Recap of the previous lesson

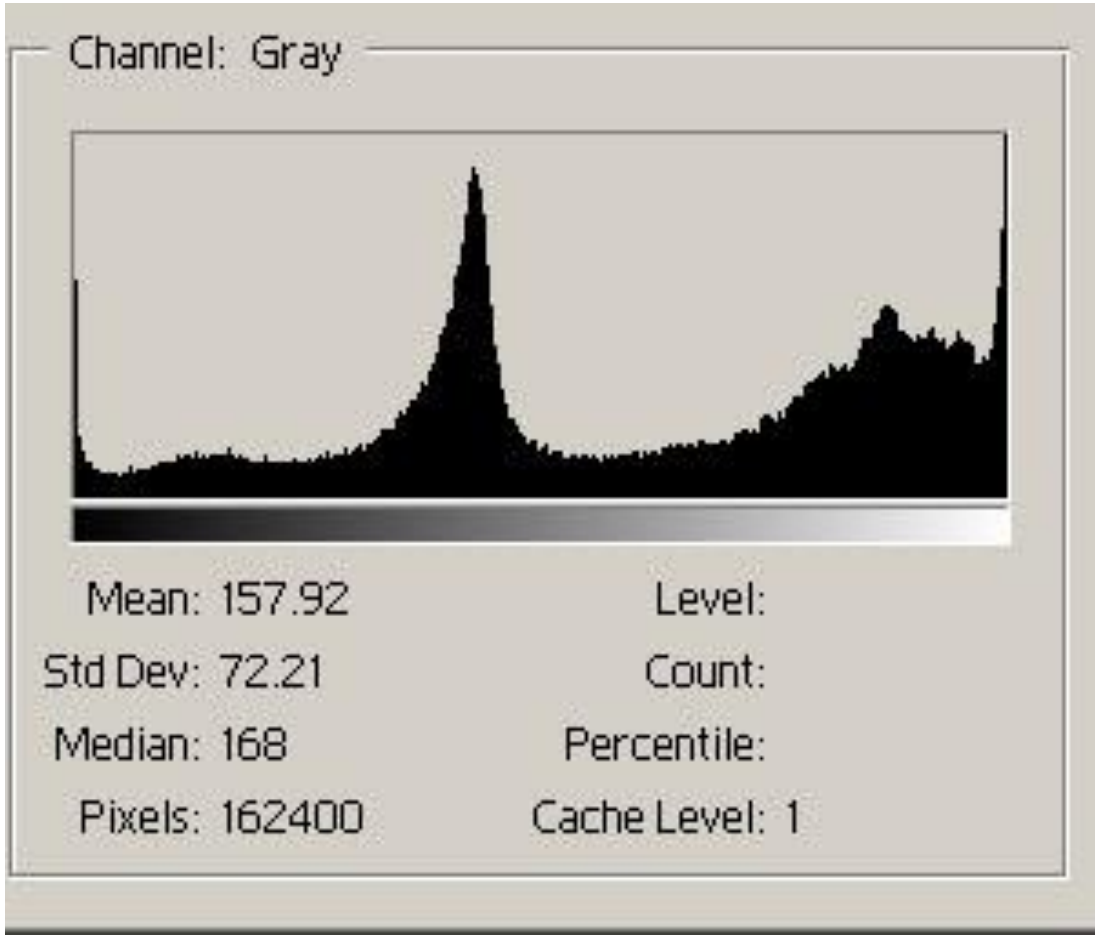
Histogram

- For each gray level, it reports the number of pixels of that color.
- For an image I[m,n] we have

H(k) = number of pixels of value k

- And the sum of all H is exactly *m*x*n*
- The histogram is useful for an immediate understanding of the characteristics of the image.

Histogram


Channel: Gray
Mean: 157.92
Std Dev: 72.21
Median: 168
Pixels: 162400
Level:
Count:
Percentile:
Cache Level: 1

Arithmetic on images

Operating arithmetically, it may happen that a pixel has:

a) A negative value;

b) A value greater than the maximum (typically 255);

c) A non-integer value (easily solved by approximation or truncation);

Normalization

Problems (a) and (b) on the previous slide are called range problems.

Two are the most common solutions:

- Set negative values to 0 (black) and values greater than 255 to 255 (white).
- Re-normalize the range by transforming each value according to the equation:

$$v_{new} = \underbrace{255}_{\text{Max value}} * \frac{v_{old} - min_{observed}}{max_{observed} - min_{observed}} \underbrace{+ 0}_{\text{Min value}}$$

Generic Formula

- Normalize in the range [MIN, MAX]

$$v_{new} = (MAX - MIN) * \frac{v_{old} - min_{observed}}{max_{observed} - min_{observed}} + MIN$$

Operations on images

Processing in the spatial domain can be expressed as:

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

f being the input image to the processing, ***g*** being the output image, and ***T*** being an operator on ***f*** defined in a neighborhood of ***(x,y)***.

Types of operations

The size of the neighborhood of (x,y) defines the character of the processing:

- punctual (the neighborhood coincides with the pixel itself);
- local (for example, a small square region centered on the pixel);
- global (the neighborhood coincides with the entire f).

Types of operations

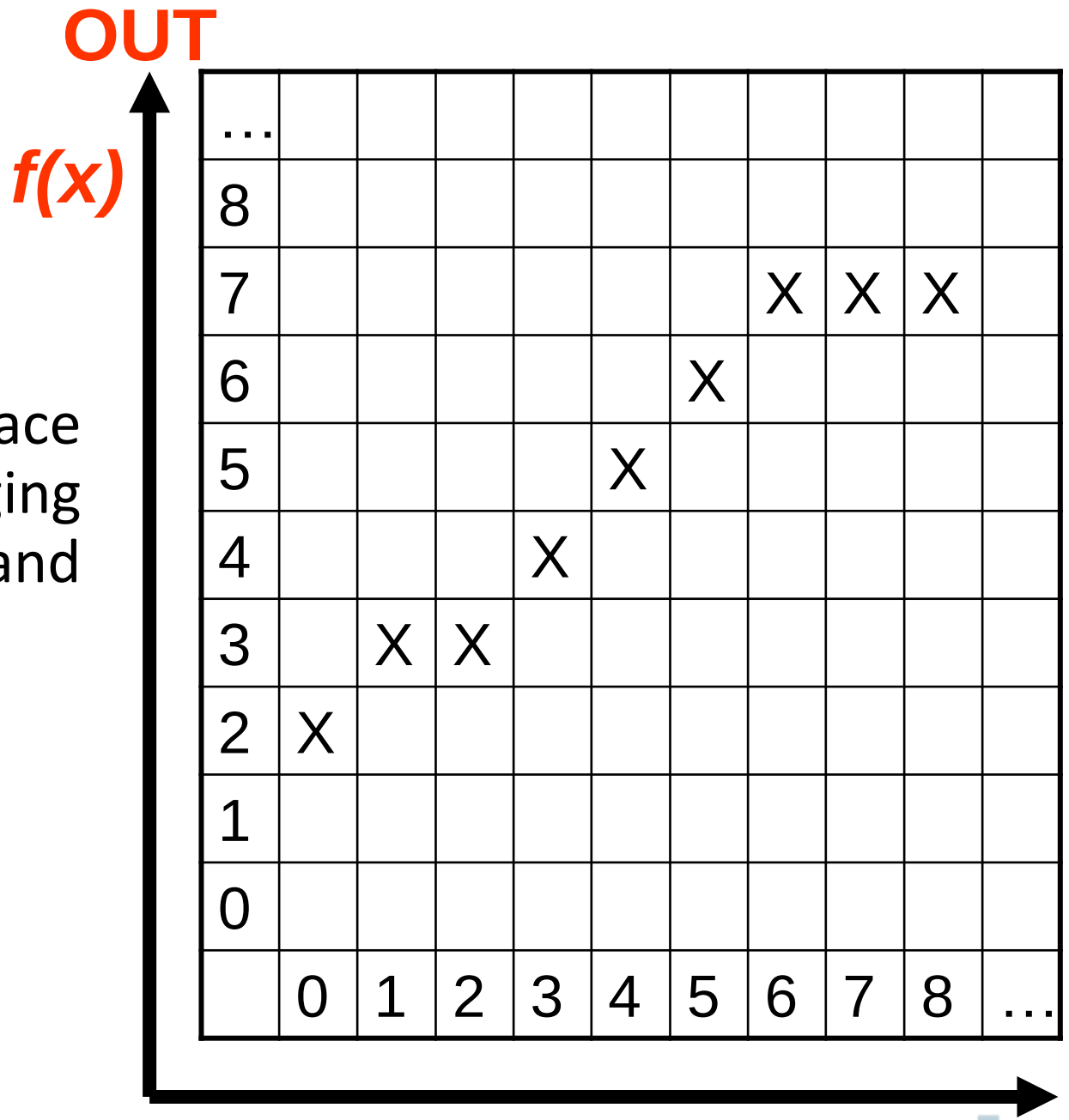
The size of the neighborhood of (x,y) defines the character of the processing:

- **punctual (the neighborhood coincides with the pixel itself);**
- local (for example, a small square region centered on the pixel);
- global (the neighborhood coincides with the entire f).

Typical point operations:

- addition or subtraction of a constant to all pixels (to compensate for underexposure or overexposure);
- gray scale inversion (negative);
- contrast expansion;
- changing (equalizing or specifying) the histogram;

This is universally the interface that all commercial imaging programs offer for viewing and managing point operations

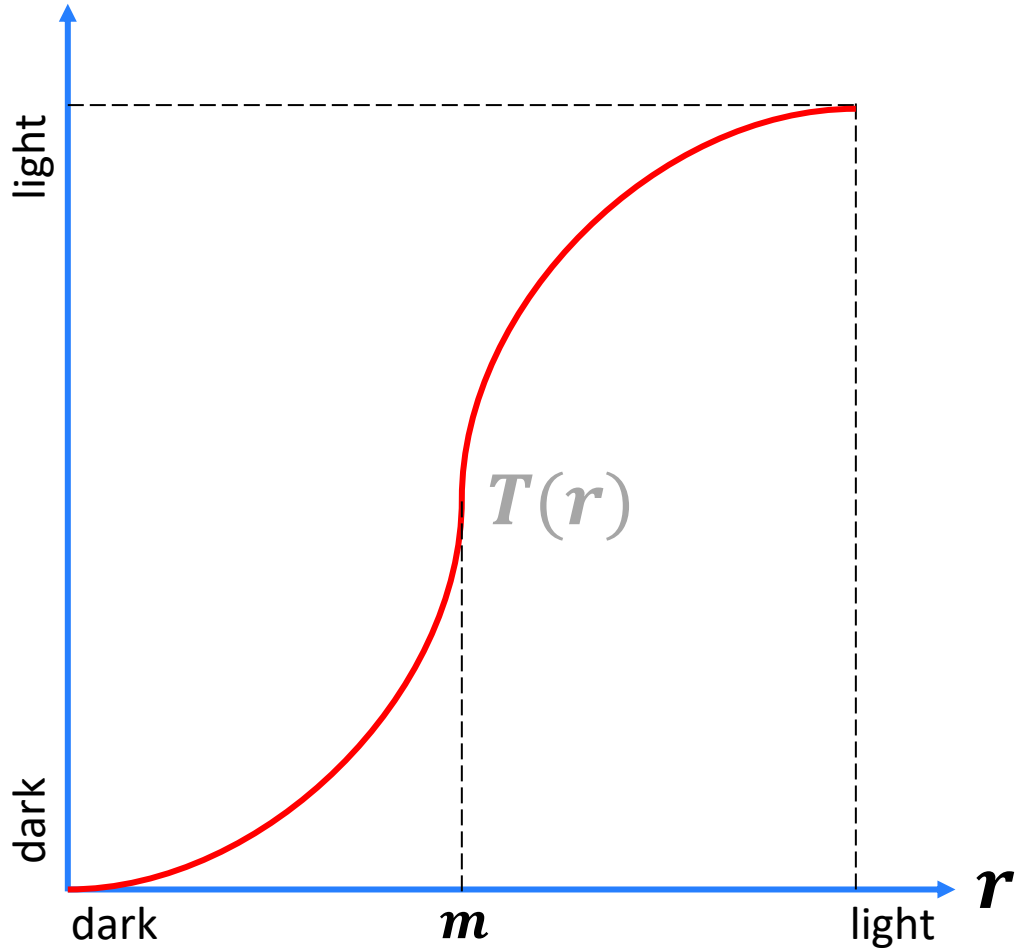


Look-up Tables (LUT)

$$s = T(r)$$

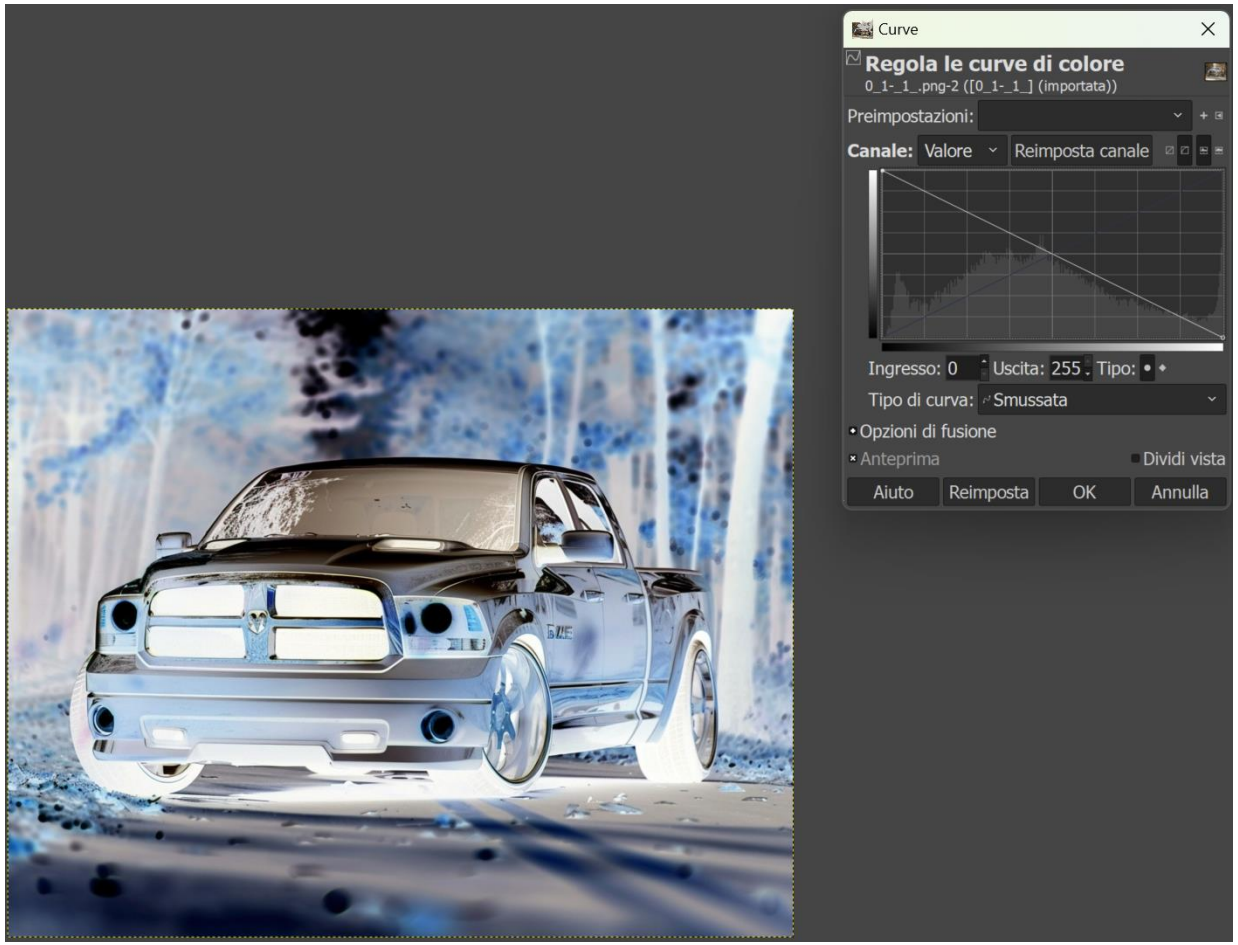
- This type of chart is called look-up tables (LUT).

A **Look-Up Table (LUT)** is a mathematical table used to transform input values into desired output values. In image processing, LUTs are commonly applied to adjust colors, brightness, contrast, or tone within an image. By mapping each pixel's original value to a new value based on predefined settings, LUTs can quickly apply complex color grading or correction without requiring extensive calculations.



Negative


Curve
Regola le curve di colore
0_1-_1_.png-2 ([0_1-_1_] (importata))
Preimpostazioni:
Canale: Valore
Reimposta canale
Ingresso: 255
Uscita: 255
Tipo:
Tipo di curva: Smussata
Opzioni di fusione
Anteprima
Dividi vista
Aiuto
Reimposta
OK
Annulla

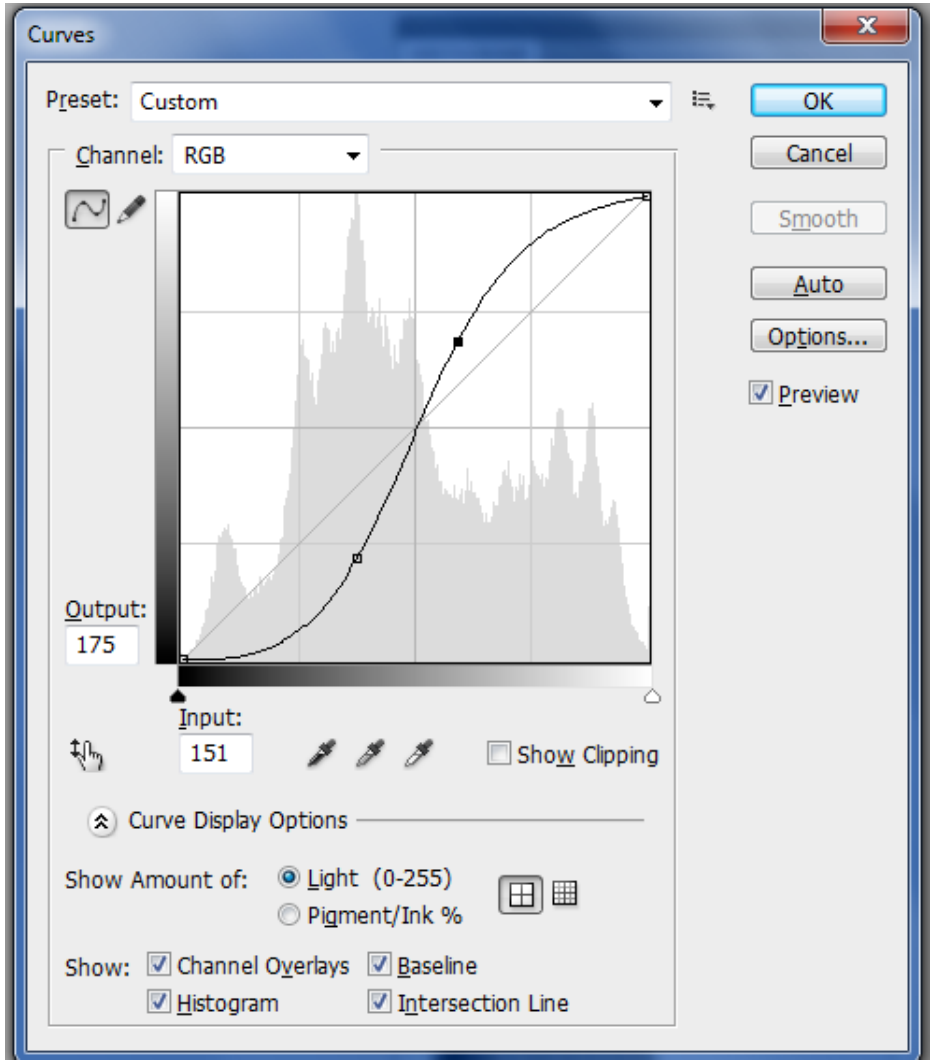

Curve
Regola le curve di colore
0_1-_1_.png-2 ([0_1-_1_] (importata))
Preimpostazioni:
Canale: Valore
Reimposta canale
Ingresso: 0
Uscita: 255
Tipo:
Tipo di curva: Smussata
Opzioni di fusione
Anteprima
Dividi vista
Aiuto
Reimposta
OK
Annulla

Variations in contrast

- Increasing the contrast, means making color differences more evident.
- This is achieved by going to change the value of one pixel to another that is darker or lighter.

Increased contrast

- How should the curve be changed?



Types of operations

The size of the neighborhood of (x,y) defines the character of the processing:

- punctual (the neighborhood coincides with the pixel itself);
- **local (for example, a small square region centered on the pixel);**
- global (the neighborhood coincides with the entire f).

Convolutions

Luca Guarnera, Ph.D.

Research Fellow

luca.guarnera@unict.it

University of Catania
Dipartimento di Economia e Impresa

Canonical base

- Given a vector of length N, this can be thought of as an element of an N-dimensional space.

234	204	34	16	44	134	12	11	56

- Then we can decompose it using the canonical basis of that space.

- What is the canonical basis?

		234	204	34	16	44	134	12	11	56	=
234	*	1	0	0	0	0	0	0	0	0	+
204	*	0	1	0	0	0	0	0	0	0	+
34	*	0	0	1	0	0	0	0	0	0	+
16	*	0	0	0	1	0	0	0	0	0	+
44	*	0	0	0	0	1	0	0	0	0	+
134	*	0	0	0	0	0	1	0	0	0	+
12	*	0	0	0	0	0	0	1	0	0	+
11	*	0	0	0	0	0	0	0	1	0	+
56	*	0	0	0	0	0	0	0	0	1	

Lo stesso ragionamento vale per le immagini

234	204	34
16	44	134
12	11	56

=

234 * | 1 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |

+ 204 * | 0 | 1 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |

+ 34 * | 0 | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |

16 * | 0 | 0 | 0 |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 0 | 0 |

+ 44 * | 0 | 0 | 0 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 0 |

+ 134 * | 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 0 |

12 * | 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |

+ 11 * | 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

+ 56 * | 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |

Why use the canonical base?

Decomposing an image into its canonical basis serves to represent the image **as a linear combination of fundamental elements (the "basis vectors")** that uniquely describe the image. This process has several purposes, including:

1. **Compression**: Representing the image with only a few significant coefficients reduces the amount of data needed to describe it.
2. **Feature Analysis**: Decomposition allows for separating components like details, textures, edges, or other relevant features of the image, facilitating analysis.
3. **Simplified Processing**: Manipulating or analyzing an image in terms of a canonical basis can make certain operations, such as filtering, noise reduction, or reconstruction, simpler and more efficient.
4. **Dimensionality Reduction**: Reducing data to principal components or other bases can be useful for machine learning and classification, improving performance and reducing noise.

In general, decomposition into a canonical base is useful for breaking down the image into its simplest and most comprehensible elements, facilitating the manipulation, analysis and enhancement of visual data.

Local operators

Local operators

- The output value of each pixel depends on a limited neighborhood of the corresponding input point.
- They are used to improve image quality or to extract information from the image.
- They can be thought of as image filtering.
- A filtering is achieved by convolving between the image and a matrix.

Linear operators

An operator

$$F : V \longrightarrow W$$

is said to be LINEAR if for each pair of vectors v1 and v2 in V and for each pair of scalars a, b we have that:

$$F(a\,v_1 + b\,v_2) = a\,F(v_1) + b\,F(v_2)$$

Consequence: if I know a basis of V and the behavior of operator F on each element of that basis, I can calculate the behavior of F on each element of V.

Is the function f(x,y)=(x/2,y/3) linear? YES

To be linear, the equality should occur.

$$a*f(x_1,y_1)+b*f(x_2,y_2)=f(ax_1+bx_2,ay_1+by_2)$$

The **first member** is

$$a*(x_1/2,y_1/3)+b*(x_2/2,y_2/3)=(ax_1/2+bx_2/2,ay_1/3+by_2/3)$$

The **second member** is.

$$((ax_1+bx_2)/2,(ay_1+by_2)/3)$$

Clearly, the two members are equal, so **the function is linear**.

Is the function f(x,y)=(255-x,255-y) linear? NO

To be linear, the equality should occur.

$$a*f(x_1,y_1)+b*f(x_2,y_2)=f(ax_1+bx_2,ay_1+by_2)$$

The **first member** is

$$a*(255-x_1,255-y_1)+b*(255-x_2,255-y_2)=$$

$$=(a*255-a*x_1,a*255-a*y_1) + (b*255-b*x_2,b*255-b*y_2)=$$

$$=(a*255-a*x_1+ b*255-b*x_2, a*255-a*y_1+ b*255-b*y_2)$$

The **second member** is.

$$(255 - (ax_1+bx_2), 255 - (ay_1+by_2)) = (255 - ax_1-bx_2), 255 - ay_1 - by_2))$$

Clearly the two members are different so **the function is NOT linear**.

Shift invariant?

- The examples in the previous slides have behavior that is not the same on all elements of the canonical base of R^N.
- In fact: the behavior varies from element to element depending on the position within the image.
- This is a NON-invariant operator for shifts!
- To describe these operators, we need to know its behavior on each "pulse" at each location in the images!
- Please note: these operators are not "bad" but only "difficult" to study....

Translation invariant operators

An operator is said to be translation invariant (shift invariant) when its behavior on impulsive images is always the same regardless of the position at which the pixel is located.

All point operators are translation invariant (even if they are nonlinear).

Negative operator

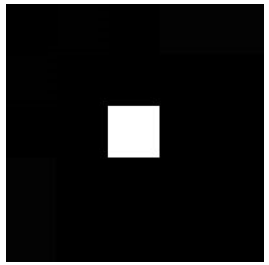
f(x,y)=(255-I(x,y))

- The negative operator is translation invariant.
- The operator takes the gray value at the point (x,y) and subtracts it from 255.
- This behavior is always the same for all the same layers, regardless of their position in the image.
- **WARNING. The function f(x,y)=(255-x,255-y) is not the negative function!**

A linear and shift invariant operator corresponds to a mask, but the reverse is also true: a mask corresponds to such an operator

EXAMPLE

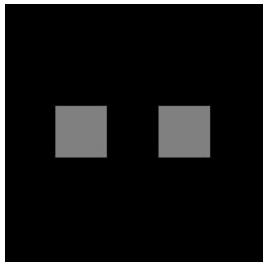
Consider the operation taking an impulse:



$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

transforms it into:

$$\begin{matrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{matrix}$$



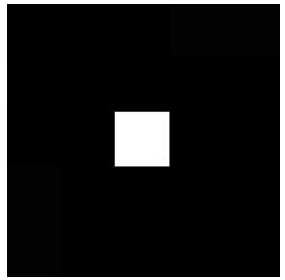
Such an "impulse response" completely defines a linear, invariant operator for translations F.

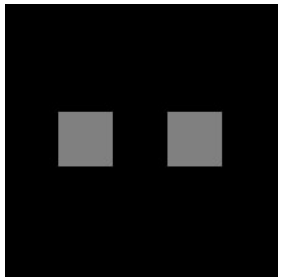
Often an operator on an image is called a "filter."

The matrix describing the impulse response is also called the kernel or mask of the operator.

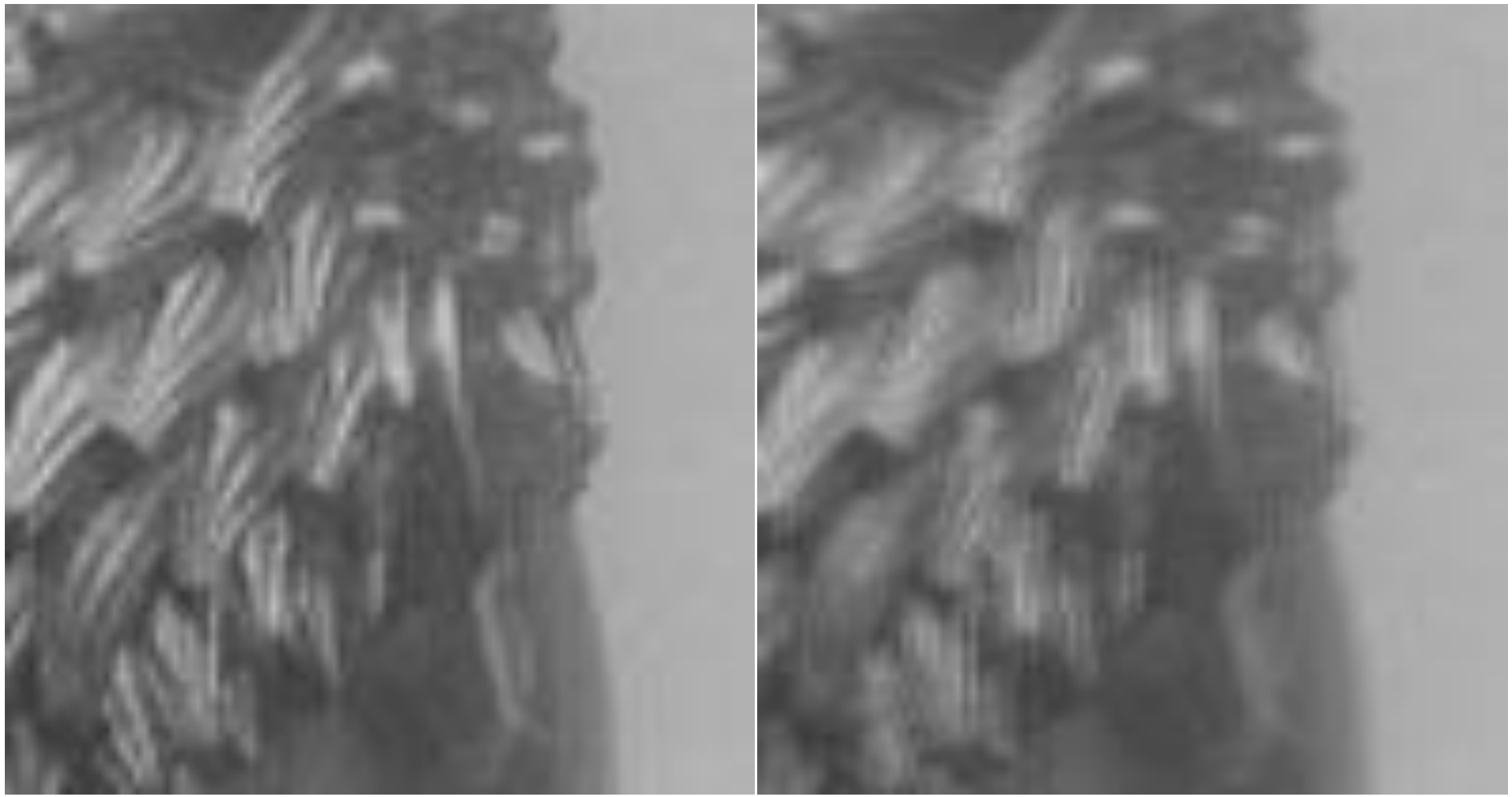
It is also called the convolution mask of F for reasons we will see shortly.

before





after



Finite or infinite kernels and complexity

- The size of the kernel can vary until it is infinite
- For practical reasons, however, only kernels with finite size are used.
- The size of the kernel affects the complexity of the filtering operation.
- This complexity obviously also depends on the number of pixels in an image.

Why "convolutional" filters?

- Linear and translation invariant filters are also called **convolutional filters**.
- We need to study the **convolution operation** to better understand how a filter can be computed.
- In addition, convolution is an extremely important phenomenon for all kinds of signal processing and for describing many physical events.

Convolution: properties

- To indicate the convolution operation, we use the notation

$$h = f \otimes g$$

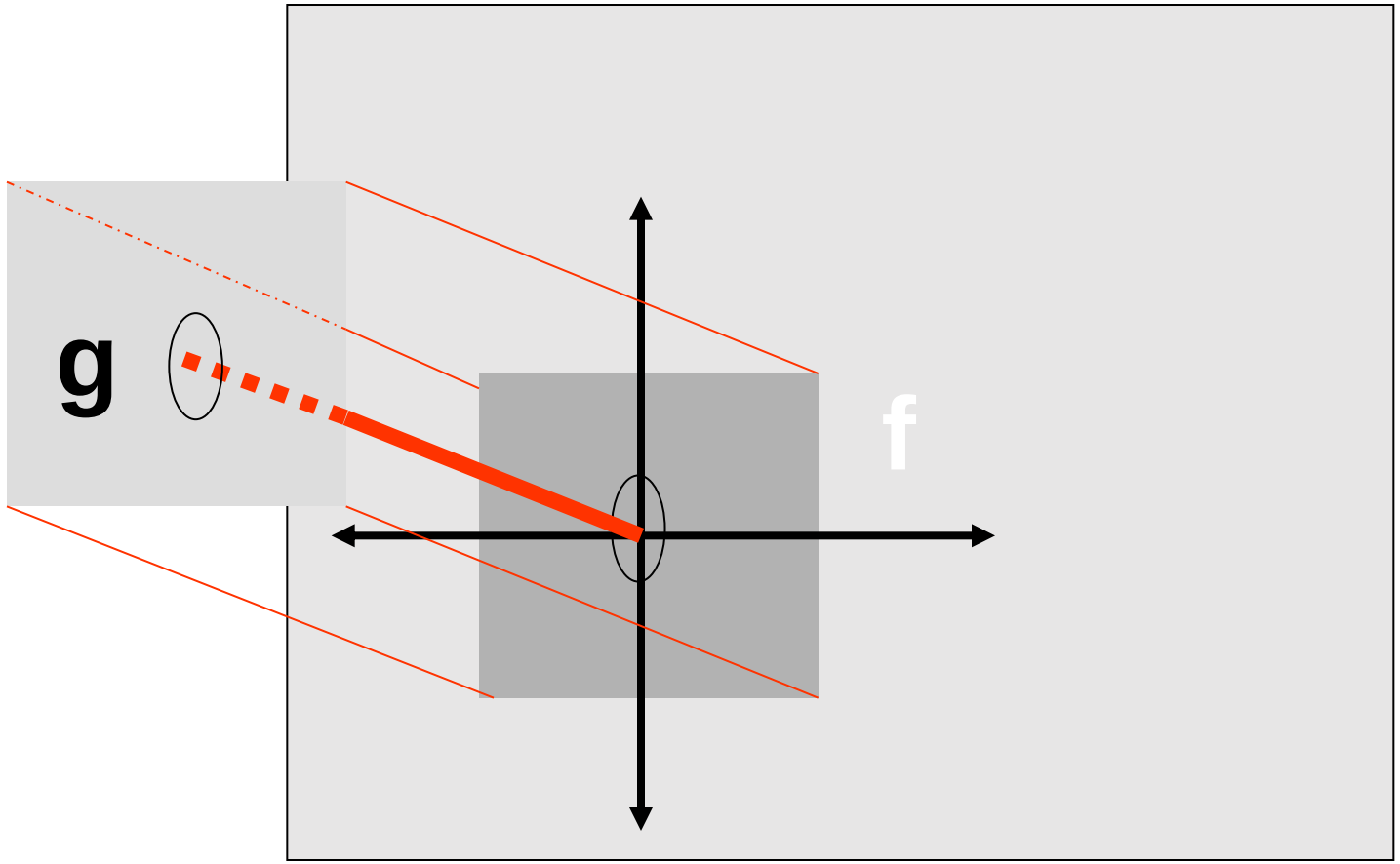
- The convolution is commutative

$$f \otimes g = g \otimes f$$

- The convolution is associative

$$(f \otimes g) \otimes h = f \otimes (g \otimes h)$$

Convolution Operation



In the finite case (1)

- If kernel f has size ***kxh***, the formula should be rewritten as follows.

$$h_{m,n} = \sum_{i=-\lfloor k/2 \rfloor}^{\lceil k/2 \rceil - 1} \sum_{j=-\lfloor h/2 \rfloor}^{\lceil h/2 \rceil - 1} \left(f_{i,j} * g_{m+i,n+j} \right)$$

	-1	*0*	*1*
-1	**a**	**b**	**c**
0	**d**	**e**	**f**
1	**g**	**h**	**i**

- If the kernel indexes are arranged so that the coordinate point (0,0) is at the central position.

Nel caso finito (2)

- If kernel f has size ***kxh***, the formula should be rewritten as follows.

$$h_{m,n} = \sum_{i=1,\, j=1}^{k,h} f_{i,j} * g_{m+(i-k+\lfloor k/2 \rfloor),\, n+(j-h+\lfloor h/2 \rfloor)}$$

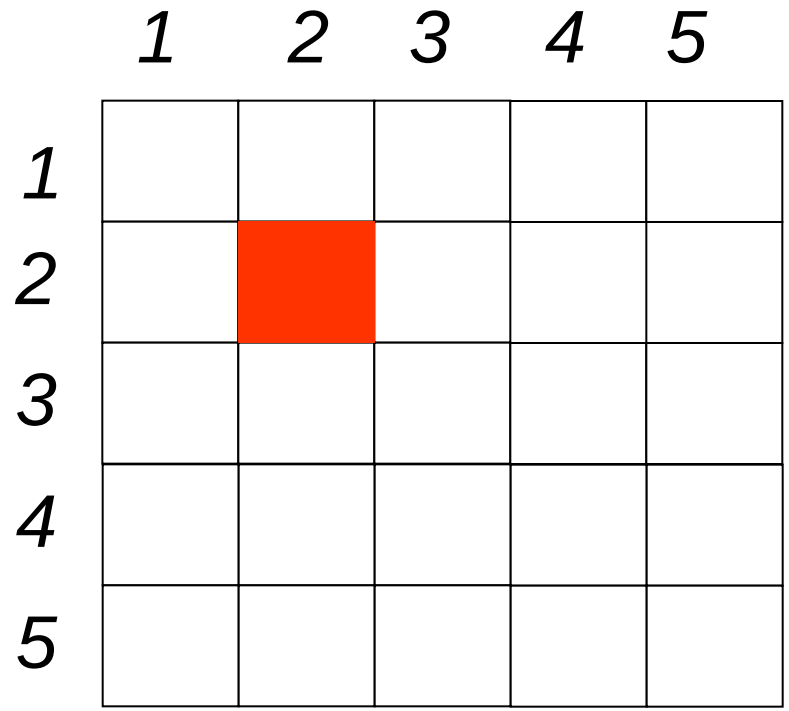
	1	2	3
1	**a**	**b**	**c**
2	**d**	**e**	**f**
3	**g**	**h**	**i**

- If the kernel indexes are arranged starting from 1 until h or k.

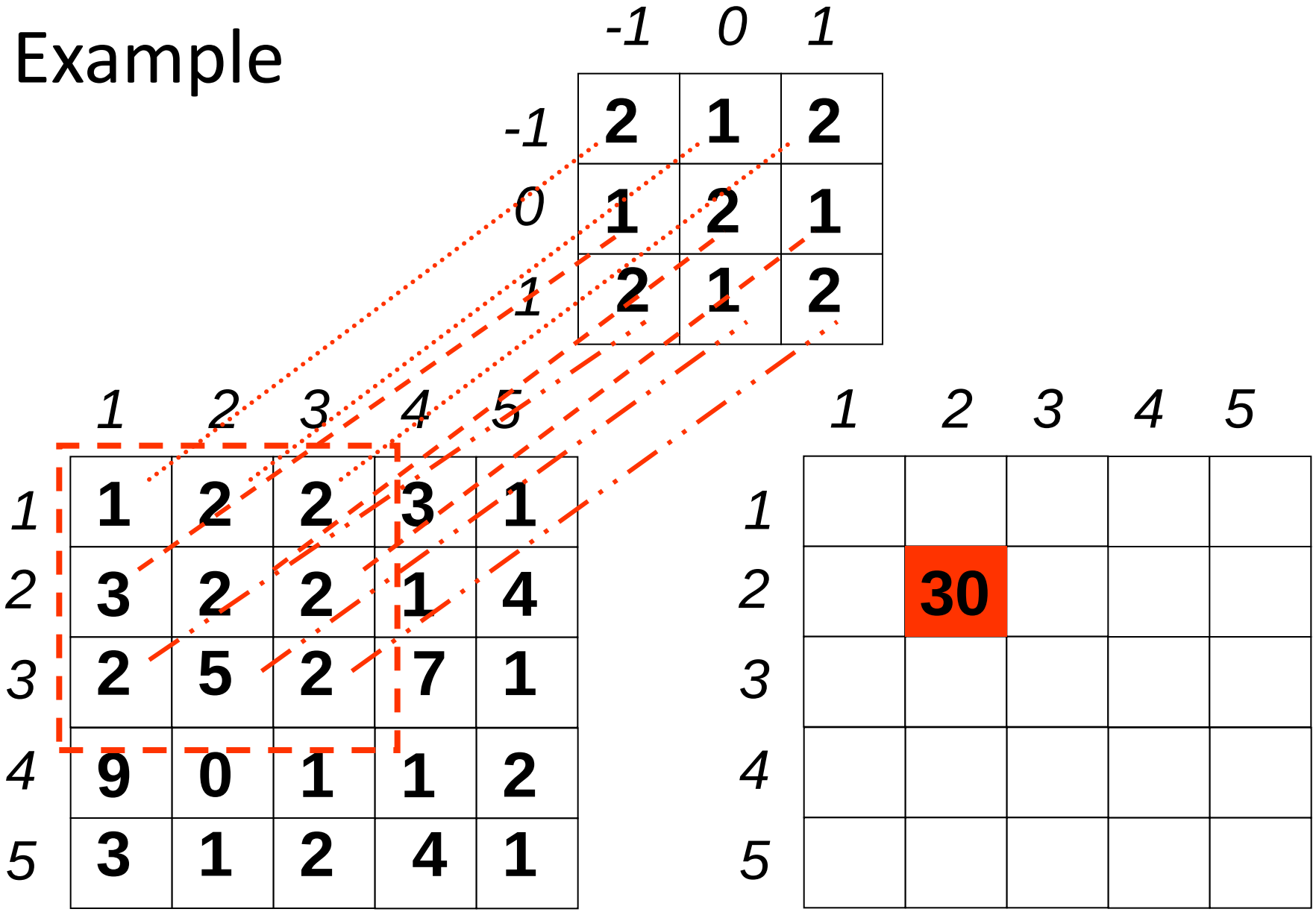
Example

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

	1	2	3	4	5
1	1	2	2	3	1
2	3	2	2	1	4
3	2	5	2	7	1
4	9	0	1	1	2
5	3	1	2	4	1



Example


-1 0 1
-1 2 1 2
0 1 2 1
1 2 1 2
1 2 3 4 5
1 1 2 2 3 1
2 3 2 2 1 4
3 2 5 2 7 1
4 9 0 1 1 2
5 3 1 2 4 1
1 2 3 4 5
1
2 30
3
4
5

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

	1	2	3	4	5
1	1	2	2	3	1
2	3	2	2	1	4
3	2	5	2	7	1
4	9	0	1	1	2
5	3	1	2	4	1

	1	2	3	4	5
1					
2		30	45	30	
3					
4					
5					

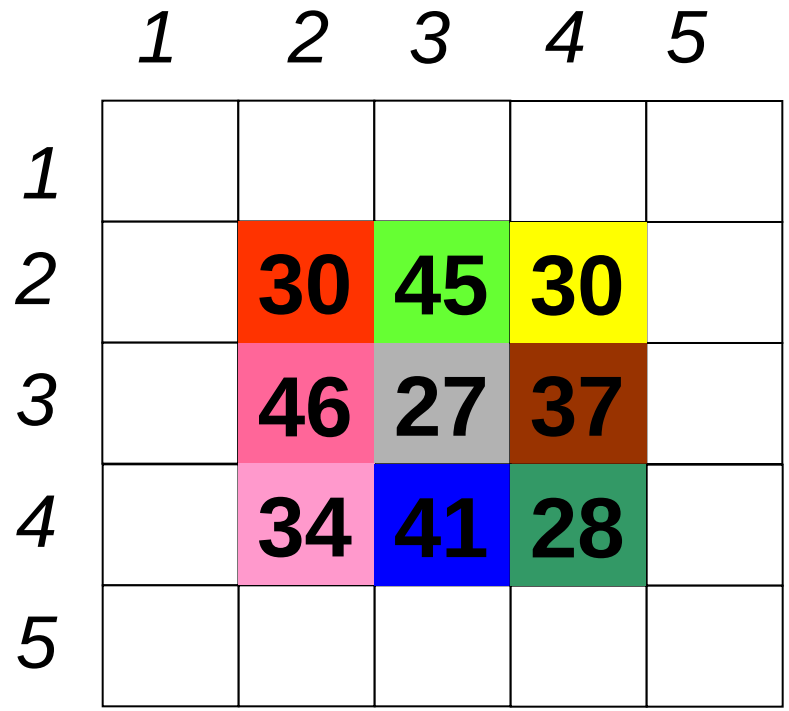
	-1	0	1
-1	2	1	2
0	1	2	1
1	2	1	2

	1	2	3	4	5
1	1	2	2	3	1
2	3	2	2	1	4
3	2	5	2	7	1
4	9	0	1	1	2
5	3	1	2	4	1

	1	2	3	4	5
1					
2		30	45	30	
3		46	27	37	
4					
5					

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

	1	2	3	4	5
1	1	2	2	3	1
2	3	2	2	1	4
3	2	5	2	7	1
4	9	0	1	1	2
5	3	1	2	4	1



Convolution and filtering

Applying a linear and shift invariant filter to an image is equivalent to computing the convolution of the filter kernel with the image.

In the implementation

One problem is with edges: how to do convolution and filtering at edges?

POSSIBLE SOLUTIONS:

a) Filter only the central areas of the image

b) Assume that all around the image there is 0

c) Assume a "toroidal" topology: when you "overflow to the right" you indent to the left, when you "overflow" to the bottom you indent to the top and vice versa;

d) Add a row at the beginning equal to the previous rows, a row at the end equal to the last row, a column at the beginning equal to the starting column, and a column at the end equal to the ending column.

(a) Filter only the central areas of the image.

Areas in gray will not be calculated

input

1	2	2	3	1
3	2	2	1	4
2	5	2	7	1
9	0	1	1	2
3	1	2	4	1

output

	30	45	30	
	46	27	37	
	34	41	28	

(b) Assume that all around the input image is the "0"

0	0	0	0	0	0	0
0	1	2	2	3	1	0
0	3	2	2	1	4	0
0	2	5	2	7	1	0
0	9	0	1	1	2	0
0	3	1	2	4	1	0
0	0	0	0	0	0	0

output

11	19	17	22	11
25	30	45	30	31
25	46	27	37	19
35	34	41	28	29
16	27	12	18	10

(c) Fill in the added rows and columns in a "toroidal" manner.

1	3	1	2	4	1	3
1	1	2	2	3	1	1
4	3	2	2	1	4	3
1	2	5	2	7	1	2
2	9	0	1	1	2	9
1	3	1	2	4	1	3
1	1	2	2	3	1	1

output

27	30	29	32	33
33	30	45	30	40
38	46	27	37	45
41	34	41	28	48
28	35	24	27	40

(d) Fill in the added rows and columns with the nearest values.

input

1	1	2	2	3	1	1
1	1	2	2	3	1	1
3	3	2	2	1	4	4
2	2	5	2	7	1	1
9	9	0	1	1	2	2
3	3	1	2	4	1	1
3	3	1	2	4	1	1

output

25	27	29	31	33
34	30	45	30	39
51	46	27	37	32
54	34	41	28	35
48	32	24	34	26

Python Lab!

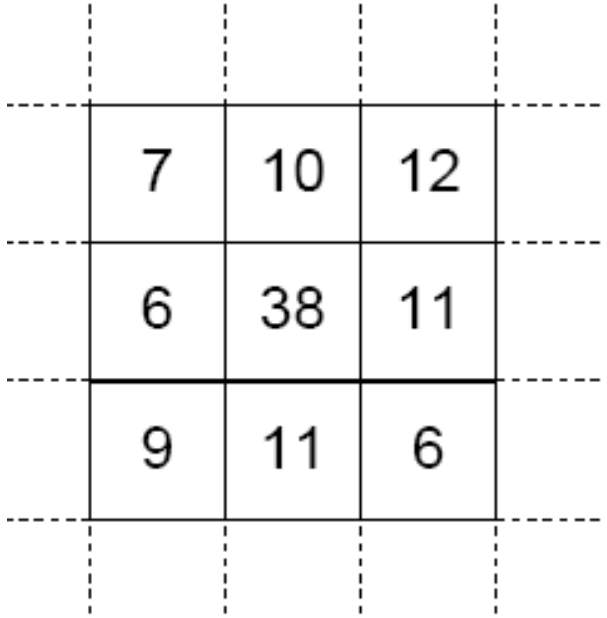
- Create a numpy randon array (3x3) with values from 0 to 10
- Given the input image I, apply the convolution formula
  - Show the first results without taking angles into account

- Define 3 functions to calculate the convolution in the corners:
  - Function 1: Consider padding 0 in I
  - Function 2: Considering the toroidal approach
  - Function 3: Adding rows and columns with the closest values.

Examples of local operators

Median

- It is a nonlinear filter that outputs the median value of the pixel's neighborhood.



6 6 7 9 10 11 11 12 38

valore mediano

Median filter

Minimum and maximum

In addition to the median filter, there are other statistical filters called order statistics.

- The minimum filter taken an **mxm neighborhood** of a pixel (with **m** generally **odd**), replace the value of the pixel with the minimum value of all values observed in that neighborhood.
- The maximum filter taken an **mxm neighborhood** of a pixel (with **m** generally **odd**), replace the value of the pixel with the maximum value of all the values observed in that neighborhood.

If you replace it with the minimum you get a darkening of the image (clear speckles are removed, for example);

If you replace it with the maximum, you get a brightening of the image (remove black speckles, for example).

Minimum filter

Maximum filter

N-box (or mean)

- They are defined by N x N kernels with each element equal to $1/N^2$.
- An odd N value is generally chosen.
- They have the effect of blurring images.
- The blurring is very strong horizontally and vertically but less so diagonally.
- EXAMPLES:

3-box

1/9 *

1	1	1
1	1	1
1	1	1

5-box

1/25 *

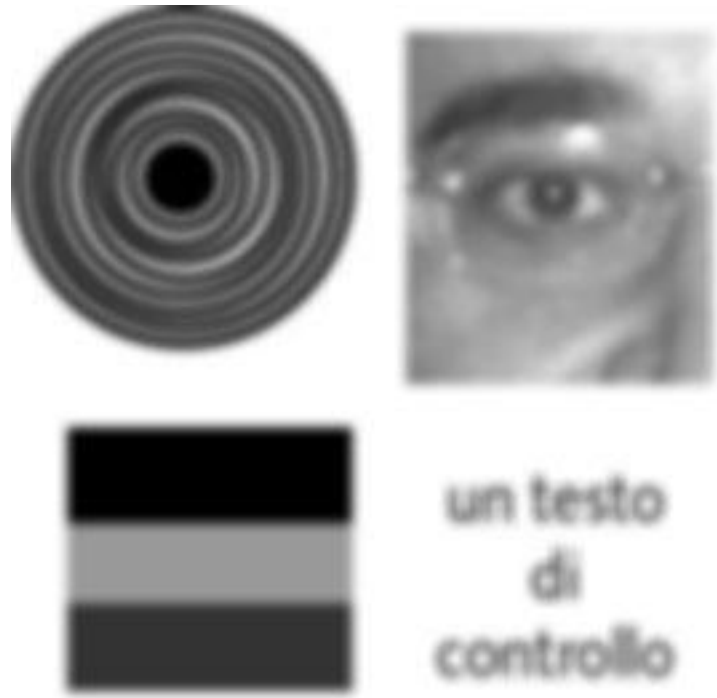
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

3-box filter

originale

3-box

5-box



7-box

N-binomial

These are smoothing filters with kernels derived from the binomial distribution. Since that distribution is a discrete approximation of the Gaussian distribution they are also called **Gaussian filters**.

They have the merit of smoothing equally in all directions.

They smooth less vigorously than n-boxes.

3-binomial

1/16 *

1	2	1
2	4	2
1	2	1

5-binomial

1/256 *

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

3-binomial filter

originale

un testo
di
controllo

3-binomiale

un testo
di
controllo

5-binomiale

un testo
di
controllo

7-binomiale

un testo
di
controllo

Noise cleaning e smoothing

- The filters just seen are also used to reduce noise in an image. In this case, the larger the kernel, the better the result will be even though there is a risk of increasing blurring.

- N-box and N-binomial filters are also used to blur the image (smoothing). In this case, the larger the kernel, the greater the blurring but the better the noise is reduced.

The noise

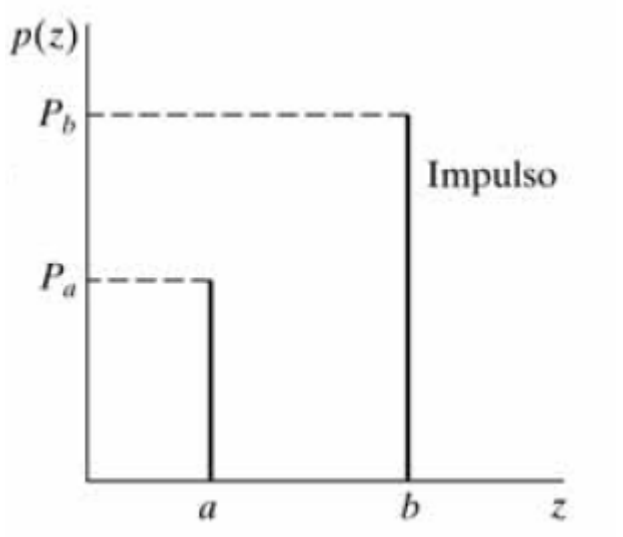
There are two main types of noise:

- Impulsive noise, also called "**salt and pepper**" noise. It is characterized by the fraction of the modified image (in %);
- **White Gaussian noise**. It is characterized by the mean and variance.

Impulsive noise (salt and pepper)

$$p(z) = \begin{cases} P_a & \text{per } z = a \\ P_b & \text{per } z = b \\ 0 & \text{altrimenti} \end{cases}$$

- If a and b are "saturated" value, that is, they are equal to the maximum and minimum values of the image (usually a=0 and b=255), we have salt-and-pepper noise.



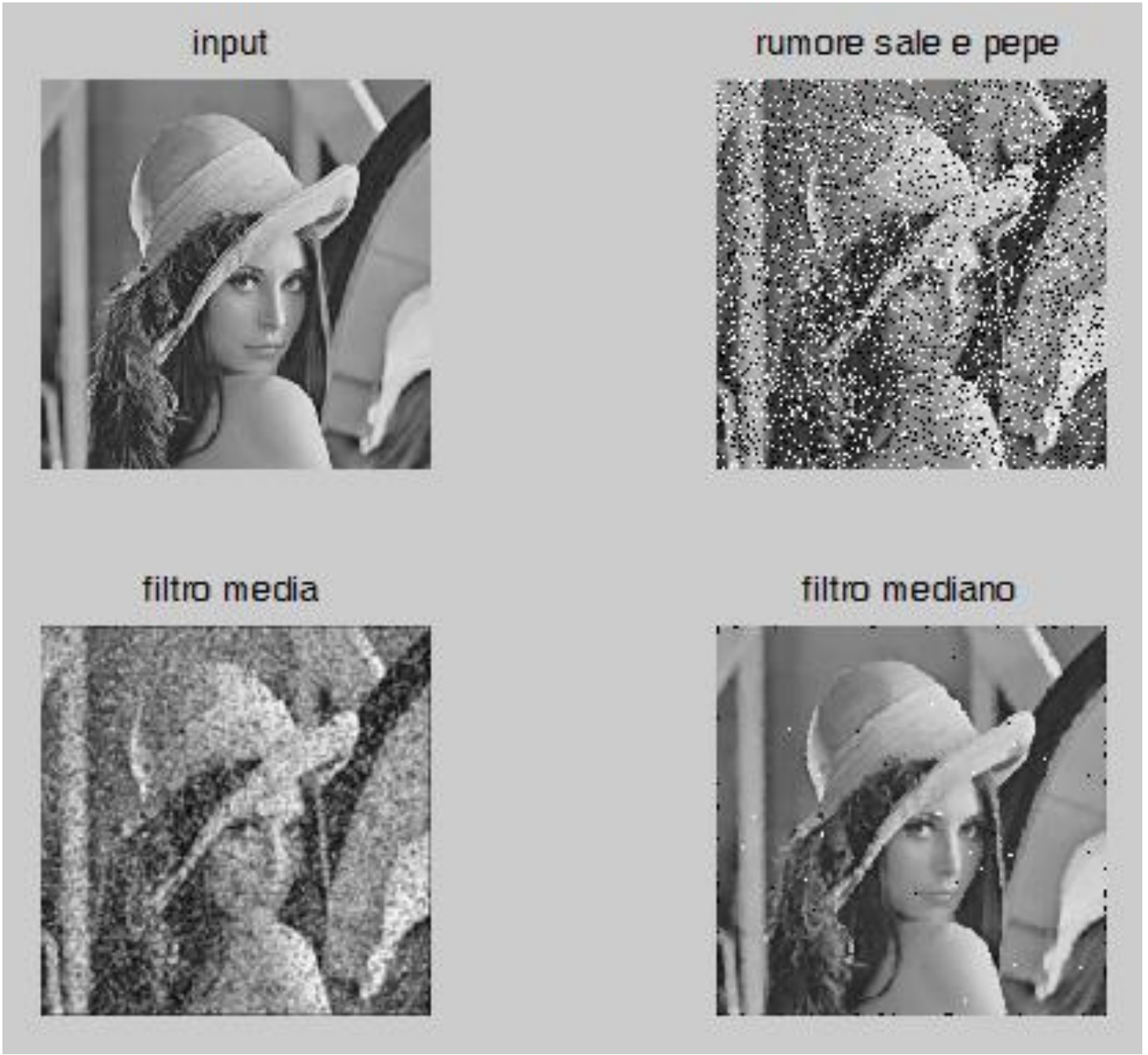
Examples of "salt and pepper" noise with 1% corrupted pixels

Examples of "salt and pepper" noise with 10% corrupted pixels

Examples of "salt and pepper" noise with 20% corrupted pixels

S&P Noise Removal

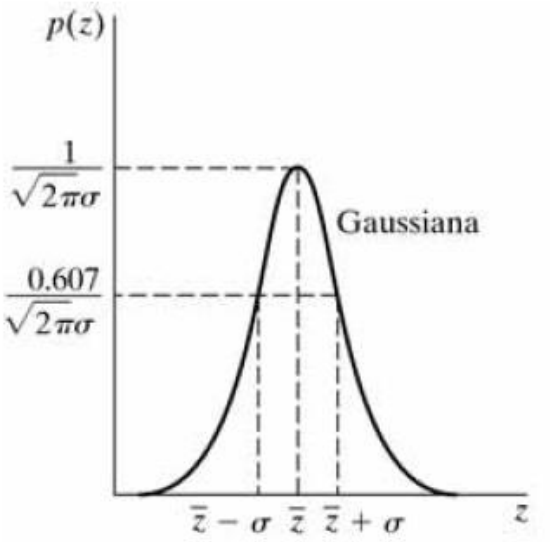
- Both the mean and median filters have 3x3 dimensions.
- For this type of noise, **the median filter** works best.



Gaussian noise

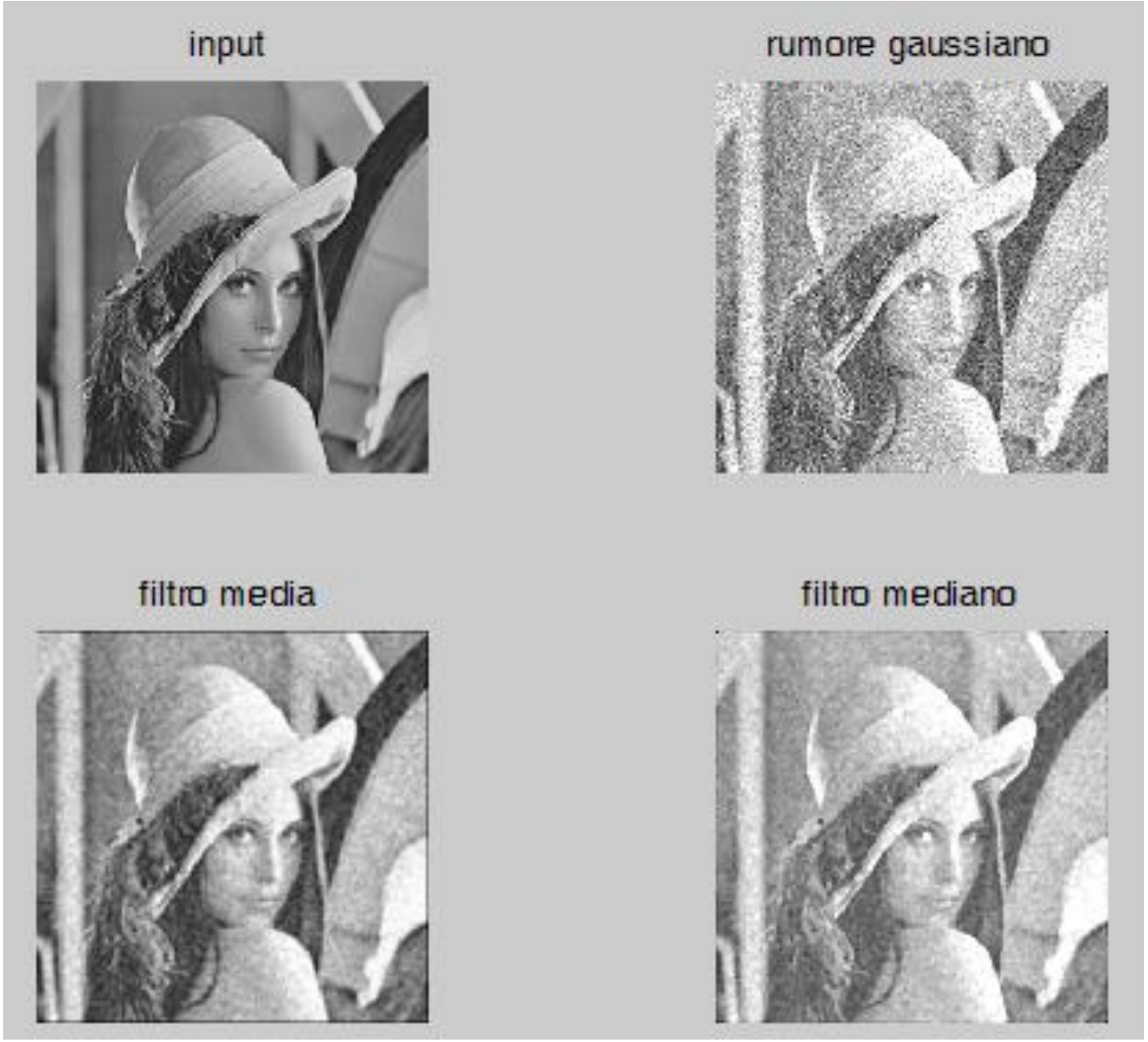
$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\bar{z})^2/2\sigma^2} \qquad (5.2\text{-}1)$$

dove z rappresenta l'intensità, $\bar{z}$ è il valore medio[1] (media) di z e σ è la sua deviazione standard. La deviazione standard al quadrato σ^2 è detta *varianza* di z. Il grafico di questa funzione è mostrato nella Figura 5.2a. Quando z viene descritta dall'Equazione (5.2-1), circa il 70% dei suoi valori ricade nell'intervallo $[(\bar{z} - \sigma), (\bar{z} + \sigma)]$ e circa il 95% ricade nell'intervallo $[(\bar{z} - 2\sigma), (\bar{z} + 2\sigma)]$.



Gaussian noise removal

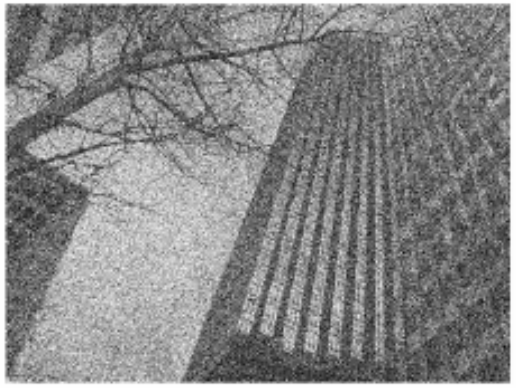
- Both the mean and median filters have 3x3 dimensions.
- For this type of noise, the median filter works best



Examples of Gaussian noise

Gaussiano

M=0,3 Var=0,02



M=0,1 Var=0,2

M=0,1 Var=0,02

Gaussian noise removal

Gaussiano

Filtro Mediano 5x5

Filtro 5-box

Kernel size

- If the noise is widespread, is a larger kernel better or is it better to iteratively apply the same kernel?

- Let's do a test on 20% salt-and-pepper noise.
- First we apply a 3x3 kernel twice.
- Then we apply a 5x5 kernel to the image with noise.
- In both cases the noise is removed, but the first approach blurs the final image less.

input

rumore sale e pepe

filtro mediano 5x5

filtro mediano 3x3 (2 volte)

S&P Noise Removal

Filtro mediano 3x3

S&P Noise Removal

Filtro mediano 5x5

Mean vs. Median

Why do median filters give better results than mean filters?

- The mean filter tends to create gray levels that did not exist before.
- The mean filter attenuates not only noise but also all high spatial frequencies indiscriminately resulting in blurred images.
- The median filter does not deteriorate the sides, but eliminates peaks with small base relative to the kernel.

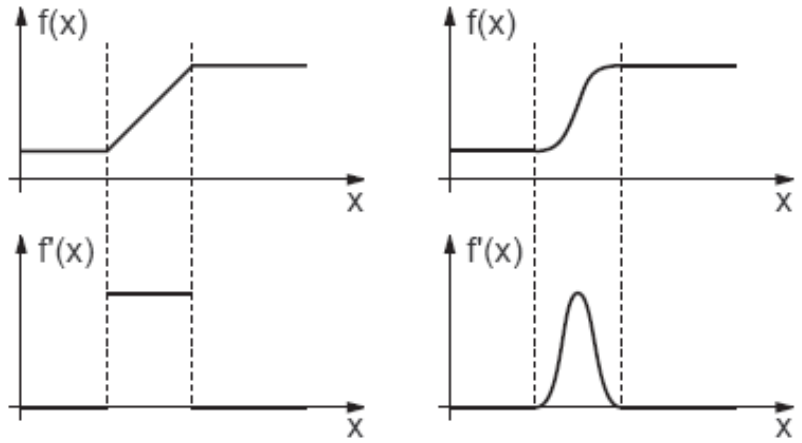
Edge extraction

Edge extraction

- Local operators help us extract edges from an image.
- **Edges are defined as local luminance discontinuities.**
- **Edge detectors provide images in which luminance variations are preserved and all other information is removed.**

Edge detectors based on the first derivative

- If I have a one-dimensional signal and calculate the first derivative, I find that the **edges are the correspondences of the maxima of the derivative**.



- Then the filters have to calculate the derivative in the x-direction that in the y-direction and then combine them together.

Notable kernels: horizontal sides

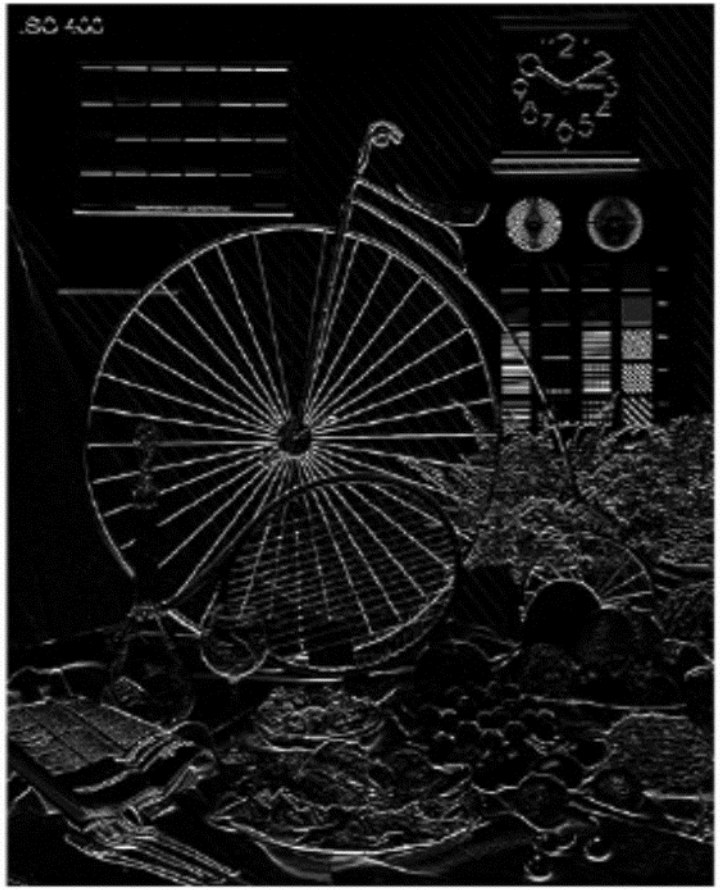
There are many of them; we present two:

$$Sobel_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \Pr ewitt_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel x

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

Filtro x-Sobel



Prewitt x

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

Filtro x-Prewitt

Notable kernels: vertical sides

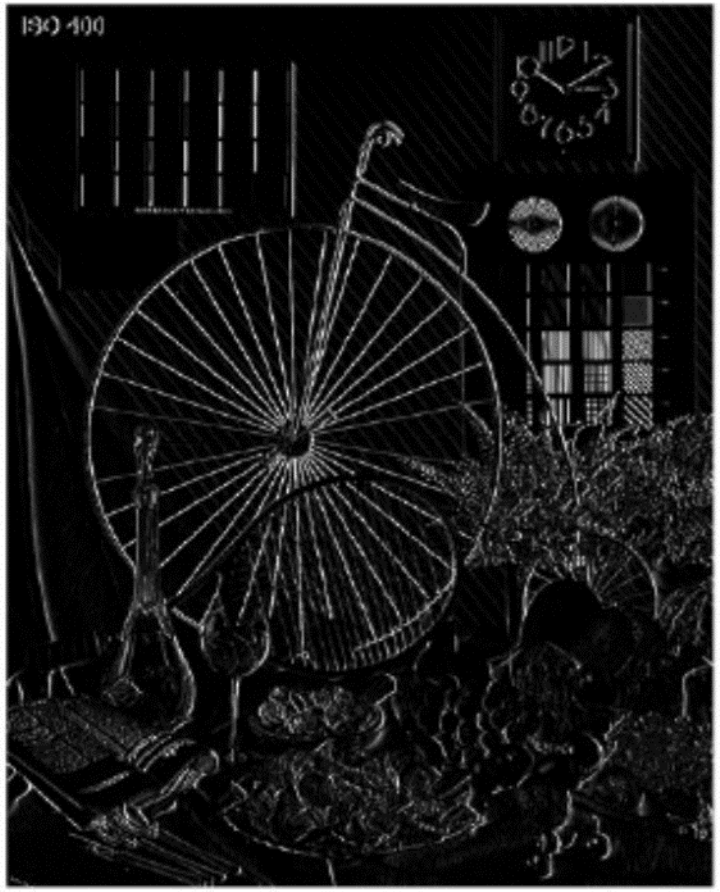
The situation is identical to the case of horizontal sides, the filters are just rotated 90 degrees.

$$Sobel_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad Prewitt_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel y

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

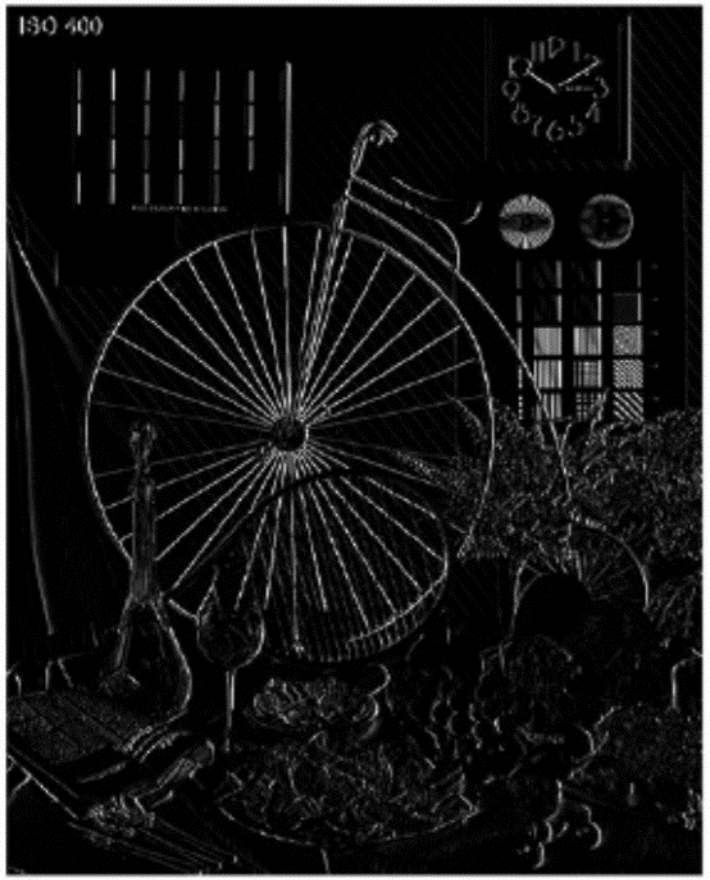
Filtro y-Sobel



Prewitt y

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

Filtro y-Prewitt



- Sobel x provides a matrix with horizontal sides (and horizontal components of the oblique sides) that have nonzero values.

- Sobel y provides a matrix with vertical sides (and vertical components of the oblique sides) that have nonzero values.

- The two matrices can be combined together by the following formula

$$\mathbf{sqrt(Sobel_x^2+Sobel_y^2)}$$

- the obtained matrix has nonzero values for the "sideways" pixels. If a suitable threshold is set, a binary matrix can be obtained that for each pixel tells us whether it is or is not sideways.

- Of course, the same considerations apply to Prewitt.

Sobel x

sobel y

modulo

modulo + soglia

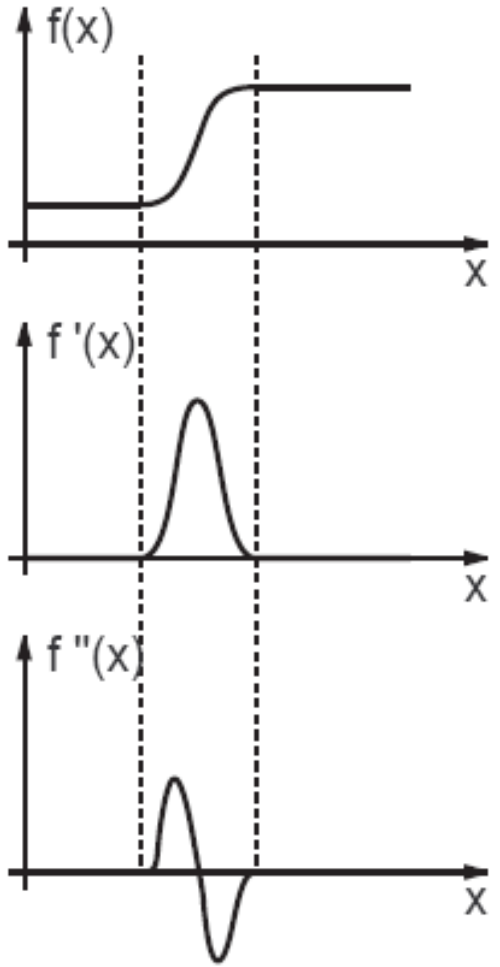
Better results...

- **They are obtained with more sophisticated (nonlinear) algorithms for calculating the magnitude of the gradient (sum of the square of the response of a horizontal edge finder and the square of the response of a vertical edge finder)**

- **They are obtained with more "intelligent" strategies (Canny's algorithm, fuzzy algorithms, backtracking techniques, etc.)**

Edge detectors based on the second derivative

- If I have a one-dimensional signal and calculate the second derivative, I find that at the side it passes through zero.



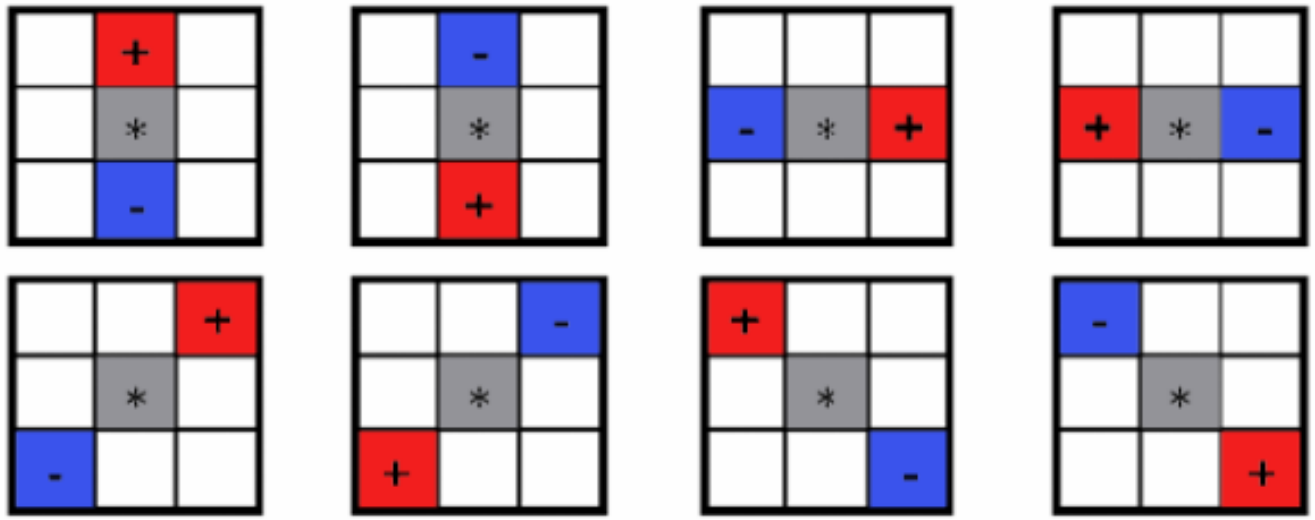
Notable Kernels: Laplacian

The most popular filter for calculating the second derivative is called the Laplacian, and it is defined by the mask:

$$Laplaciano = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

Zero-crossing

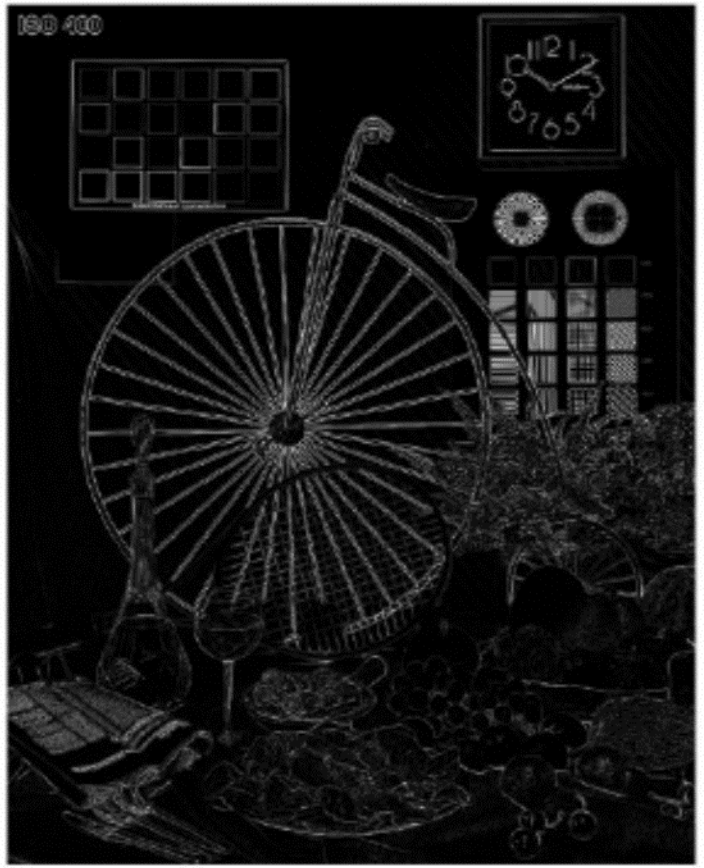
- After applying the Laplacian operator, it is necessary for the zero-crossing condition to occur. That is, it must always happen that with respect to the point in question there is in its neighborhood a positive value and a negative value.



Laplacian

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

Filtro Laplaciano



Laplacian

Confr