

15-12-2022

Ripasso convoluzione

La convoluzione si applica **localmente** se si verifica le proprietà:

- **invarianza per traslazione;**
- **linearità dell'operatore** scelto.

La **formula** della convoluzione è la seguente:

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

In parole povere si considera il **KERNEL** (cioè la risposta all'operatore se viene applicato un impulso) e:

1. si sovrappone il kernel al punto scelto;
 2. si fa la somma dei prodotti punto a punto, ovvero:
 - si fa il prodotto degli elementi sovrapposti
 - si sommano tutti i risultati ottenendo il valore da mettere al centro (nella cella sul quale si sovrappone il kernel)
- Ai bordi invece si applica una delle 4 possibili soluzioni viste in precedenza

Mediano

L'operatore mediano è un **filtro non lineare** perchè usa un *ordinamento*.

E' **operatore locale** che usa la teoria del "calcolare statistica secondo l'*ordinamento*".

In parole povere: prende l'insieme dei pixel della "*finestra considerata*", li ordina, prende il valore mediano (valore **AL CENTRO**).

Mediano sta al centro dell'ordinamento (se si ha un insieme dispari di elementi).

Esempio mediano applicato ad una finestra 3x3

1. Si ordinano i pixel che si trovano all'interno della finestra;
2. Per esempio si trova il seguente ordinamento: 0 0 1 2 **5** 1 20 30 70
3. In questo caso il **mediano** è il valore che sta al centro, ovvero il 5.

Considerazioni

- Si usa lo stesso metodo quando si sovrappone il kernel a una porzione di matrice.
 - *come nell'utilizzo di un kernel*: se il mediano 3x3 è sovrapposto al valore di posizione (1, 1) allora nella nuova matrice il valore mediano trovato va nella

posizione (1, 1)

- In caso di **IMMAGINI A COLORI**, si prendono le 3 matrici dei 3 canali e si fa la stessa operazione per ogni canale.
- Dai **RISULTATI** dell'applicazione del filtro mediano si nota che l'immagine è leggermente sfocata perchè il mediano tende a eliminare valori più alti e/o più bassi.
 - Si abbassano localmente i contrasti e di conseguenza i bordi sono meno netti
- E' meglio una dimensione del filtro **DISPARI** in modo tale da poter avere un "*centro unico*";
 - in caso di filtro di **dimensioni PARI**, allora si prendono i **2 valori centrali** e si fa la media aritmetica fra essi.

Minimo e massimo

- Il filtro di minimo e/o massimo è un filtro **NON LINEARE** perchè esso fa uso di un ordinamento.
- Nella finestra considerata si prende il valore di massimo/minimo. In particolare:
 - il filtro di **MINIMO ABBASSA LA LUMINOSITA'** dell'immagine;
 - il filtro di **MASSIMO AUMENTA LA LUMINOSITA'** dell'immagine;

N-box (o "media aritmetica")

In questo caso si filtra la media di una finestra. Si mettono nel kernel degli valori specifici. Esso è un filtro di dimensioni $N \times N$ dove ogni valore(cella) del kernel è uguale $\frac{1}{N^2}$

Un filtro 3-box è così fatto:

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

oppure più semplicemente:

$$\frac{1}{9} \times \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Generalmente si preferisce filtro di dimensioni dispari per avere un punto centrale determinato.

Esempi filtro n-box:

- **3-box:**

1/9 *

1	1	1
1	1	1
1	1	1

•

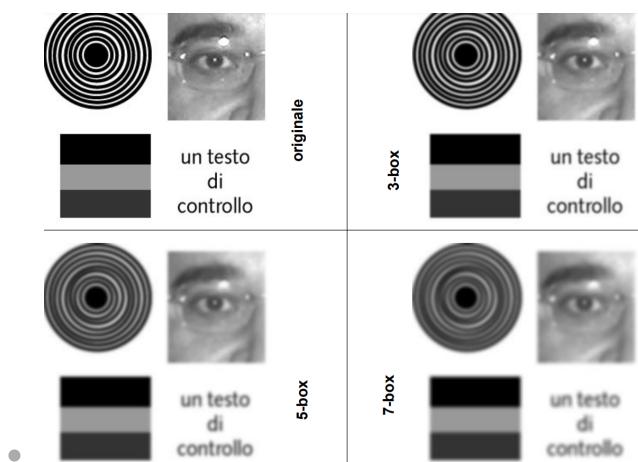
- 5-box:

5-box	<table border="1"> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr> <td>1/25 *</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	1	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																						
1/25 *	1	1	1	1																						
1	1	1	1	1																						
1	1	1	1	1																						

●

Considerazioni

- Il filtro n-box **sfoca l'immagine** perchè prende un valore e lo fa diventare quello che stava al centro, *fa avvicinare tutto verso il centro*.
- La **sfocatura** è molto **forte** in **orizzontale e verticale** ma meno in **diagonale**
- A differenza del mediano, la media aritmetica **produce valori di luminanza NON PRESENTI** nell'immagine originale. (**ESAME**)
- più è grande il filtro n-box e maggiore è l'effetto sfocatura che si ottiene.



- Questo filtro è di tipo **CONSERVATIVO** perchè conserva energia
- Esso ha **somma 1** e usa solo valori **positivi**.
 - Infatti $\frac{1}{n^2} > 0$. Per esempio: $\frac{1}{9} > 0$
 - $\frac{1}{9} + \frac{1}{9} = 1$
- Il segnale di **output**, quindi, **appartiene allo stesso dominio** del dominio di partenza.

Applicazione filtro n-box (media)

Data un'immagine 3×3 :

- per calcolare il valore in posizione $(1, 1)$ si usa la *convoluzione*;
- moltiplico ogni valore dell'input per $\frac{1}{9}$;
- sommo tutti i valori ottenuti dal passo precedente e scrivo il risultato nella nuova matrice in posizione $(1, 1)$.
- il risultato può venire con la virgola e, se siamo interessati ai numeri interi si arrotonda, altrimenti non si arrotonda e si lascia il risultato con la virgola.

N-binomiale ("gaussiano")

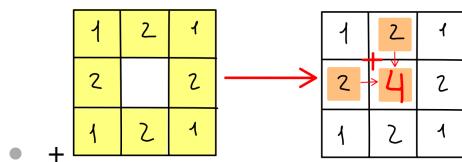
Il kernel di un filtro n-binomiale è fatto nel seguente modo:

3-binomiale	5-binomiale
$1/16 * \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$	$1/256 * \begin{matrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{matrix}$

I valori nel kernel seguono la **distribuzione statistica binomiale**:

Si ricavano quali sono i coefficienti di $(a + b)^n$

- Come si trovano i valori di tutto il kernel?
 - Dopo aver inserito i bordi (da ricordare a memoria), si possono trovare i successivi (quindi i valori via via più centrali del kernel) sommando i vicini in diagonale.



Considerazioni

- Il filtro n-binomiale non pesa tutti i pixel di un'immagine allo stesso modo come faceva n-box, infatti per ogni cella c'è un "peso" diverso;
- n-binomiale si può applicare per **convoluzione**;
- Si sfoca l'immagine in maniera più **CONTROLLATA** in tutte le direzioni:
 - tiene conto della distanza dal punto in cui sto facendo l'applicazione
- Anche in questo caso si preferisce un **kernel di dimensione dispari**
- E' essenziale ricordare il pattern 3-binomiale e 5-binomiale (basta ricordare gli elementi della 1 riga e 1 colonna dei 2 filtri per poter trovare quelli più centrali)

Conservazione dell'energia

Si dice che un filtro “*conserva l'energia*” se la **somma dei suoi pesi** fa 1. In tal caso l'energia viene solo distribuita nelle varie celle e quindi non la si perde e non la si guadagna.

- I filtri **n-box** e **n-binomiale** sono **Energy Preserving** visto che la somma dei valori nel kernel di entrambi i filtri è sempre 1.

NOISE CLEANING E SMOOTHING

Il **RUMORE** è un **segnale indesiderato** ed è **casuale**.

Il rumore può essere di diversi tipi: (sono abbastanza frequenti)

- **IMPULSIVO**, detto anche sale e pepe: compaiono valori estremi, ovvero valori massimi e minimi. In particolare compaiono pixel bianchi per i massimi e pixel neri per i minimi;
- **GAUSSIANO BIANCO**, caratterizzato dalla **varianza** e dalla **media**.
 - E' rumore che induce uno spostamento nel valore dei pixel originali come se fosse una matrice a parte (che ha valori più o meno alti) che si **somma all'immagine originale**.
 - Normalmente i spostamenti più grandi (**variazione più grande** di valori di pixel) sono **più rari**.
 - Questo tipo di filtro segue la una **distribuzione gaussiana di valori**.

Compare ogni volta che si scatta una foto in condizioni di scarsa luminosità. In questo caso si può ridurre il rumore con questi filtri appena visti (n-box e n-binomiale)

I filtri visti fino ad ora riducono il rumore. In che modo?

Perchè per questi tipi di rumori questi filtri vanno bene?

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 $a = 0$ e $b = 255$), abbiamo il **rumore sale e pepe**.

In particolare:

- $p(z)$ è la probabilità di osservare il valore z dove z è un certo impulso;
- se $a = 0$ e $z = 255$ allora ho una certa probabilità di trovare un pixel bianco alterato e uno nero alterato.

Probabilità di osservare un rumore:

- se la probabilità è alta c'è più rumore e viceversa. Quindi **alta probabilità di osservare rumore \Rightarrow più rumore**
- $p(z) = 10\%$ se $z = 0$ oppure 15% se $z = 255$, 0 altrimenti
 - se ho un'immagine che effettua questo rumore $p(z) = 10\%$ allora il 10% dei pixel sono neri e non dovevano esserlo. Idem per quelli bianchi.

Esempi di rumore sale e pepe con 1% di pixel danneggiati



Cioè 1/100 pixel diventa nero e 1/100 pixel diventa bianco

Esempi di rumore sale e pepe con 10% di pixel danneggiati



Questa situazione (*formazione di rumore*) si crea quando si ha un dispositivo di digitalizzazione che funziona male o in una fotocamera dove si sono bruciati pixel.

Questo rumore si attenua con l'**operatore mediano**:

- In ogni finestra il massimo(bianco) e il minimo (nero) che vengono prodotti direttamente dal rumore sono agli **estremi dell'ordinamento**;
- Vengono eliminati grazie all'applicazione filtro mediano che non considera, appunto, gli estremi.

Applicazione:



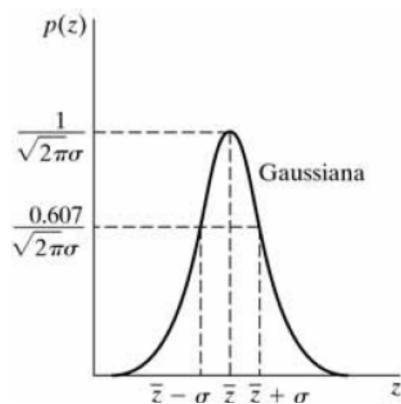
Un esempio di applicazione è il filtro "correggi" nei programmi di editing che stima i valori vicini quando si eliminano le imperfezioni. (Si tratta di **inpainting**)

Rumore gaussiano

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

Questa formula descrive la funzione gaussiana (*formula da imparare*). In particolare si ha:

- \bar{z} **media** della curva gaussiana;
- σ **deviazione standard** della curva gaussiana
- z è un valore di luminanza.



La domanda da farsi è: *Qual è la probabilità di osservare uno spostamento di luminanza pari a z ?*

Per ogni possibile spostamento ci dice la probabilità di avere quello spostamento.

Considerazioni

In pratica, più si allontana dalla media, più la probabilità è bassa e viceversa.

- Di quanto si estende la probabilità (σ , detta anche **varianza**)?
 - Lo spostamento che posso osservare è più piccolo comporta σ piccolo e viceversa
- Di norma la *media* = 0, σ molto piccola (varianza piccolissima, ovvero si ha poco rumore)

Esempio:

- $-3\sigma < \text{valore} < 3\sigma$
- se $\sigma = 3$ allora il valore sarà molto probabilmente $-9 < \sigma < 9$ e $\bar{z} = 0$

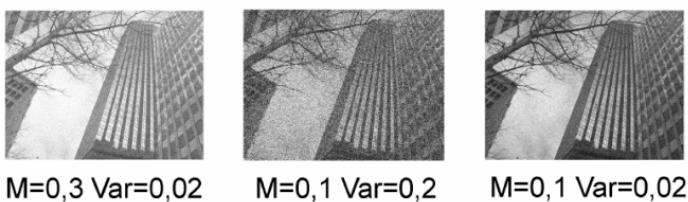
| La varianza σ dovrebbe essere quanto più piccola possibile



Per il **rumore gaussiano**, il **filtro di media** si è comportato leggermente meglio.
Questo rumore gaussiano avrà media $\bar{z} > 0$ perché l'immagine appare più chiara, mediamente ho sommato un valore positivo.



Gaussiano



M=0,3 Var=0,02

M=0,1 Var=0,2

M=0,1 Var=0,02

Ridurre il rumore nell'immagine a destra è più semplice rispetto all'immagine centrale.



Gaussiano



Filtro Mediano 5x5



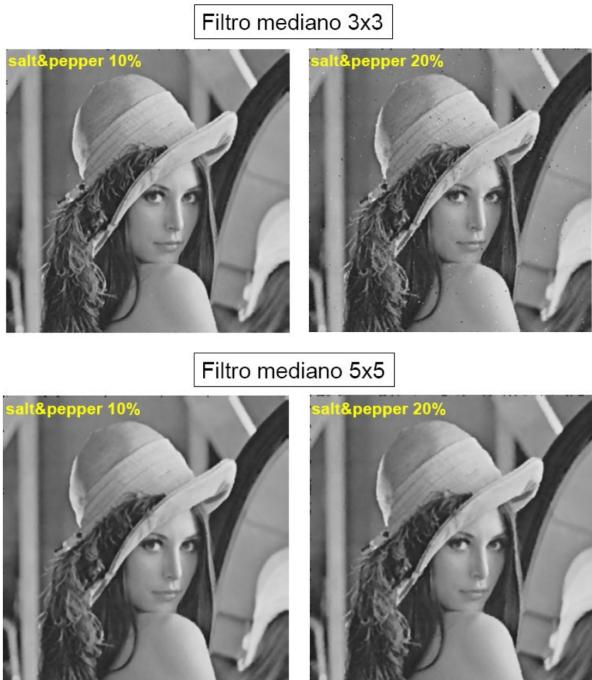
Filtro 5-box

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.



Rimozione del rumore Sale e Pepe



Quindi applicare 2 volte il filtro mediano preserva meglio il contenuto di partenza.

Note

- è più conveniente applicare più volte un filtro piccolo (per SALE e PEPE)
- mentre per poco rumore (in percentuale) uso kernel piccolo.

Differenze fra media e mediano

- *Perché i filtri mediani danno risultati migliori rispetto a quelli di media?*
 - Perchè i mediani non creano livelli di grigio.
- 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.

Non si può applicare con convoluzione con Outlier e Olimpico.

Con l'**Olimpico** si possono anche eliminare più massimi o minimi (cioè si parla di "*alpha-trimmer*" dove α è la percentuale di massimi/minimi da rimuovere)

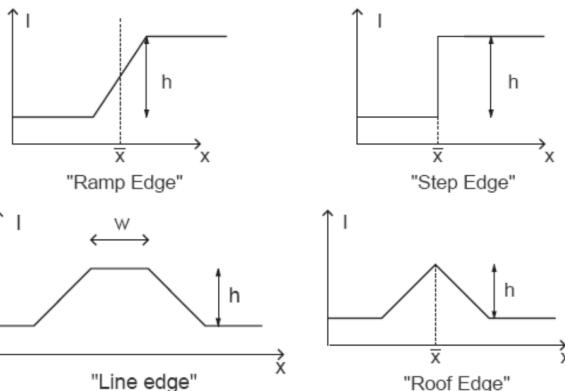
ESTRAZIONE DEI CONTORNI

Il contorno (*bordo*) rappresenta delle discontinuità geometriche o luminose. In particolar modo è una variazione più o meno repentina del colore. La differenza di colore è molto ampia. Si evidenziano i contorni con il processo di "estrazione di contorni".

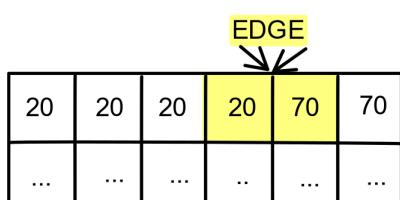
Esistono diversi modi di estrazione dei contorni (*edge detector*). Solitamente si usa convoluzione, quindi il kernel (di base), ma ne esistono altri che non usano convoluzione.

Com'è fatto un edge (contorno)? Come variano i valori in presenza di un edge?

E' come se si osservasse solo una linea di pixel come nelle immagini sottostanti:



-

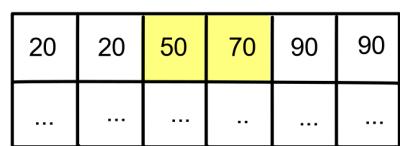


-

Questo è uno Step Edge nelle immagini grafiche.

Ramp Edge

Nelle immagini realistiche trovo variazioni più graduali e non così "improvvisi", (cioè si trovano più spesso edge di tipo **Ramp Edge**)



-

Questa situazione crea una variazione di valori più "curvata" (cioè più graduale e meno netta come nel caso precedente).

Roof Edge

20	20	50	70	20	20
...

- cioè qualcosa ha un certo andamento, sale fino ad un picco e poi scende fino a *ristabilizzarsi*.

Line Edge

- E' un Roof Edge dove l'intervallo non è più un punto ma un'ampiezza rappresentata da w .

20	20	50	50	20	20
...

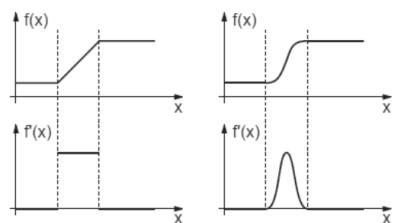
-

Per trovare un Edge uso la derivata prima che serve per descrivere l'andamento di una curva. Si fa la differenza fra il pixel che si sta considerando e il successivo Si presentano 2 casi.

- Dove c'è una **maggior variazione** allora sicuramente lì c'è un Edge e la derivata avrà un certo valore diverso da 0;
- Dove **non c'è variazione** la derivata = 0 quindi non c'è Edge perchè si ha una situazione di funzione costante.

Formalmente:

- Se ho un **segnalet monodimensionale** e calcolo la **derivata prima**, scopro che i lati sono i corrispondenti 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

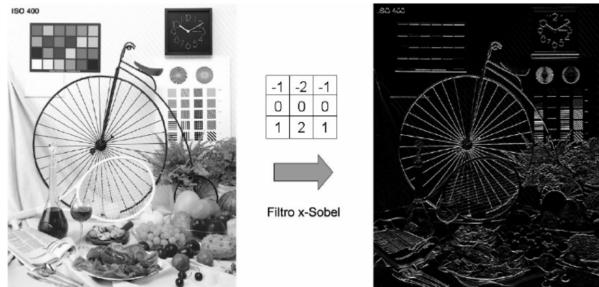
I 2 kernel che si usano per i **LATI ORIZZONTALI** sono $Sobel_x$ e $Prewitt_x$ e sono così fatti:

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

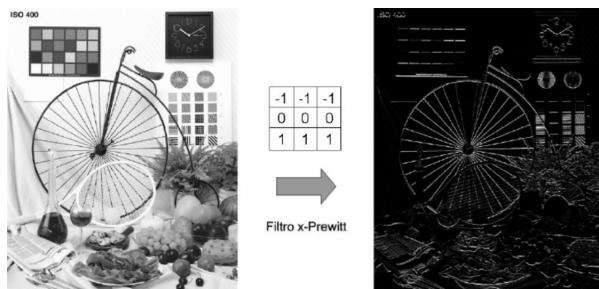
Questi kernel non calcolano la derivata prima esattamente ma sono basati su di essa.
Per gli edge verticali non vengono applicati questi due filtri perchè essi non li rilevano.

- il risultato di un edge detector non è un'immagine perchè può avere valori maggiore di 255 e minori di 0.

Sobel_x



Prewitt_x



- Più è chiaro l'edge e più è netto il bordo.

Si nota che i raggi verticali della ruota non vengono rilevati perchè, appunto, questi due filtri funzionano su edge orizzontali.

A differenza di Prewitt_x , Sobel_x rende un po' più netti gli edge rispetto a

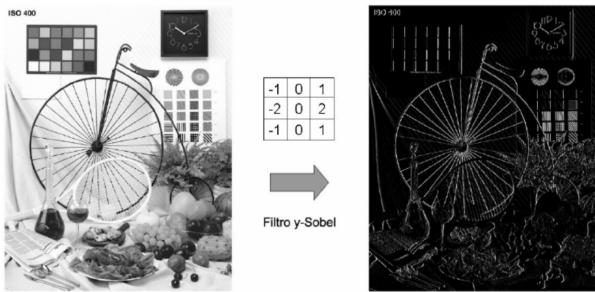
Kernel notevoli: lati verticali

I 2 kernel che si usano per i **LATI VERTICALI** sono Sobel_y e Prewitt_y . Essi assomigliano ai kernel usati per i lati orizzontali con l'unica differenza che consiste nella rotazione di 90° di tale kernel:

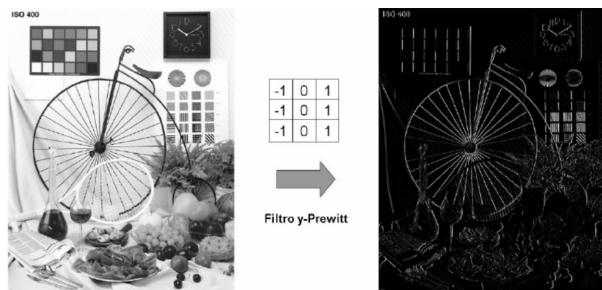
$$\text{Sobel}_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{Prewitt}_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Piuttosto che dire "ruotati i 90°" si può dire che le matrici dei kernel sono **trasposte**.

Sobel_y



Prewitt_y



| La domanda da farsi è: *Come si mettono insieme i valori di sobel x/y e prewitt x/y?*

Avendo le 2 componenti Sobel x e Sobel y, la risposta totale la trovo:

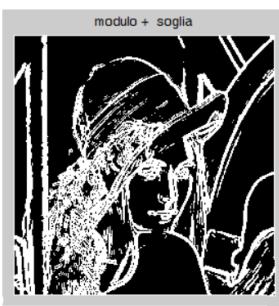
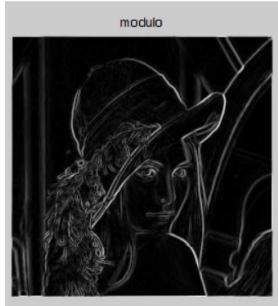
1. Sobel x fornisce una matrice con i lati orizzontali (e le componenti orizzontali dei lati obliqui) che hanno valori non nulli;
2. Sobel y fornisce una matrice con i lati verticali (e le componenti verticali dei lati obliqui) che hanno valori non nulli
3. Le due matrici possono essere combinate insieme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

$$\sqrt{Sobel_x^2 + Sobel_y^2}$$

Il risultato di tale operazione è detto **intensità dell'edge (o modulo)**

4. 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
- soglia = binarizzazione usata se si vuole classificare una matrice del tipo: "qui c'è un (bianco) edge e qui no (nero)"

| Ovviamente le stesse considerazioni valgono per Prewitt



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)
- con **strategie più “intelligenti”**, come l'algoritmo di **Canny** dal quale si ottiene un edge di dimensione 1 ed esso reagisce bene quando si è in presenza di **rumore impulsivo** o altro tipo di rumore. In particolare Canny differenzia il rumore dall'edge. Altre strategie possono essere: algoritmi **fuzzy**, tecniche di **backtracking** ecc..