Low Level Vision: Edge

Prof. Sebastiano Battiato

Caratteristiche di immagini

In Visione Computazionale il termine *feature di immagini* si può riferire a due entità possibili:

- Una proprietà globale di un'immagine o una regione (es. il livello di grigio medio, la distribuzione di colore, ...);
- Una parte dell'immagine con alcune proprietà speciali (es. una linea, un'ellisse,...)

Le caratteristiche o proprietà di cui devono godere tali *features*, per poter essere utilizzate in maniera robusta dipendono dal contesto applicativo

Feature locali

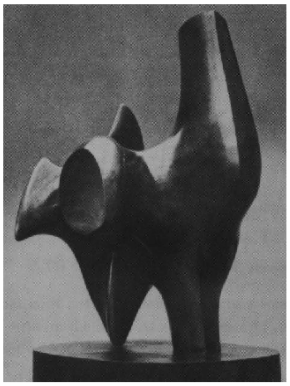
Un passo importante dell'elaborazione visiva consiste nell'identificare *feature locali* che siano utili ad interpretare l'immagine.

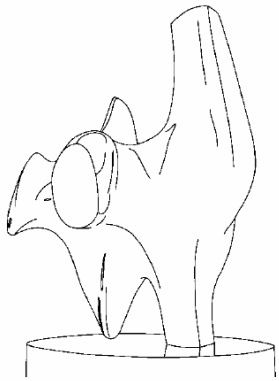
Le feature locali sono parti dell'immagine "facilmente" rilevabili, che possono corrispondere o meno a parti della scena.

Ci occuperemo nell'ordine di: Edge, linee (Trasformata di Hough), Corner, SIFT e Texture

Edge Detection

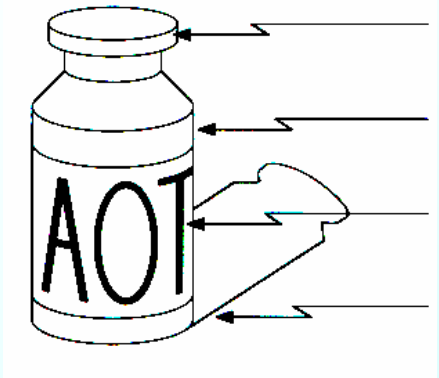
L'estrazione dei contorni (**edge**) è sicuramente uno degli argomenti che hanno ricevuto più attenzione nella letteratura sull'image processing. Il contorno di un oggetto rappresenta infatti la separazione tra l'oggetto e lo sfondo o tra l'oggetto ed altri oggetti, per cui la sua estrazione è molto spesso il primo passo verso l'individuazione dell'oggetto.





Edge su Lena

Origine degli edge



surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

Edge Detection

Un **edge** si presenta in una immagine come il confine tra due **regioni** caratterizzate da proprietà dei livelli di grigio in qualche modo distinguibili.

Ipotizziamo inizialmente che le regioni in questione siano sufficientemente omogenee, di modo che la determinazione della transizione tra le due regioni sia possibile sulla sola base della discontinuità dei valori di grigio.

Le prime tecniche di ***edge detection*** che analizziamo sono basate sull'applicazione di un operatore locale di derivata.

Edge Detection: Teoria

Consideriamo il comportamento delle derivate in corrispondenza a particolari andamenti del livello di grigio f(x): andamento costante (segmento piatto nel caso 1-D) e discontinuità, sia a gradino che a rampa, che possono modellare ***punti isolati, linee*** ed ***edge*** in una immagine.

Le derivate di una funzione digitale possono essere definite in termini delle differenze tra i valori assunti dalla funzione in punti vicini. Qualunque sia il modo con cui si utilizzano queste differenze, devono essere soddisfatti i seguenti requisiti. La ***derivata prima*** deve essere

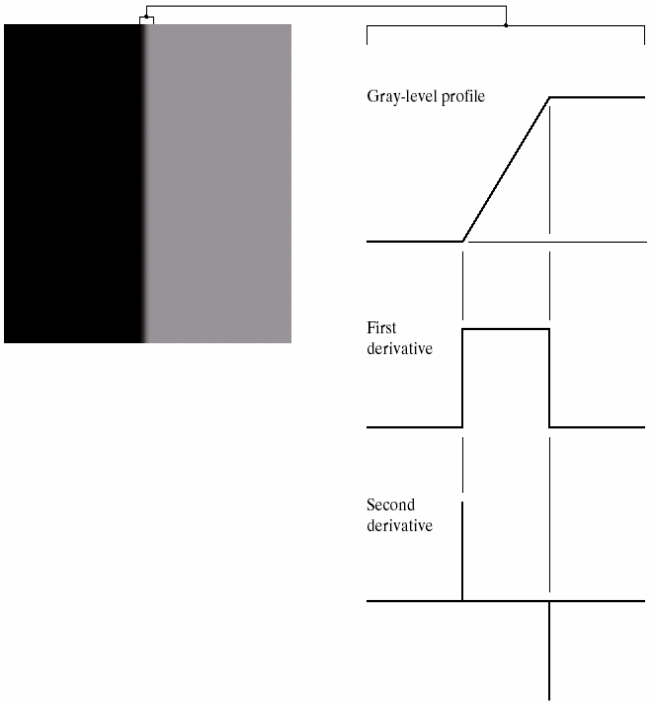
- nulla nelle zone piatte
- non nulla in presenza di una discontinuità
- non nulla lungo una rampa

Edge Detection: Teoria

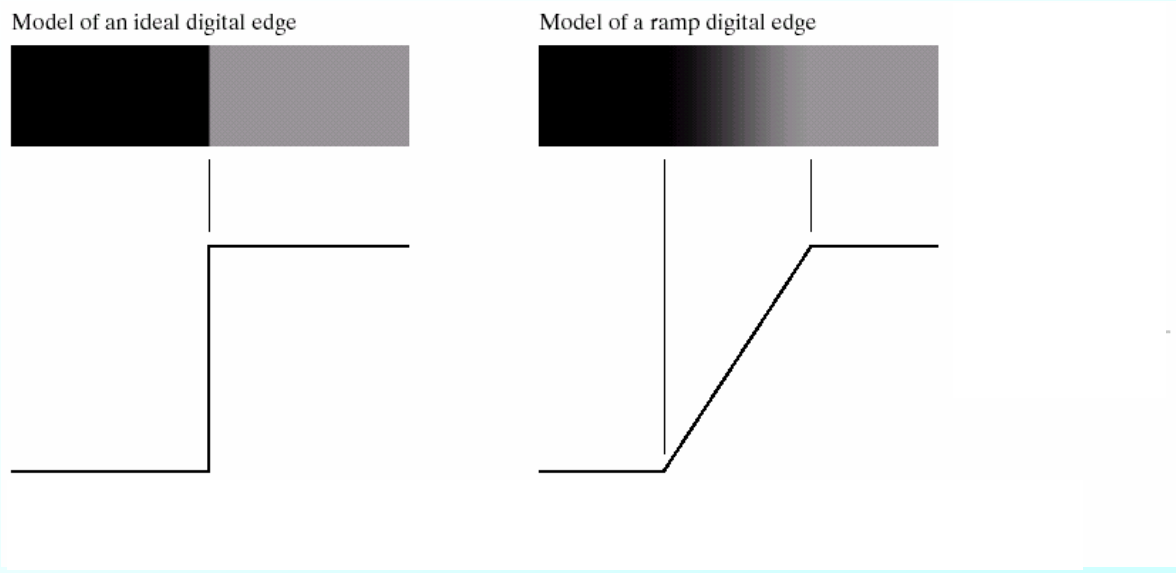
La ***derivata seconda*** deve essere

- nulla nelle zone piatte
- non nulla all'inizio e alla fine di una discontinuità
- nulla lungo una rampa di pendenza costante

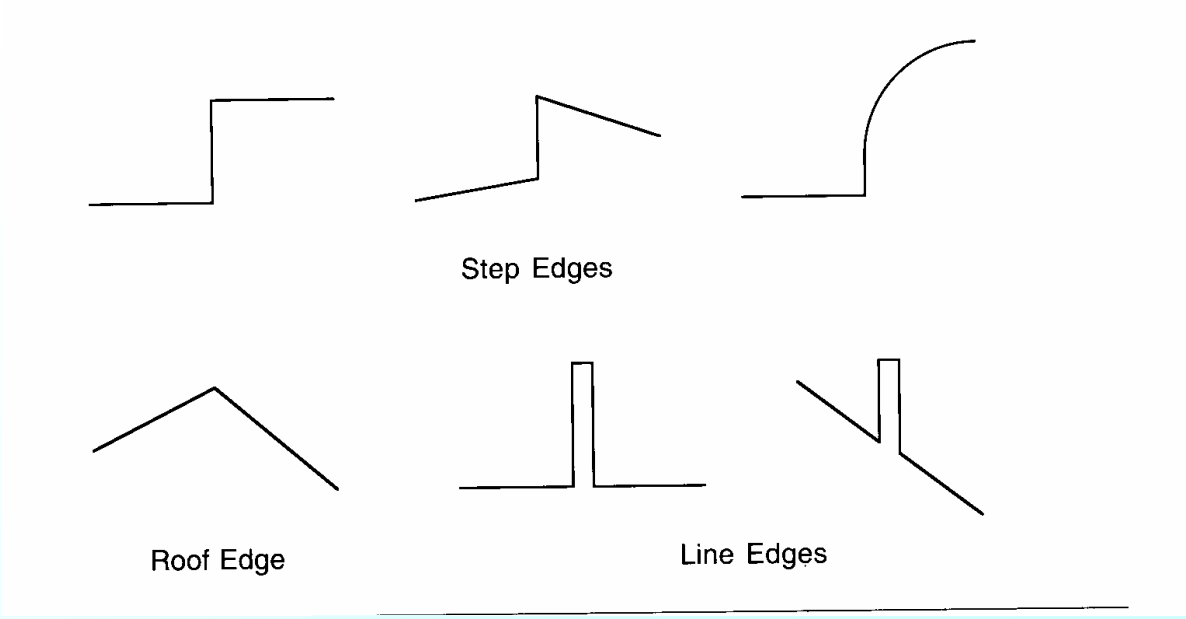
Poiché trattiamo con quantità digitali di valore finito, anche la massima variazione possibile di grigio è finita, e la minima distanza entro la quale può prodursi la variazione è tra due pixel adiacenti.



Edge Ideali



Edge in pratica



+ RUMORE!!!

Filtri Derivativi

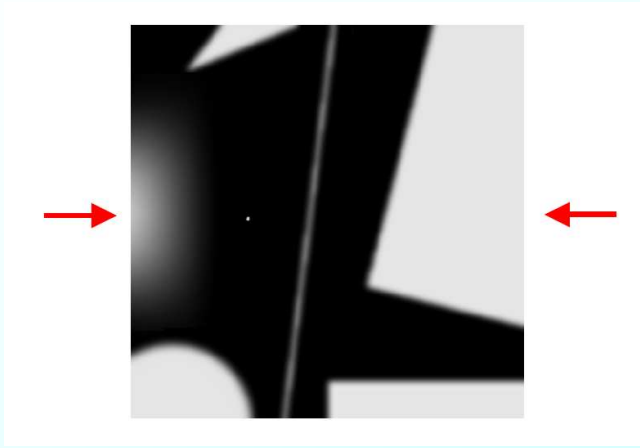
Una definizione usuale di derivata prima di una *f(x)* discreta è semplicemente la differenza:

$$f'(x)=f(x+1)-f(x)$$

Una definizione di derivata seconda di una *f(x)* discreta è la differenza:

$$f''(x)= f(x+1)+f(x-1)-2f(x)$$

Filtri Derivativi



Filtri Derivativi: Esempio

f	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	--
f'	--	-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	--	--
f''	--	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	--	--

Dai valori riportati, si può osservare che la derivata prima è diversa da 0 lungo tutta la rampa, mentre la derivata seconda è diversa da 0 solo all'inizio e alla fine della rampa

Filtri derivativi

Poiché i contorni nelle immagini digitali sono di questo tipo (più o meno lenti) a causa dell'effetto di blurring indotto dal campionamento, si può concludere che la derivata prima produce ***edge*** piuttosto spessi, mentre la derivata seconda dà luogo a ***edge*** più sottili ma "doppi".

In corrispondenza al punto isolato, la risposta della derivata seconda è più intensa, come è ovvio, dato il carattere più aggressivo della derivata seconda rispetto alla derivata prima nella evidenziazione delle brusche variazioni del livello di grigio

Quindi la derivata seconda è in grado di accentuare i dettagli fini (compreso il rumore) più della derivata prima. Anche la linea sottile è un dettaglio fine, quindi si giustifica un comportamento delle due derivate analogo al precedente.

Filtri Derivativi

La risposta delle due derivate è di intensità simile nel caso dell'edge, ma si può ancora notare il "doppio" ***edge*** rilevato dalla derivata seconda. Questa caratteristica viene sfruttata negli algoritmi di estrazione dei contorni (***edge detection***).

Si noti però che in caso di edge di ampiezza uguale a quello preso ad esempio (7), ma causato da una variazione di livelli di grigio differenti (per esempio 3 e 10), la risposta della derivata prima avrebbe la stessa intensità, mentre quella della derivata seconda sarebbe molto meno ampia.

Riassumendo, in generale:

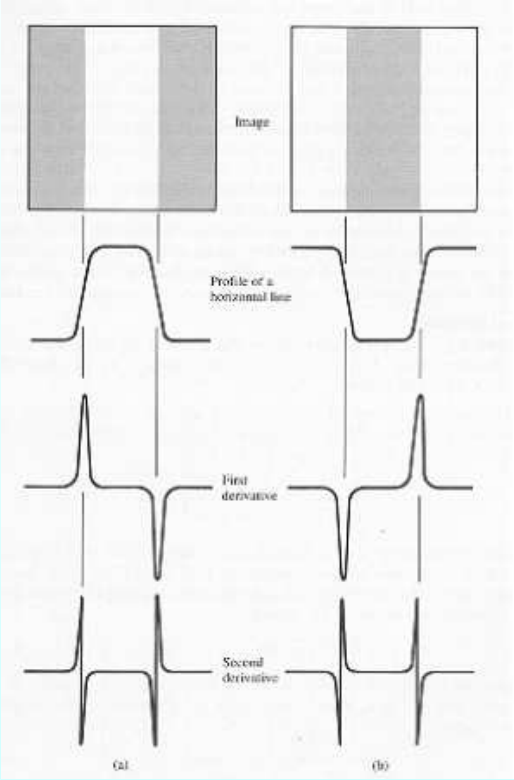
- **le derivate prime danno luogo a contorni più spessi**
- **le derivate seconde hanno una risposta più forte ai dettagli fini**
- **le derivate prime hanno una risposta più forte agli edge ripidi**
- **le derivate seconde producono una risposta doppia agli edge**

Filtri derivativi:discussioni

Il fatto che la derivata prima e la derivata seconda del profilo siano significativamente diverse da 0 soltanto in corrispondenza alle transizioni costituisce la motivazione dell'uso di operatori derivativi per l'estrazione dei contorni. Ci sono però delle differenze:

La derivata prima è **positiva** in corrispondenza ad una transizione scuro-chiaro, **negativa** in corrispondenza ad una transizione chiaro-scuro, **nulla** nelle zone a livello di grigio costante.

La derivata seconda è **positiva** in prossimità di un contorno, dalla parte scura del contorno stesso, **negativa** dalla parte chiara del contorno, **nulla** nelle zone a livello di grigio costante, ed esibisce un passaggio per lo zero o ***zero crossing*** esattamente in corrispondenza alle transizioni (notare in figura la rappresentazione "continua" delle derivate)



Filtri Derivativi:discussioni

Riassumendo, il valore della derivata prima può essere utilizzato per determinare la presenza di contorni in una immagine

Gli ***zero crossing*** della derivata seconda ne possono consentire la precisa localizzazione.

Il **segno** della derivata seconda permette di stabilire l'appartenenza di un pixel al versante scuro o al versante chiaro di un contorno.

L'applicazione di questi concetti necessita tuttavia di alcune cautele, essenzialmente legate alla natura digitale delle immagini.

Filtri derivativi: il gradiente

Una comune implementazione della derivata prima nell'ambito dell'*image processing* è costituita dal modulo del gradiente. Ricordiamo che data una funzione *f(x,y)*, il gradiente di *f* in (*x*,*y*) è il vettore:

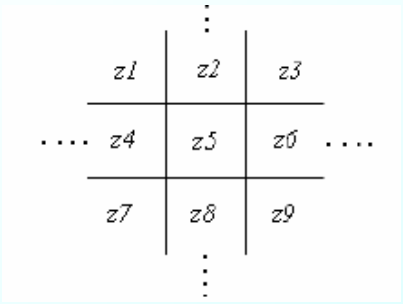
$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \approx |G_x| + |G_y|$$

La direzione $\alpha(x,y)=\tan^{-1}(G_y/G_x)$. L'approssimazione mediante somma dei valori assoluti delle componenti rende computazionalmente meno oneroso il calcolo del gradiente, ma in generale fa perdere la proprietà di isotropia.

Il Gradiente

Tuttavia, le maschere di gradiente più usuali manifestano invarianza rotazionale per step di 90°. La maniera più semplice di generare una approssimazione discreta del gradiente prende in considerazione la differenza mobile (*running difference*) dei pixel nelle due direzioni. Per esempio:



Le componenti del gradiente nel pixel z_5 sono: $G_x=z_8-z_5$ e $G_y=z_6-z_5$, per cui

$$\nabla f=[(z_8-z_5)^2+(z_6-z_5)^2]^{1/2} \approx |z_8-z_5|+|z_6-z_5|$$

In alternativa, si possono usare le differenze incrociate:

$$\nabla f=[(z_9-z_5)^2+(z_8-z_6)^2]^{1/2} \approx |z_9-z_5|+|z_8-z_6|$$

Filtri basati sul gradiente

L'implementazione di queste equazioni può essere effettuata usando le seguenti maschere 2x2 dette anche operatori di Roberts:

-1	0			0	-1
0	1			1	0

I valori assoluti delle risposte delle due maschere vengono sommati per determinare ∇f. Più comune è tuttavia l'impiego di maschere 3x3, che rendono più semplici le operazioni di filtraggio:

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

Filtri basati sul Gradiente

Gli operatori di gradiente sicuramente più diffusi, in grado di effettuare simultaneamente la **differenziazione** lungo una direzione e una **media spaziale** lungo la direzione ortogonale, che riduce la sensibilità al rumore, sono:

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

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$
$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

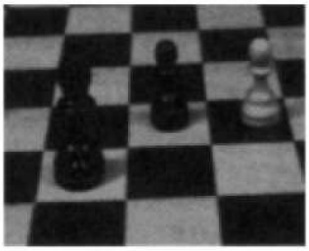
Prewitt

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

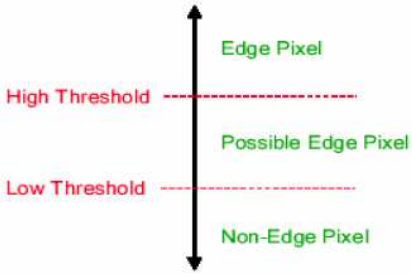
$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

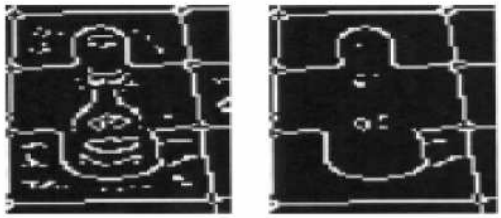
Sobel

Edge Detection in pratica



Original

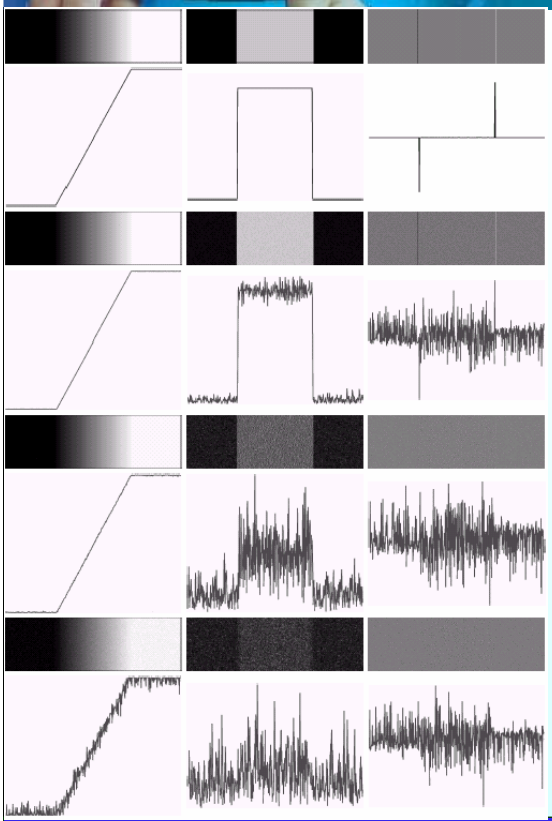




Low Threshold

High Threshold

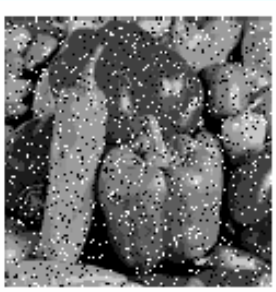
Edge Detection e Rumore



All'aumentare della potenza del rumore aggiunto (in questo caso rumore Gaussiano a media nulla con sigma= 0.0, 0.1, 1.0, 10.0) i risultati dei filtri derivativi sono via via meno affidabili.

Ancora Noise

Gli operatori di gradiente risultano quindi poco efficaci in presenza di rumore. Per esempio, applicando gli operatori di Sobel ad una versione rumorosa dell'immagine di test si ottiene:



Il problema può essere alleviato estendendo l'area di calcolo del gradiente, il che consente una migliore azione di media (Farlo per esercizio). Più in generale, possono essere presi in considerazione degli operatori composti, nei quali una operazione di **smoothing** per la riduzione del rumore è compiuta prima dell'operazione di differenziazione.

Edge in pratica

Gradiente smussato

In generale, la risposta impulsiva di un operatore di gradiente composto è definita come $G_C(x,y)=G(x,y)*M(x,y)$

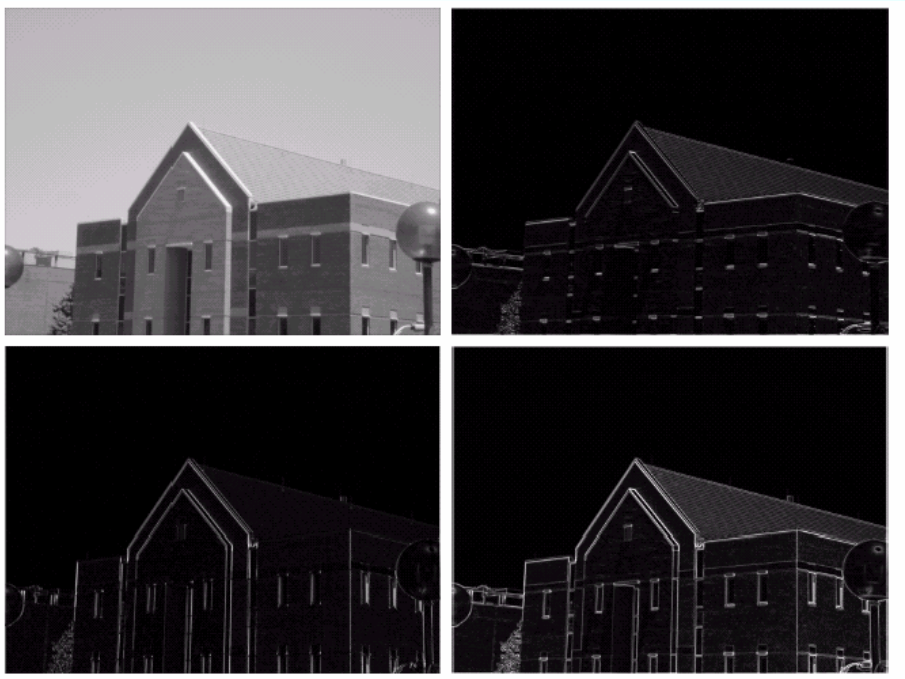
dove $G(x,y)*M(x,y)$ sono, rispettivamente, la risposta impulsiva di uno degli operatori di gradiente già visti e la risposta impulsiva di un filtro passa-basso

- Per esempio, considerando rispettivamente il gradiente di riga 5x5 di Prewitt e il filtro 5x5 di media mobile, si ottiene:

$$G_{Cy} = \frac{1}{18} \times$$

1	1	0	-1	-1
2	2	0	-2	-2
3	3	0	-3	-3
2	2	0	-2	-2
1	1	0	-1	-1

Gradiente smussato:esempio



Operatore DroG

Un esempio molto noto di operatore di gradiente composto è la **Derivata della Gaussiana (DroG)**, nel quale l'operazione di smoothing utilizza una funzione gaussiana.

La **gaussiana** è una funzione a simmetria rotazionale la cui equazione nel caso 2-D continuo è la seguente:

$$h(x,y) = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} = e^{\left(-\frac{r^2}{2\sigma^2}\right)}$$

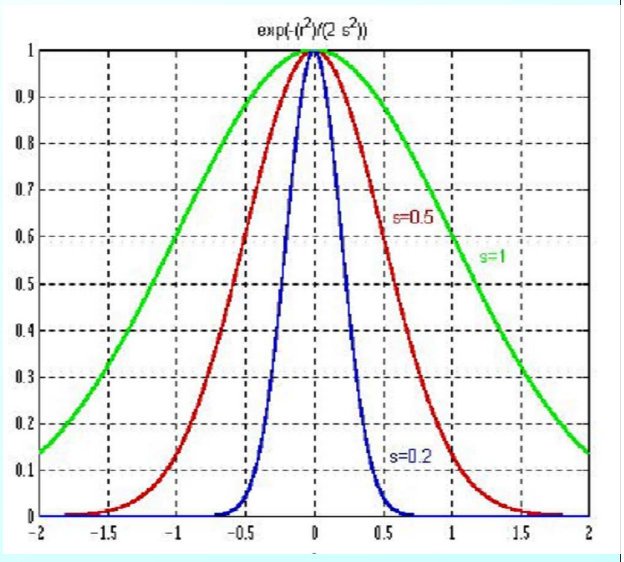
Il valore di σ determina l'apertura della Gaussiana (ovvero la deviazione standard ossia ossia il valore di *r* per il quale *h* si riduce a 1/√e del massimo), che aumenta al crescere di σ.

Operatore DroG

In figura una funzione gaussiana in cui *h* è rappresentata, per diversi valori di σ, al variare di r, quindi in un piano passante per il suo asse di simmetria.

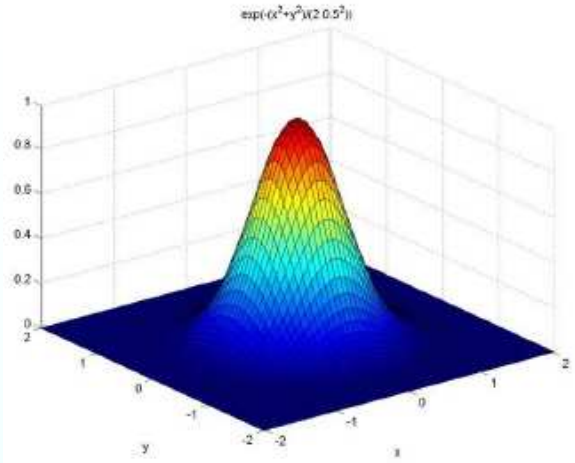
Dalla risposta impulsiva è possibile ricavare i coefficienti del relativo kernel di convoluzione attraverso un opportuno campionamento della funzione continua, a partire dalla origine, che rappresenta il punto di applicazione della maschera.

Nel caso della gaussiana il ruolo di σ è determinante nella definizione dei pesi della maschera: se l'apertura è maggiore, l'azione di filtraggio può riguardare un intorno più ampio del punto centrale. A tal fine, σ è normalmente espresso in pixel.



Computer Vision A.A. 2008/2009 – Prof. Sebastiano Battiato

Gaussiana con Kernel 3x3, 5x5 (sigma = 0.5)



$$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

$$\begin{bmatrix} 0.0000 & 0.0000 & 0.0002 & 0.0000 & 0.0000 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0002 & 0.0837 & 0.6187 & 0.0837 & 0.0002 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0000 & 0.0000 & 0.0002 & 0.0000 & 0.0000 \end{bmatrix}$$

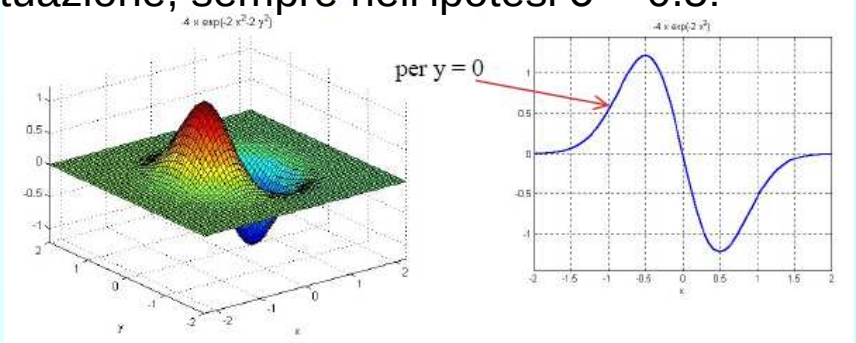
Per quanto riguarda l'operatore **DroG**, le risposte impulsive lungo le due direzioni, che lo caratterizzano completamente, si possono determinare effettuando la convoluzione delle due componenti del gradiente, nella formulazione preferita, con la maschera precedente che caratterizza la gaussiana.

Operatore DroG

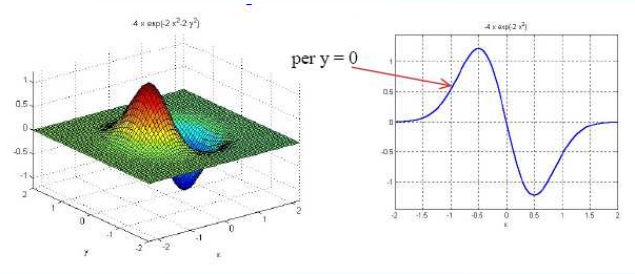
In alternativa, si può procedere al calcolo delle derivate della gaussiana rispetto a x e a y, che rappresentano le componenti dell'operatore **DroG** nel caso continuo:

$$\frac{\partial h}{\partial x} = -\frac{x}{\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \qquad \frac{\partial h}{\partial y} = -\frac{y}{\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

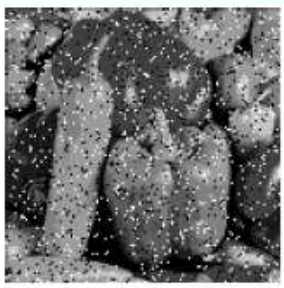
Le corrispondenti maschere si possono ottenere campionando le due funzioni. Per esempio, per la componente lungo x si ha la seguente situazione, sempre nell'ipotesi σ = 0.5:



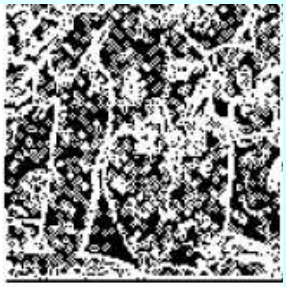
Operatore DroG



$$\begin{bmatrix} 0.1129 & 0.7733 & 0.1129 \\ 0 & 0 & 0 \\ -0.1129 & -0.7733 & -0.1129 \end{bmatrix}$$



Sobel



DroG

Operatori basati sul Laplaciano

La determinazione degli ***zero crossing*** della derivata seconda della *f*(*x*,*y*) localizza con precisione i contorni dell'immagine, mentre il segno della derivata seconda permette di stabilire l'appartenenza di un pixel al versante scuro o al versante chiaro di un contorno.

Un modo molto comune di effettuare le operazioni di derivata seconda di una *f*(*x*,*y*) in un punto è quello di calcolare il ***laplaciano*** in quel punto. Ricordiamo che data una funzione *f*(*x*,*y*), il laplaciano di *f* in (*x*,*y*) è definito come:

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

Operatori basati sul Laplaciano

Il modo più semplice di approssimare il laplaciano nel caso discreto consiste nel calcolo delle differenze delle derivate prime lungo i due assi:

$$L(x,y)=[f(x,y)-f(x-1,y)]-[f(x+1,y)-f(x,y)]+[f(x,y)-f(x,y-1)]-[f(x,y+1)-f(x,y)]$$

Pertanto

$$L(x,y)=4\,f(x,y)-f(x-1,y)-f(x+1,y)-f(x,y-1)-f(x,y+1)$$

Il Laplaciano si può quindi implementare come un filtro la cui risposta impulsiva è:

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

Operatori basati sul Laplaciano

La versione normalizzata prevede un fattore moltiplicativo pari a ¼

Una versione (normalizzate) del laplaciano riferito agli 8-vicini, quindi con le differenze delle derivate prime mediate su tre righe e tre colonne, è:

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

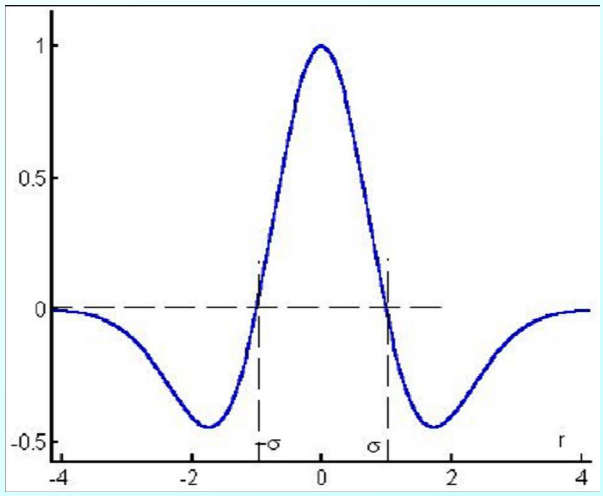
In questo caso il fattore moltiplicativo è pari a 1/8.

Il laplaciano è eccessivamente sensibile al rumore (in quanto operatore di derivata seconda) ed è incapace di rilevare la direzione del contorno (in quanto entità scalare).

Per tali motivi, il laplaciano è raramente usato da solo per l'edge detection.

Laplacian of Gaussian

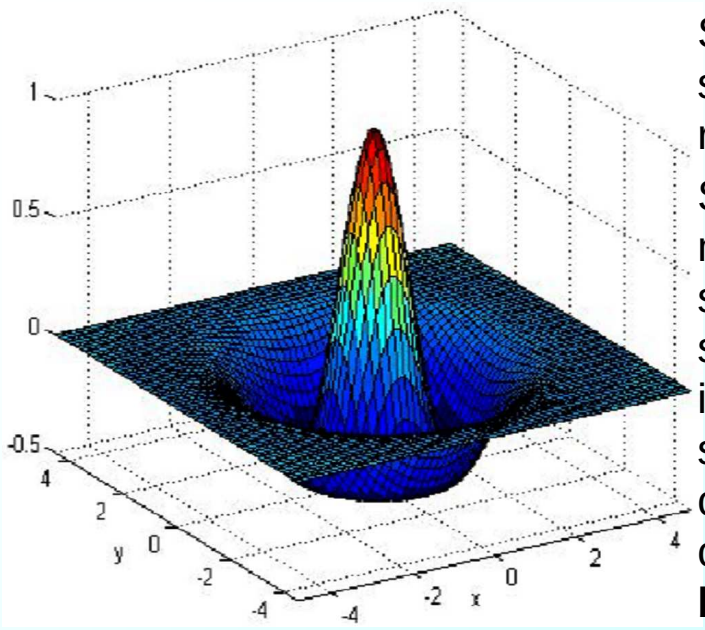
Di Marr e Hildreth è stata l'idea di utilizzare il laplaciano in connessione con un filtro di *smoothing*, ancora una volta una gaussiana, realizzando un operatore detto **Laplaciano della Gaussiana** o ***LoG*** (in figura una sezione trasversale ottenuta per σ=1)



$$h(x,y) = e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} = e^{\left(-\frac{r^2}{2\sigma^2}\right)}$$

$$\nabla^2 h = -\frac{r^2-\sigma^2}{2\sigma^2} e^{\left(-\frac{r^2}{2\sigma^2}\right)}$$

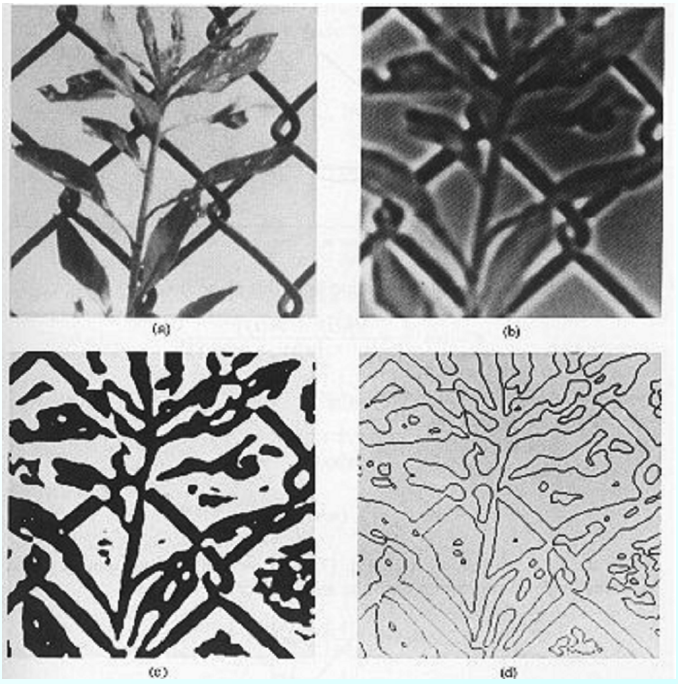
LoG in 2D



Si tratta quindi di una funzione a simmetria circolare, con ZC per r=±σ

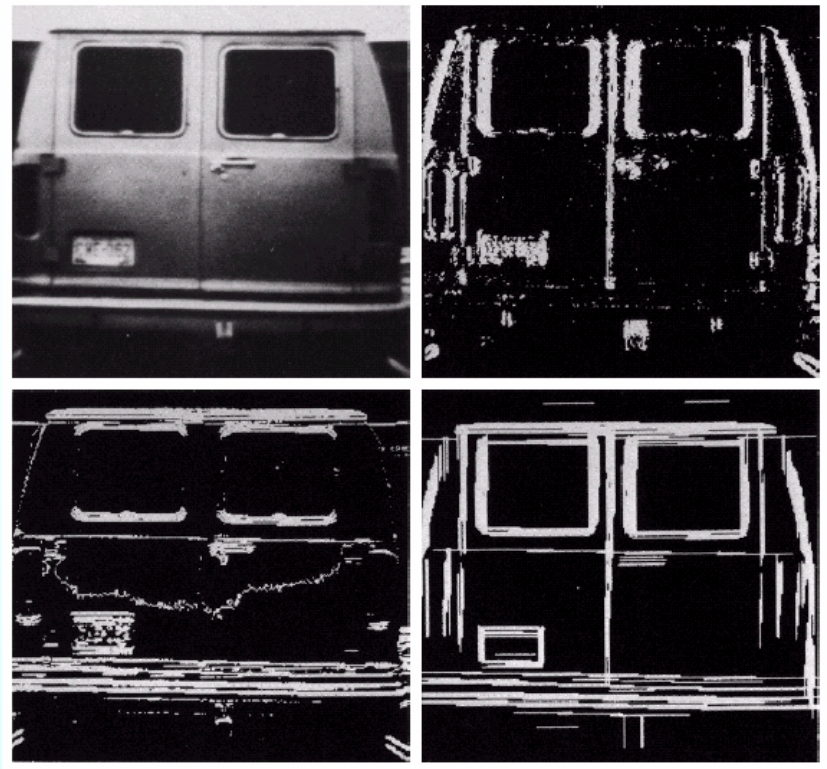
Si può dimostrare che il valor medio della funzione è 0, e lo stesso avviene per il risultato della sua convoluzione con una immagine. La tipica forma a sombrero indica inoltre che la convoluzione dell'operatore LoG con una immagine provoca un **blurring** (di entità proporzionale a σ) dell'immagine stessa, e quindi ha un effetto positivo in termini di riduzione del rumore.

LoG



Il vantaggio principale offerto dall'operatore LoG resta comunque quello legato alla presenza degli ZC. All'immagine in alto a sinistra viene applicato l'operatore LoG, con il risultato mostrato in alto a destra come immagine di intensità, nella quale i neri rappresentano i valori più negativi, i bianchi i valori più positivi. Dopo la binarizzazione: valori negativi a 0 e i valori positivi a 1 (in basso a sinistra). Gli ZC, cioè i confini tra zone nere e bianche nell'immagine binaria, sono ora facilmente individuabili (in basso a destra)

Edge Linking



DEMO

MATLAB

HIPR2

Metodo di Canny

Un approccio analitico è stato invece seguito da J. Canny, che ha studiato in dettaglio il comportamento dell'operatore gradiente applicato ad un contorno rumoroso.

Il modello di bordo considerato è un fronte ripido monodimensionale b(x) cui è aggiunto rumore Gaussiano bianco.

Si assume che l'individuazione del bordo sia realizzata tramite una convoluzione con un filtro *f(x)* avente risposta impulsiva *h*(x) antisimmetrica e nulla al di fuori di un intervallo [-W,W].

Un bordo è individuato da un massimo locale della convoluzione tra l'immagine ed il filtro. Il filtro è scelto sulla base di **tre criteri di efficacia definiti da Canny.**

Criteri di Canny

- Buona capacità di individuazione:

  l'operatore ha una bassa probabilità di non individuare un bordo reale (elevata sensibilità – *falsi negativi*) ed una bassa probabilità di individuare falsi bordi (elevata specificità – *falsi positivi*)

- Buona capacità di localizzazione:

  i punti evidenziati dall'operatore dovrebbero essere quanto più vicini possibile al centro del bordo reale

- Unicità della risposta:

  l'operatore dovrebbe fornire una sola risposta in corrispondenza di un bordo reale

Canny edge detector

Immagine Input (Lena)

Canny edge detector

Norma del gradiente

Canny edge detector

thresholding

Canny edge detector

thinning

(non-maximum suppression)

Canny – Primo Step

L' Individuazione (detection) dell'edge, si realizza attraverso la massimizzazione del rapporto segnale-rumore (delle ampiezze) del gradiente. L'espressione adoperata per il rapporto *s/r* è:

$$\text{snr} = \frac{h_E}{\sigma_n} S(h), \text{ con } S(h) = \frac{\int_{-W}^{0} h(x)dx}{\int_{-W}^{W} [h(x)]^2 dx}$$

dove h_E è l'ampiezza del gradino mentre σ_n è la deviazione standard del rumore.

Canny – Step 2

I punti riconosciuti come di edge dovrebbero essere il più possibile vicini al centro dell'edge effettivo. A tal fine è definito un fattore di localizzazione (h'è la derivata di h):

$$\mathrm{LOC} = \frac{h_E}{\sigma_n} L(h) \text{, con } L(h) = \frac{h'(0)}{\int_{-W}^{W} [h'(x)]^2 \, dx}$$

Canny – Step 3

Per quanto riguarda l'unicità della risposta dell'operatore la distanza x_m tra due picchi del gradiente, in presenza solo di rumore, è supposta uguale ad una frazione k della larghezza W dell'operatore:

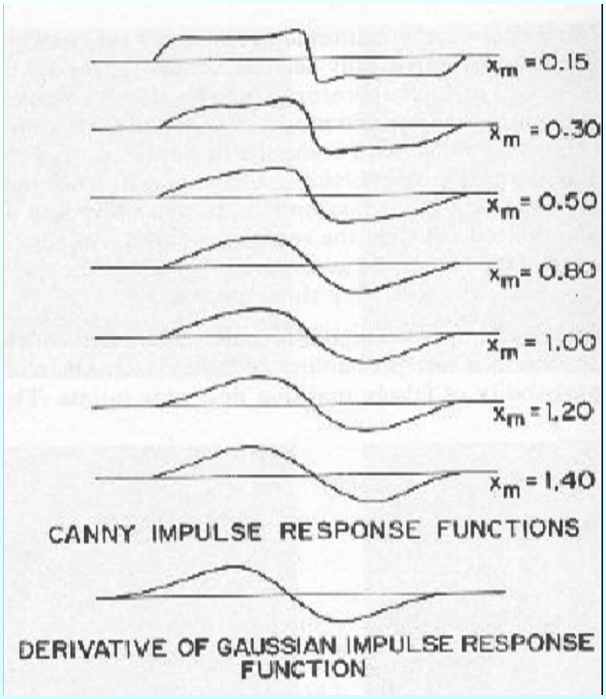
$$x_m = kW$$

Canny ha combinato i tre criteri, cercando il massimo del prodotto $S(h)L(h)$ soggetto al vincolo espresso dalla condizione di cui sopra

Canny - Discussioni

Anche se la complessità della formulazione impedisce la determinazione di una soluzione analitica, è possibile procedere alla ricerca del massimo con metodi numerici.

La figura mostra i profili di diversi valori della risposta impulsiva del filtro di Canny al crescere di x_m. Si può notare come per grandi valori di x_m l'operatore di Canny è ben approssimato dall'operatore ***DroG***. In effetti, il filtro adoperato nelle implementazioni del metodo di Canny è proprio di questo tipo.

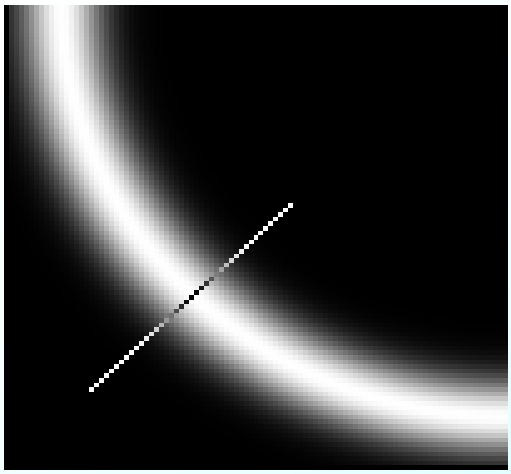


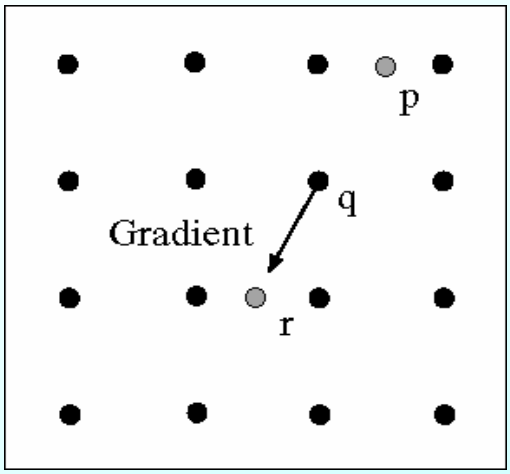
Canny:Non-Maximum Suppression

L'immagine di risposta all'operatore, se vista come una superfice 3D, è caratterizzata da valli (*valleys*) e rilievi. Le curve di massimo dei rilievi, le creste, sono detti *ridges*. Per ottenere una risposta univoca dall'edge detector è necessario un algoritmo che sopprima tutti i *responsi multipli*, ***ovvero i pixel che sono caratterizzati da valori alti di risposta all'operatore ma che non sono massimi locali per esso***.

La **non-maximum suppression** si ottiene cercando i pixel che sono di massimo per la magnitudo lungo la direzione del gradiente. Si tengono solo i *ridges* sopprimendo tutti gli altri punti che non sono all'altezza massima locale.

Non-maximum suppression





Si verifica l'esistenza di un massimo locale lungo la direzione del gradiente. Ciò richiede l'interpolazione dei pixel mancanti (pixel p ed r in figura)

Canny: Thresholding

La qualità dei risultati ottenibili con il metodo di Canny, superiore a quella di tutti gli altri operatori di gradiente, si giustifica con il fatto che il metodo utilizza due soglie, una per la individuazione degli edge più netti, l'altra per l'individuazione degli edge più deboli. Questi ultimi sono però presi in considerazione solo se risultano connessi ad edge netti.

Ruolo fondamentale nell'algoritmo di Canny gioca anche la scelta di sostituire il tradizionale approccio di sogliatura (thresholding) a soglia singola con una tecnica a *doppia soglia* detta ***histeresys thresholding***.

Canny: Thresholding

La doppia sogliatura (***Histeresys thresholding***) viene operata dopo l'applicazione della non-maximum suppression.

Si fissano due soglie T1 e T2 con T1>T2

Tutti i punti di valore maggiore di T1 sono di edge;

Tutti i punti di valore compreso fra T1 e T2 sono detti *weak edges.*

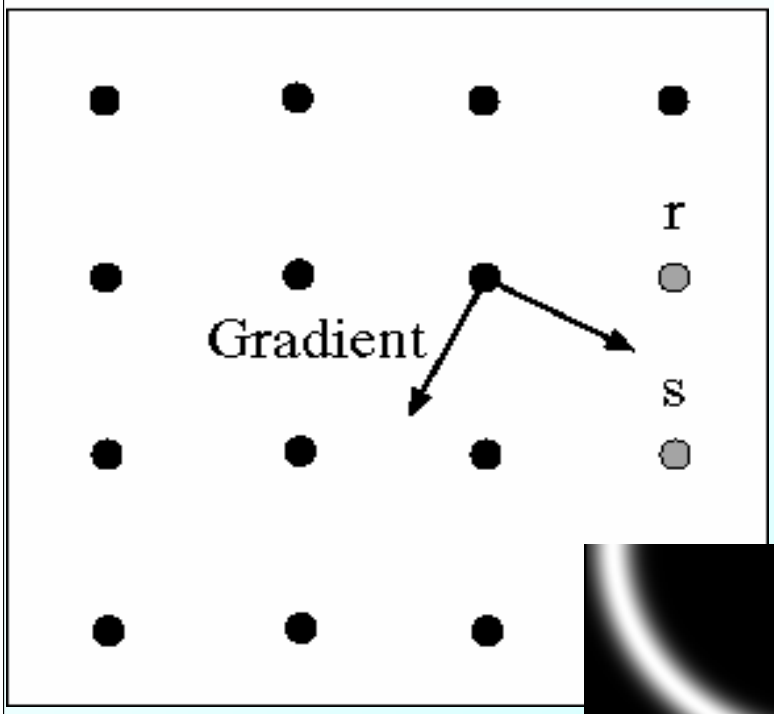
Un weak edge diventa edge solo se è contiguo ad un edge.

Canny

Diminuisce pertanto l'influenza del rumore, ed aumenta la probabilità di rivelare 'veri' edge deboli. In tutti i metodi di edge detection basati sul gradiente, occorre confrontare il risultato dell'operazione di derivazione in ogni punto dell'immagine con uno (o più) valori di soglia, per determinare se si tratta di un punto di edge. Il valore di soglia determina direttamente la sensibilità dell'edge detector.

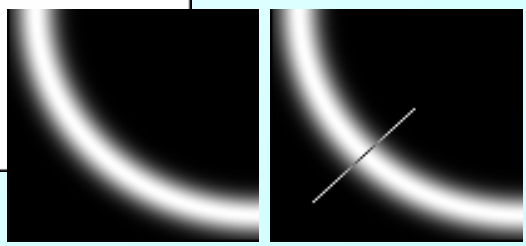
Per immagini non rumorose, la soglia può essere scelta in modo che le discontinuità di ampiezza, anche relative a zone a basso contrasto, siano interpretate come edge. Nelle immagini rumorose la scelta del valore di soglia è molto più critica, diventando un elemento di *tradeoff* tra la possibilità di rivelare falsi contorni (indotti dal rumore) e la possibilità di mancare contorni veri (relativi a piccoli dettagli).

Edge Linking

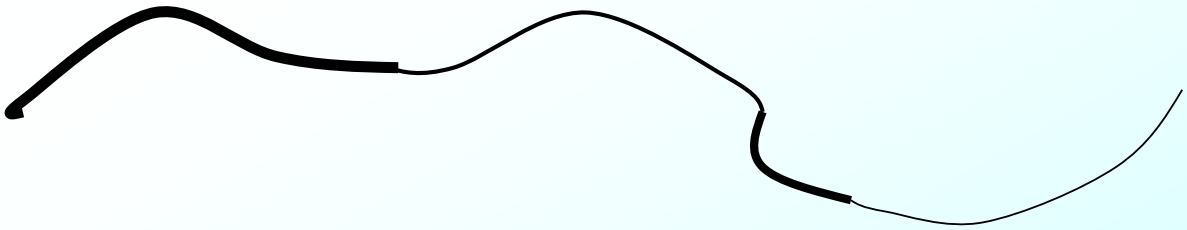


(Forsyth & Ponce)

Individuato il punto di edge si costruisce la tangente alla curva (normale al gradiente in quel punto) e si usa questa informazione per predire il successivo punto (in questo caso r o s)



Canny



Integrando la fase di thresholding con quella di linking il metodo ne guadagna in robustezza e versatilità.

Canny

Step 1 – Calcolo Gradiente (**Drog**): magnitudo e direzione

Step 2 – Eliminazione dei punti il cui valore del gradiente non è superiore ai valori dei vicini (interpolazione lineare) nella direzione del gradiente

Step 3 – Sogliatura con Isteresi mediante 2 soglie T1 e T2

Step 4 – Edge linking per legare gli *strong edge* individuati con eventuali *weak edge* adiacenti

Impatto del parametro σ (Gaussian kernel size)

Input

Canny con $\sigma=1$

Canny con $\sigma=2$

La scelta di σ impatta sulla detection dei relativi edge come segue:

- Alti valori di σ permettono di trovare edge a scale più grandi
- Piccoli valori di σ permettono di scovare I dettagli più fini

Tempo di DEMO

- MatLab Edge Demo
- Canny Demo on line
- HIPR2