Relazione di Elaborazione delle Immagini: Filtraggio Spaziale

Carlo Franzelli

Anno Accademico 2008/2009

Indice

1 Consegna

Scrivere un breve codice Matlab capace di eseguire il filtraggio spaziale di una immagine. Impostare i coefficienti della maschera in modo da ottenere un filtro di smoothing. La maschera spaziale deve Gaussiana con sigma=2 (che dimensioni deve avere dunque la maschera? 13 x 13 è un buon compromesso, perché?). Applicare il filtro all'immagine 1.

2 Implementazione

2.1 Filtro spaziale

Il filtro spaziale è stato implementato usando la formula

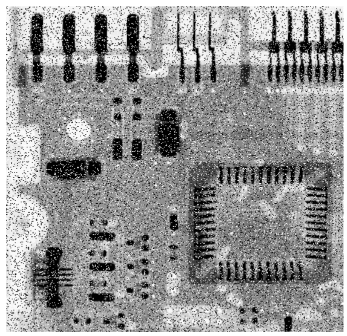

Figura 1: L'immagine di ingresso

$$f(x,y) = Ae^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2}\right)}.$$

Dove i valori di alcune variabili sono stati impostati come costanti:

- $A = 1$
- $x_0 =$ `round(M/2)`
- $y_0 =$ `round(N/2)`

In particolare i due ultimi valori presuppongono che la maschera sia centrata. Nelle formule abbiamo M e N come dimensioni del filtro. Un'altra approssimazione della formula è stata quella di usare $\sigma = \sigma_x = \sigma_y$.
Per l'implementazione è bastato usare un filtro 13x13, questo è stato sufficiente poichè è stato chiesto di utilizzatare $\sigma = 2$, e la maggior parte dell'informazione è contenuta in $(-3\sigma; 3\sigma)$ (vedi figura 2) Inoltre la maschera, una volta generata

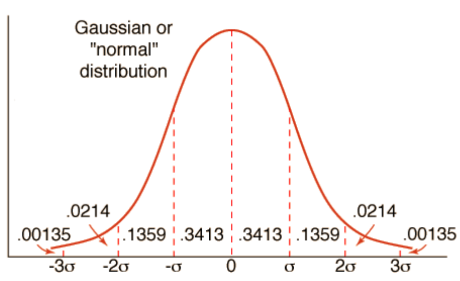

Figura 2: Densità di probabilità della Gaussiana

viene normalizzata a valori compresi tra $0, 1$, e per l'elaborazione nella convoluzione verrà ulteriormente compressa in modo tale che la sommatoria dei valori sia 1.

$$\sum_{M}^{i=0} \sum_{N}^{j=0} h(i,j) = 1.$$

Il risultato è mostrato in figura 3.

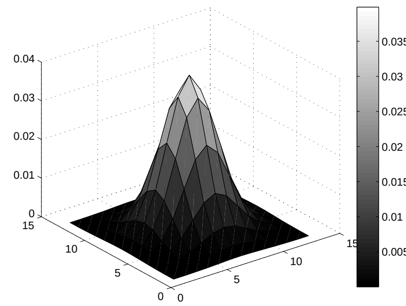

Figura 3: Visualizzazione tridimensionale della maschera

3 L'algoritmo di convoluzione

L'algoritmo è stato ottenuto tramite la formula della convoluzione (eq. 1). Il costo computazionale è pari a MNh_xh_y, con M e N dimensioni dell'immagine e h_x, h_y come dimensioni del filtro. Inoltre per poter far la convoluzione si è dovuto allargare l'immagine tramite il *padding*. La dimensione del padding è stata di `floor`$(\frac{\min(h_x,h_y)}{2})$.

$$g(x,y) = \sum_{s=-h_x}^{h_x} \sum_{t=-h_y}^{h_y} w(s,t) f(x-s, y-t) \tag{1}$$

Si nota che per filtri gaussiani discreti di dimensione dispari, la rotazione non modifica il filtro.

4 Risultato

Il risultato dell'elaborazione è mostrato in figura 4. Come ci si aspettava il risultato risulta sfocata (*blurring*) dell'immagine in ingresso. E' stato calcolato anche il risultato della computazione attraverso gli strumenti `conv2,fspecia` di `matlab`, e il tutto è stato confrontato attraverso l'RMSE (eq. 2).

$$RMSE = \frac{\sum_{i=0}^{M} \sum_{j=0}^{N} (f(i,j) - \widehat{f}(i,j))^2}{MN} \tag{2}$$

Il confronto produce un RMSE pari a $2.55e^{-13}$, che indica che i due risultati sono pressochè identici.

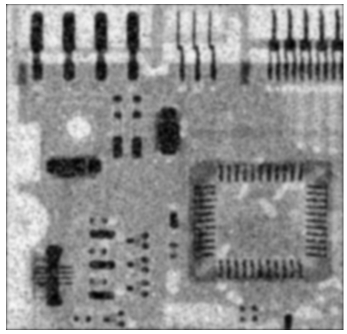

Figura 4: Immagine risultante

5 Code

5.1 Funzione per la maschera gaussiana

```
function [ gaus_im ] = h_gauss(sigma, m , n)

%creo i valori per le coordinate
x = 1:1:m;
y = 1:1:n;

%il centro viene arrotondato
x0 = round(m/2);
y0 = round(n/2);

% calcolo i valori
for i = 1:m
    for j = 1:n
        g(i,j) = exp( - (((x(i)-x0).^2)./(2*sigma^2) + ((y(j)-y0).^2)./(2*sigma^2) ));
    end
end

%assegno il risultato
gaus_im = g / max(max(g));

end
```

5.2 Codice della correlazione

```
% codice per il progetto di elaborazione dell'informazione


f = imread('Fig0335(a)(ckt_board_saltpep_prob_pt05).tif');

f = double(f)/255;
```

```
[m n] = size(f);

h_dim_x = 13;
h_dim_y = 13;
% sigma fissato
sigma = 2;


% genero la maschera di conv 0 < h < 1
h = h_gauss(sigma,h_dim_x,h_dim_y);

h = h/sum(sum(h));

pad_x = floor(h_dim_x/2);
pad_y = floor(h_dim_y/2);

m_padded = m + pad_x*2;
n_padded = n + pad_y*2;

g_padded = zeros(m_padded,n_padded);

g_padded(pad_x+1:pad_x+m,pad_y+1:pad_y+n) = f(:,:);

res_padded = zeros(m_padded,n_padded);

for r = pad_x+1:pad_x+m
    for s = pad_y+1:pad_y+n
        S = 0;
        s_count = 0;
        for t = 0:h_dim_x-1
            for u = 0:h_dim_y-1
                 S = S + g_padded(r+t-pad_x,s+u-pad_y) * h(t+1,u+1);
                 s_count = s_count + 1;
            end
        end
        assert(s_count == 13 * 13);
        res_padded(r,s) = S;
    end
end

%res_padded = res_padded / max(max(res_padded));

res_depadded(:,:) = res_padded(pad_x+1:pad_x+m,pad_y+1:pad_y+n);

%imshow(uint8(res_depadded*255));

%% matlab part

h_matl = fspecial('gaussian',[h_dim_x h_dim_y],sigma);
```

```
res_matl = conv2(f,h_matl,'same');

rmse = sqrt(sum(sum((res_depadded - res_matl).^2))/m*n);

rmse
```