Elaborazione di Segnali Multimediali

Filtraggio spaziale

L.Verdoliva, D.Cozzolino

Il filtraggio spaziale valuta la correlazione tra un'immagine e una finestra scorrevole (maschera) per valutare l'uscita, operazione che corrisponde di fatto ad una convoluzione (quindi ad un filtraggio) se la maschera scelta è simmetrica. In base alla scelta dei coefficienti della maschera, l'immagine può essere una versione "blurred" (sfocata) di quella originale, in tal caso si parla di filtri di *smoothing*. Se invece si enfatizzano i dettagli contenuti in un'immagine i filtri si dicono di *sharpening*. Di seguito esamineremo anche filtri non lineari come il filtro mediano che risulta particolarmente efficiente per il filtraggio di immagini affette da rumore impulsivo. Prima di iniziare importiamo il modulo `scipy.ndimage` con l'alias `ndi`, in quanto fornisce varie funzioni utili per eseguire il filtraggio spaziale.

```
import scipy.ndimage as ndi
```

1 Filtri di smoothing

I filtri di smoothing realizzano una media pesata dei pixel dell'immagine producendo un'immagine in cui vengono ridotte le transizioni tra i livelli di grigio (si attenuano le discontinuità tra gli oggetti). Quando tutti i coefficienti della maschera sono uguali tra loro e pari proprio all'inverso delle dimensioni della maschera, il valore in uscita non è altro che una media aritmetica dei pixel appartenenti alla finestra. Esaminiamo adesso l'effetto di un filtro che realizza la media aritmetica dei pixel appartenenti ad una finestra 3×3 per l'immagine `test.jpg`, scrivendo il corrispondente codice in Python:

```
x = np.float64(io.imread('test.jpg'))
k = 3; h = np.ones((k,k))/(k**2)
y = ndi.correlate(x, h, mode='reflect')
plt.figure(); plt.imshow(y,clim=[0,255],cmap='gray');
```

La funzione `ndi.correlate` realizza la correlazione bidimensionale tra l'immagine e la maschera del filtro e produce in uscita un'immagine delle stesse dimensioni di quella in ingresso. Ai bordi dell'immagine, dove la maschera del filtro ha bisogno per l'elaborazione anche dei valori dei pixel non appartenenti all'immagine, questi di default vengono estesi per simmetria. Possiamo specificare il tipo di estensione ai bordi tramite il parametro `mode`. Per esempio con il codice:

```
y = ndi.correlate(x, h, mode='constant')
```

Assumiamo i pixel non appartenenti all'immagine pari a zero (*zero-padding*). In oltre, è possibile estendere simmetricamente i valori dei pixel con l'opzione `mode='reflect'`, replicarli semplicemente (`mode='nearest'`) oppure rendere il segnale periodico con l'opzione `mode='wrap'`.

Provate ad aumentare la dimensione del filtro che realizza la media aritmetica per $k = 5, 9, 15, 35$ e vedete cosa succede ai bordi dell'immagine usando la funzione `ndi.correlate` con le diverse opzioni di estensione ai bordi (per quest'ultimo esperimento usate l'immagine lena.jpg).

Il modulo `scipy.ndimage` mette a disposizione funzioni specifiche per vari filtri, ad esempio la funzione `ndi.uniform_filter` per il filtro media aritmetica o la funzione `ndi.gaussian_filter` per un filtro gaussiano. Le funzioni specifiche risultano estremamente veloci. A tal proposito confrontate i tempi per un un filtraggio media aritmetica usando la funzione `ndi.uniform_filter` e la funzione `ndi.correlate`. Per una lista complete potete riferirvi alle documentazione ufficiale alla pagina:
https://docs.scipy.org/doc/scipy/reference/ndimage.html.

1.1 Esercizi proposti

1. *Filtro di smoothing.* Scrivete una funzione con prototipo: `function y = smooth(x)` che produca il filtraggio di un'immagine `x` con il filtro di smoothing con la maschera seguente:

$$h(m,n) = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} /16$$

2. *Smoothing seguito da thresholding.* Un'importante applicazione dei filtri che realizzano la media spaziale è quello di sfocare l'immagine in modo da confondere con lo sfondo oggetti piccoli di poco interesse ed enfatizzare oggetti più grandi, che quindi possono essere facilmente rilevati. Consideriamo, ad esempio, l'immagine `spazio.jpg`, proveniente dal telescopio Hubble, in orbita intorno alla terra. Realizzate le seguenti operazioni:

   (a) visualizzate l'immagine;

   (b) applicate il filtro che effettua la media aritmetica su una finestra di dimensioni 15×15 e visualizzate il risultato;

   (c) realizzate un'operazione a soglia (*thresholding*) per eliminare oggetti piccoli, in particolare considerate una soglia pari al 25 per cento del valore massimo presente nell'immagine filtrata;

   (d) visualizzate il risultato dell'elaborazione.

   Noterete che la scelta della dimensione della maschera deve essere confrontabile con gli oggetti che si vogliono trascurare, provate a modificarne la dimensione e valutatene gli effetti, variando anche la soglia opportunamente.

3. *Denoising.* Aggiungete del rumore gaussiano bianco ad un immagine `x` con il comando: `noisy = x + n` con `n = d*np.random.randn(M,N)` dove `d` è la deviazione standard del rumore. Effettuate il denoising dell'immagine con i filtri a media mobile (al variare della dimensione della finestra). Valutate l'efficacia del filtraggio sia visivamente, sia calcolando l'errore quadratico medio tra `x` e l'immagine "ripulita", che rappresenta una misura quantitativa per stabilire quanto l'immagine elaborata sia simile all'originale. L'MSE (Mean Squared Error) tra due immagini si definisce come:

$$\mathrm{MSE} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [x(m,n) - y(m,n)]^2$$

4. *Filtraggio spaziale adattativo*. Un modo per migliorare il filtro media aritmetica è quello di adottare una strategia di tipo adattativo. Detta σ^2 la varianza del rumore sull'intera immagine, μ_l la media locale e σ_l^2 la varianza locale di $y(i,j)$ calcolata su blocchi 7×7, si effettua la seguente trasformazione:

$$\widehat{x}(i,j) = y(i,j) - \frac{\sigma^2}{\sigma_l^2}\left[y(i,j) - \mu_l\right]$$

In questo modo, se $\sigma_l^2 > \sigma^2$ si restituisce un valore prossimo all'originale; invece se le varianze sono simili si effettua lo smoothing. Aggiungete rumore gaussiano per $\sigma = 5, 10, ..., 35$ all'immagine barbara.gif e tracciate la curva in cui mostrate l'MSE tra immagine originale e filtrata al variare di σ. Per realizzare un confronto tracciate poi sullo stesso grafico la curva relativa al filtro media aritmetica.

N.B. Utilizzare il comando `generic_filter` che permette di realizzare un'operazione a finestra scorrevole.

2 Filtri di sharpening

Obiettivo dei filtri di sharpening è quello di evidenziare o enfatizzare i dettagli di un'immagine. A tal fine si utilizzano operazioni che coinvolgono derivazioni del primo e secondo ordine in termini di differenze tra i valori dei pixel di un'immagine. Provate il filtro laplaciano sull'immagine `luna.jpg` e visualizzate il risultato:

$h(m,n) =$

0	1	0
1	-4	1
0	1	0

Scrivete il codice per calcolare il laplaciano di un'immagine, e visualizzate il risultato.

Il calcolo del laplaciano può essere usato per enfatizzare i dettagli in un'immagine, se si effettua la seguente operazione:

$$y(m,n) = x(m,n) - \nabla^2 x(m,n)$$

Provate allora a visualizzare l'immagine $y(m,n)$ e confrontatela con quella originale. Questa stessa operazione si può realizzare equivalentemente nel seguente modo:

```
h = np.array([[0,-1,0],[-1,5,-1],[0,-1,0]], dtype=np.float64)
z = ndi.correlate(x, h)
```

2.1 Point e line detection

I filtri di sharpening si usano spesso in combinazione con un'operazione di thresholding, che conserva solo le discontinuità più forti presumibilmente associate ai bordi. Di seguito sono presentati diversi esempi.

Per rilevare punti isolati in un'immagine si può procedere nel seguente modo:

- filtrare l'immagine utilizzando la seguente maschera:

-1	-1	-1
-1	8	-1
-1	-1	-1

- sottoporre l'immagine filtrata ad un'operazione di thresholding per individuare il punto isolato.

Applicate questa procedura all'immagine turbina.jpg, che mostra un'immagine a raggi X in cui è presente una porosità in alto a destra, in cui si trova un singolo pixel nero. Scegliendo la soglia pari al 90% del livello di grigio più grande in valore assoluto, è possibile rilevare il punto isolato. E' evidente che questo approccio si basa sul fatto che i punti da rilevare si trovino su uno sfondo omogeneo.

Se invece si vogliono individuare delle linee in un'immagine (*line detection*), è necessario definire più maschere in base alla direzione che caratterizza la linea da estrarre:

-1	-1	-1
2	2	2
-1	-1	-1

-1	-1	2
-1	2	-1
2	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

2	-1	-1
-1	2	-1
-1	-1	2

Provate ad applicare queste maschere all'immagine quadrato.jpg e confrontate le 4 mappe che evidenziano contorni orizzontali, verticali e obliqui (nelle due direzioni). Quindi effettuate l'elaborazione che vi permette di produrre un'unica mappa in cui sono presenti linee di qualsiasi tipo nell'immagine. Fate attenzione alla scelta della soglia e al fatto che le maschere così definite vi permettono di rilevare bordi spessi un pixel.

2.2 Edge detection basata sul gradiente

L'ampiezza del gradiente in un'immagine valuta quanto sono intense le variazioni di luminosità locali presenti in regioni definite dalle dimensioni della maschera utilizzata, ed è definita per un'immagine $x(m, n)$ come:

$$|\nabla \mathbf{x}| = \sqrt{\left(\frac{\partial x}{\partial m}\right)^2 + \left(\frac{\partial x}{\partial n}\right)^2}$$

Il calcolo richiede la conoscenza delle derivate direzionali, che possono essere valutate in diversi modi in base a come si definisce la derivata lungo una direzione. Se si usa la differenza prima si ottengono le maschere:

0	0
-1	1

-1	0
1	0

Mentre per la differenza centrale si ha:

0	0	0
-1	0	1
0	0	0

0	-1	0
0	0	0
0	1	0

Per cercare di ridurre la sensibilità al rumore è possibile definire le maschere in cui viene inglobata anche l'operazione di smoothing, così come accade per le maschere di *Prewitt*:

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

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

e per quelle di *Sobel*:

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

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

Realizzate l'edge detection dell'immagine house.y di dimensioni 512×512 usando le maschere di Sobel per il calcolo del gradiente e scegliendo il valore della soglia pari a `T = 1.5*np.mean(grad)`. Visualizzate l'immagine originale, il gradiente e la mappa dei contorni.

Provate a considerare anche le maschere che danno più importanza ai bordi lungo la direzione obliqua, come quelle di *Roberts*:

0	1
-1	0

1	0
0	-1

e di *Sobel*:

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

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

Applichiamo la procedura descritta all'immagine angiogramma.jpg, che rappresenta un angiogramma, ottenuto tramite l'introduzione di un catetere in un'arteria per raggiungere la zona in cui viene introdotto il mezzo di contrasto allo scopo di rilevare eventuali malformazioni nei vasi sanguigni. La mappa dei contorni ottenuta è particolarmente rumorosa, per cui potrebbe essere utile pre-elaborare l'immagine con un filtro di smoothing. A tale scopo applichiamo un filtro gaussiano bidimensionale con deviazione standard σ:

```
import scipy.ndimage as ndi
y = ndi.gaussian_filter(x, (sigma,sigma))
```

Provate a realizzare lo smoothing dell'immagine anche con il filtro media aritmetica e confrontate i risultati con quello gaussiano, al variare della dimensione della maschera.

2.3 Laplaciano di una gaussiana

Le tecniche che si basano sulla derivata seconda fanno invece ricorso al laplaciano di un'immagine. Tuttavia esso presenta due problemi: è estremamente sensibile al rumore e crea bordi doppi, che rendono più difficile l'operazione di segmentazione. Per questo motivo si cerca di sfruttare di questo operatore la proprietà per cui i passaggi per lo zero (*zero-crossing*) del laplaciano sono in grado di individuare la posizione delle discontinuità. Un interessante esempio è il *Laplaciano di una Gaussiana* (LoG) in cui l'operatore laplaciano è combinato con lo smoothing. Applichiamo allora la seguente procedura:

1. filtriamo l'immagine con il laplaciano di una funzione gaussiana, usando la `ndi.gaussian_laplace`:

```
sigma = 5
y = ndi.gaussian_laplace(x, (sigma,sigma))
```

2. valutiamo lo zero crossing, tramite la funzione `zero_crossing` presente nello script allegato *seg_utils.py*:

```
from seg_utils import zero_crossing
z = zero_crossing(y)
```

Applicate questa procedura all'immagine angio.16bit.png. Noterete come i risultati dell'elaborazione presenti numerosi anelli chiusi (*spaghetti effect*), che è il principale svantaggio di questo metodo.

3 Filtraggio non lineare

Il filtro mediano realizza un'operazione non lineare sui pixel appartenenti alla maschera: i valori vengono ordinati e poi calcolato il valore che si trova nella posizione centrale dell'ordinamento (valore mediano). Questo filtro è in grado di rimuovere molto efficacemente il rumore impulsivo *salt-and-pepper*, un tipo di disturbo caratterizzato dal fatto che alcuni pixel dell'immagine diventano bianchi o neri. Consideriamo per esempio l'immagine a raggi X di un circuito distorta da questo tipo di rumore, `circuito_rumoroso.jpg`. Se la maschera ha dimensioni 5×5, il filtro mediano può essere realizzato usando la funzione `np.median` nel seguente modo:

```
x = np.float64(io.imread('circuito_rumoroso.jpg'))
y = ndi.generic_filter(x, np.median, (5,5))
plt.subplot(1,2,1); plt.imshow(x,clim=[0,255],cmap='gray');
plt.subplot(1,2,2); plt.imshow(y,clim=[0,255],cmap='gray');
```

In ogni caso il modulo `scipy.ndimage` fornisce la funzione `ndi.median_filter` per realizzare il filtraggio mediano di un'immagine:

```
y = ndi.median_filter(x, (5,5))
```

3.1 Esercizi proposti

1. *Rumore sale e pepe.* Scegliete un'immagine, quindi aggiungete rumore sale e pepe (usate la funzione `skimage.util.random_noise`), applicate il filtro mediano e valutate il risultato sia visivamente sia tramite l'errore quadratico medio se la dimensione della finestra è $5, 7, 9$.

2. *Enhancement.* Data la seguente definizione di derivata prima lungo le direzioni orizzontale e verticale:

$$\frac{\partial x(m,n)}{\partial m} = x(m+1,n) - x(m-1,n) \qquad \frac{\partial x(m,n)}{\partial n} = x(m,n+1) - x(m,n-1)$$

   Usatela per costruire la maschera `h` del filtro che realizza il laplaciano di un'immagine. Realizzate l'enhancement mediante il laplaciano dell'immagine contenuta nel file luna.jpg. Visualizzate l'immagine elaborata e confrontatelo con il risultato ottenuto in precedenza.

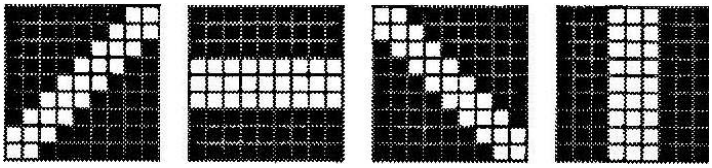
3. *Enhancement locale.* Si consideri l'immagine bebe.jpg, che rappresenta una vecchia fotografia di un neonato, in cui sono presenti dei difetti dovuti al passare del tempo. Si vuole realizzare un algoritmo iterativo che effettui l'enhancement *locale* dell'immagine, riguardante cioè le sole parti danneggiate, individuate dalla maschera binaria mask.bmp. Detta X_1 l'immagine iniziale, che coincide con l'originale eccetto per le parti danneggiate, che sono poste a zero,

   (a) filtrare l'immagine X_k con il filtro $h(m,n) = \begin{array}{|c|c|c|}\hline a & b & a \\ \hline b & 0 & b \\ \hline a & b & a \\ \hline \end{array}$ dove $a = 0.073$ e $b = 0.177$;

(b) generare l'immagine X_{k+1}, che coincide ovunque con X_k tranne che per la parte danneggiata, dove coincide invece con X_k filtrata;

Scrivete la funzione `function y = enhanc(x,mask,k)` che realizza il compito descritto e visualizzate l'immagine risultante per $k = 10, 50, 100$, verificando che per $k = 100$ si ottiene un risultato accettabile.

4. *Filtri direzionali.* Allo scopo di migliorare il filtraggio di immagini rumorose si vogliono utilizzare filtri direzionali. In particolare, considerate le seguenti maschere 9×9 per il filtraggio:



A questo punto aggiungete rumore gaussiano con deviazione standard 25 all'immagine zebre.y (321×481, uint8). Quindi realizzate i seguenti passi per filtrare ogni pixel dell'immagine:

(a) calcolate la varianza locale usando le 4 maschere mostrate in figura considerando solo i pixel selezionati (zona bianca);

(b) usate per il filtraggio del pixel la maschera corrispondente alla varianza minima con pesi tutti uguali.

Confrontate il risultato con un filtro media aritmetica 5×5 e valutate il PSNR tra l'originale e la filtrata.

5. *Filtraggio guidato.* Si vuole realizzare un filtraggio in cui si trasferisce la struttura dei bordi da un'immagine (detta guida), g, alla sua mappa di segmentazione binaria, x. A tal scopo scrivete una funzione `function y = filtro_guidato(x,g,B)` in cui si realizzano le seguenti elaborazioni:

(a) normalizzate sia la maschera che la guida nell'intervallo $(0, 1)$;

(b) calcolate le immagini delle medie locali, Med_x e Med_g, l'immagine delle varianze locali Var_g su blocchi $B \times B$;

(c) calcolate l'immagine delle correlazioni locali, $Corr_{gx}$, calcolando per ogni blocco $B \times B$ la seguente quantità:

$$\frac{1}{B^2} \sum_m \sum_n g(m,n)x(m,n)$$

(d) calcolate l'uscita del filtro come:

$$y = \mu_a g + \mu_b$$

dove μ_a e μ_b sono le immagini delle medie locali su blocchi $B \times B$ di:

$$a = \frac{Corr_{gx} - Med_x Med_g}{Var_g + \epsilon} \qquad b = Med_x - aMed_g$$

Applicate la funzione alla mappa mask.png e all'immagine guida guida.png usando $B = 10$ e $\epsilon = 2^{-60}$. Visualizzate la mappa in uscita e quella originale. Mostrate a video anche il prodotto xg e yg e un loro zoom 100×100 pixel vicino ad un bordo.

6. *Estrazione di Keypoint.* Un modo per individuare i punti salienti (keypoint) di un'immagine è quello di adottare la seguente strategia.

   Si valuta la derivata nella direzione verticale, $V(i,j)$, orizzontale, $H(i,j)$ e diagonale, $D_1(i,j)$ e $D_2(i,j)$, dell'immagine $x(i,j)$:

$$\begin{aligned} V(i,j) &= x(i,j) - x(i-1,j) \\ H(i,j) &= x(i,j) - x(i,j-1) \\ D_1(i,j) &= x(i,j) - x(i-1,j-1) \\ D_2(i,j) &= x(i,j) - x(i-1,j+1) \end{aligned}$$

   Per ognuna di queste quattro immagini si valutano le medie dei valori al quadrato $Q(i,j)$ calcolati tramite finestra scorrevole su blocchi di 5×5 pixel. Per esempio facendo riferimento alla derivata verticale:

$$Q_V(i,j) = \frac{1}{25} \sum_{m=-2}^{2} \sum_{n=-2}^{2} V^2(i+m, j+n)$$

   Quindi si determina:

$$Q_{min}(i,j) = \min \{Q_V(i,j), Q_H(i,j), Q_{D_1}(i,j), Q_{D_2}(i,j)\}$$

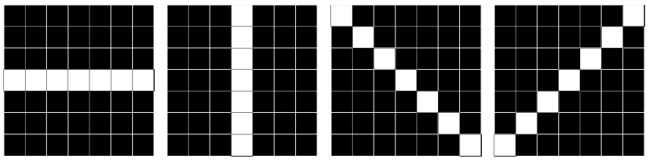
   utilizzando la funzione `np.minimum`. A questo punto si calcola il valore massimo $MQ_{min}(i,j)$ tramite finestra scorrevole su blocchi di 3×3 pixel. Infine i punti salienti sono quelli per cui:

$$SP(i,j) = Q_{min}(i,j) > 500 \ \text{ AND } \ Q_{min}(i,j) = MQ_{min}(i,j)$$

   Applicate l'algoritmo all'immagine tetto.png e visualizzate la mappa binaria $SP(i,j)$.

7. *Segmentazione delle vene.* Si vogliono identificare le vene nell'immagine del fondo oculare retina.tif attraverso opportuna segmentazione. A tale scopo scrivete una funzione `function y = segmenta(x)` che realizza i passi descritti di seguito sulla banda verde (G) dell'immagine.

   (a) Effettuate un'elaborazione a finestra scorrevole usando le seguenti 4 maschere 7×7:



   Per ogni maschera bisogna valutare la differenza tra la media dei pixel nelle quattro direzioni di interesse (regione evidenziata in bianco) e la media calcolata nella restante parte del blocco (regione evidenziata in nero);

   (b) per ogni pixel dell'immagine calcolate il minimo valore delle quattro differenze così ottenute;

   (c) realizzate un thresholding con soglia pari a -5 per ottenere una mappa binaria;

   (d) infine, eliminate il cerchio esterno che limita la retina.

   Mostrate a video l'immagine originale e la mappa di segmentazione.