

3×3 "Squared" blurring

P_2^*

Convolution with H_9 kernel



$P_9 = I \otimes H_9$

3×3 "Squared" blurring

P_9^*

Laws filter– recognition phase

- Pixel level decision of the input map

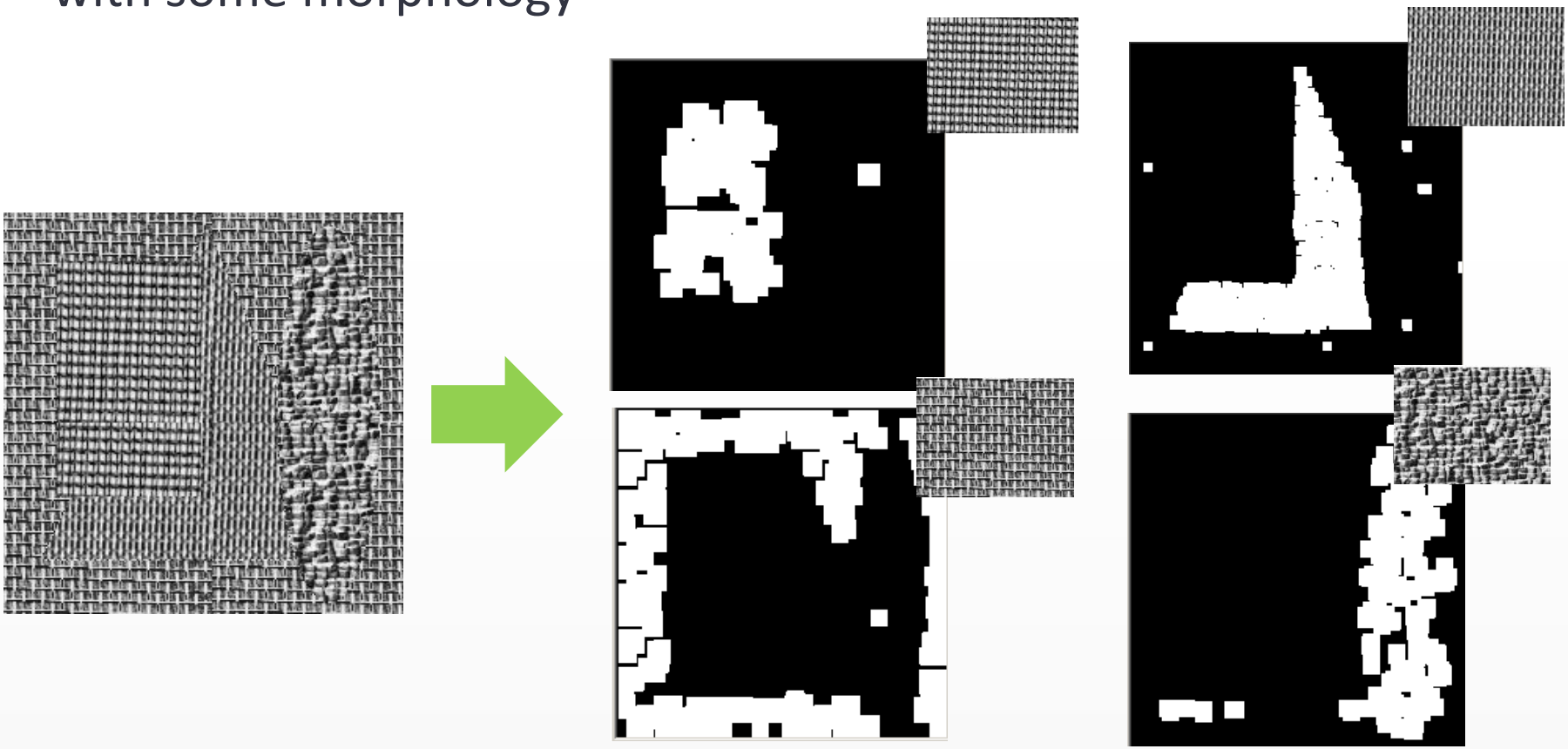
$$\mathrm{ClassMap}(x,y) = \underset{j=1\ldots4}{\operatorname{argmin}} \sum_{i=1\ldots9} \left| P_i^*(x,y) - M_i^j \right|$$

where

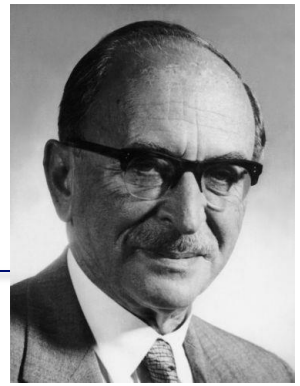
$$P_i^*(x,y) = \frac{1}{9} \sum_{r=x-1}^{x+1} \sum_{s=y-1}^{y+1} P_i^2(x+r, y+s)$$

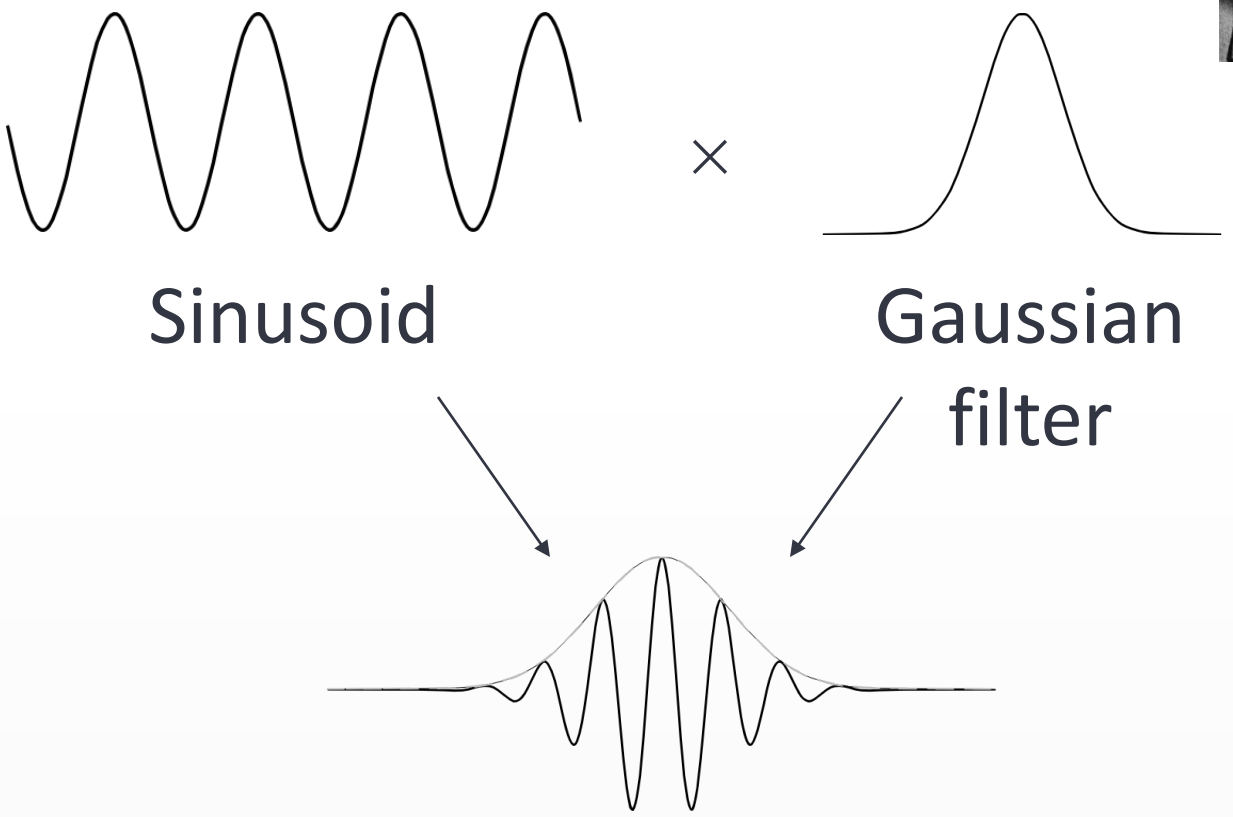
Texture segmentation result

- Output „winner class” maps for the four textures – enhanced with some morphology

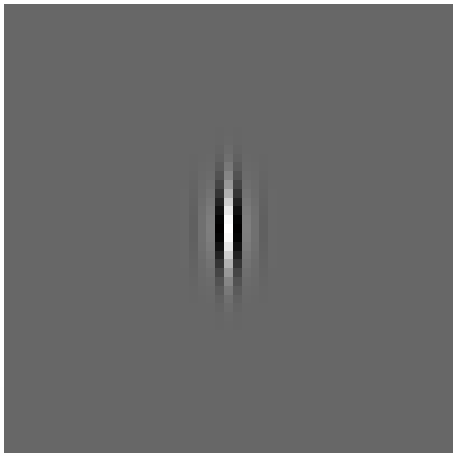


Gabor filters – 1D illustration



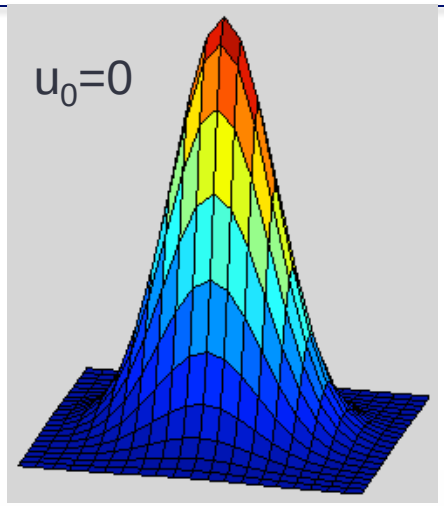


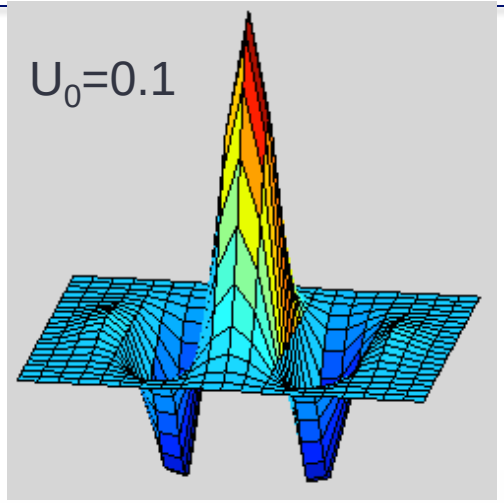
Gabor filters- 2D kernel example

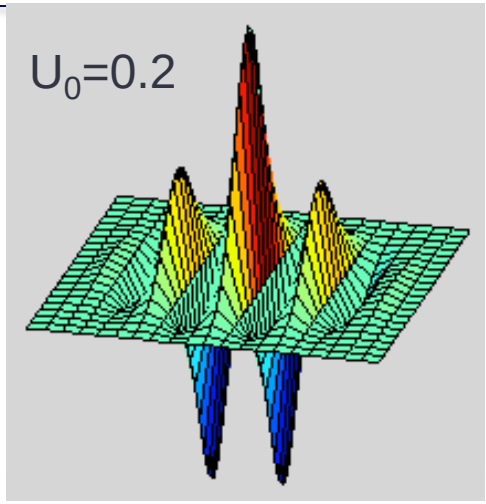


2D - Gabor wavelets

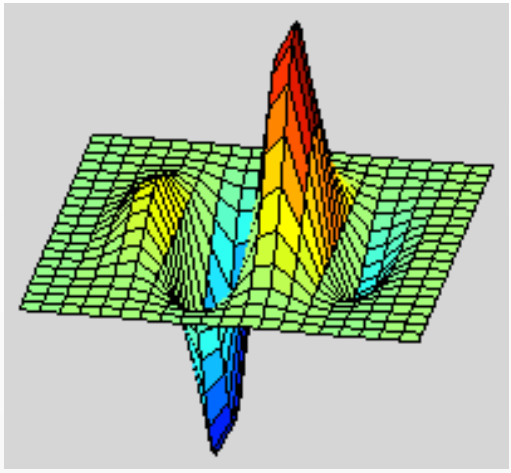
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

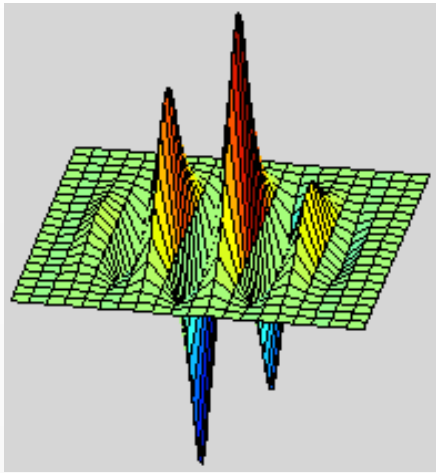


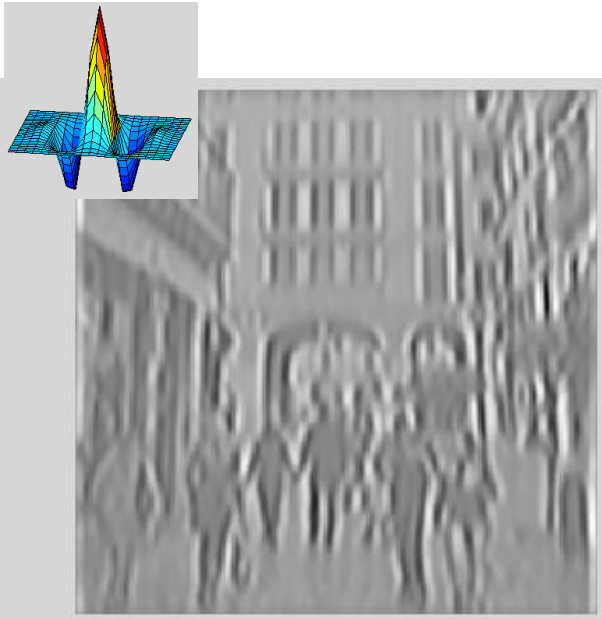




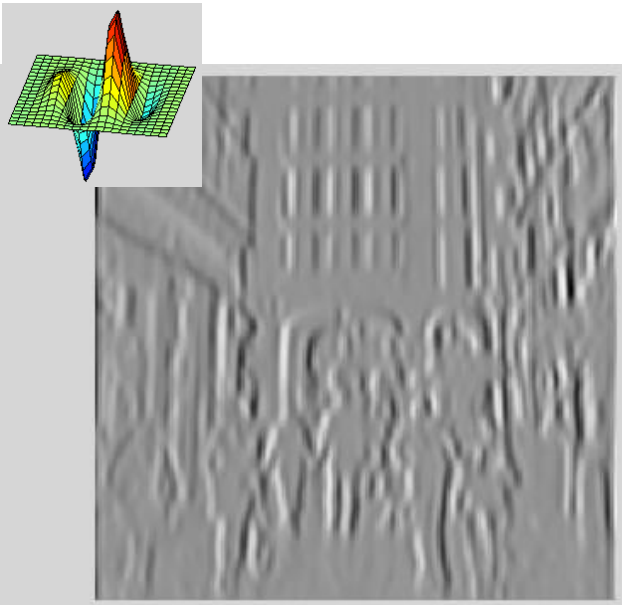
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$



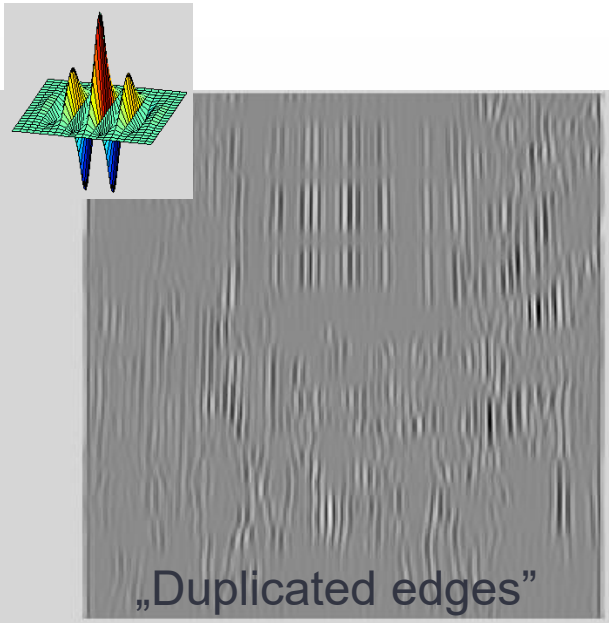




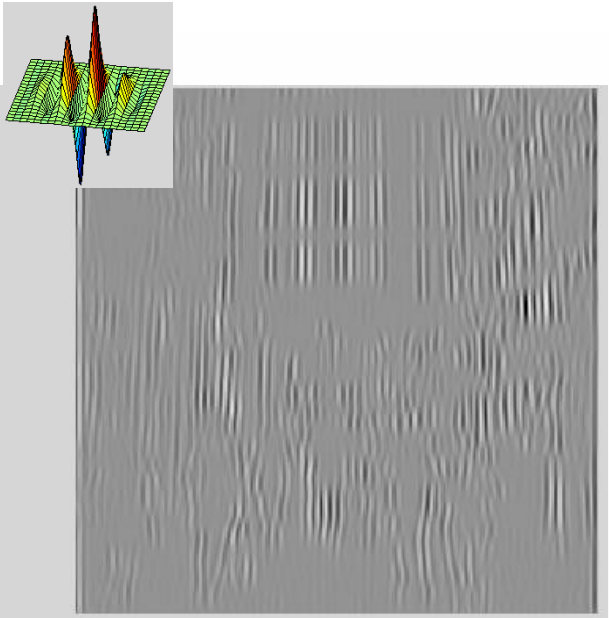
Salient bright thin vertical lines



Dark-bright transitions in horizontal direction



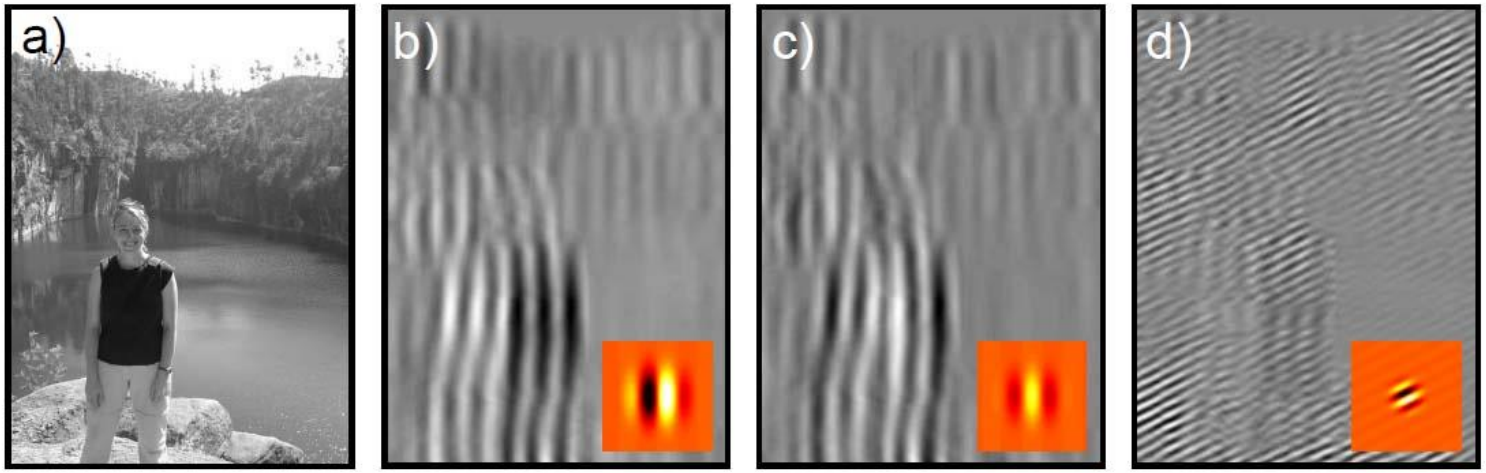
„Duplicated edges”



Application of Gabor filters in image processing

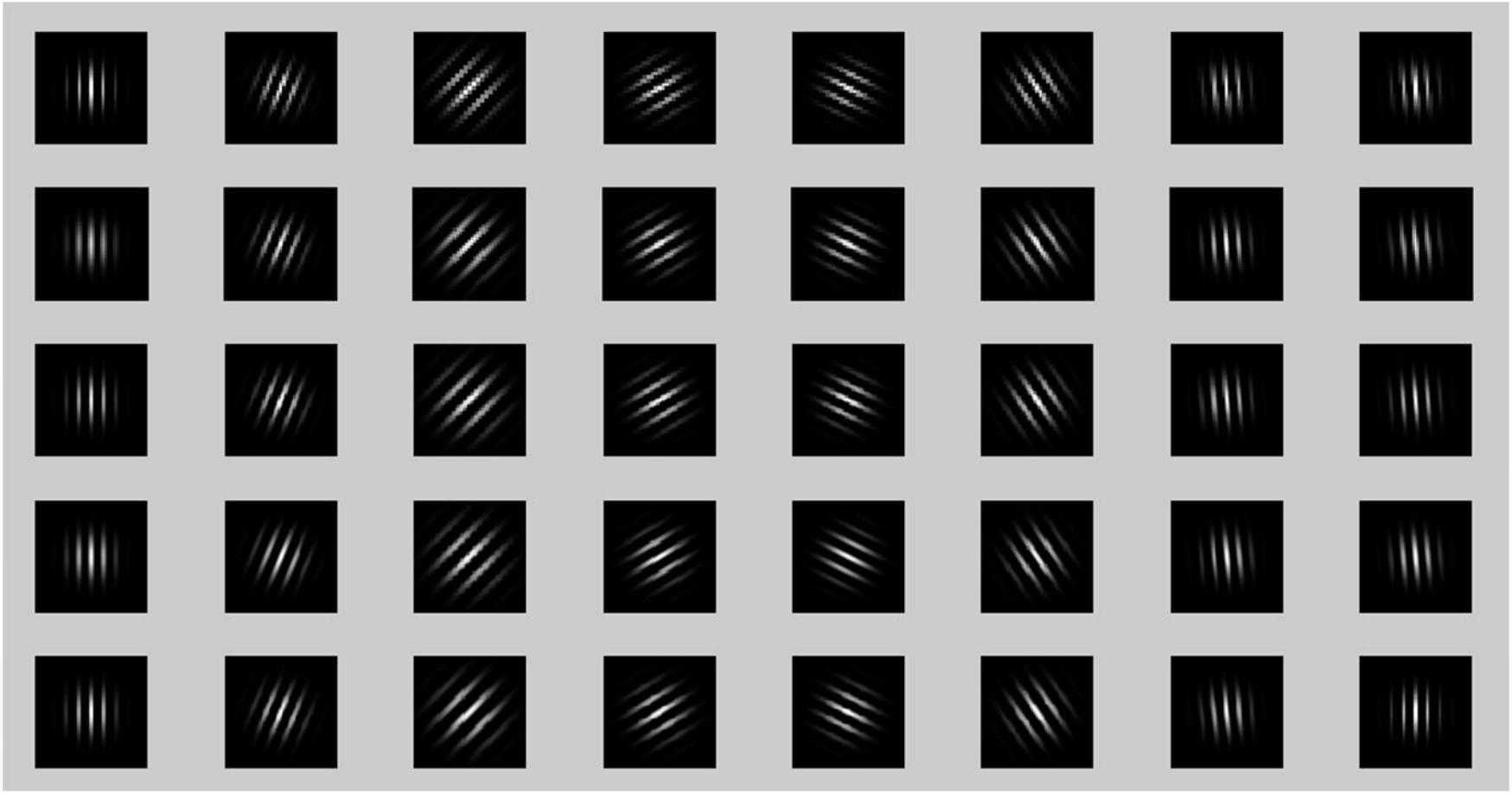
- Gabor filters can selectively highlight specific image elements according to their appropriately set frequency, orientation and phase parameters
- Invariant for additive changes of illumination (in case of asymmetric sinusoid functions)
- Motivation: vision mechanism of mammals – one of the first processing operations of visual stimuli in the brain

Application of Gabor filters in image processing



a) Input image. b) Output of a low frequency, horizontal, asymetric Gabor filter. c) Output of a low frequency, horizontal, symetric Gabor filter d) Output of a diagonal Gabor filter

Direction selective Gabor filter bank



Texture Databases

- Categorize them in to *four areas*
  - → Texture databases in *medical* imaging
  - → *Natural* texture image database
  - → Texture of *materials* database
  - → *Dynamic* texture database

Databases: Various Properties

- Image size
- No. of classes
- No. of images
- Gray-scale vs. color image
- Image rotation
- Illumination – static/varied; indoor/outdoor
- Camera/sensors
- Image depth/distance from the camera

Some Key Databases

- Brodatz texture database
- MRI brain database
- USF Digital database for screening mammography
- Vision texture database (VisTex)
- USC-SIPI texture database
- PhoTex database
- ALOT database
- UMD dataset
- CUReT database
- UIUC database
- KTH-TIPS database
- UCLA dynamic database
- DynTex database
- MIT Szummer database

Natural textures - Brodatz

- Phil Brodatz,
  - "Textures: A Photographic Album for Artists and Designers",
  - „Land Sea and Skies: A Photographic Album for Artists and Designers”,
  - „Wood and Wood Grains: A Photographic Album for Artists and Designers”,
  - „The Elements of Landscape: A Photographic Handbook for Artists”