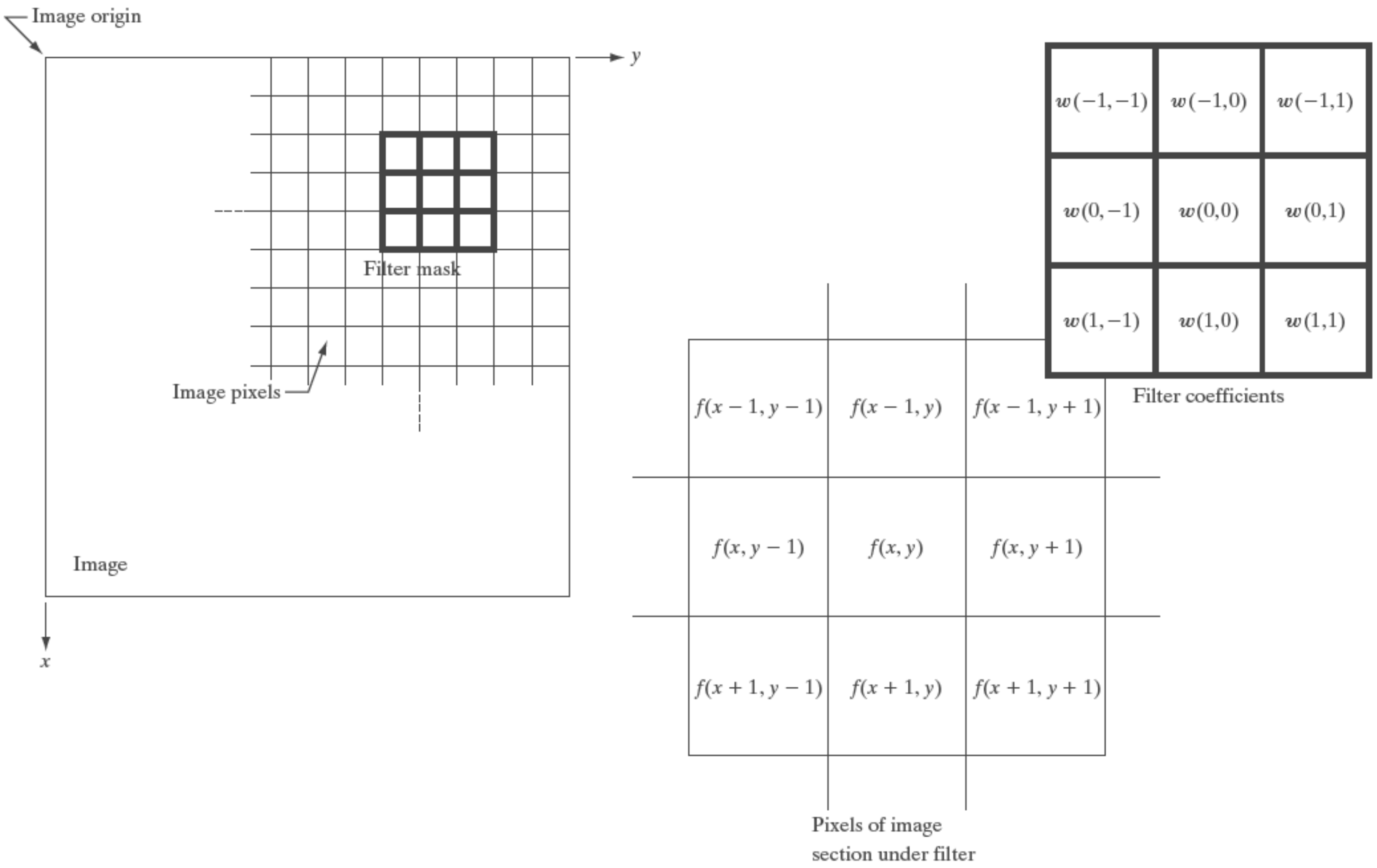
ELABORAZIONE DELLE IMMAGINI

FILTRAGGIO SPAZIALE

FILTRAGGIO SPAZIALE

- Le tecniche di **filtraggio spaziale** operano sui pixel di un'immagine prendendo in considerazione i valori di intensità in un intorno (**neighborhood**)
  - Per **ogni pixel** dell'**immagine originale**, è calcolata l'**intensità** del pixel corrispondente nell'**immagine filtrata**
  - La **regola di trasformazione** spesso è descritta da una **matrice**, chiamata **filtro** (maschera o kernel), della stessa dimensione dell'**intorno**
  - Se la regola di **trasformazione** è una funzione **lineare** delle intensità nell'intorno, la tecnica è chiamata **filtraggio lineare spaziale** (altrimenti, non lineare)

FILTRAGGIO LINEARE



FILTRAGGIO LINEARE

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

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

114				

FILTRAGGIO LINEARE

- Il **pixel** nell'**immagine filtrata**, g(x,y), è ottenuto come **combinazione lineare** dei pixel nell'immagine originale, f(), in un intorno di (x, y):

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

- La matrice peso, w(), è il filtro (maschera o kernel)
- Per comodità, spesso è usata una **matrice** con un numero **dispari** di righe 2a+1, e colonne, 2b+1

CORRELAZIONE E CONVOLUZIONE 2-D

- Correlazione

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

- Convoluzione

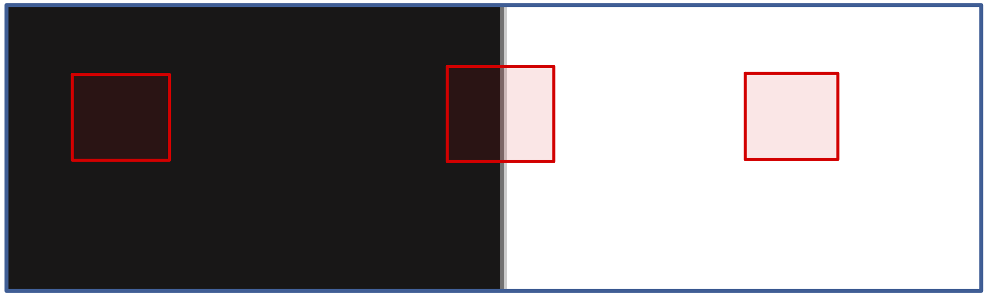
$$g(x, y) = w(x, y) * f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)\, f(x-s, y-t)$$

significa che sto valutando il simmetrico
(x fare la convoluzione devo pre ruotare l'immagine)

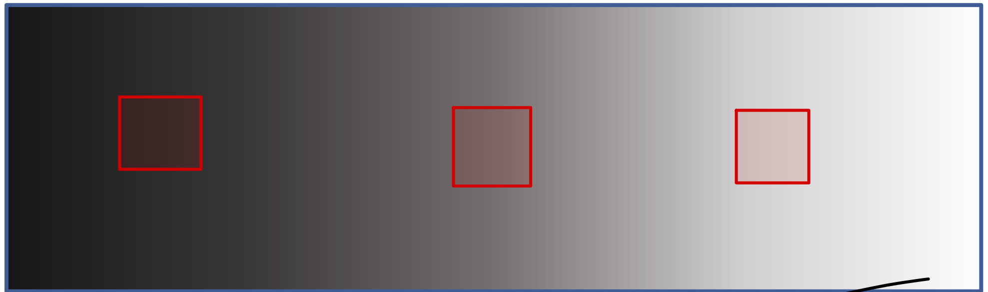
SHARPENING

- Il termine **sharpening** si riferisce alle tecniche adatte ad **evidenziare** le **transizioni** di intensità
- Nelle immagini, i bordi (**edge**) tra gli **oggetti** sono percepiti per il **cambio di intensità**: più nette sono le transizioni di intensità e più l'immagine è percepita in modo definito
- La transizione di intensità tra i pixel adiacenti è connessa alla **derivata dell'immagine** in quella posizione
- Definiremo gli **operatori di sharpening** attraverso l'uso delle **derivate**, in cui la qualità della **risposta** è **proporzionale** al grado di **variazione di intensità** nel punto in cui è applicato l'opratore.

FILTRAGGIO



*2.



la pendenza (rampa) è costante

lo voglio trovare dove ho delle variazioni, in modo da individuare gli edge/bordo

La risposta del mio filtro deve esse proporzionale alla intensità della mia variazione

dove individuo i bordi significa che in quel punto devo renderla + nitida

DERIVATA PRIMA DI UN'IMMAGINE

- Poiché l'immagine è una funzione discreta, la definizione classica di derivata non può essere applicata
- È necessario definire un operatore che soddisfi le principali **proprietà della derivata prima**:
  1. Uguale a zero dove l'intensità è costante (la derivata di una costante è 0)
  2. Diversa da zero per una transizione di intensità (cioè ho 2 livelli di grigio diversi)
  3. Costante sulle rampe in cui la transizione di intensità è costante es. 2 *
- L'operatore di derivazione naturale è la differenza tra l'intensità dei pixel vicini (**differenziazione spaziale**)

$$\frac{\partial f}{\partial x} = \underbrace{f(x+1) - f(x)}_{\text{differente finite}}$$

cioè al pixel successivo sottraggo il precedente

DERIVATA SECONDA DI UN'IMMAGINE

- Analogamente, l'operatore di **derivata seconda** può essere definito come:

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

=> DIFFERENZE CENTRALI

$$= f(x+1) - 2f(x) + f(x-1) \; *$$

SUCCESSIVO – 2 VOLTE QUELLO AL CENTRO + PRECEDENTE

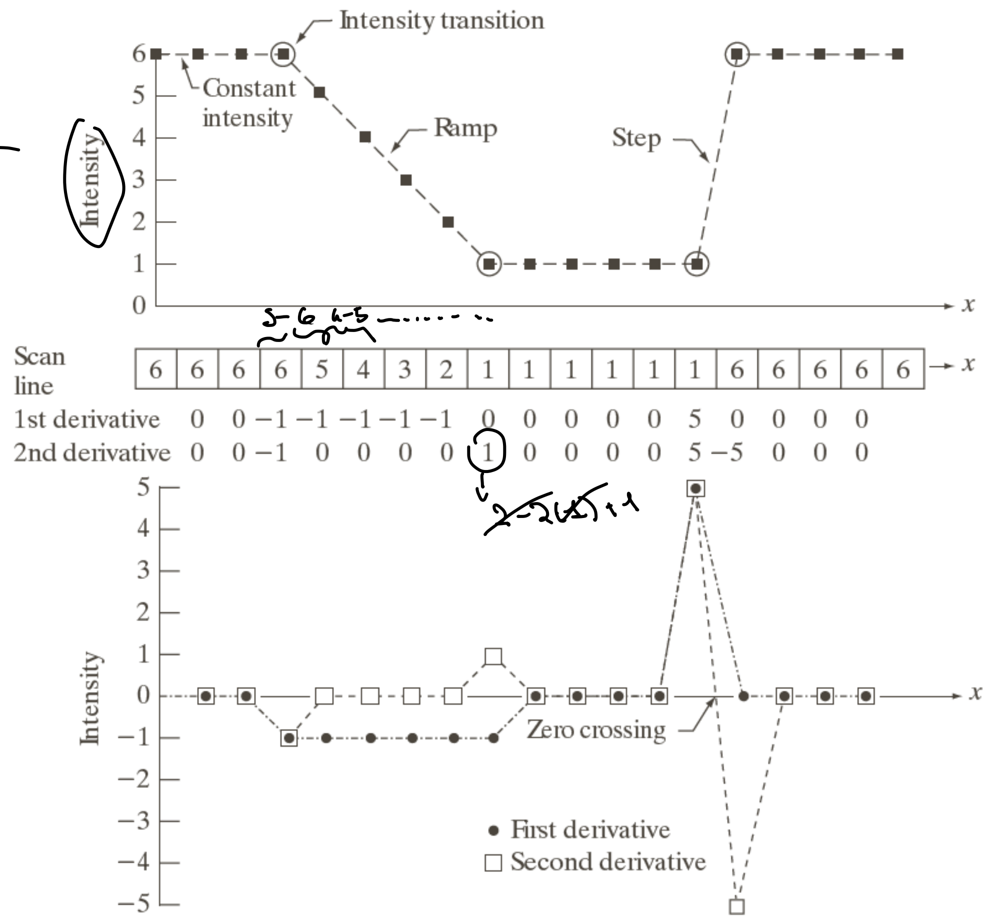
- Soddisfa le seguenti **proprietà**:
  1. È uguale a zero dove l'intensità è costante
  2. È diverso da zero all'inizio di un passo (o rampa) di intensità
  3. È uguale a zero sulle pendenze costanti delle rampe

- $\frac{\partial^2 f}{\partial^2 x}$ è definita usando i pixel precedente e successivo

* LA DERIVATA SECONDA NON LA POSSO APPLICARE NEL SUL PRIMO, NE SULL'ULTIMO, PERCHÉ NON AVREI I VICINI

ESEMPIO: DERIVATA DISCRETA

livelli di grigio



LAPLACIANO

- Il **Laplaciano** implementa una **derivata del secondo ordine 2D**
  - Definire una formulazione discreta della derivata seconda
  - Costruire una maschera filtro che la implementi
- In particolare vogliamo realizzare **filtri *isotropici*** la cui risposta è **indipendente** dalla **direzione** delle **disconitnuità** dell'immagine (invarianti per rotazione)

→ invariante alle rotazioni

- Il Laplaciano è l'operatore derivativo isotropico più semplice, definito per un'immagine f(x,y)

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

somma delle derivate parziali II rispetto a x e rispetto a y

∇ = nabla

FILTRO LAPLACIANO

- In un'immagine digitale, le derivate seconde rispetto a x e y sono calcolate come:

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

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

0	1	0
1	-4	1
0	1	0

↑ Filtro isotropico a 90°

- Quindi, il Laplaciano risulta:

$$\begin{aligned}\nabla^2 f(x,\, y) &= f(x+1,\, y) + f(x-1,\, y) + f(x,\, y+1) \\ &\quad + f(x,\, y-1) - 4f(x,\, y)\end{aligned}$$

- Può essere considerata anche la derivata lungo la diagonale:

$$\begin{aligned}\nabla^2 f(x,\, y) &+ f(x-1,\, y-1) + f(x+1,\, y+1) \\ &+ f(x-1,\, y+1) + f(x+1,\, y-1) - 4f(x,\, y)\end{aligned}$$

FILTRO LAPLACIANO

0	1	0
1	-4	1
0	1	0

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Filtro Laplaciano invariante alle rotazioni di 90°

1	1	1
1	-8	1
1	1	1

$$\nabla^2 f(x, y) + f(x-1, y-1) + f(x+1, y+1) + f(x-1, y+1) + f(x+1, y-1) - 4f(x, y)$$

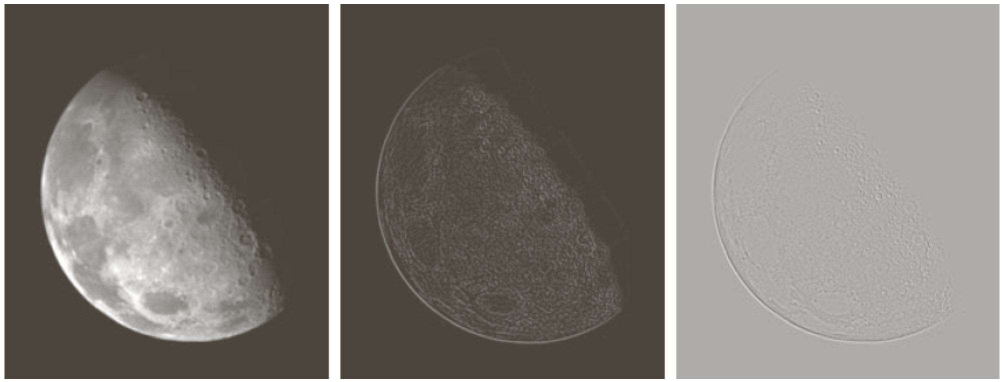
Filtro Laplaciano invariante alle rotazioni di 45°

se aumento i valori nelle diagonali devo aumentare rispettivamente dello stesso valore quello al centro

ESEMPIO: FILTRO LAPLACIANO

- Il Laplaciano spesso ha valori negativi → questi si appiattiscono a 0
- Per visualizzarlo, deve essere opportunamente scalato nell'intervallo di rappresentazione [0, L-1]

(a) (b) (c)



- (a) Immagine originale, (b) il suo Laplaciano, (c) il suo Laplaciano scalato in modo che 0 sia visualizzato come livello di grigio intermedio

ESEMPIO: FILTRO LAPLACIANO

- Sottraendo il Laplaciano (o una sua frazione) dall'immagine, l'altezza del gradino è aumentata

$$g = f + c\nabla^2 f, \ -1 \le c \le 0$$

(a) (b) (c)

- (a) Immagine originale (b) filtrato con Laplaciano 90° (c) filtrato con Laplaciano 45°

LAPLACIANO IN OPENCV

- In OpenCV la funzione che implementa il Laplaciano è

```
void cv::Laplacian(
  cv::InputArray  src,                     // Input image
  cv::OutputArray dst,                     // Result image
  int             ddepth,                  // Depth of output image (e.g., CV_8U)
  cv::Size        ksize     = 3,           // Kernel size
  double          scale     = 1,           // Scale applied before assignment to dst
  double          delta     = 0,           // Offset applied before assignment to dst
  int             borderType = cv::BORDER_DEFAULT  // Border extrapolation to use
);
```

UNSHARP MASKING

- E' un metodo di uso comune in grafica per rendere le immagini più nitide
- Consiste:
  1. Sfocare l'immagine originale (smoothing) $(f * h)$
  2. Ottenere una maschera come differenza tra l'immagine originale e quella sfocata
  3. Aggiungere la maschera all'immagine originale
- Il processo può essere formalizzato come:

$$g = f + k \cdot (f - f * h)$$

UNSHARP MASKING

$$g = f + k \cdot (f - f * h)$$

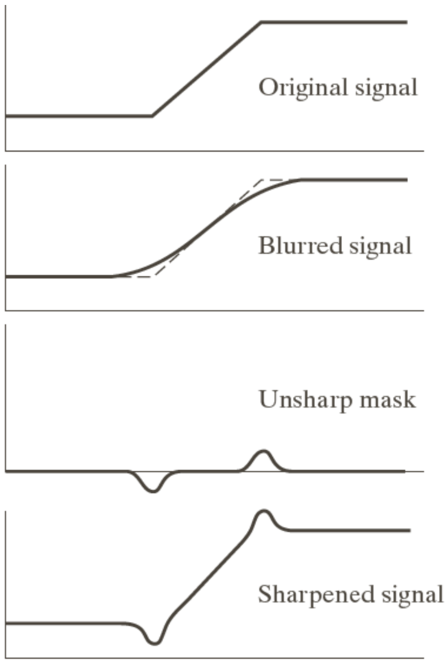


Immagine originale

Immagine sfocata

Maschera di unsharping

Immagine unsharped (k<=1)

Immagine highboosted (k>1)

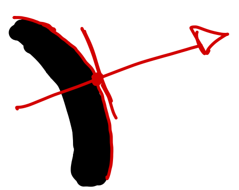
GRADIENTE

- La **derivata prima** viene implementata con un'approssimazione del **gradiente**
- Il gradiente è un **vettore** formato dalle sue **derivate parziali**

$$\nabla f \equiv \mathrm{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Sovrappongo le immagini ma non sommo le 2 componenti come le elevano e quindi preservo le 2 intensità

- Il vettore **gradiente punta** verso la **direzione** di **massima variazione** = in che direzione si sviluppa l'edge



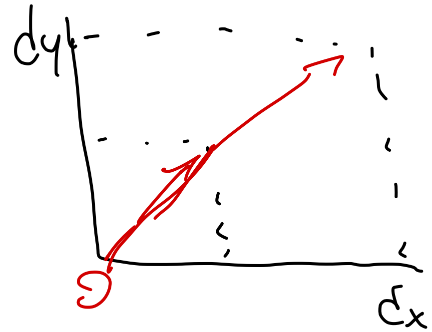
- La **magnitudo del gradiente**, M(x, y) è un'immagine delle stesse dimensioni di f()

(è la norma del vettore)

$$M(x, y) = \mathrm{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

$$M(x, y) \approx |g_x| + |g_y|$$

mi serve x capire l'intensità della variazione



OPERATORI DI DERIVAZIONE

- Definizioni base:

$$g_x(x,\, y) = f(x+1,\, y) - f(x,\, y)$$

$$g_y(x,\, y) = f(x,\, y+1) - f(x,\, y)$$

g_x:

-1	1

g_y:

-1
1

- Operatori di Roberts:

$$g_x(x,\, y) = f(x+1,\, y+1) - f(x,\, y)$$

$$g_y(x,\, y) = f(x,\, y+1) - f(x-1,\, y)$$

g_x:

-1	0
0	1

g_y:

0	-1
1	0

OPERATORI DERIVATIVI

- Operatori di Sobel:

Utile x applicare convoluzione e correlazione

$$
\begin{aligned}
g_x(x, y) = & -f(x-1, y-1) - 2f(x-1, y) \\
& - f(x-1, y+1) + f(x+1, y-1) \\
& + 2f(x+1, y) + f(x+1, y+1)
\end{aligned}
$$

$$
\begin{aligned}
g_y(x, y) = & -f(x-1, y-1) - 2f(x, y-1) \\
& - f(x+1, y-1) + f(x-1, y+1) \\
& + 2f(x, y+1) + f(x+1, y+1)
\end{aligned}
$$

g_x:

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

g_y:

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

OPERATORI DERIVATIVI

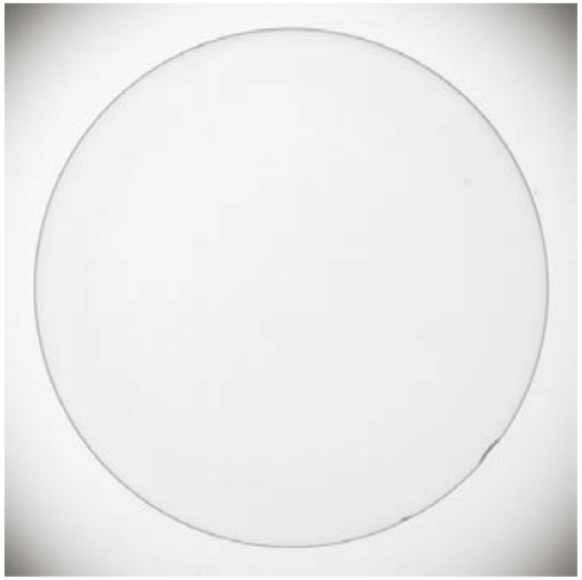
- Operatori di Prewitt:

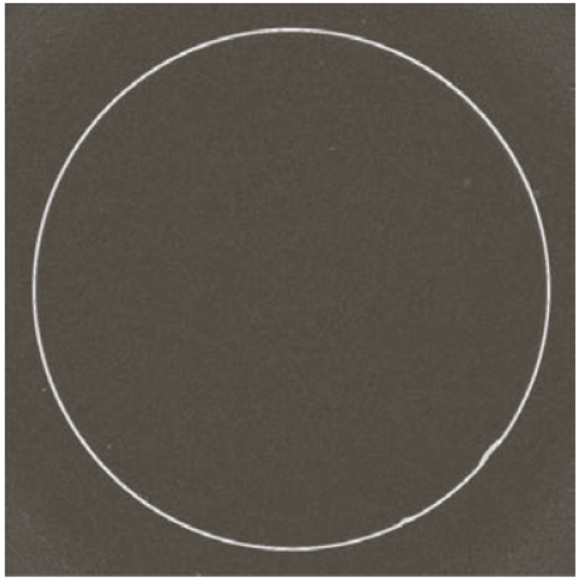
-1	-1	-1
0	0	0
1	1	1

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

ESEMPIO: APPLICAZIONE BASATA SU GRADIENTE

- Il filtraggio di **Sobel** riduce la visibilità di quelle regioni in cui l'intensità cambia lentamente, permettendo di **evidenziare i dettagli** (rendendo l'individuazione dei difetti più semplice per un'elaborazione automatica)





SOBEL IN OPENCV

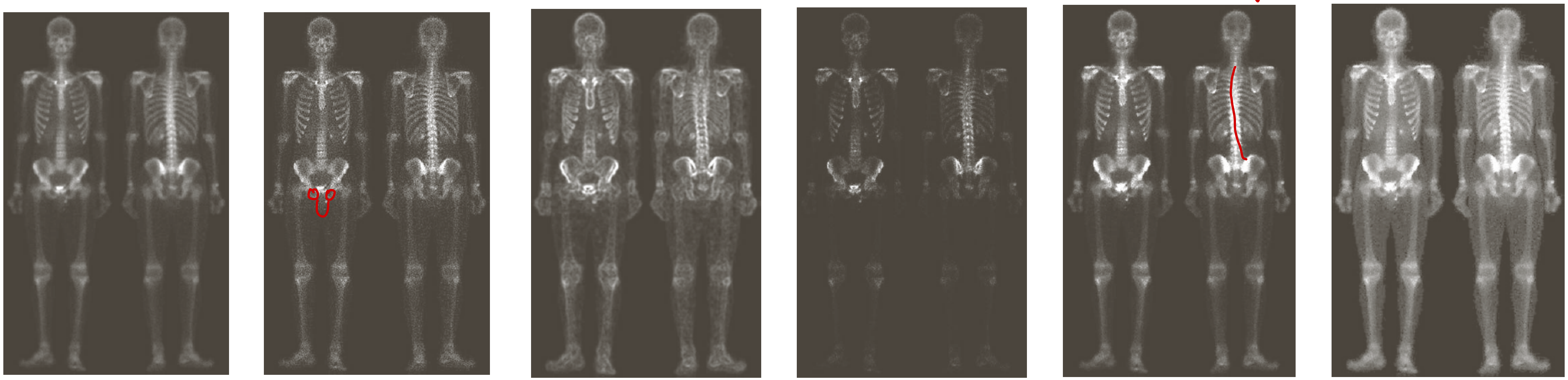
- In OpenCV la funzione che implementa il filtro di Sobel è

```
void cv::Sobel(
  cv::InputArray  src,                     // Input image
  cv::OutputArray dst,                     // Result image
  int             ddepth,                  // Pixel depth of output (e.g., CV_8U)
  int             xorder,                  // order of corresponding derivative in x
  int             yorder,                  // order of corresponding derivative in y
  cv::Size        ksize      = 3,          // Kernel size
  double          scale      = 1,          // Scale (applied before assignment)
  double          delta      = 0,          // Offset (applied before assignment)
  int             borderType = cv::BORDER_DEFAULT  // Border extrapolation
);
```

- xorder e yorder servono a specificare l'ordine della derivata lungo la direzione x e y

METODI COMBINATI

scoliosi



(a) (b) (c) (d) (e) (f)

è maschio

ESERCIZI

- Implementare il Laplaciano con kernel isotropico a 45° e 90° utilizzando la funzione di correlazione/convoluzione (o filter2D())
  - Per normalizzare i livelli di grigio è possibile usare la funzione normalize()
  - **normalize(src, dst, 0, 255, NORM_MINMAX, CV_8U);**
- Implementare il filtro di Sobel (g_x e g_y) utilizzando la funzione di correlazione/convoluzione (o filter2D())
  - Calcolare la risposta di entrambi i filtri
  - Calcolare la magnitudo del gradiente (entrambe le formulazioni)
- Utilizzare la risposta ottenuta per effettuare lo sharpening di un'immagine

OPERAZIONI UTILI

- dst = src1 +/- src2
- dst = src * k
- dst = abs(src)
- pow(src,2,dst)
- sqrt(src,dst)