Elaborazione di Segnali Multimediali
a.a. 2017/2018

Elaborazioni nel dominio spaziale (1)

L.Verdoliva

In questa lezione vedremo come caratterizzare immagini digitali in Matlab tramite media e varianza e ne valuteremo l'istogramma. Inoltre presenteremo le principali elaborazioni puntuali, cioè quelle operazioni che modificano singolarmente l'intensità di ogni pixel dell'immagine per migliorarne l'aspetto complessivo (*Image Enhancement*).

1 Caratteristiche delle immagini

Prima di realizzare qualsiasi tipo di elaborazione sulle immagini, dopo averle lette, convertitele in double: `x = double(x)`. Questa operazione è necessaria quando usate il comando `imread`, infatti, in tal caso i dati sono memorizzati come interi senza segno su 8 bit (`uint8`) e il Matlab forza anche l'uscita delle operazioni che realizzate ad essere dello stesso tipo. Per esempio, supponete che il valore di intensità $x(1,1) = 2$, allora $x(1,1) + 1.7 = 4,\ x(1,1)/4 = 1,\ x(1,1) - 4 = 0,\ x(1,1) + 255 = 255$.

Il toolbox images di Matlab fornisce i seguenti comandi per il calcolo della media e della deviazione standard di un'immagine:

```
xmed = mean2(x)     % media
xstd = std2(x)      % deviazione standard
var  = xstd^2;      % varianza
```

Queste grandezze possono anche essere valutate localmente all'immagine. Scegliete un'immagine tra quelle disponibili e supponete di voler stimare l'immagine delle medie locali, cioè l'immagine che in ogni punto presenta il valor medio calcolato in un vicinato 3×3 di ogni pixel. Il codice è il seguente (N.B. per evitare il problema ai bordi, lasciamo invariati i pixel nella cornice più esterna):

```
MED=zeros(M-2,N-2);
for i=1:M-2,
  for j=1:N-2,
     MED(i,j)=mean2(x(i:i+2,j:j+2));
  end;
end;
figure; imshow(MED,[0 255]);
```

Digitate i comandi `tic` e `toc`, rispettivamente, all'inizio e alla fine delle righe di codice per conoscere esattamente i tempi di calcolo, che verranno visualizzati a video.

Il calcolo delle statistiche locali può essere realizzato anche usando la funzione `nlfilter` di Matlab nel seguente modo:

```
y = nlfilter(x,[3 3],'mean2(x)');
subplot(1,2,1); imshow(x,[0 255]);
subplot(1,2,2); imshow(y,[0 255]);
```

La funzione `nlfilter` realizza l'operazione specificata attraverso una finestra scorrevole su tutta l'immagine. Può risultare piuttosto lenta, tuttavia è possibile usare la funzione `colfilt` che velocizza questa operazione dal momento che vettorizza il blocco da elaborare. E' chiaro quindi che quest'ultima soluzione può essere adottata solo nel caso in cui l'elaborazione fatta sul blocco è equivalente a quella realizzata sul vettore ottenuto linearizzando i dati, come è il caso del calcolo della media:

```
y = colfilt(x,[3 3],'sliding',@mean);
```

Usando i comandi `tic` e `toc`, cronometrate il tempo di esecuzione delle due elaborazioni.

Un'analisi più approfondita delle caratteristiche di un'immagine è fornita dal suo istogramma, che rappresenta la frequenza di occorrenza di ogni livello di grigio:

```
n = hist(x(:), 0:255);          % istogramma
figure; bar(n);                 % grafico a barre
axis([0 255 0 1.1*max(n)]);     % estremi per ascisse e ordinate
```

Il comando `hist` ha in ingresso il *vettore* `x` dei campioni di luminanza, e gli intervalli in cui viene suddiviso il range di tali valori; in uscita restituisce in `n(i)` il numero di volte che $x(m,n)$ appartiene all'i-esimo intervallo, in questo caso quindi il numero di volte in cui $x(m,n) \in (i-1/2, i+1/2)$. Provate a visualizzare gli istogrammi di alcune delle immagini che avete a disposizione. Il listato precedente è equivalente al comando `hist(x(:),0:255)` solo che quest'ultimo produce solo la figura, e non il vettore delle occorrenze `n`. Provate poi a visualizzare l'istogramma usando i comandi `stem` e `plot` al posto di `bar`.

1.1 Esercizi proposti

1. *Immagine delle medie*. Scrivete una funzione dal prototipo `function MED = medie(x,K)` che fornisce l'immagine delle medie locali su finestre $K \times K$. Fate poi un esperimento in cui calcolate l'immagine delle medie al variare di $K = 3, 5, 7, 9$ e visualizzate il risultato. Che osservazioni potete fare?

2. *Immagine delle varianze*. Scrivete adesso una funzione dal prototipo `function VAR = varianze(x,K)` per calcolare l'immagine delle varianze locali su finestre $K \times K$. Usate questa funzione per valutare e visualizzare l'immagine delle varianze dell'immagine `filamento.jpg` usando blocchi 3×3. Che tipo di informazioni vi dà sull'immagine?

3. *Istogramma*. Il toolbox *images* vi fornisce il comando `imhist`, usate l'help in linea per conoscerne le funzionalità e provate ad utilizzarlo.

2 Operazioni puntuali

Le elaborazioni puntuali si applicano ad ogni pixel dell'immagine senza considerare interazioni e dipendenze tra pixel vicini, e sono definite in funzione dell'intensità del pixel. Vedremo operazioni lineari e non lineari.

2.1 Offset additivo e cambiamento di scala

Cominciamo col valutare l'effetto di un offset additivo su un'immagine, realizziamo cioè la trasformazione: $y = x+b$, possiamo usare le immagini `vista_aerea.jpg` e `granelli.jpg`. La visualizzazione dell'istogramma può aiutarvi a scegliere il valore da assegnare all'offset b. In particolare, per la fotografia aerea un valore ragionevole è $b = -50$, quindi scrivendo `y = x-50` si ottiene l'immagine elaborata. Se si vuole fare in modo che l'immagine in uscita abbia la stessa dinamica dell'ingresso, bisogna fare attenzione al fatto che i valori dell'immagine potrebbero diventare negativi. Si può scrivere allora il seguente codice:

```
mask=x>0;     % crea una matrice di vero (1) e falso (0)
x=x.*mask;    % si annullano tutti i valori minori di 0
```

o equivalentemente

```
x(x<0) = 0;
```

In questo modo si gestiscono solo i valori negativi, per evitare anche di avere valori più grandi di 255:

```
x(x>255) = 255;
```

Visualizzate la maschera per vedere in quali punti eventualmente si ha un effetto di saturazione.

Notate come con questa operazione le immagini presentino globalmente un aspetto più o meno luminoso, tuttavia l'istogramma, che è stato traslato verso sinistra o destra, mostra come non tutti i valori siano stati usati. Provate ad effettuare un cambiamento di scala $y = ax$ sulle due immagini proposte, determinando il valore appropriato di a in entrambi i casi e confrontando questa soluzione con quella precedente.

2.2 Negativi di un'immagine

Considerate l'immagine della mammografia di fig.3.4, realizzatene il negativo $(y = -x + K - 1)$ con $K = 256$ e confrontate il risultato con quello che si ottiene tramite la funzione `imcomplement` di Matlab.

2.3 Full Scale Histogram Stretch

Il Full Scale Histogram Stretch – o Contrast Stretch – è una delle operazioni più utili nell'ambito dell'elaborazione e della visualizzazione di immagini digitali. Scrivete una funzione con il seguente prototipo: `function y = FSHS(x)` che realizza questa operazione nell'ipotesi in cui $K = 256$. Applicate questa funzione ad immagini il cui istogramma non copre l'intera dinamica e visualizzate il risultato. Come già detto Matlab realizza automaticamente questa operazione attraverso il comando `imshow(x,[])`, provate allora a confrontare il risultato ottenuto con quello che avreste visualizzando l'immagine originale con tale comando.
N.B. Per calcolare massimo e minimo potete usare i seguenti comandi: `xmin = min(min(x))` e `xmax = max(max(x))`. E' necessario chiamare due volte la funzione perché per matrici tali funzioni forniscono un

vettore riga che contiene il minimo (massimo) di ogni colonna, a meno che i dati non siano vettorizzati prima: `xmin = min(x(:))` (`xmax = max(x(:))`).

2.4 Trasformazione logaritmo e potenza

Applichiamo la trasformazione logaritmo $y = \log(x + 1)$ in modo da migliorare il contrasto nelle zone scure per la trasformata di Fourier rappresentata nell'immagine `spettro.jpg`:

```
y = log(x+1);
figure; imshow(y,[]);
```

Provate invece ad applicare la trasformata potenza $y = x^{\gamma}$ all'immagine di `vista_aerea.jpg`, con $\gamma = 3$ e all'immagine di `spina_dorsale.jpg` (risonanza magnetica di una spina dorsale fratturata) con $\gamma = 0.3$ (ricordate di realizzare sempre il contrast stretch dopo l'operazione non lineare). Realizzate l'enhancement di immagini troppo chiare o scure, con $\gamma > 1$ e $\gamma < 1$, rispettivamente, visualizzando il risultato al variare di γ.

2.5 La funzione imadjust

Nel toolbox images di Matlab è presente `imadjust` funzione base per le trasformazioni puntuali su immagini monocromatiche. la sua sintassi generale è:

```
y = imadjust(x, [low_in high_in], [low_out high_out], gamma);
```

Il primo intervallo rappresenta il range dei livelli di grigio dell'immagine in ingresso, mentre il secondo quello dell'immagine di uscita. I valori di default sono [0 1], presentati quindi in forma normalizzata. Il valore di gamma invece definisce la forma della curva di trasformazione, per $\gamma = 1$ (valore di default) il mapping è lineare. Come esempio, scriviamo il seguente codice ed applichiamolo all'immagine mammografia.jpg:

```
y = imadjust(x, [0.5 0.75], [0 1]);
```

L'elaborazione espande l'intervallo di livelli di grigio compresi tra 0.5 e 0.75 al range [0 1] ed è utile qualora si voglia enfatizzare un particolare intervallo di livelli di grigio. Modificate opportunamente i parametri della funzione e realizzate nuovamente il negativo dell'immagine.

2.6 Equalizzazione dell'istogramma

L'equalizzazione dell'istogramma permette di migliorare il contrasto di un'immagine, quando sono stati usati tutti i valori appartenenti alla dinamica, ma la distribuzione appare fortemente sbilanciata. Caricate l'immagine marte.jpg, visualizzate il suo istogramma e poi effettuate l'equalizzazione dell'istogramma usando la funzione `histeq` del toolbox *images* di Matlab.

2.7 Uso delle statistiche locali

In questa sezione realizziamo l'enhancement solo su una parte dell'immagine `filamento.jpg`, allo scopo di evidenziare una struttura nascosta. Vogliamo effettuare un'elaborazione (amplificazione del valore) solo dei pixel appartenenti alle aree scure, che abbiano basso contrasto, ma non nelle regioni piatte. Questo si traduce in termini di condizioni su media e deviazione standard locale dell'immagine. Un pixel viene elaborato se:

1. la luminosità media attorno al pixel (media locale) è opportunamente minore della luminosità media dell'intera immagine;
2. la deviazione standard locale è relativamente bassa (indice di basso contrasto) e non è troppo piccola (per non enfatizzare le aree omogenee).

A questo punto solo se la media locale di un pixel è minore di quella globale opportunamente scalata ($\mu_l \leq k_0\mu$ per individuare zone scure) e se la deviazione standard locale soddisfa la disuguaglianza ($k_1\sigma \leq \sigma_l \leq k_2\sigma$ per assicurare un basso contrasto), con k_0, k_1 e k_2 costanti, il pixel va amplificato di un fattore E. Assumeremo $k_0 = 0.4$, $k_1 = 0.02$, $k_2 = 0.4$ e $E = 4$. Per realizzare il codice che effettua tali operazioni bisogna determinare la matrice di test (maschera), che vale 1 laddove il pixel va elaborato, 0 altrimenti: Nell'ipotesi di aver già calcolato media, `med`, e deviazione standard, `std`, globali dell'immagine, di conoscere l'immagine delle medie locali, `MED`, e quella delle deviazioni standard, `DEV`, su blocchi 3×3, il comando che genera la maschera è:

```
Mask = ((MED<=0.4*med)).*(DEV<=0.4*dev)).*(DEV>=0.02*dev)));
```

Scrivete uno script dal nome `localenhanc.m` che realizza le operazioni appena descritte e che richiama due funzioni per il calcolo delle medie e delle deviazioni standard, rispettivamente. Ricordate di visualizzare l'immagine originale e quella finale.