Elaborazione di Segnali Multimediali
a.a. 2017/2018

Filtraggio spaziale
Soluzioni

1 Filtri di smoothing

1. *Smoothing seguito da thresholding*.

```
x = double(imread('spazio.jpg'));
figure(1); imshow(x);
h = ones(15)/225; % maschera del filtro
y = filter2(h,x);
figure(2); imshow(y,[]);
th = (25/100)*max(y(:)); % soglia = 25 per cento del valore max
z = y>th; % calcolo la maschera
figure(3); imshow(z,[]);
w = z.*x; % risultato dell'elaborazione
figure(4); imshow(w);
```

2. *Denoising*. Aggiungiamo rumore gaussiano ad un'immagine e poi valutiamo l'effetto di un filtro media aritmetica al variare della dimensione della finestra calcolando l'MSE tra immagine originale e ripulita.

```
x = double(imread('lena.jpg'));
noisy = x + 20*randn(size(x));
d = [3 5 7 9];
for k = 1:length(d),
    h = fspecial('average',[d(k) d(k)]);
    y = imfilter(noisy,h,'symmetric','same');
    figure(1);
    subplot(1,2,1); imshow(x,[]); title('originale');
    subplot(1,2,2); imshow(y,[]); title(sprintf('filtrata per d=%d',d(k)));
    mse(k) = mean2((x-y).^2);
    pause;
end
figure(2); plot(d,mse,'r-*');
xlabel('dimensione della finestra'); ylabel('MSE');
```

3. *Filtraggio spaziale adattativo.* Innanzitutto scriviamo uno script che richiama la funzione che realizza il filtraggio adattativo (`filtra`) al variare della deviazione standard del rumore e produce il grafico

sia dell'MSE che del PSNR (rapporto segnale-rumore di picco) definito (per immagini con dinamica $[0:255]$) come:

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{MSE}}$$

```
[M N] = size(x);
sigma = [5 10 15 20 25 30 35];
for i = 1:length(sigma),
   y = x + sigma(i)*randn(M,N);
   xr = filtra(y,sigma(i));
   MSE(i) = mean2((xr-x).^2);
   PSNR(i) = 10*log10(255^2/MSE(i));
end
figure; plot(sigma,MSE,'k-o');  xlabel('sigma rumore'); ylabel('MSE');
figure; plot(sigma,PSNR,'k-o'); xlabel('sigma rumore'); ylabel('PSNR');
```

Per realizzare il filtraggio il codice può essere scritto in diversi modi. Cominciamo con la soluzione che prevede il doppio ciclo for:

```
function y = filtra(x,sigma);

[M,N] = size(x);
d = 1;
sn2 = sigma^2;
y1 = zeros(M-2*d,N-2*d);
for i=d+1:M-d,
    for j=d+1:N-d,
        b = x(i-d:i+d,j-d:j+d);
        sl2 = std2(b)^2;
        ml  = mean2(b);
        y1(i-d,j-d) = x(i,j) - (sn2/sl2)*(x(i,j)-ml);
    end;
end;
y = padarray(y1, [d d], 'symmetric');
```

In alternativa si possono calcolare la matrice delle medie e delle varianze locali su blocchetti 3×3 (come già visto nella seconda esercitazione) e poi procedere al filtraggio direttamente sull'immagine come matrice:

```
function y = filtra(x,sigma);

sn2 = sigma^2;
MED = medie(x);
VAR = varianze(x);
y = x - (sn2./VAR).*(x-MED);

function MED = medie(x);
MED = colfilt(x,[3 3],'sliding',@mean);

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

Confrontate i tempi di elaborazione delle due soluzioni, la prima impiega circa 6 secondi, la seconda intorno a 0.2 secondi. Infine si può usare la funzione `nlfilter` per specificare il filtraggio adattativo:

```
function y = filtra(x,sigma);

sn2 = sigma^2;
y = nlfilter(x,[3 3],@adatt,sn2);

function y = adatt(x,sn2);
sl2 = std(x(:))^2;
ml  = mean(x(:));
y = x - (sn2/sl2)*(x-ml);
y = y(2,2);
```

In questo caso l'operazione impiega circa 8 secondi.

2 Filtri non lineari

1. *Rumore sale e pepe*. Il codice che fa uso del doppio ciclo for è:

```
x=double(imread('circuito_rumoroso.jpg'));
[M,N]=size(x);
d=2;
y=zeros(M-2*d,N-2*d);
for i=d+1:M-d,
  for j=d+1:N-d,
     b=x(i-d:i+d,j-d:j+d);
     y(i-d,j-d)=median(b(:));
  end;
end;
figure(2); imahow(y,[]);
```

In alternativa è possibile usare la funzione `nlfilter` o la sua versione ottimizzata `colfilt` (ricordate che quest'ultima funzione può essere usata solo nel caso in cui l'operazione coinvolta si possa realizzare sui dati vettorizzati):

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

Notate come nel primo caso l'immagine risultante non ha problemi ai bordi dato che è stata fatta un'estensione simmetrica, nel secondo caso invece è presente un fastidioso effetto causato dal fatto che di default la funzione `colfilt` effettua lo zero-padding dell'immagine.

Infine, potete usare la funzione di Matlab che realizza il filtraggio mediano di un'immagine:

```
>> y = medfilt2(x,[k k]);
```

2. *Enhancement*.

```
clear all; close all; clc;
x = double(imread('luna.jpg'));
figure; imshow(x);

% Di seguito la maschera del filtro che realizza il Laplaciano
% secondo la definizione di derivata indicata nel testo
h = [0 0 1 0 0; 0 0 0 0 0; 1 0 -4 0 1; 0 0 0 0 0; 0 0 1 0 0];
y = imfilter(x,h);
y_enhanc = x-y;
figure; imshow(y_enhanc,[]);
```

3. *Estrazione di keypoint*.

```
clear all; close all; clc;
x = double(imread('tetto.png'));

% Creazione dei filtri per il calcolo delle derivate
V  = [0 -1 0; 0 1 0; 0 0 0];
H  = [0 0 0; -1 1 0; 0 0 0];
D1 = [-1 0 0; 0 1 0; 0 0 0];
D2 = [0 0 -1; 0 1 0; 0 0 0];

% Calcolo delle derivate prime
v = imfilter(x,V,'replicate');
h = imfilter(x,H,'replicate');
d1 = imfilter(x,D1,'replicate');
d2 = imfilter(x,D2,'replicate');

% Quadrati delle derivate
vv  = v.^2;
hh  = h.^2;
dd1 = d1.^2;
dd2 = d2.^2;
```

```
% Calcolo su finestra scorrevole dei valori al quadrato
Qv  = colfilt(vv,[5 5], 'sliding', @sum);
Qh  = colfilt(hh,[5 5], 'sliding', @sum);
Qd1 = colfilt(dd1,[5 5], 'sliding', @sum);
Qd2 = colfilt(dd2,[5 5], 'sliding', @sum);

% Calcolo del minimo
Qmin  = min(min(Qv,Qh), min(Qd1,Qd2));
MQmin = colfilt(Qmin, [3 3], 'sliding', @max);

% Calcolo dei punti salienti
SP = (Qmin > 500) & (Qmin == MQmin);

% Visualizzazione
figure;
subplot(1,2,1); imshow(x,[]); title('Immagine test');
subplot(1,2,2); imshow(SP,[]); title('Mappa dei Keypoint');
```