Elaborazione di Segnali Multimediali
a.a. 2017/2018

Filtraggio spaziale

L.Verdoliva

In questo laboratorio ci occuperemo di filtraggio spaziale. Questa elaborazione 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.

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 Matlab:

```
x=double(imread('test.jpg'));
k=3; h=ones(k)/k^2;
y=filter2(h,x);
figure; imshow(y,[]);
```

La funzione `filter2` realizza la correlazione bidimensionale tra l'immagine e la maschera del filtro e di default 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 vengono assunti per default pari a zero (*zero-padding*). Per questo motivo si crea una distorsione ai bordi nell'immagine elaborata, che tipicamente si presenta come una cornice scura che circonda l'immagine, tanto più grande quanto più è grande la dimensione della maschera. Utilizzando l'opzione `'valid'` è possibile ottenere in uscita solo i valori elaborati dal filtro per cui la maschera si sovrappone perfettamente all'immagine (ovviamente l'uscita avrà dimensioni minori di quella in ingresso):

```
y=filter2(h,x,'valid');
```

In alternativa è possibile usare il comando conv2, che effettua proprio la convoluzione bidimensionale, ovviamente sarà necessario ribaltare la maschera del filtro (a meno che non sia simmetrica):

```
h=fliplr(h); h=flipud(h);
y=conv2(x,h);
```

Si tenga presente però che l'immagine in uscita adesso ha dimensioni maggiori di quella in ingresso. In particolare, dette $M \times N$ le dimensioni dell'immagine e $A \times B$ quelle del filtro, l'immagine in uscita avrà dimensioni $M + A - 1 \times N + B - 1$. Tuttavia il comando conv2 può fornire anche un'immagine delle stesse dimensioni di quella in ingresso se si aggiunge l'opzione `'same'`, in tal caso si preleva la parte centrale dell'immagine. Di seguito considereremo spesso maschere simmetriche per cui è possibile confondere l'operazione di correlazione con quella di convoluzione.
Infine Il pacchetto image di Matlab mette a disposizione anche il comando `imfilter`, che di default ha lo stesso comportamento di `filter2`:

```
y=imfilter(x,h);
```

Oltre alle opzioni viste precedentemente, `imfilter` dà la possibilità di usarne una che modifica i valori dell'immagine all'esterno dei bordi. In particolare, è possibile estendere simmetricamente i valori dei pixel con l'opzione `'symmetric'`, replicarli semplicemente (`'replicate'`) oppure rendere il segnale periodico con l'opzione `'circular'`.

Provate ad aumentare la dimensione del filtro che realizza la media aritmetica per $k = 5, 9, 15, 35$ e verificate, usando i comandi `whos` oppure `size` che le dimensioni di y siano quelle attese quando tale variabile è prodotta con i comandi `filter2` e `conv2` con le diverse possibili scelte `'same'` `'valid'` o `'full'`. Provate, infine, a vedere cosa succede ai bordi dell'immagine usando il comando `imfilter` con le diverse opzioni di estensione ai bordi (per quest'ultimo esperimento usate l'immagine lena.jpg).

Per filtrare un'immagine può essere molto utile usare il comando `fspecial`, che permette di creare un determinato tipo di filtro bidimensionale. Per esempio, se si vuole creare un filtro media aritmetica di dimensioni 3×3, basta scrivere:

```
h = fspecial('average', [3 3])
```

Diversi sono i filtri che si possono specificare, usate l'help in linea per avere maggiori informazioni. Confrontate poi il risultato del filtro media aritmetica con quello ottenuto mediante un filtro gaussiano.

1.1 Esercizi proposti

1. *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.

2. *Denoising*. Aggiungete del rumore gaussiano bianco ad un immagine `x` con il comando: `noisy = x + n` con `n = d*randn(size(x))` 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:

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

3. *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.

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. Proviamo a realizzare l'enhancement dell'immagine `luna.jpg` con un filtro Laplaciano la cui maschera è:

$$h(m,n) = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array}$$

Scrivete il codice Matlab 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=[0 -1 0; -1 5 -1; 0 -1 0];
z=imfilter(x,h);
```

Utilizzate adesso la seguente maschera del filtro, che porta in conto anche le variazioni lungo le direzioni oblique: `h=[1 1 1; 1 -8 1; 1 1 1];` Che differenze notate rispetto all'elaborazione precedente?

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 `median` nel seguente modo:

```
x = double(imread('circuito_rumoroso.jpg'));
y = nlfilter(x,[5 5],'median(x(:))')
subplot(1,2,1); imshow(x,[]);
subplot(1,2,2); imshow(y,[]);
```

oppure:

```
y = colfilt(x,[5 5],'sliding',@median);
```

In ogni caso il toolbox image fornisce la funzione `medfilt2` per realizzare il filtraggio mediano di un'immagine.

3.1 Esercizi proposti

1. *Rumore sale e pepe.* Scegliete un'immagine, quindi aggiungete rumore sale e pepe (usate il comando `imnoise` di matlab), 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. *Elaborazioni locali.* Matlab permette in modo abbastanza semplice di elaborare solo una regione dell'immagine mediante il comando `roipoly`, in particolare con il mouse è possibile definire i vertici del poligono della sezione da estrarre. La maschera binaria relativa viene memorizzata automaticamente nella variabile mask (uscita di roipoly):

   ```
   mask = roipoly(x); imshow(mask);
   ```

   A questo punto si può filtrare solo la regione selezionata con una determinata maschera `h`:

   ```
   y = roifilt2(h,x,mask); imshow(y);
   ```

   Scegliete un'immagine, provate a selezionarne una regione e a realizzare un'operazione di enhancement che ne migliori il contrasto, per esempio usando il laplaciano.

3. *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. Scrivete quindi una funzione `function y = enhanc(x,h)`, che realizza l'enhancement mediante il laplaciano dell'immagine contenuta nel file luna.jpg. Visualizzate l'immagine elaborata e confrontatelo con il risultato ottenuto in precedenza.

4. *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 i valori al quadrato $Q(i,j)$ calcolati tramite finestra scorrevole su blocchi di 5×5 pixel. Per esempio facendo riferimento alla derivata verticale:

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

   Quindi si determina:

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

   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 $SP(i,j)$.