Elaborazione di Segnali Multimediali
a.a. 2017/2018

Elaborazioni nel dominio spaziale (1)
Soluzioni

1 Caratteristiche delle immagini

1. Riprendiamo il codice per il calcolo della media locale su una finestra 3×3 dell'immagine, usando il comando `mean2` (presente nel toolbox image) che calcola direttamente la media di una matrice:

```
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;
```

Questo codice non prevede di gestire i bordi, infatti produce un'immagine delle medie più piccola di quella originale. Per gestire il problema dei bordi è necessario estendere l'immagine di partenza mediante riempimento con zeri (*zero padding*) o estensione periodica o simmetrica (in base alle esigenze dell'applicazione). Per realizzare questa operazione si può usare il comando `padarray`:

```
>> xext = padarray(x,[1,1],0);
```

In questo caso è stata aggiunta una cornice esterna di zeri all'immagine. Usando le opzioni 'symmetric' e 'replicate' è possibile effettuare un'estensione simmetrica e periodica, rispettivamente. Modifichiamo allora il codice usando l'estensione ai bordi:

```
xext = padarray(x,[1,1],0);
MED=zeros(M,N);
for i=1:M,
  for j=1:N,
     MED(i,j)=mean2(xext(i:i+2,j:j+2));
  end;
end;
```

Tenete presente che la soluzione che prevede l'uso delle funzioni `nlfilter` e `colfilt` effettua un'estensione con tutti zeri ai bordi. Inoltre notate come la sintassi:

```
>> y = nlfilter(x,[3 3],@mean2);
```

risulti molto più veloce di:

```
>> y = nlfilter(x,[3 3],'mean2(x)');
```

A questo punto è semplice scrivere la funzione per il calcolo delle medie:

```
function MED = medie(x,K);
% calcola l'immagine delle medie

MED = nlfilter(x,[K K],@mean2);
```

2. Per il calcolo dell'immagine delle varianze su blocchi 3×3, la soluzione è la seguente:

```
DEV = zeros(M-2,N-2);
for i=1:M-2,
    for j=1:N-2,
        x = x(i:i+2,j:j+2);
        DEV(i,j) = std(x(:),1);
    end;
end;
VAR = DEV.^2;
```

Anche in questo caso si possono usare le funzioni di Matlab:

```
>> DEV = nlfilter(x,[3 3],@std2);
>> DEV = colfilt(x,[3 3],'sliding',@std);
>> VAR = DEV.^2;
```

2 Operazioni puntuali

1. *Negativo di un'immagine.*

   Di seguito si presenta il codice per determinare, visualizzare e salvare su file il negativo di un'immagine. Fate attenzione al fatto che prima di salvare i dati in formato JPEG è necessario convertire il loro formato, rappresentandoli su 8 bit.

```
>> x = double(imread('mammografia.jpg'));
>> y1 = -x+255;
>> y2 = imcomplement(x);
>> subplot(1,2,1); imshow(y1,[0 255]);
>> subplot(1,2,2); imshow(y2,[-255 0]);
```

   Potete in alternativa usare la funzione imadjust:

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

Fate attenzione perché in quest'ultimo caso i dati devono essere uint8.

2. *Full-Scale Histogram Stretch*.

```
function y = fshs(x)
% FSHS Full-Scale Histogram Stretch
% y = FSHS(x) effettua il Full-scale Histogram Stretch

m = min(x(:));
M = max(x(:));
Y = 255*(x-m)/(M-m);
figure; imshow(y);
```

3. *Equalizzazione*.

```
clear all; close all; clc;
x = imread('tire.tif');
figure; imshow(x);

[n,h] = imhist(x);
figure; bar(n);
axis([0 255 0 1.1*max(n)]);

y = histeq(x);
figure; imshow(y);

[n,h] = imhist(y);
figure; bar(n);
axis([0 255 0 1.1*max(n)]);
```

4. *Uso delle statistiche locali*.

Nell'ipotesi di aver scritto due funzioni: `medie` e `devstd` per il calcolo delle statistiche su blocchi $K \times K$ dell'immagine, il codice per l'enhancement è presentato di seguito.

```
clear all; close all; clc;
x = vediJPG('Fig3.24.jpg');
MED = medie(x);
DEV = devstd(x));
med = mean2(x);
dev = std2(x);
Mask = ((MED<=0.4*med)).*(DEV<=0.4*dev)).*(DEV>=0.02*dev)));
y = x(2:M-1,2:N-1)+3*x(2:M-1,2:N-1).*Mask;
figure; image(y); colormap(gray(256)); axis image;
```

3 Suggerimenti

Alcuni suggerimenti per l'uso di Matlab.

- Le funzioni che scrivete in Matlab devono sempre trovarsi nella directory corrente affinché possiate utilizzarle. In realtà è possibile modificare il *path* del Matlab, cioè aggiungere il percorso in cui si trovano le vostre funzioni al percorso di default in cui Matlab cerca la libreria standard. In questo modo non sarà necessario trovarsi nella directory corrente per richiamare le funzioni. Da *File* scegliete l'opzione *Set path* della tendina a menù, quindi inserite il percorso in cui si trovano le vostre funzioni e poi cliccate su *Add Folder*, infine salvate e uscite.

  (N.B. Non potete fare questa modifica in laboratorio, ma solo sul vostro PC).

- E' possibile realizzare il *debug* del codice che scrivete usando tasto debug dall'editor di testo.

- Matlab vi permette di effettuare l'indentazione automaticamente selezionando il testo desiderato, quindi premendo *smart indent* dal menù text.