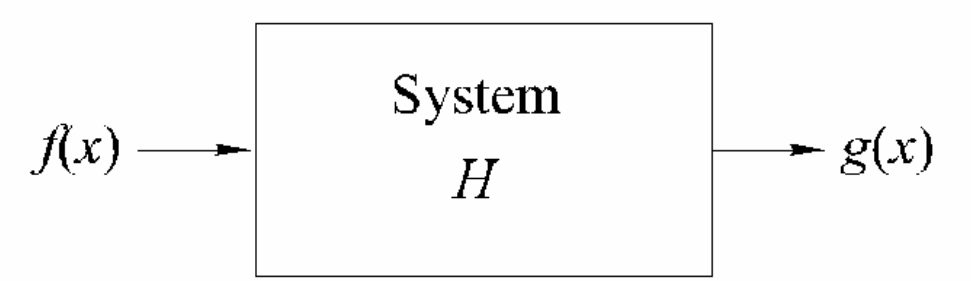
Introduction to Computer Vision

2D Linear Systems

Review: Linear Systems

- We define a system as a unit that converts an input function into an output function

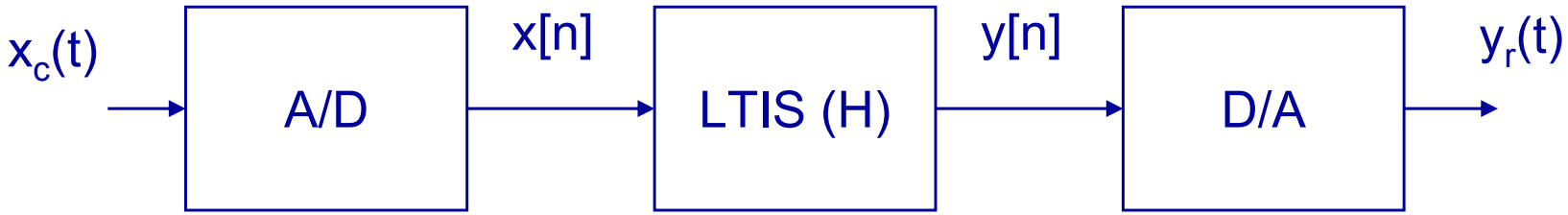


$$g(x) = H[f(x)]$$

Independent variable

System operator or Transfer function

Linear Time Invariant Discrete Time Systems



$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$

$$Y_r(j\Omega) = H(j\Omega)X_c(j\Omega)$$

$$H(j\Omega) = \begin{cases} H(j\Omega) & |\Omega| < \pi/T \\ 0 & |\Omega| \geq \pi/T \end{cases}$$

IF
- The input signal is bandlimited
- The Nyquist condition for sampling is met
- The digital system is linear and time invariant

THEN
The overall continuous time system is equivalent to a LTIS whose frequency response is H.

Overview of Linear Systems

- Let $$g_i(x) = H[f_i(x)]$$

  where $f_i(x)$ is an arbitrary input in the class of all inputs $\{f(x)\}$, and $g_i(x)$ is the corresponding output.

- If $$\begin{aligned} H[a_i f_i(x) + a_j f_j(x)] &= a_i H[f_i(x)] + a_j H[f_{ji}(x)] \\ &= a_i g_i(x) + a_j g_j(x) \end{aligned}$$

  Then the system H is called a *linear system*.

- A linear system has the properties of *additivity* and *homogeneity*.

Linear Systems

- The system H is called *shift invariant* if

$$g_i(x) = H[f_i(x)] \text{ implies that } g_i(x + x_0) = H[f_i(x + x_0)]$$

for all $f_i(x) \in \{f(x)\}$ and for all x_0.

- This means that offsetting the independent variable of the input by x_0 causes the same offset in the independent variable of the output. Hence, the input-output relationship remains the same.

Linear Systems

- The operator *H* is said to be *causal*, and hence the system described by *H* is a *causal system*, if there is no output before there is an input. In other words,

$$f(x) = 0 \text{ for } x < x_0 \text{ implies that } g(x) = H[f(x)] = 0 \text{ for } x < x_0.$$

- A linear system *H* is said to be *stable* if its response to any bounded input is bounded. That is, if

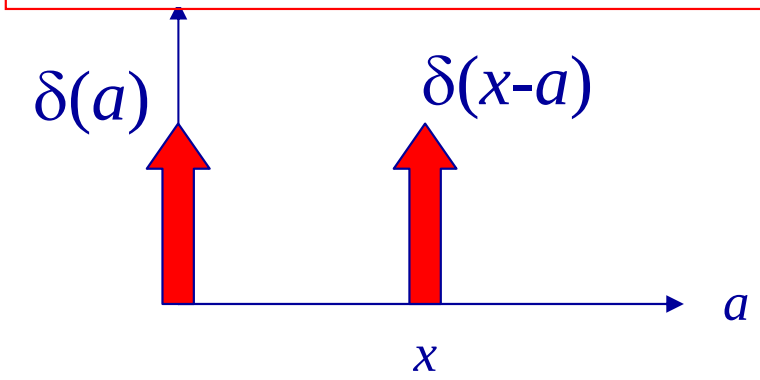
$$|f(x)| < K \text{ implies that } |g(x)| < cK$$

where *K* and *c* are constants.

Linear Systems

- A *unit impulse function*, denoted $\delta(a)$, is *defined* by the expression

$$\int_{-\infty}^{\infty} f(\alpha)\delta(x-\alpha)d\alpha = f(x).$$



- The response of a system to a unit impulse function is called the *impulse response* of the system.

$$h(x) = H[\delta(x)]$$

Linear Systems

- If *H* is a linear shift-invariant system, then we can find its reponse to any input signal f(x) as follows:

$$g(x) = \int_{-\infty}^{\infty} f(\alpha)h(x-\alpha)d\alpha.$$

- This expression is called the *convolution integral*. It states that the response of a linear, fixed-parameter system is completely characterized by the convolution of the input with the system impulse response.

Linear Systems

- Convolution of two functions of a continuous variable is defined as

$$f(x) * h(x) = \int_{-\infty}^{\infty} f(\alpha) h(x - \alpha) d\alpha$$

- In the discrete case

$$f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[m] h[n - m]$$

Linear Systems

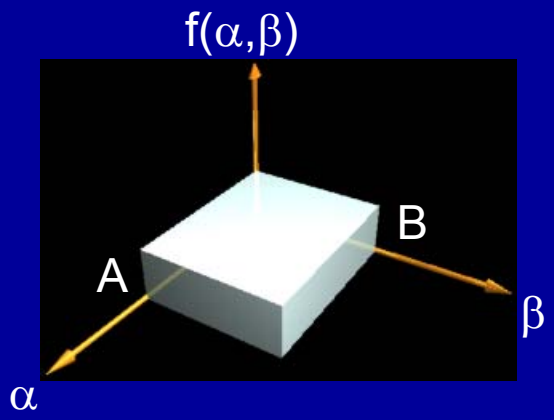
- In the 2D discrete case

$$f[n_1, n_2] ** h[n_1, n_2] = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f[m_1, m_2] h[n_1 - m_1, n_2 - m_2]$$

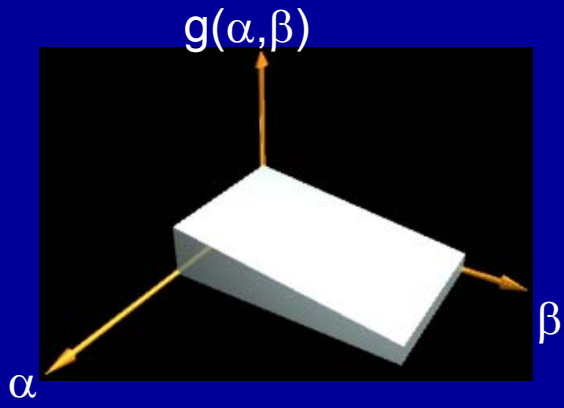
$h[n_1, n_2]$ is a linear filter.

Illustration of the folding, displacement, and multiplication steps needed to perform two-dimensional convolution

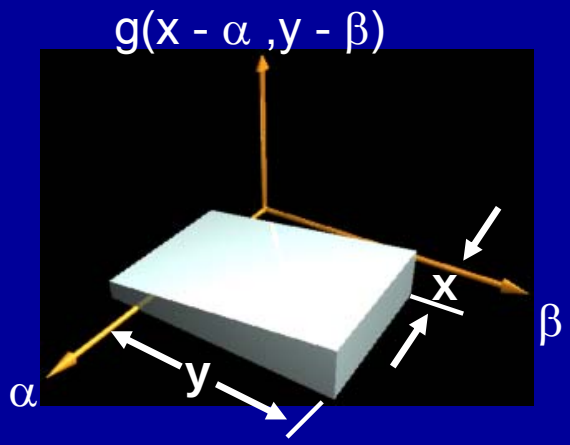
(a)



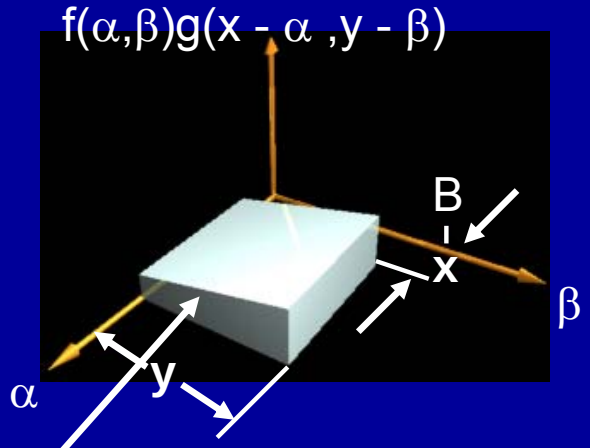
(b)



(c)

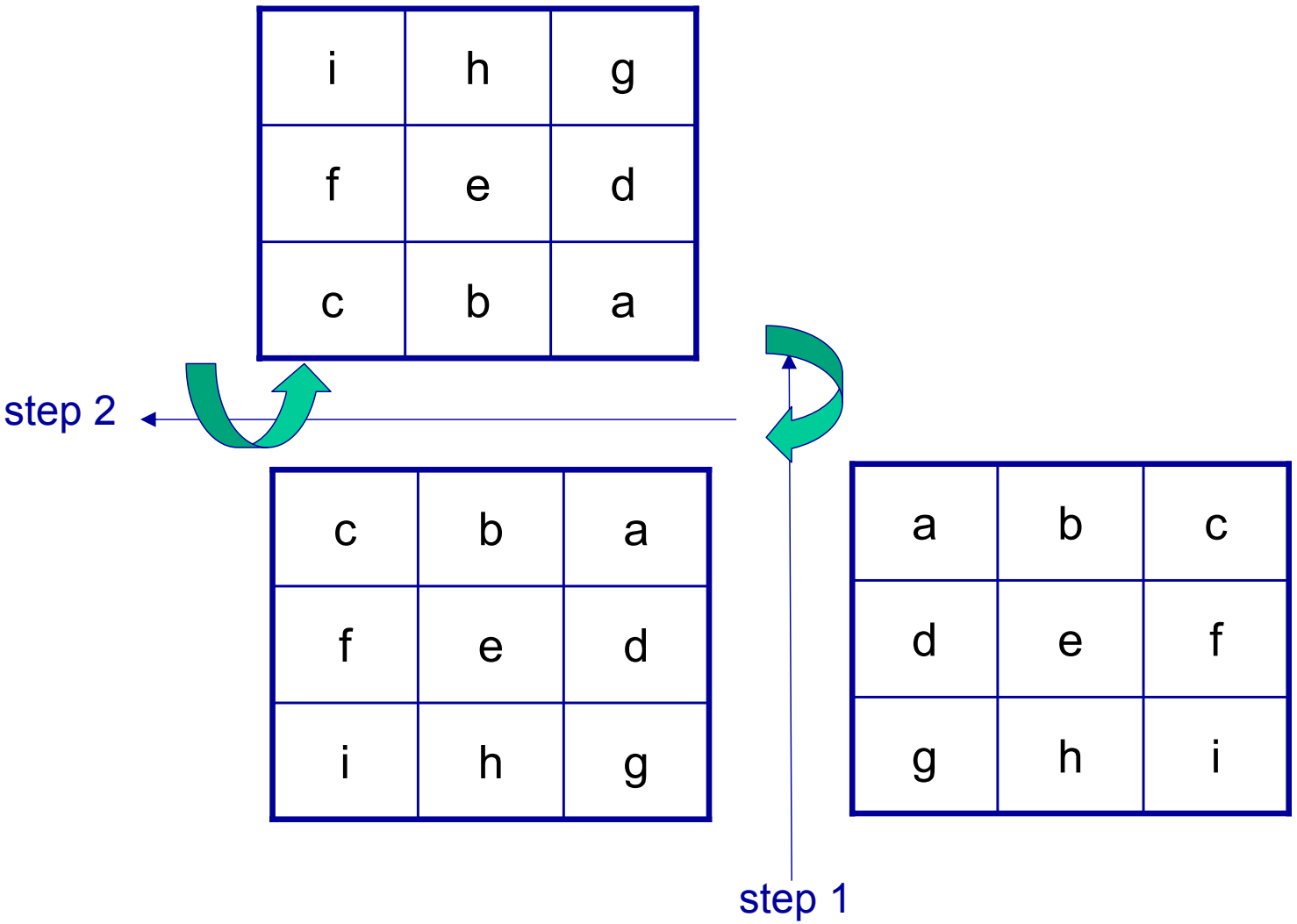


(d)



Volume = f(x,y) * g(x,y)

Matrix perspective



Convolution Example

h

1	-1	-1
1	2	-1
1	1	1

Rotate

1	1	1
-1	2	1
-1	-1	1

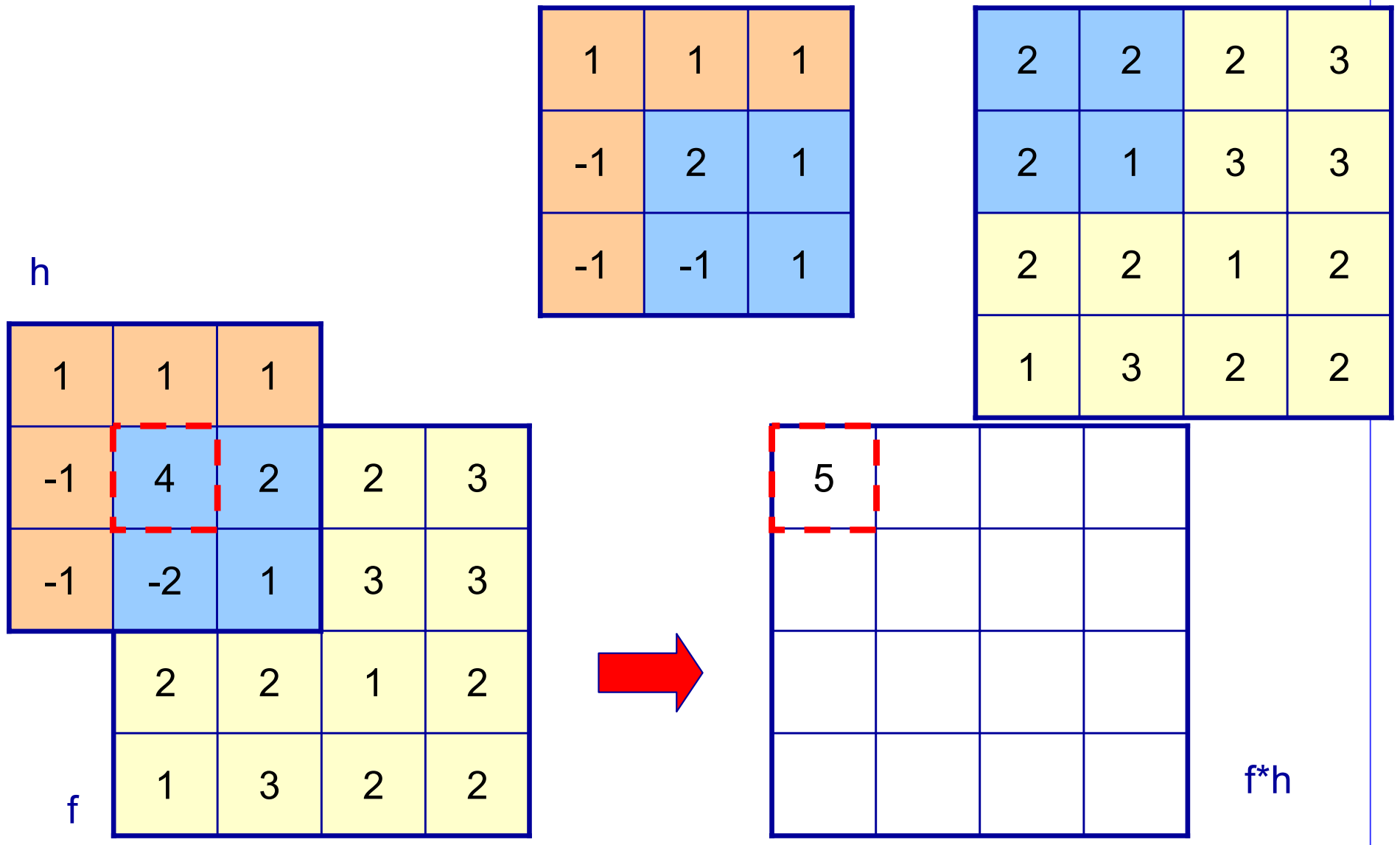
$$f[n_1,n_2]**h[n_1,n_2]=\sum_{m_1=-\infty}^{\infty}\sum_{m_2=-\infty}^{\infty}f[m_1,m_2]h[n_1-m_1,n_2-m_2]$$

f

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

From C. Rasmussen, U. of Delaware

Convolution Example

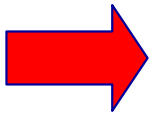


Convolution Example

h

1	1	1
-1	2	1
-1	-1	1

1	1	1	
-2	4	2	3
-2	-1	3	3
2	2	1	2
1	3	2	2



2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

5	4		

f*h

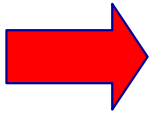
Convolution Example

1	1	1
-1	2	1
-1	-1	1

h

1	1	1	
2	-2	4	3
2	-1	-3	3
2	2	1	2
1	3	2	2

f

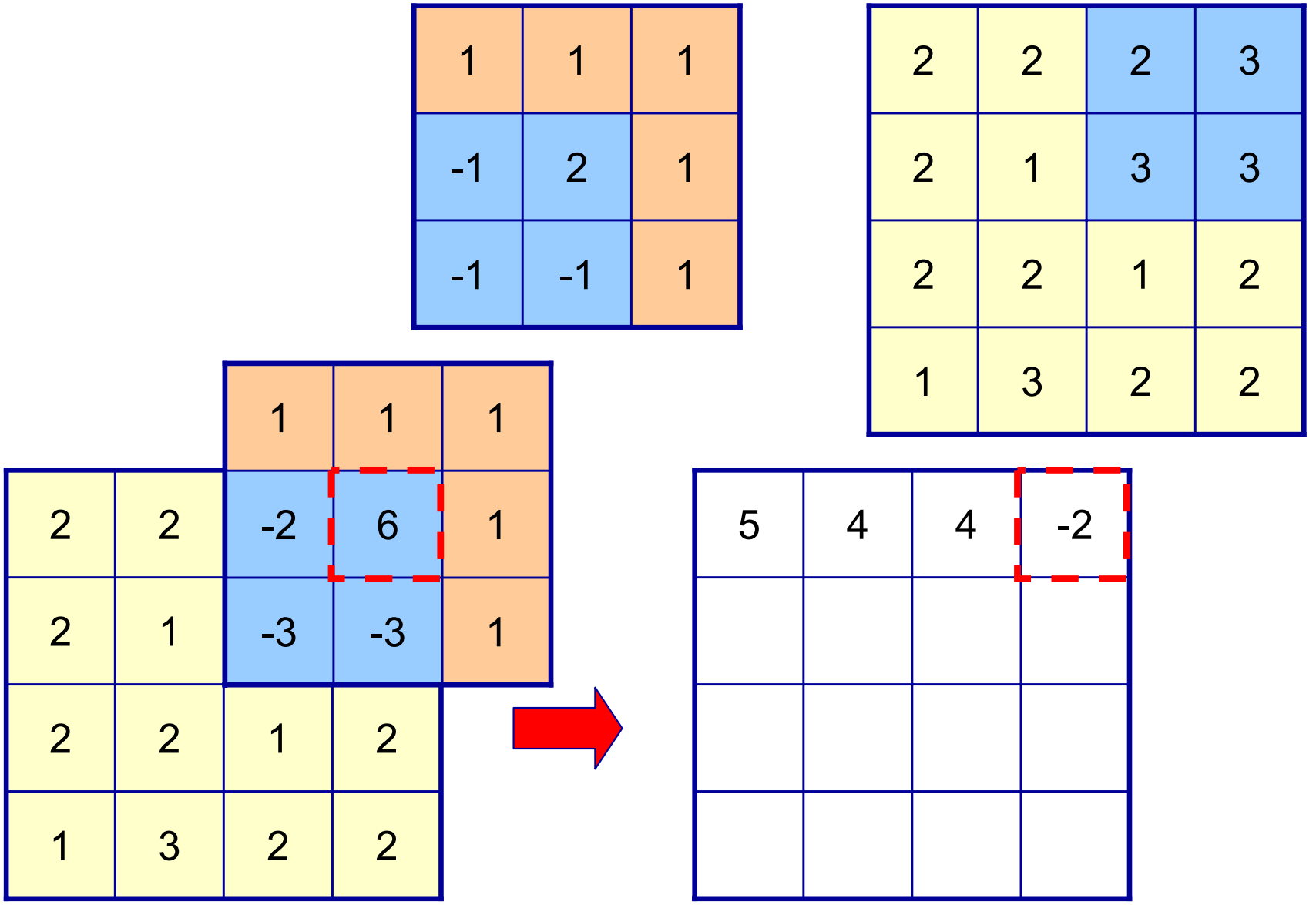


5	4	4	

f*h

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Convolution Example



Convolution Example

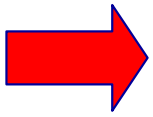
1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

h

1	2	2	2	3
-1	4	1	3	3
-1	-2	2	1	2
	1	3	2	2

f



5	4	4	-2
9			

f*h

Convolution Example

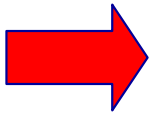
1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

h

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2

f



5	4	4	-2
9	6		

f*h

Convolution Example

and so on…

From C. Rasmussen, U. of Delaware

Example: averaging

$$* \quad \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad =$$

Integration

Example: edge detection

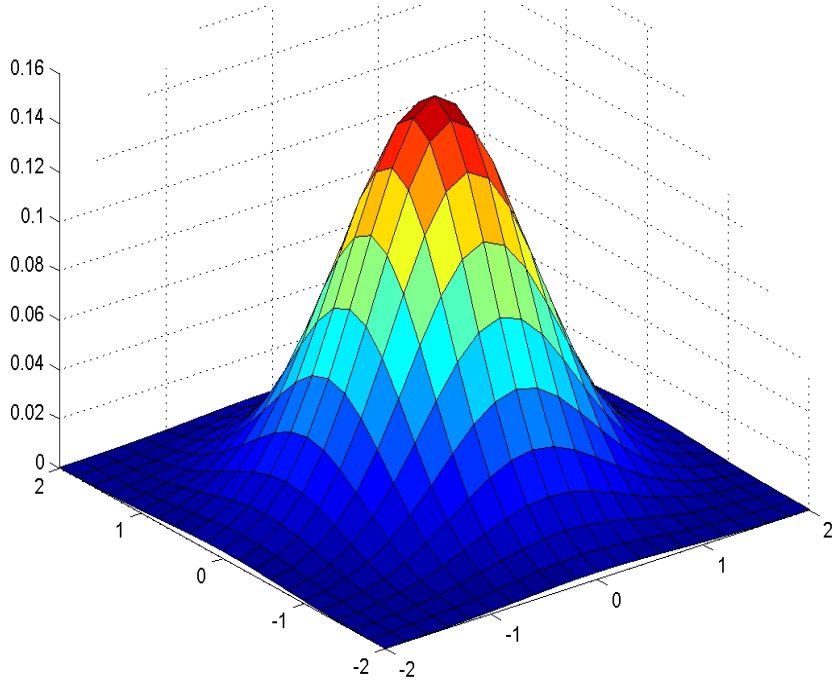
$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Deriving

Try MATLAB

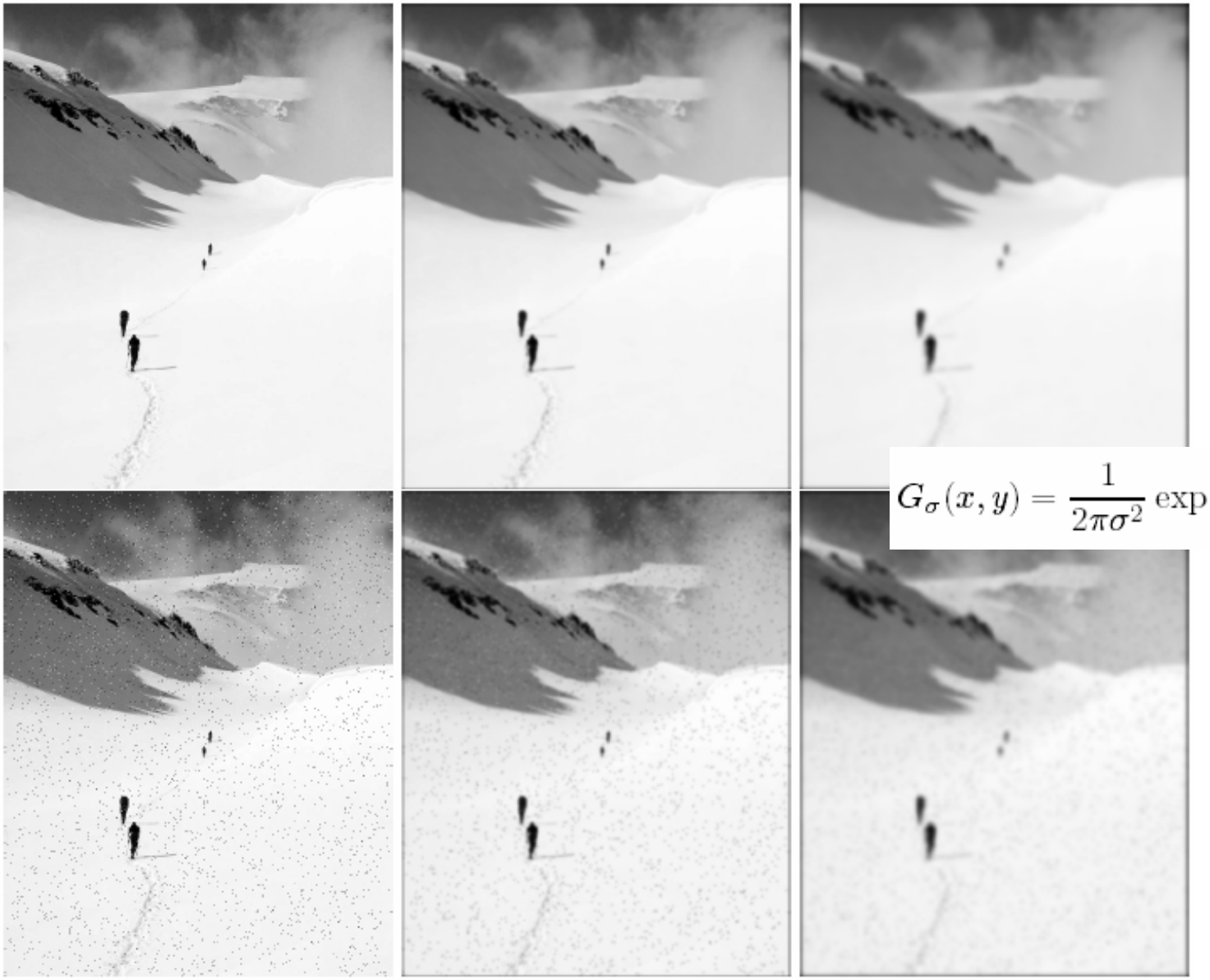
```
f=imread('saturn.tif');
figure; imshow(f);
[height,width]=size(f);
f2=f(1:height/2,1:width/2);
figure; imshow(f2);
[height2,width2=size(f2);
f3=double(f2)+30*rand(height2,width2);
figure;imshow(uint8(f3));
h=[1 1 1 1; 1 1 1 1; 1 1 1 1; 1 1 1 1]/16;
g=conv2(f3,h);
figure;imshow(uint8(g));
```

Gaussian Lowpass Filter



$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

Gaussian Lowpass Denoising



$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

Original σ = 2 σ = 4

Image Processing Algorithms

IP Algorithms

Spatial domain

- Operations are performed in the image domain
- Image ⇔ matrix of number
- Examples
  - luminance adaptation
  - chromatic adaptation
  - contrast enhancement
  - spatial filtering
  - edge detection
  - noise reduction

Transform domain

- Some operators are used to project the image in another space
- Operations are performed in the transformed domain
  - Fourier (DCT, FFT)
  - Wavelet (DWT,CWT)
- Examples
  - coding
  - denoising
  - image analysis

Most of the tasks can be implemented both in the image and in the transformed domain. The choice depends on the context and the specific application.

Spatial domain processing

Pixel-wise

- Operations involve the single pixel
- Operations:
  - histogram equalization
  - change of the colorspace
  - addition/subtraction of images
  - get negative of an image
- Applications:
  - luminance adaptation
  - contrast enhancement
  - *chromatic adaptation*

Local-wise

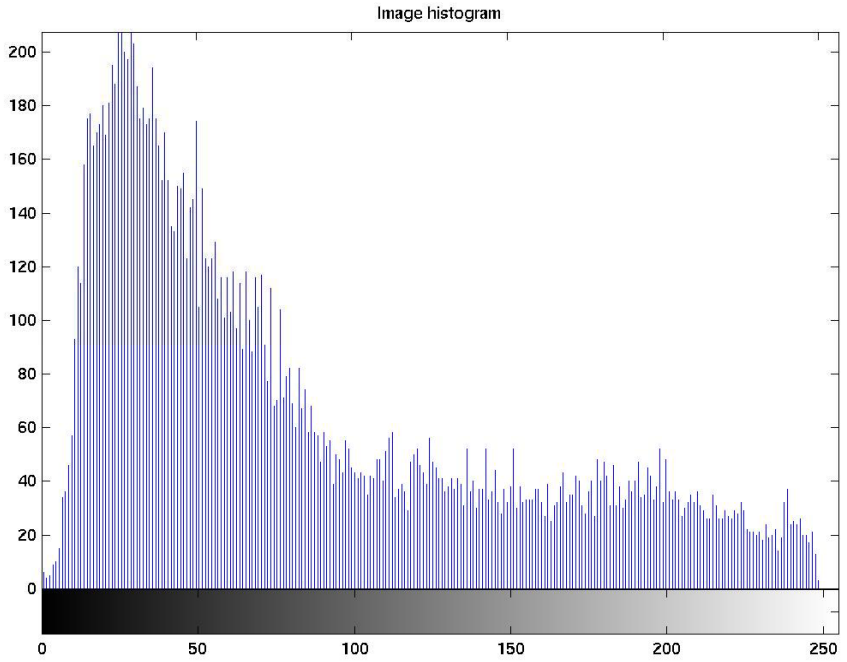
- The neighbourhood of the considered pixel is involved
  - Any operation involving digital filters is local-wise
- Operations:
  - correlation
  - convolution
  - filtering
  - transformation
- Applications
  - smoothing
  - sharpening
  - noise reduction
  - edge detection

Pixel-wise operations

- Histogram
  - Straching/shrinking, sliding
  - Equalization

Pixel-wise: Histogram equalization

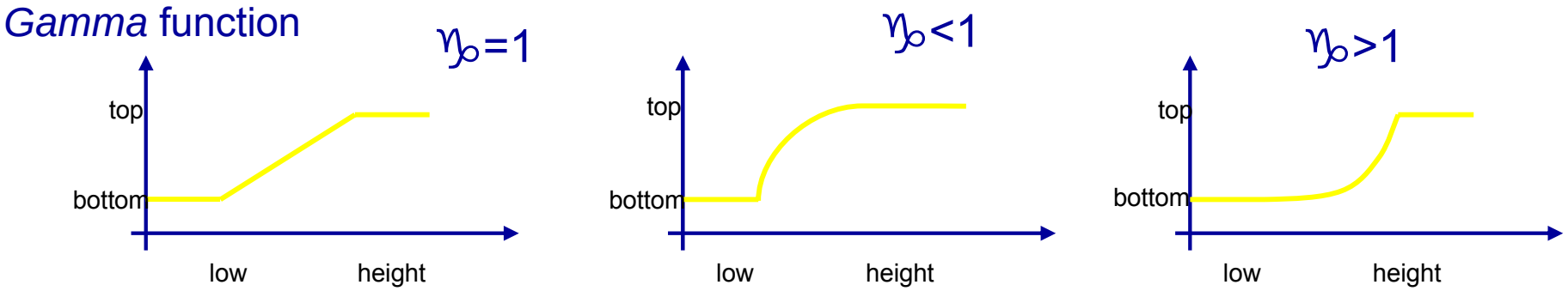
- Pixel features: luminance, color,
- Histogram equalization: shapes the intensity histogram to approximate a specified distribution
  - It is often used for enhancing contrast by shaping the image histogram to a uniform distribution over a given number of grey levels. The grey values are redistributed over the dynamic range to have a constant number of samples in each interval (i.e. histogram bin).
  - Can also be applied to colormaps of color images.

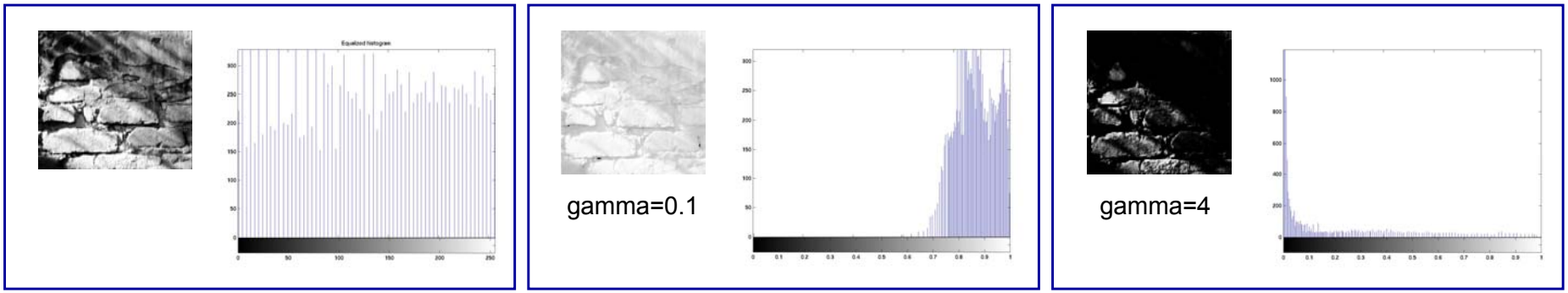


Histogram equalization

Can be used to compensate the distortions in the gray level distribution due to the non-linearity of a system component

Gamma function



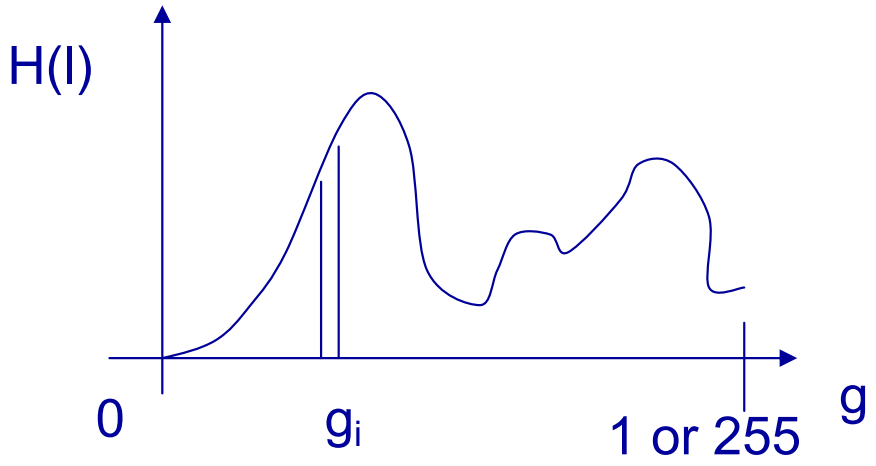


Histogram equalization

- Enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram
  - also applies to the values in the colormap of an indexed image

Histogram

- Function H=H(g) indicating the number of pixels having gray-value equal to g
  - Non-normalized images: 0≤g ≤255 → bin-size≥1, can be integer
  - Normalized images: 0≤g ≤1 → bin-size<1



$$A = \int_0^{\max} H(g)dg$$

area under the curve=number of pixels

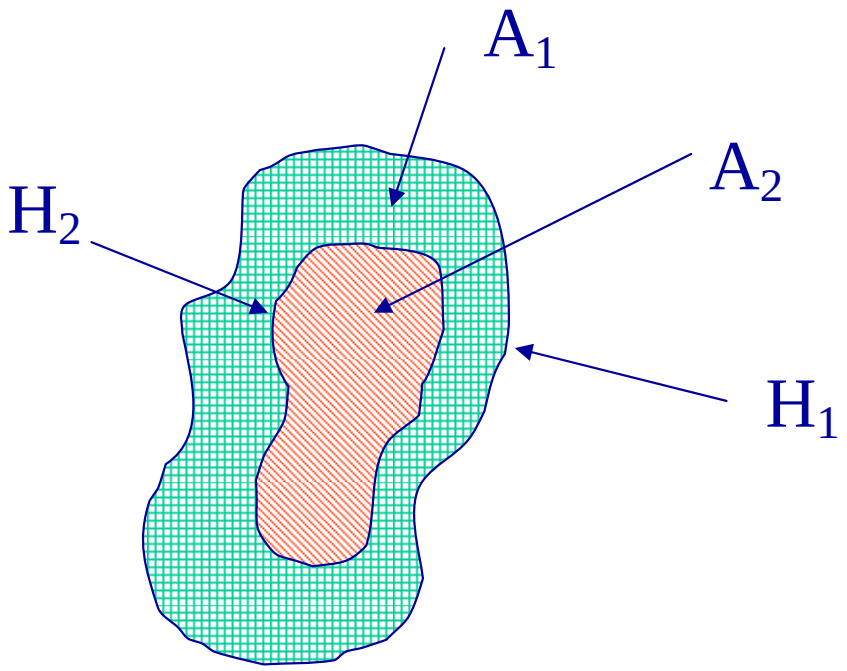
$$A = \sum_{i=1}^{N_g} H[g_i]$$

In the continuous case

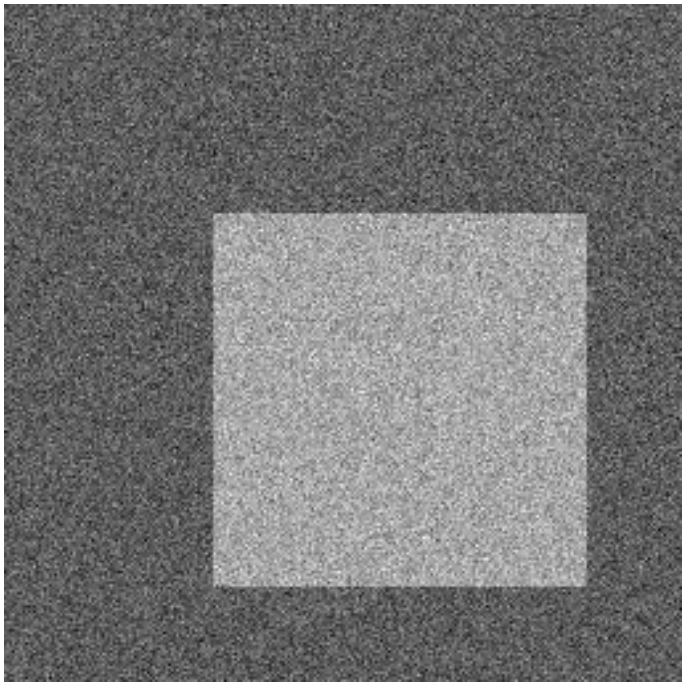
$$H(g) = -\frac{dA(g)}{dg} = \lim_{\Delta g \to 0} \frac{A(g) - A(g + \Delta g)}{\Delta g}$$

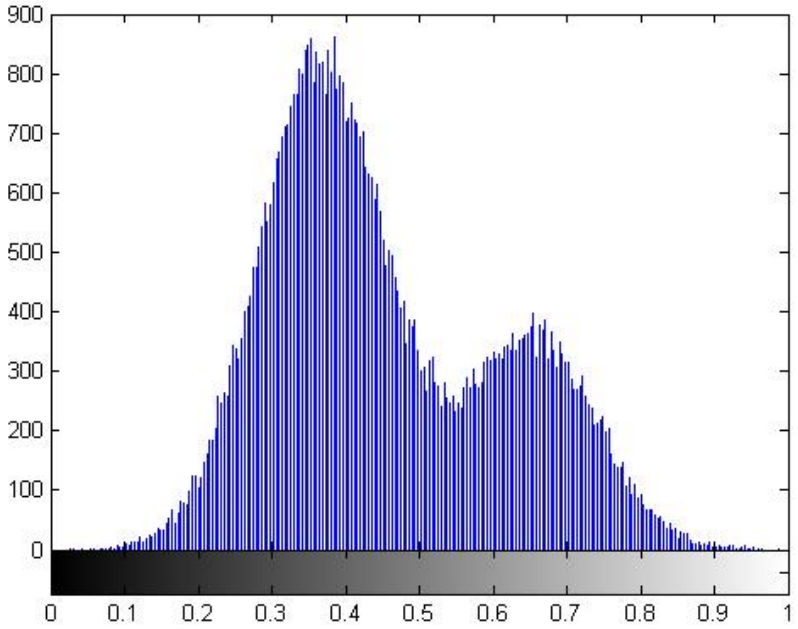
Example: region-based segmentation

- If the two regions have different graylevel distributions (histograms) then it is possible to split them by exploiting such an information



Example: region-based segmentation





Types of operations

- Histogram equalization → contrast enhancement
- Histogram stratching/shrinking → expansion/compression of the dynamic range
  - Loss of resolution (the same set of pixel values are represented by a subset of graylevel values)
- Histogram sliding → change of the mean level

H. stratching/shrinking

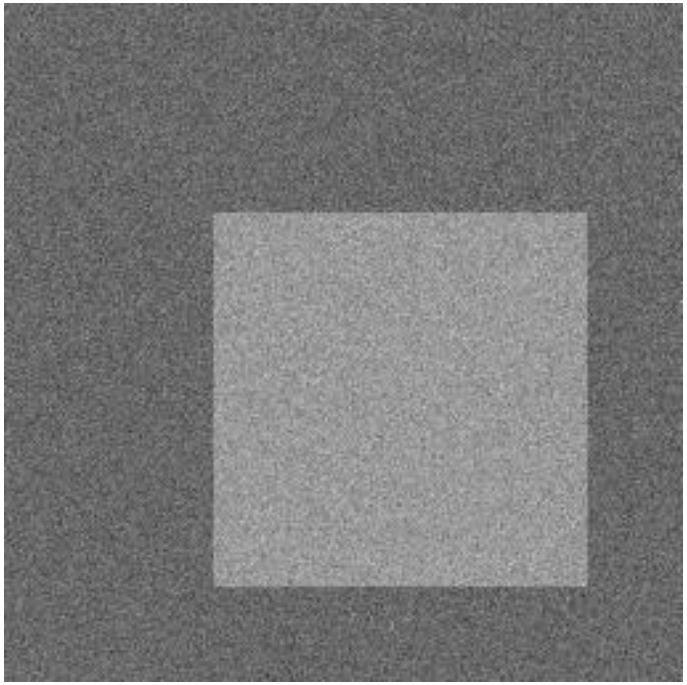
$$stretch[I] = \left[\frac{I - g^{0}_{\min}}{g^{0}_{\max} - g^{0}_{\min}}\right]\left[g^{1}_{\max} - g^{1}_{\min}\right] + g^{1}_{\min}$$

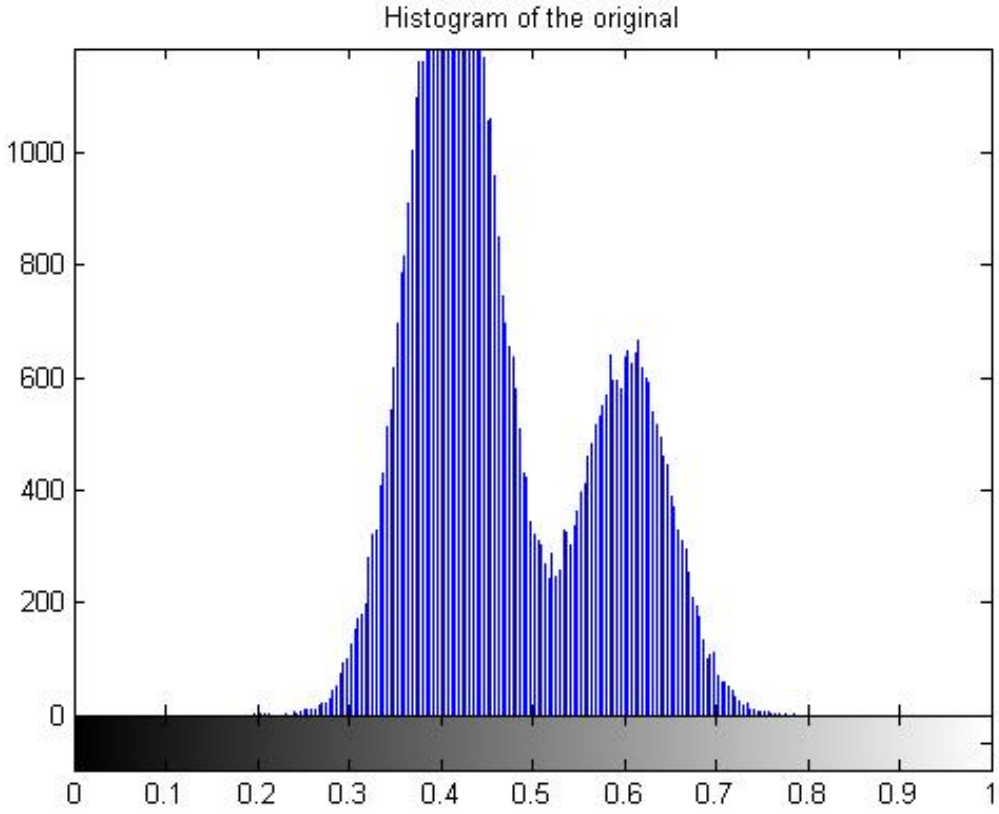
$g^{0}_{\max} = \max_{g} I[i, j] =$ maximum gray value in the original image

$g^{0}_{\min} \min_{g} I[i, j] =$ minimum gray value in the original image

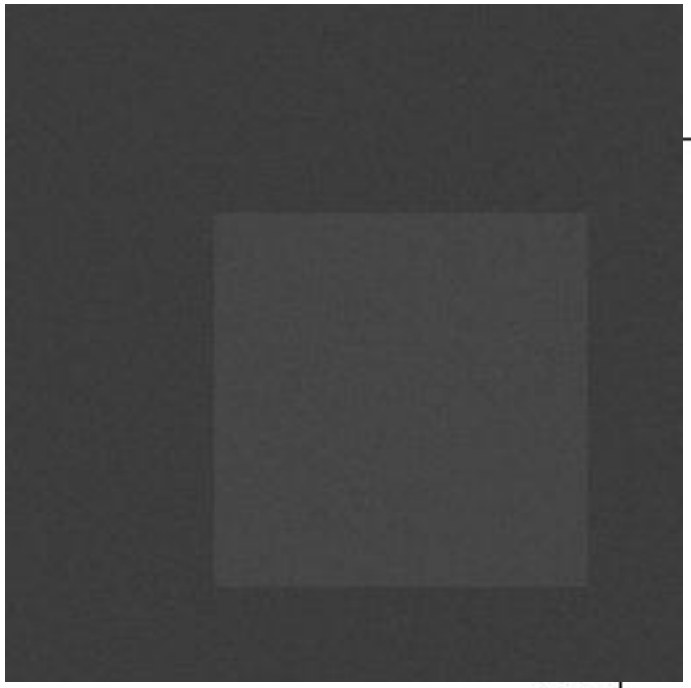
$g^{1}_{\max}, g^{1}_{\min} =$ maximum and minimum gray values in the processed image

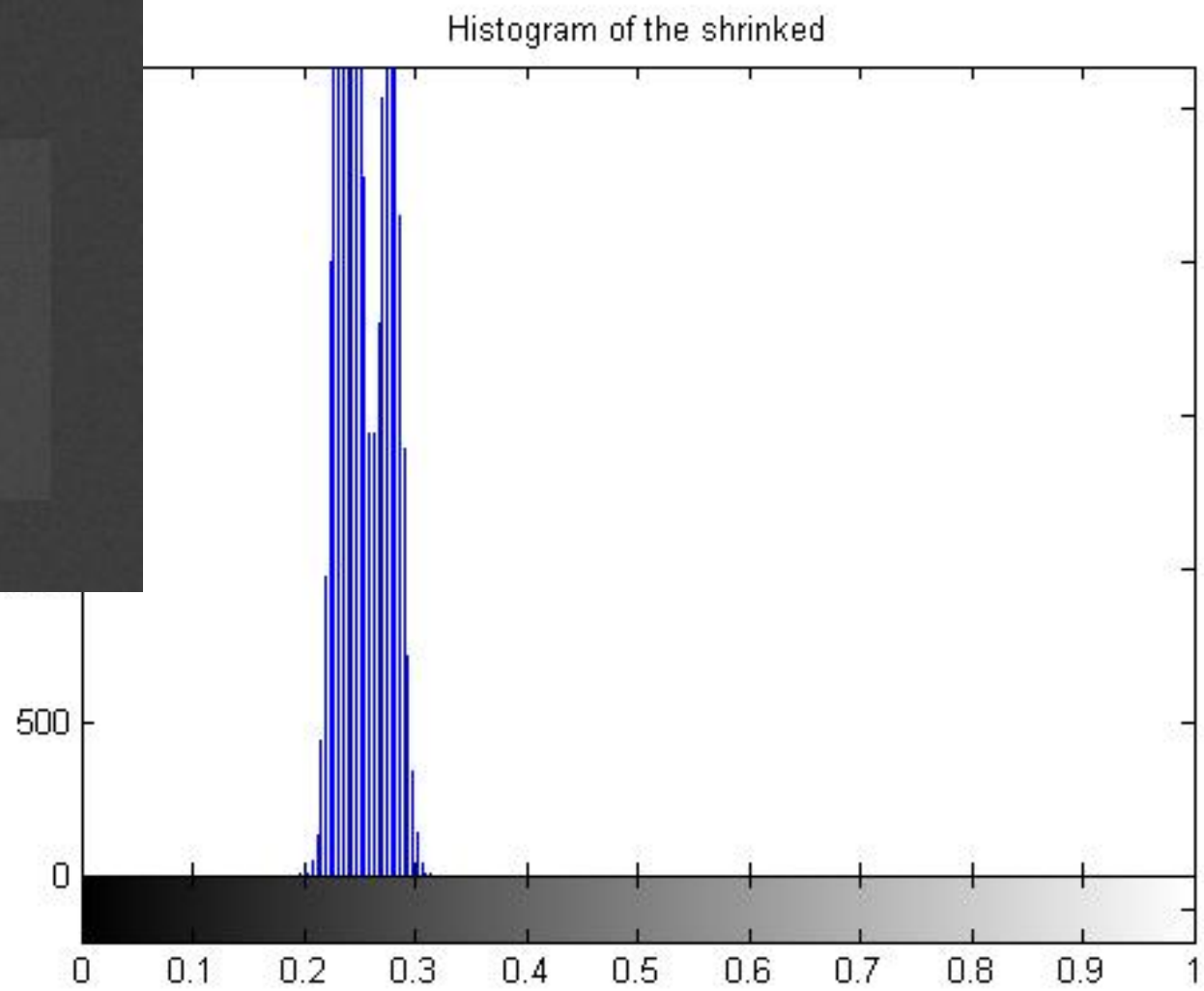
H. original



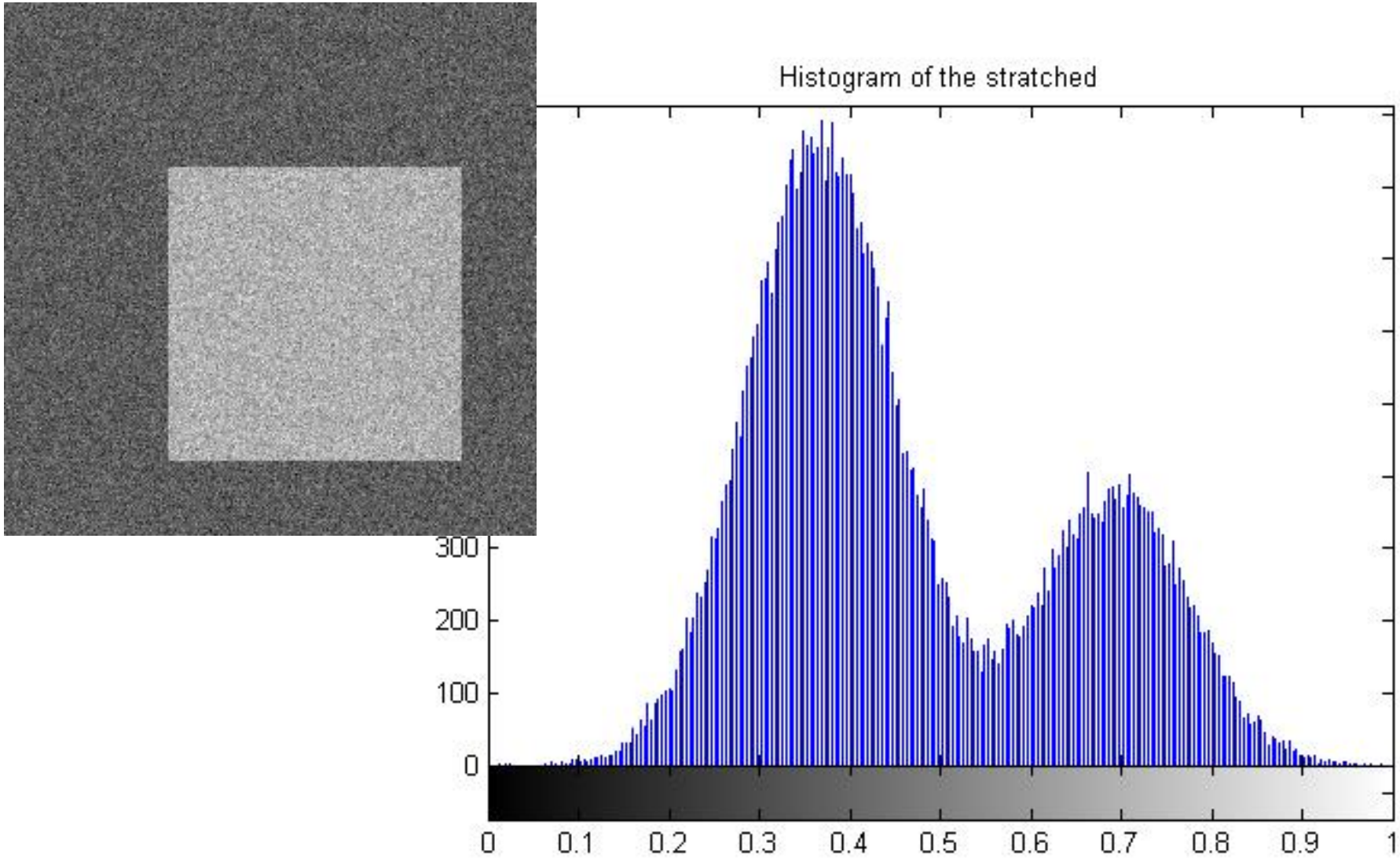

Histogram of the original
1000
800
600
400
200
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

H. shrinked

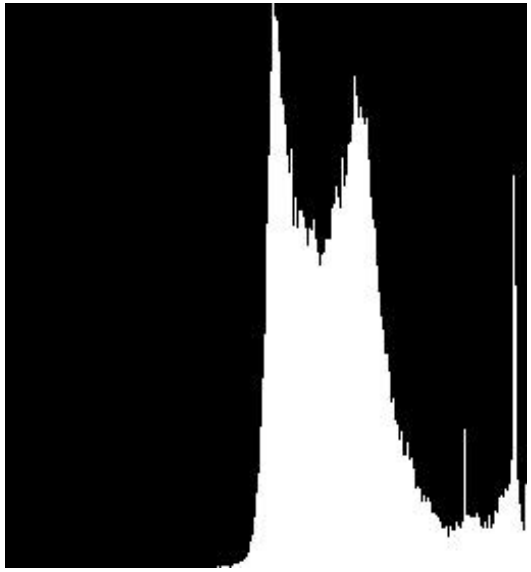


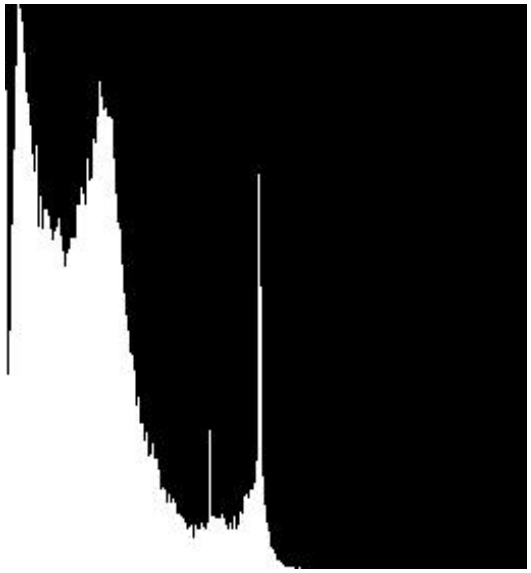


H. stratched

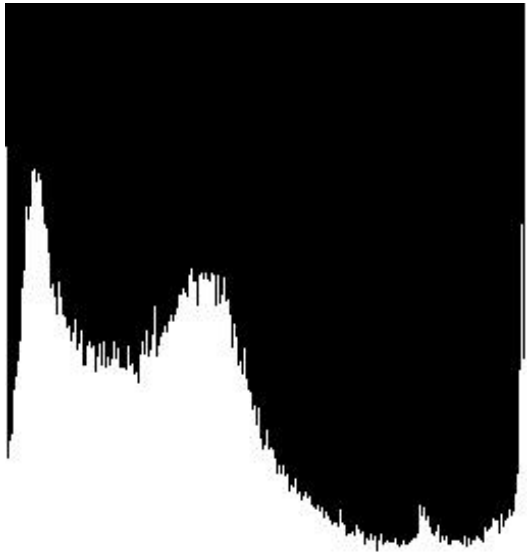

Histogram of the stratched
300
200
100
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

H. sliding



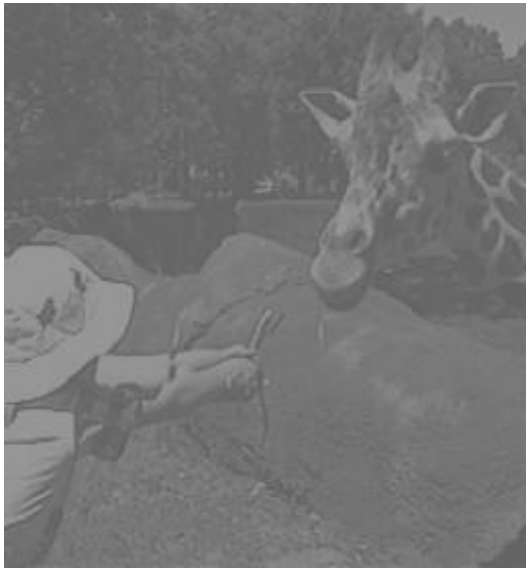


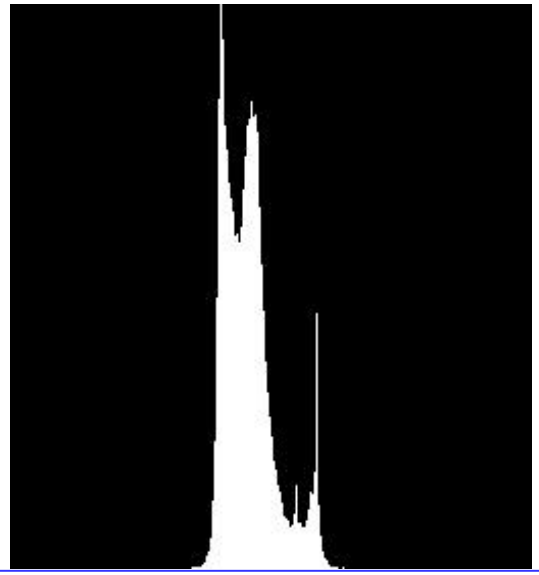
H. stratching/shrinking

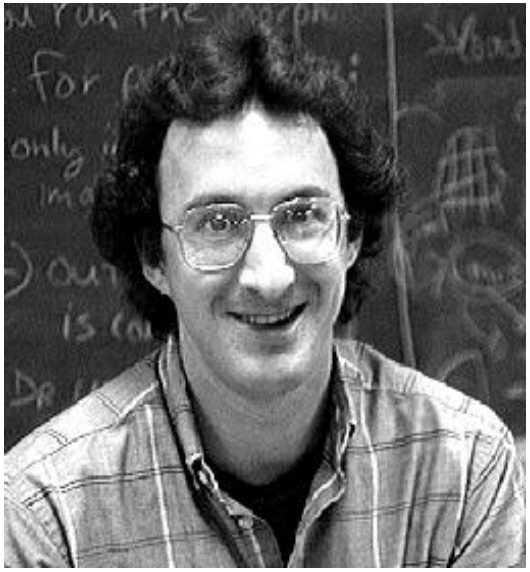


stratching

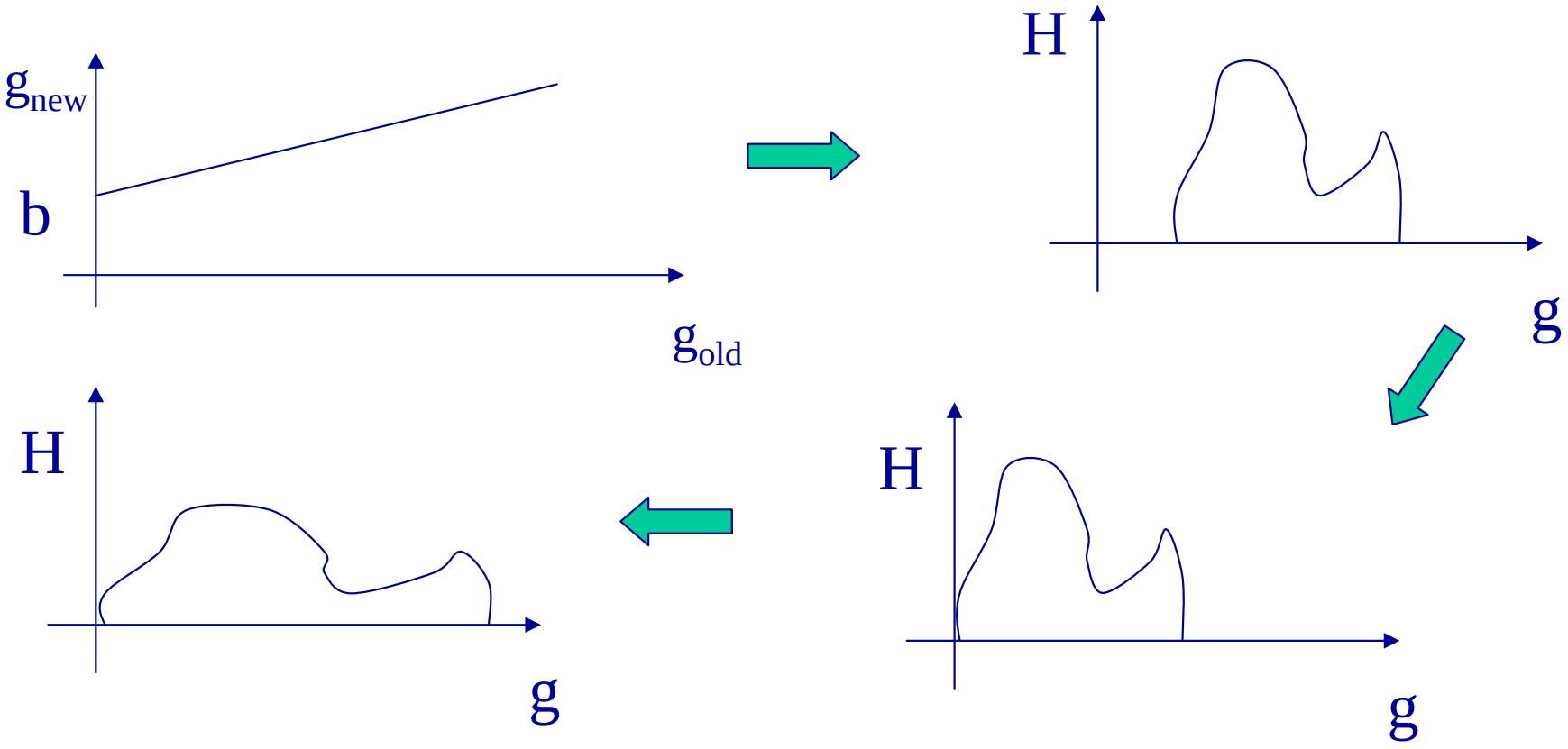
shrinking





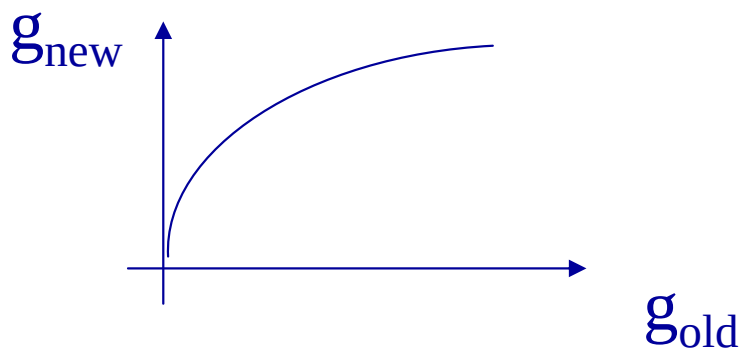


Linear graylevel transformations

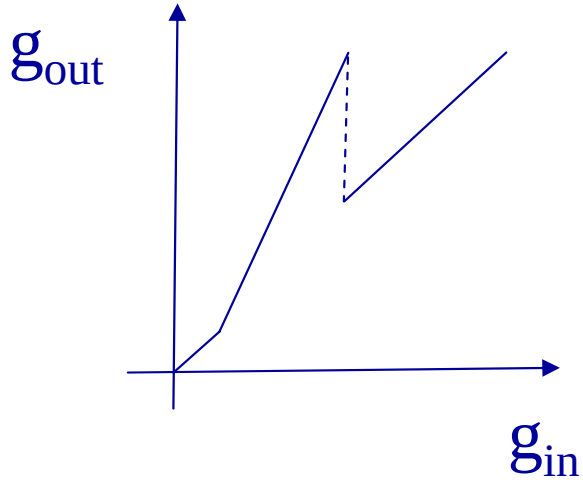


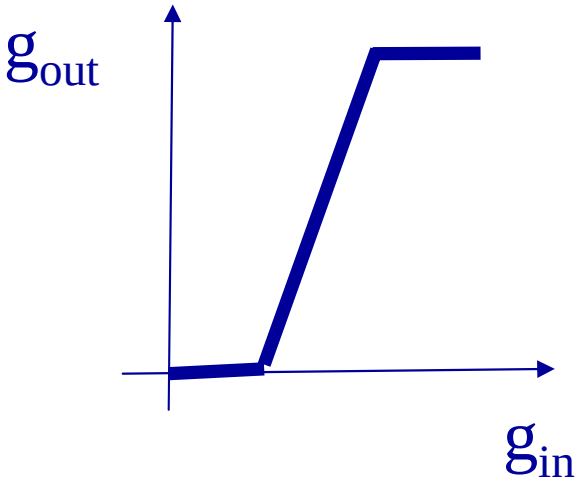
Non-linear tranformations

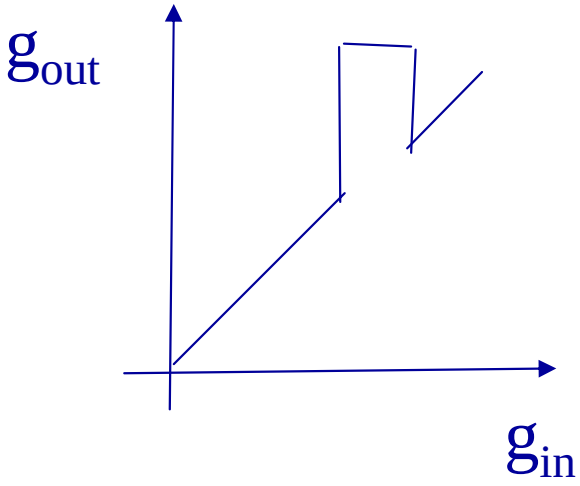
- Used to emphasize mid-range levels
- $g_{new} = g_{old} + g_{old}\, C\, (g_{old,max} - g_{old})$

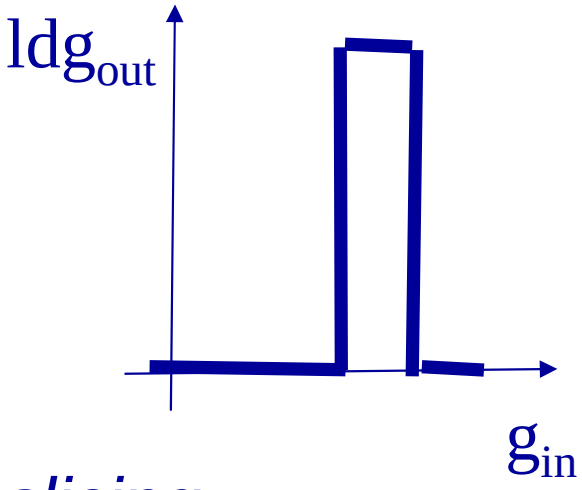


Common transformations



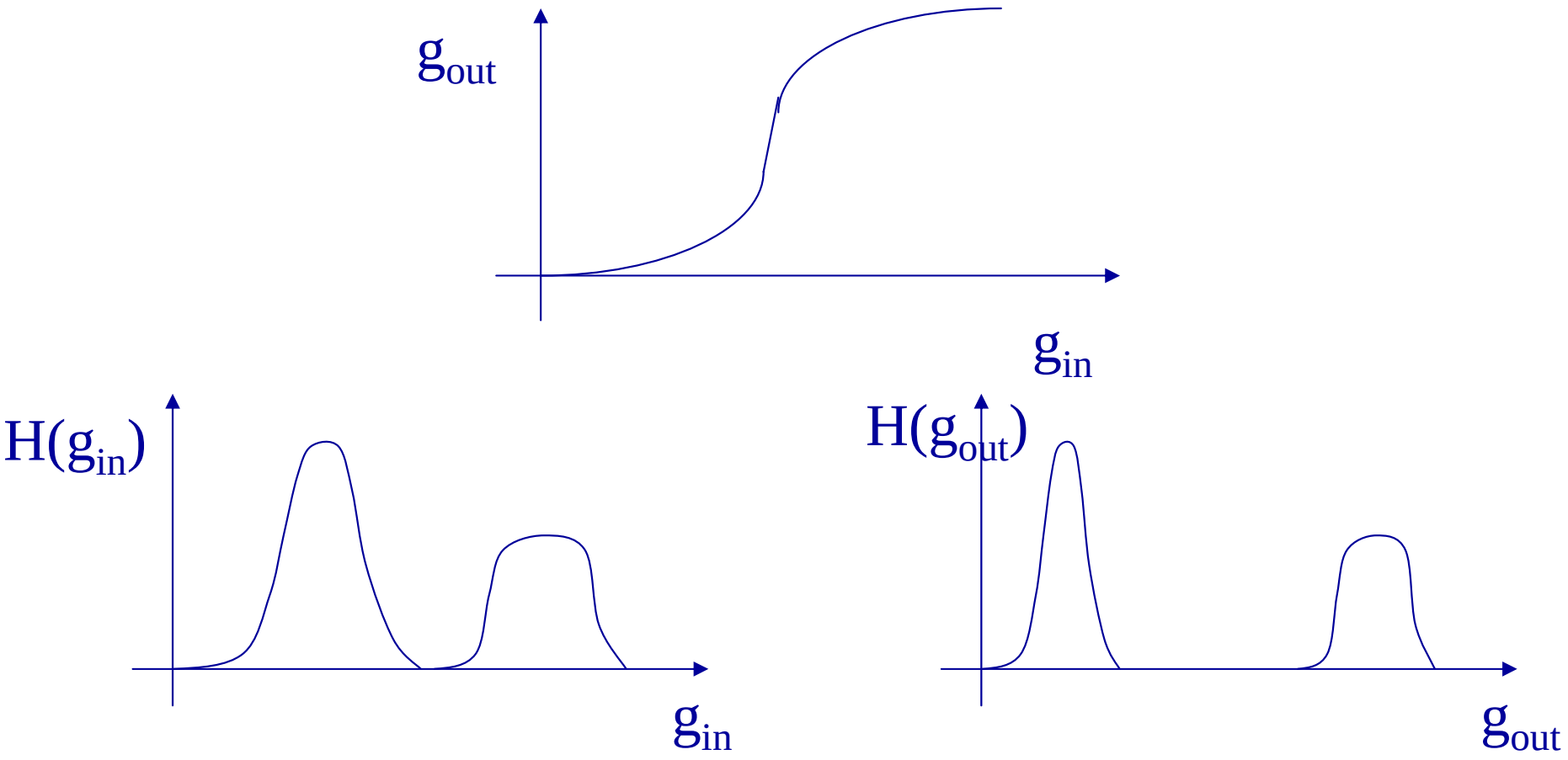






Intensity slicing

Sigmoid transformation (*soft thresholding*)



Histogram transformation

$$g_{out} = f(g_{in}) \Rightarrow g_{in} = f^{-1}(g_{out}), \quad f \quad \text{non-decreasing function}$$

$$H(g_{in}) \Rightarrow H(g_{out}), \quad \text{namely}$$

$$H(g_{out}) = \frac{H[f^{-1}(g_{out})]}{f'[f^{-1}(g_{out})]}, \quad f' = \frac{\partial f}{\partial g}$$

Histogram equalization

- Let x be a random variable and let n_i be the number of occurrences of gray level i. The probability of an occurrence of (a pixel of level) i in the image is

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \leq i < L$$

  - L being the total number of gray levels in the image, n being the total number of pixels in the image, and p_x being in fact the image's histogram, normalized to [0,1].

- Let us also define the *cumulative distribution function* corresponding to p_x as

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

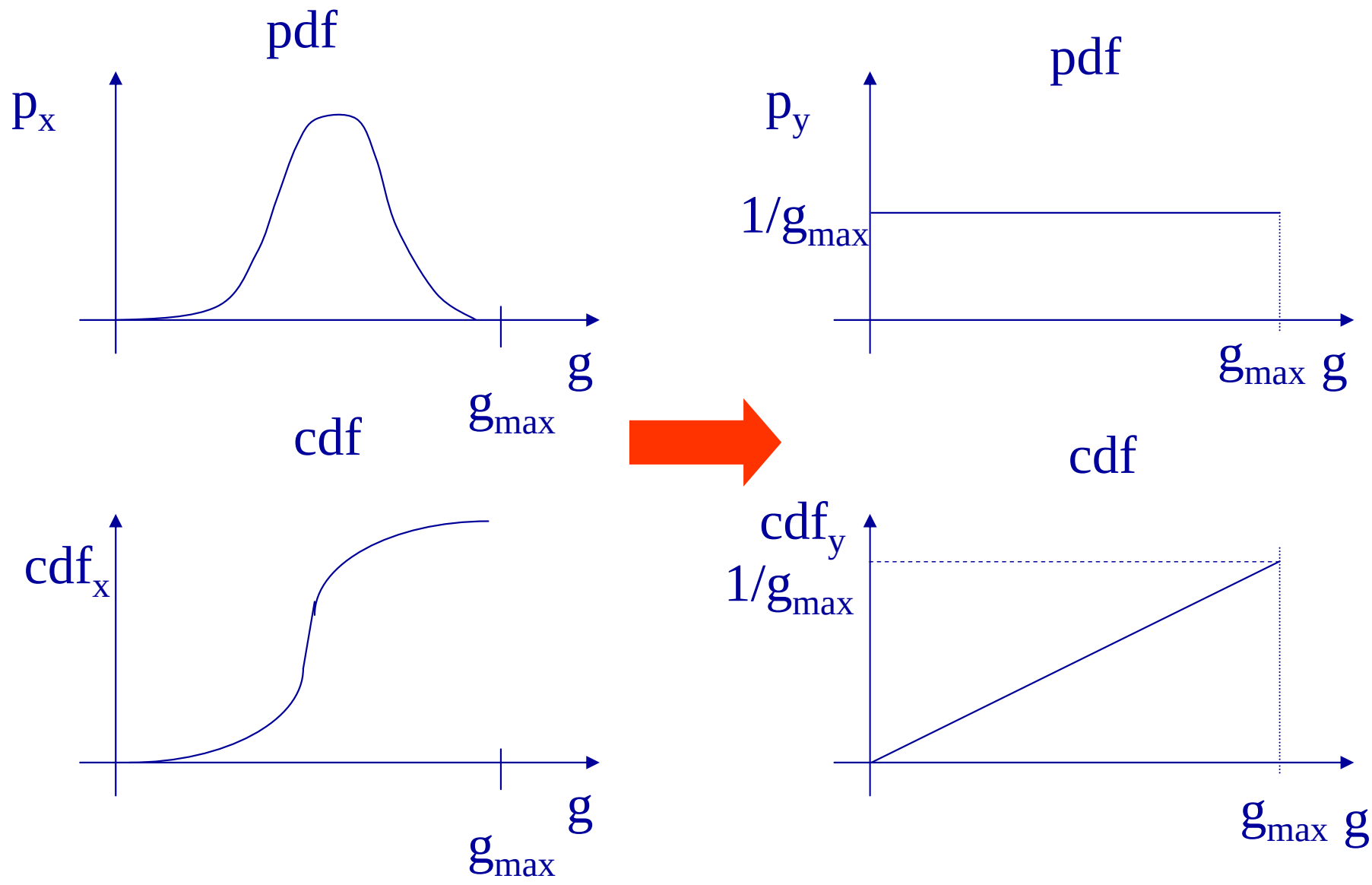
- We would like to create a transformation of the form $y = T(x)$ to produce a new image $\{y\}$, such that its CDF will be linearized across the value range, i.e.

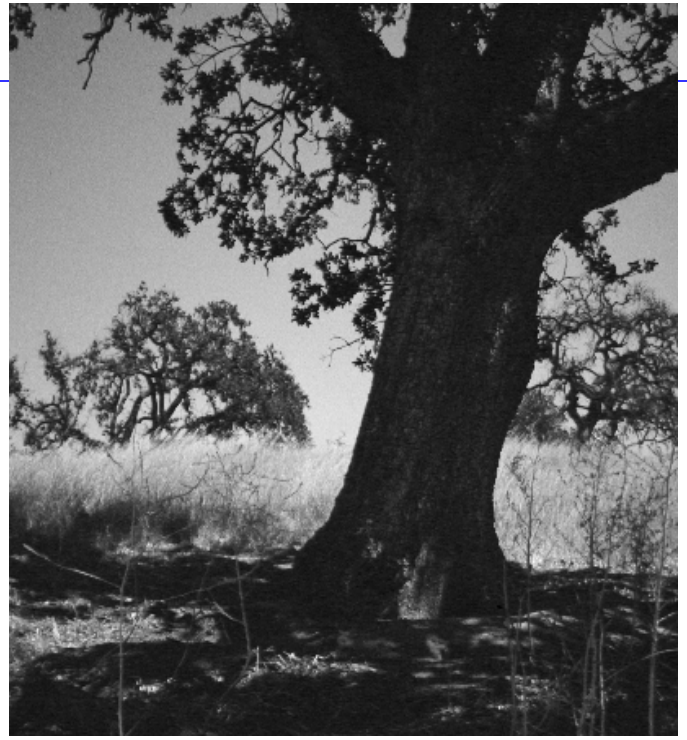
$$cdf_y(i) = K \cdot i$$

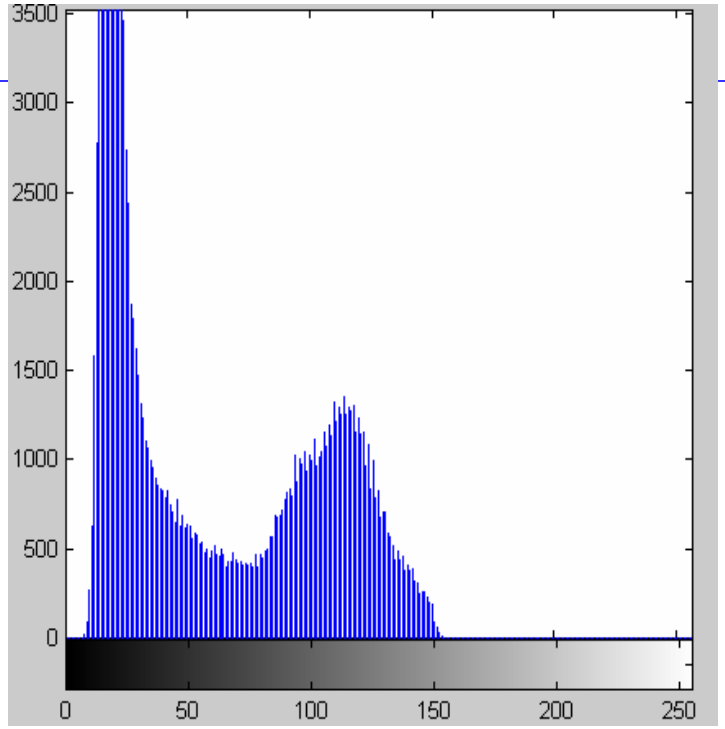
- The properties of the CDF allow us to perform such a transform it is defined as

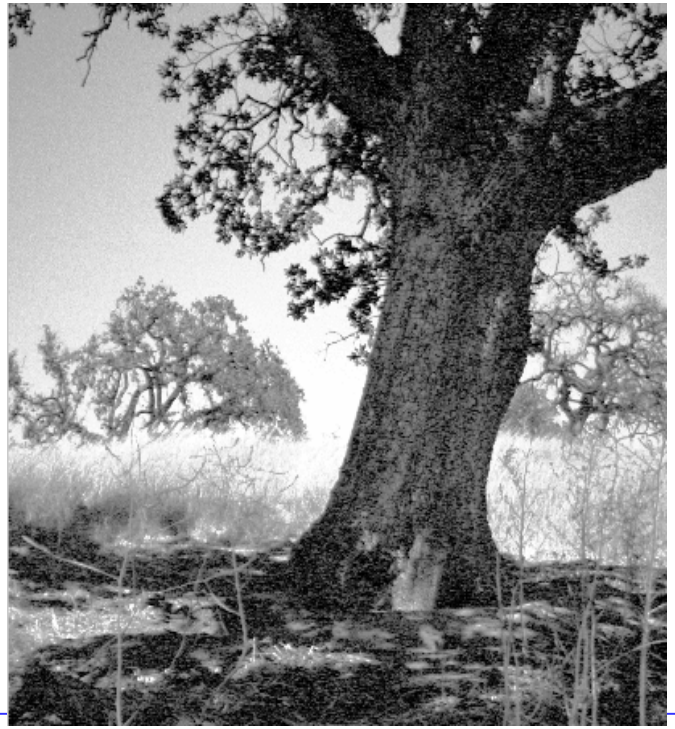
$$y = T(x) = cdf_x(x)$$

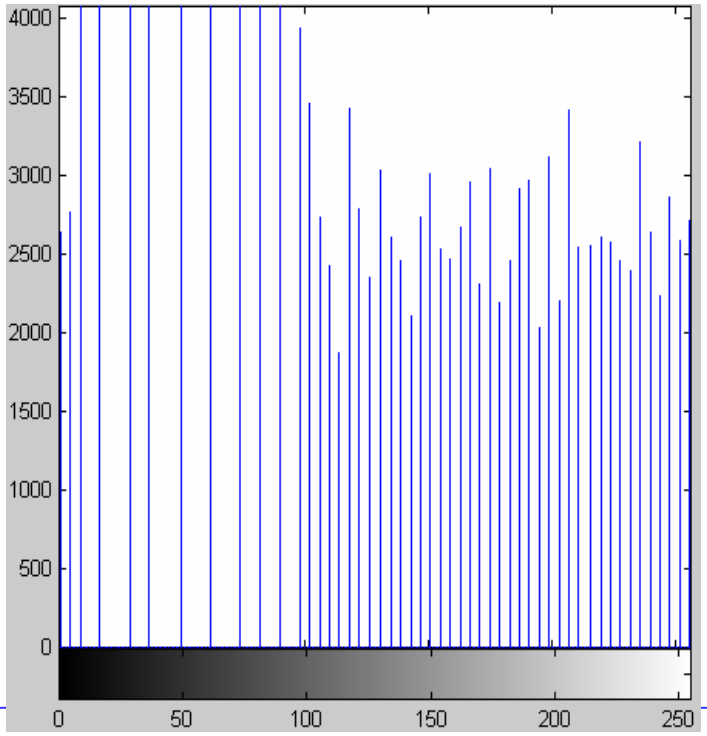
Pictorially






3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250




4000
3500
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250

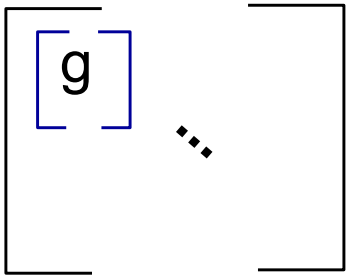
Neighborhood operations

- Correlation ↔ pattern recognition
- Convolution ↔ Linear filtering
  - Edge detection
  - Denoising

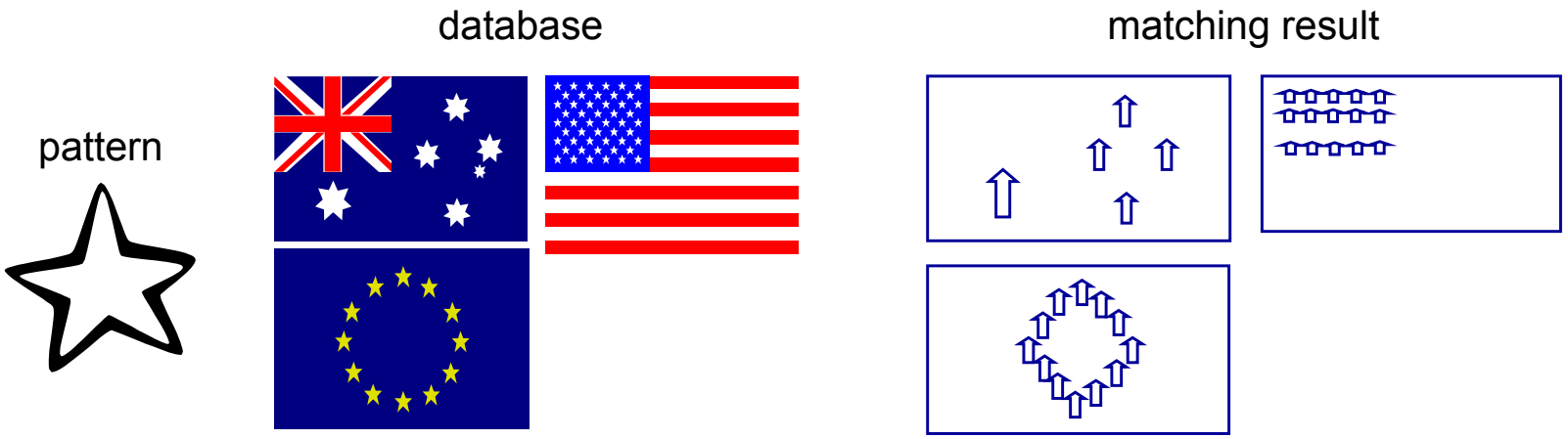
Correlation

- Correlation
  - Measures the similarity between two signals
  - Difference from convolution: no ‘minus’ signs in the formula
    - the signals need only to be translated

$$C(m,n) = \sum_k \sum_r f[m,n] h_{template}[m+k, n+r]$$



Application



Convolution

$$g[m,n] = f[m,n] * h_{filter}[m,n] = \sum_{k,r} f[m,n] h_{filter}[k-m, r-n]$$

$$G(j\omega_x, j\omega_y) = F(j\omega_x, j\omega_y) H_{filter}(j\omega_x, j\omega_y)$$

$f[m,n]$: original (input) image

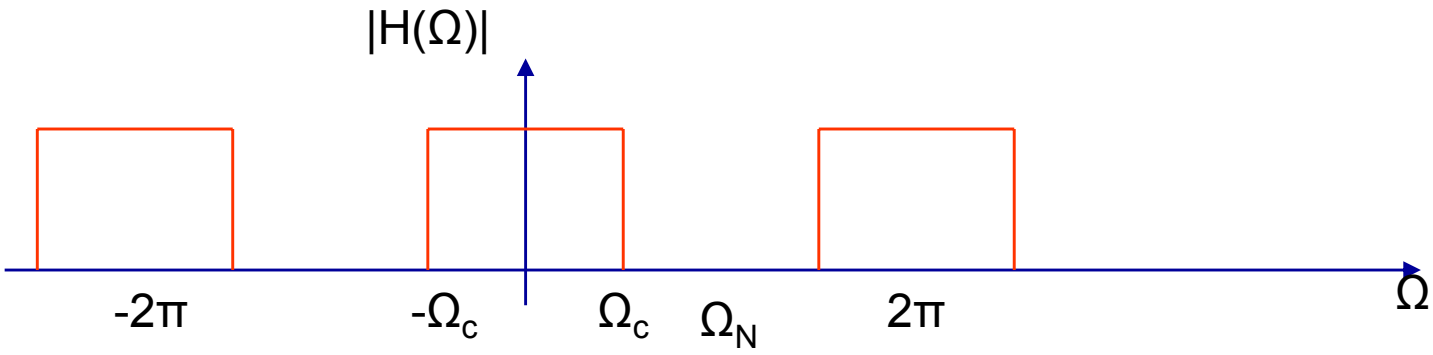
$g[m,n]$: filtered (output) image

$h_{filter}[m,n]$: filter impulse response

Convolution and digital filtering

- Digital filtering consists of a convolution between the image and the impulse response of the filter, which is also referred to as convolution kernel.
- Warning: both the image and the filter are matrices (2D). If the filter is separable, then the 2D convolution can be implemented as a cascade of two 1D convolutions
- Filter types
  - FIR (Finite Impulse Response)
  - IIR (Infinite IR)

Ideal low-pass *digital* filter



Digital LP filter (discrete time)

The boundary between the pass-band and the stop-band is sharp

The spectrum is periodic

The repetitions are located at integer multiples of 2π

The low-pass filtered signal is *still* a digital signal, but with a different frequency content

The impulse response h[n] in the signal domain is discrete time and corresponds to the `sinc[]` function

Reconstruction LP filter (continuous time)

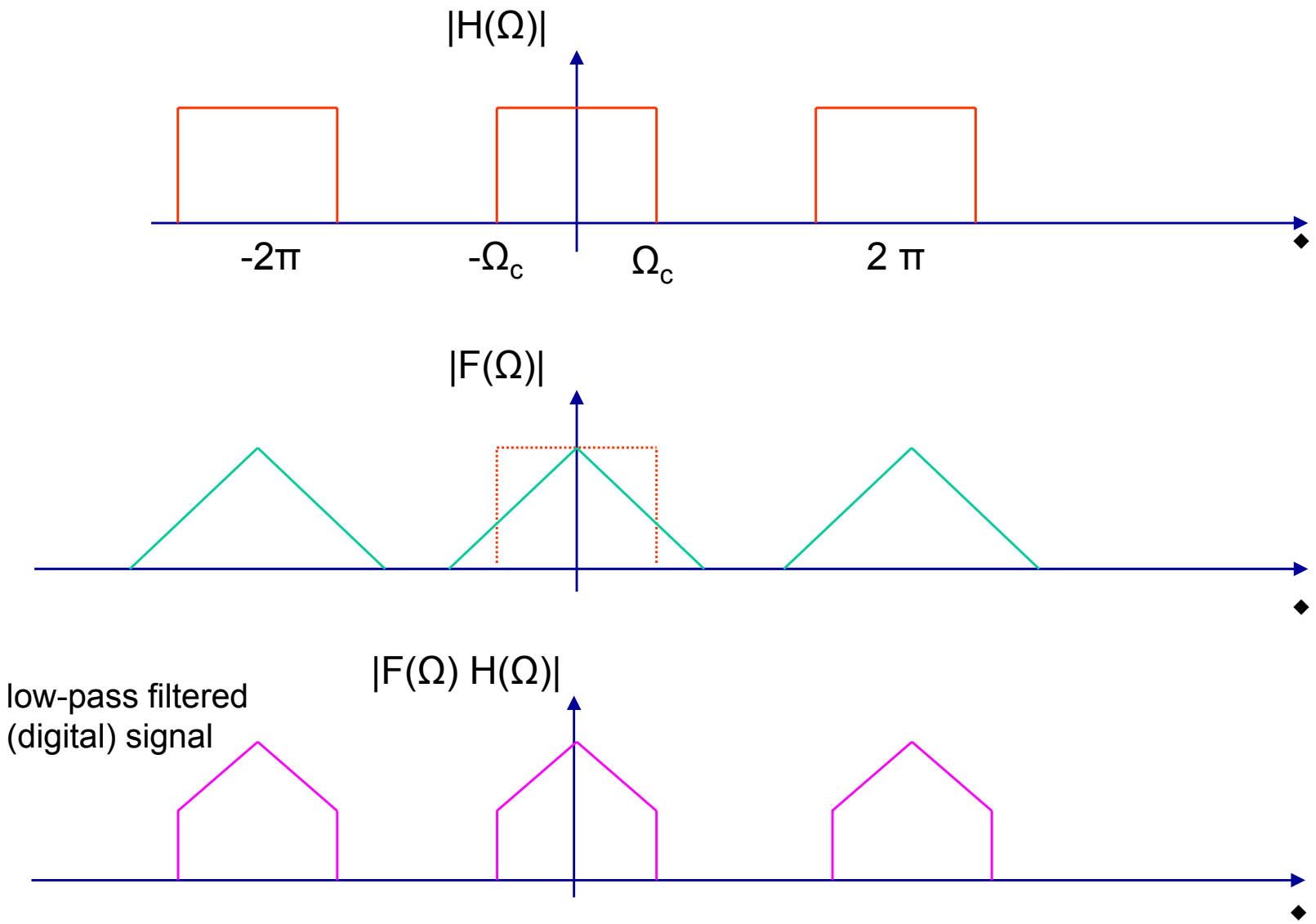
The boundary between the pass-band and the stop-band is sharp

The spectrum consists of ***one repetition only***

The low-pass filtered signal is a continuous time signal, that might have a different frequency content

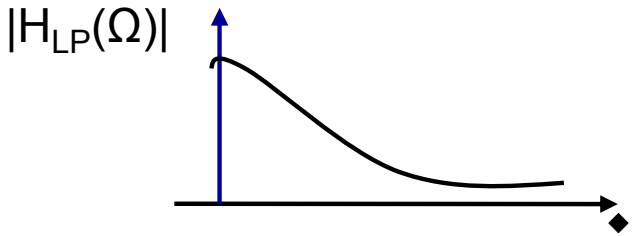
The impulse response h(t) in the signal domain is continuous time and corresponds to the `sinc()` function

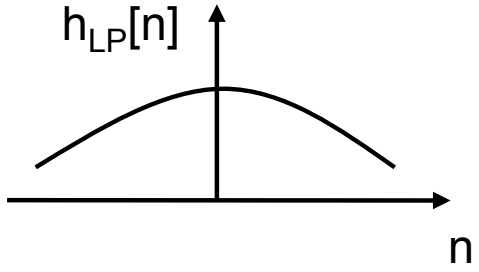
Digital LP filtering



LP and HP filtering

Low-pass

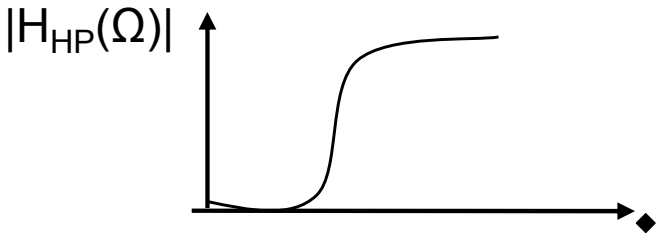


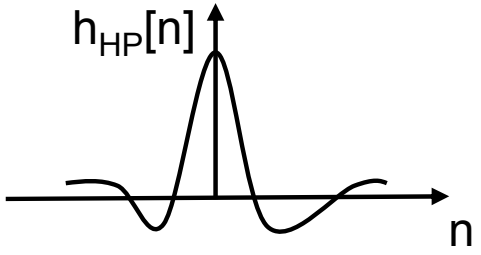


integration

averaging

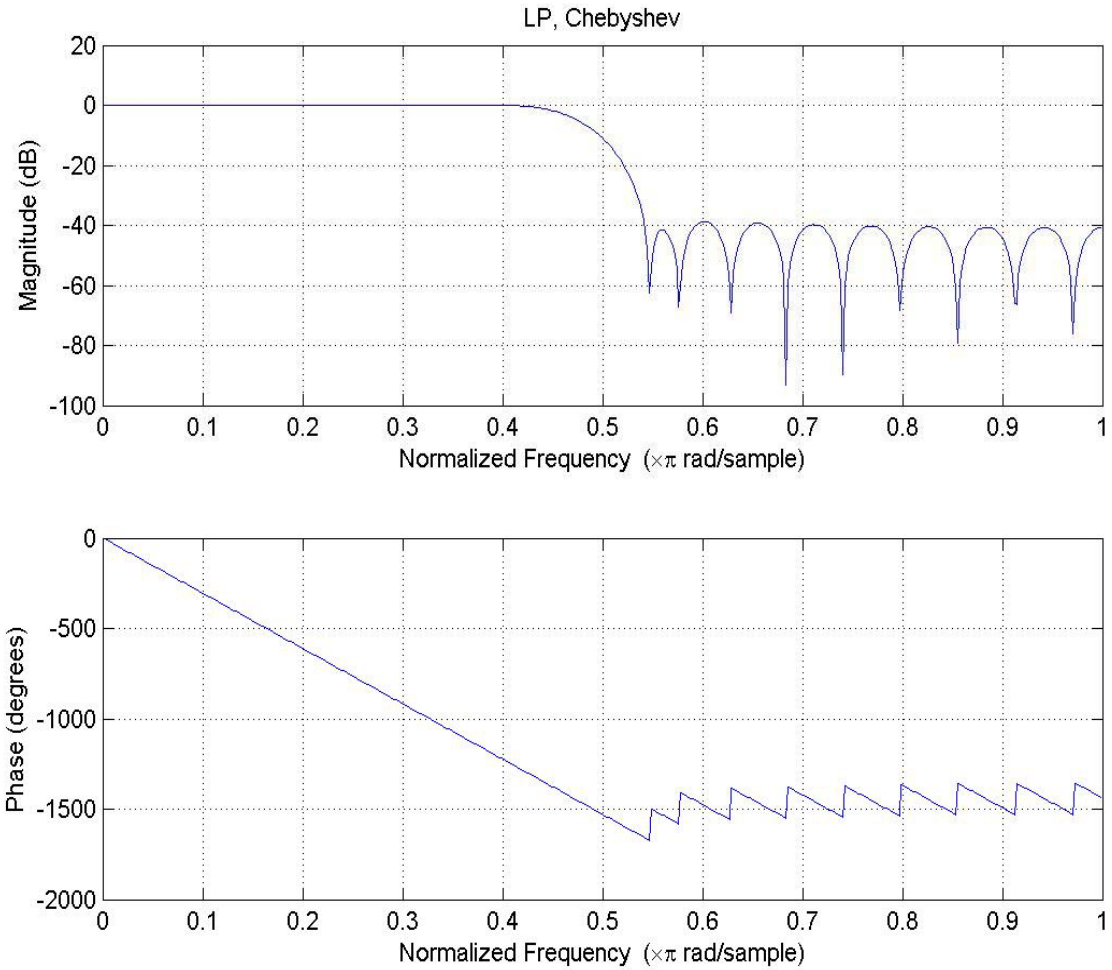
High-pass



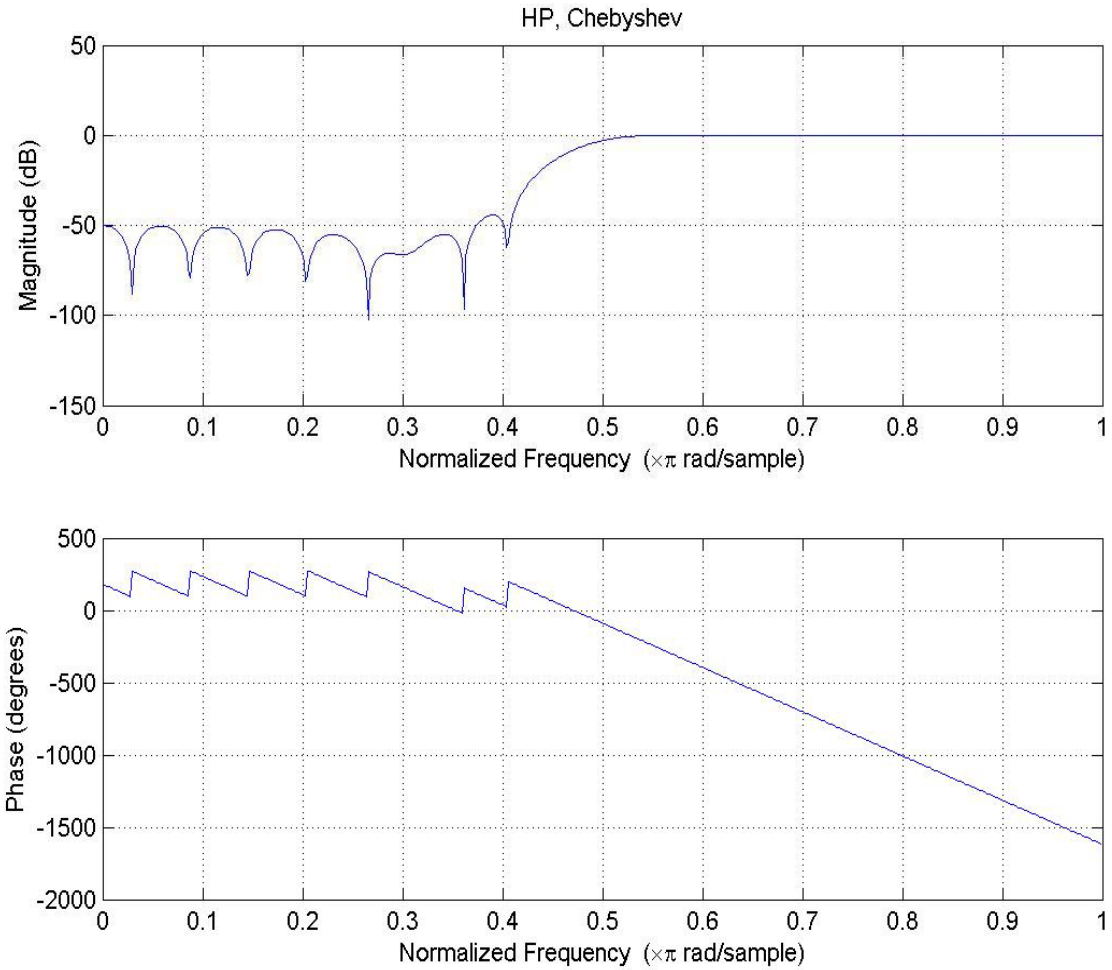


differentiation

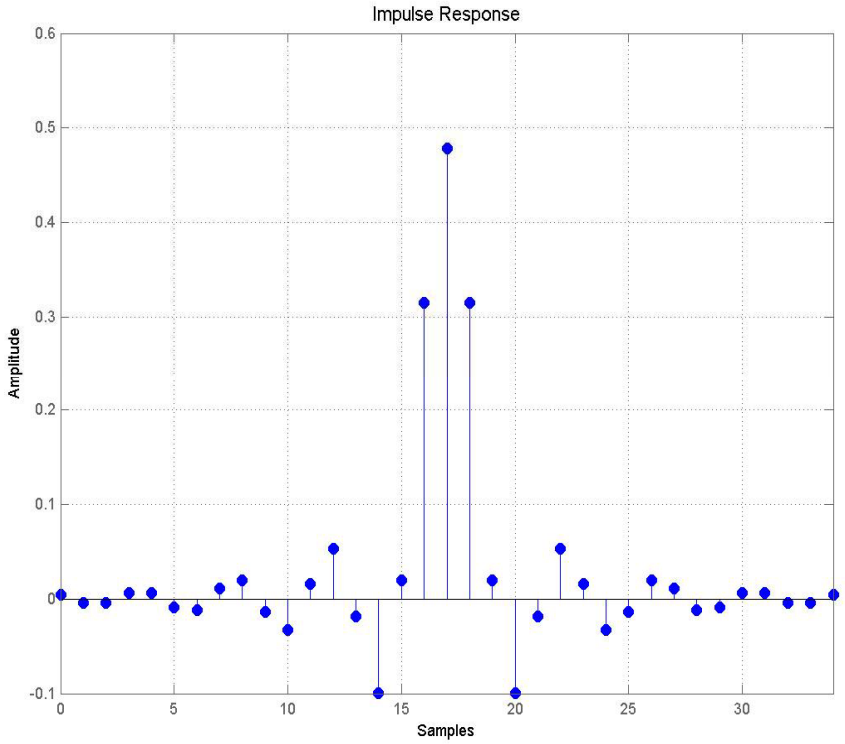
Example: Chebyshev LP

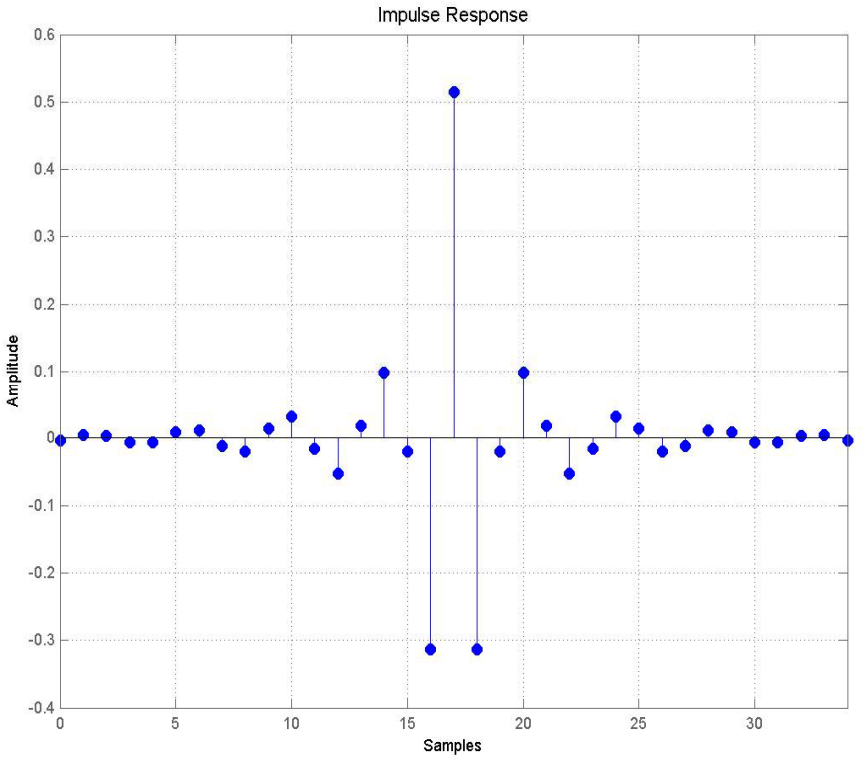


Example: Chebyshev HP

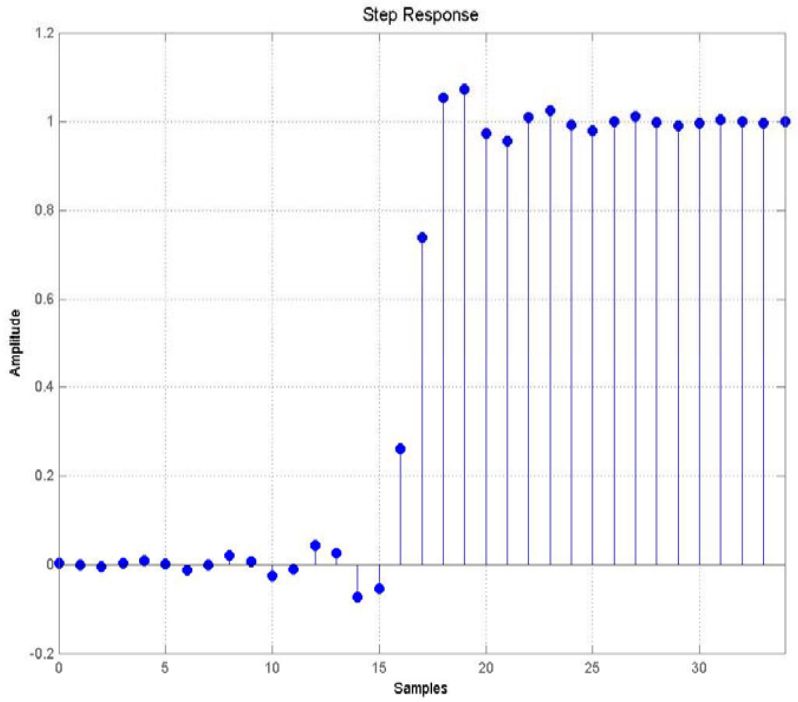


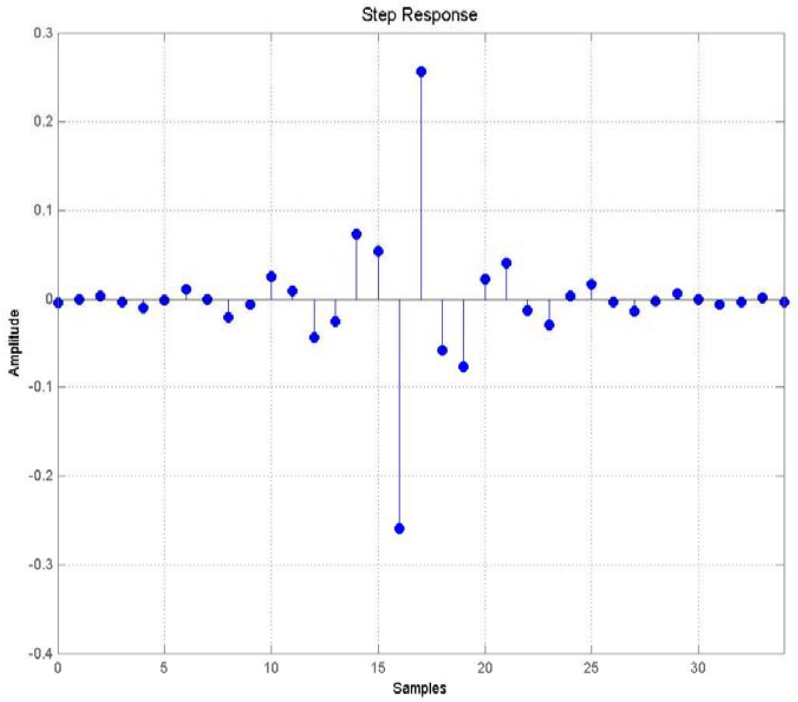
Example: Chebyshev Impulse Response



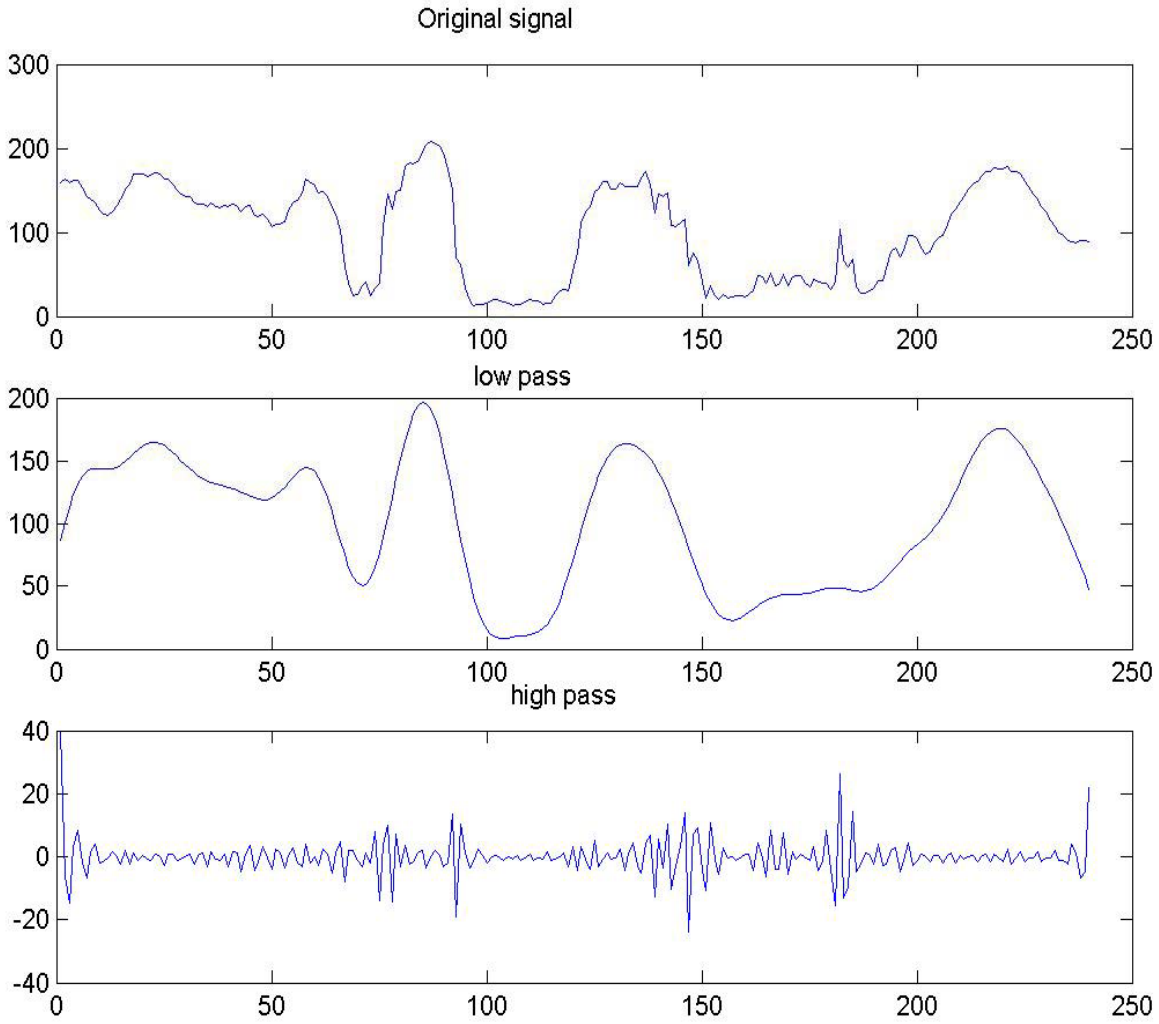


Example: Chebyshev Step Response





Example: filtered signal



The transfer function (or, equivalently, the impulse response) of the filter determines the characteristics of the resulting signal

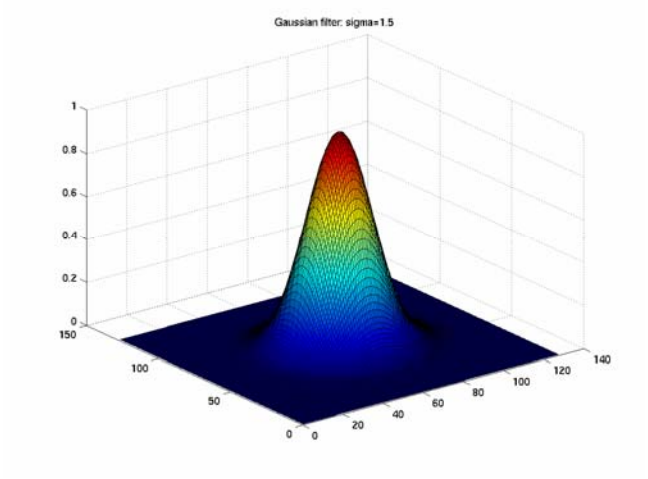
Switching to images qui

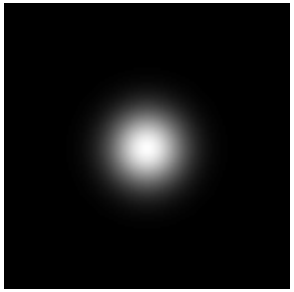
- Images are 2D digital signals (matrices) → filters are matrices
- Low-pass ↔ *averaging* (interpolation) ↔ smoothing
- High-pass ↔ differentiation ↔ emphasize image *details* like lines, and, more in general, sharp variations of the luminance
- Band-pass: same as high pass but selecting a predefined range of spatial frequencies

Splitting low-pass and high-pass image features is the ground of multi-resolution. It is advantageous for many tasks like contour extraction (edge detection), image compression, feature extraction for pattern recognition, image denoising etc.

2D filters

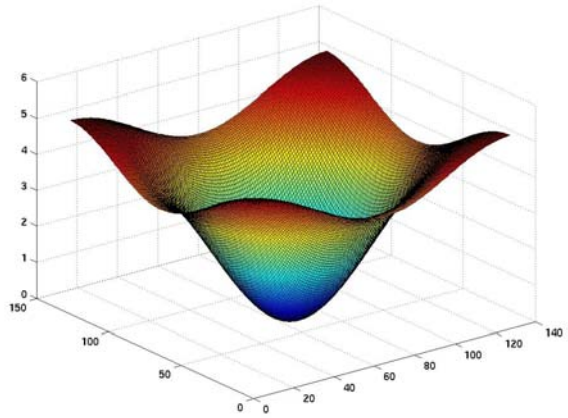
Low-pass

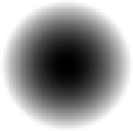




$$h_{lowpass} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

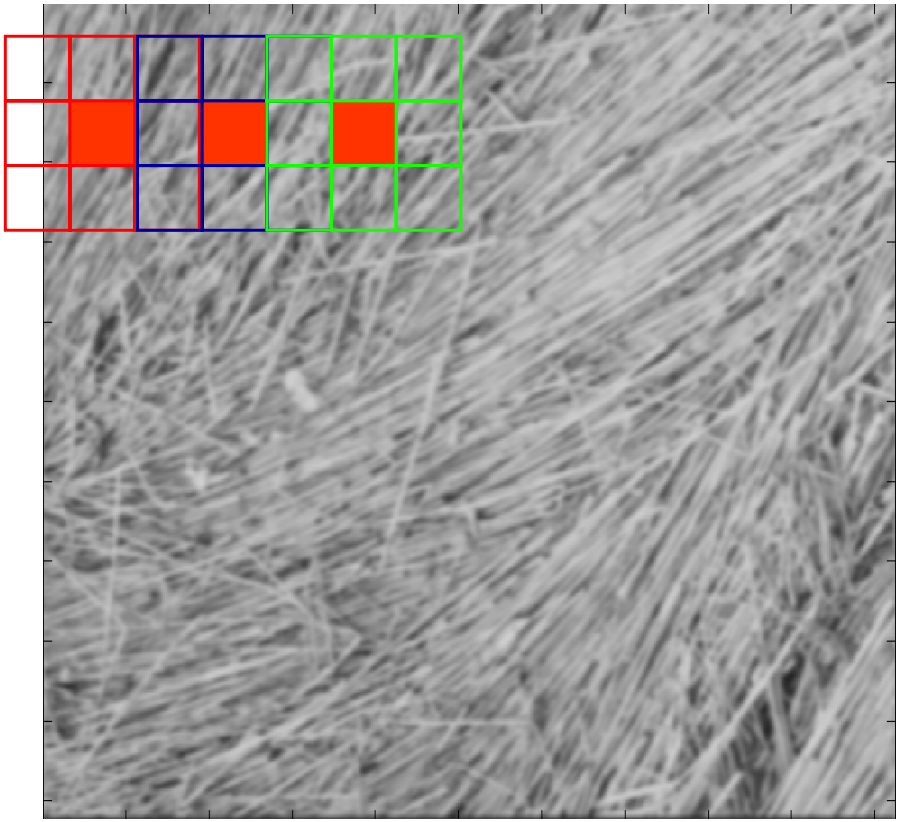
High-pass





$$h_{highpass} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

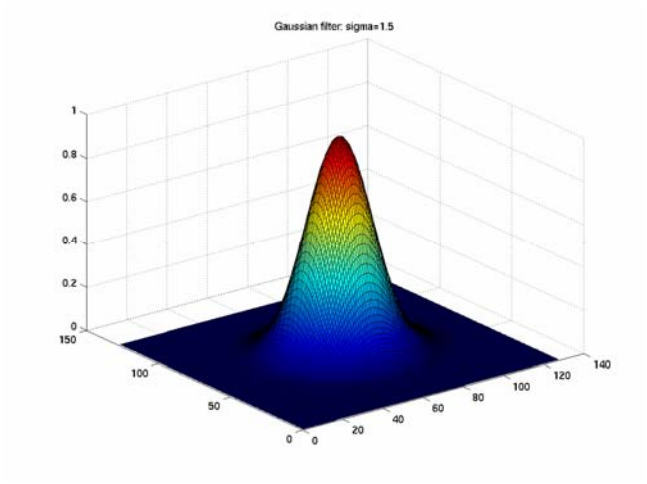
Filtering in image domain

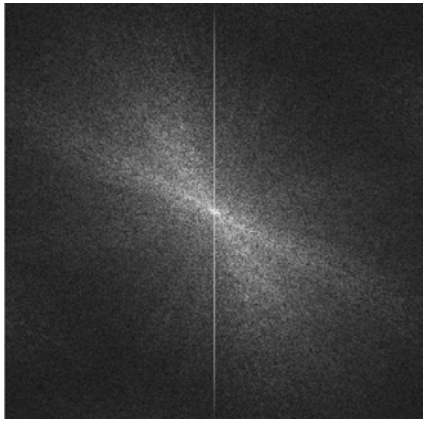


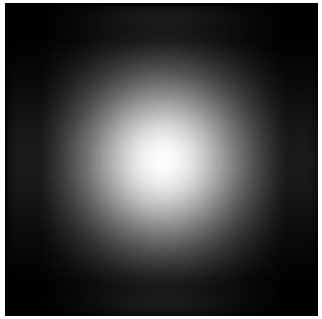
Filtering in image domain is performed by convolving the image with the filter kernel. This operation can be though of as a pixel-by-pixel product of the image with a moving kernel, followed by the sum of the pixel-wise output.

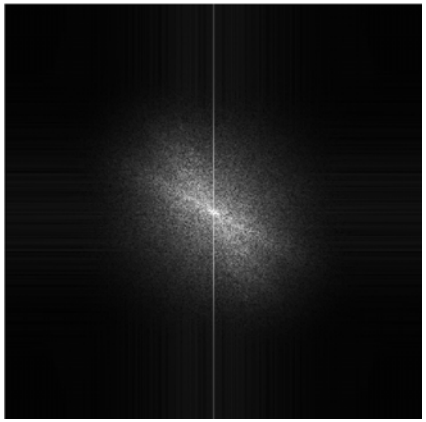
Low-pass filtering: example

$f[m,n]$ $h_{lowpass}[m,n]$ $g[m,n]$



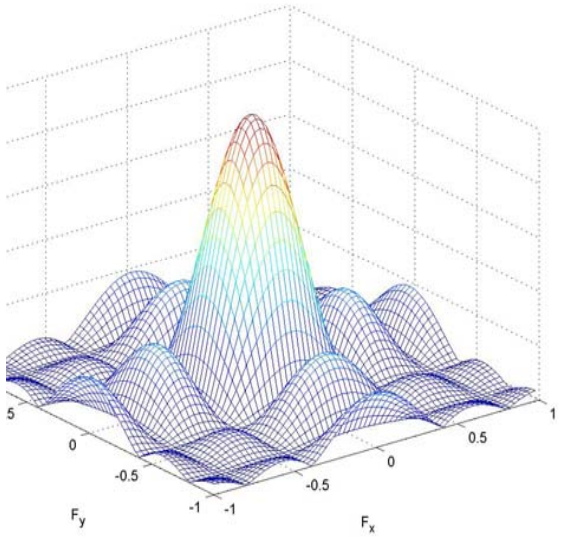






Averaging

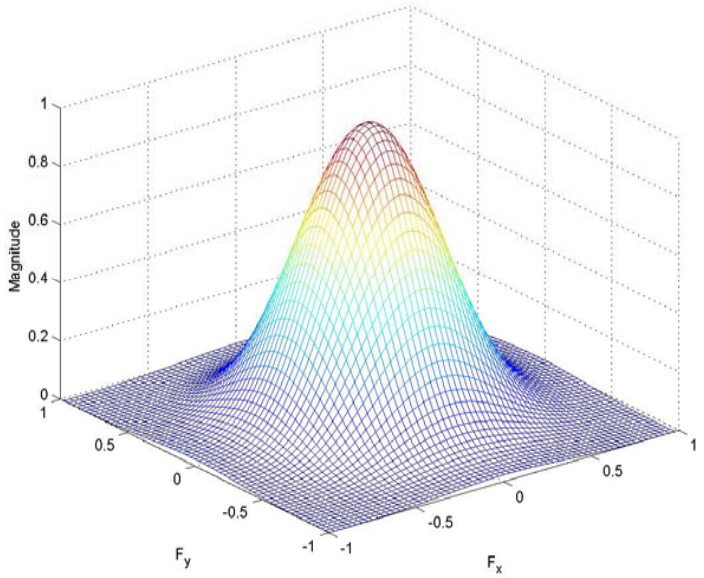
$$h_{lp}=1/25 \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$



Gaussian

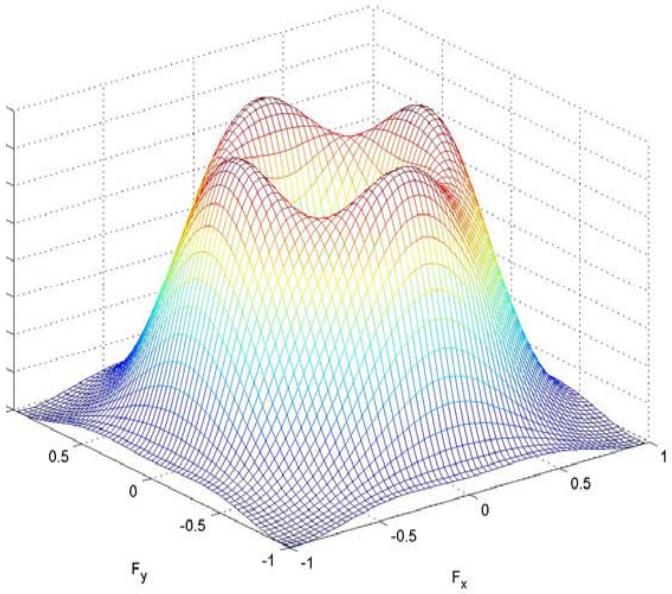
h_{lp}=

0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030



Log

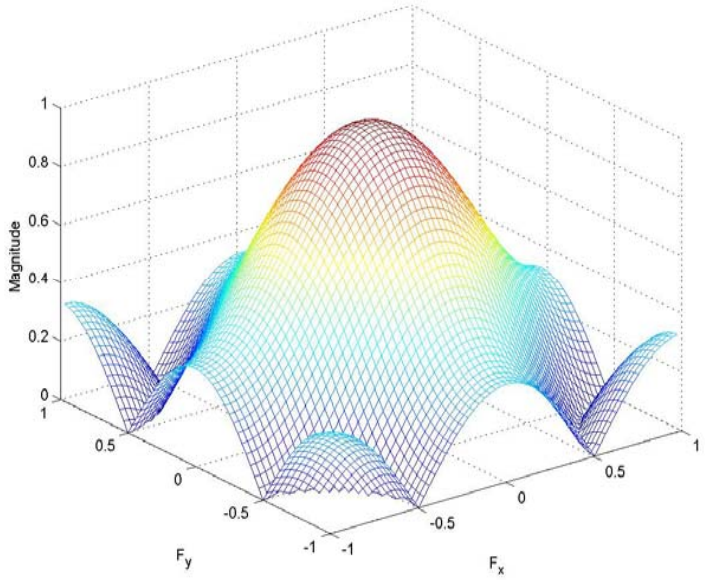
).0239	0.0460	0.0499	0.0460	0.0239
0.0460	0.0061	-0.0923	0.0061	0.0460
0.0499	-0.0923	-0.3182	-0.0923	0.0499
0.0460	0.0061	-0.0923	0.0061	0.0460
0.0239	0.0460	0.0499	0.0460	0.0239



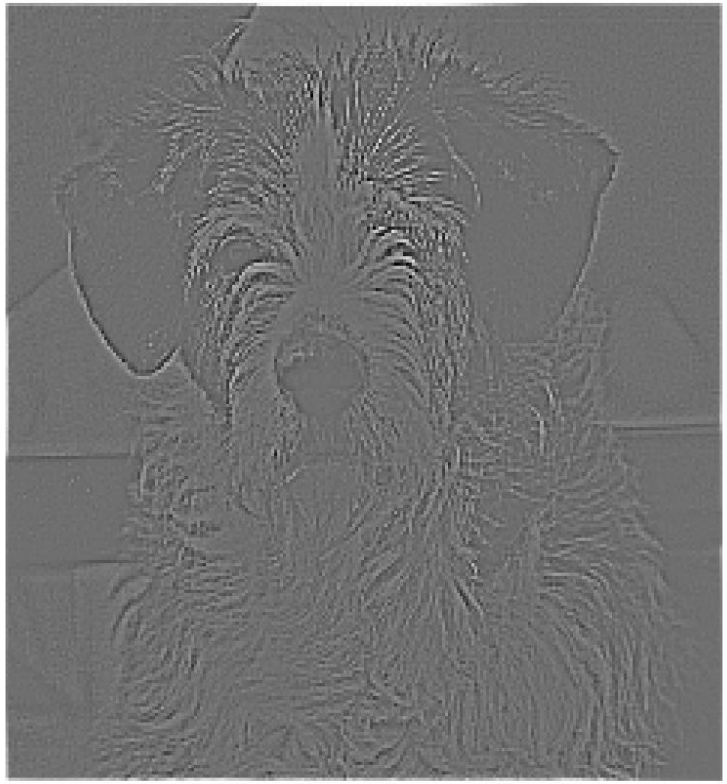
Asymmetric LP

$h_{lp}=1/6$

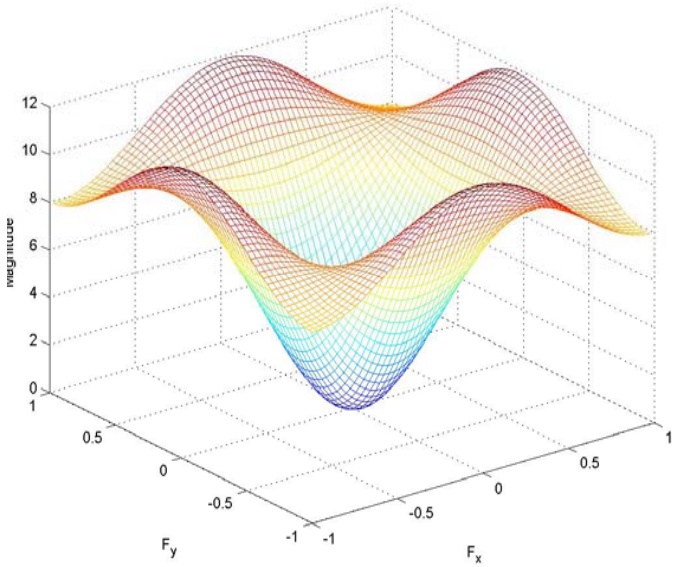
0	1	0
1	2	1
0	1	0



Asymmetric HP

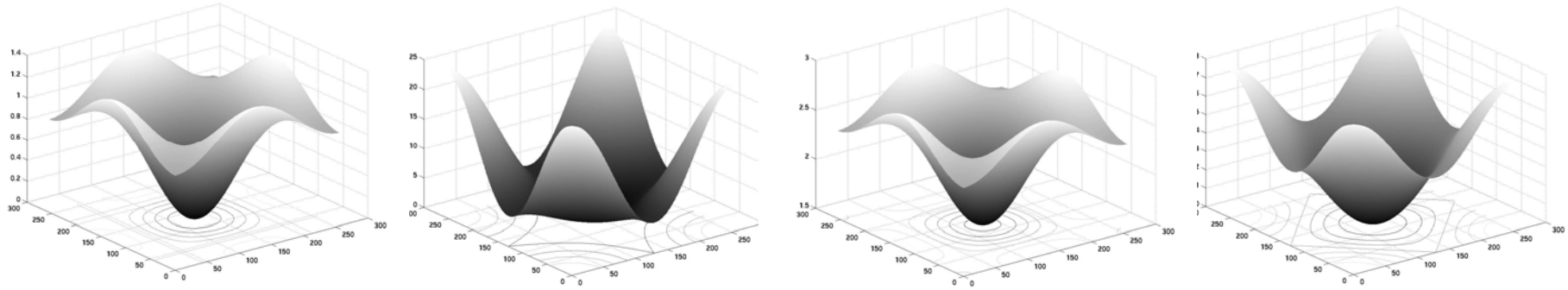


$$h_{lp} = \begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$



Sharpening

- Goal: "*improve*" image quality
- Solutions
  - increase *relative* importance of details, by increasing the relative weight of high frequencies components
    - Increase a subset of high frequencies (non symmetric HP)
    - High-boost filter
    - Laplacian gradient
  - The original image is assumed to be available

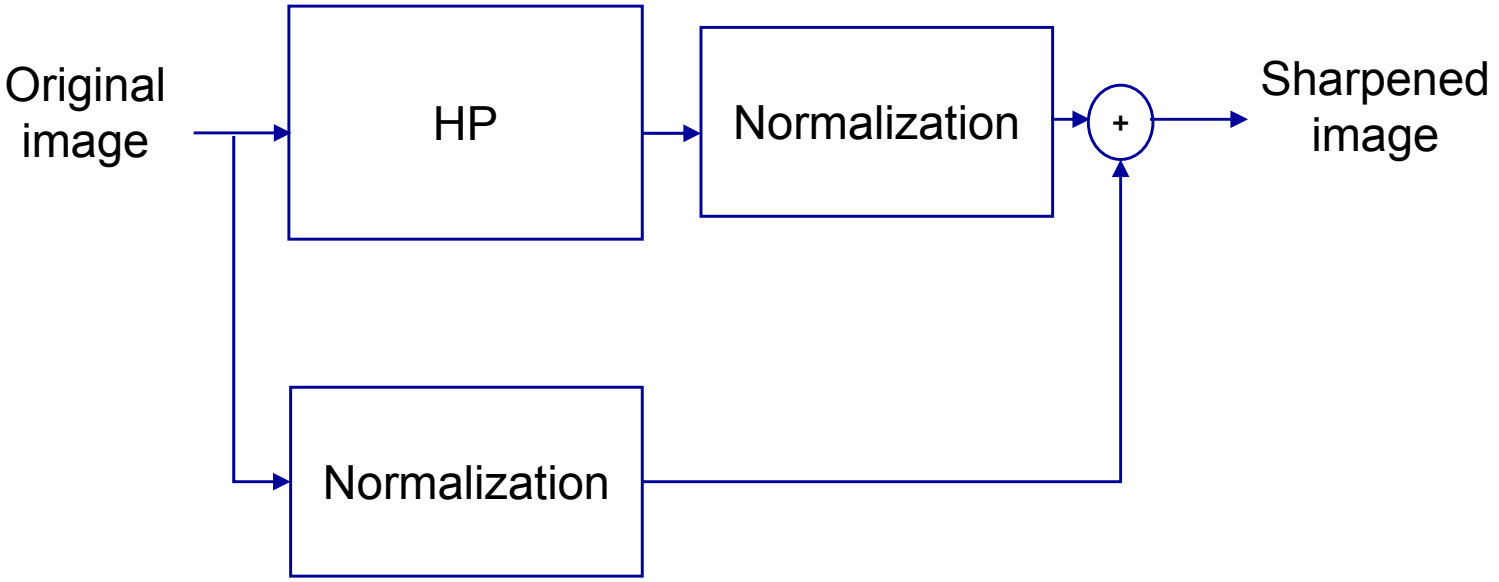


Sharpening Filters

- To highlight fine detail or to enhance blurred details
  - Averaging filters smooth out noise but also blur details
- Sharpening filters enhance details
- May also create artifacts (amplify noise)
- Background: Derivative is higher when changes are abrupt

- Categories of sharpening filters:
  - Basic highpass spatial filtering
  - High-boost filtering

Sharpening



The normalization step subtracts the mean and scales the amplitude of the resulting image by dividing it for the dynamic range (graylevel values are now in the range 0-255)

For the sharpening to be visible, the sharpened and original images must then be displayed using the same set of graylevel values

Basic Highpass Spatial Filtering

- The filter should have positive coefficients near the center and negative in the outer periphery:

Laplacian mask

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

a b
c d

FIGURE 3.39
(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

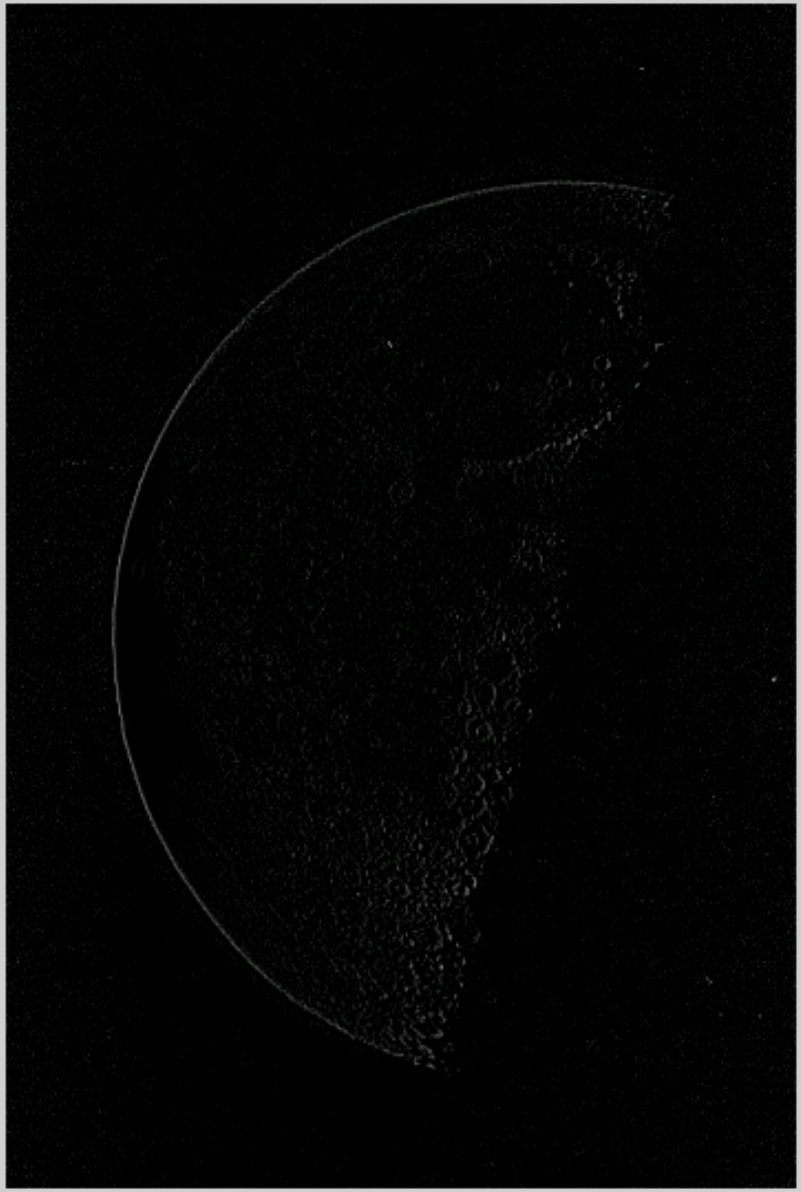
Other Laplacian masks (normalization factor is missing)

Basic Highpass Spatial Filtering

- The sum of the coefficients is 0, indicating that when the filter is passing over regions of almost stable gray levels, the output of the mask is 0 or very small.
- The output is high when the center value differ from the periphery.
- The output image does not look like the original one.
- The output image depicts all the fine details
- Some scaling and/or clipping is involved (to compensate for possible negative gray levels after filtering).

originale

laplaciano



High boost

$$I_{highboost} = cI_{original} + I_{highpass} = (cW_{allpass} + W_{highpass}) * I_{original} = W_{highboost} * I_{original}$$

Examples

$$W_{highboost} = cW_{allpass} + W_{highpass} = c\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4+c & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$W_{highboost} = cW_{allpass} + W_{highpass} = c\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8+c & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

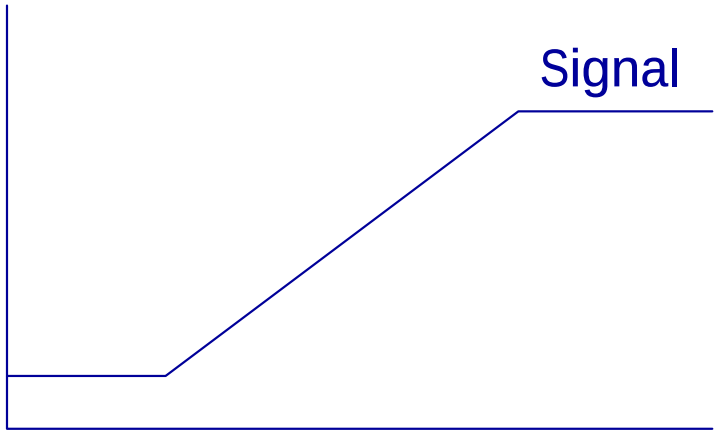
High-boost Filtering

- Highpass filtered image = Original – lowpass filtered image

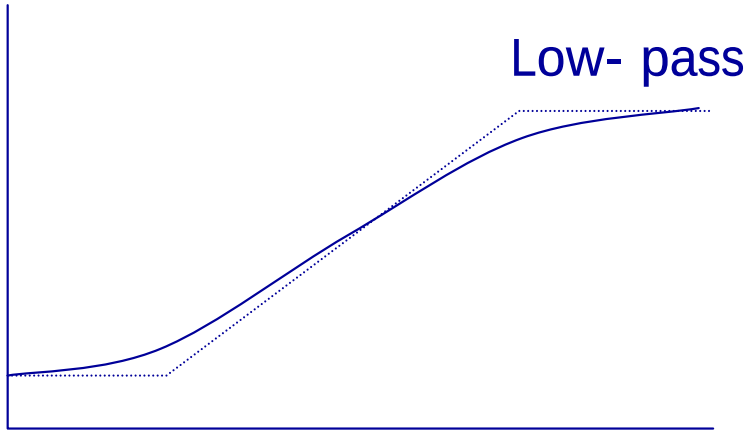
- If A is an amplification factor, then:
  - High-boost = A · original – lowpass
  - = (A-1) · original + original – lowpass
  - = (A-1) · original + highpass

Unsharp masking (if A=2)

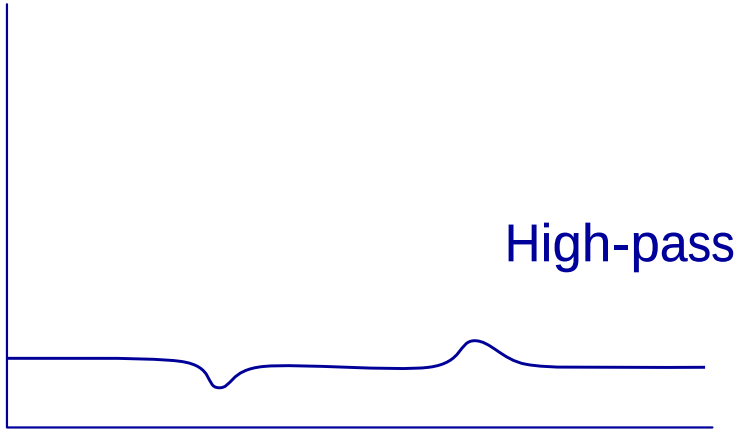
Unsharp Masking and Sharpening operation



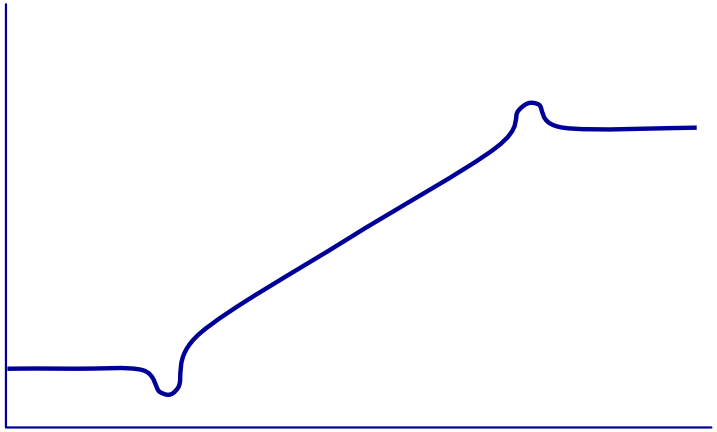
(1)



(2)



(3) = (1) - (2)



(A-1) (1) + (3)

High-boost Filtering

- A=1 : standard highpass result
- A > 1 : the high-boost image looks more like the original with a degree of edge enhancement, depending on the value of A.

$$\frac{1}{9}\times\begin{bmatrix}-1 & -1 & -1\\ -1 & w & -1\\ -1 & -1 & -1\end{bmatrix}$$

w=9A-1, A≥1

A > 1 → Unsharp masking

originale+laplaciano

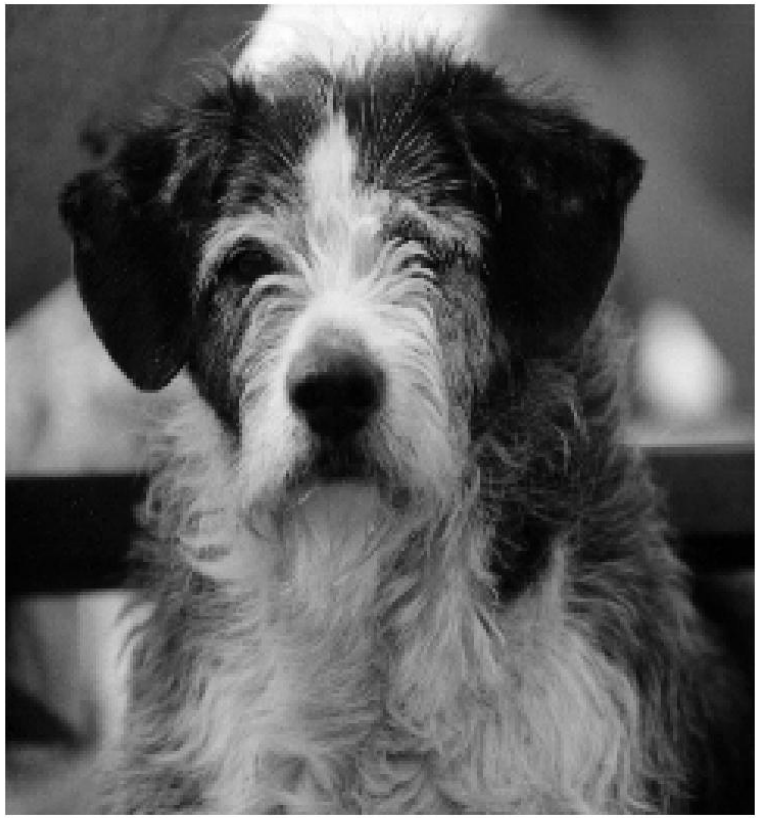
originale

originale+laplaciano

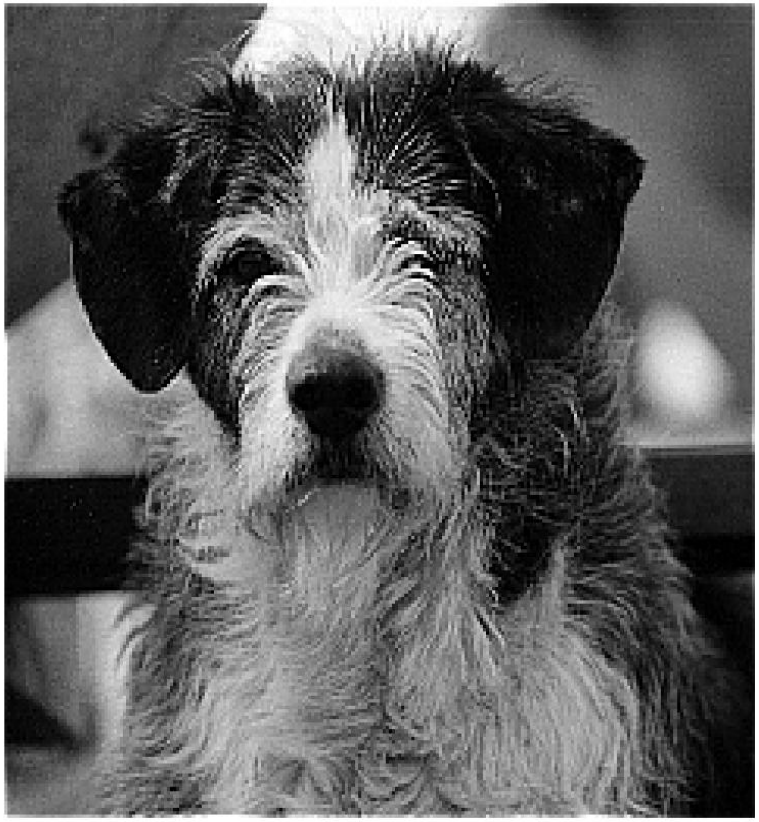
unsharp masking

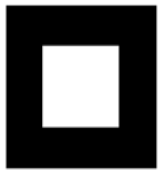
Sharpening: asymmetric HP

Original image



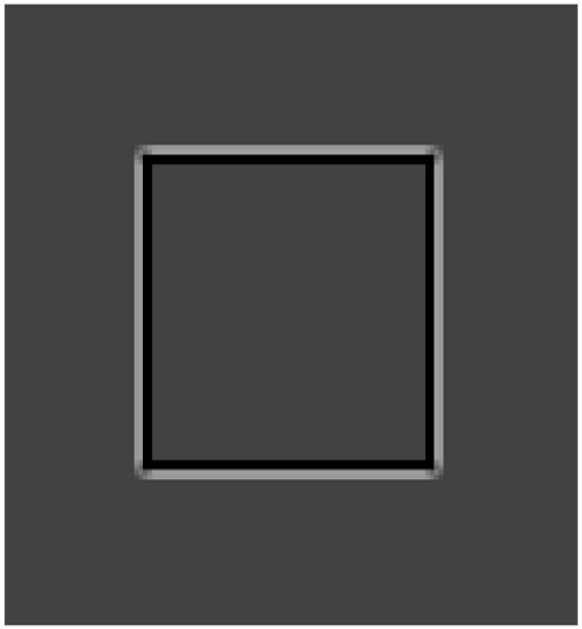
Sharpened image

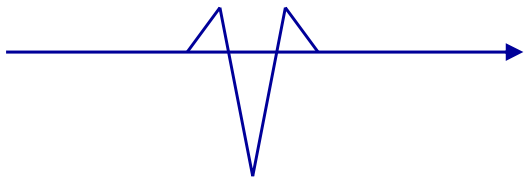




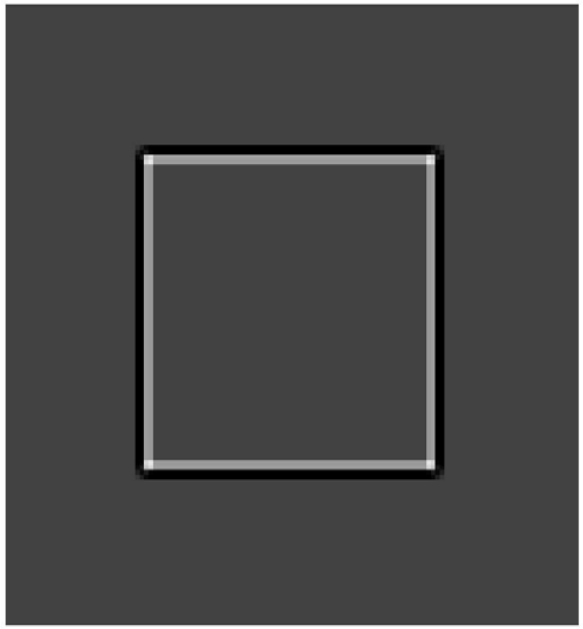
Sharpening: the importance of phase

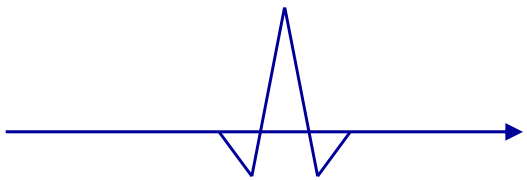
log



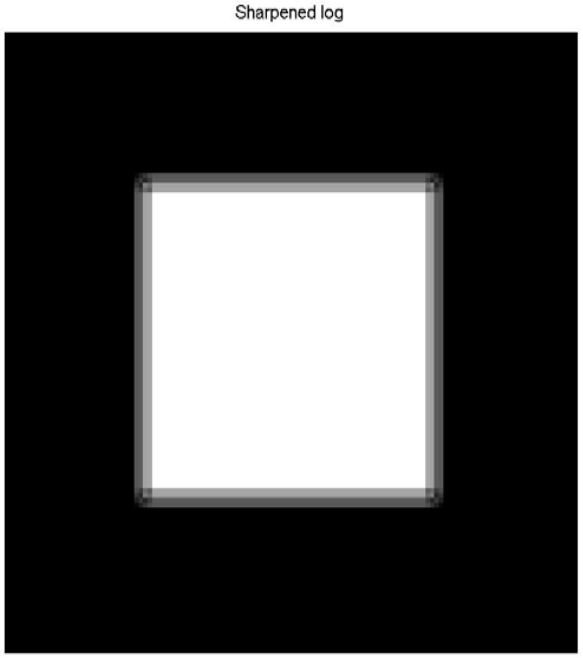


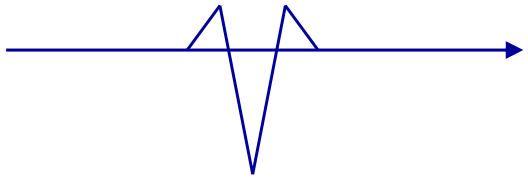
-log

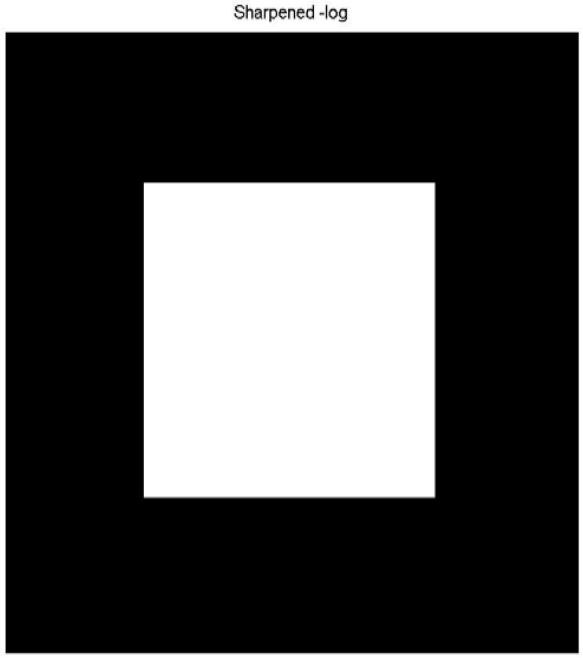


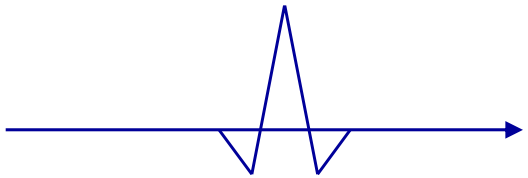


Sharpening: the importance of phase

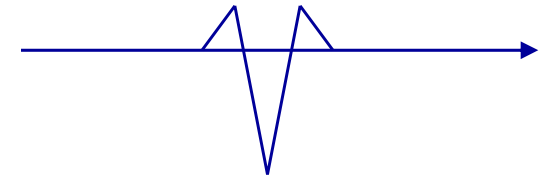


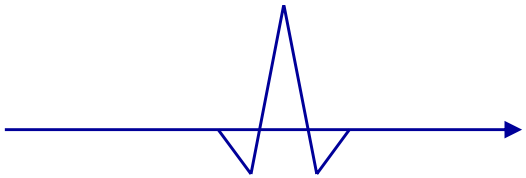


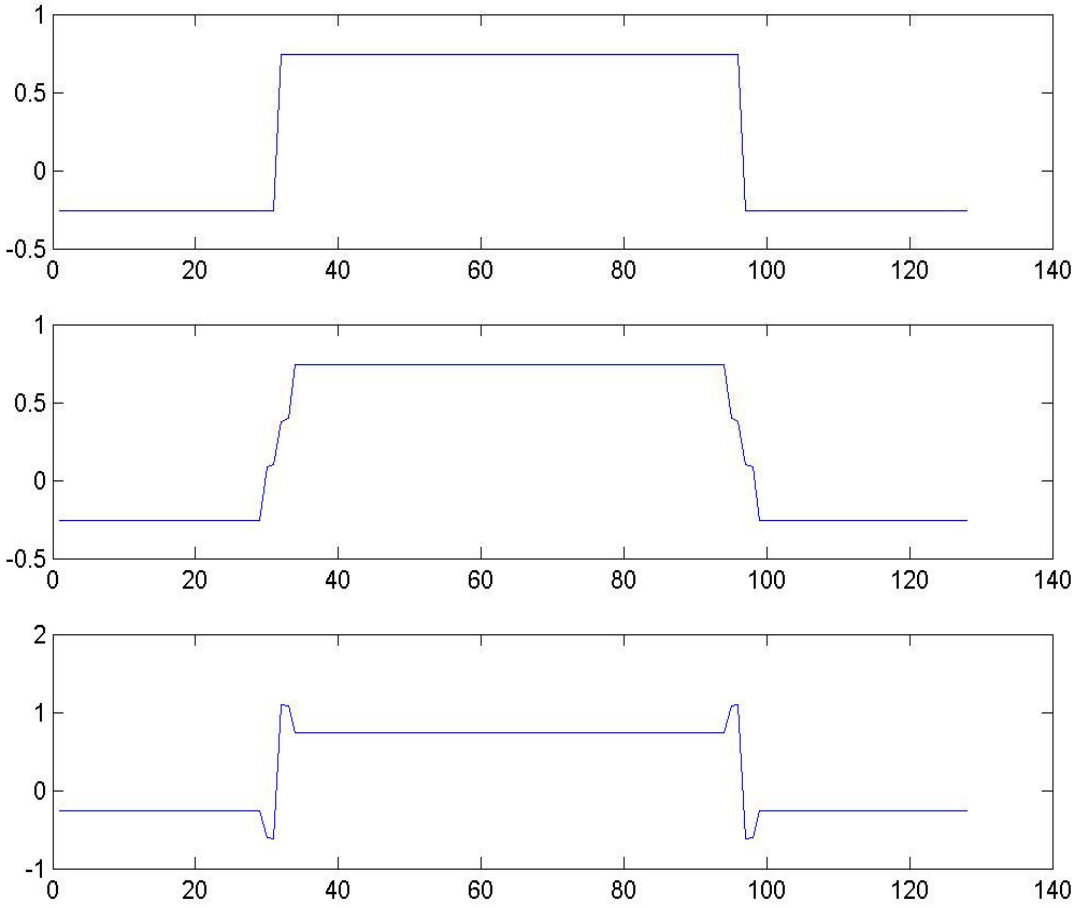




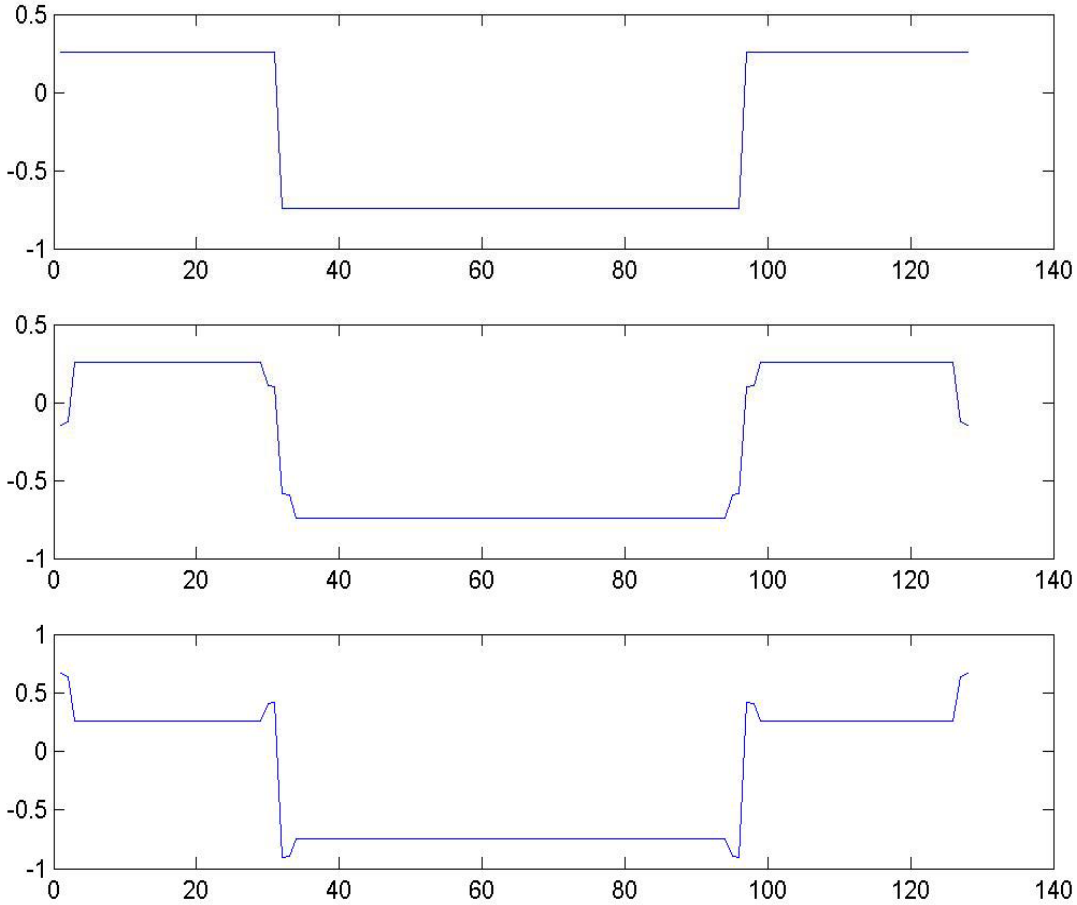
Sharpening: the importance of phase







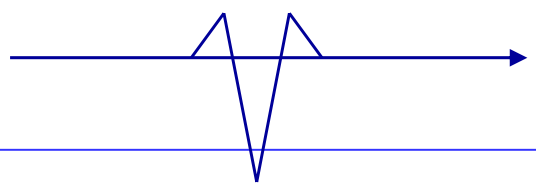
Sharpening: the importance of phase



Original image

Back to the natural image

Sharpened image



Sharpened image