Image Processing

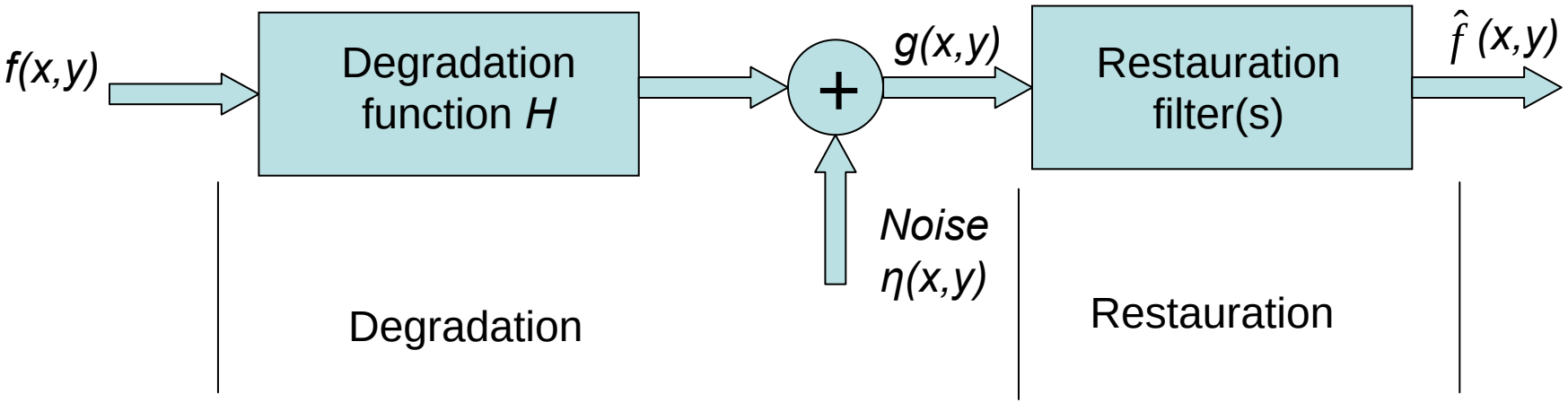
Traitement d'images

Yuliya Tarabalka
yuliya.tarabalka@hyperinet.eu
yuliya.tarabalka@gipsa-lab.grenoble-inp.fr
Tel. 04 76 82 62 68

Noise reduction

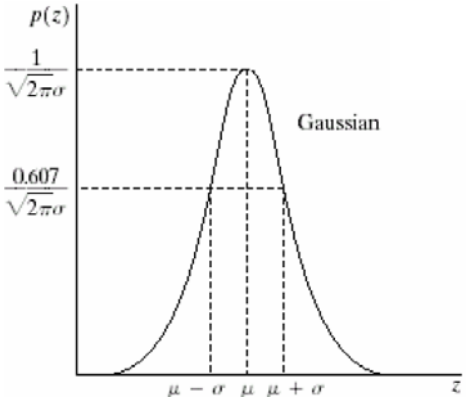
Image restoration

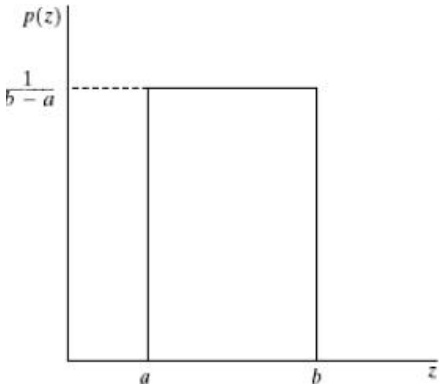
- ***Restoration*** **attempts to *reconstruct* an image that has been degraded by using <u>a priori knowledge of the degradation phenomenon</u>**
  - Modeling the degradation
  - Applying the inverse process in order to recover the original image

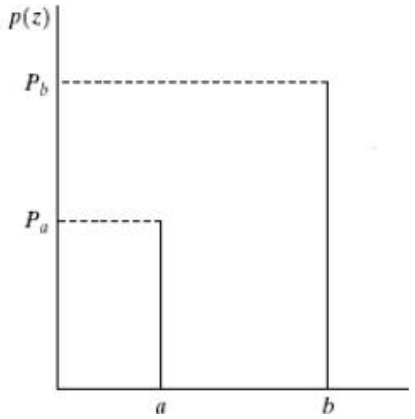


Noise models (Models of noisy images)

Gaussian noise	Uniform noise	Impulse (salt-and-pepper) noise
$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(z-\mu)^2}{2\sigma^2}\}$	$p(z) = \begin{cases} 1/(b-a) & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$	$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$







- used frequently in practice
- macroscopic effect of many microscopic effects

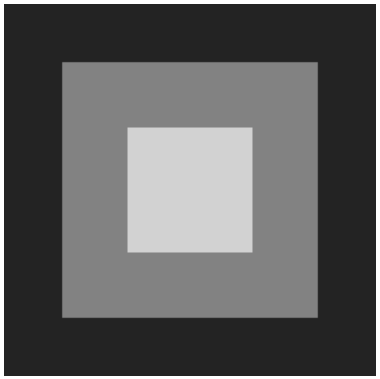
imnoise(I,'gaussian',…)

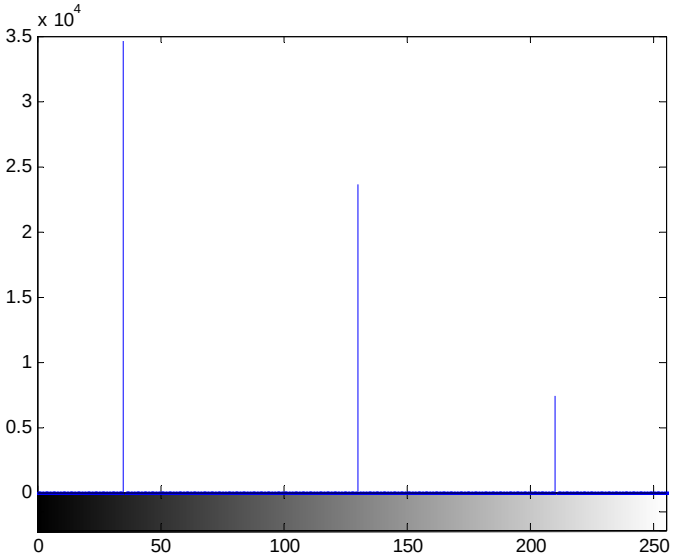
- in acquiring images with a CDD camera

imnoise(I,'salt & pepper',…)

Noise models (Models of noisy images)

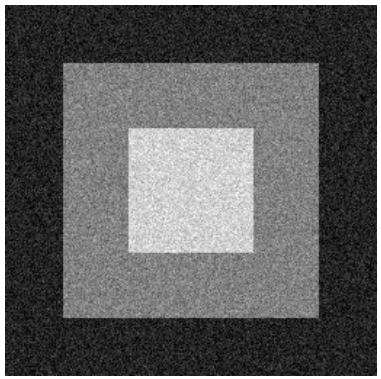
- **Test pattern**

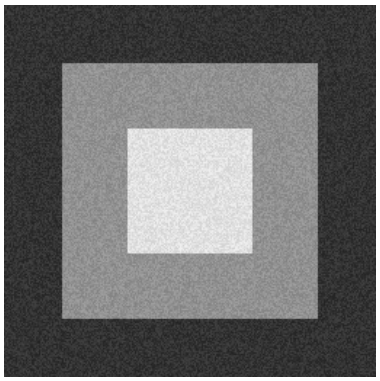


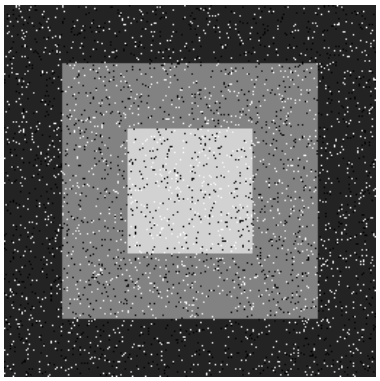


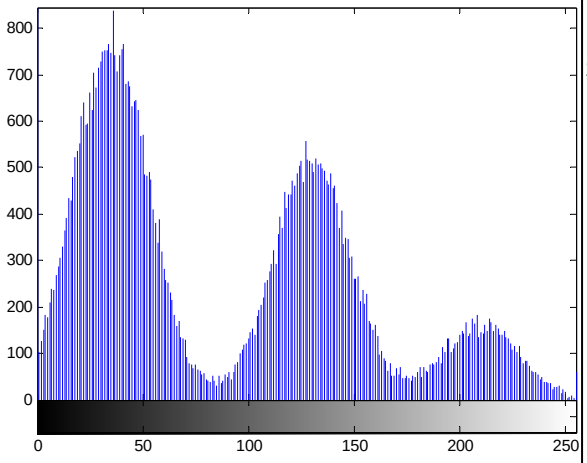
Noise models (Models of noisy images)

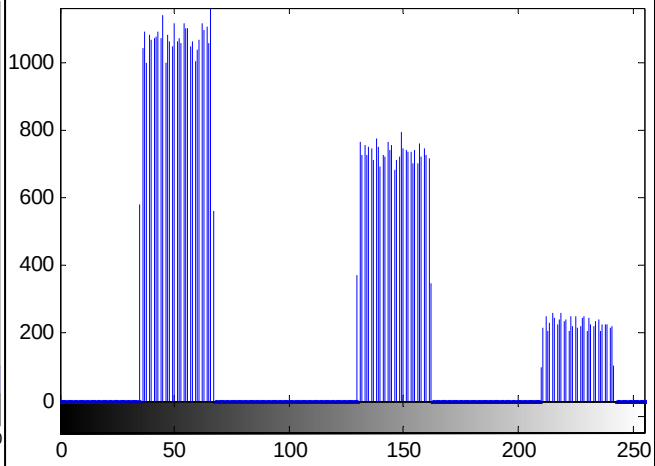
Gaussian noise	Uniform noise	Impulse (salt-and-pepper) noise

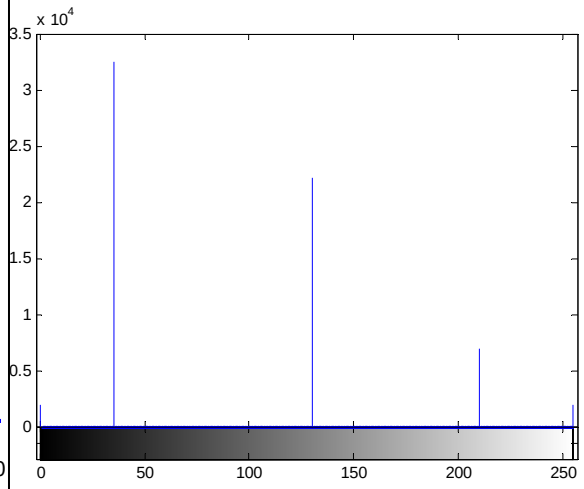






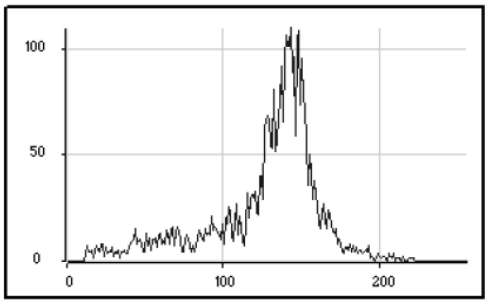


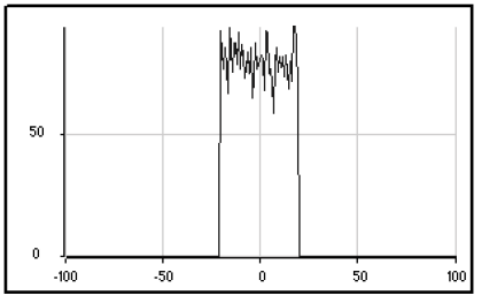


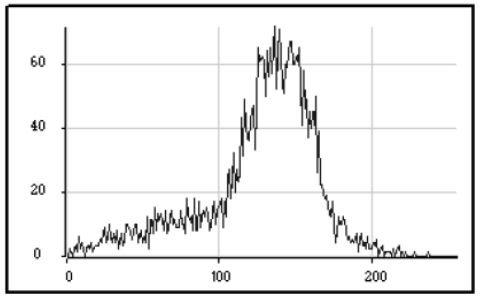


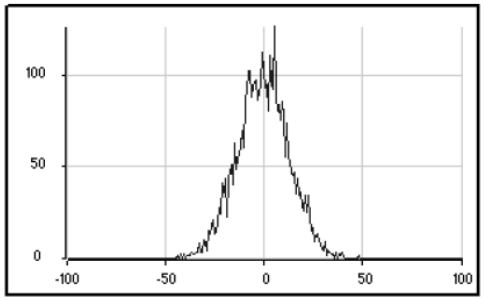
Noise models (Models of noisy images)

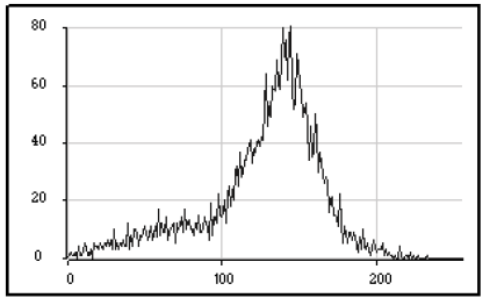
Original image

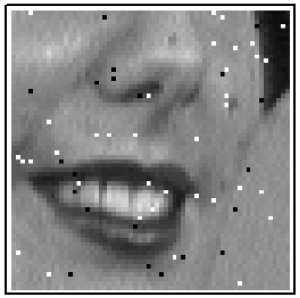


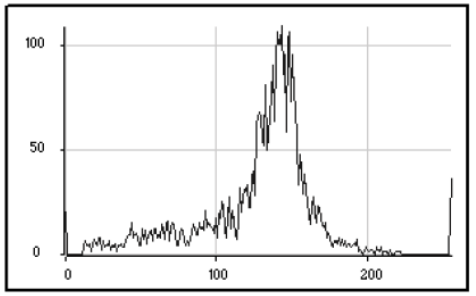












Convolution

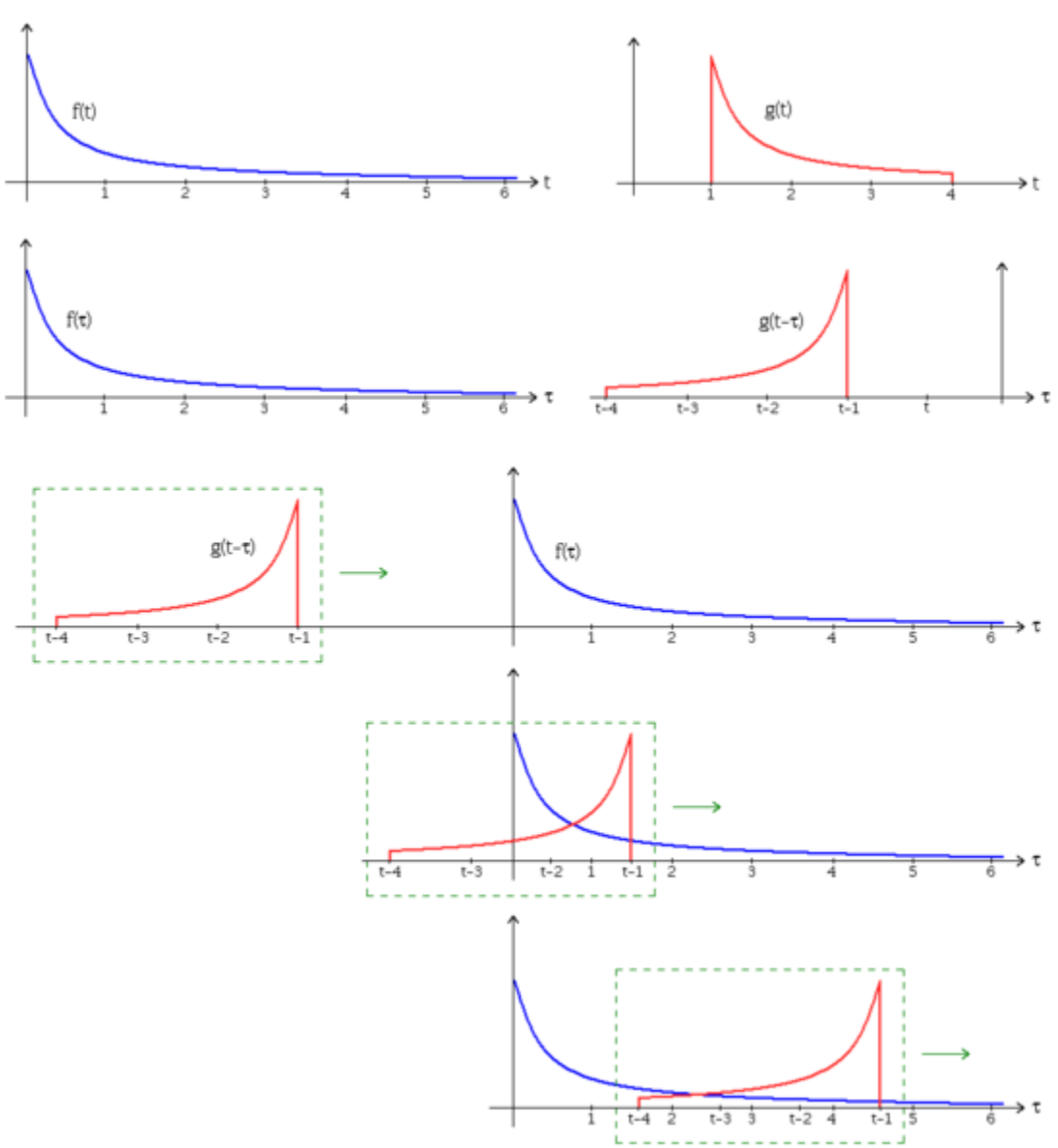
1 D, continue (continuous space) : f*h (t)= $\int_{-\infty}^{\infty} (f(x)) . (h(t - x)) \, dx$

2 D, continue : f*h (u,v)= $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (f(x, y)) . (h(u - x, v - y)) \, dx \, dy$

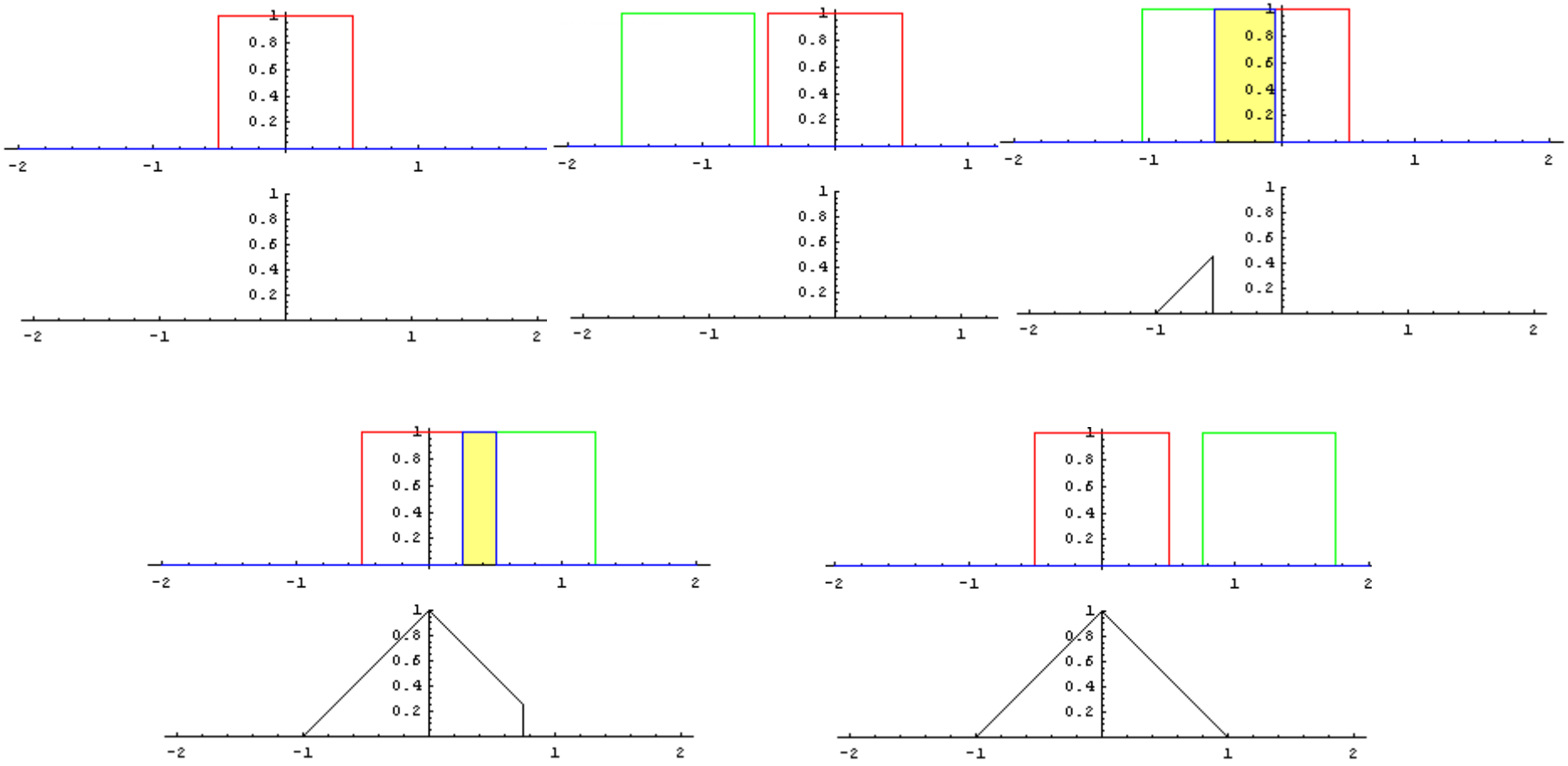
2D, discrete space:

$$f(x,y) * h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) h(x-m, y-n)$$

Convolution

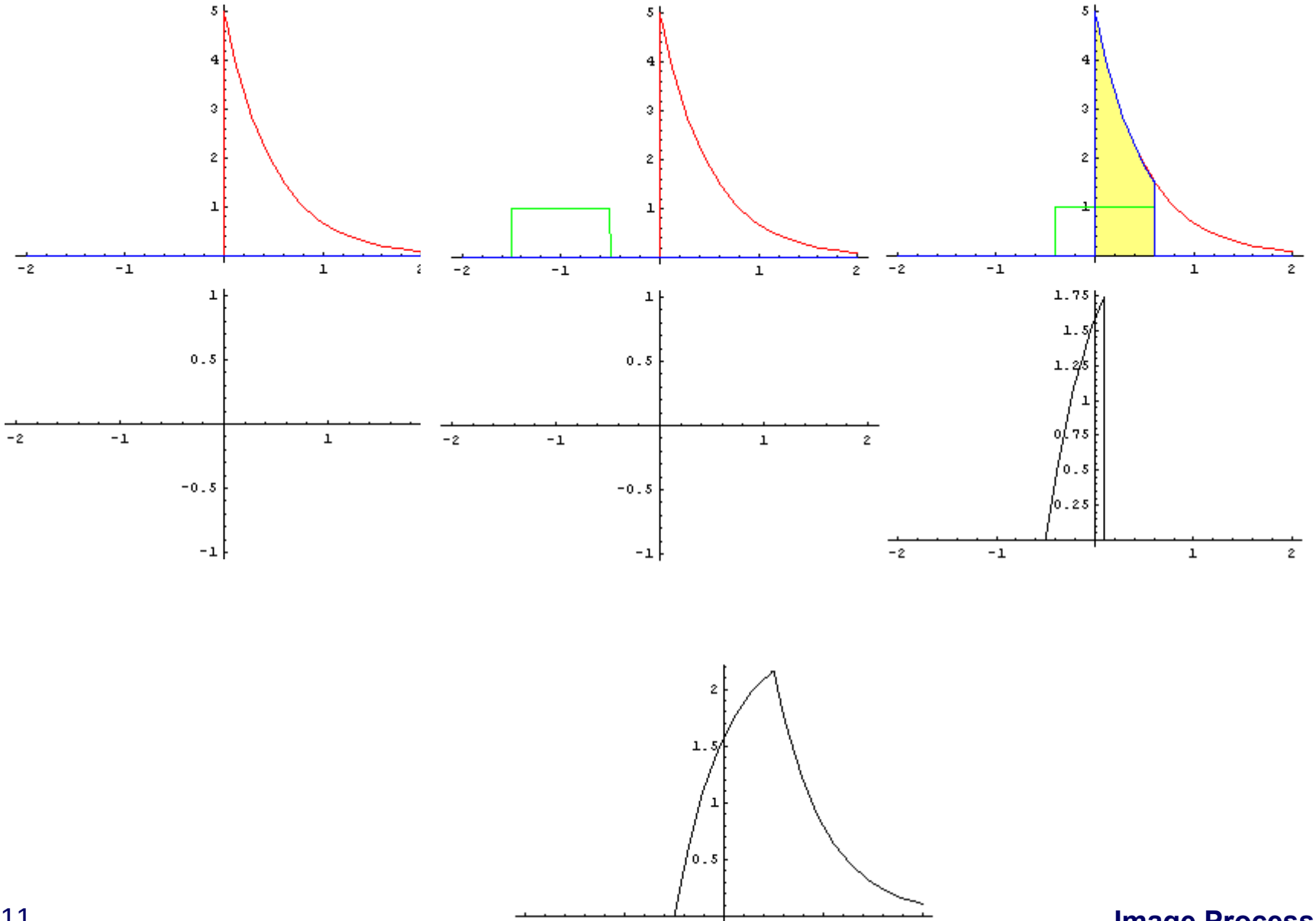


Convolution



Convolution of two square pulses: the resulting waveform is a triangular pulse. The integral of their product is the area of the yellow region

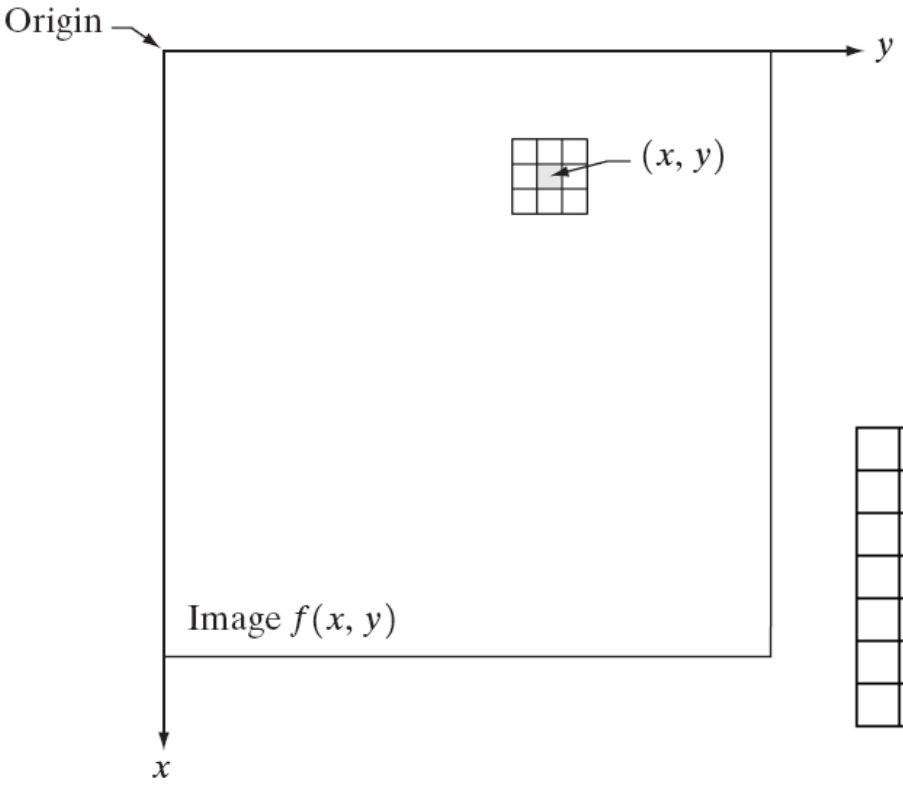
Convolution



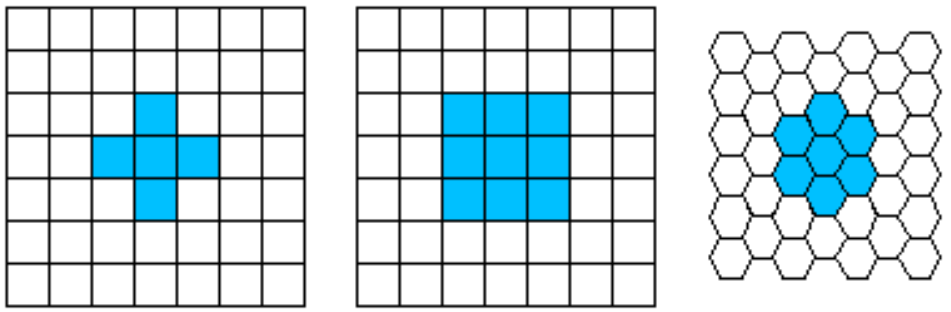
Spatial filtering (neighborhood operations)

- **The value of each pixel (x,y) in the output image is computed from the values of the pixels in the *neighborhood* of (x,y):**

$$g(x,y) = T[f(x,y)]$$



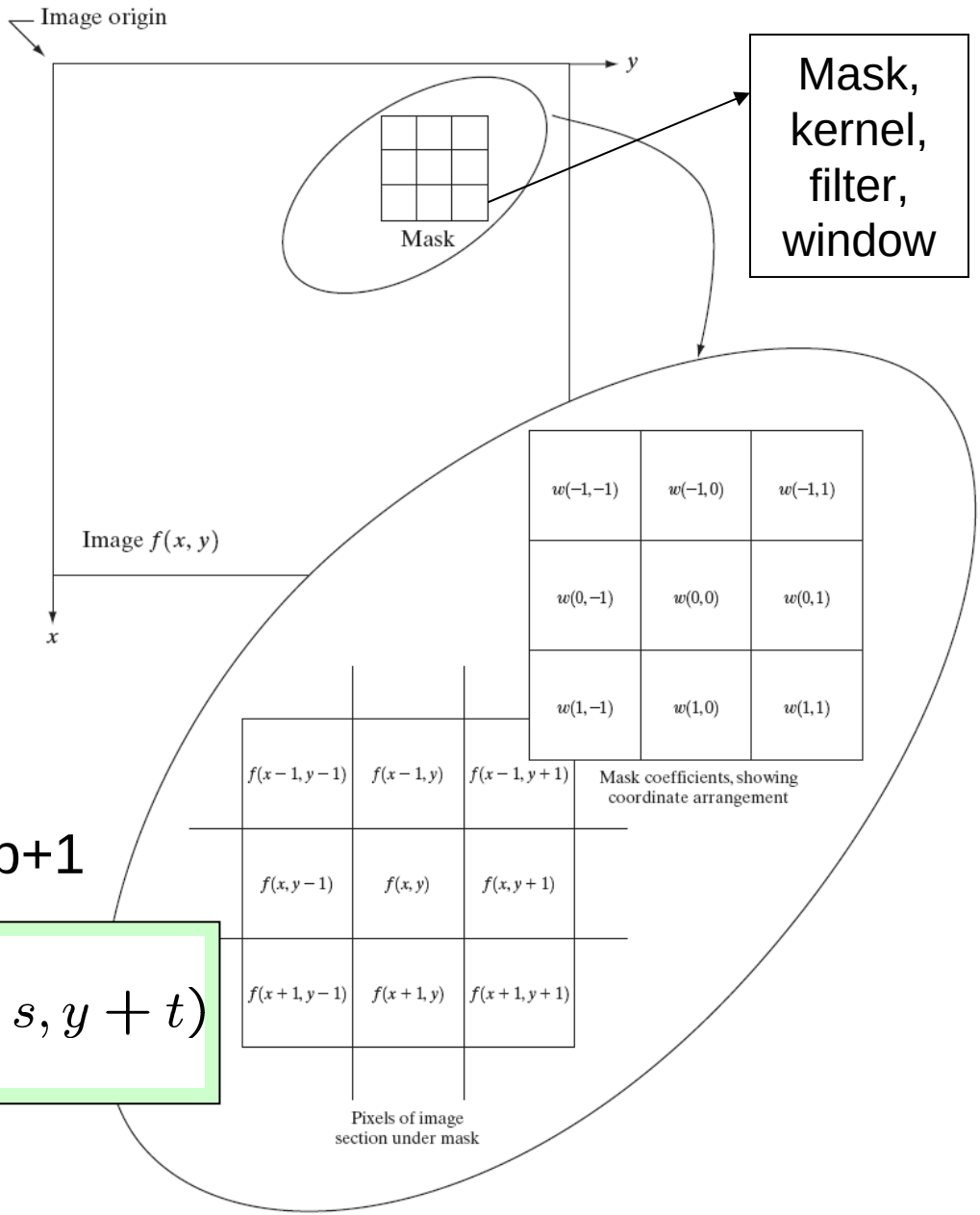
Sliding filtering windows:



Spatial filtering (neighborhood operations)

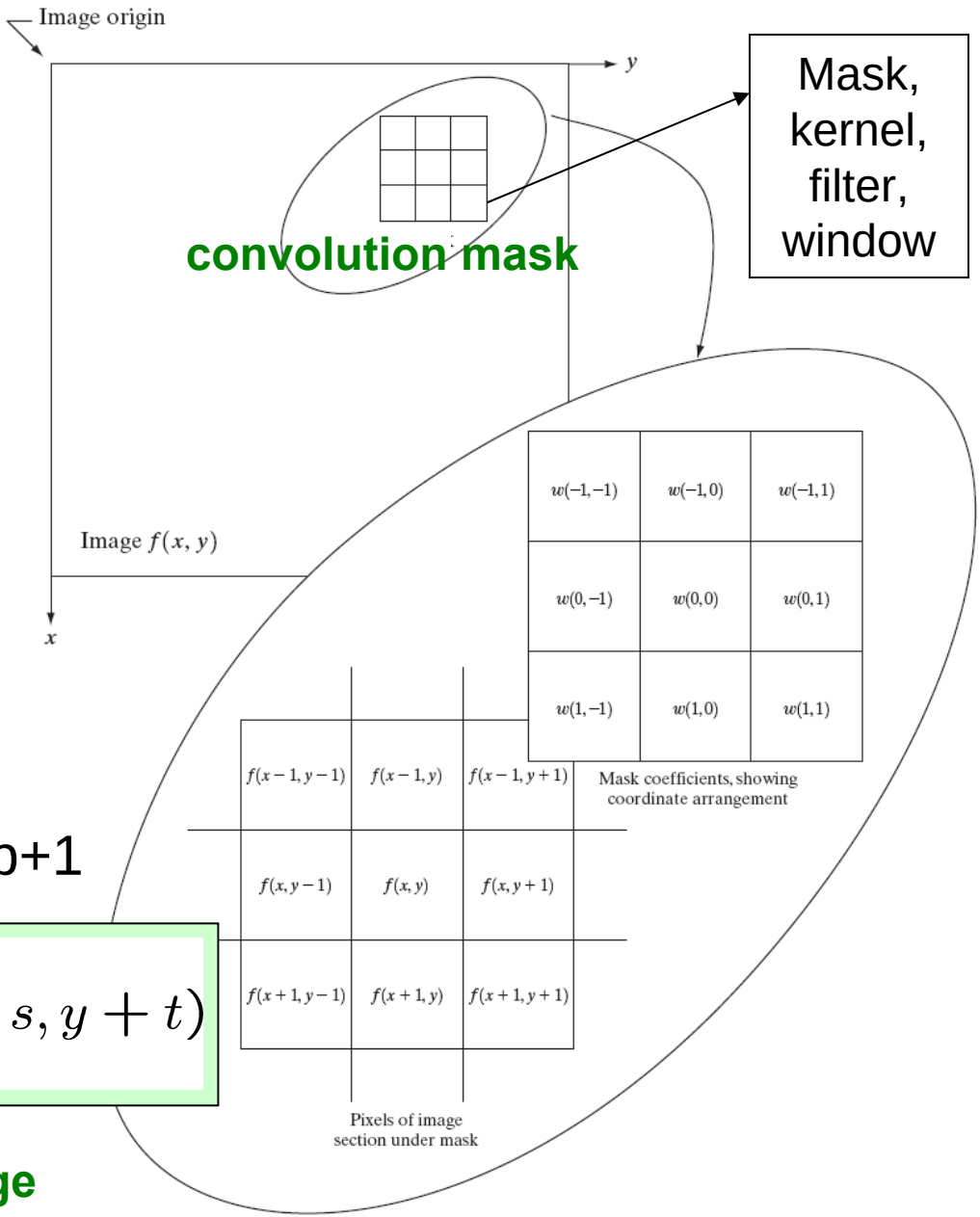
- ***Linear filtering*: the response of the filter is given by a sum of products of the filter coefficients and the corresponding image pixels within the filter mask**
  - The mask of size m x n is centered at (x,y)
  - We assume m=2a+1, n=2b+1

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$



Spatial filtering (neighborhood operations)

- ***Linear filtering*: the response of the filter is given by a sum of products of the filter coefficients and the corresponding image pixels within the filter mask**
  - The mask of size m x n is centered at (x,y)
  - We assume m=2a+1, n=2b+1



$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

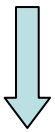
convolving a mask with an image

Smoothing linear filters

- **The output: (weighted) average of the pixels contained in the neighborhood of the filter mask**
- **Averaging, lowpass filters**

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

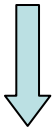


Mean filter: replace each pixel by the arithmetic mean of its neighbors and itself

- reduce noise (e.g., gaussian)
- but blur edges

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

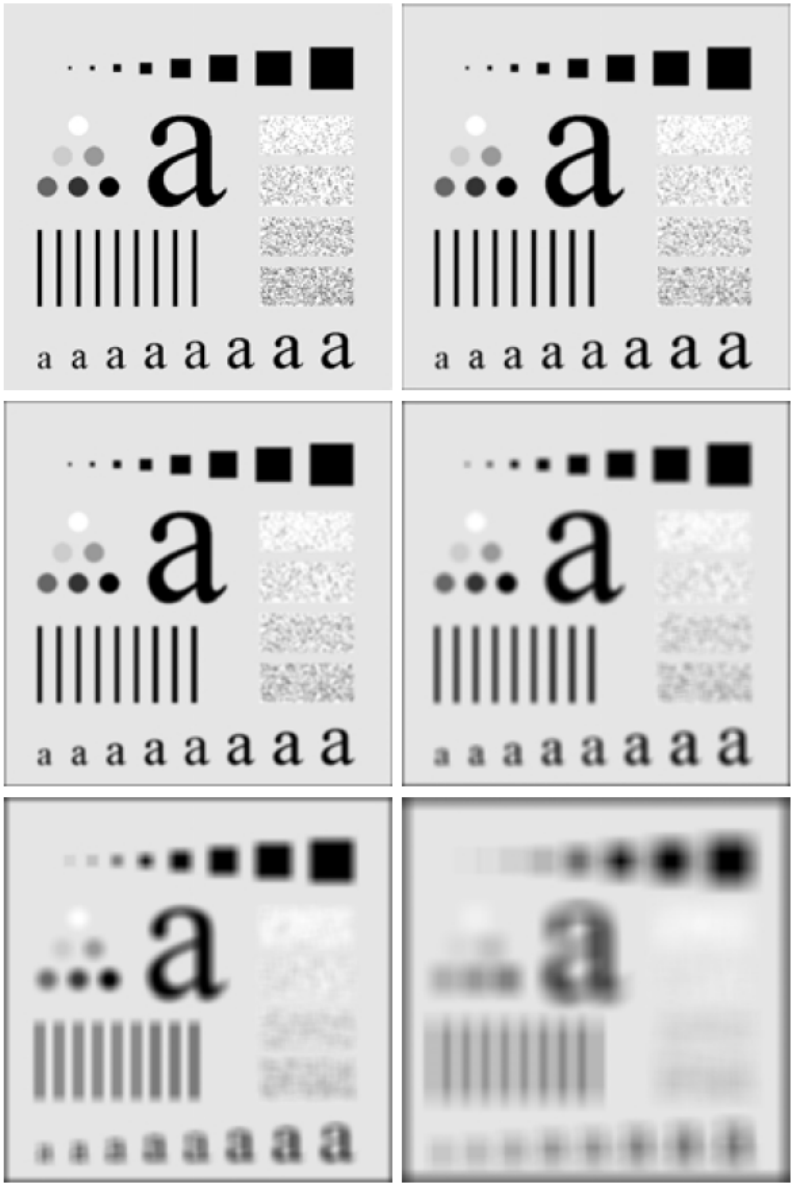


Weighted average: reduce blurring in the smoothing process

Smoothing linear filters

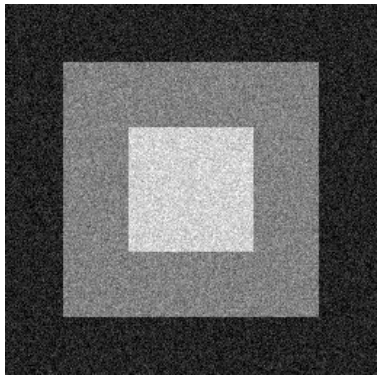
- **Matlab:**
  conv2(), imfilter()

Results of smoothing with square averaging Filter masks of sizes n = 3, 5, 9, 15, and 35

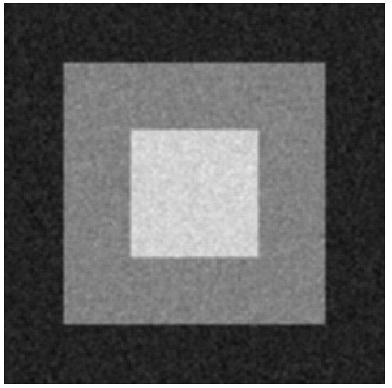


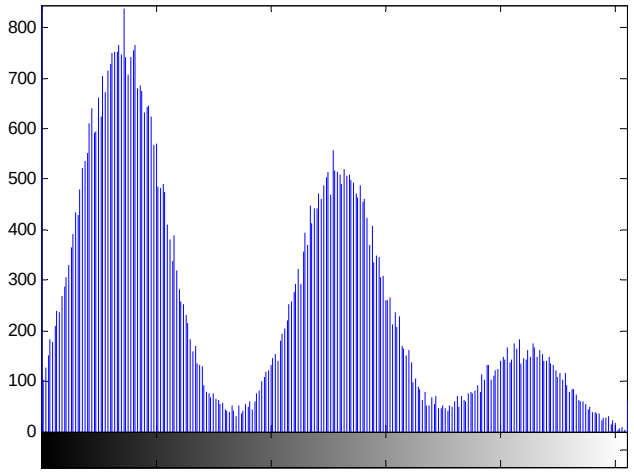
Smoothing linear filters

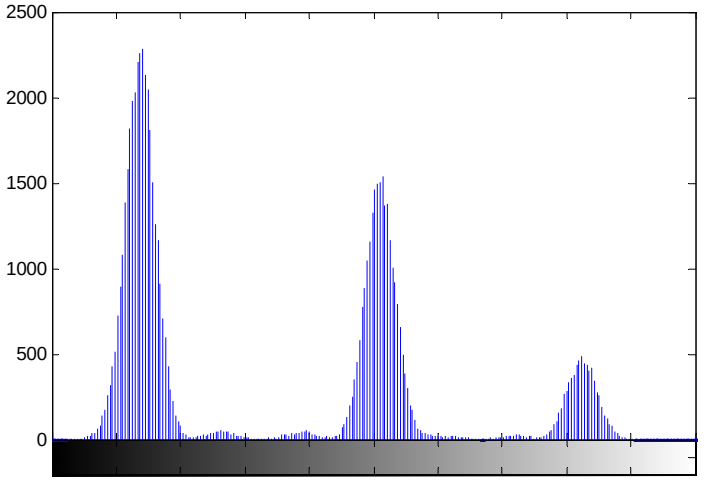
Image corrupted by Gaussian noise



Mean filtering (3x3 mask)







Smoothing non-linear filtering

- ***Order filters***: **response is based on ordering (ranking) the pixels contained within the considered window**

$$\begin{bmatrix} 10 & 134 & 124 \\ 152 & 45 & 51 \\ 61 & 51 & 88 \end{bmatrix}$$

Smoothing non-linear filtering

***Order filters*: response is based on ordering (ranking) the pixels contained within the considered window**

$$\begin{bmatrix} 10 & 134 & 124 \\ 152 & 45 & 51 \\ 61 & 51 & 88 \end{bmatrix}$$

10<45<51<51<61<88<124<134<152

- ***Median filter* replaces the value of a pixel by the median of the gray levels in its neighborhood**
  - For our example it is 61:
    10<45<51<51<**61**<88<124<134<152
  - Especially effective for impulse (salt-and-pepper) noise

Smoothing non-linear filtering

***Order filters*: response is based on ordering (ranking) the pixels contained within the considered window**

$$\begin{bmatrix} 10 & 134 & 124 \\ 152 & 45 & 51 \\ 61 & 51 & 88 \end{bmatrix}$$

10<45<51<51<61<88<124<134<152

- ***Middle filter:* the value of a pixel is computed as the average of the maximum and minimum values in its neighborhood**
  - For our example it is 81:
    10<45<51<51<61<88<124<134<**152**
  - Particularly effective for uniform noise

Smoothing non-linear filtering

Image corrupted by salt-and-pepper noise

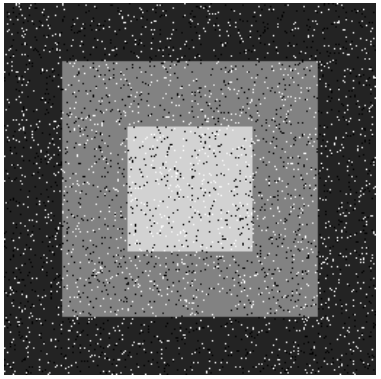


Image after median filtering (Matlab: *medfilt2()*)

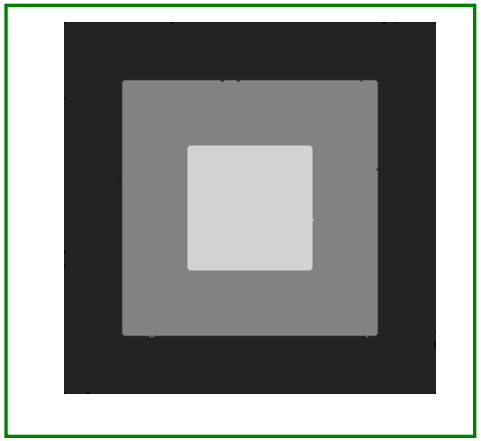
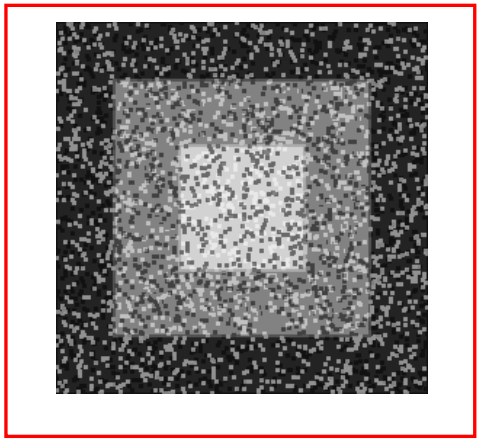
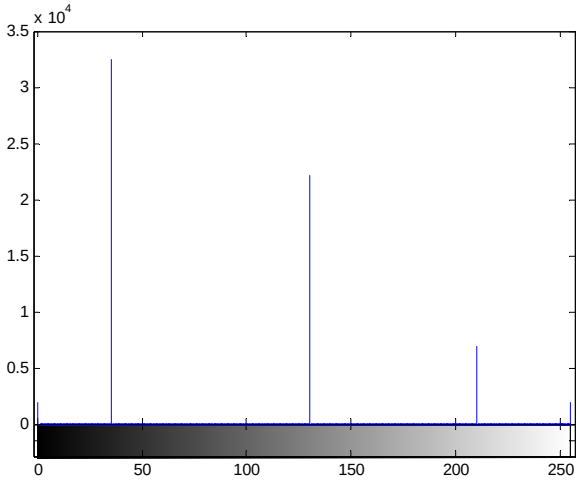
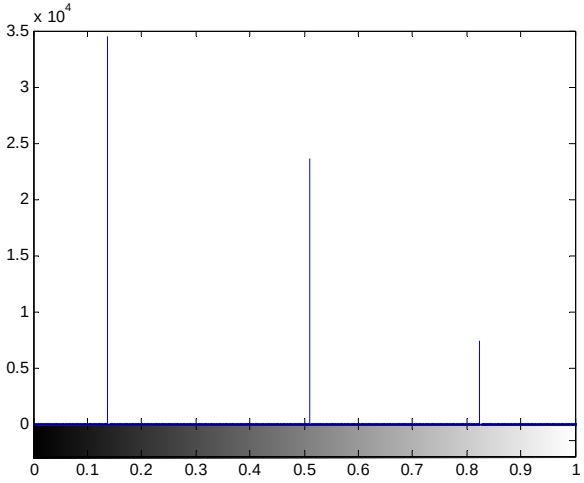
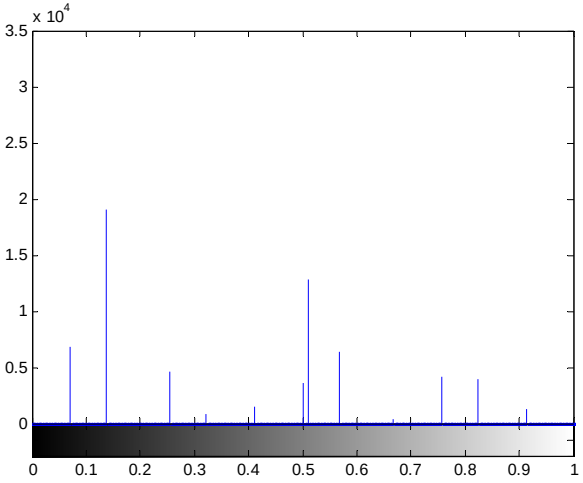


Image after middle filtering









Contrast enhancement

If *mean* > *middle*,

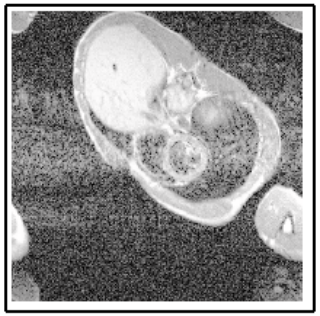
***max(mean, middle)*;**

otherwise *min(mean, middle)*

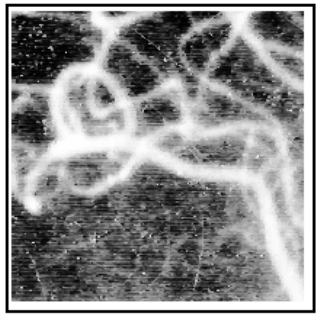
- *Contrast enhancement accentuates edges, but also noise*

Real images

- **Reference, non-noisy image is not available**
- **One must assume (guess) a type of noise**
  - Choose the part of an image that is assumed to be uniform (homogeneous) without the noise
  - Test the hypothesis and choose the best filter



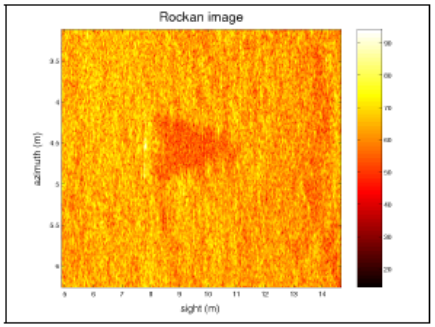
MRI



angiography (X-Ray)

echography (Ultra-Sound)

CCD



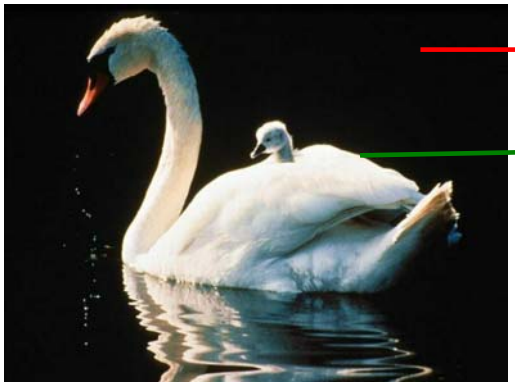
sonar

Edge detection

Segmentation & edge detection

- ***Segmentation*** **partitions an image into its constituent regions or objects**
- **Image segmentation algorithms are based on one of two (or two) basic properties of intensity values: discontinuity and similarity**
- ***Edge detection*** **is used to isolate objects based on abrupt changes in intensity**

Edge = fast change in intensity

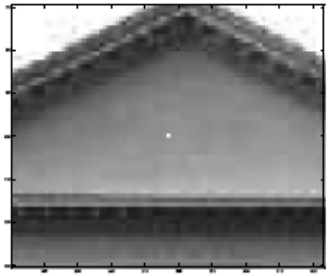


Detection of discontinuities

- **The most common way to look for discontinuities is to convolve a mask with an image**
- **Point detection:**
  - filtering an image with a mask

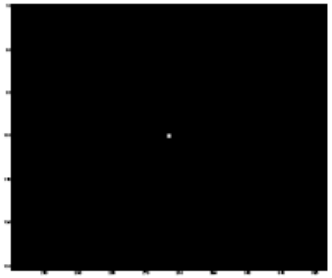
-1	-1	-1
-1	8	-1
-1	-1	-1

  - thresholding



Original image

After filtering



After thresholding

Edge detection

- ***Edge*** **is a set of connected pixels that lie on the boundary between two regions**
- ***Derivative*** **is a measure of how a function changes as its input changes**
- **First-order derivative of a one-dimensional function f(x):**

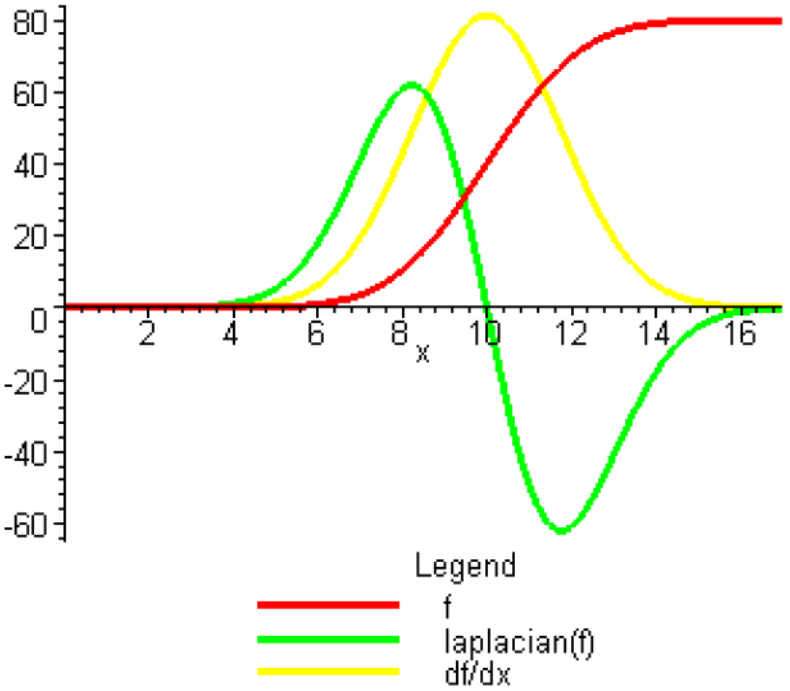
$$\frac{\partial f}{\partial x} = f(x) - f(x-1)$$

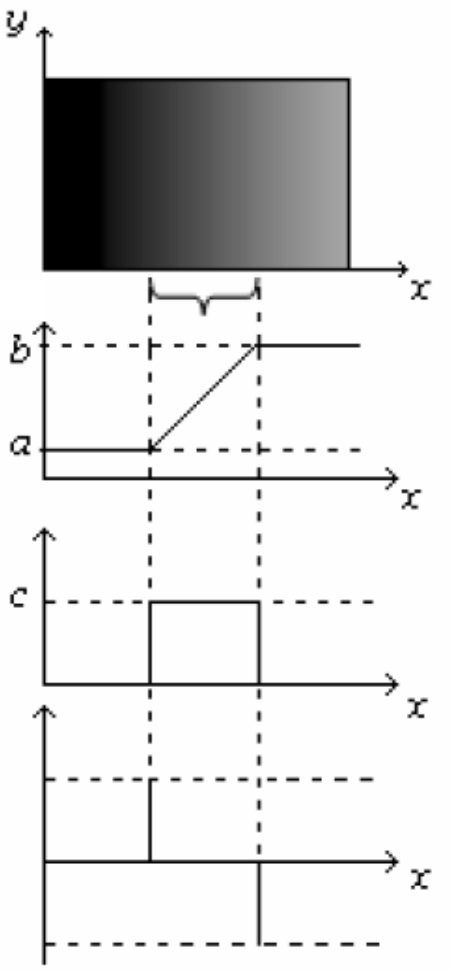
- **Second-order derivative:**

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Edge detection

Model of an ideal edge





Model of a ramp edge

Grey-level profile

First derivative

Second derivative

Edge detection

- **The gradient of an image f(x,y) at location (x,y) is defined as the vector**

$$\nabla \vec{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- **Magnitude of the gradient** $\left\| \nabla \vec{f} \right\| = \sqrt{G_x^2 + G_y^2}$

- **Convolution masks:**

G_x

0	0	0
-1	1	0
0	0	0

G_y

0	-1	0
0	1	0
0	0	0

Edge detection

- **Convolution masks:**

0	0	0
0	-1	0
0	0	1

Roberts

0	0	0
0	0	-1
0	1	0

"Cross" operator

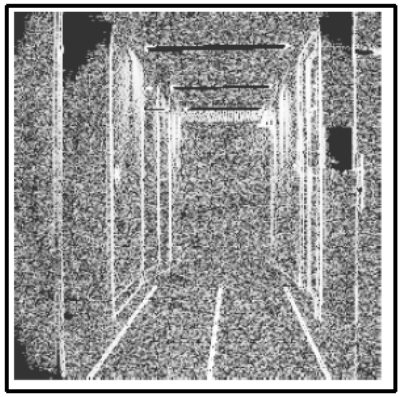
G_x

0	0	0
-1	0	1
0	0	0

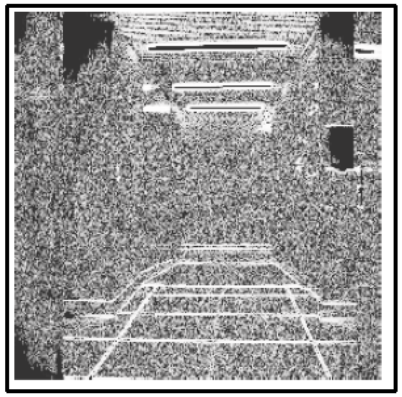
G_y

0	-1	0
0	0	0
0	1	0

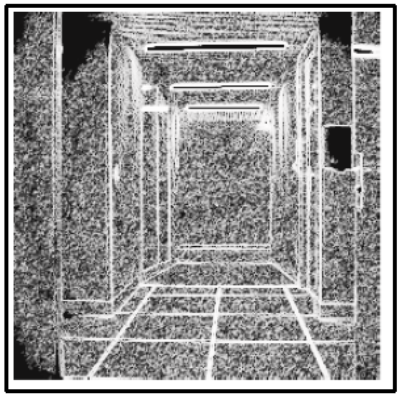
Edge detection



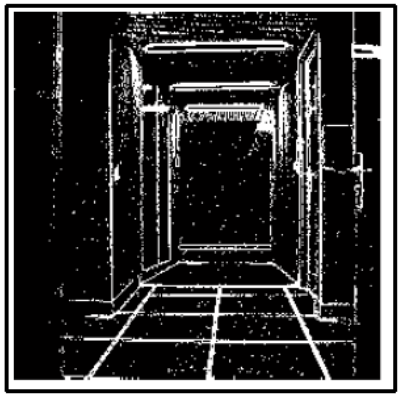
|grad1|
Horizontal gradient



|grad2|
Vertical gradient



max(|grad1|, |grad2|)



Threshold

- **We define a point in an image as being an *edge point* if its two-dimensional first-order derivative is greater than a specific threshold**
- **Problem 1: gradient is sensitive to noise!**

Edge detection

- **Filtering in one dimension + detection in another dimension**

Prewitt:

G_x

-1	0	1
-1	0	1
-1	0	1

G_y

-1	-1	-1
0	0	0
1	1	1

Sobel:

G_x

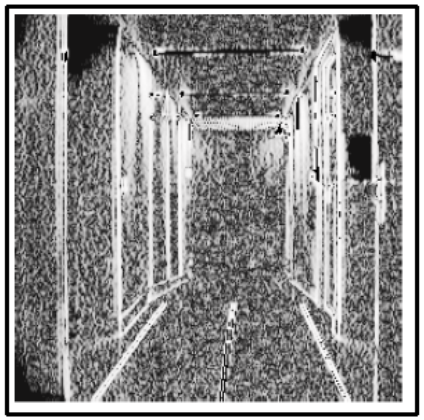
-1	0	1
-2	0	2
-1	0	1

G_y

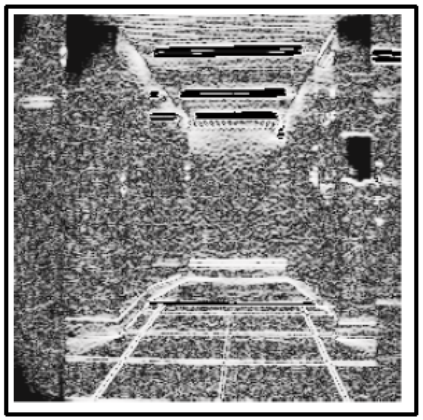
-1	-2	-1
0	0	0
1	2	1

Edge detection

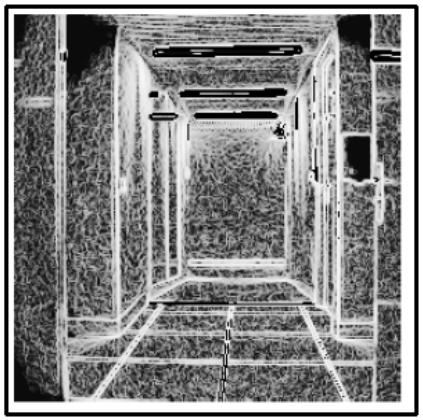
- **Filtering in one dimension + detection in another dimension**



|sobel1|



|sobel2|



max

threshold

- **Problem 2: Some edges are not closed → post-processing to link (close) edges**

Edge detection

- **The Laplacian of a 2-D function f(x, y) is a second-order derivative defined as**

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

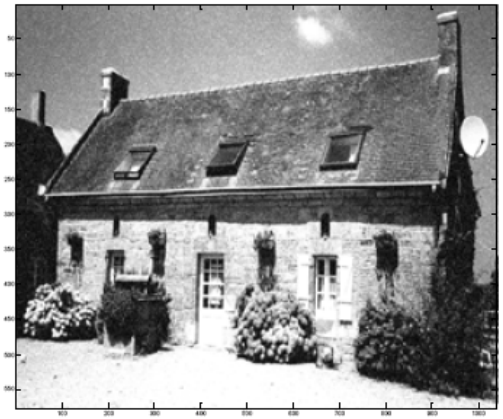
The Laplacian mask is isotropic

0	-1	0
-1	4	-1
0	-1	0

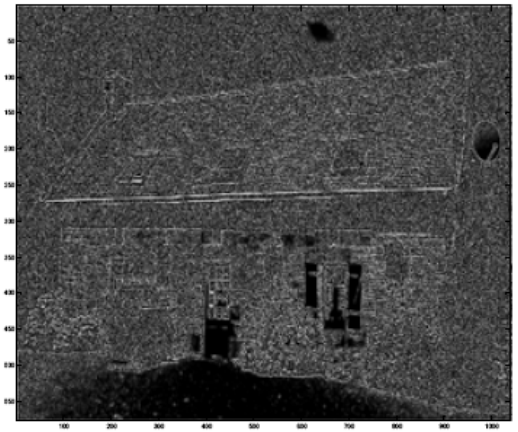
-1	-1	-1
-1	8	-1
-1	-1	-1

- **Edge points are defined as the zero-crossings of the Laplacian**
- **Problem: the Laplacian is very sensitive to noise**

Edge detection



Original noisy image



After Laplacian filtering

After Prewitt filtering

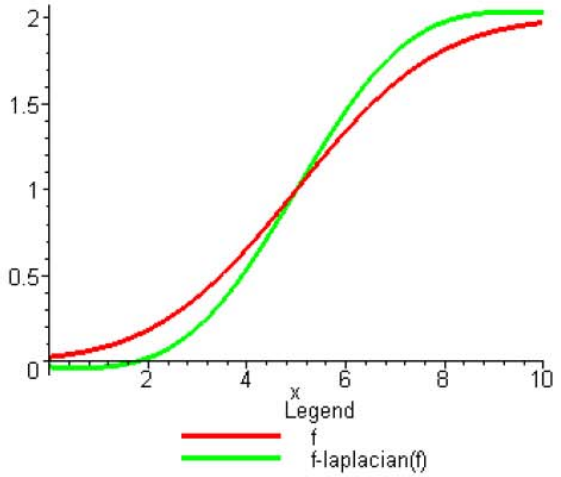
- **There are other advanced edge detection algorithms:**
  - ➢ Canny edge detector

Enhancement

- **Edges can be accentuated by applying $f(x, y) - \Delta f(x,y)$**

0	-1	0
-1	5	-1
0	-1	0

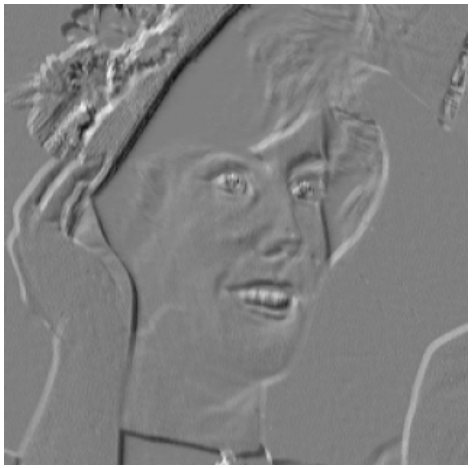
-1	-1	-1
-1	9	-1
-1	-1	-1

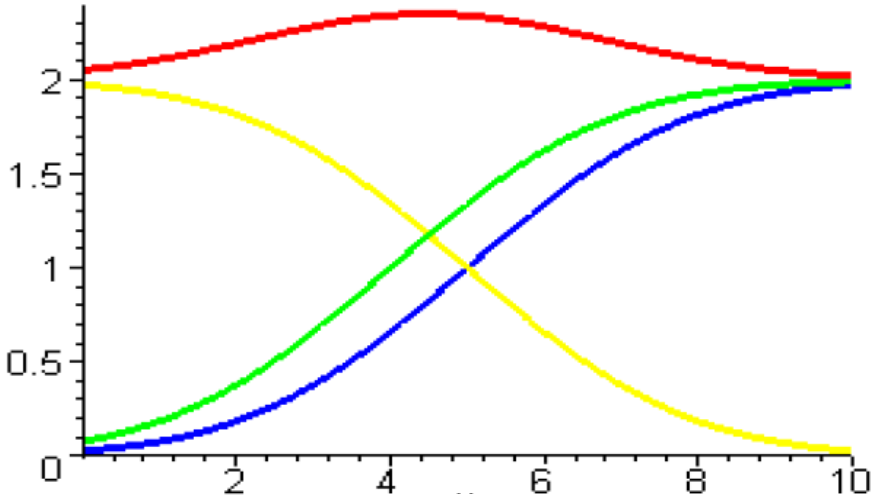


- **Sensitive to noise**

Relief effect (“emboss”)

- The ***relief effect*** **can be obtained by summing an image negative with the translated original image**





Orientation analysis