

# Università degli studi di Udine

Corso di Immagini e Multimedialità – 2016-17

prof.Vito Roberto

## Relazione finale

Galante Gregorio  
mat.: 120337

## Esercizio 1.1

### Traccia esercizio

Con riferimento alla Lezione 1.3 svolgere le varie fasi della visualizzazione e dell'analisi esplorativa di immagini statiche. Realizzare uno script MATLAB per ciascuno dei seguenti compiti:

- (a)- Visualizzare l'immagine 'cameraman'; i metadati; la sua look-up table (LUT); l'istogramma dei livelli di grigio.
- (b)- Visualizzare l'immagine 'peppers'; i metadati.
- (c)- Visualizzare ciascuna componente di colore R, G, B dell'immagine 'peppers' e il relativo istogramma dei livelli di intensità.
- (d)- Visualizzare separatamente, in forma di immagini, le tre componenti di colore R, G, B.

#### Punto (a)

##### Codice

Visualizzazione immagine:

```
% utilizzo imread per caricare l'immagine  
img = imread('cameraman.tif');  
% utilizzo imshow per visualizzare l'immagine  
imshow(img);
```

Visualizzazione dei metadati dell'immagine:

```
% utilizzo imfinfo per ottenere i metadati dell'immagine  
metadati = imfinfo('cameraman.tif');  
% utilizzo imageinfo per visualizzare i metadati dell'immagine  
imageinfo(imgshow, metadati);
```

Visualizzazione LUT dell'immagine:

```
% visualizzo la LUT dell'immagine  
imshow(img, 'InitialMagnification', 'fit'), colorbar;
```

Visualizzazione istogramma scala di grigi dell'immagine:

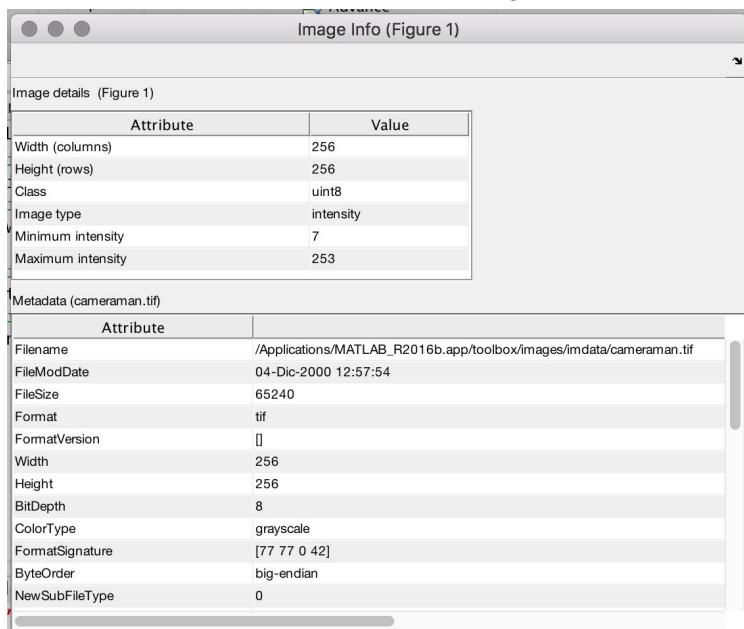
```
% utilizzo imhist per visualizzare l'istogramma della scala di grigi  
imhist(img), grid;
```

## Risultati

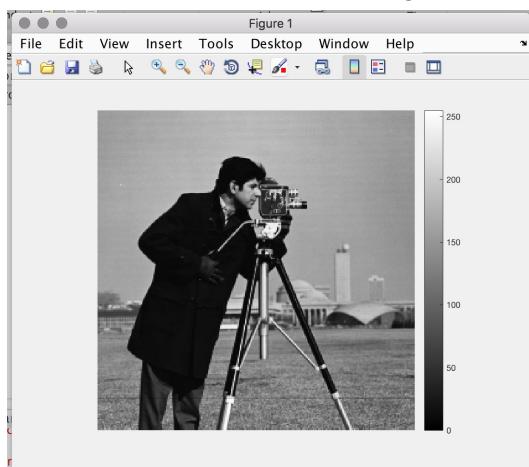
Visualizzazione immagine:



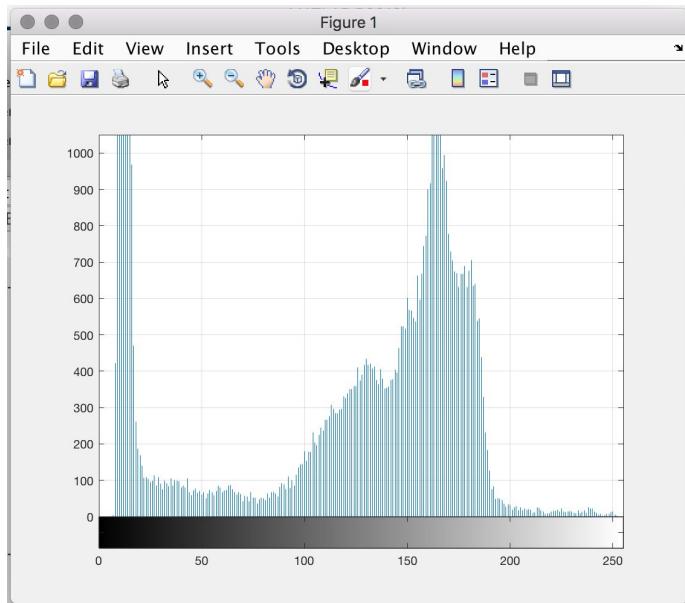
Visualizzazione dei metadati dell'immagine:



Visualizzazione LUT dell'immagine:



Visualizzazione istogramma scala di grigi dell'immagine:



## Osservazioni

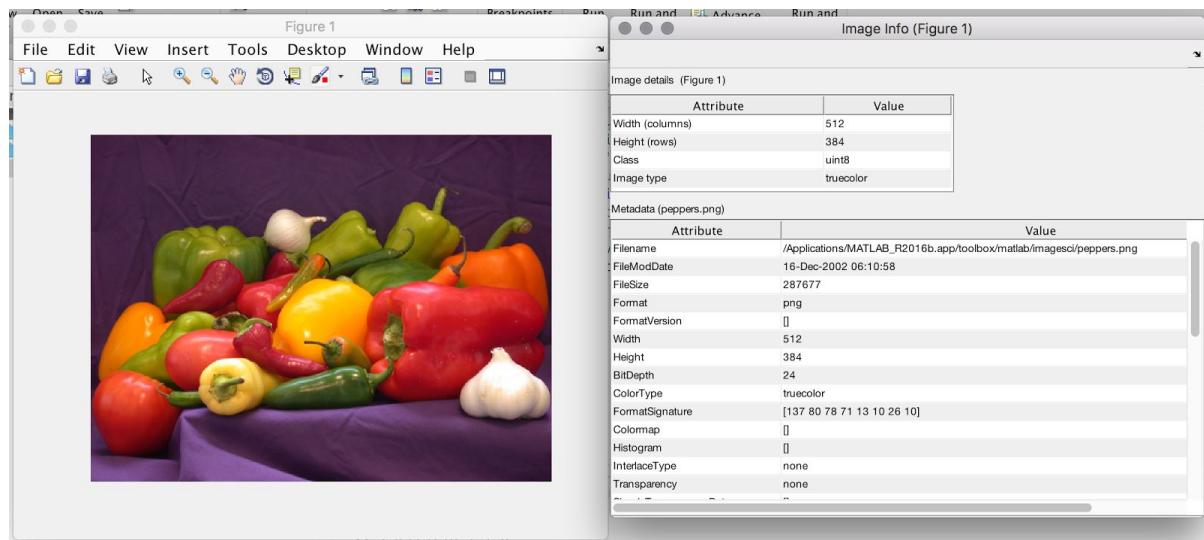
Si noti come l'utilizzo dei parametri 'InitialMagnification' e 'fit' nella funzione imshow() hanno permesso di visualizzare l'immagine a dimensione aumentata (nel caso della visualizzazione del LUT dell'immagine).

## Punto (b)

### Codice

```
% utilizzo imread per caricare l'immagine
img = imread('peppers.png');
% utilizzo imshow per visualizzare l'immagine
imshow(img, 'InitialMagnification', 'fit');
% utilizzo imfinfo per ottenere i metadati dell'immagine
metadati = imfinfo('peppers.png');
% utilizzo imageinfo per visualizzare i metadati dell'immagine
imageinfo(imshow, metadati);
```

## Risultati



## Punto (c)

### Codice

Visualizzazione di ciascuna componente RGB:

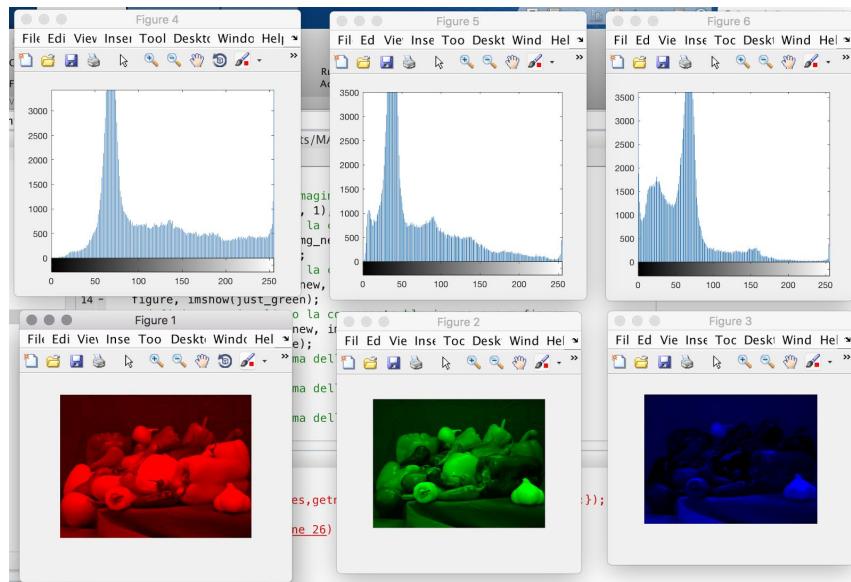
```
% utilizzo imread per caricare l'immagine
img = imread('peppers.png');
% estraggo le componenti rgb dall'immagine
red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);
% definisco una nuova immagine con le dimensioni di img
img_new = zeros(size(img, 1), size(img, 2));
% definisco e visualizzo la componente rossa in una nuova figura
just_red = cat(3, red, img_new, img_new);
figure, imshow(just_red);
% definisco e visualizzo la componente verde in una nuova figura
just_green = cat(3, img_new, green, img_new);
figure, imshow(just_green);
% definisco e visualizzo la componente blu in una nuova figura
just_blue = cat(3, img_new, img_new, blue);
figure, imshow(just_blue);
```

c

Visualizzazione dei relativi istogrammi dei livelli di intensità:

```
% visualizzo l'istogramma della componente rossa
figure, imhist(red);
% visualizzo l'istogramma della componente verde
figure, imhist(green);
% visualizzo l'istogramma della componente blu
figure, imhist(blue);
```

## Risultati

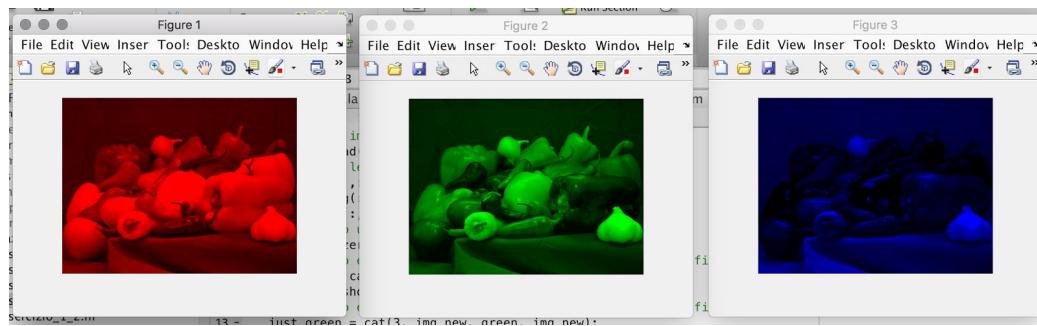


## Punto (d)

## Codice

```
% utilizzo imread per caricare l'immagine
img = imread('peppers.png');
% estraigo le componenti rgb dall'immagine
red = img(:,:,1);
green = img(:,:,2);
blue = img(:,:,3);
% definisco una nuova immagine con le dimensioni di img
img_new = zeros(size(img, 1), size(img, 2));
% definisco e visualizzo la componente rossa in una nuova figura
just_red = cat(3, red, img_new, img_new);
figure, imshow(just_red);
% definisco e visualizzo la componente verde in una nuova figura
just_green = cat(3, img_new, green, img_new);
figure, imshow(just_green);
% definisco e visualizzo la componente blu in una nuova figura
just_blue = cat(3, img_new, img_new, blue);
figure, imshow(just_blue);
```

## Risultati



## Esercizio 1.2

A partire dall'immagine sintetica della barra, effettuare le operazioni seguenti:

- Aggiungere rumore gaussiano a media nulla, con tre livelli diversi di varianza: 0.01, 0.1, 1. Visualizzare l'immagine sorgente e le tre immagini rumorose. Per la simulazione del rumore usare la funzione MATLAB imnoise()
- Rispetto all'immagine senza rumore presa come riferimento, calcolare l'errore quadratico medio MSE (funzione immse()); il rapporto segnale-rumore di picco (PSNR); il rapporto segnale rumore (SNR) per tutti i livelli di rumore simulato;
- Visualizzare i risultati in forma di tabelle (funzione table()).

### Codice

Aggiunta del rumore gaussiano nullo con i diversi livelli di varianza e visualizzazione immagini:

```
 function esercizio_1_2()
% costruisco l'immagine
img = zeros(50, 50);
img(10:40, 20:30) = 1.0;
% definisco le immagini con i tre livelli di varianza
img_noise_1 = imnoise(img, 'gaussian', 0, 0.01);
img_noise_2 = imnoise(img, 'gaussian', 0, 0.1);
img_noise_3 = imnoise(img, 'gaussian', 0, 1);
% visualizzo le immagini
subplot(1, 4, 1);
imshow(img, 'InitialMagnification', 'fit')
subplot(1, 4, 2);
imshow(img_noise_1, 'InitialMagnification', 'fit')
subplot(1, 4, 3);
imshow(img_noise_2, 'InitialMagnification', 'fit')
subplot(1, 4, 4);
imshow(img_noise_3, 'InitialMagnification', 'fit')
```

Calcolo dell'errore quadratico medio e del rapporto segnale-rumore di picco:

```
% calcolo l'errore quadratico medio (MSE)
mse_1 = immse(img_noise_1, img);
mse_2 = immse(img_noise_2, img);
mse_3 = immse(img_noise_3, img);
% calcolo il rapporto segnale-rumore di picco (PSNR)
psnr_1 = 10*log10(255*255/mse_1);
psnr_2 = 10*log10(255*255/mse_2);
psnr_3 = 10*log10(255*255/mse_3);
```

Calcolo del rapporto segnale-rumore:

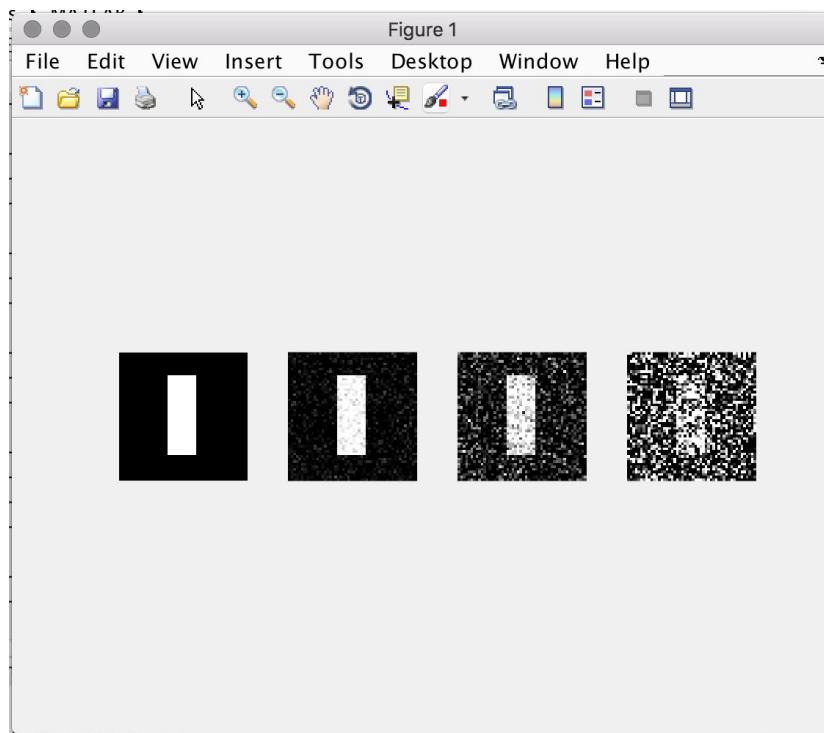
```
% calcolo il rapporto segnale-rumore (SNR) (utilizzo funzione 2 lez. 5)
% -- identifico le regioni sulle quali eseguire le stime
region_SN = [10:14,20:24];
region_N = [1:5,1:5];
% -- stimo il livello medio di grigio delle regioni segnale-rumore
est_aver_gray_1 = mean2(img_noise_1(region_SN));
est_aver_gray_2 = mean2(img_noise_2(region_SN));
est_aver_gray_3 = mean2(img_noise_3(region_SN));
% -- stimo il livello medio di grigio delle regioni con solo rumore
est_std_gray_1 = mean2(img_noise_1(region_N));
est_std_gray_2 = mean2(img_noise_2(region_N));
est_std_gray_3 = mean2(img_noise_3(region_N));
% -- calcolo dell'SNR
snr_1 = est_aver_gray_1/est_std_gray_1;
snr_2 = est_aver_gray_2/est_std_gray_2;
snr_3 = est_aver_gray_3/est_std_gray_3;
```

Visualizzazione dei dati in tabella:

```
% visualizzo i dati sotto forma di tabella
Noises = {'Noise 0.01'; 'Noise 0.1'; 'Noise 1'};
Mse = [mse_1; mse_2; mse_3];
Psnr = [psnr_1; psnr_2; psnr_3];
Snr = [snr_1; snr_2; snr_3];

disp(table(Mse, Psnr, Snr, 'RowNames', Noises));
end
```

## Risultati



>> esercizio\_1\_2

	Mse	Psnr	Snr
<b>Noise 0.01</b>	<b>0.0048613</b>	<b>71.263</b>	<b>17.267</b>
<b>Noise 0.1</b>	<b>0.051291</b>	<b>61.03</b>	<b>0.59492</b>
<b>Noise 1</b>	<b>0.25601</b>	<b>54.048</b>	<b>0.79278</b>

## Esercizio 2.1

Con riferimento alla Lezione 2.1 si consideri l'immagine 'pout.tif'.

- (a) Applicare all'immagine l'operatore 2 a) di variazione dell'intensità, con tre valori del parametro L: 30, 60, 90.
- (b) Applicare l'operatore 2 b) con tre valori del parametro P: 0.5, 1.5, 2.
- (c) Calcolare il negativo dell'immagine usando l'operatore 2 d).

Tutte le immagini devono essere visualizzate assieme al rispettivo istogramma dei livelli di grigio.

Strutturare il codice MATLAB in moduli: un unico script richiama tre procedure, ciascuna per rispondere ai punti 1., 2., 3. rispettivamente.

Discutere brevemente i risultati di ciascun punto e riportare la discussione nelle osservazioni finali.

### Codice

Funzione per l'esecuzione del punto (a):

```
function esercizio_2_1_a(img)
    % definisco immagine con applicazione dei vari livelli di intensità
    img_30 = img + 30;
    img_60 = img + 60;
    img_90 = img + 90;
    % visualizzo i risultati
    figure;
    subplot(2, 4, 1);
    imshow(img, 'InitialMagnification', 'fit')
    subplot(2, 4, 2);
    imshow(img_30, 'InitialMagnification', 'fit')
    subplot(2, 4, 3);
    imshow(img_60, 'InitialMagnification', 'fit')
    subplot(2, 4, 4);
    imshow(img_90, 'InitialMagnification', 'fit')
    subplot(2, 4, 5);
    imhist(img), grid;
    subplot(2, 4, 6);
    imhist(img_30), grid;
    subplot(2, 4, 7);
    imhist(img_60), grid;
    subplot(2, 4, 8);
    imhist(img_90), grid;
end
```

Funzione per l'esecuzione del punto (b):

```
function esercizio_2_1_b(img)
    % definisco immagine con applicazione dei vari contrasti
    img_0_5 = img * 0.5;
    img_1_5 = img * 1.5;
    img_2 = img * 2;
    % visualizzo i risultati
    figure;
    subplot(2, 4, 1);
    imshow(img, 'InitialMagnification', 'fit')
    subplot(2, 4, 2);
    imshow(img_0_5, 'InitialMagnification', 'fit')
    subplot(2, 4, 3);
    imshow(img_1_5, 'InitialMagnification', 'fit')
    subplot(2, 4, 4);
    imshow(img_2, 'InitialMagnification', 'fit')
    subplot(2, 4, 5);
    imhist(img), grid;
    subplot(2, 4, 6);
    imhist(img_0_5), grid;
    subplot(2, 4, 7);
    imhist(img_1_5), grid;
    subplot(2, 4, 8);
    imhist(img_2), grid;
end
```

e

Funzione per l'esecuzione del punto (c):

```
function esercizio_2_1_c(img)
    % definisco immagine con applicazione del negativo
    img_neg = 255 - img;
    % visualizzo i risultati
    figure;
    subplot(2, 2, 1);
    imshow(img, 'InitialMagnification', 'fit')
    subplot(2, 2, 2);
    imshow(img_neg, 'InitialMagnification', 'fit')
    subplot(2, 2, 3);
    imhist(img), grid;
    subplot(2, 2, 4);
    imhist(img_neg), grid;
end
```

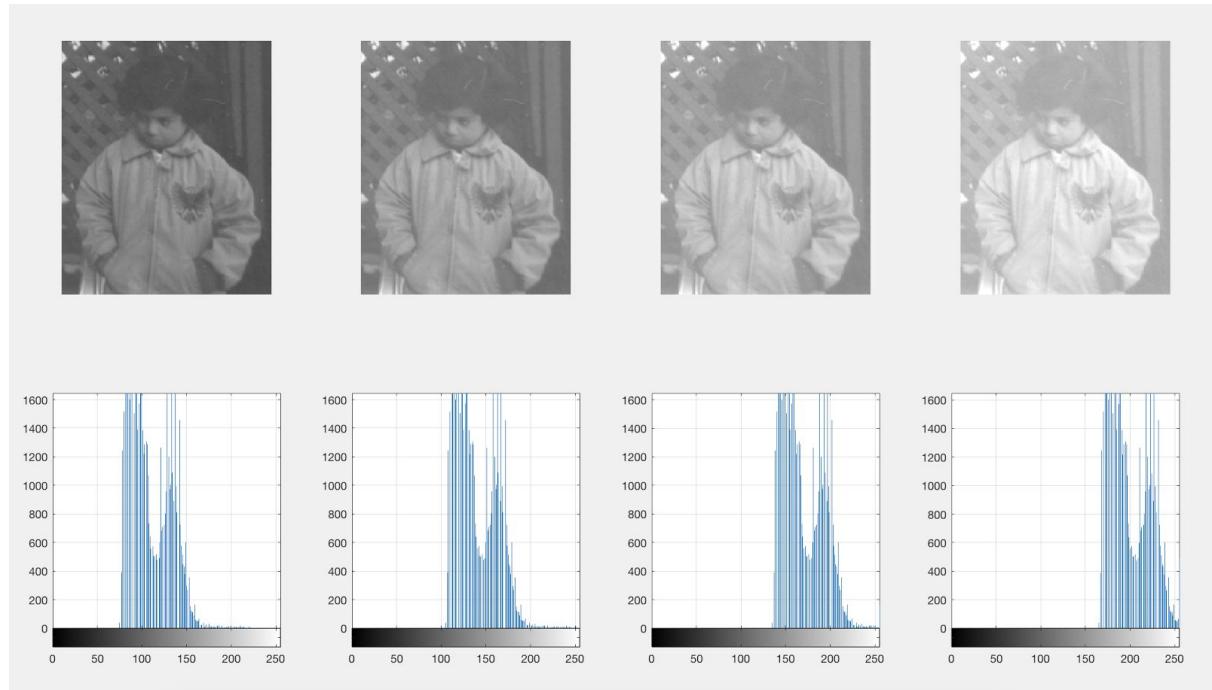
Funzione per chiamare le varie procedure:

---

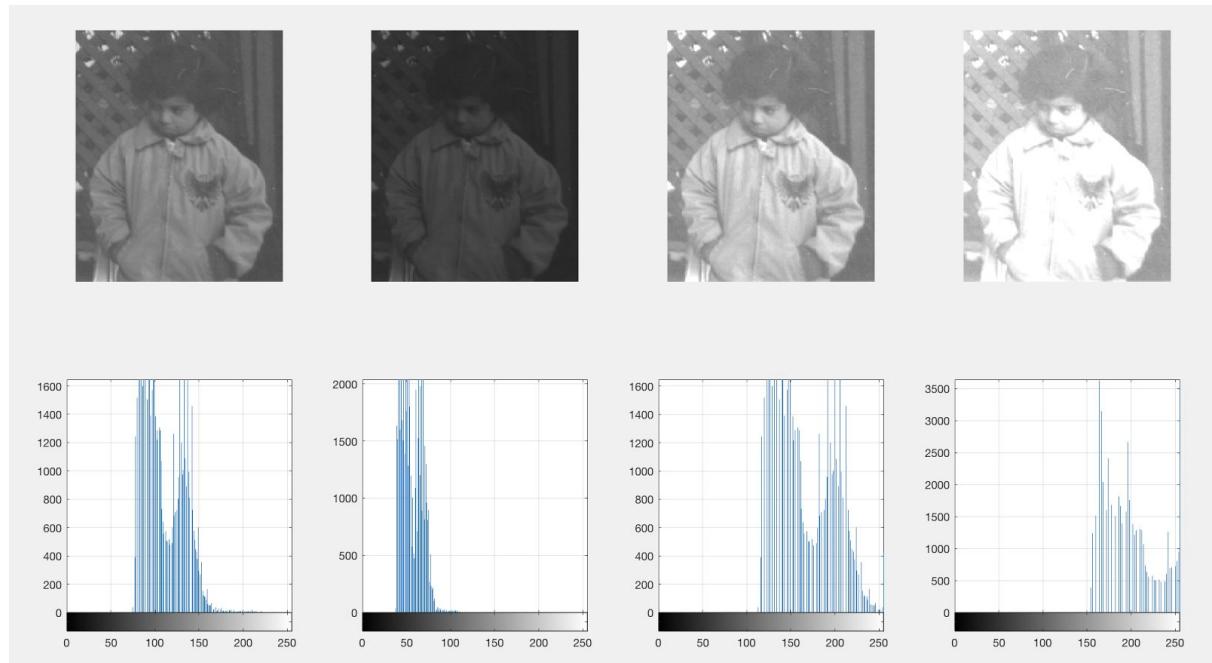
```
function esercizio_2_1()
    % carico l'immagine richiesta
    img = imread('pout.tif');
    % eseguo la prima procedura
    esercizio_2_1_a(img);
    % eseguo la seconda procedura
    esercizio_2_1_b(img);
    % eseguo la terza procedura
    esercizio_2_1_c(img);
end
```

## Risultato

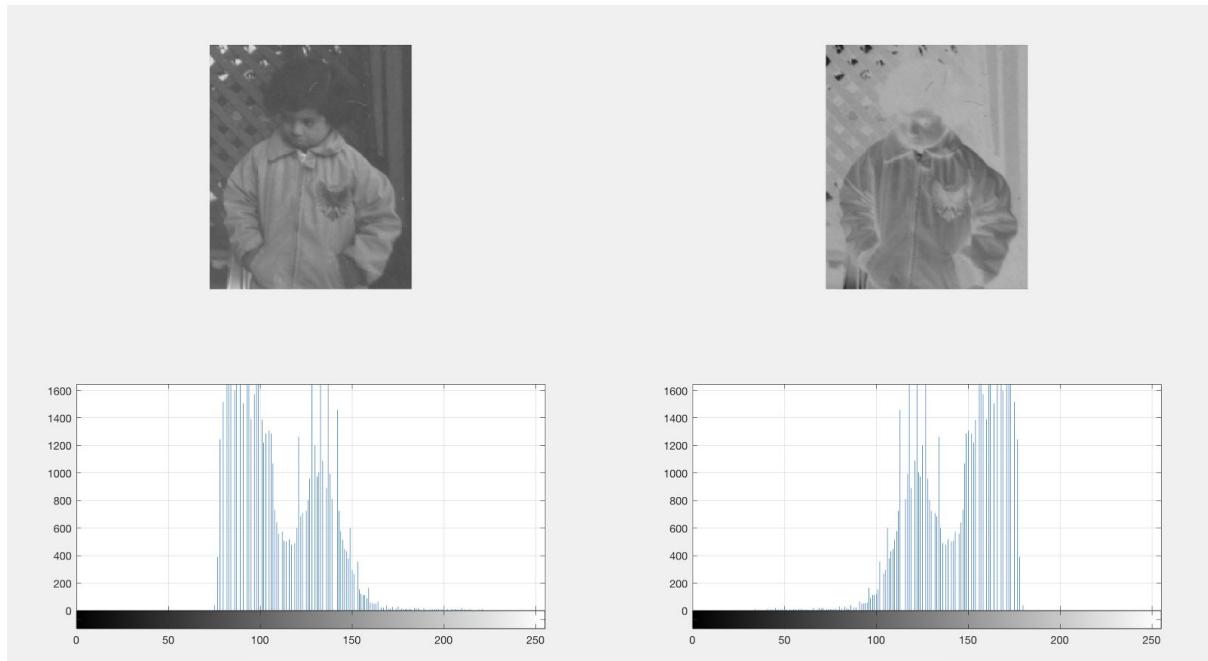
Applicazione delle variazioni di intensità:



Applicazione delle variazioni di contrasto:



Visualizzazione del negativo:



## Osservazioni

Punto (a):

La variazione dell'intensità ha traslato il diagramma verso destra in quanto all'aumento dell'intensità i singoli punti dell'immagine tendono ad avere un colore più chiaro (e quindi tendono verso il bianco).

Punto (b):

La variazione di contrasto ha portato il diagramma a contrarsi con valori inferiori all'uno e ad espandersi con valori superiori.

Punto (c):

L'applicazione dell'immagine negativa ha portato anche l'istogramma a risultare l'opposto dell'immagine originale in quanto i grigi vengono invertiti in ogni punto.

## Esercizio 2.2

Con riferimento alla Lezione 2.1 si consideri l'immagine sintetica della barra come sorgente.

- Applicare all'immagine sorgente rumore additivo gaussiano a media nulla e varianza 0.1. Generare complessivamente cinque immagini affette dallo stesso tipo di rumore. Associare a ciascuna immagine il proprio istogramma.
- Calcolare la media pixel per pixel (stacking) delle cinque immagini con rumore.
- Stimare il MSE, rispetto all'immagine sorgente, di ciascuna delle immagini rumorose e dell'immagine calcolata al punto 2.

I valori del MSE devono essere riportati in tabella. Discutere brevemente i risultati e riportare la discussione nelle osservazioni finali.

### Codice

Costruzione delle 5 immagini con il rumore:

```
function esercizio_2_2()
% costruisco l'immagine
img = zeros(50, 50);
img(10:40, 20:30) = 1.0;
% genero le 5 immagini con il rumore
img_noise_1 = imnoise(img, 'gaussian', 0, 0.1);
img_noise_2 = imnoise(img, 'gaussian', 0, 0.1);
img_noise_3 = imnoise(img, 'gaussian', 0, 0.1);
img_noise_4 = imnoise(img, 'gaussian', 0, 0.1);
img_noise_5 = imnoise(img, 'gaussian', 0, 0.1);
```

Visualizzazione delle immagini insieme all'istogramma:

```
% visualizzo le 5 immagini con il relativo istogramma
% -- riga 1
figure;
subplot(4, 3, 1);
imshow(img, 'InitialMagnification', 'fit')
subplot(4, 3, 2);
imshow(img_noise_1, 'InitialMagnification', 'fit')
subplot(4, 3, 3);
imshow(img_noise_2, 'InitialMagnification', 'fit')
% -- riga 2
subplot(4, 3, 4);
imhist(img), grid;
subplot(4, 3, 5);
imhist(img_noise_1), grid;
subplot(4, 3, 6);
imhist(img_noise_2), grid;
% -- riga 3
subplot(4, 3, 7);
imshow(img_noise_3, 'InitialMagnification', 'fit')
subplot(4, 3, 8);
imshow(img_noise_4, 'InitialMagnification', 'fit')
subplot(4, 3, 9);
imshow(img_noise_5, 'InitialMagnification', 'fit')
% -- riga 4
subplot(4, 3, 10);
imhist(img_noise_3), grid;
subplot(4, 3, 11);
imhist(img_noise_4), grid;
subplot(4, 3, 12);
imhist(img_noise_5), grid;
```

Calcolo della media delle 5 immagini e calcolo degli MSE:

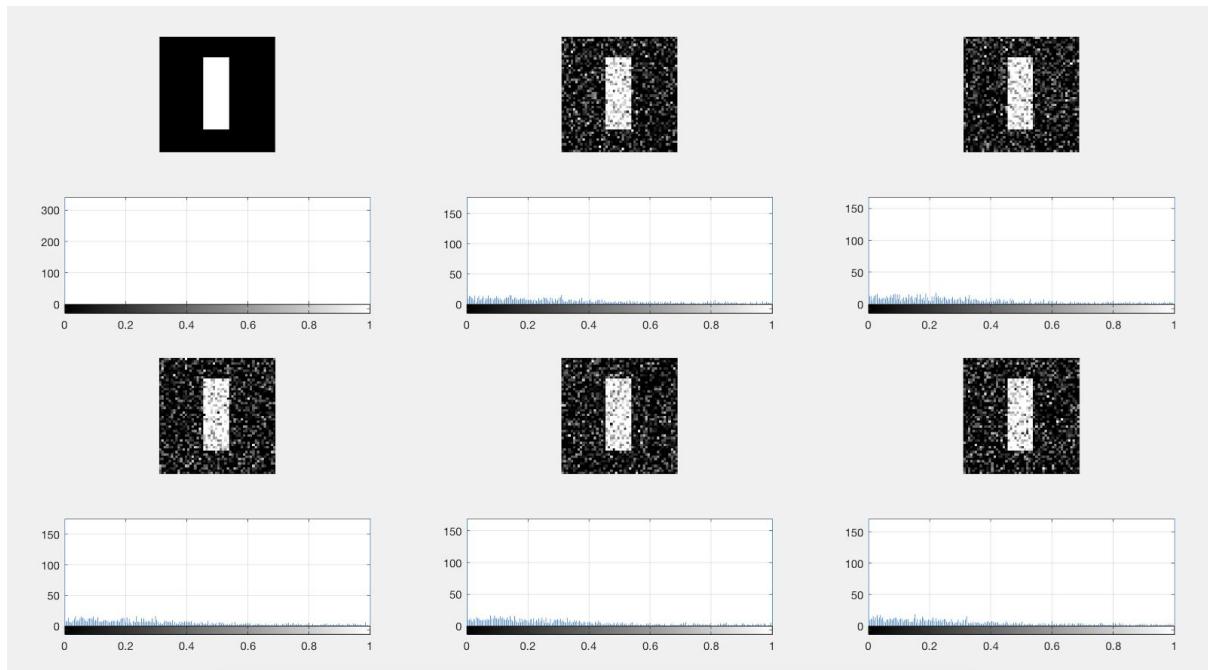
```
% calcolo la media pixel per pixel delle 5 immagini con rumore
img_final = (img_noise_1 + img_noise_2 + img_noise_3 + img_noise_4 + img_noise_5) / 5;
% stimo l'MSE di ciascuna delle immagini rispetto all'immagine sorgente
mse_original = immse(img, img);
mse_1 = immse(img_noise_1, img);
mse_2 = immse(img_noise_2, img);
mse_3 = immse(img_noise_3, img);
mse_4 = immse(img_noise_4, img);
mse_5 = immse(img_noise_5, img);
mse_final = immse(img_final, img);
```

Costruzione della tabella per la visualizzazione dei risultati:

```
% visualizzazione dei risultati in tabella
Images = {'Original'; 'Noise 1'; 'Noise 2'; 'Noise 3'; 'Noise 4'; 'Noise 5'; 'Noise final'};
Mse = [mse_original; mse_1; mse_2; mse_3; mse_4; mse_5; mse_final];
disp(table(Mse, 'RowNames', Images));
end
```

## Risultato

Visualizzazione delle 5 immagini insieme all'originale e i relativi istogrammi:



Visualizzazione della tabella con i risultati degli MSE calcolati:

```
>> esercizio_2_2
      Mse
```

	Mse
Original	0
Noise 1	0.047855
Noise 2	0.048362
Noise 3	0.048452
Noise 4	0.04714
Noise 5	0.049867
Noise final	0.021791

## Osservazioni

Visualizzando i risultati del calcolo dell'errore quadratico medio delle immagini si nota come le cinque immagini affette da rumore hanno tutti valori simili (compresi tra 0.0478 e 0.0498) mentre l'immagine calcolata come la media delle cinque ha un errore quadratico medio pari a 0.0217.

Tali valori dimostrano come l'applicazione del calcolo della media di immagini permetta di ridurre il rumore casuale rispetto all'immagine originale.

## Esercizio 2.3

Con riferimento alla Lezione 2.1 si considerino le immagini 'moon.tif' , da prendere come riferimento, e 'eight.tif' da considerare come maschera.

- Usando la funzione MATLAB imresize() modificare le dimensioni della maschera in modo da renderle uguali a quelle del riferimento.
- Calcolare la combinazione delle due immagini applicando l'operatore 3 c) per cinque valori del parametro  $\alpha$  : 0., 0.25, 0.5, 0.75, 1. Tutte le immagini devono essere visualizzate separatamente assieme al rispettivo istogramma dei livelli di grigio.

Facoltativo: saranno valutati i due possibili sviluppi 3) e 4) qui di seguito.

- Le cinque immagini ottenute al punto 2. potrebbero essere concatenate (procedura cat() di MATLAB) in un collage, cioè un'immagine unica da visualizzare a sua volta;
- Il collage potrebbe essere composto da coppie di immagini diverse da quelle sopra individuate, allo scopo di ottenere risultati efficaci dal punto di vista espressivo.

### Codice

Resize dell'immagine usata come maschera e costruzione delle immagini costruite attraverso l'operatore specificato:

```
function esercizio_2_3()
% carico le due immagini
img = imread('moon.tif');
mask = imread('eight.tif');
% modifco mask per imostargli la stessa dimensione di img
mask = imresize(mask, [size(img, 1) size(img, 2)]);
% genero le immagini combinate con i diversi valori a
img_1 = my_imunion(img, mask, 0);
img_2 = my_imunion(img, mask, 0.25);
img_3 = my_imunion(img, mask, 0.5);
img_4 = my_imunion(img, mask, 0.75);
img_5 = my_imunion(img, mask, 1);
```

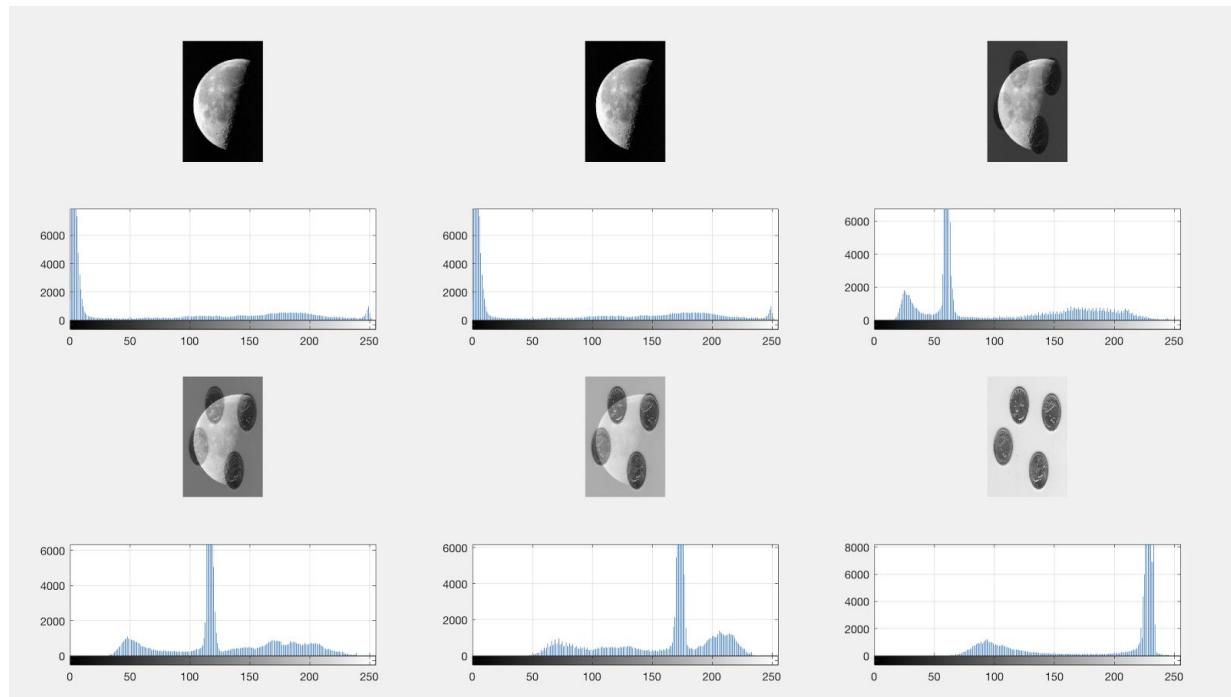
Visualizzazione delle immagini risultanti insieme ai relativi istogrammi:

```
% visualizzo le immagini insieme ai relativi istogrammi
figure;
% -- riga 1
subplot(4, 3, 1);
imshow(img, 'InitialMagnification', 'fit')
subplot(4, 3, 2);
imshow(img_1, 'InitialMagnification', 'fit')
subplot(4, 3, 3);
imshow(img_2, 'InitialMagnification', 'fit')
% -- riga 2
subplot(4, 3, 4);
imhist(img), grid;
subplot(4, 3, 5);
imhist(img_1), grid;
subplot(4, 3, 6);
imhist(img_2), grid;
% -- riga 3
subplot(4, 3, 7);
imshow(img_3, 'InitialMagnification', 'fit')
subplot(4, 3, 8);
imshow(img_4, 'InitialMagnification', 'fit')
subplot(4, 3, 9);
imshow(img_5, 'InitialMagnification', 'fit')
% -- riga 4
subplot(4, 3, 10);
imhist(img_3), grid;
subplot(4, 3, 11);
imhist(img_4), grid;
subplot(4, 3, 12);
imhist(img_5), grid;
end
```

Funzione my\_imunion() utilizzata per applicare l'operatore specificato:

```
% Function my_imunion(img_1, img_2, a)
% This function generates the union of the two images with a specific alpha.
function[img] = my_imunion(img_1, img_2, a)
    if size(img_1, 1) ~= size(img_2, 1) || size(img_1, 2) ~= size(img_2, 2)
        error('images have not the same size');
    end
    img = (1-a) * img_1 + a * img_2;
end
```

## Risultati



## Facoltativo (punto 3)

### Codice

```
function esercizio_2_3_fac_3()
% carico le due immagini
img = imread('moon.tif');
mask = imread('eight.tif');
% modifoco mask per imostargli la stessa dimensione di img
mask = imresize(mask, [size(img, 1) size(img, 2)]);
% genero l'immagine concatenando le varie immagini con alpha diverso
img_final = my_imunion(img, mask, 0);
for alpha = 0.25:0.25:1
    img_alpha = my_imunion(img, mask, alpha);
    img_final = cat(2, img_final, img_alpha);
end
% visualizzo l'immagine finale
figure, imshow(img_final);
end
```

## Risultati



## Esercizio 2.4

Con riferimento alla Lezione 2.2 si consideri l'immagine 'pout.tif'.

- Trasformare l'immagine in un'altra applicando la trasformazione lineare 1a) della Lezione 2.2. Visualizzare le immagini d'ingresso e uscita assieme ai loro istogrammi.
- Visualizzare tramite plot() la dipendenza tra le scale di uscita e d'ingresso outputSc(lev) e inputSc(lev).

### Codice

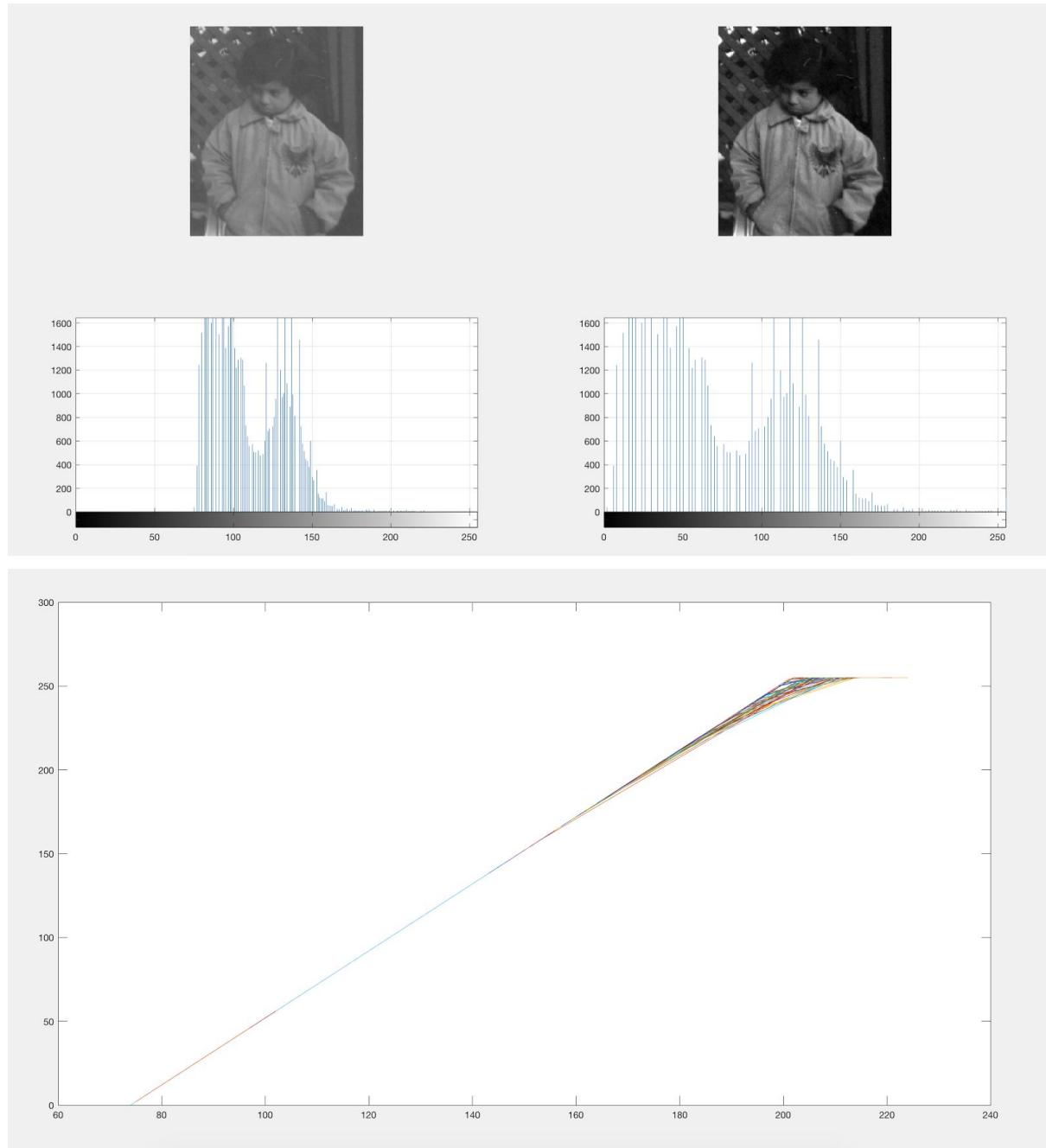
Trasformazione dell'immagine e visualizzazione dell'input e dell'output:

```
% function esercizio_2_4()
% carico l'immagine iniziale
img = imread('pout.tif');
% genero l'immagine trasformata
min_gray_level = min(min(img));
max_gray_level = max(max(img));
img_final = 255 / (max_gray_level - min_gray_level) * (img - min_gray_level);
% visualizzo le due immagini con i relativi istogrammi
figure;
% -- riga 1
subplot(2, 2, 1);
imshow(img, 'InitialMagnification', 'fit')
subplot(2, 2, 2);
imshow(img_final, 'InitialMagnification', 'fit')
% -- riga 2
subplot(2, 2, 3);
imhist(img), grid;
subplot(2, 2, 4);
imhist(img_final), grid;
```

Visualizzazione della dipendenza tra le scale di uscita e di ingresso:

```
% visualizzazione su plot
figure, plot(img, img_final);
end
```

## Risultati



## Osservazioni

Come si nota dagli istogrammi, l'applicazione della trasformazione puntuale lineare ha effettuato uno stiramento dell'istogramma dei livelli di grigio sull'intero range dinamico reso disponibile dalla codifica (0-255).

## Esercizio 2.5

Con riferimento alla Lezione 2.2 si consideri l'immagine 'moon.tif' .

- Definire numericamente le due scale  $\text{inputSc}(\text{lev})$  e  $\text{outputSc}(\text{lev})$ ,  $0 \leq \text{lev} \leq 1$  La  $\text{inputSc}(\text{lev})$  ha due punti discontinuità in  $R_1, R_2$ ; La  $\text{outputSc}(\text{lev})$  è non-nulla solo in  $[R_1 R_2]$ ; è nulla altrove in  $[0 1]$  L'andamento della funzione di trasformazione tra le scale è riportato in Figura 1. Effettuare il calcolo per i seguenti valori dei parametri:  $R_1=0.4$ ;  $R_2=0.6$ ;  $S_2=1$ ; nel sottointervallo  $[0 R_1]$  è  $S_1=0$ ; nel sottointervallo  $[R_2 1]$  è  $S_3=0$ . Visualizzare tramite `plot()` di MATLAB la dipendenza tra  $\text{outputSc}(\text{lev})$  e  $\text{inputSc}(\text{lev})$
- Trasformare i livelli di grigio dell'immagine d'ingresso  $f(i,j)$  applicando le regole definite al precedente punto 1. L'implementazione può essere effettuata tramite istruzioni di condizione `if/elseif`. Visualizzare le immagini di ingresso e uscita dopo la trasformazione.
- Ripetere il calcolo e la visualizzazione per le bande di livelli di grigio seguenti:  $[R_1 R_2] = [0. 0.2] ; [0.2 0.4] ; [0.4 0.6]$  già calcolato;  $[0.6 0.8] ; [0.8 1.]$  Il parametro  $S_2 = 1$ . in tutti i sottointervalli precedenti. Al di fuori di questi è  $S_1=S_3=0$  in tutti i casi.

Tutte le immagini devono essere visualizzate assieme al rispettivo istogramma dei livelli di grigio.

## Esercizio 2.6

Con riferimento alla Lezione 2.2 si consideri l'immagine ‘westconcordaerial.png’.

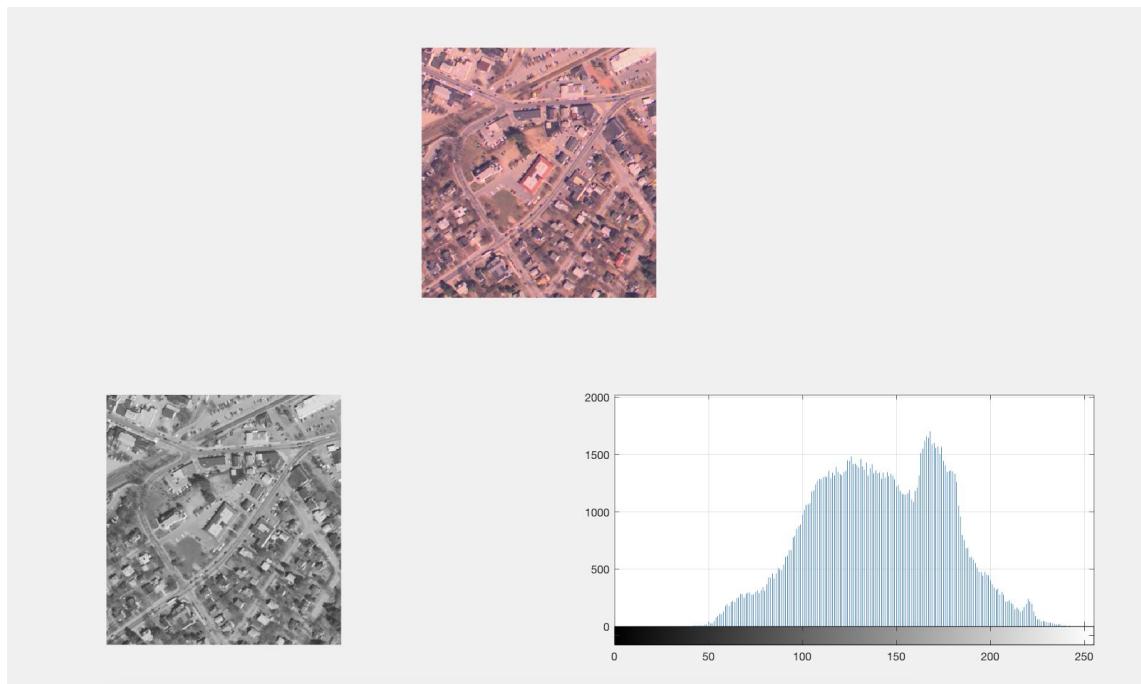
- Visualizzare l'immagine sorgente. Convertire l'immagine in un'altra a livelli di grigio tramite la procedura `rgb2gray()`. Visualizzare l'immagine ottenuta e il relativo istogramma.
- Usare la procedure MATLAB `()` per partizionare l'immagine a livelli di grigio in 16 bande. Visualizzare l'immagine partizionata usando la colormap `jet()`.

Facoltativo. Navigare in Google Earth relativamente a una zona del Friuli-Venezia Giulia opp. del Veneto. Usare lo zoom per ottenere un'immagine più ravvicinata della zona che si è scelta. Effettuare le operazioni ai punti 1., 2. sopra specificati, eventualmente modificando il numero di bande opp. la colormap in modo da ottenere risultati visivamente utili.

### Codice

```
function esercizio_2_6()
% carico l'immagine
img = imread('westconcordaerial.png');
% genero l'immagine convertita
img_gray = rgb2gray(img);
% visualizzo i risultati
figure;
% -- riga 1
subplot(2, 1, 1);
imshow(img, 'InitialMagnification', 'fit');
% -- riga 2
subplot(2, 2, 3);
imshow(img_gray, 'InitialMagnification', 'fit');
subplot(2, 2, 4);
imhist(img_gray), grid;
% partiziono l'immagine a livelli di grigio in 16 bande
img_slice = grayslice(img_gray, 16);
% visualizzo l'immagine partizionata con la colormap jet
figure, imshow(img_slice, jet, 'InitialMagnification', 'fit');
end
```

## Risultato

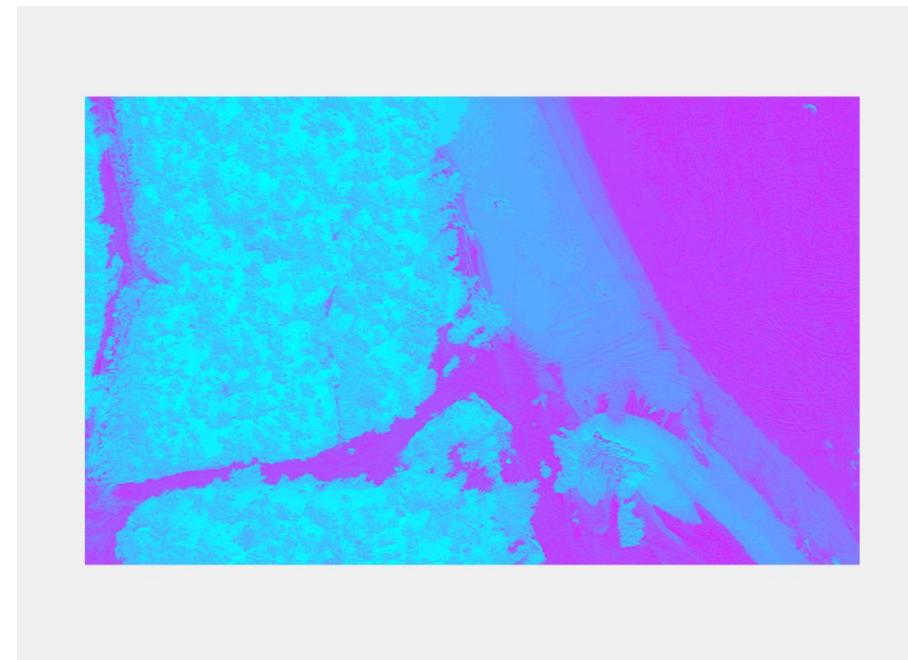
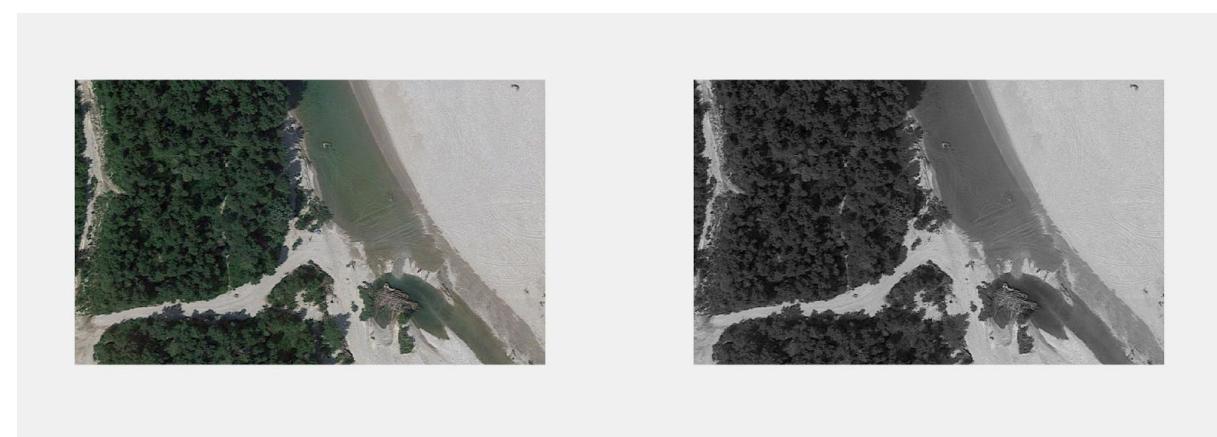


## Facoltativo

### Codice

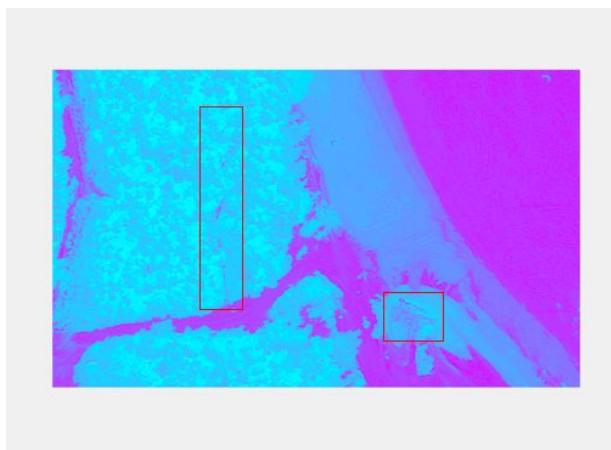
```
function esercizio_2_6_fac()
% carico l'immagine
img = imread('tagliamento.png');
% identifico l'immagine gray
img_gray = rgb2gray(img);
% visualizzo le due immagini
figure;
subplot(1, 2, 1);
imshow(img, 'InitialMagnification', 'fit');
subplot(1, 2, 2);
imshow(img_gray, 'InitialMagnification', 'fit');
% partiziono l'immagine a livelli di grigio in 16 bande
img_slice = grayslice(img_gray, 16);
% visualizzo l'immagine partizionata
figure, imshow(img_slice, cool(16), 'InitialMagnification', 'fit');
end
```

### Risultati



## Osservazioni

Il partizionamento dell'immagine a livelli di grigio in 16 bande e la visualizzazione ha permesso di riconoscere più facilmente nell'immagine il percorso del sentiero in mezzo ai boschi e la forma della ramaglia posta nel letto del fiume (sotto evidenziati in rosso).



## Esercizio 2.7

Con riferimento alla Lezione 2.2 si consideri l'immagine 'cameraman.tif'.

- Selezionare gli otto piani di bit usando la procedura MATLAB bitget() Visualizzare gli otto piani di bit dell'immagine.
- Passo di verifica. Partendo dal piano del bit 1 (il meno significativo), generare i piani di bit moltiplicando il piano n-esimo per il fattore 2  $n-1$  dove  $n=1,\dots,8$  Verificare, calcolando le differenze, che i piani così generati coincidano con quelli ricavati al punto 1.
- Ricostruire l'immagine originale a partire dai suoi piani di bit calcolati al punto 2. Verificare, calcolando la differenza, che l'immagine originale e quella ricostruita coincidono. Ogni immagine deve essere associata al proprio istogramma dei livelli di grigio.
- Effettuare ricostruzioni parziali dell'originale sommando soltanto alcuni piani di bit, e valutare quantitativamente la perdita di informazione tramite stima del MSE nelle situazioni seguenti: (a) immagine approssimata dal solo piano del bit 8 (il più significativo); (b) sommando soltanto i piani 7,8; (c) i piani 6,7,8; (d) i piani 5,6,7,8.  
Riportare i risultati tramite table().

### Codice

Selezione degli 8 piani di bit dell'immagine:

```
function esercizio_2_7()
% carico l'immagine
img = imread('cameraman.tif');
% selezioni gli otto piani di bit dell'immagine
img_bit_1 = bitget(img, 1);
img_bit_2 = bitget(img, 2);
img_bit_3 = bitget(img, 3);
img_bit_4 = bitget(img, 4);
img_bit_5 = bitget(img, 5);
img_bit_6 = bitget(img, 6);
img_bit_7 = bitget(img, 7);
img_bit_8 = bitget(img, 8);
```

Visualizzazione degli 8 piani di bit ottenuti:

```
% visualizzo gli otto piani di bit
figure;
% -- riga 1
subplot(2, 4, 1);
imshow(logical(img_bit_1), 'InitialMagnification', 'fit');
subplot(2, 4, 2);
imshow(logical(img_bit_2), 'InitialMagnification', 'fit');
subplot(2, 4, 3);
imshow(logical(img_bit_3), 'InitialMagnification', 'fit');
subplot(2, 4, 4);
imshow(logical(img_bit_4), 'InitialMagnification', 'fit');
% -- riga 2
subplot(2, 4, 5);
imshow(logical(img_bit_5), 'InitialMagnification', 'fit');
subplot(2, 4, 6);
imshow(logical(img_bit_6), 'InitialMagnification', 'fit');
subplot(2, 4, 7);
imshow(logical(img_bit_7), 'InitialMagnification', 'fit');
subplot(2, 4, 8);
imshow(logical(img_bit_8), 'InitialMagnification', 'fit');
```

Generazione e dei piani di bit e verifica delle delle differenze rispetto a quelli ottenuti con bitget():

```
% genero i piani di bit partendo da quello meno significativo
img_bit_gen_1 = img_bit_1 * (2^(0));
img_bit_gen_2 = img_bit_2 * (2^(1));
img_bit_gen_3 = img_bit_3 * (2^(2));
img_bit_gen_4 = img_bit_4 * (2^(3));
img_bit_gen_5 = img_bit_5 * (2^(4));
img_bit_gen_6 = img_bit_6 * (2^(5));
img_bit_gen_7 = img_bit_7 * (2^(6));
img_bit_gen_8 = img_bit_8 * (2^(7));
% verifico che i piani di bit generati coincidano con gli originali
if img_bit_gen_1 - img_bit_1 ~= 0; error('images are not the same'); end;
if img_bit_gen_2 - img_bit_2 ~= 0; error('images are not the same'); end;
if img_bit_gen_3 - img_bit_3 ~= 0; error('images are not the same'); end;
if img_bit_gen_4 - img_bit_4 ~= 0; error('images are not the same'); end;
if img_bit_gen_5 - img_bit_5 ~= 0; error('images are not the same'); end;
if img_bit_gen_6 - img_bit_6 ~= 0; error('images are not the same'); end;
if img_bit_gen_7 - img_bit_7 ~= 0; error('images are not the same'); end;
if img_bit_gen_8 - img_bit_8 ~= 0; error('images are not the same'); end;
```

Rigenerazione dell'immagine originale a partire dai singoli piani di bit:

```
% rigenero l'immagine originale a partire dai singoli piani di bit
img_final = zeros(size(img));
img_final = bitset(img_final, 1, img_bit_gen_1);
img_final = bitset(img_final, 2, img_bit_gen_2);
img_final = bitset(img_final, 3, img_bit_gen_3);
img_final = bitset(img_final, 4, img_bit_gen_4);
img_final = bitset(img_final, 5, img_bit_gen_5);
img_final = bitset(img_final, 6, img_bit_gen_6);
img_final = bitset(img_final, 7, img_bit_gen_7);
img_final = bitset(img_final, 8, img_bit_gen_8);
img_final = uint8(img_final);
% verifico che l'immagine originale e quella generata coincidano
if img_final - img ~= 0; error('images are not the same'); end;
```

Visualizzazione dell'immagine originale e quella rigenerata insieme all'istogramma:

```
% visualizzo l'immagine originale e quella calcolata con i relativi
% istogrammi di grigio
figure;
% -- riga 1
subplot(2, 2, 1);
imshow(img, 'InitialMagnification', 'fit'), title('Originale');
subplot(2, 2, 2);
imshow(img_final, 'InitialMagnification', 'fit'), title('Generata');
% -- riga 2
subplot(2, 2, 3);
imhist(img), grid;
subplot(2, 2, 4);
imhist(img_final), grid;
```

Ricostruzioni parziali dell'immagine utilizzando solo i piani richiesti e calcolo dell'MSE:

```
% eseguo la ricostruzione parziale con il solo piano di bit 8
img_partial_8 = zeros(size(img));
img_partial_8 = bitset(img_partial_8, 8, img_bit_gen_8);
img_partial_8 = uint8(img_partial_8);
mse_partial_8 = immse(img_partial_8, img);
% eseguo la ricostruzione parziale con i piani di bit 7 e 8
img_partial_78 = zeros(size(img));
img_partial_78 = bitset(img_partial_78, 7, img_bit_gen_7);
img_partial_78 = bitset(img_partial_78, 8, img_bit_gen_8);
img_partial_78 = uint8(img_partial_78);
mse_partial_78 = immse(img_partial_78, img);
% eseguo la ricostruzione parziale con i piani di bit 6, 7 e 8
img_partial_678 = zeros(size(img));
img_partial_678 = bitset(img_partial_678, 6, img_bit_gen_6);
img_partial_678 = bitset(img_partial_678, 7, img_bit_gen_7);
img_partial_678 = bitset(img_partial_678, 8, img_bit_gen_8);
img_partial_678 = uint8(img_partial_678);
mse_partial_678 = immse(img_partial_678, img);
% eseguo la ricostruzione parziale con i piani di bit 5, 6, 7 e 8
img_partial_5678 = zeros(size(img));
img_partial_5678 = bitset(img_partial_5678, 5, img_bit_gen_5);
img_partial_5678 = bitset(img_partial_5678, 6, img_bit_gen_6);
img_partial_5678 = bitset(img_partial_5678, 7, img_bit_gen_7);
img_partial_5678 = bitset(img_partial_5678, 8, img_bit_gen_8);
img_partial_5678 = uint8(img_partial_5678);
mse_partial_5678 = immse(img_partial_5678, img);
```

Visualizzazione dei valori MSE calcolati in tabella:

```
% visualizzo i vari MSE ottenuti in tabella
Levels = {'Bit 8'; 'Bit 7, 8'; 'Bit 6, 7, 8'; 'Bit 5, 6, 7, 8'};
Mse = [mse_partial_8; mse_partial_78; mse_partial_678; mse_partial_5678];
disp(table(Mse, 'RowNames', Levels));
end
```

## Risultati

Visualizzazione dei singoli piani di bit:



Visualizzazione dell'immagine originale e quella rigenerata con i relativi istogrammi:

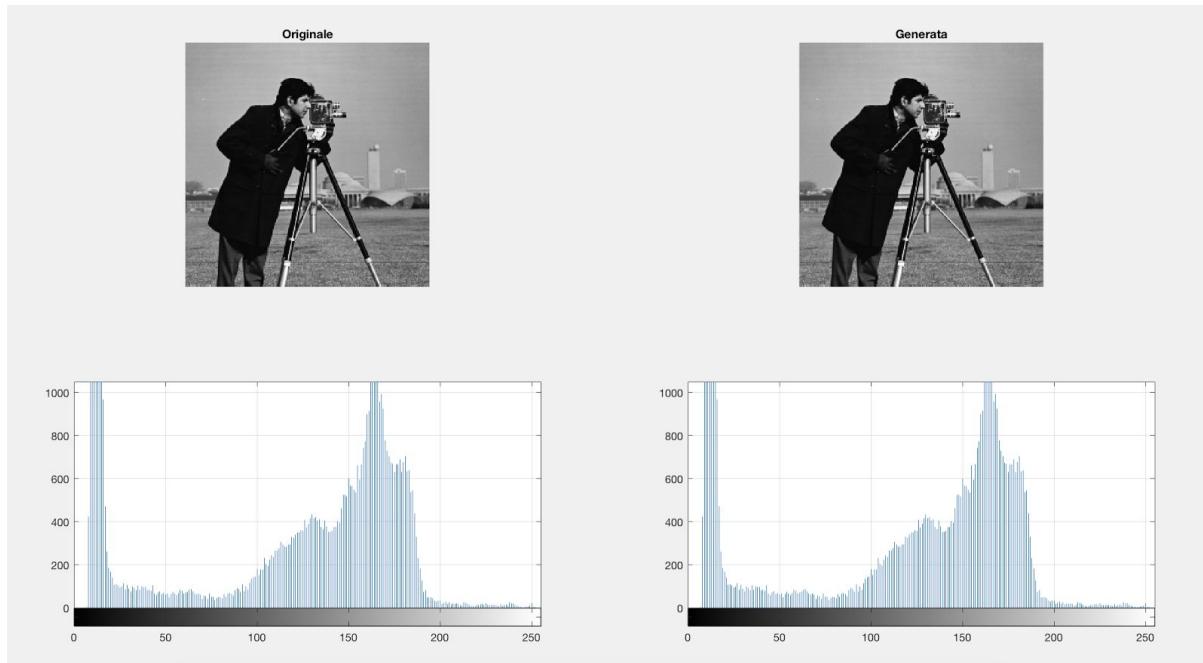


Tabella per la visualizzazione degli MSE calcolati sulle immagini parziali:

>> esercizio\_2\_7

**Mse**

	<hr/>
<b>Bit 8</b>	2924.7
<b>Bit 7, 8</b>	1263.3
<b>Bit 6, 7, 8</b>	286.54
<b>Bit 5, 6, 7, 8</b>	82.156

## Esercizio 2.8

E' data l'immagine sintetica della scacchiera (checkerboard), che deve essere rappresentata in forma binaria, di dimensioni complessive 256x256. Applicare alla scacchiera rumore additivo gaussiano a media nulla, una volta con varianza 0,1 (rumore basso); un'altra con varianza 1 (rumore alto).

- Visualizzare l'immagine sorgente della scacchiera e le due immagini con rumore, assieme ai rispettivi istogrammi dei livelli di grigio;
- Effettuare il filtraggio per la riduzione del rumore tramite filtro di media (average) con maschera 3x3. Visualizzare le seguenti immagini: originale; affetta da rumore basso; affetta da rumore alto; immagine filtrata dal rumore basso; filtrata dal rumore alto; tutti gli istogrammi relativi;
- Il filtraggio è stato utile a ridurre il rumore, sia basso che alto? Come si può stimare quantitativamente l'efficacia del filtraggio? Applicare la stima e calcolare l'efficacia del filtraggio. Riportarla in una tabella;
- Ripetere i calcoli effettuando il filtraggio solo nel caso di rumore basso, con lo stesso operatore di media (average) e maschere di dimensione 5x5 e 7x7; visualizzare le immagini filtrate e i rispettivi istogrammi;
- Confrontare il risultato dei filtri ottenuti con le tre finestre: 3x3, 5x5, 7x7, a rumore basso. Riportare le stime in una tabella. Quale finestra ha dato il risultato migliore?
- Riportare i risultati dei confronti nelle osservazioni finali. Usare le seguenti primitive MATLAB: checkerboard(), imnoise(), fspecial(), imfilter(), immse(), table().

Facoltativo: ripetere i calcoli effettuati ai punti 4 e 5 applicandoli anche al caso di rumore alto. Ricavare la nuova tabella e le rispettive conclusioni.

### Codice

Generazione delle immagini con rumore:

```
function esercizio_2_8()
% genero la scacchiera
board = double(checkerboard(32)>0.5);
% genero le due board affette da rumore
board_noise_1 = imnoise(board, 'gaussian', 0, 0.1);
board_noise_2 = imnoise(board, 'gaussian', 0, 1);
```

Visualizzazione delle immagini con rumore e rispettivi istogrammi:

```
% visualizzo le tre immagini con i rispettivi istogrammi
figure;
% -- riga 1
subplot(2, 3, 1);
imshow(board, 'InitialMagnification', 'fit'), title('Originale');
subplot(2, 3, 2);
imshow(board_noise_1, 'InitialMagnification', 'fit'), title('Rumore basso');
subplot(2, 3, 3);
imshow(board_noise_2, 'InitialMagnification', 'fit'), title('Rumore alto');
% -- riga 2
subplot(2, 3, 4);
imhist(board), grid;
subplot(2, 3, 5);
imhist(board_noise_1), grid;
subplot(2, 3, 6);
imhist(board_noise_2), grid;
```

Generazione delle immagini con rumore filtrate con maschera 3x3:

```
% genero le immagini filtrate a partire da quelle affette da rumore
filter = fspecial('average', 3);
board_filter_1 = imfilter(board_noise_1, filter);
board_filter_2 = imfilter(board_noise_2, filter);
```

Visualizzazione delle immagini con rumore e filtrate con i rispettivi istogrammi:

```
% visualizzo le immagini con i relativi istogrammi
figure;
% -- riga 1
subplot(2, 5, 1);
imshow(board, 'InitialMagnification', 'fit'), title('Originale');
subplot(2, 5, 2);
imshow(board_noise_1, 'InitialMagnification', 'fit'), title('Rumore basso');
subplot(2, 5, 3);
imshow(board_noise_2, 'InitialMagnification', 'fit'), title('Rumore alto');
subplot(2, 5, 4);
imshow(board_filter_1, 'InitialMagnification', 'fit'), title('Filtro rumore basso');
subplot(2, 5, 5);
imshow(board_filter_2, 'InitialMagnification', 'fit'), title('Filtro rumore alto');
% -- riga 2
subplot(2, 5, 6);
imhist(board), grid;
subplot(2, 5, 7);
imhist(board_noise_1), grid;
subplot(2, 5, 8);
imhist(board_noise_2), grid;
subplot(2, 5, 9);
imhist(board_filter_1), grid;
subplot(2, 5, 10);
imhist(board_filter_2), grid;
```

Calcolo degli MSE e visualizzazione in tabella:

```
% calcolo gli MSE delle immagini con rumore e filtrate per verificare se il
% filtro ha avuto effetto positivo
mse_noise_1 = immse(board_noise_1, board);
mse_noise_2 = immse(board_noise_2, board);
mse_filter_1 = immse(board_filter_1, board);
mse_filter_2 = immse(board_filter_2, board);
% visualizzo in tabella gli MSE calcolati
Levels = {'MSE noise low', 'MSE noise hight', 'MSE filter low', 'MSE filter hight'};
Mse = [mse_noise_1; mse_noise_2; mse_filter_1; mse_filter_2];
disp(table(Mse, 'RowNames', Levels));
```

Applicazione dei filtri 5x5 e 7x7 all'immagine con rumore basso:

```
% ripeto i calcoli solo sull'immagine con rumore basso con maschera 5x5
filter_5 = fspecial('average', 5);
board_filter_1_5 = imfilter(board_noise_1, filter_5);
% ripeto i calcoli solo sull'immagine con rumore basso con maschera 7x7
filter_7 = fspecial('average', 7);
board_filter_1_7 = imfilter(board_noise_1, filter_7);
```

Visualizzazione delle immagini filtrate con i rispettivi istogrammi:

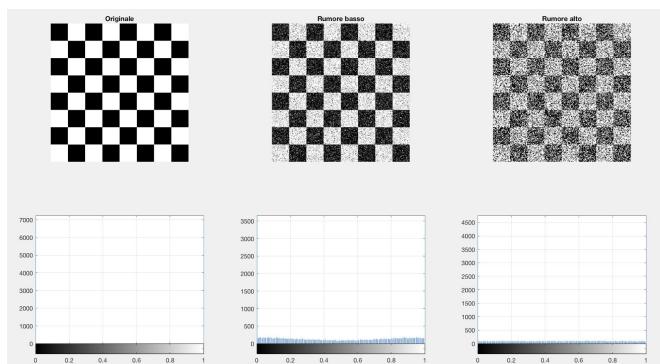
```
% visualizzo le immagini filtrate e i relativi istogrammi
figure;
% -- riga 1
subplot(2, 4, 1);
imshow(board_noise_1, 'InitialMagnification', 'fit'), title('Rumore');
subplot(2, 4, 2);
imshow(board_filter_1, 'InitialMagnification', 'fit'), title('Filtro 3x3');
subplot(2, 4, 3);
imshow(board_filter_1_5, 'InitialMagnification', 'fit'), title('Filtro 5x5');
subplot(2, 4, 4);
imshow(board_filter_1_7, 'InitialMagnification', 'fit'), title('Filtro 7x7');
% -- riga 2
subplot(2, 4, 5);
imhist(board_noise_1), grid;
subplot(2, 4, 6);
imhist(board_filter_1), grid;
subplot(2, 4, 7);
imhist(board_filter_1_5), grid;
subplot(2, 4, 8);
imhist(board_filter_1_7), grid;
```

Calcolo degli MSE delle immagini filtrate e visualizzazione in tabella:

```
% calcolo gli MSE delle immagini filtrate 5x5 e 7x7
mse_filter_1_5 = immse(board_filter_1_5, board);
mse_filter_1_7 = immse(board_filter_1_7, board);
% visualizzo gli MSE calcolati in tabella
Levels = {'MSE noise low', 'MSE filter 3x3', 'MSE filter 5x5', 'MSE filter 7x7'};
Mse = [mse_noise_1; mse_filter_1; mse_filter_1_5; mse_filter_1_7];
disp(table(Mse, 'RowNames', Levels));
end
```

## Risultati

Visualizzazione delle immagini con rumore:



Visualizzazione delle immagini con rumore e quelle con il filtro di matrice 3x3:

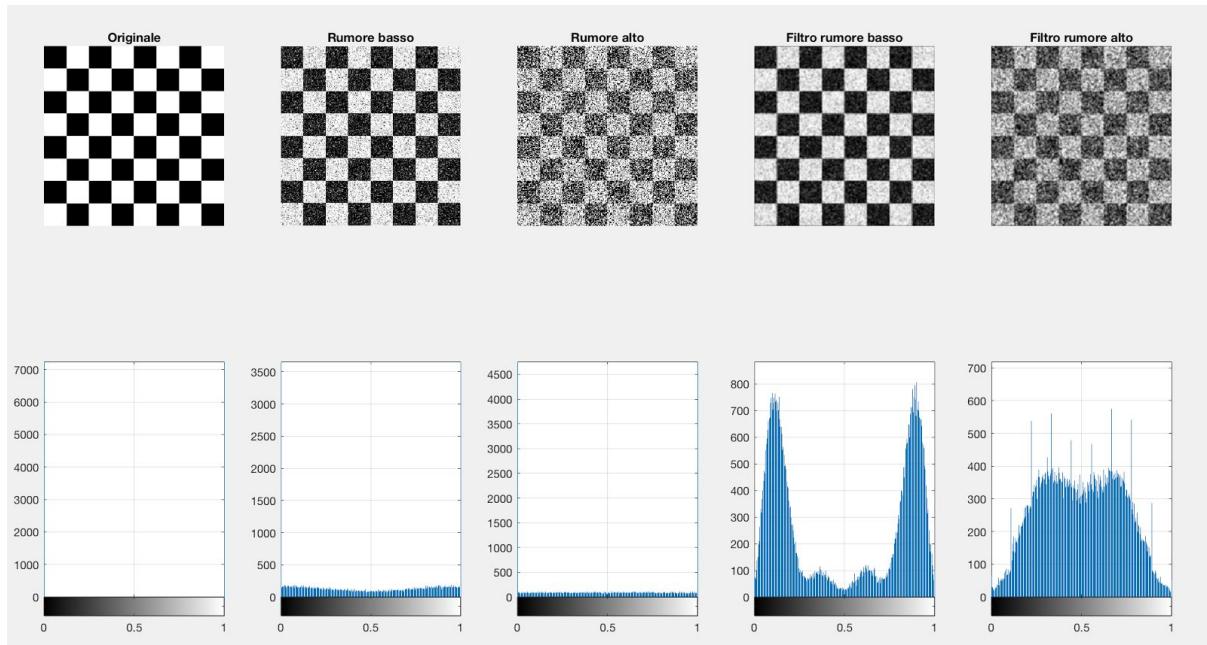


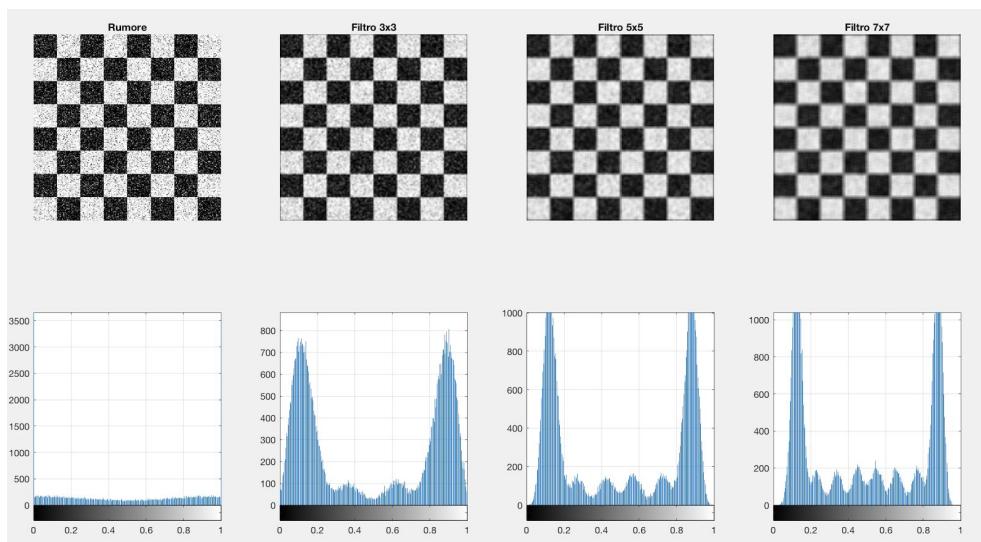
Tabella con gli MSE delle immagini con rumore e delle relative immagini filtrate:

>> esercizio\_2\_8

Mse

MSE noise low	0.049033
MSE noise hight	0.25681
MSE filter low	0.03403
MSE filter hight	0.12787

Visualizzazione delle immagine con rumore basso e di quelle filtrate con filtri di matrice 3x3, 5x5 3 7x7:



Visualizzazione degli MSE delle immagini con i vari filtri (3x3, 5x5, 7x7) in tabella:

### Mse

---

<b>MSE noise low</b>	<b>0.049033</b>
<b>MSE filter 3x3</b>	<b>0.03403</b>
<b>MSE filter 5x5</b>	<b>0.043164</b>
<b>MSE filter 7x7</b>	<b>0.053356</b>

### Osservazioni

Nel punto 3, calcolando gli MSE delle immagini con rumore (basso e alto) e delle stesse alla quale è stato applicato un filtro con matrice 3x3, si nota come in entrambi i casi l'MSE è sceso e quindi ha portato dei benefici all'immagine.

Nel punto 4 e 5 invece si nota come il miglioramento è stato portato solo dalle maschere 3x3 e 5x5, non da quella 7x7. In particolare, quella che ha portato a maggiori benefici è stata la maschera 3x3.

### Facoltativo

#### Codice

```
function esercizio_2_8_fac()
    % genero la scacchiera
    board = double(checkerboard(32)>0.5);
    % genero la board affetta da rumore
    board_noise = imnoise(board, 'gaussian', 0, 1);
    % genero le immagini filtrate a partire da quelle affette da rumore
    filter_3 = fspecial('average', 3);
    board_filter_3 = imfilter(board_noise, filter_3);
    filter_5 = fspecial('average', 5);
    board_filter_5 = imfilter(board_noise, filter_5);
    filter_7 = fspecial('average', 7);
    board_filter_7 = imfilter(board_noise, filter_7);
    % calcolo l'MSE delle immagini
    mse_noise = immse(board_noise, board);
    mse_filter_3 = immse(board_filter_3, board);
    mse_filter_5 = immse(board_filter_5, board);
    mse_filter_7 = immse(board_filter_7, board);
    % visualizzo i risultati in tabella
    Levels = {'Noise', 'Filter 3x3', 'Filter 5x5', 'Filter 7x7'};
    Mse = [mse_noise; mse_filter_3; mse_filter_5; mse_filter_7];
    disp(table(Mse, 'RowNames', Levels));
end
```

## Risultati

	Mse
Noise	0.25911
Filter 3x3	0.12921
Filter 5x5	0.1263
Filter 7x7	0.13116

## Osservazioni

In questo caso si nota come tutti e tre i filtri abbiano portato miglioramenti all'immagine ma i valori tra loro non hanno una grande variazione.

## Esercizio 2.9

E' data l'immagine 'text.png'.

- Usando la procedura imtool() individuare una regione rettangolare dell'immagine che corrisponda al carattere 'e'. Ritagliare la sottoimmagine corrispondente usando la procedura imcrop(). Visualizzare l'immagine sorgente e la sottoimmagine ritagliata.
- Calcolare la correlazione incrociata normalizzata con la procedura normxcorr2() ; visualizzare la correlazione rappresentandola in livelli di grigio (assieme al proprio istogramma) e in grafica 3d tramite la procedura surf()
- Binarizzare l'immagine della correlazione tramite sogliatura (thresholding), usando la im2bw() con valore di soglia threshold=0.8.
- Presentare i risultati nelle osservazioni finali.

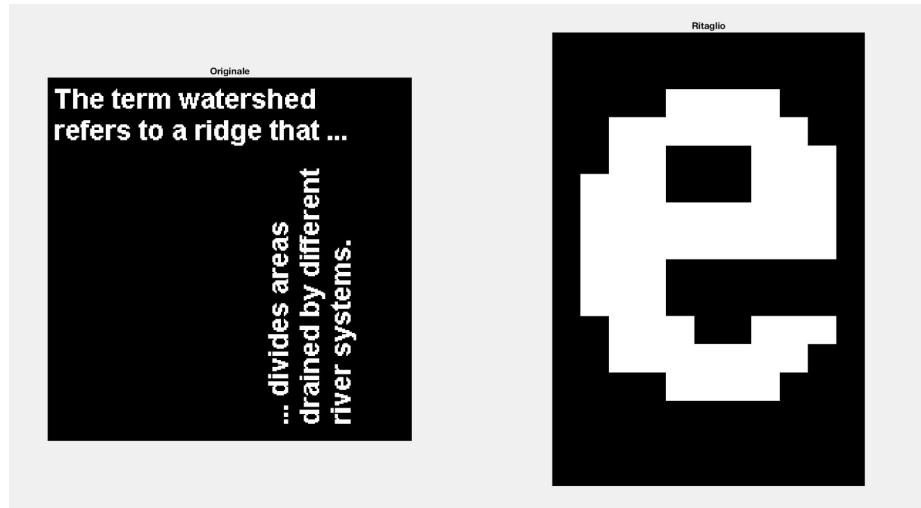
Facoltativo: Ripetere il calcolo della correlazione (passo 2) avendo ruotato di 90° l'immagine ritagliata del carattere 'e'. Ripetere i passi 3, 4 con i nuovi risultati.

### Codice

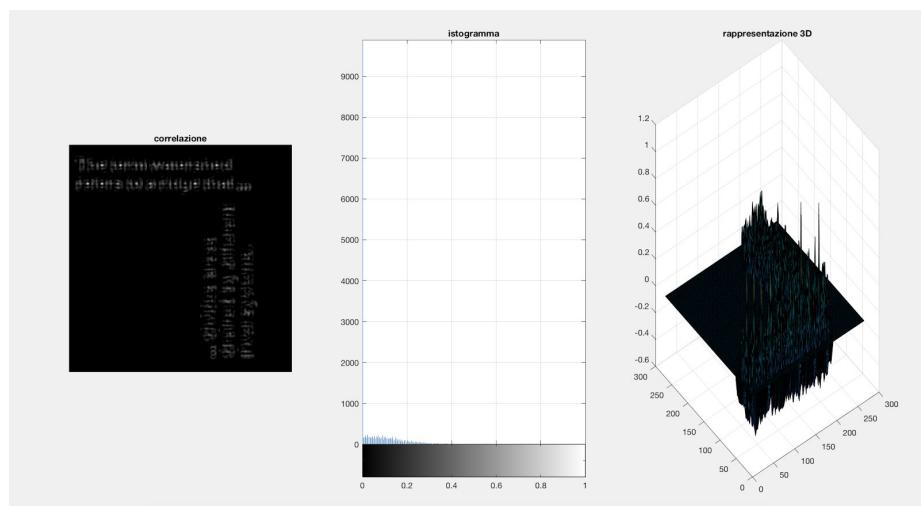
```
function esercizio_2_9()
% carico l'immagine
img = imread('text.png');
% ottengo l'immagine ritagliata
crop = imcrop(img, [125 10 10 15]);
% visualizzo l'immagine sorgente e la sottoimmagine ritagliata
figure;
subplot(1, 2, 1);
imshow(img, 'InitialMagnification', 'fit'), title('Originale');
subplot(1, 2, 2);
imshow(crop, 'InitialMagnification', 'fit'), title('Ritaglio');
% calcolo la correlazione incrociata normalizzata
norm = normxcorr2(crop, img);
% visualizzo la correlazione calcolata
figure;
subplot(1, 3, 1);
imshow(norm, 'InitialMagnification', 'fit'), title('correlazione');
subplot(1, 3, 2);
imhist(norm), grid, title('istogramma');
subplot(1, 3, 3);
surf(norm), title('rappresentazione 3D');
% binarizzo l'immagine della correlazione tramite sogliatura con valore di
% soglia 0.8
binarized = im2bw(norm, 0.8);
figure, imshow(binarized, 'InitialMagnification', 'fit');
end
```

## Risultati

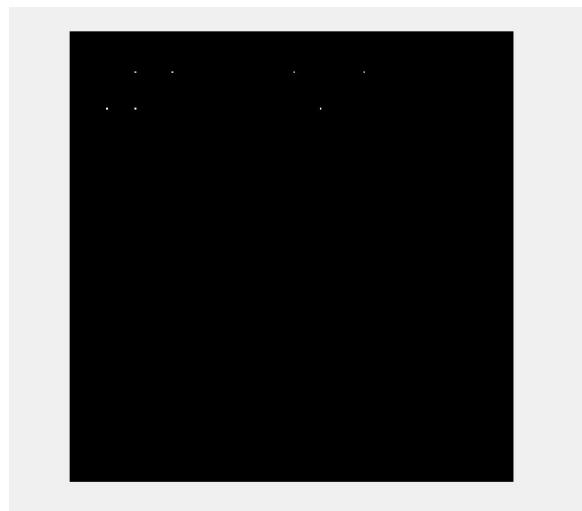
Visualizzazione immagine originale e ritaglio:



Visualizzazione della correlazione:



Visualizzazione dell'immagine binarizzata tramite sogliatura:



## Osservazioni

La correlazione incrociata è un calcolo utilizzato per stimare il grado di somiglianza tra una immagine e una sottoimmagine sovrapponendo le parti pixel per pixel.

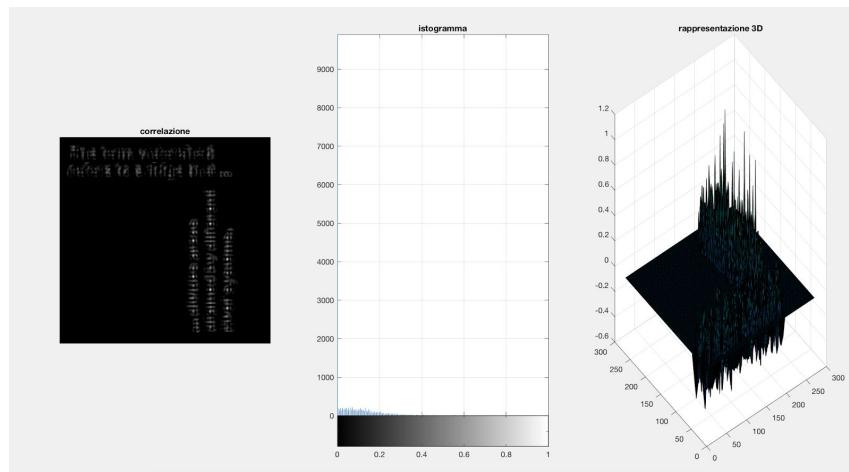
In particolare nell'immagine binarizzata tramite sogliatura è evidente che sono stati riconosciuti tutti i punti in cui la lettera “e” è ripetuta nel testo orizzontale.

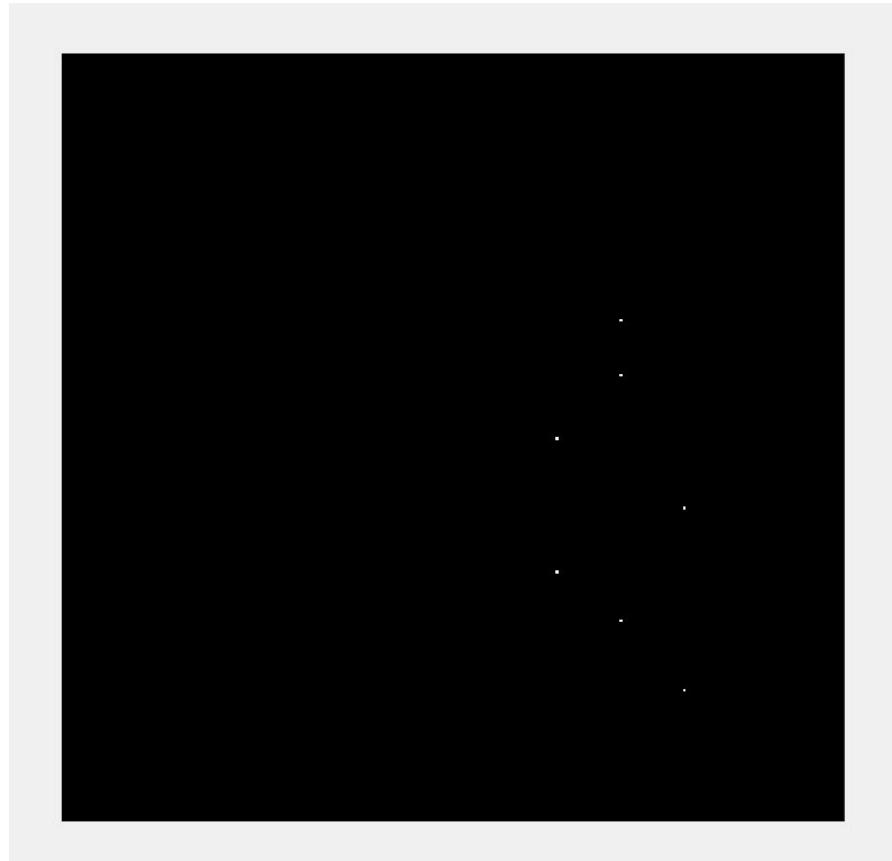
## Facoltativo

### Codice

```
function esercizio_2_9_fac()
% carico l'immagine
img = imread('text.png');
% ottengo l'immagine ritagliata
crop = imcrop(img, [125 10 10 15]);
crop_90 = imrotate(crop, 90);
% calcolo la correlazione incrociata normalizzata
norm = normxcorr2(crop_90, img);
% visualizzo la correlazione calcolata
figure;
subplot(1, 3, 1);
imshow(norm, 'InitialMagnification', 'fit'), title('correlazione');
subplot(1, 3, 2);
imhist(norm), grid, title('istogramma');
subplot(1, 3, 3);
surf(norm), title('rappresentazione 3D');
% binarizzo l'immagine della correlazione tramite sogliatura con valore di
% soglia 0.8
binarized = im2bw(norm, 0.8);
figure, imshow(binarized, 'InitialMagnification', 'fit');
end
```

## Risultati





### Osservazioni

Come nel caso precedente è evidente che sono state evidenziati tutti i punti in cui compare la lettera “e”, solo che in questo caso ci si riferisce al testo verticale.

## Esercizio 2.10

Data l'immagine 'eight.tif', simulare la presenza di rumore casuale additivo di tipo gaussiano a media nulla, varianza 0.01 e 0.1 (basso e alto rumore rispettivamente).

Analisi del rumore

- Visualizzare l'immagine senza rumore (ideale); l'immagine a basso e alto rumore di entrambi i tipi. Visualizzare il profilo dei livelli di grigio lungo una riga significativa (scanline) nelle tre immagini. Usare la procedura imtool() in modalità interattiva per individuare le coordinate-pixel iniziale e finale di un segmento dell'immagine, e la procedura improfile() da codice.
- Considerare le immagini con rumore basso-alto. Rispetto all'immagine originale presa come riferimento, stimare: il MSE usando la immse(); il rapporto segnale-rumore di picco (psnr) usando la procedura psnr(). Riportare i risultati in una tabella.

Filtraggio gaussiano del rumore gaussiano

- Applicare a ciascuna delle due immagini con rumore gaussiano tre filtri anch'essi gaussiani con finestra 3x3 e tre valori di deviazione standard: 0.1, 0.5, 1.
- Stimare il MSE e il psnr delle immagini dopo i filtraggi.
- Recuperare le stime di MSE e psnr calcolate al punto 2. Confrontare le stime prima e dopo il filtraggio, per ciascun valore di deviazione standard del filtro, e riportare i dati in una nuova tabella. Stabilire quale dei tre filtri ha dato il risultato migliore e riportarlo nelle osservazioni finali.

Facoltativo Ripetere i calcoli ai passi 4, 5 per le immagini con rumore 'sale e pepe' basso (param. 0.1) e alto (param. 0.3).

**Codice**

Applicazione dei rumori all'immagine e visualizzazione della stessa con imtool():

```
function esercizio_2_10()
% carico l'immagine
img = imread('eight.tif');
% simulo i rumori nell'immagine
img_noise_low = imnoise(img, 'gaussian', 0, 0.01);
img_noise_high = imnoise(img, 'gaussian', 0, 0.1);
% utilizzo imtool in modalità interattiva per individuare le coordinate
imtool(img);
x = [1, 308];
y = [80, 80];
```

Visualizzazione delle immagini e dei profili di livello di grigio lungo una scanline:

```
% visualizzo l'immagine originale e quelle con rumore applicato
figure;
% -- riga 1
subplot(2, 3, 1);
imshow(img, 'InitialMagnification', 'fit'), title('Originale');
subplot(2, 3, 2);
imshow(img_noise_low, 'InitialMagnification', 'fit'), title('Rumore basso');
subplot(2, 3, 3);
imshow(img_noise_high, 'InitialMagnification', 'fit'), title('Rumore alto');
% -- riga 2
subplot(2, 3, 4);
improfile(img, x, y), grid;
subplot(2, 3, 5);
improfile(img_noise_low, x, y), grid;
subplot(2, 3, 6);
improfile(img_noise_high, x, y), grid;
```

Calcolo MSE e PSNR delle immagini e visualizzazione in tabella:

```
% stimo l'MSE e il PSNR delle immagini con rumore
mse_low = immse(img_noise_low, img);
mse_high = immse(img_noise_high, img);
psnr_low = psnr(img_noise_low, img);
psnr_high = psnr(img_noise_high, img);
% visualizzo i dati in una tabella
Names = {'MSE noise low', 'MSE noise high', 'PSNR noise low', 'PSNR noise high'};
Values = [mse_low; mse_high; psnr_low; psnr_high];
disp(table(Values, 'RowNames', Names));
```

Applicazione dei filtri e calcolo del MSE e PSNR:

```
% applico alle immagini con rumore tre filtri con matrice 3x3 e valori di
% deviazione standard diversi
matrix_1 = fspecial('gaussian', 3, 0.1);
matrix_2 = fspecial('gaussian', 3, 0.5);
matrix_3 = fspecial('gaussian', 3, 1);
low_m1 = imfilter(img_noise_low, matrix_1);
low_m2 = imfilter(img_noise_low, matrix_2);
low_m3 = imfilter(img_noise_low, matrix_3);
high_m1 = imfilter(img_noise_high, matrix_1);
high_m2 = imfilter(img_noise_high, matrix_2);
high_m3 = imfilter(img_noise_high, matrix_3);
% calcolo MSE e PSNR per le immagini con l'applicazione del filtro
mse_low_m1 = immse(low_m1, img);
mse_low_m2 = immse(low_m2, img);
mse_low_m3 = immse(low_m3, img);
mse_high_m1 = immse(high_m1, img);
mse_high_m2 = immse(high_m2, img);
mse_high_m3 = immse(high_m3, img);
psnr_low_m1 = psnr(low_m1, img);
psnr_low_m2 = psnr(low_m2, img);
psnr_low_m3 = psnr(low_m3, img);
psnr_high_m1 = psnr(high_m1, img);
psnr_high_m2 = psnr(high_m2, img);
psnr_high_m3 = psnr(high_m3, img);
```

Visualizzazione dei dati calcolati insieme ai precedenti in tabella:

```
% visualizzo i dati finali in una tabella insieme a quelli delle immagini
% con rumore senza filtro
Names = {'MSE', 'PSNR'};
LowImage = [mse_low; psnr_low];
HighImage = [mse_high; psnr_high];
LowImageM1 = [mse_low_m1; psnr_low_m1];
LowImageM2 = [mse_low_m2; psnr_low_m2];
LowImageM3 = [mse_low_m3; psnr_low_m3];
HighImageM1 = [mse_high_m1; psnr_high_m1];
HighImageM2 = [mse_high_m2; psnr_high_m2];
HighImageM3 = [mse_high_m3; psnr_high_m3];
disp(table(LowImage, HighImage, LowImageM1, LowImageM3, HighImageM1, HighImageM2, HighImageM3, 'RowNames', Names));
end
```

## Risultati

Visualizzazione delle immagini con rumore e dei relativi profili di livello di grigio:

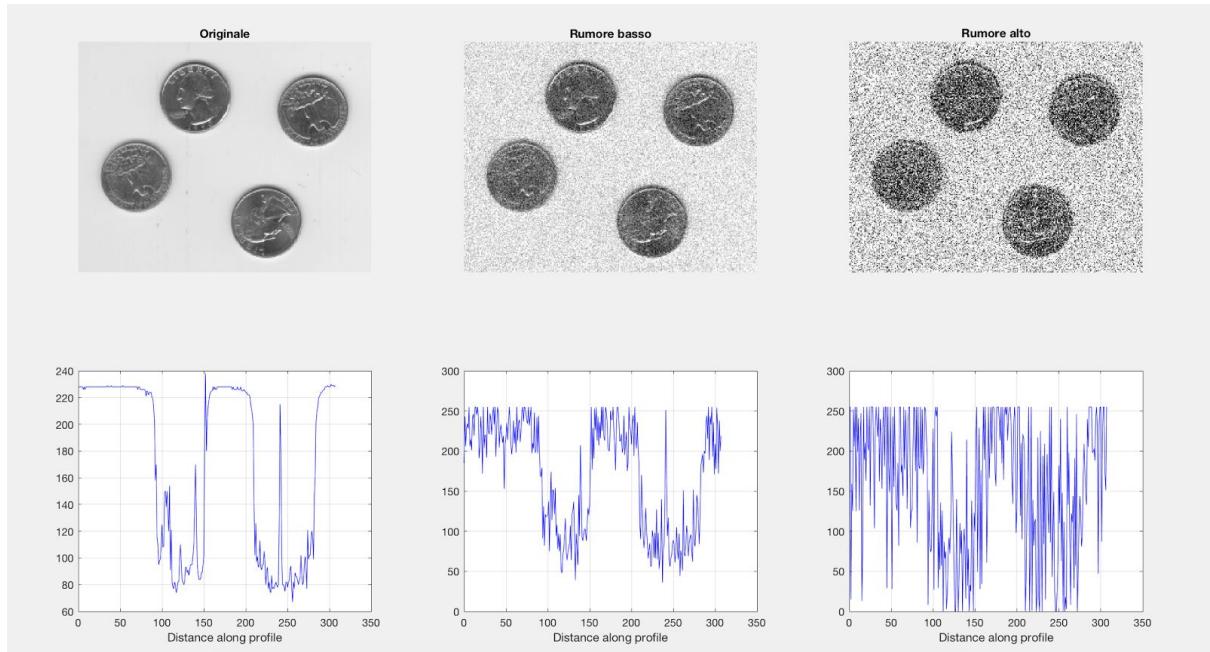


Tabella con gli MSE e PSNR calcolati sulle immagini con rumore:

### Values

<b>MSE noise low</b>	539.26
<b>MSE noise high</b>	3865.3
<b>PSNR noise low</b>	20.813
<b>PSNR noise high</b>	12.259

Tabella con gli MSE e PSNR calcolati sulle immagini con rumore e quelle filtrate:

	LowImage	HighImage	LowImageM1	LowImageM2	LowImageM3	HighImageM1	HighImageM2	HighImageM3
<b>MSE</b>	539.26	3865.3	539.26	243.34	174.84	3865.3	1813.5	889.64
<b>PSNR</b>	20.813	12.259	20.813	24.269	25.704	12.259	15.546	18.639

## Osservazioni

Come si nota dall'ultima tabella riportata nei risultati il filtro che ha portato ai risultati migliori è quello gaussiano con matrice 3x3 e deviazione standard di valore 1.

## Facoltativo

### Codice

```
function esercizio_2_10_fac()
% carico l'immagine
img = imread('eight.tif');
% simulo i rumori nell'immagine
img_noise_low = imnoise(img, 'salt & pepper', 0.1);
img_noise_high = imnoise(img, 'salt & pepper', 0.3);
% applico alle immagini con rumore tre filtri con matrice 3x3 e valori di
% deviazione standard diversi
matrix_1 = fspecial('gaussian', 3, 0.1);
matrix_2 = fspecial('gaussian', 3, 0.5);
matrix_3 = fspecial('gaussian', 3, 1);
low_m1 = imfilter(img_noise_low, matrix_1);
low_m2 = imfilter(img_noise_low, matrix_2);
low_m3 = imfilter(img_noise_low, matrix_3);
high_m1 = imfilter(img_noise_high, matrix_1);
high_m2 = imfilter(img_noise_high, matrix_2);
high_m3 = imfilter(img_noise_high, matrix_3);

% calcolo MSE e PSNR delle immagini
mse_low = immse(img_noise_low, img);
mse_low_m1 = immse(low_m1, img);
mse_low_m2 = immse(low_m2, img);
mse_low_m3 = immse(low_m3, img);
mse_high = immse(img_noise_high, img);
mse_high_m1 = immse(high_m1, img);
mse_high_m2 = immse(high_m2, img);
mse_high_m3 = immse(high_m3, img);
psnr_low = psnr(img_noise_low, img);
psnr_low_m1 = psnr(low_m1, img);
psnr_low_m2 = psnr(low_m2, img);
psnr_low_m3 = psnr(low_m3, img);
psnr_high = psnr(img_noise_high, img);
psnr_high_m1 = psnr(high_m1, img);
psnr_high_m2 = psnr(high_m2, img);
psnr_high_m3 = psnr(high_m3, img);

% visualizzo i dati finali in una tabella
Names = {'MSE', 'PSNR'};
LowImage = [mse_low; psnr_low];
HighImage = [mse_high; psnr_high];
LowImageM1 = [mse_low_m1; psnr_low_m1];
LowImageM2 = [mse_low_m2; psnr_low_m2];
LowImageM3 = [mse_low_m3; psnr_low_m3];
HighImageM1 = [mse_high_m1; psnr_high_m1];
HighImageM2 = [mse_high_m2; psnr_high_m2];
HighImageM3 = [mse_high_m3; psnr_high_m3];
disp(table(LowImage, HighImage, LowImageM1, LowImageM2, LowImageM3, HighImageM1, HighImageM2, HighImageM3, 'RowNames', Names));
end
```

## Risultati

>> esercizio_2_10_fac								
	LowImage	HighImage	LowImageM1	LowImageM2	LowImageM3	HighImageM1	HighImageM2	HighImageM3
MSE	2439.5	7233.1	2439.5	1080.3	495.37	7233.1	3427.1	1654.2
PSNR	14.258	9.5376	14.258	17.796	21.182	9.5376	12.782	15.945

## Esercizio 2.11

E' data l'immagine 'circuit.tif'.

- Aggiungere rumore gaussiano a media nulla e varianza 0.01 (basso) e 0.1 (alto rumore); aggiungere rumore 'sale e pepe' con parametro 0.1 (basso) e 0.4 (alto). Visualizzare l'immagine originale e quelle con rumore;
- Effettuare il filtraggio delle quattro immagini rumorose tramite due filtri: un filtro gaussiano 3x3 con deviazione standard 0.5; un filtro mediano (funzione medfilt2()), anche questo con maschera 3x3;
- Confrontare l'effetto dei due filtri sulle quattro immagini rumorose e riportare i risultati in tabella: quale filtro dà le prestazioni migliori a basso/alto rumore? Riportare le osservazioni nelle conclusioni.

### Codice

Aggiunta del rumore all'immagine iniziale e visualizzazione delle immagini con rumore:

```
function esercizio_2_11()
    % carico l'immagine
    img = imread('circuit.tif');
    % aggiungo i rumori all'immagine
    gaus_low = imnoise(img, 'gaussian', 0, 0.01);
    gaus_high = imnoise(img, 'gaussian', 0, 0.1);
    sp_low = imnoise(img, 'salt & pepper', 0.1);
    sp_high = imnoise(img, 'salt & pepper', 0.4);
    % visualizzo le immagini
    figure;
    % -- riga 1
    subplot(1, 5, 1);
    imshow(img, 'InitialMagnification', 'fit'), title('Originale');
    subplot(1, 5, 2);
    imshow(gaus_low, 'InitialMagnification', 'fit'), title('Gaussiano basso');
    subplot(1, 5, 3);
    imshow(gaus_high, 'InitialMagnification', 'fit'), title('Gaussiano alto');
    subplot(1, 5, 4);
    imshow(sp_low, 'InitialMagnification', 'fit'), title('Sale e pepe basso');
    subplot(1, 5, 5);
    imshow(sp_high, 'InitialMagnification', 'fit'), title('Sale e pepe alto');
```

Applicazione dei filtri alle immagini e calcolo degli MSE:

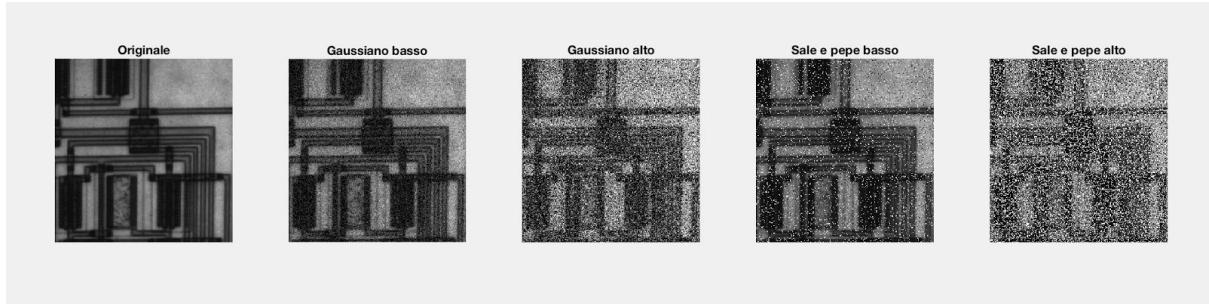
```
% effettuo il filtraggio delle immagini
matrix_gauss = fspecial('gaussian', 3, 0.5);
fg_gaus_low = imfilter(gaus_low, matrix_gauss);
fg_gaus_high = imfilter(gaus_high, matrix_gauss);
fg_sp_low = imfilter(sp_low, matrix_gauss);
fg_sp_high = imfilter(sp_high, matrix_gauss);
fm_gaus_low = medfilt2(gaus_low, [3,3]);
fm_gaus_high = medfilt2(gaus_high, [3,3]);
fm_sp_low = medfilt2(sp_low, [3,3]);
fm_sp_high = medfilt2(sp_high, [3,3]);
% calcolo MSE delle immagini
mse_gaus_low = immse(gaus_low, img);
mse_gaus_high = immse(gaus_high, img);
mse_sp_low = immse(sp_low, img);
mse_sp_high = immse(sp_high, img);
mse_fg_gaus_low = immse(fg_gaus_low, img);
mse_fg_gaus_high = immse(fg_gaus_high, img);
mse_fg_sp_low = immse(fg_sp_low, img);
mse_fg_sp_high = immse(fg_sp_high, img);
mse_fm_gaus_low = immse(fm_gaus_low, img);
mse_fm_gaus_high = immse(fm_gaus_high, img);
mse_fm_sp_low = immse(fm_sp_low, img);
mse_fm_sp_high = immse(fm_sp_high, img);
```

Visualizzazione dei risultati in tabella:

```
% visualizzo i risultati in tabella
Names = {'Gaussiano basso', 'Gaussiano alto', 'Sale e pepe basso', 'Sale e pepe alto'};
MseNoise = [mse_gaus_low; mse_gaus_high; mse_sp_low; mse_sp_high];
MseFilterGauss = [mse_fg_gaus_low; mse_fg_gaus_high; mse_fg_sp_low; mse_fg_sp_high];
MseFilterMed = [mse_fm_gaus_low; mse_fm_gaus_high; mse_fm_sp_low; mse_fm_sp_high];
disp(table(MseNoise, MseFilterGauss, MseFilterMed, 'RowNames', Names));
end
```

## Risultati

Visualizzazione delle immagini con rumore:



Visualizzazione degli MSE calcolati in tabella:

```
>> esercizio_2_11
```

	MseNoise	MseFilterGauss	MseFilterMed
<b>Gaussiano basso</b>	596.29	249.16	128.24
<b>Gaussiano alto</b>	4373.6	1906.1	977
<b>Sale e pepe basso</b>	2119.6	908.71	20.947
<b>Sale e pepe alto</b>	8419.2	3927.1	914.93

## Osservazioni

Sia nel caso dei rumori alti, sia in quello dei rumori bassi, i risultati migliori sono stati portati dal filtro mediano.

## Esercizio 2.12

Generazione di profili di intensità per simulare modelli di edge e filtri in una dimensione.

- (a) Generare un profilo di intensità (intensity profile) che simuli edge di tipo rampa (ramp-edge) e tetto (roof-edge). Generare un secondo profilo di intensità che simuli un edge di tipo gradino (step-edge); visualizzare i due segnali ottenuti;
- (b) Aggiungere rumore gaussiano a media nulla basso/alto ai due profili di intensità ottenuti al punto (a). Per il livello basso la deviazione standard di rumore è l'1% della massima intensità; per il livello alto la deviazione standard è il 20% della massima intensità. Visualizzare i risultati;
- (c) Generare due profili di filtri unidimensionali, rispettivamente alle somme – maschera [1 1] - e alle differenze, maschera [1 -1]. Visualizzare i due segnali ottenuti.

Per l'aggiunta del rumore (punto (b)) usare la procedura MATLAB randn(.)

### Codice

Funzione per la generazione e visualizzazione di un profilo di intensità che simuli edge di tipo rampa e tetto:

```
function[profile] = edge_ramp_roof(seq, show)
    samples = 100;
    x = 0:1:(length(seq) - 1);
    delta_x = ((max(x) - min(x)) / (length(seq) * samples - 1));
    interpolator_x = 0 : delta_x : (length(seq) - 1);
    profile = interp1(x, seq, interpolator_x);

    if strcmp(show, 'show')
        figure, plot(interpolator_x, profile);
    end
end
```

Funzione per la generazione e visualizzazione di un profilo di intensità che simuli edge di tipo gradino:

```
function[profile] = edge_step(seq, show)
    profile = [];
    dim = length(seq);
    for i = 1 : dim
        profile = [profile seq(i) * ones(1,100)];
    end

    if strcmp(show, 'show')
        figure, plot(profile);
    end
end
```

Definizione dei due profili e visualizzazione (attraverso le due funzioni precedenti):

```
function esercizio_2_12()
% procedura per generare un profilo di intensità che simuli edge di tipo
% rampa e tetto e visualizzarlo
seq_rr = [0 0 0 4 8 12 12 0 0 0 12 0 0];
profile_rr = edge_ramp_roof(seq_rr, 'show');
% procedura per generare un profilo di intensità che simuli edge di tipo
% gradino e visualizzarlo
step_s = [0 0 4 4 0 0 0 4 4 0 0];
profile_s = edge_step(step_s, 'show');
```

Aggiunta dei rumori ai profili ottenuti:

```
% aggiungo un rumore di tipo gaussiano ai due profili ottenuti
stdev_low_rr = 0.01 * max(profile_rr);
stdev_high_rr = 0.2 * max(profile_rr);
stdev_low_s = 0.01 * max(profile_s);
stdev_high_s = 0.2 * max(profile_s);
profile_rr_low = profile_rr + stdev_low_rr * randn(1, length(profile_rr));
profile_rr_high = profile_rr + stdev_high_rr * randn(1, length(profile_rr));
profile_s_low = profile_s + stdev_low_s * randn(1, length(profile_s));
profile_s_high = profile_s + stdev_high_s * randn(1, length(profile_s));
```

Visualizzazione dei profili con rumore:

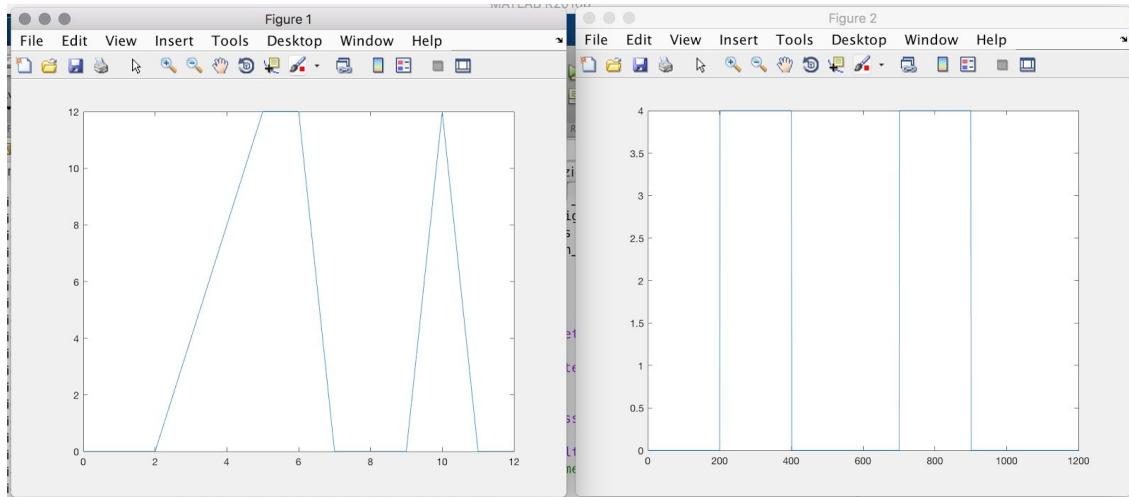
```
% visualizzo i risultati
figure;
% -- riga 1
subplot(2, 2, 1);
plot(profile_rr_low), title('Rampa e tetto basso');
subplot(2, 2, 2);
plot(profile_rr_high), title('Rampa e tetto alto');
% -- riga 2
subplot(2, 2, 3);
plot(profile_s_low), title('Gradino basso');
subplot(2, 2, 4);
plot(profile_s_high), title('Gradino alto');
```

Creazione dei profili di filtri e visualizzazione:

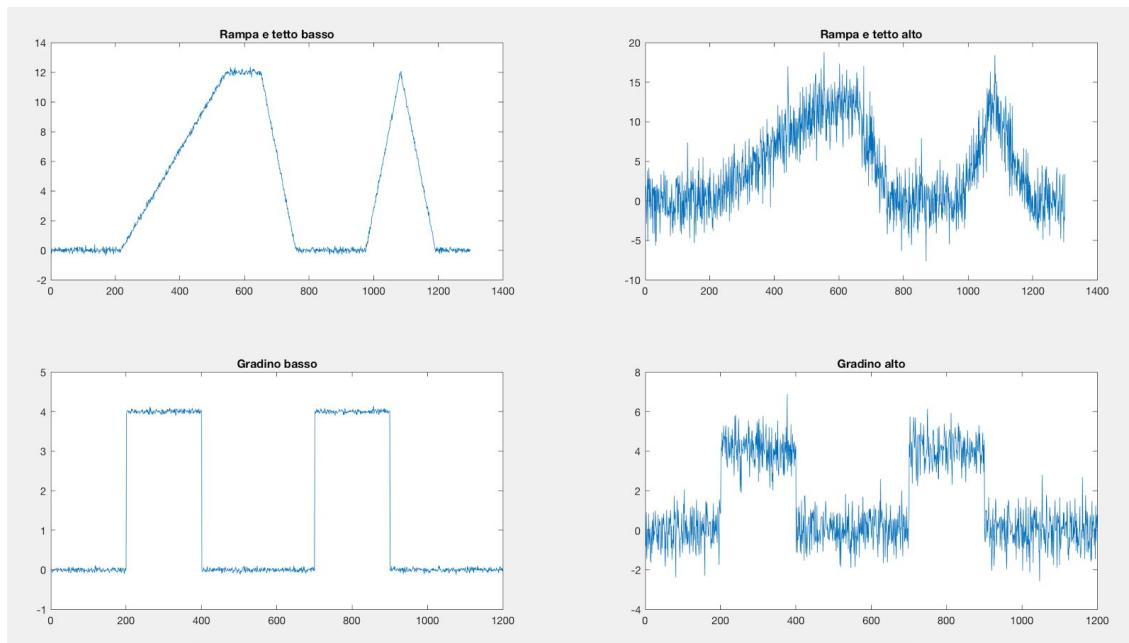
```
% genero i due profili di filtri unidimensionali alle somme e alle
% differenze
sum_mask = (1/4) * ones(1, 5);
diff_mask = [-ones(1, 5) ones(1, 5)];
% visualizzo i due profili
figure;
% -- riga 1
subplot(2, 2, 1);
plot(sum_mask), title('Somma');
subplot(2, 2, 2);
plot(diff_mask), title('Differenza');
```

## Risultati

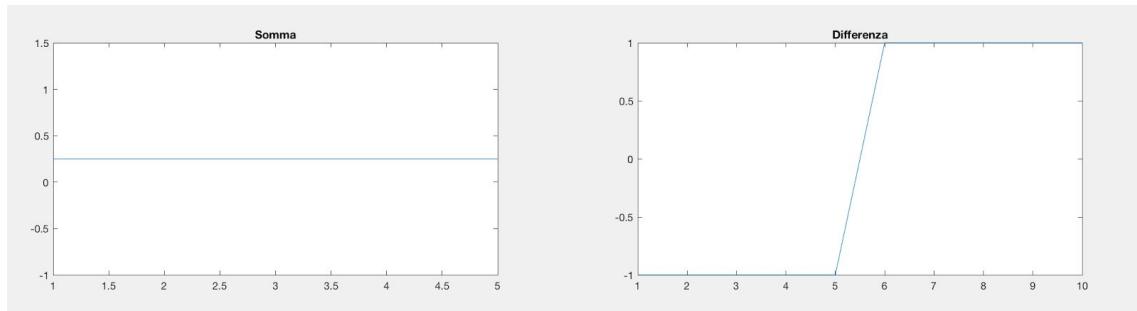
Visualizzazione dei due profili:



Visualizzazione dei profili con rumore:



Visualizzazione dei profili di filtri:



## Esercizio 2.13

Analisi della risposta a profili di intensità in una dimensione da parte di filtri alle somme e alle differenze.

(a) Filtrare i due segnali generati all'esercizio 2.12(a) senza rumore con i filtri alle somme e alle differenze, maschere riportate al precedente esercizio 2.12(c). Osservare i risultati e riportare le osservazioni nelle conclusioni;

(b) Filtrare i segnali con rumore basso/alto con filtri alle somme e alle differenze (punto precedente). Riportare le osservazioni nelle conclusioni;

(c) Effettuare un doppio filtraggio, applicando ai segnali senza rumore e con rumore filtrati alle differenze, un ulteriore filtro alle differenze. Osservare e discutere i risultati;

Utilizzare per i filtraggi la procedura MATLAB conv(.) che calcola la convoluzione in una dimensione.

### Codice

#### Definizione dei profili e applicazione dei filtri

```
% function esercizio_2_13()
% procedura per generare un profilo di intensità che simuli edge di tipo
% rampa e tetto e visualizzarlo
seq_rr = [0 0 0 4 8 12 12 0 0 0 12 0 0];
profile_rr = edge_ramp_roof(seq_rr, 'not_show');
% procedura per generare un profilo di intensità che simuli edge di tipo
% gradino e visualizzarlo
step_s = [0 0 4 4 0 0 0 4 4 0 0 0 0];
profile_s = edge_step(step_s, 'not_show');
% genero i due profili di filtri unidimensionali alle somme e alle
% differenze
sum_mask = (1/4) * ones(1, 5);
diff_mask = [-ones(1, 5) ones(1, 5)];
% filtro i profili con i filtri costruiti
profile_rr_sum = conv(profile_rr, sum_mask);
profile_rr_diff = conv(profile_rr, diff_mask);
profile_s_sum = conv(profile_s, sum_mask);
profile_s_diff = conv(profile_s, diff_mask);
```

Visualizzazione dei due profili con l'applicazione dei filtri:

```
% visualizzo i risultati
figure;
% -- riga 1
subplot(2, 2, 1);
plot(profile_rr_sum), title('Rampa e tetto somme');
subplot(2, 2, 2);
plot(profile_rr_diff), title('Rampa e tetto differenze');
% -- riga 2
subplot(2, 2, 3);
plot(profile_s_sum), title('Gradino somme');
subplot(2, 2, 4);
plot(profile_s_diff), title('Gradino differenze');
```

Generazione dei profili affetti da rumore e applicazione dei filtri:

```
% genero i profili affetti da rumore
stdev_low_rr = 0.01 * max(profile_rr);
stdev_high_rr = 0.2 * max(profile_rr);
stdev_low_s = 0.01 * max(profile_s);
stdev_high_s = 0.2 * max(profile_s);
profile_rr_low = profile_rr + stdev_low_rr * randn(1, length(profile_rr));
profile_rr_high = profile_rr + stdev_high_rr * randn(1, length(profile_rr));
profile_s_low = profile_s + stdev_low_s * randn(1, length(profile_s));
profile_s_high = profile_s + stdev_high_s * randn(1, length(profile_s));
% filtro i profili affetti da rumore con i filtri costruiti
profile_rr_low_sum = conv(profile_rr_low, sum_mask);
profile_rr_high_sum = conv(profile_rr_high, sum_mask);
profile_s_low_sum = conv(profile_s_low, sum_mask);
profile_s_high_sum = conv(profile_s_high, sum_mask);
profile_rr_low_diff = conv(profile_rr_low, diff_mask);
profile_rr_high_diff = conv(profile_rr_high, diff_mask);
profile_s_low_diff = conv(profile_s_low, diff_mask);
profile_s_high_diff = conv(profile_s_high, diff_mask);
```

Visualizzazione dell'applicazione dei filtri:

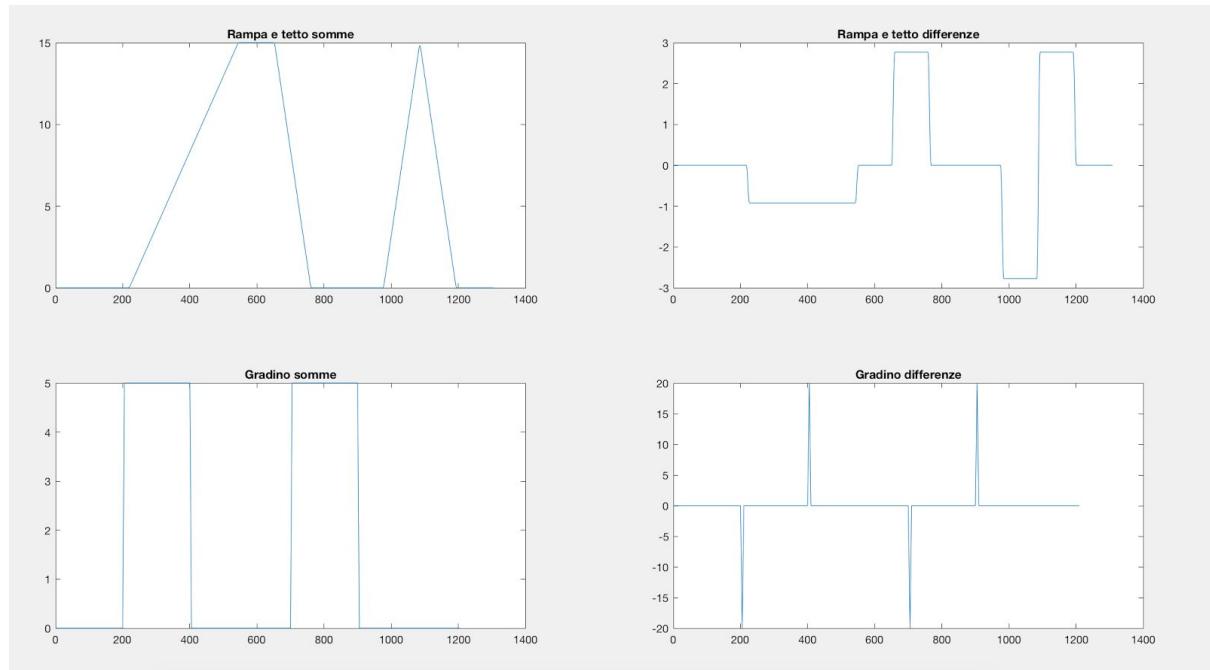
```
% visualizzo i vari risultati
figure;
% -- riga 1
subplot(6, 2, 1);
plot(profile_rr_sum), title('Rampa e tetto somme');
subplot(6, 2, 2);
plot(profile_s_sum), title('Gradino somme');
% -- riga 2
subplot(6, 2, 3);
plot(profile_rr_diff), title('Rampa e tetto differenze');
subplot(6, 2, 4);
plot(profile_s_diff), title('Gradino differenze');
% -- riga 3
subplot(6, 2, 5);
plot(profile_rr_low_sum), title('Rampa e tetto basso somme');
subplot(6, 2, 6);
plot(profile_s_low_sum), title('Gradino basso somme');
% -- riga 4
subplot(6, 2, 7);
plot(profile_rr_low_diff), title('Rampa e tetto basso differenze');
subplot(6, 2, 8);
plot(profile_s_low_diff), title('Gradino basso differenze');
% -- riga 5
subplot(6, 2, 9);
plot(profile_rr_high_sum), title('Rampa e tetto alto somme');
subplot(6, 2, 10);
plot(profile_s_high_sum), title('Gradino alto somme');
% -- riga 6
subplot(6, 2, 11);
plot(profile_rr_high_diff), title('Rampa e tetto alto differenze');
subplot(6, 2, 12);
plot(profile_s_high_diff), title('Gradino alto differenze');
```

Applicazione del doppio filtraggio ai profili già filtrati alle differenze e visualizzazione:

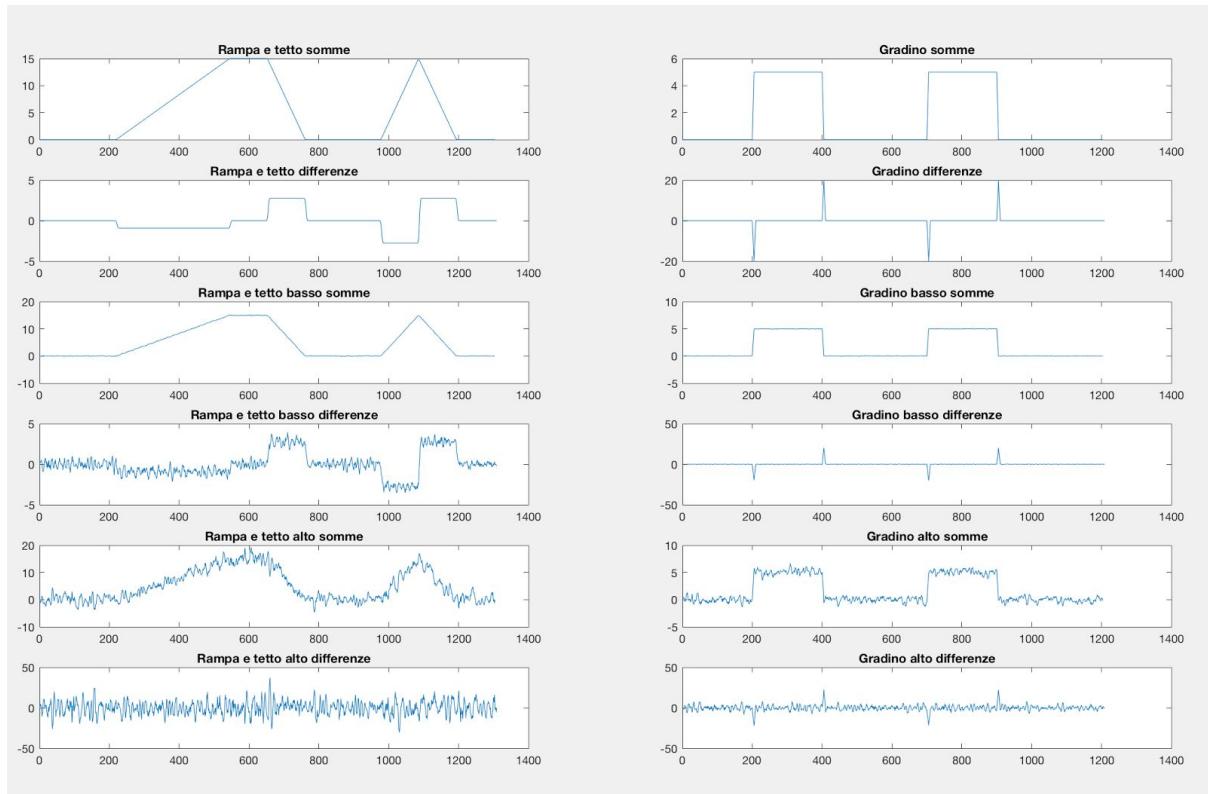
```
% applico un doppio filtraggio alle differenze
profile_rr_diff2 = conv(profile_rr_diff, diff_mask);
profile_rr_low_diff2 = conv(profile_rr_low_diff, diff_mask);
profile_rr_high_diff2 = conv(profile_rr_high_diff, diff_mask);
profile_s_diff2 = conv(profile_s_diff, diff_mask);
profile_s_low_diff2 = conv(profile_s_low_diff, diff_mask);
profile_s_high_diff2 = conv(profile_s_high_diff, diff_mask);
% visualizzo i vari risultati del doppio filtraggio
figure;
% -- riga 1
subplot(4, 2, 1);
plot(profile_rr_diff), title('Rampa e tetto somme');
subplot(4, 2, 2);
plot(profile_s_diff), title('Gradino differenza');
% -- riga 2
subplot(4, 2, 3);
plot(profile_rr_low_diff2), title('Rampa e tetto differenza doppia');
plot(profile_s_low_diff2), title('Gradino basso differenza doppia');
% -- riga 3
subplot(4, 2, 5);
plot(profile_rr_low_diff2), title('Rampa e tetto basso differenza doppia');
subplot(4, 2, 6);
plot(profile_s_low_diff2), title('Gradino basso differenza doppia');
% -- riga 4
subplot(4, 2, 7);
plot(profile_rr_high_diff2), title('Rampa e tetto alto differenza doppia');
subplot(4, 2, 8);
plot(profile_s_high_diff2), title('Gradino alto differenza doppia');
end
```

## Risultati

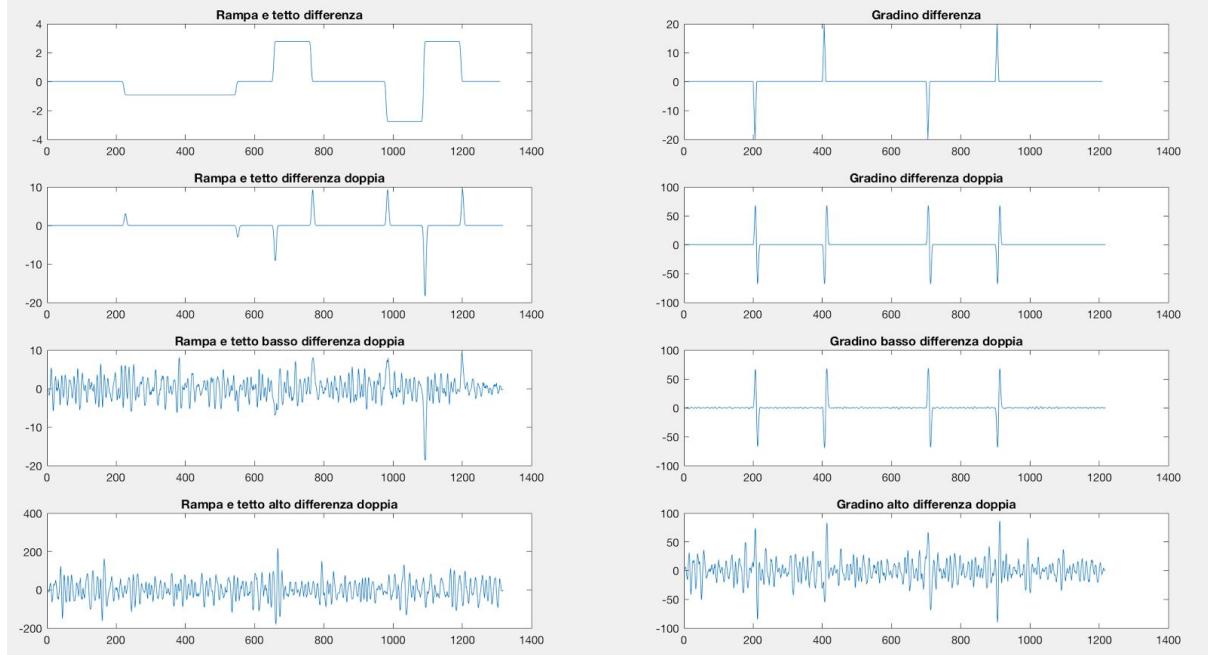
Visualizzazione dell'applicazione dei filtri ai due profili:



Visualizzazione dell'applicazione dei filtri ai due profili ed ai relativi filtri affetti da rumore:



Visualizzazione dell'applicazione doppia del filtro differenza:



## Osservazioni

L'applicazione del filtro differenza nei due segnali evidenzia le variazioni di valori che il segnale mostra. Nel caso di segnali affetti da rumore, le differenze sono più evidenti in quanto i valori del segnale risultano più discontinui.

## Esercizio 2.14

Considerare l'immagine sintetica della barra.

- (a) Calcolare la mappa delle ampiezze e delle direzioni di edge sull'immagine senza rumore effettuando un filtraggio derivativo tipo gradiente di Roberts e di Sobel. Osservare i risultati e riportare le osservazioni nelle conclusioni;
- (b) Sull'immagine della barra con rumore gaussiano additivo con deviazione standard  $stdev=0.1$ , ripetere il calcolo delle mappe di edge con filtri di Roberts e Sobel. Riportare le osservazioni nelle conclusioni;
- (c) Considerare l'immagine ‘blobs.png’ a cui aggiungere rumore gaussiano additivo con  $stdev=0.1$ . Effettuare il calcolo al punto (b) dei gradienti di Roberts e Sobel – soltanto le ampiezze di edge. Riportare le osservazioni;
- (d) Considerare l'immagine ‘circuit.tif’. Calcolare la mappa delle ampiezze e delle direzioni degli edge tramite gradiente di Sobel operando sulla ‘circuit.tif’ senza aggiunta di rumore;
- (e) Considerare l'immagine ‘circuit.tif’. Effettuare il calcolo delle ampiezze di edge con i gradienti di Roberts e Sobel. Sulle immagini delle ampiezze effettuare una sogliatura tramite la `im2bw()`; visualizzare le mappe sogliate (binarizzate) affiancate usando la `imshowpair(..., ..., 'montage')`;

Per il filtraggio di Roberts si deve definire le maschere e usare la sola `imfilter()` di MATLAB; per il filtraggio di Sobel si deve usare le procedure `fspecial()` e `imfilter()`.

Per visualizzare le mappe delle direzioni degli edge candidati usare la funzione `quiver()`.

Non si devono usare le funzioni `imgradient()` e `edge()` di MATLAB.

## Codice

Funzione utilizzata per la definizione delle maschere, l'applicazione dei filtri all'immagine, la visualizzazione delle immagini filtrate, della mappa delle ampiezze e delle direzioni:

```
function[robert, sobel] = esercizio_2_14_apply_rs(img, show)
    % definisco le maschere
    mask_r_x = [-1 0; 0 1];
    mask_r_y = [0 -1; 1 0];
    mask_s_x = fspecial('sobel');
    mask_s_y = mask_s_x';

    % applico i filtri all'immagine senza rumore
    filter_r_x = imfilter(img, mask_r_x);
    filter_r_y = imfilter(img, mask_r_y);
    filter_s_x = imfilter(img, mask_s_x);
    filter_s_y = imfilter(img, mask_s_y);

    robert = abs(filter_r_x) + abs(filter_r_y);
    sobel = abs(filter_s_x) + abs(filter_s_y);

    % visualizzo i risultati
    if strcmp(show, 'show')
        figure;
        % -- riga 1
        subplot(2,3,1);
        imshow(img, title('Immagine originale'));
        subplot(2,3,2);
        imshow(robert, title('Filtro gradiente Robert'));
        subplot(2,3,3);
        quiver(filter_r_x, filter_r_y), title(''), grid on;
        % -- riga 2
        subplot(2,3,5);
        imshow(sobel, title('Filtro gradiente Sobel'));
        subplot(2,3,6);
        quiver(filter_s_x, filter_s_y), title(''), grid on;
    end
end
```

Codice principale per il punto A:

```
function esercizio_2_14()
% (A)
% costruisco l'immagine
img = zeros(50, 50);
img(10:40, 20:30) = 1.0;
% visualizzo l'applicazione dei filtri
esercizio_2_14_apply_rs(img, 'show');
```

Codice principale per il punto B:

```
-- -- -- -- --
% (B)
% costruisco l'immagine con rumore
img_noise = imnoise(img, 'gaussian', 0, 0.1);
% visualizzo l'applicazione dei filtri
esercizio_2_14_apply_rs(img_noise, 'show');
```

Codice principale per il punto C:

```
% (C)
% carico l'immagine blobs.png
img_blobs = imread('blobs.png');
% costruisco l'immagine con rumore
img_blobs_noise = imnoise(im2double(img_blobs), 'gaussian', 0, 0.1);
% visualizzo l'applicazione dei filtri
esercizio_2_14_apply_rs(img_blobs_noise, 'show');
```

Codice principale per il punto D:

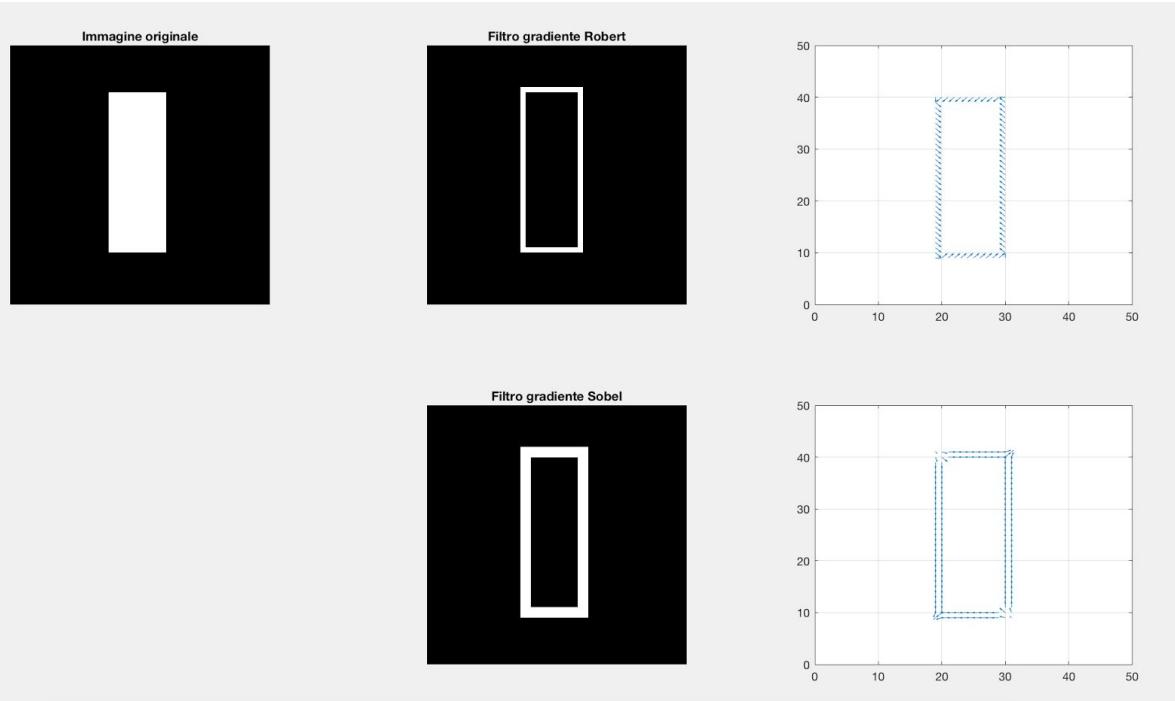
```
% (D)
% carico l'immagine circuit.tif
img_circuit = imread('circuit.tif');
% visualizzo l'applicazione dei filtri
[circuit_r, circuit_s] = esercizio_2_14_apply_rs(img_circuit, 'show');
```

Codice principale per il punto E:

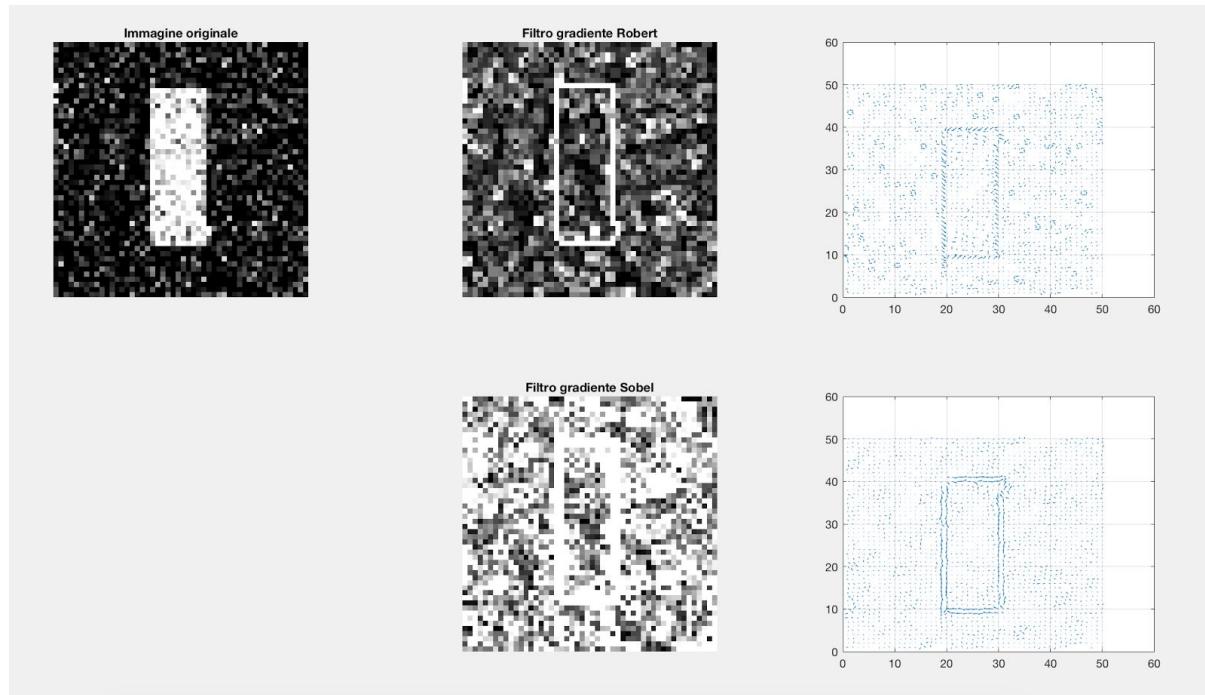
```
% (E)
% effettuo la sogliatura sull'immagine
bw_r = im2bw(circuit_r, 0.1);
bw_s = im2bw(circuit_s, 0.1);
% visualizzo le mappe affiancate
figure;
subplot(1,2,1);
imshowpair(circuit_r, bw_r, 'montage'), title('Mappa binarizzata filtraggio Robert');
subplot(1,2,2);
imshowpair(circuit_s, bw_s, 'montage'), title('Mappa binarizzata filtraggio Sobel');
end
```

## Risultati

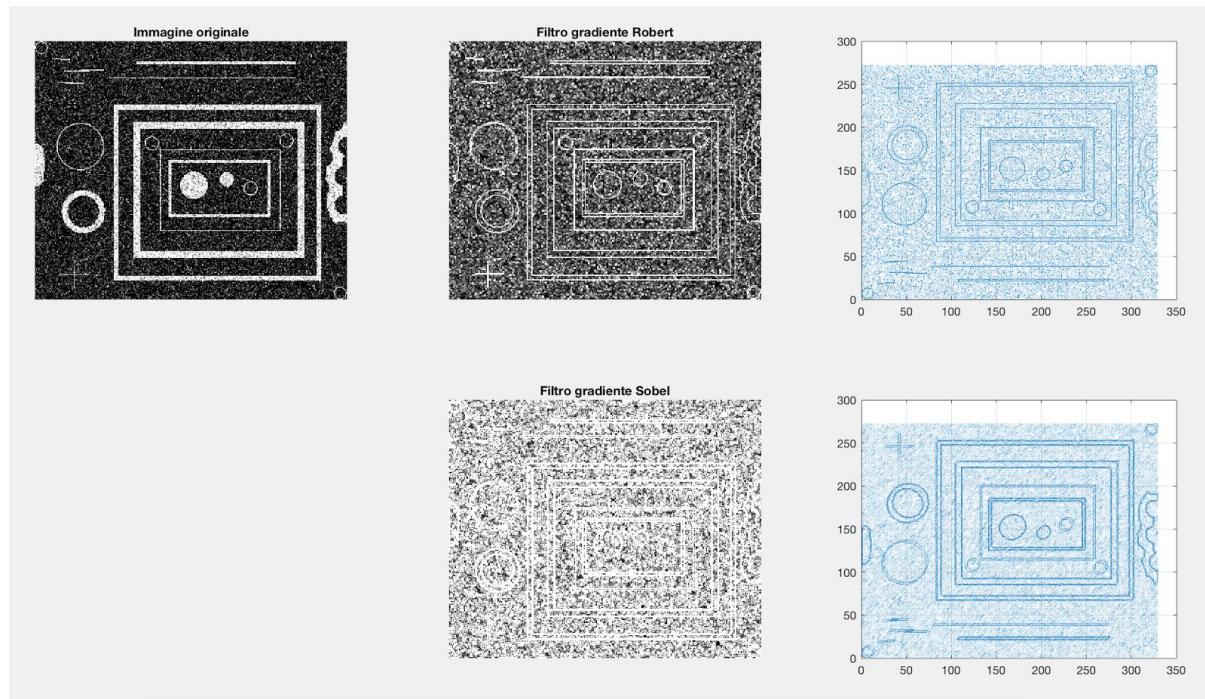
Visualizzazione immagine barra:



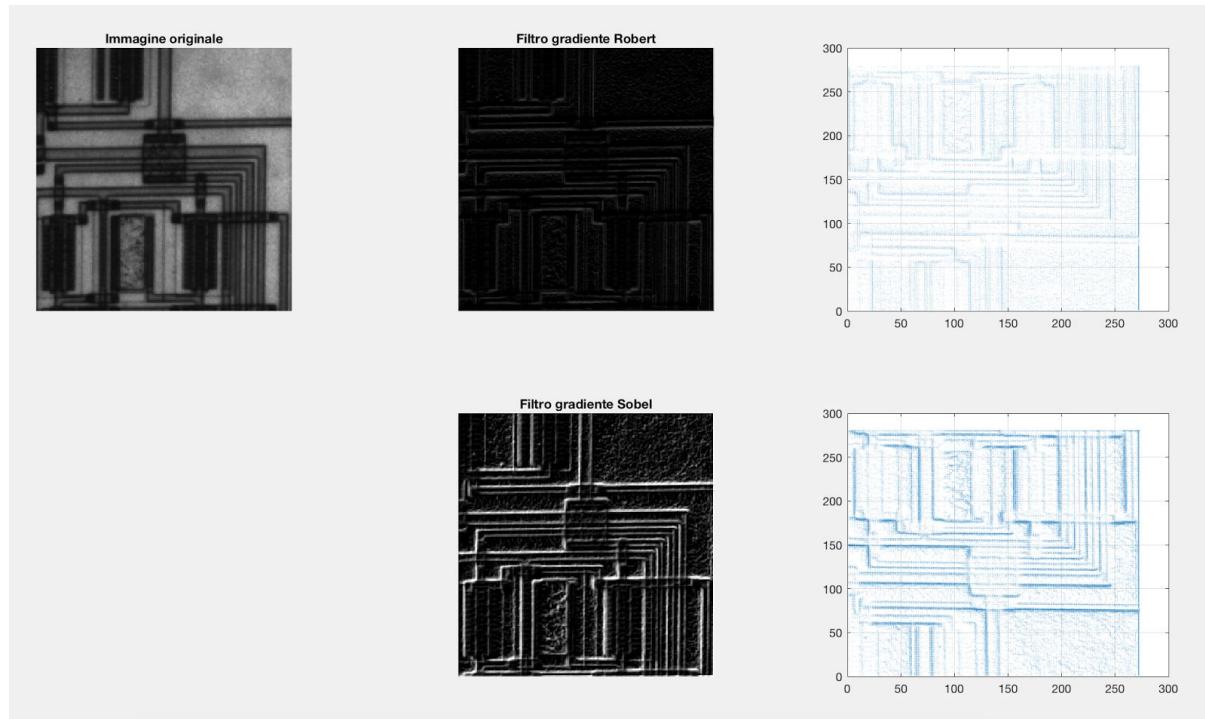
Visualizzazione immagine barra con rumore:



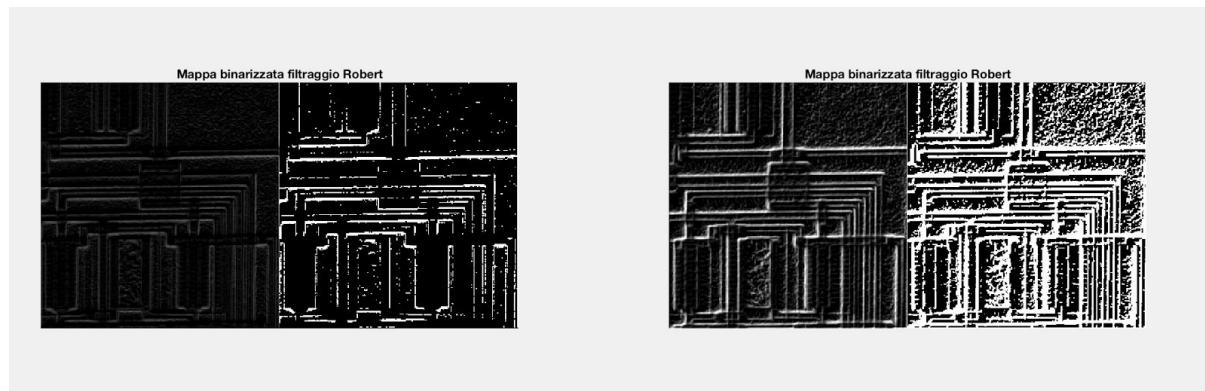
Visualizzazione immagine blobs con rumore:



Visualizzazione immagine circuit:



Visualizzazione mappe binarizzate:



## Osservazioni

Osservazioni A:

Nell'immagine senza rumore della barra l'applicazione del filtro Robert e Sobel hanno permesso di identificare facilmente la mappa delle ampiezze e delle direzioni dell'immagine senza rappresentare falsi positivi.

Osservazioni B:

L'applicazione del rumore all'immagine ha reso più complessa l'estrazione degli edge in quanto i filtri sono sensibili a tali rumori e tendono ad amplificarli nell'immagine risultante.

Tra i due filtri, quello di Sobel, in particolare, tende a rendere quasi irriconoscibile il profilo di edge della figura dell'immagine.

Osservazioni C:

Come per l'immagine della barra con rumore, l'estrazione degli edge è più complessa. In questo caso il profilo di edge risulta avere comunque una riconoscibilità maggiore in quanto la dimensione dell'immagine è maggiore e i pixel colpiti dal rumore risultano meno evidenti rispetto alla figura originale.

## Esercizio 2.15

- (a) Estrazione di edge tramite filtraggio di gradiente di Sobel. Usare l'immagine 'blobs.png'. Calcolare la mappa di ampiezze di edge con l'operatore di Sobel usando la edge() di MATLAB. Calcolare la stessa mappa di ampiezze di edge di Sobel direttamente, usando fspecial(), imfilter(). Binarizzare quest'ultima mappa per selezionare i massimi delle ampiezze (soglia=0.9). Visualizzare l'immagine sorgente. Visualizzare nella stessa finestra grafica le due mappe di ampiezze combinate usando la imshowpair(...,'montage').
- (b) Stimare la precisione delle mappe di ampiezze di Sobel calcolate al punto (a), usando il MSE. Assumere come riferimento l'immagine sorgente. Riportare i risultati usando la uitable(), e verbalmente nelle conclusioni.
- (c) Estrazione di edge tramite filtraggio LoG (Laplaciano di Gaussiana). Usare l'immagine 'coins.png'. Calcolare la mappa di ampiezze di edge 'LoG' usando la funzione edge() con i parametri di default. Effettuare un filtraggio analogo sull'immagine 'mri.tif'. Riportare le osservazioni nelle conclusioni.
- (d) Effettuare tre filtri LoG dell'immagine 'mri.tif', usando la edge() rispettivamente con il parametro sigma = 0.5 ; 2 ; 3 della gaussiana. Riportare le osservazioni nelle conclusioni.

### Codice

```
function esercizio_2_15()
% (A)
% carico l'immagine
img = imread('blobs.png');
dimg = im2double(img);
% calcolo mappa ampiezze con edge()
bw_1 = edge(dimg, 'Sobel');
% calcolo mappa ampiezze usando fspecial() e imfilter()
mask_x = fspecial('sobel');
mask_y = mask_x';
filter_x = imfilter(dimg, mask_x);
filter_y = imfilter(dimg, mask_y);
amp = abs(filter_x) + abs(filter_y);
bw_2 = im2bw(amp, 0.9);
% visualizzo il tutto
figure;
subplot(1, 2, 1)
imshow(img, 'InitialMagnification', 'fit'), title('Originale');
subplot(1, 2, 2)
imshowpair(bw_1, bw_2, 'montage'), title('');
% (B)
% stimo gli mse
mse_1 = immse(im2double(bw_1), dimg);
mse_2 = immse(im2double(bw_2), dimg);
% definisco la uitable da visualizzare
Data = {'Mse edge()', mse_1; 'Mse generato', mse_2};
figure;
uitable('Data', Data, 'Position', [20 20 262 204]);
```

```

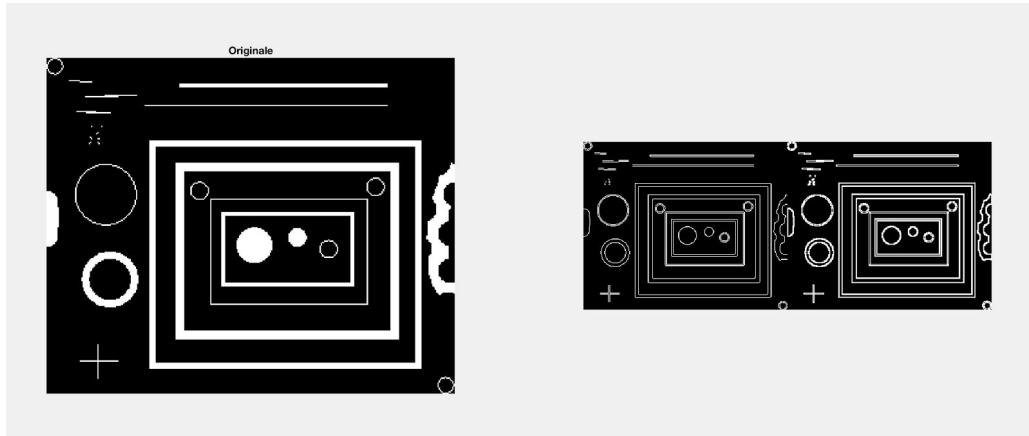
% (C)
% carico l'immagine
img_coins = imread('coins.png');
dimg_coins = im2double(img_coins);
img_mri = imread('mri.tif');
bw_coins = edge(dimg_coins, 'log');
bw_mri = edge(img_mri, 'log');
% visualizzo le immagini
figure;
subplot(2, 2, 1)
imshow(img_coins, 'InitialMagnification', 'fit'), title('Coins');
subplot(2, 2, 2)
imshow(img_mri, 'InitialMagnification', 'fit'), title('Mri');
subplot(2, 2, 3)
imshow(bw_coins, 'InitialMagnification', 'fit'), title('Filtro Coins');
subplot(2, 2, 4)
imshow(bw_mri, 'InitialMagnification', 'fit'), title('Filtro Mri');

% (D)
% effettuo i filtri all'immagine
bw_mri_1 = edge(img_mri, 'log', 0, 'both', 0.5);
bw_mri_2 = edge(img_mri, 'log', 0, 'both', 2);
bw_mri_3 = edge(img_mri, 'log', 0, 'both', 3);
% visualizzo i risultati
figure;
subplot(1, 3, 1)
imshow(bw_mri_1, 'InitialMagnification', 'fit'), title('Filtro 0.5');
subplot(1, 3, 2)
imshow(bw_mri_2, 'InitialMagnification', 'fit'), title('Filtro 2');
subplot(1, 3, 3)
imshow(bw_mri_3, 'InitialMagnification', 'fit'), title('Filtro 3');
end

```

## Risultati

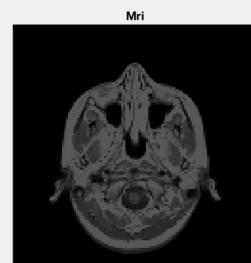
Visualizzazione immagine blobs originale e le relative mappe di ampiezza:



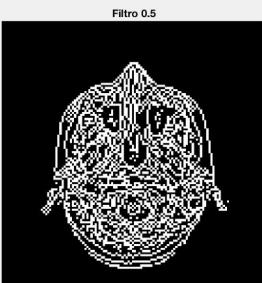
Visualizzazione degli MSE in una componente uitable():

	1	2
1	Mse edge()	0.1794
2	Mse generato	0.1809

Visualizzazione delle immagini coins e mri con i relativi edge di tipo log:



Visualizzazione dell'immagine mri con l'edge di tipo log e sigma di diversi valori:



## Osservazioni

Tra il profilo di edge ottenuto applicando la funzione edge e quello generato e successivamente binarizzato con livello di sogliatura 0.9, il primo risulta avere un MSE migliore.

## Esercizio 2.16

Rilevamento di punti isolati e segmenti rettilinei.

- (a) Definire un'immagine sintetica costituita da soli tre punti isolati; visualizzarla.
- (b) Applicare un filtro derivativo di Laplace usando fspecial() e imfilter() e ottenere la mappa delle ampiezze di edge; visualizzarla anche in modalità grafica 3D, cioè mesh().
- (c) Invertire l'immagine filtrata (negativopositivo) ed eliminare i valori negativi mediante sogliatura usando im2bw(); visualizzare la nuova immagine anche in grafica.
- (d) Definire una nuova immagine sintetica costituita da due segmenti di retta che si incrociano.
- (e) Ripetere il filtraggio e le operazioni effettuate ai punti (b),(c).
- (f) Riportare nelle conclusioni le osservazioni sui risultati ottenuti ai punti (c), (e).

### Codice

Punti A, B, C:

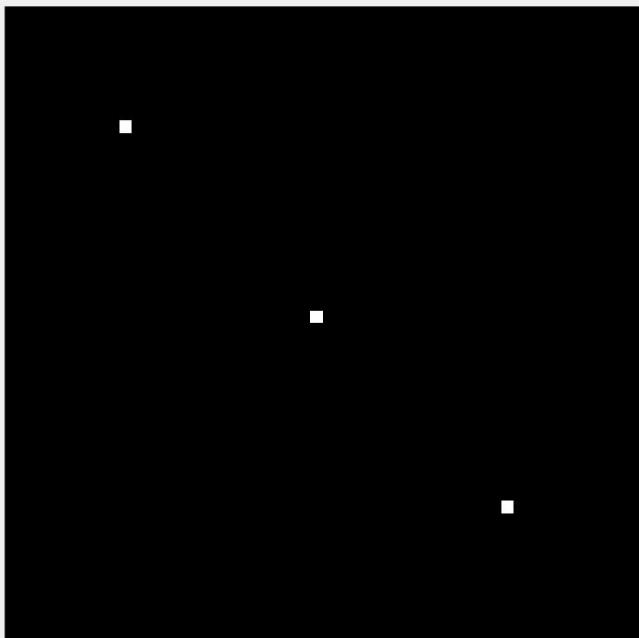
```
% function esercizio_2_16()
% definisco una immagine costituita da tre punti isolati
img = zeros(50,50);
img(10,10)=1;
img(25,25)=1;
img(40,40)=1;
% visualizzo l'immagine generata
imshow(img, 'InitialMagnification', 'fit');
% applico un filtro derivante da Laplace
mask = fspecial('laplacian');
filter = imfilter(img, mask);
% visualizzo il fitro, anche in modalità 3D
figure;
subplot(1, 2, 1)
imshow(filter, 'InitialMagnification', 'fit'), title('Visualizzazione Fitro');
subplot(1, 2, 2)
mesh(filter), title('Visualizzazione 3D');
% inverto l'immagine filtrata ed elimino i valori negativi tramite
% sogliatura
img_inver = imcomplement(filter);
bw = im2bw(img_inver);
% visualizzo risultato
figure;
imshow(bw, 'InitialMagnification', 'fit');
```

Punti D, E:

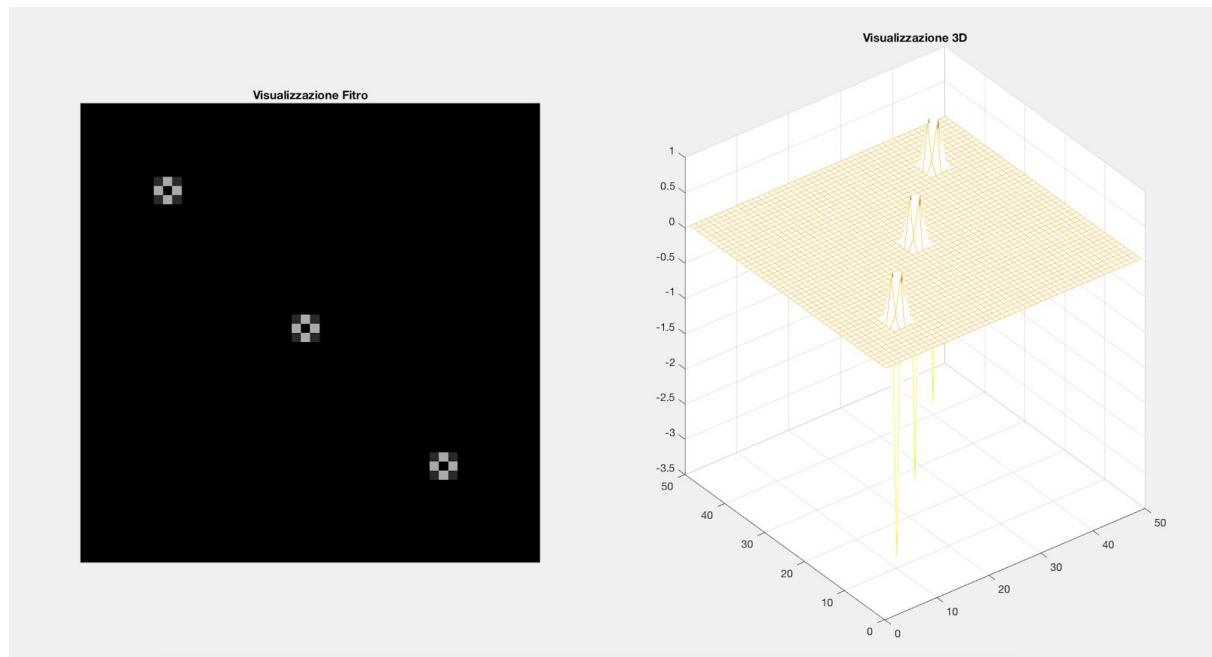
```
% creo una nuova immagine composta da due segmenti intersecati
img_seg = zeros(50,50);
img_seg(5:20, 10:10)=1;
img_seg(10:10, 5:20)=1;
% applico un filtro derivante da Laplace
filter_seg = imfilter(img_seg, mask);
% visualizzo il filtro, anche in modalità 3D
figure;
subplot(1, 2, 1)
imshow(filter_seg, 'InitialMagnification', 'fit'), title('Visualizzazione Filtro');
subplot(1, 2, 2)
mesh(filter_seg), title('Visualizzazione 3D');
% inverto l'immagine filtrata ed elimino i valori negativi tramite
% sogliatura
img_inver_seg = imcomplement(filter_seg);
bw_seg = im2bw(img_inver_seg);
% visualizzo risultato|
figure;
imshow(bw_seg, 'InitialMagnification', 'fit');
end
```

## Risultati

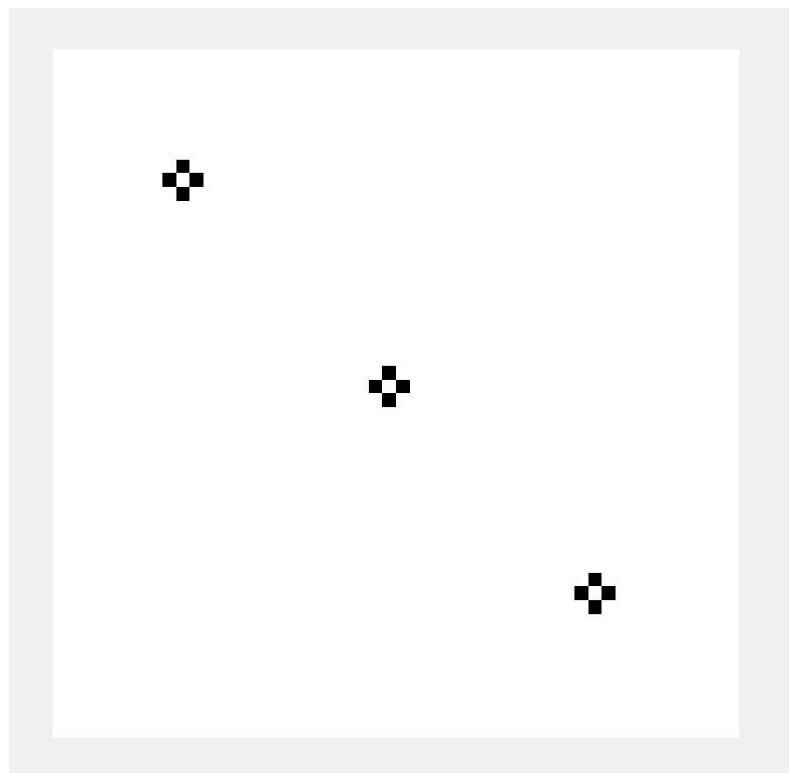
Visualizzazione immagine composta da tre punti:



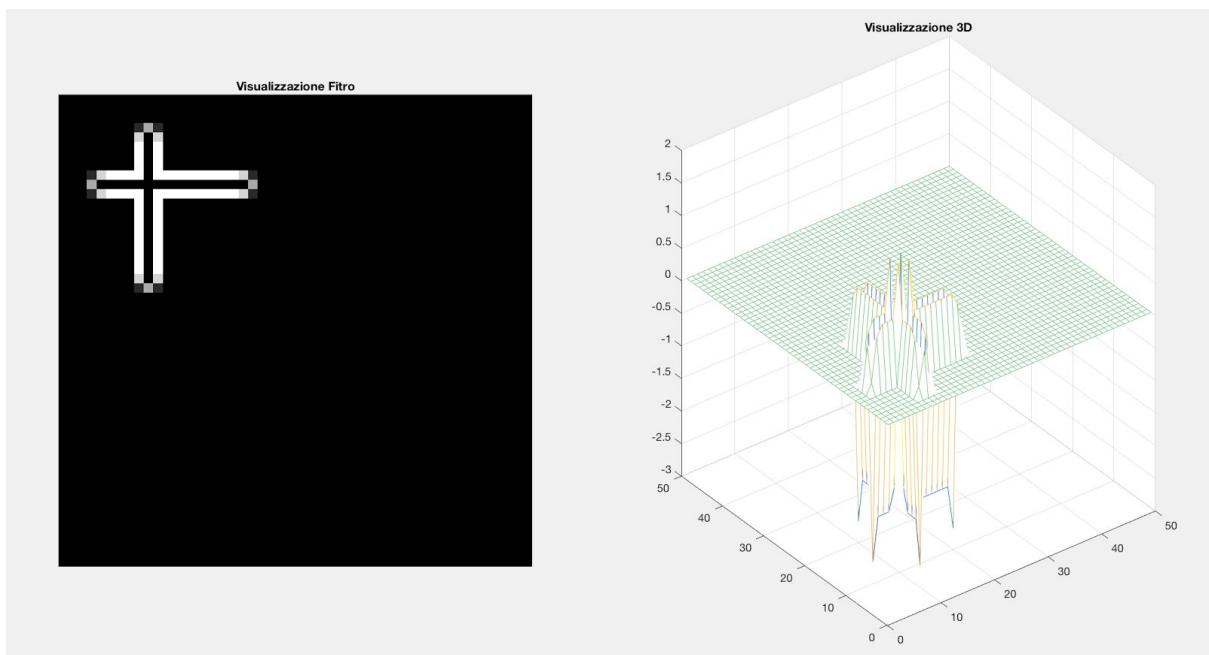
Visualizzazione filtro derivante da Laplace e visualizzazione dello stesso in 3D:



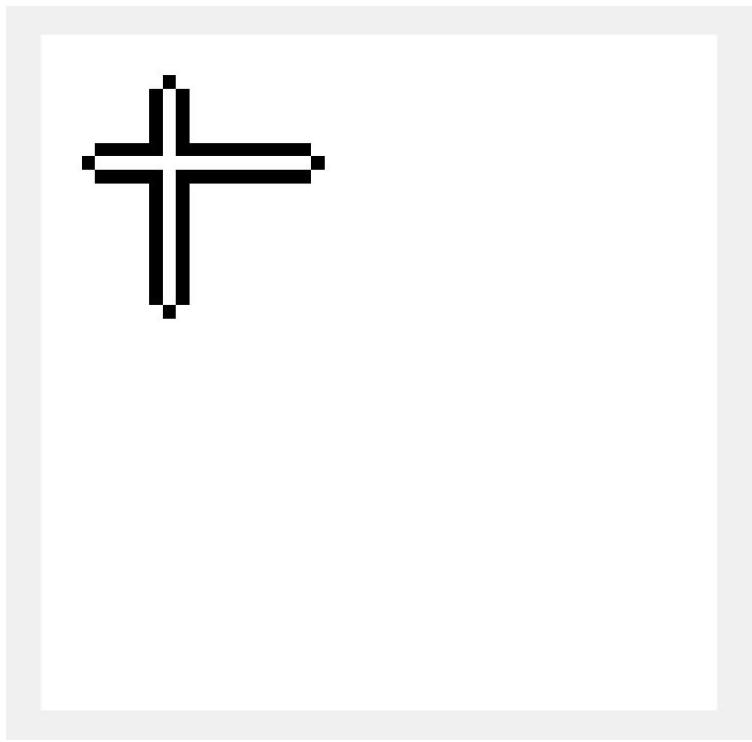
Visualizzazione immagine invertita senza valori negativi:



Visualizzazione filtro derivante da Laplace e visualizzazione dello stesso in 3D (immagine segmenti):



Visualizzazione immagine invertita senza valori negativi (immagine segmenti):



## Osservazioni

Sia nel caso dei punti, sia nel caso delle barre sovrapposte, l'applicazione del filtro di Laplace ha permesso di evidenziare i contorni dell'immagine. La successiva inversione dell'immagine eliminando i valori negativi hanno permesso di costruire una immagine in cui tali contorni sono maggiormente definiti e visibili.

## Esercizio 2.17

Definire un'immagine sintetica costituita da almeno due rettangoli pieni non sovrapposti (barre).

- (a) Studiare la documentazione MATLAB sulle procedure bwconncomp() per l'individuazione di regioni connesse nell'immagine - e regionprops() per la stima di caratteristiche geometriche (features) delle regioni individuate.
- (b) Applicare nell'ordine prima la bwconncomp(), poi la regionprops() all'immagine per estrarre i centroidi, i perimetri, le aree dei rettangoli che vi sono rappresentati.
- (c) Presentare i risultati numericamente per mezzo di una table();
- (d) Sovrapporre all'immagine di partenza i centroidi calcolati e visualizzare i risultati.

### Codice

```
function esercizio_2_17()
% definisco l'immagine composta da due rettangoli non sovrapposti
img = zeros(50,50);
img(10:30, 10:15)=1;
img(10:30, 20:22)=1;
% converto l'immagine in un array di valori logici (true-false)
img_binary = logical(img);
% applico bwconncomp() e visualizzo i valori contenuti
img_info = bwconncomp(img_binary);
% applico regionprops() per ottenere le proprietà dell'immagine
props = regionprops(img_info, {'Centroid','Perimeter','Area'});
% identifico i dati richiesti
centroids = cat(1, props.Centroid);
disp(centroids);
perimeters = cat(1, props.Perimeter);
disp(perimeters);
areas = cat(1, props.Area);
disp(areas);
% visualizzo i risultati in tabella
disp(struct2table(props));
% sovrappongo all'immagine iniziale i centroidi calcolati
figure;
imshow(img), hold on;
plot(centroids(:,1), centroids(:,2), 'm*');
end
```

## Risultati

Tabella con le info ricavate:

<b>Area</b>	<b>Centroid</b>	<b>Perimeter</b>
126	12.5	20
63	21	20

Visualizzazione dei centroidi nell'immagine:

