

8 Operatori_locali_filtri_di_effetti

Base canonica

Immaginiamo un vettore:

$T = [234, 204, 34, 16, 44, 134, 12, 11, 56]$

Possiamo pensarla così:



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

La stessa cosa vale per le immagini:

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

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

$$\begin{array}{r}
 & \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \\
 + 204 & * & + 34 & * \\
 \end{array}$$

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

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

$$\begin{array}{r}
 & 0 & 0 & 0 \\
 + 44 * & 0 & 1 & 0 \\
 & 0 & 0 & 0
 \end{array}
 \quad
 \begin{array}{r}
 + 134 * \\
 \hline
 \end{array}$$

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

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

$$\begin{array}{r}
 \begin{array}{c|c|c}
 & 0 & 0 & 0 \\
 \hline
 0 & 0 & 0 & 0 \\
 \hline
 0 & 1 & 0 & \\
 \end{array}
 \\[10pt]
 + 11 * \qquad \qquad \qquad + 56 *
 \end{array}$$

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

Questo ragionamento si chiama Base canonica

Operatori Locali

Sono operazioni che come output producono un pixel il cui valore si basa su un intorno dello stesso pixel (come gli operatori puntuali ma anziché usare solo il pixel scelto, usano un intorno di quel pixel)

Sono usati per migliorare la qualità delle immagini o per estrarre delle informazioni dall'immagine

Si possono pensare come filtri dell'immagine

Un filtro è ottenuto facendo la convoluzione tra l'immagine ed una matrice

Operatori Lineari

Alcuni operatori locali possono avere questa proprietà, ovvero essere lineari, infatti un operatore:

$F : V \rightarrow W$ 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:

$$F(av_1 + bv_2) = aF(v_1) + bF(v_2)$$

Dove:

- v_1 e v_2 sono dei vettori (array) che contengono dei valori di determinati pixel: ad esempio un vettore di lunghezza 9 può contenere un gruppo 3x3 di pixel all'interno di un'immagine più grande
 - a e b sono 2 numeri reali

La formula detta in maniera semplice:

Se prima mescoli due pezzetti di immagine e poi fai la funzione, ottieni lo stesso risultato che avresti facendo la funzione sui due pezzetti separati e poi mescolando i risultati.

Questa cosa vuol dire che un operatore è lineare

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. Una base è un insieme di elementi parte di V con cui posso "costruire" il resto di V, quindi mi basta sapere come funziona F su questi elementi di base e so come si comporta su tutti gli altri elementi di V

Per controllare quindi se data una funzione questa sia lineare o no basta vedere se entrambi i membri della formula sopra siano uguali

 | **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

$$\begin{aligned} a*(255-x_1, 255-y_1) + b*(255-x_2, 255-y_2) &= \\ (a*255-a*x_1, a*255-a*y_1) + (b*255-b*x_2, b*255-b*y_2) &= \\ (a*255-a*x_1 + b*255-b*x_2, a*255-a*y_1 + b*255-b*y_2) \end{aligned}$$

Il secondo membro è

$$\begin{aligned} (255 - (ax_1 + bx_2), 255 - (ay_1 + by_2)) &= \\ (255 - ax_1 - bx_2, 255 - ay_1 - by_2) \end{aligned}$$

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

Invarianza per traslazione

Gli operatori possono avere anche questa proprietà, ovvero dare lo stesso effetto indipendentemente dalla posizione dell'intorno di pixel scelto, se rispettano questa

condizione sono invarianti per traslazione. Es. sfocatura su tutta l'immagine.
Altrimenti si dice che sono NON invarianti per traslazione. Es. sfocatura solo sui bordi dell'immagine (quindi dipende dalla posizione del pixel scelto)

Definizione accademica

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), ad esempio il negativo è shift invariant)

Riassumendo:

- Se F è **lineare** per descriverlo basta conoscere il comportamento su tutte le immagini impulsive (Un'**immagine impulsiva** è un'immagine che è **tutta zero**, tranne **un singolo pixel** messo a 1 (o a un altro valore noto) in una certa posizione. È l'equivalente, per le immagini, di un vettore base)
- 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

MASCHERA DELL'OPERATORE

La "risposta all'impulso" o "point spread function" di F è la carta d'identità di tale operatore

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

Si consideri l'operazione che preso un impulso:

$$\begin{matrix} & \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} & \text{Io trasforma in:} & \begin{matrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{matrix} \end{matrix}$$

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 vedi su (quella a destra con i due 0,5) è la matrice che descrive la risposta all'impulso e si chiama **kernel**, per ragioni che vedremo a breve è detta anche maschera di convoluzione di F

Kernel finiti, infiniti e complessità

La grandezza del kernel può variare fino ad essere infinita, per ragioni pratiche, però, si usano solo kernel con dimensioni finite

Attenzione: → per kernel infinito non si intende che l'immagine ha grandezza infinita

Esempio: (la gaussiana è una sfocatura sull'immagine)

Teoricamente → kernel infinito

La gaussiana produce valori tipo:

| distanza | valore |
|----------|----------------------------|
| 0 | 1.0 |
| 1 | 0.6 |
| 2 | 0.14 |
| 3 | 0.01 |
| 4 | 0.0003 |
| 5 | 0.000006 |
| ... | non è mai esattamente zero |

Non finisce mai.

Praticamente → kernel finito

Ci si ferma quando i valori sono troppo piccoli per fare differenza.

Per esempio si taglia a distanza 2:

| posizione | valore |
|-----------|--------|
| -2 | 0.14 |
| -1 | 0.60 |
| 0 | 1.00 |
| +1 | 0.60 |
| +2 | 0.14 |

E si ottiene un kernel **5×5** (in 2D).

Le dimensioni del kernel influenzano la complessità della operazione di filtraggio. Tale complessità dipende ovviamente anche dal numero dei pixel di una immagine

Filtri convolutivi

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

La convoluzione è un'operazione fondamentale per ogni tipologia di signal processing e perciò dobbiamo studiarla.

Convoluzione: proprietà

1. La convoluzione si indica con: $g = f \cdot h$
2. La convoluzione è commutativa $f \cdot h = h \cdot f$
3. La convoluzione è associativa $(f \cdot h) \cdot h_1 = f \cdot (h \cdot h_1)$

Abbiamo quindi 2 casi

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

Se il kernel h ha dimensioni $s \times t$ la formula va riscritta nella seguente maniera:

$$g_{m,n} = \sum_{i=-s/2}^{s/2-1} \sum_{j=-t/2}^{t/2-1} (h_{i,j} \cdot f_{m+i,n+j})$$

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

2. Se gli indici del kernel sono disposti partendo da 1 fino ad arrivare a s e t :

Se il kernel h ha dimensioni $s \times t$ la formula va riscritta nella seguente maniera:

$$g_{m,n} = \sum_{i=1,j=1}^{s,t} h_{i,j} \cdot f_{m+(i-s+s/2),n+(j-t+t/2)}$$

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

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

(la convoluzione dovrebbe essere la formula stessa dell'operazione)

Problemi nell'implementazione

Così come si verificano problemi ai bordi per le operazioni di interpolazione si verificano anche con gli operatori locali, perché giustamente hanno bisogno di pixel che non ci sono se ci troviamo ai bordi dell'immagine.

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; (toroidale significa che i bordi si toccano con il loro opposto, immagina una cartina geografica in cui il lato destro si tocca con il sinistro e il lato sopra con quello sotto, la forma che si otterrebbe è una sorta di ciambella)
- 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)

Le aree in grigio non verranno calcolate

| input | | | | | output | | |
|-------|---|---|---|---|--------|----|----|
| 1 | 2 | 2 | 3 | 1 | | | |
| 3 | 2 | 2 | 1 | 4 | 30 | 45 | 30 |
| 2 | 5 | 2 | 7 | 1 | 46 | 27 | 37 |
| 9 | 0 | 1 | 1 | 2 | 34 | 41 | 28 |
| 3 | 1 | 2 | 4 | 1 | | | |

b)

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)

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)

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

Operatore mediano

E' un operatore non lineare (quindi non vale la formula in ambo i membri) e restituisce la

media dei valori dei pixel in un intorno, è un operatore statistico detto *order statistic*

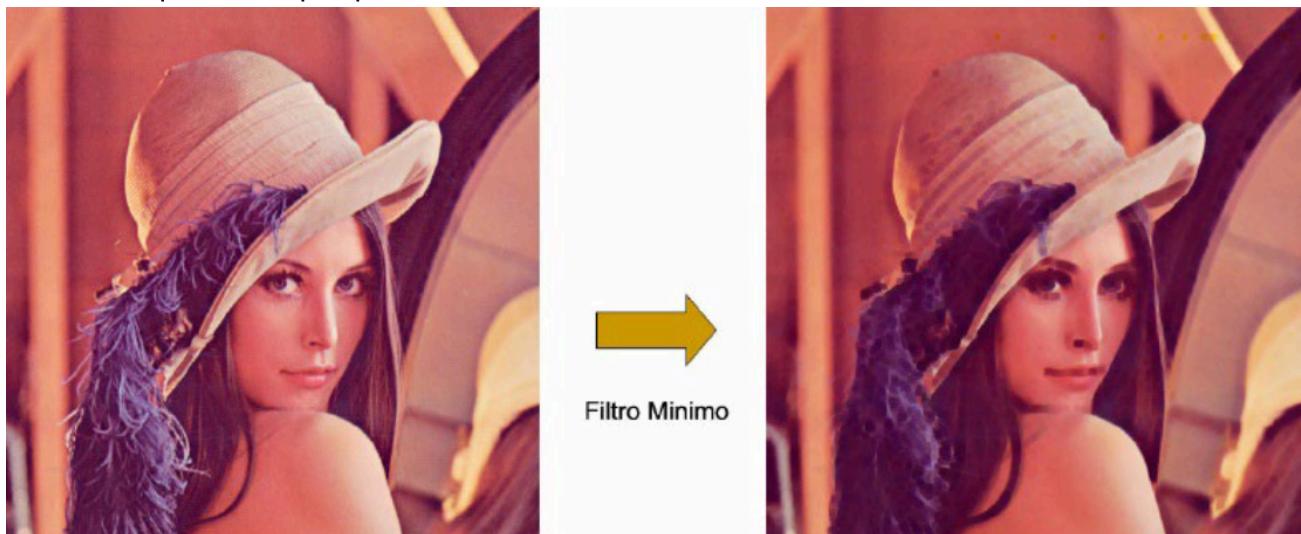


di operatori order statistic ne esistono altri come:

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

1. Se si sostituisce con il minimo si ottiene un incupimento dell'immagine (si eliminano per esempio macchie chiare);
2. Se si sostituisce con il massimo si ottiene uno schiarimento dell'immagine (si eliminano per esempio punti neri)





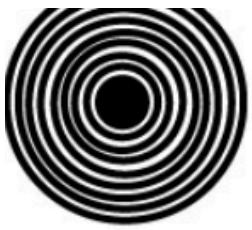
Filtro N-box

E' un filtro che si occupa di sfocare l'immagine, si opera in un intorno NxN e il filtro kernel si compone con $\frac{1}{N^2}$ con N generalmente dispari

Esempi:

$$\text{3-box} \quad \frac{1}{9} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

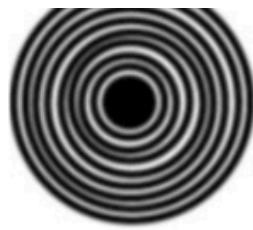
Quindi ogni elemento della matrice è 1/9 perché N = 3
con N = 5 diventa 1/25 ecc...



originale

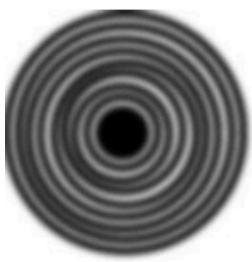


un testo
di
controllo



3-box

un testo
di
controllo



5-box



un testo
di
controllo



7-box

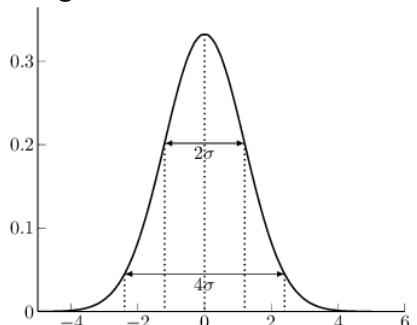
un testo
di
controllo

Filtro N-binomiale

E' un filtro che smussa/sfoca l'immagine ma in maniera meno vigorosa del N-box

Hanno un kernel derivato da una gaussiana, infatti si chiamano filtri gaussiani

(la gaussiana si descrive con questo grafico:



quindi il valore aumenta più si ci avvicina ad un elemento centrale)

3-binomiale

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

1/16 *



originale

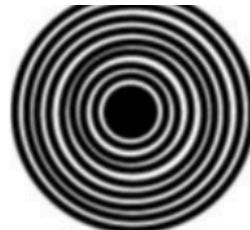


un testo
di
controllo

5-binomiale

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

1/256 *



3-binomiale

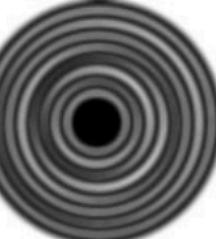


un testo
di
controllo

5-binomiale



7-binomiale



un testo
di
controllo

Ridurre il rumore

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

Esistono 2 tipi principali di rumore:

1. rumore sale e pepe, caratterizzato dalla modifica di alcuni pixel (in %)
2. Rumore gaussiano bianco, caratterizzato dalla media e dalla varianza

Sale e pepe

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

Esempi di rumore «sale e pepe» con:

1% di pixel danneggiati



10% di pixel danneggiati



Rimozione del rumore sale e pepe

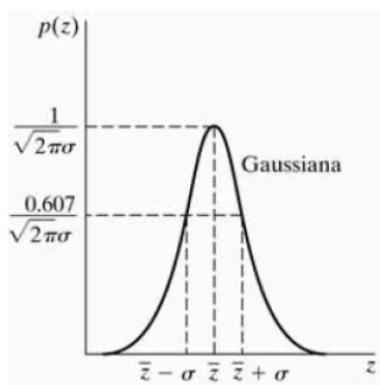
Si possono usare sia un filtro mediano che uno di media, quello mediano in questo caso è migliore

input**rumore sale e pepe****filtro media****filtro mediano**

Rumore gaussiano

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

dove z rappresenta l'intensità, \bar{z} è il valore medio¹ (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 $[(\bar{z} - 2\sigma), (\bar{z} + 2\sigma)]$.



Rimozione del rumore gaussiano

Anche qui usiamo filtri mediano e media, qui a differenza del sale e pepe è il filtro media a essere migliore



Dimensione del kernel

Per togliere il rumore in maniera migliore è meglio usare un kernel grande o uno più piccolo e applicarlo più volte?

Entrambi riducono il rumore ma applicare più filtri piccoli sfoca di meno l'immagine

input



rumore sale e pepe



filtro mediano 5x5



filtro mediano 3x3 (2 volte)



altri esempi con rumore sale e pepe

Filtro mediano 3x3



Filtro mediano 5x5



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

Ci sono anche altri 2 filtri per rimuovere il rumore:

- **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

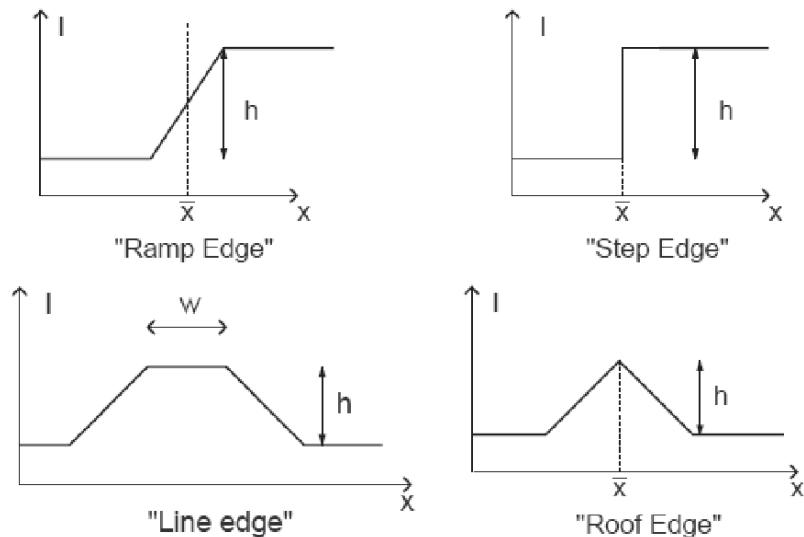
Si dice che un filtro è **energy preserving** se non modifica la luminanza di un'immagine
Esistono anche filtri che la alterano

Estrazione dei contorni

Un contorno è una discontinuità locale della luminosità di un'immagine, e gli operatori locali ci aiutano a estrarre questi contorni.

Gli **edge detector** restituiscono un'immagine in cui sono preservate le variazioni di luminosità e sono eliminate tutte le altre informazioni.

Questi sono esempi di contorni

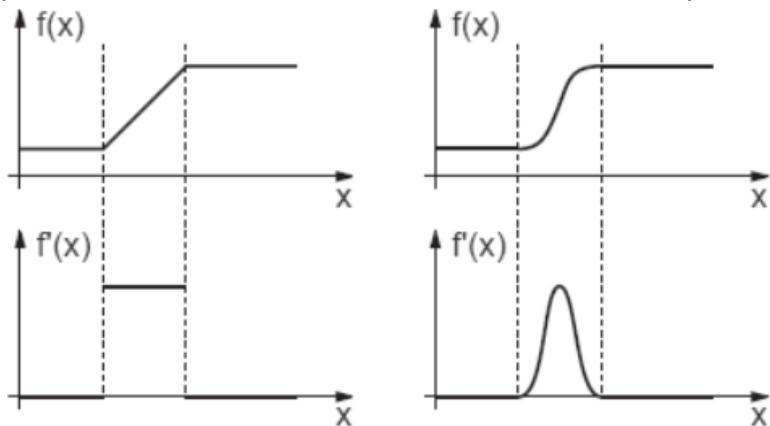


Edge detector basati sulla derivata prima

Il modo più semplice per trovare un contorno in un'immagine è cercare dove la luminosità (intensità) cambia più rapidamente. Matematicamente, questo cambiamento rapido si calcola con la **Derivata Prima**

Se stai attraversando un contorno, la luminosità passa bruscamente da un valore all'altro.

- La **Derivata Prima** ($f'(x)$) misura la "pendenza" di questo cambiamento, i contorni quindi si trovano esattamente dove la derivata prima ha il suo massimo (picco)



Nel punto in cui la derivata prima ha il suo picco c'è un contorno

Kernel notevoli: lati orizzontali

Ne esistono molti, ne presentiamo 2, questi quindi sono edge detector che calcolano la derivata prima:

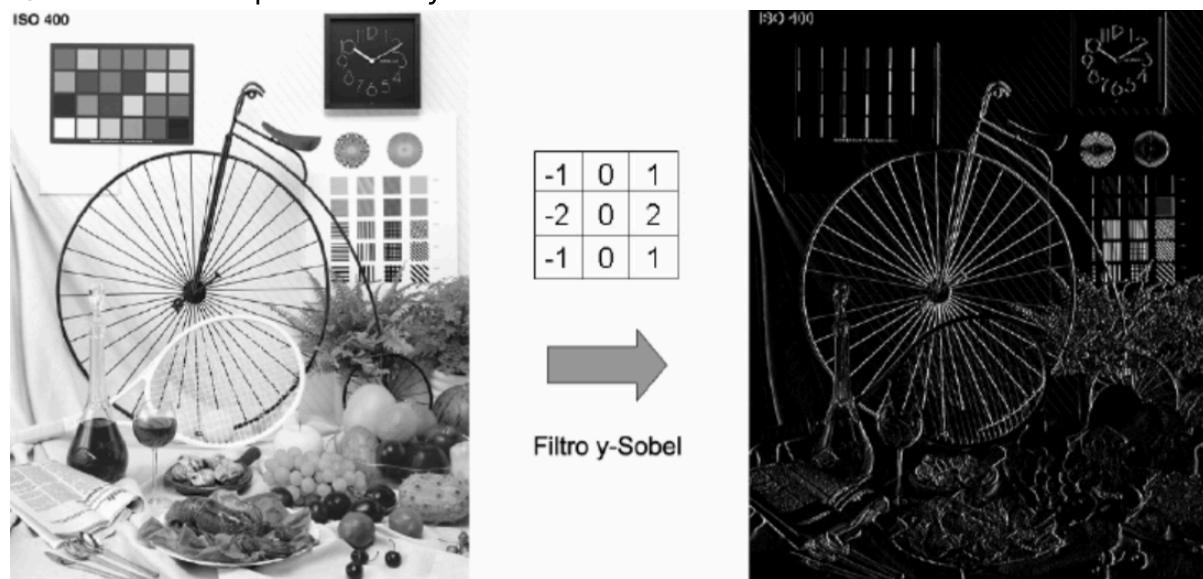
$$Sobel_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad Prewitt_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Latì verticali

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

Questi 2 kernel sono appunto degli *edge detector*, evidenziano quindi le differenze di luminosità

Questo ad esempio è sobel y



Sobel x fornisce una matrice, sobel y un'altra, queste matrici si possono combinare seguendo questa formula:

$$magnitudo = \sqrt{sobel_x^2 + sobel_y^2}$$

Questa cosa altro non è che una matrice che ci indica se un pixel appartiene o no ad un

contorno.

Ovviamente le stesse considerazioni valgono per Prewitt

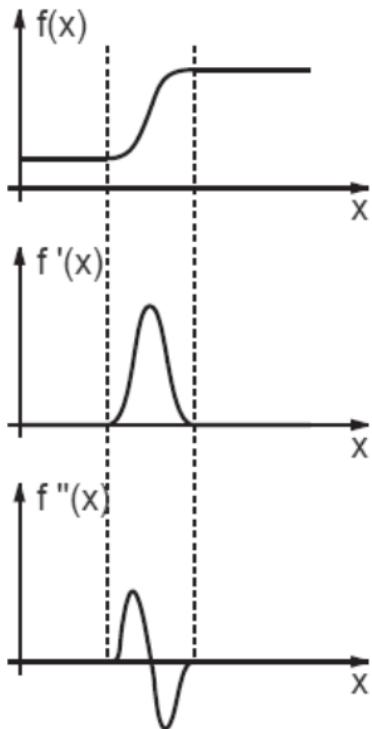
Migliori risultati...

Si ottengono con algoritmi più sofisticati (non lineari) per il calcolo della grandezza del gradiente (magnitudo)

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 contorno essa passa per lo zero



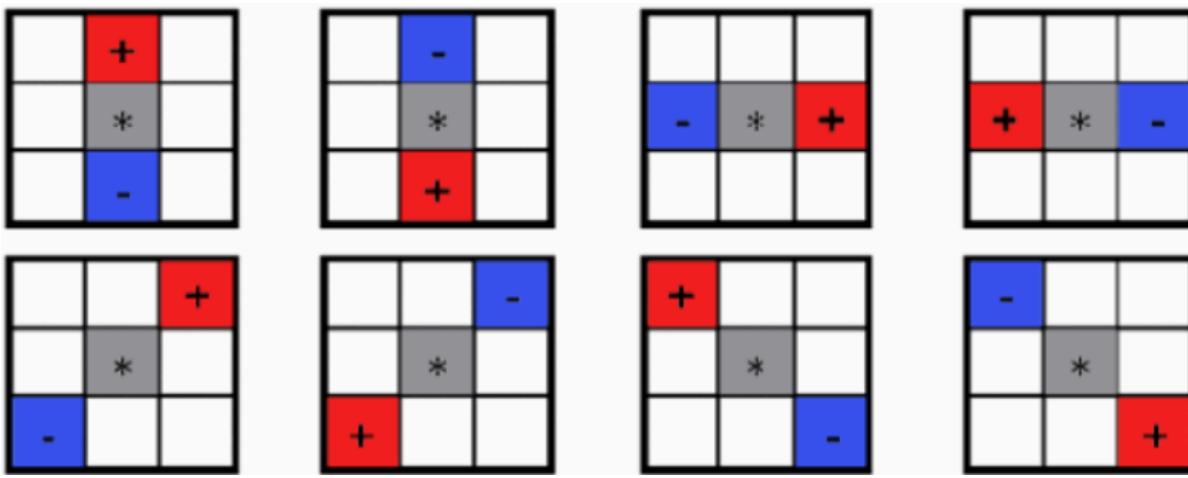
Kernel notevole: laplaciano

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

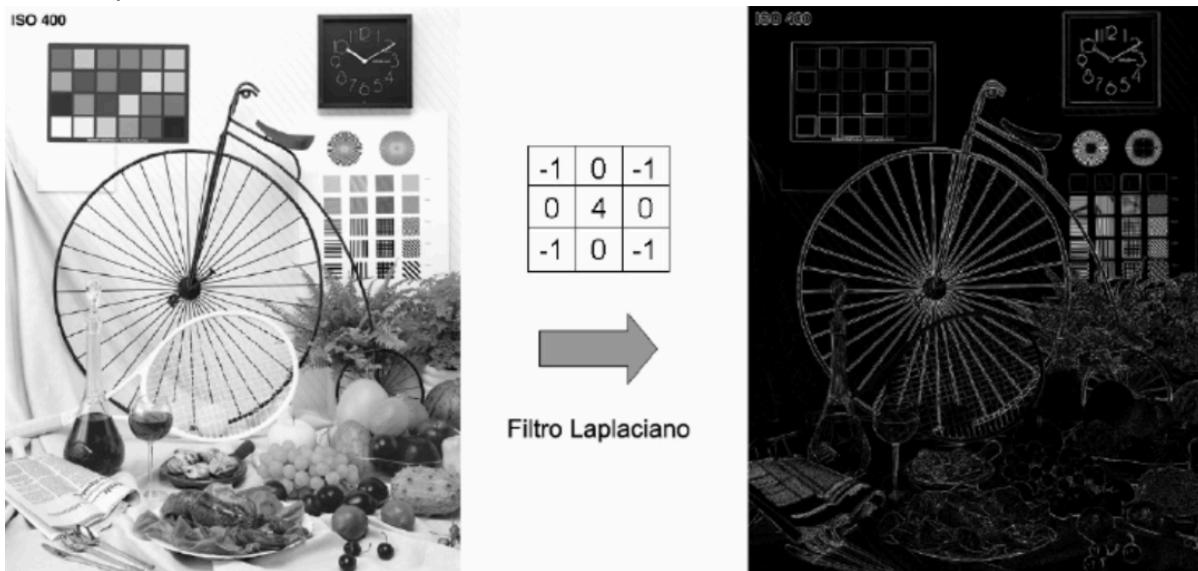
$$\text{Laplaciano} = \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

Zero crossing

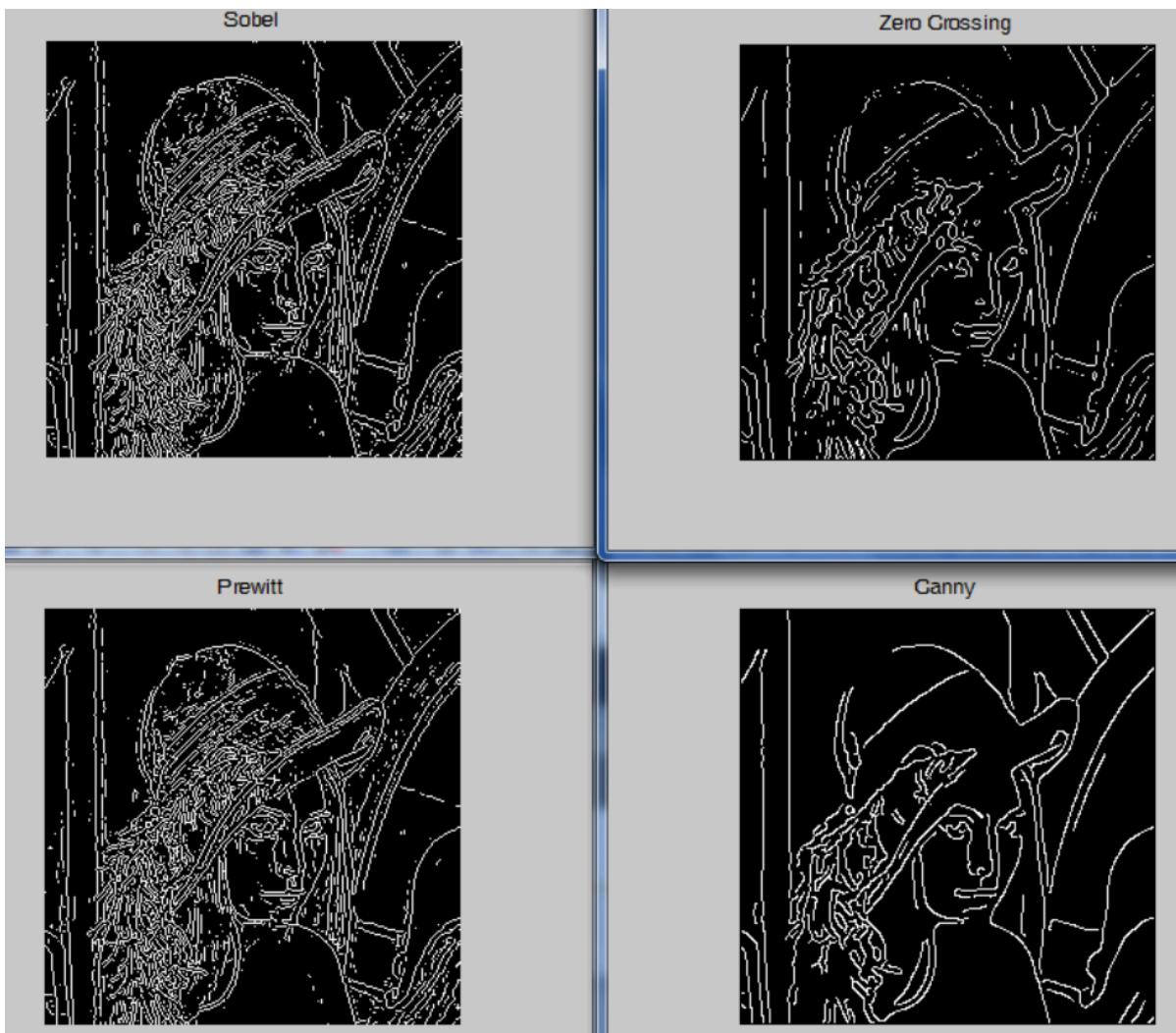
Dopo aver applicato l'operatore Laplaciano è necessario che si verifichi la condizione di Zero-crossing. Cioè, deve sempre accadere che rispetto al punto in questione ci sia nel suo intorno un valore positivo e un valore negativo



Filtro laplaciano



Confronti tra edge detector



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!

