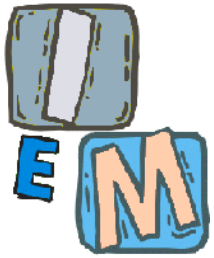


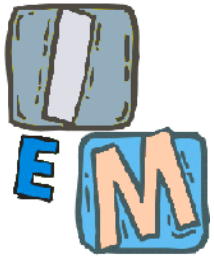

# Base canonica

Dato un vettore di lunghezza N, questo può essere pensato come un elemento di uno spazio N dimensionale.

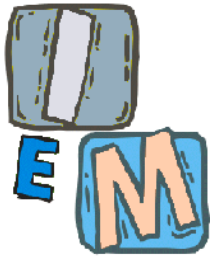
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 234 | 204 | 34 | 16 | 44 | 134 | 12 | 11 | 56 |

Quindi possiamo scomporlo usando la base canonica di tale spazio.

Qual è la base canonica?




| | 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 |

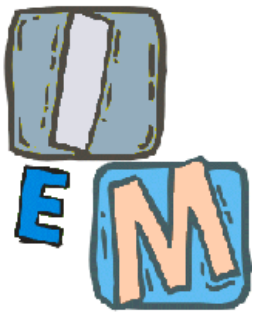
**+ 44 *** | 0 | 0 | 0 |
|---|---|---|
| 0 | 1 | 0 |
| 0 | 0 | 0 |

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

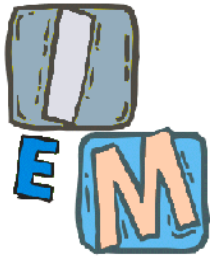
**12 *** | 0 | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |

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

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

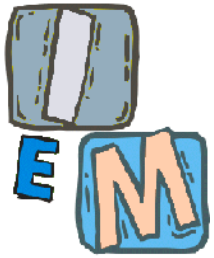



# Operatori locali




# Operazioni locali

- Il valore d'uscita di ogni pixel dipende da un limitato intorno del corrispondente punto in input.
- Sono usati per migliorare la qualità delle immagini o per estrarre delle informazioni dall'immagine.
- Si possono pensare come filtraggi dell'immagine.
- Un filtraggio è ottenuto facendo la convoluzione tra l'immagine ed una matrice.



# Operatori Lineari

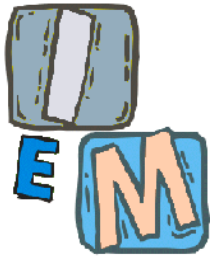
**Un operatore**

$$F : V \longrightarrow W$$

**si dice LINEARE se per ogni coppia di vettori $v_1$ e $v_2$ in V e per ogni coppia di scalari $a,b$ si ha che:**

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

***Conseguenza:*** ***se conosco una base di V ed il comportamento dell'operatore F su ogni elemento di tale base, posso calcolare il comportamento di F su ogni elemento di V.***




# La funzione f(x,y)=(x/2,y/3) è lineare? SI

Per essere lineare dovrebbe verificarsi l'uguaglianza

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

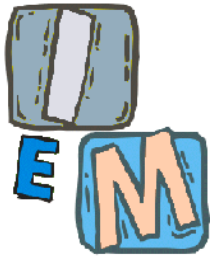
Il primo membro è

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

Il secondo membro è

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

Ovviamente i due membri sono uguali e quindi la funzione è lineare.




# La funzione f(x,y)=(255-x,255-y) è lineare? NO

Per essere lineare dovrebbe verificarsi l'uguaglianza

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

Il primo membro è

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

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

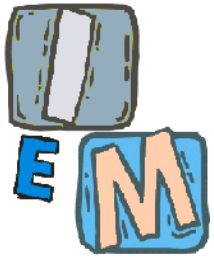
$(a*255-a*x_1+ b*255-b*x_2, a*255-a*y_1+ b*255-b*y_2)$

Il secondo membro è

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

$(255 – ax_1-bx_2), 255 – ay_1 - by_2))$

Ovviamente i due membri sono differenti quindi la funzione NON è lineare.




# Invariante per traslazione?

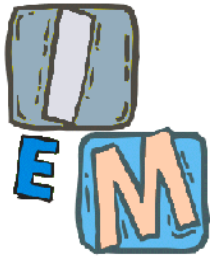
Gli esempi nelle slide precedenti hanno un comportamento che non è lo stesso su tutti gli elementi della base canonica di $R^N$.

*Infatti:* ***il comportamento varia da elemento ad elemento a seconda della posizione all'interno della immagine.***

Questo è un operatore **NON invariante per traslazioni!**

Per descrivere questi operatori, dobbiamo conoscere il suo comportamento su ciascun "impulso" in ciascuna locazione delle immagini!

Attenzione: questi operatori non sono "cattivi" ma solo "difficili" da studiare…



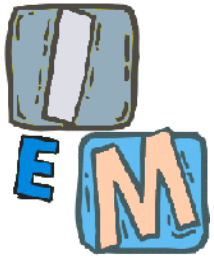

# Operatori invarianti per traslazione

Un operatore si dice **invariante per traslazione** (*shift invariant*) quando il suo comportamento sulle immagini impulsive è sempre il medesimo indipendentemente dalla posizione in cui si trova il pixel.

***Tutti gli operatori puntuali sono invarianti per traslazione (anche se non sono lineari).***

# Operatore negativo f(x,y)=(255-I(x,y))

- L'operatore negativo è invariante per traslazione.
- L'operatore prende il valore di grigio nel punto (x,y) e lo sottrae a 255.
- Questo comportamento è sempre lo stesso per tutti i medesimi livelli, indipendentemente dalla posizione che occupano nell'immagine.
- ATTENZIONE. La funzione f(x,y)=(255-x,255-y) non è la funzione negativo!



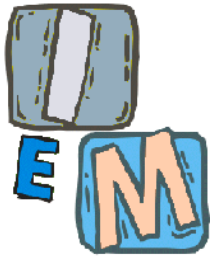

# Riassumendo:

Se F **è lineare** per descriverlo basta conoscere il comportamento su tutte le immagini impulsive

Se F **è shift invariant** si comporta allo stesso modo su tutti gli impulsi, indipendentemente dalla loro posizione

Se F è sia **lineare** che **shift invariant** per descriverlo basta conoscere come si comporta su **un solo** impulso.

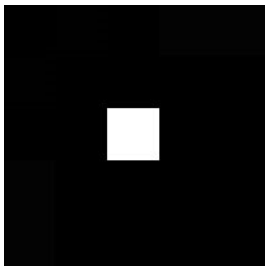
La "**risposta all'impulso**" o "**point spread function**" di F è la carta di indentità di tale operatore.




Ad un operatore lineare e shift invariante corrisponde una maschera ma vale anche il viceversa: ad una maschera corrisponde un simile operatore

**ESEMPIO**

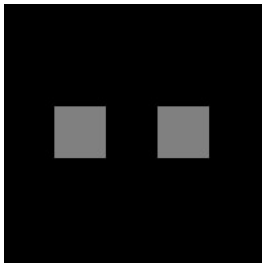
**Si consideri l'operazione che preso un impulso:**



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

**lo trasforma in:**

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

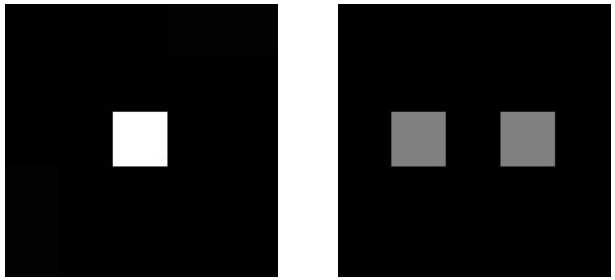


**Tale "*risposta all'impulso*" o PSF definisce completamente un operatore lineare e invariante per traslazioni F. Spesso un operatore su una immagine prende il nome di "*filtro*".**

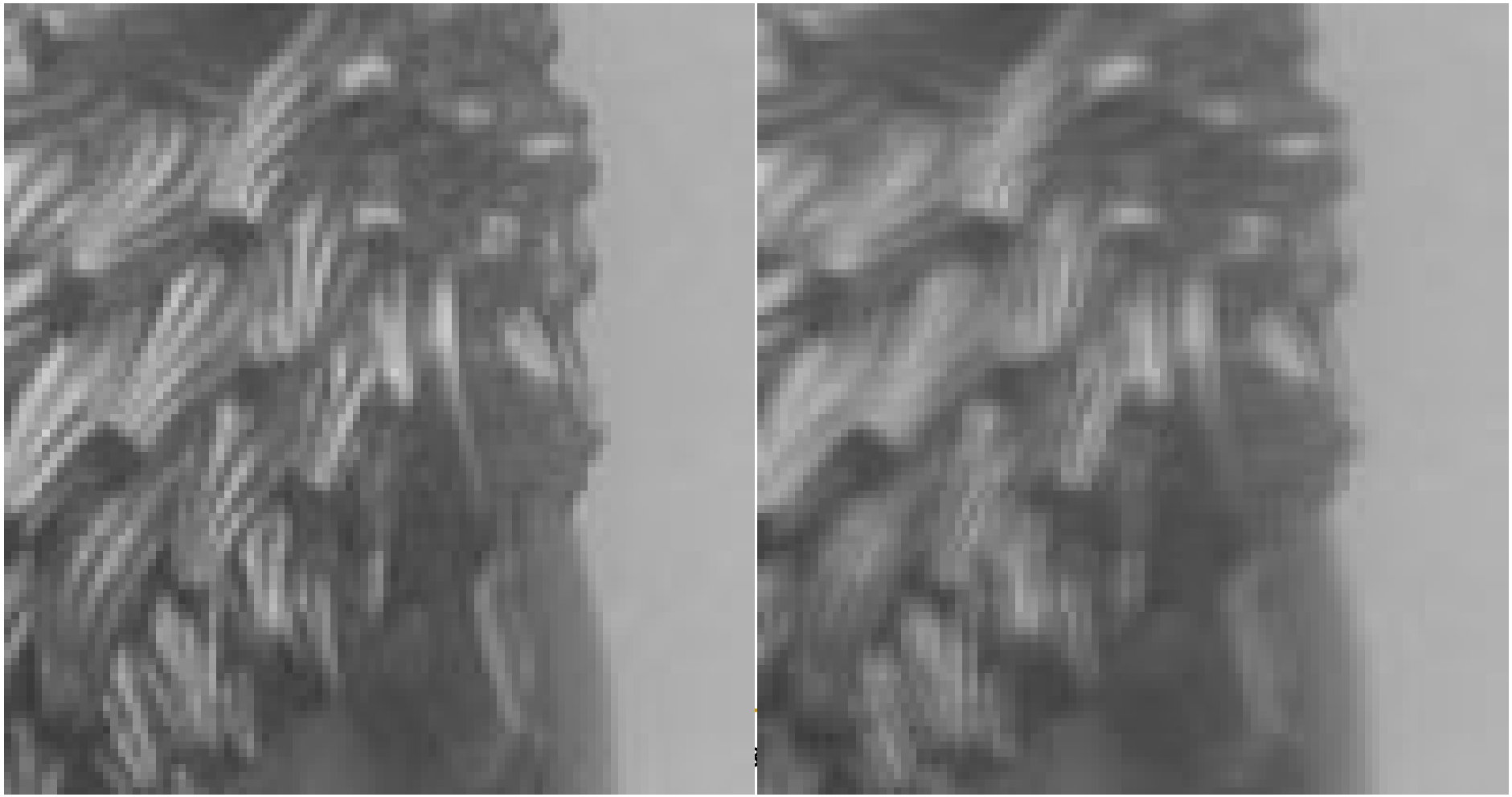
**La matrice che descrive la risposta all'impulso si chiama anche *kernel* o maschera dell'operatore.**

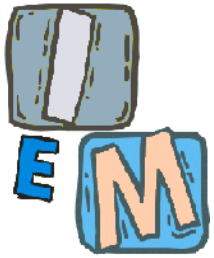
**Essa è detta anche *maschera di convoluzione* di F per ragioni che vedremo tra breve.**

prima



dopo





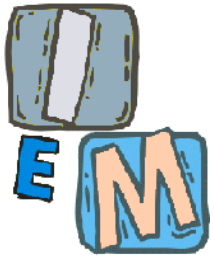

# Kernel finiti o infiniti e complessità

La grandezza del kernel può variare fino ad essere infinita

Per ragioni pratiche, però, si usano solo kernel con dimensioni finite.

Le dimensioni del kernel influenzano la complessità della operazione di filtraggio.

Tale complessità dipende ovviamente anche dal numero dei pixel di una immagine.

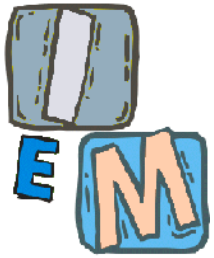



# Perché filtri "convolutivi"?

I filtri lineari e invarianti per traslazione vengono chiamati anche filtri convolutivi.

Dobbiamo studiare la **operazione di convoluzione** per capire meglio come un filtro può essere calcolato.

Inoltre la convoluzione è un fenomeno estremamente importante per ogni tipo di signal processing e per la descrizione di numerosi eventi fisici.



# Convoluzione: proprietà

- Per indicare l'operazione di convoluzione si usa la notazione

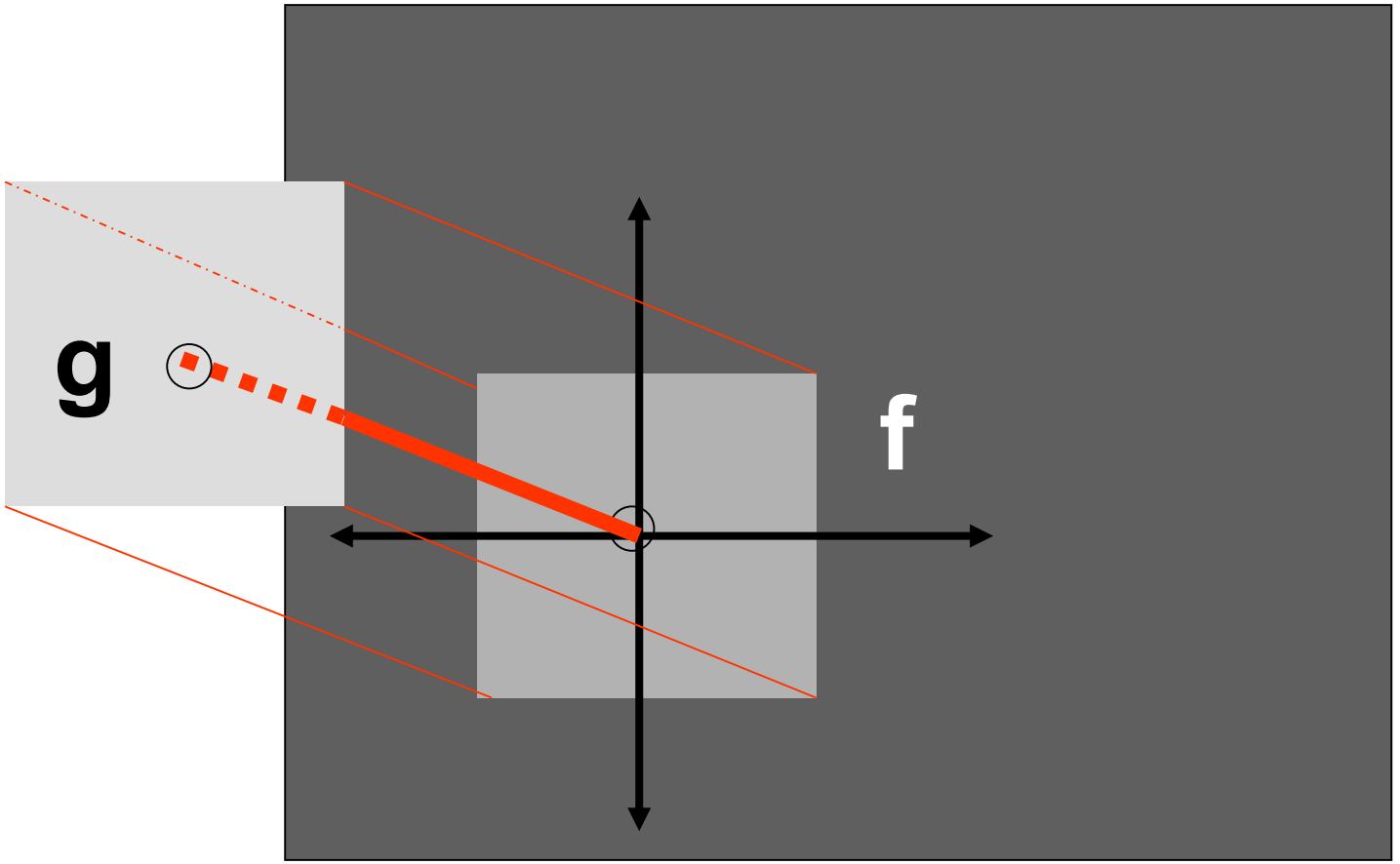
$$h = f \otimes g$$

- La convoluzione è commutativa

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

- La convoluzione è associativa

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


g
f

# Nel caso finito (1)

- Se il kernel *f* ha dimensioni *kxh* la formula va riscritta nella seguente maniera

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

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

- Se gli indici del kernel sono disposti in modo da avere il punto di coordinate (0,0) nella posizione centrale.

# Nel caso finito (2)

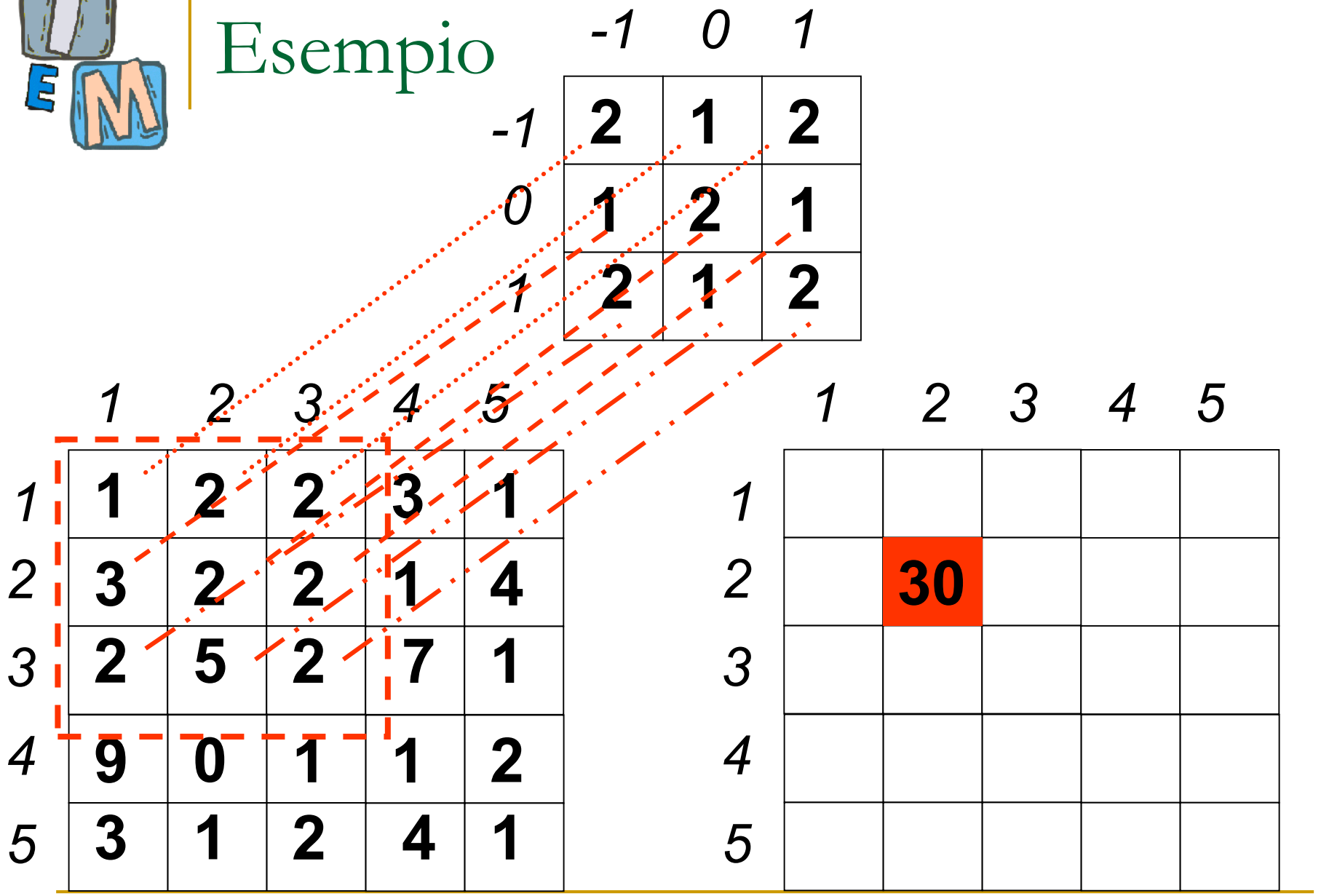
- Se il kernel *f* ha dimensioni *kxh* la formula va riscritta nella seguente maniera

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

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

- Se gli indici del kernel sono disposti partendo da 1 fino ad arrivare ad h o k.

# Esempio


-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

# Esempio

| | -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 | | | | | |

# Esempio

| | -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 |

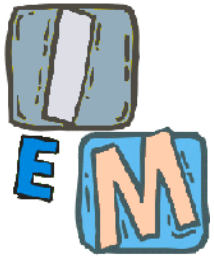
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | 30 | 45 | 30 | |
| 3 | | 46 | 27 | 37 | |
| 4 | | | | | |
| 5 | | | | | |

# Esempio

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

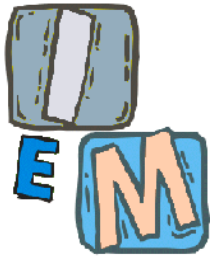
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 1 |
| 2 | 3 | 2 | 2 | 1 | 4 |
| 3 | 2 | 5 | 2 | 7 | 1 |
| 4 | 9 | 0 | 1 | 1 | 2 |
| 5 | 3 | 1 | 2 | 4 | 1 |

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | 30 | 45 | 30 | |
| 3 | | 46 | 27 | 37 | |
| 4 | | 34 | 41 | 28 | |
| 5 | | | | | |




# Convoluzione e filtraggio

Applicare un filtro lineare e shift invariante ad una immagine è equivalente a calcolare la convoluzione del kernel del filtro con l'immagine.




# Nell'implementazione

Un problema è quello dei bordi: come fare la convoluzione e il filtraggio ai bordi?

POSSIBILI SOLUZIONI:

a) Filtrare solo le zone centrali dell'immagine

b) Supporre che tutto intorno all'immagine ci sia 0

c) Assumere una topologia "toroidale": quando si "sfora a destra" si rientra a sinistra, quando si "sfora" in basso di rientra in alto e viceversa;

d) Aggiungere una riga all'inizio uguale alle riga precedente, una riga alla fine uguale all'ultima riga, una colonna all'inizio uguale alla colonna iniziale, e una colonna alla fine uguale alla colonna finale.

# a) Filtrare solo le zone centrali dell'immagine

Le aree in grigio non verranno calcolate

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) Supporre che tutto intorno all'immagine di input ci sia lo "0"

input

| | | | | | | |
|---|---|---|---|---|---|---|
| 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) Riempire le righe e le colonne aggiunte in maniera "toroidale"

input

| | | | | | | |
|---|---|---|---|---|---|---|
| 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 |

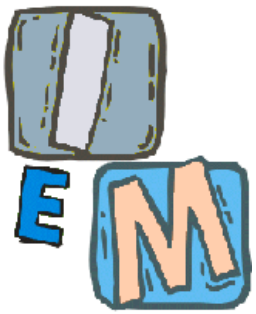
# d) Riempire le righe e le colonne aggiunte con i valori più vicini

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

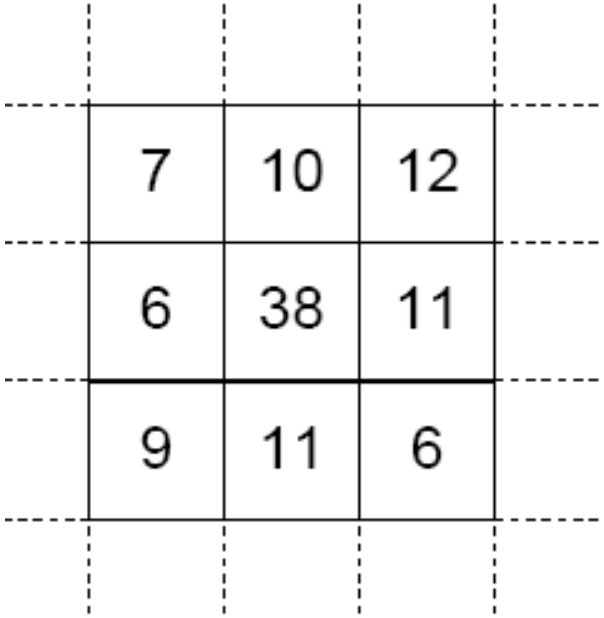
| | | | | |
|---|---|---|---|---|
| 25 | 27 | 29 | 31 | 33 |
| 34 | 30 | 45 | 30 | 39 |
| 51 | 46 | 27 | 37 | 32 |
| 54 | 34 | 41 | 28 | 35 |
| 48 | 32 | 24 | 34 | 26 |




# Esempi di operatori locali

# Mediano

- È un filtro non lineare che fornisce in uscita il valore mediano dell'intorno del pixel.




# Filtro mediano

# Minimo e massimo

Oltre al filtro mediano esistono altri filtri statistici detti **"order statistics".**

- Il filtro di minimo **preso un intorno *mxm*** di un pixel (*con m generalmente dispari*), **sostituiscono il valore del pixel con il valore minimo** di tutti i valori osservati in tale intorno.
- Il filtro di massimo **preso un intorno *mxm*** di un pixel (*con m generalmente dispari*), **sostituiscono il valore del pixel con il valore massimo** di tutti i valori osservati in tale intorno.

Se si sostituisce con il minimo si ottiene un incupimento dell'immagine (si eliminano per esempio macchioline chiare);

Se si sostituisce con il massimo si ottiene uno schiarimento dell'immagine (si eliminano per esempio punti neri).

# Filtro di minimo

# Filtro di massimo

# N-box (o di media)

Sono definiti da kernel N x N con ogni elemento pari a $1/N^2$.

Si sceglie generalmente un valore N dispari.

Hanno l'effetto di sfocare le immagini.

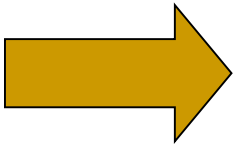
La sfocatura è molto forte in orizzontale e verticale ma meno in diagonale.

ESEMPI:

3-box

**1/9 ***

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |



5-box

**1/25 ***

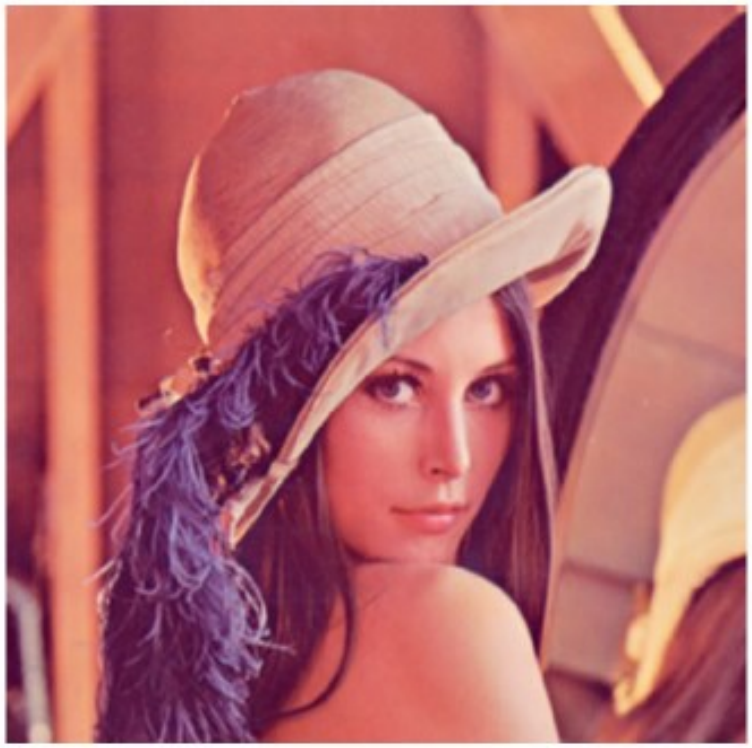
| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Filtro 3-box

1/9 *

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Filtro 3-Box



originale

un testo
di
controllo

3-box

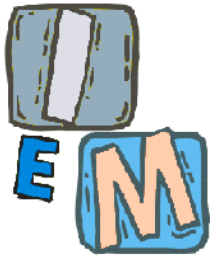
un testo
di
controllo

5-box

un testo
di
controllo

7-box

un testo
di
controllo



# N-binomiale

Sono filtri di smussamento con kernel derivati dalla distribuzione binomiale. Poiché tale distribuzione è una approssimazione discreta della distribuzione gaussiana sono anche detti filtri gaussiani.

Hanno il pregio di smussare egualmente in tutte le direzioni.

Smussano meno vigorosamente degli n-box.

**3-binomiale**

**1/16 ***

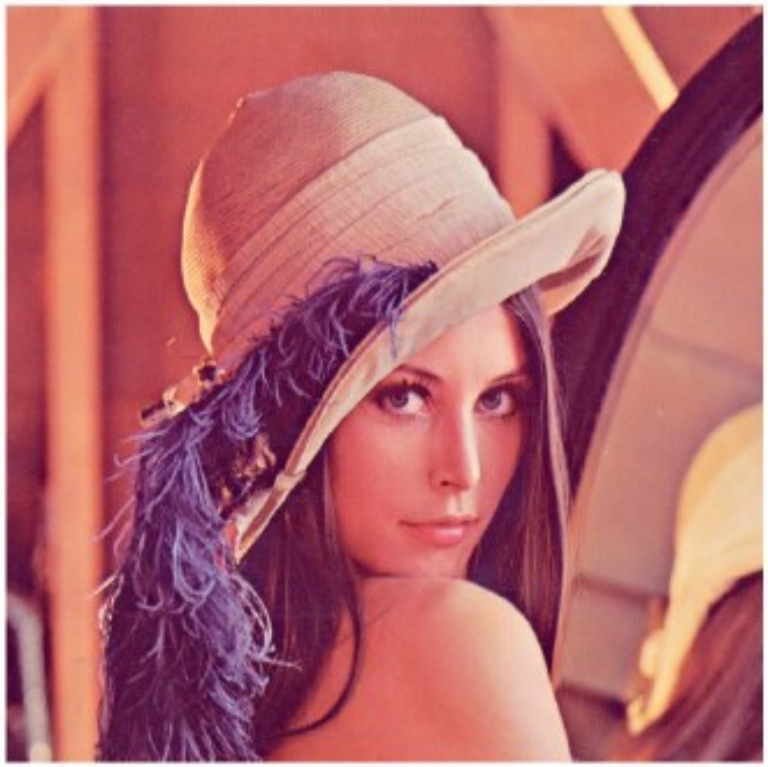
| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

**5-binomiale**

**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 |

# Filtro 3-binomiale



1/16 *

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Filtro 3-Binomiale

un testo
di
controllo

originale







un testo
di
controllo

3-binomiale







un testo
di
controllo

5-binomiale

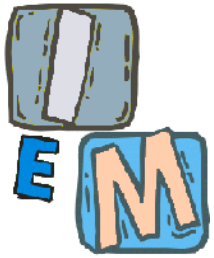






un testo
di
controllo

7-binomiale

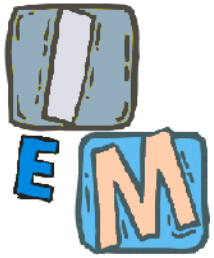



# Conservazione dell'energia

Con abuso di linguaggio, derivato dalla Fisica, si dice che un filtro "conserva l'energia" se la somma dei suoi pesi fa 1.

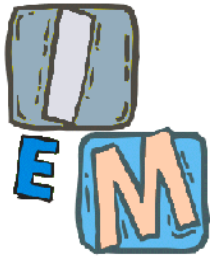
Tutti i filtri di smoothing visti prima sono "energy preserving": la somma dei valori totali della luminanza nella immagine non cambia.

Esistono anche filtri "non energy preserving"



# Noise cleaning e smoothing

- I filtri appena visti servono anche a ridurre il rumore in una immagine. In questo caso, più è grande il kernel e migliore sarà il risultato anche se si rischia di aumentare la sfocatura.
- I filtri N-box e N-binomiali sono anche usati per sfocare l'immagine (smoothing). In questo caso, più è grande il kernel e maggiore sarà la sfocatura ma si riduce meglio il rumore.




# Il rumore

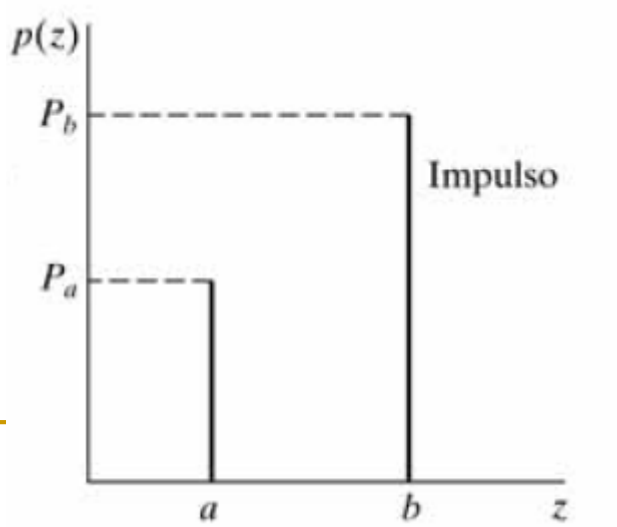
Ci sono due tipi principali di rumore:

- Rumore impulsivo, detto anche “sale e pepe”. Viene caratterizzato dalla frazione dell’immagine modificata (in %);
- Rumore gaussiano bianco. Viene caratterizzato dalla media e dalla varianza.

# Rumore impulsivo (sale e pepe)

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

- Se a e b sono valore «saturi» cioè sono uguali ai valori di massimo e di minimo dell'immagine (solitamente $P_a$=0 e $P_b$=255), abbiamo il rumore sale e pepe.



# Esempi di rumore «sale e pepe» con 1% di pixel danneggiati

# Esempi di rumore «sale e pepe» con 10% di pixel danneggiati

# Esempi di rumore «sale e pepe» con 20% di pixel danneggiati

# Rimozione del rumore S&P

- Sia il filtro media che il mediano hanno dimensioni 3x3
- Per questo tipo di rumore il filtro mediano funziona meglio.

input

rumore sale e pepe

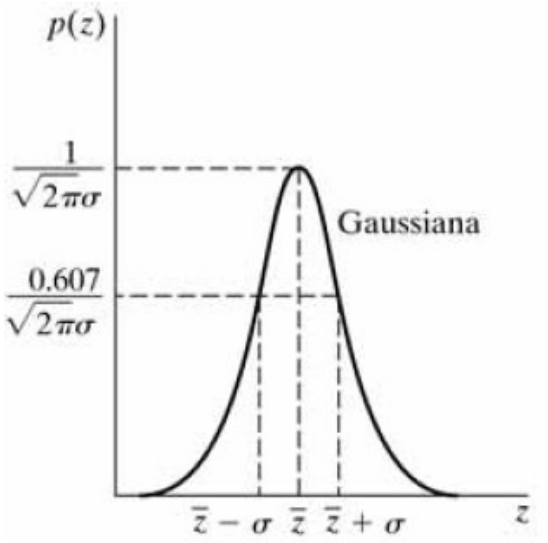
filtro media

filtro mediano

# Rumore gaussiano

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

dove $z$ rappresenta l'intensità, $\bar{z}$ è il valore medio[1] (media) di $z$ e $\sigma$ è la sua deviazione standard. La deviazione standard al quadrato $\sigma^2$ 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 $[(\bar{z} - 2\sigma), (\bar{z} + 2\sigma)]$.



# Rimozione del rumore gaussiano

- Sia il filtro media che il mediano hanno dimensioni 3x3
- Per questo tipo di rumore il filtro media funziona meglio.

input

rumore gaussiano

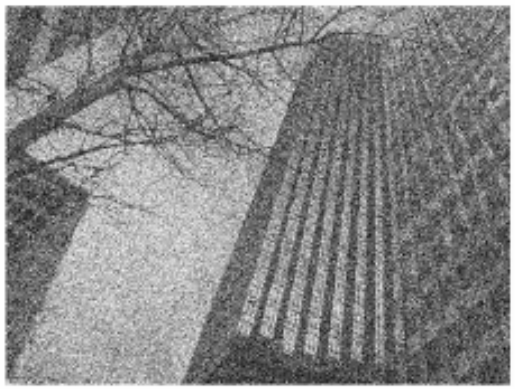
filtro media

filtro mediano

# Esempi di rumore Gaussiano

Gaussiano

M=0,3 Var=0,02



M=0,1 Var=0,2

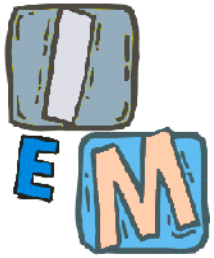
M=0,1 Var=0,02

# Rimozione del rumore Gaussiano

Gaussiano

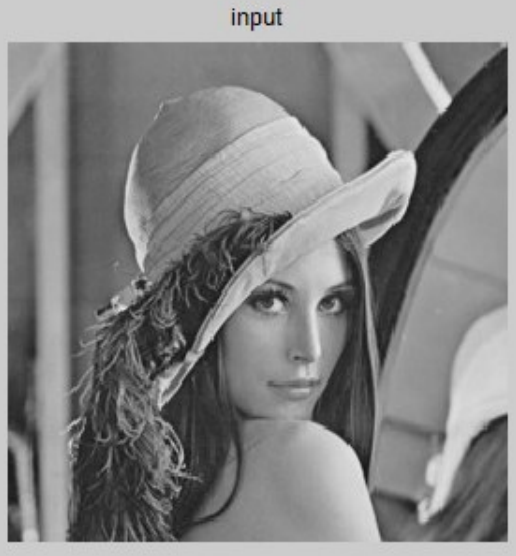
Filtro Mediano 5x5

Filtro 5-box




# Dimensioni del kernel

- Se il rumore è molto diffuso, è meglio un kernel più grande oppure è meglio applicare iterativamente lo stesso kernel?
- Facciamo una prova sul rumore sale e pepe al 20%.
- Prima applichiamo per due volte un kernel 3x3.
- poi applichiamo all'immagine con rumore un kernel 5x5.
- In entrambi i casi si elimina il rumore, ma il primo approccio sfoca di meno l'immagine finale.


input


rumore sale e pepe


filtro mediano 5x5

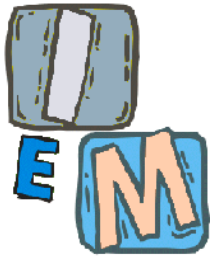

filtro mediano 3x3 (2 volte)

# Rimozione del rumore S&P

Filtro mediano 3x3

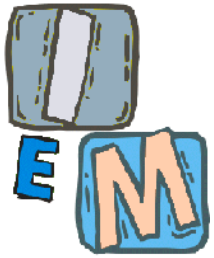

# Rimozione del rumore S&P

Filtro mediano 5x5


# Media vs mediano

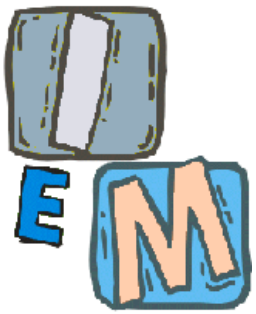
- Perché i filtri mediani danno risultati migliori rispetto a quelli di media?
- Il filtro media tende a creare dei livelli di grigio prima non esistenti.
- Il filtro di media non attenua solo il rumore ma anche tutte le alte frequenze spaziali in maniera indiscriminata dando origine ad immagini sfocate.
- Il filtro mediano non deteriora i lati, ma elimina i picchi con base piccola rispetto al kernel.



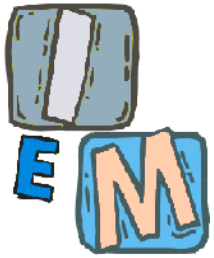

# Altri filtri (per la rimozione del rumore)

Esistono altri filtri non lineari molto importanti:

- **Outlier:** il valore del pixel centrale viene confrontato con il valore della media dei suoi 8 vicini. Se il valore assoluto della differenza è maggiore di una certa soglia, allora il punto viene sostituito dal valore medio, altrimenti non viene modificato.
- **Olimpico:** da un dato intorno si scartano i valori massimo e minimo e sul resto si fa la media.



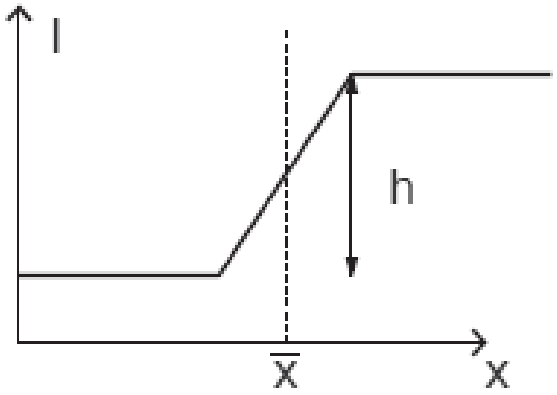

# Estrazione di contorni



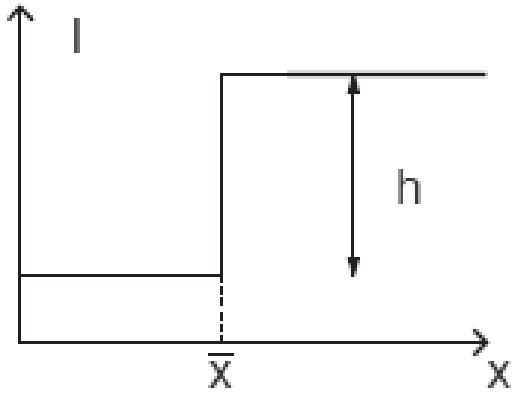

# Estrazione dei contorni

- Gli operatori locali ci aiutano ad estrarre i contorni da una immagine.
- I contorni sono definiti come delle discontinuità locali della luminanza.
- Gli *edge detector* forniscono immagini in cui sono preservate le variazioni di luminanza ed eliminate tutte le altre informazioni.

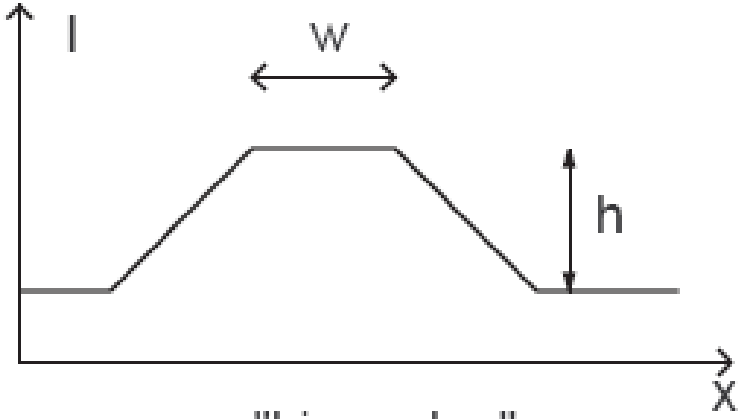
# Esempi di lato in 1D



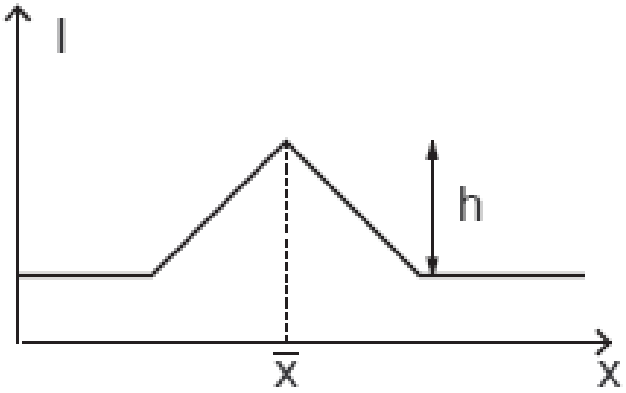

"Ramp Edge"




"Step Edge"



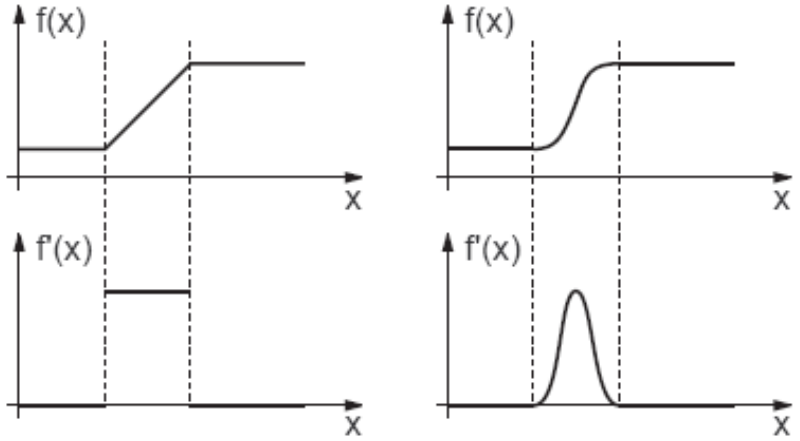

"Line edge"



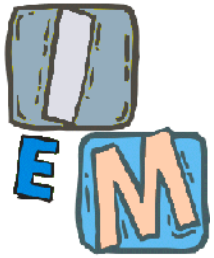

"Roof Edge"

# Edge detector basati sulla derivata prima

- Se ho un segnale monodimensionale e calcolo la derivata prima, scopro che i lati sono i corrispondenza dei massimi della derivata.




- Quindi i filtri devono calcolare la derivata in direzione x quella in direzione y e poi combinarle insieme.



# Kernel notevoli: lati orizzontali

Ne esistono molti, ne presentiamo due:

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

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

# Sobel x


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

Filtro x-Sobel

# Prewitt x

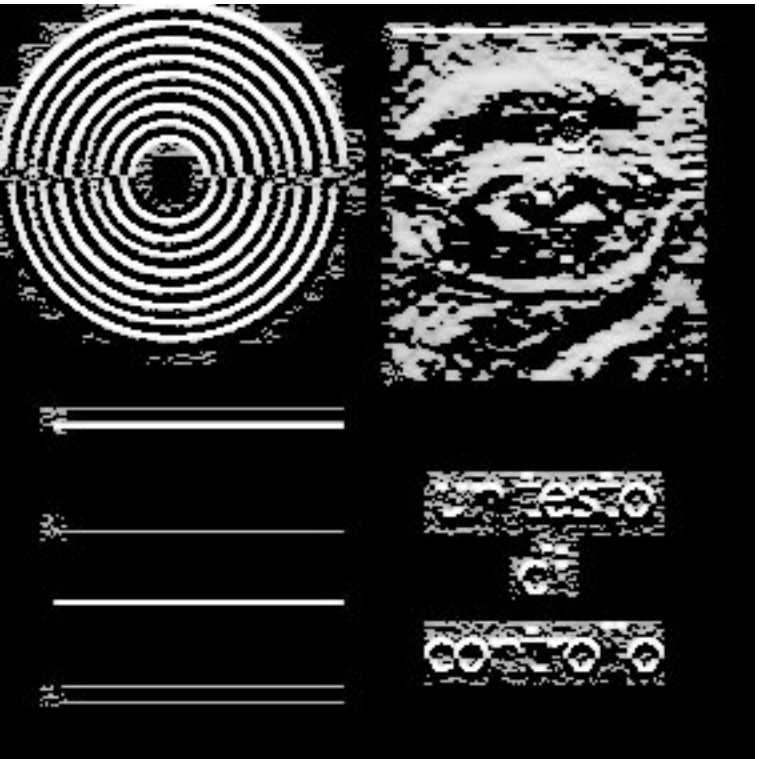

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

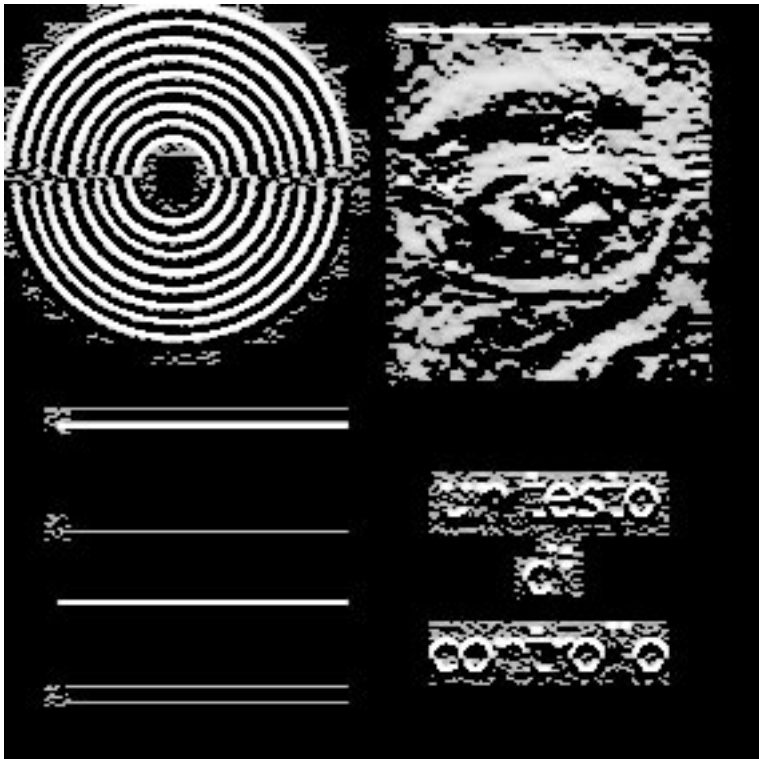
Filtro x-Prewitt

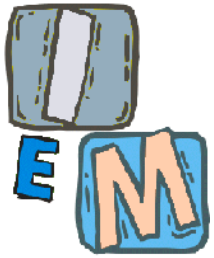
**Dopo l'applicazione di X-Prewitt e un passo di equalizzazione**

un testo
di
controllo

**Dopo l'applicazione di x-Sobel e un passo di equalizzazione**







# Kernel notevoli: lati verticali

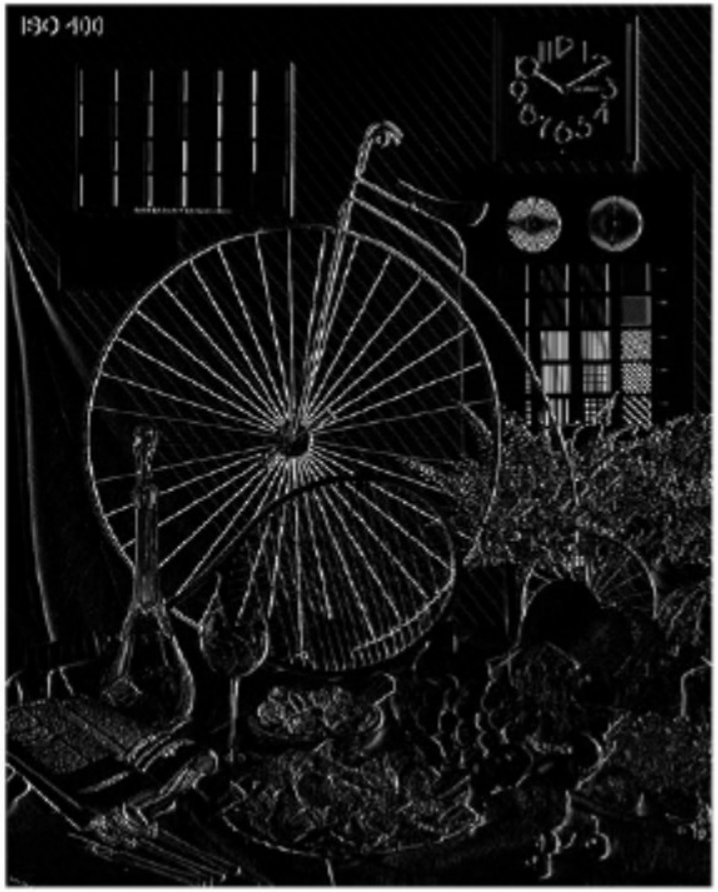
La situazione è identica al caso dei lati orizzontali, i filtri sono solo ruotati di 90 gradi.

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

# Sobel y


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

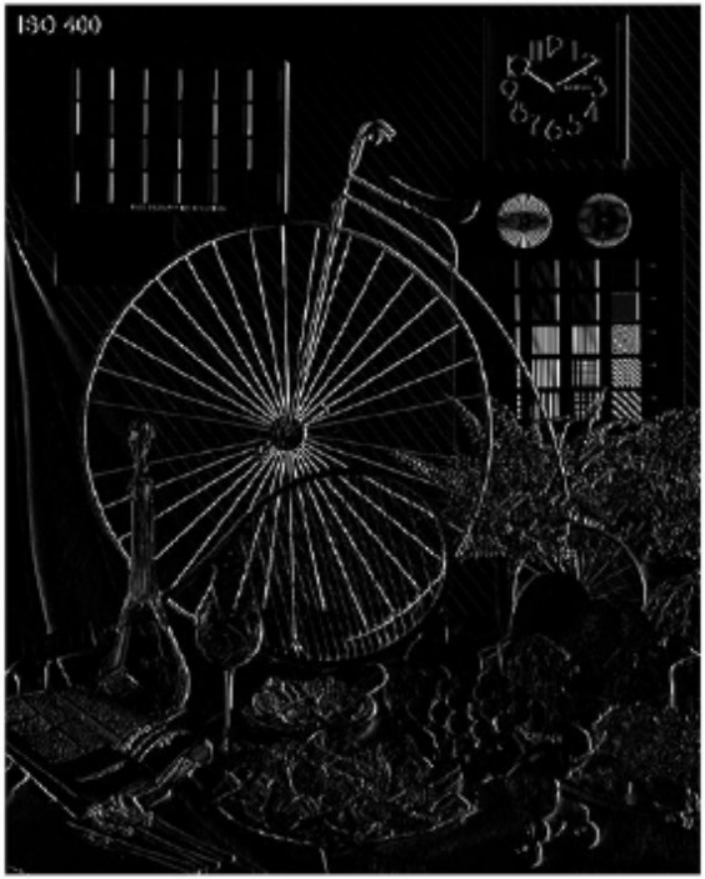
Filtro y-Sobel



# Prewitt y


| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

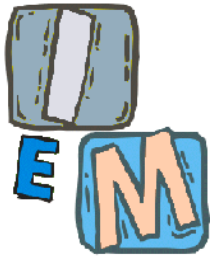
Filtro y-Prewitt



un testo
di
controllo

**Dopo l'applicazione di Y-Prewitt e un passo di equalizzazione**


**Dopo l'applicazione di y-Sobel e un passo di equalizzazione**



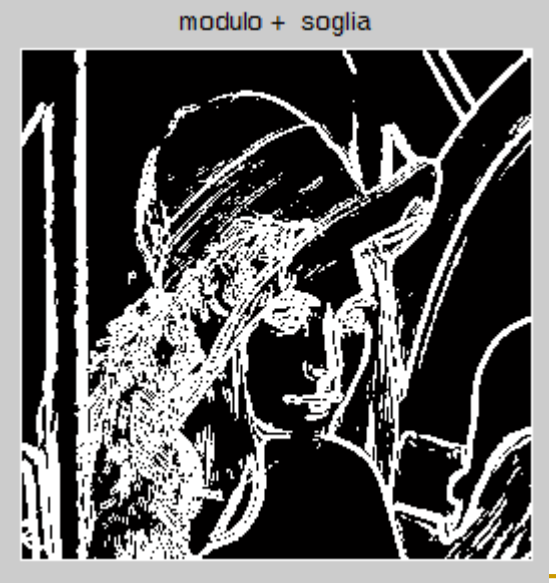

- Sobel x fornisce una matrice con i lati orizzontali (e le componenti orizzontali dei lati obbliqui) che hanno valori non nulli.
- Sobel y fornisce una matrice con i lati verticali (e le componenti verticali dei lati obbliqui) che hanno valori non nulli.
- Le due matrici possono essere combinate insieme mediante la seguente formula

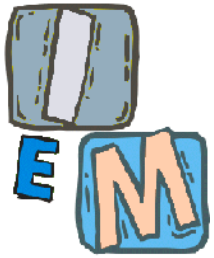
  sqrt(Sobel_x^2+Sobel_y^2)
- la matrice ottenuta ha valori non nulli per i pixel «di lato». Se si fissa una soglia adeguata, si può ottenere una matrice binaria che per ogni pixel ci dica se è o non è di lato.
- Ovviamente le stesse considerazioni valgono per Prewitt.


Sobel x


sobel y


modulo


modulo + soglia



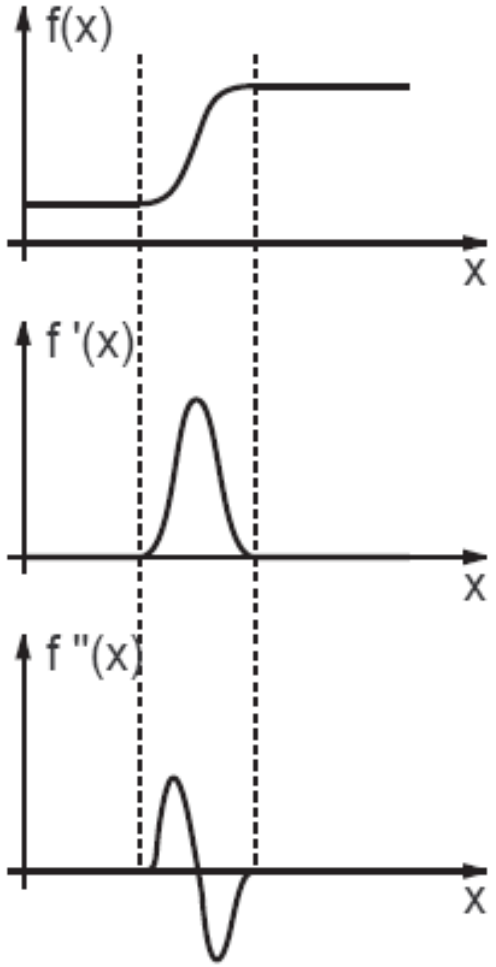

# Migliori risultati…

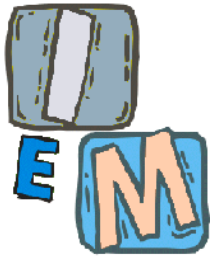
**Si ottengono** con algoritmi più sofisticati (non lineari) per il calcolo della grandezza del gradiente (somma del quadrato della risposta di un edge finder orizzontale e del quadrato della risposta di un edge finder verticale)

**Si ottengono** con strategie più "intelligenti" (algoritmo di Canny, algoritmi fuzzy, tecniche di backtracking eccetera)

# Edge detector basati sulla derivata seconda

- Se ho un segnale monodimensionale e calcolo la derivata seconda, scopro che in corrispondenza del lato essa passa per lo zero.





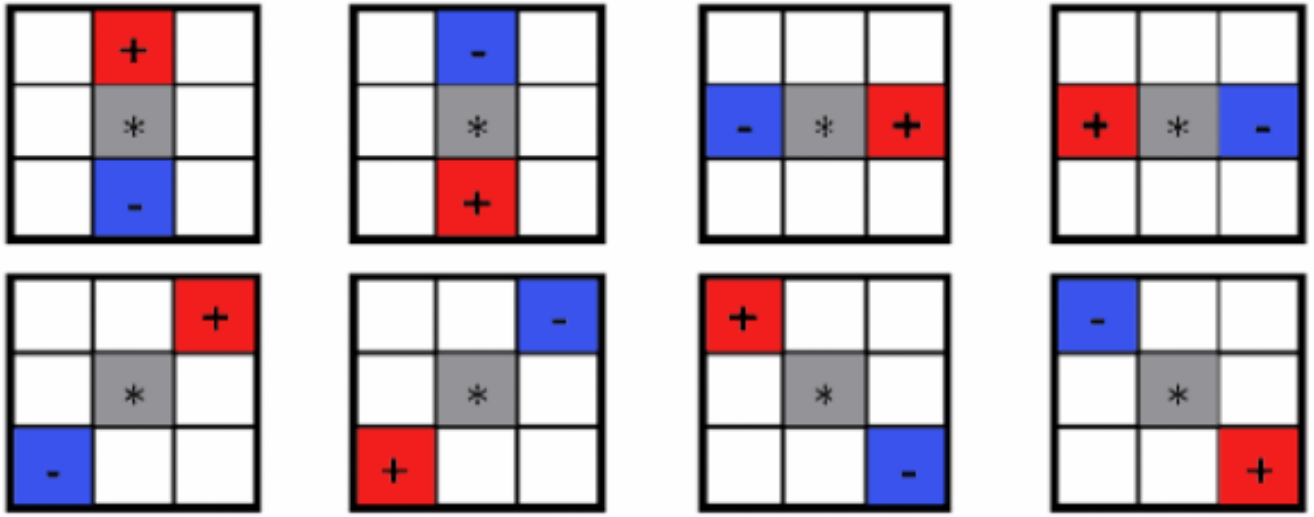
# Kernel notevoli: Laplaciano

Il filtro più diffuso per calcolare la derivata seconda è detto Laplaciano, ed è definito dalla maschera:

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

# Zero-crossing

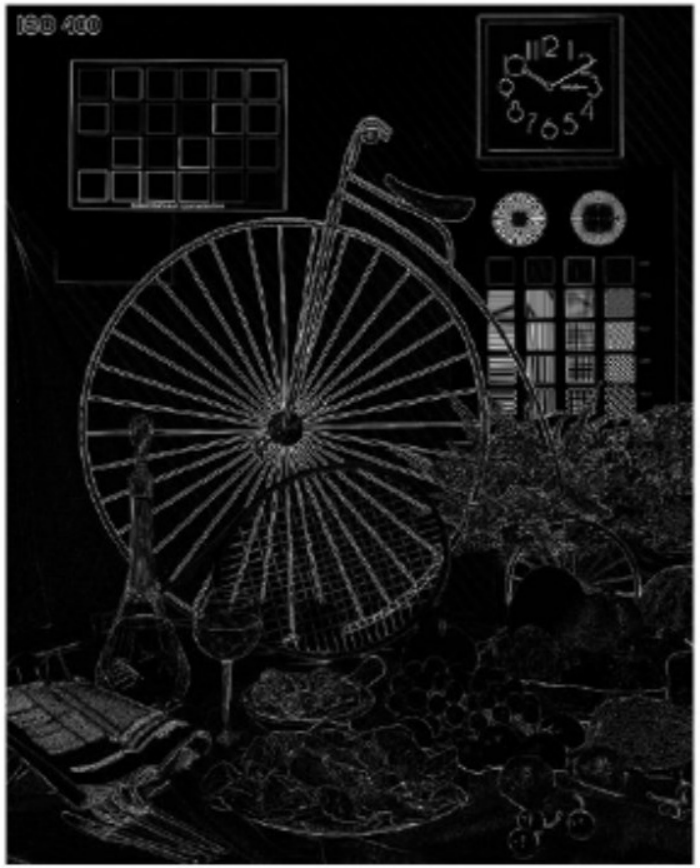
- Dopo aver applicato l'operatore Laplaciano è necessario che si verifichi la condizione di Zero-crossing. Cioè, deve sempre accadere che rispetto al punto in questione ci sia nel suo intorno un valore positivo e un valore negativo.



# Laplaciano


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

Filtro Laplaciano



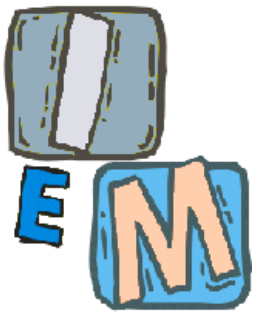
# Laplaciano


Sobel

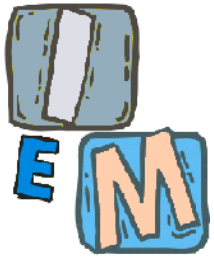
Zero Crossing

Prewitt

Canny




# Filtri di sharpening




# Filtri di sharpening

- Sono filtri il cui scopo è quello di incrementare la nitidezza di una immagine aumentando il contrasto locale.
- Questa è una operazione opposta allo sfocamento.
- Per ottenere tale effetto si può adottare una maschera che, derivata dal Laplaciano, “rinforza” i lati presenti nell’immagine.
- Purtroppo essa rinforza anche il rumore presente nella immagine!

# Filtro di sharpening

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

Filtro di Enhancing