
Laboratorio di Immagini

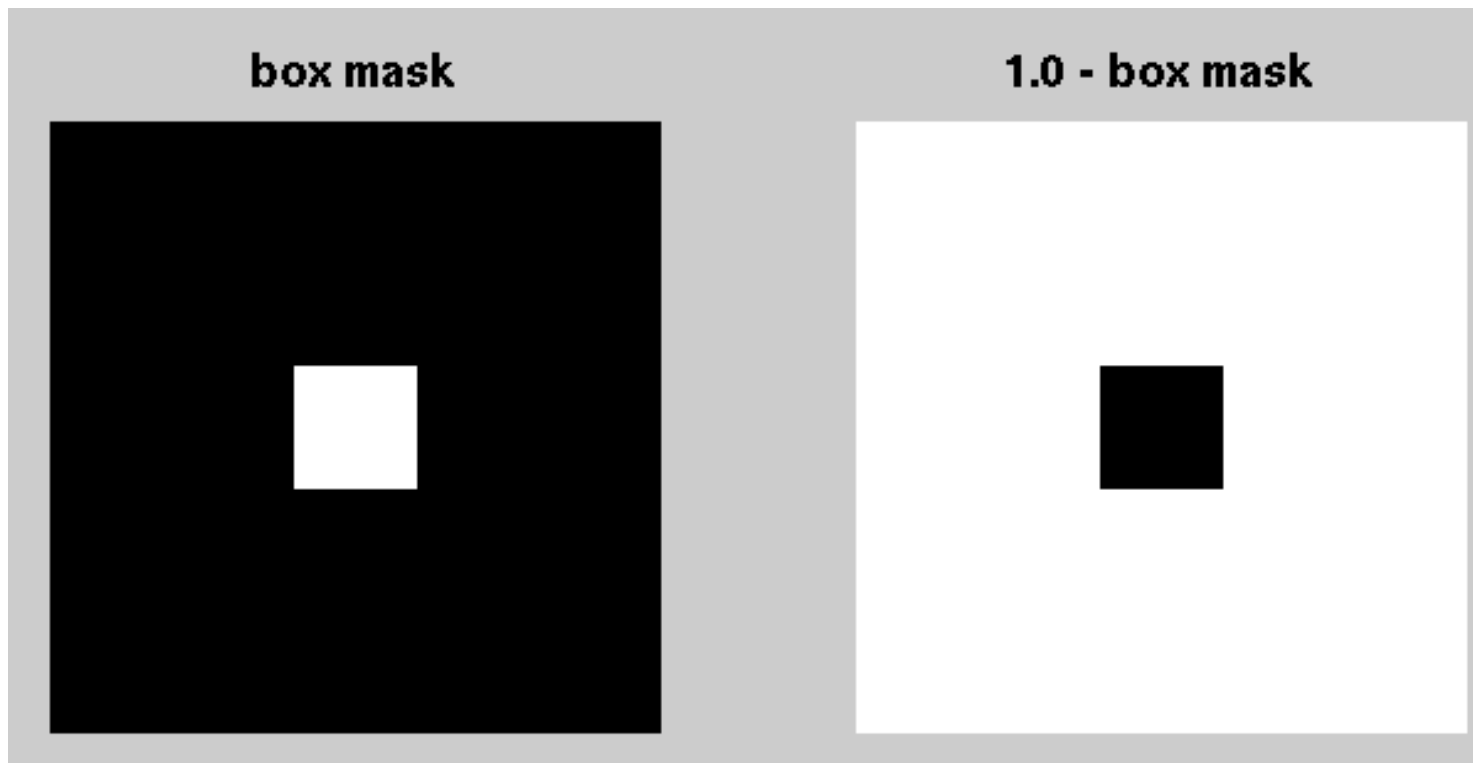
Esercitazione 4:

Filters

Riprendiamo dall'ultima volta

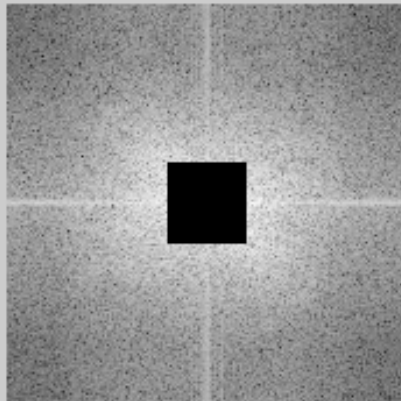
- Abbiamo visto cosa rappresenta la trasformata di Fourier di un'immagine
- In particolare abbiamo provato ad **antitrasformare** il segnale tenendo solo le *basse* o le *alte* frequenze

Box filter

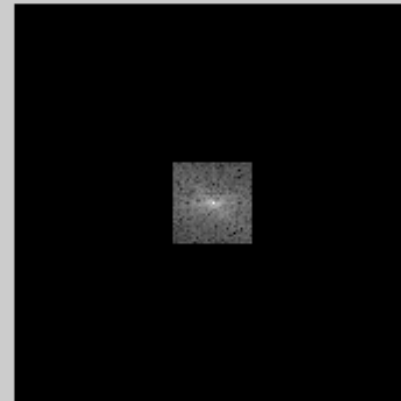


Box filter

high frequencies



low frequencies



Box filter



Problema

- La trasformata di Fourier possiede un importante proprietà:

$$F(\omega)G(\omega) = \mathcal{F}(f(x) * g(x))$$

Problema

- La trasformata di Fourier possiede un importante proprietà:

$$F(\omega)G(\omega) = \mathcal{F}(f(x) * g(x))$$

convoluzione



Convoluzione

$$F(\omega)G(\omega) = \mathcal{F}(f(x) * g(x))$$

- La moltiplicazione di due funzioni nel dominio delle frequenze equivale alla convoluzione dell'antitrasformata delle due funzioni nel dominio spaziale
-

Convoluzione

$$F(\omega)G(\omega) = \mathcal{F}(f(x) * g(x))$$

- La moltiplicazione di due funzioni nel dominio delle frequenze equivale alla convoluzione dell'antitrasformata delle due funzioni nel dominio spaziale

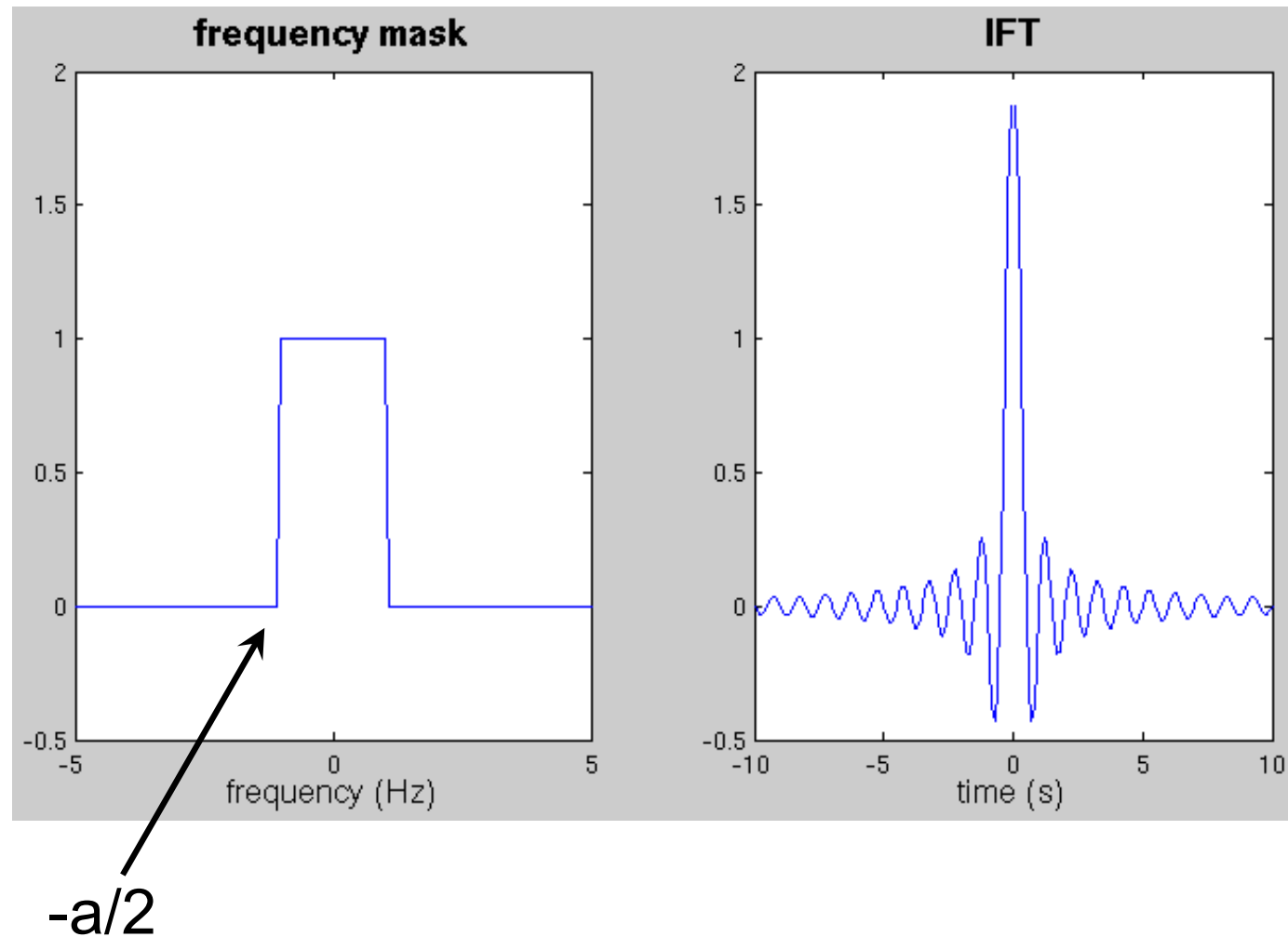
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau)g(x - \tau)d\tau$$

Convoluzione: funzione box

- L'antitrasformata della funzione box corrisponde alla funzione sinc

$$\text{sinc}(x, a) = a \frac{\sin(\pi ax)}{\pi ax}$$

Convoluzione: funzione box



Soluzione

- Invece di prendere come funzione “maschera” nel dominio delle frequenze la funzione box, usiamo un'altra funzione
 - Questa funzione deve avere come antitrasformata una funzione senza “ondine”
-

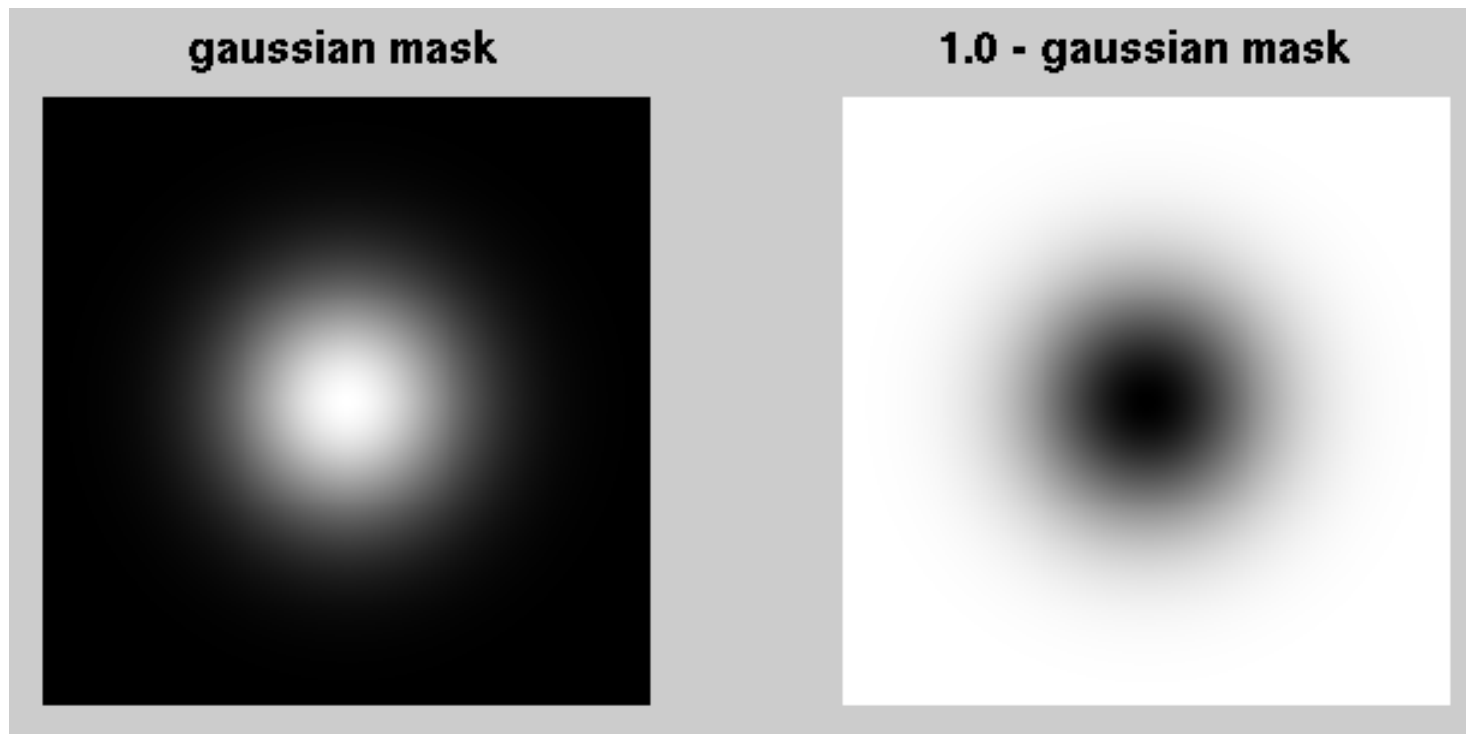
Soluzione

- Invece di prendere come funzione “maschera” nel dominio delle frequenze la funzione box, usiamo un'altra funzione
 - Questa funzione deve avere come antitrasformata una funzione senza “ondine”
 - Per esempio la funzione Gaussiana
-

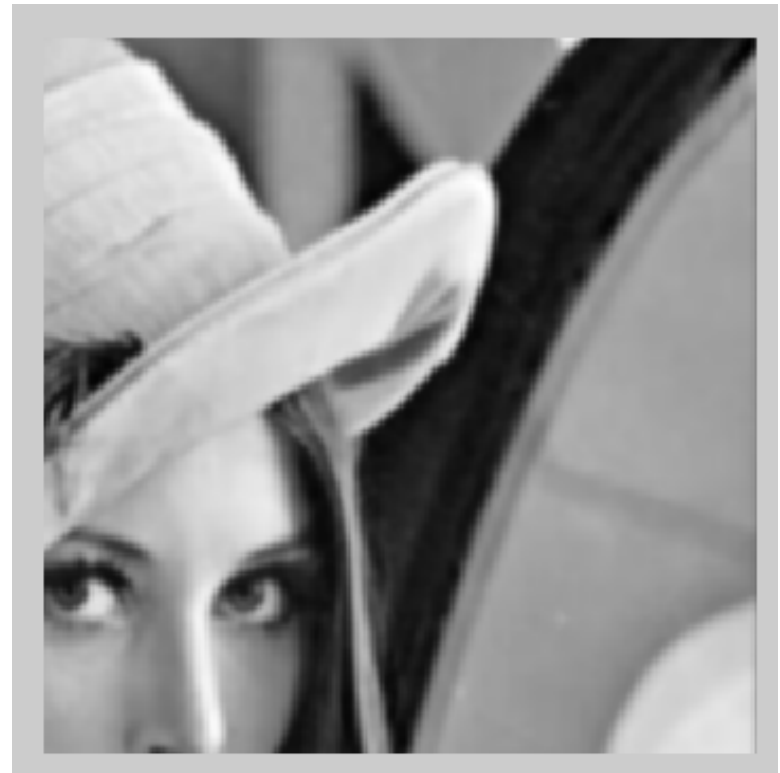
Soluzione

```
img = rgb2gray(imread('lena.jpg'));  
I = fftshift(fft2(img));  
% hight and low frequency  
[x,y] = size(I);  
alpha = 0.0001;  
g_mask = my_gaussian_filter(x,y, alpha);  
figure;  
subplot(1,2,1)  
imshow(g_mask,[])  
title('gaussian mask')  
subplot(1,2,2)  
imshow(1-g_mask,[])  
title('1.0 - gaussian mask')
```

Soluzione



Soluzione



Soluzione

```
I_high = I.*(1-g_mask);  
figure;  
subplot(2,2,1)  
imshow(log(abs(I_high)),[])  
title('high frequencies','FontSize',14,'fontweight','bold')  
I_low = I.*g_mask;  
subplot(2,2,2)  
imshow(log(abs(I_low)),[])  
title('low frequencies','FontSize',14,'fontweight','bold')  
img_high = real(ifft2(ifftshift(I_high)));  
img_low= real(ifft2(ifftshift(I_low)));  
subplot(2,2,3)  
imshow(img_high,[])  
subplot(2,2,4)  
imshow(img_low,[])
```

Filtraggio con convoluzione

- **Idea:** invece di andare a filtrare le immagini nello spazio di Fourier, possiamo farlo direttamente nello spazio delle immagini con la **convoluzione**
 - Questo ci evita di trasformare ed antitrasformare
 - In matlab
 - `fspecial` → definisce i filtri
 - `imfilter` → fa la convoluzione
-

Esempio: filtraggio gaussiano

```
h = fspecial('gaussian',11,10.0)
```

```
img_gaus = imfilter(img,h);
```

```
figure
```

```
subplot(1,3,1)
```

```
imshow(img,[])
```

```
title('original')
```

```
subplot(1,3,2)
```

```
imshow(img_gaus,[])
```

```
title('gaussian filtered')
```

```
subplot(1,3,3)
```

```
imshow(img-img_gaus,[])
```

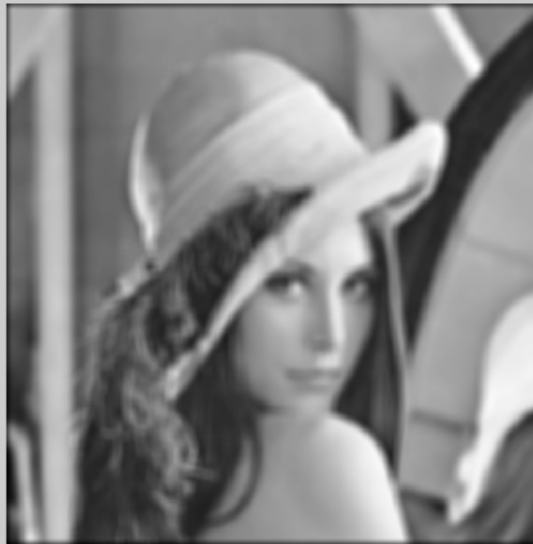
```
title('original-gaussian filtered')
```

Esempio: filtraggio gaussiano

original



gaussian filtered



original-gaussian filtered



Esempio: filtraggio gaussiano

- La versione passa alto del filtro gaussiano si ottiene **sottraendo** all'immagine originale la versione filtrata passa basso
- Se invece invece dell'immagine originale usiamo il risultato di un altro filtraggio gaussiano (con deviazione standard diversa) otteniamo il famoso filtro

DoG:

- **Difference of Gaussians**
-

Esempio: filtro DoG

- **Prima Gaussiana:** $\sigma = 10$
 - **Seconda Gaussiana:** $\sigma = 5$
 - Provate!
-

Esempio: filtraggio gaussiano

original



first gaussian



second gaussian



dog filtered



Esempio: filtraggio Laplaciano

- Il laplaciano è un tipico filtro passa alto:
 - `h = fspecial('laplacian',0.2)`
- Passa alto perché la media di `h` è uguale a zero!

Esempio: filtraggio Laplaciano

original



laplacian filtered



Esempio: filtro LoG

- Laplaciano di Gaussiana
 - `h = fspecial('log',5,0.5)`
- È un filtro passa alto o passa basso?

Esempio: filtro LoG

original



log filtered



filtraggio con operatori derivata

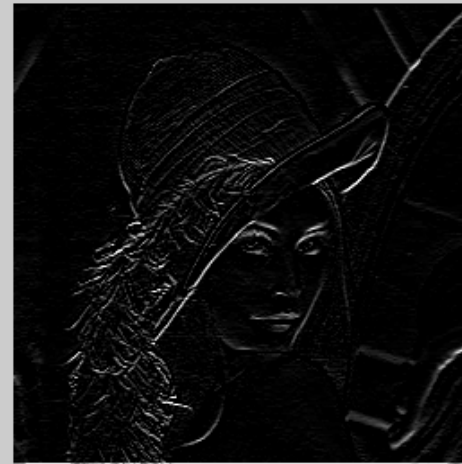
- Filtri di Prewit e Sobel
 - Calcolano il gradiente lungo le righe o lungo le colonne
 - `h = fspecial('prewit');`
 - `h = fspecial('sobel');`
 - In MATLAB l'operatore di default opera lungo le righe (verticale)
 - Per trovare il corrispettivo usate `h'`
-

filtraggio di Prewit

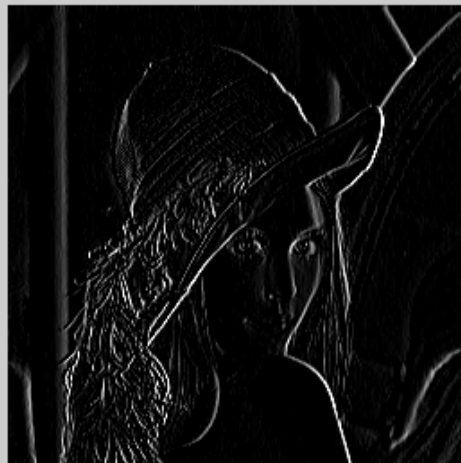
original



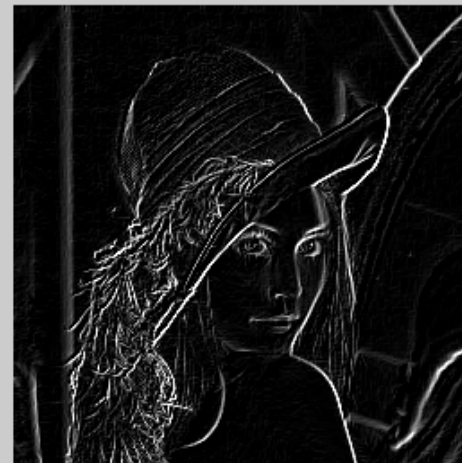
prewit vertical



prewit horizontal



prewit total



filtraggio di Sobel

original



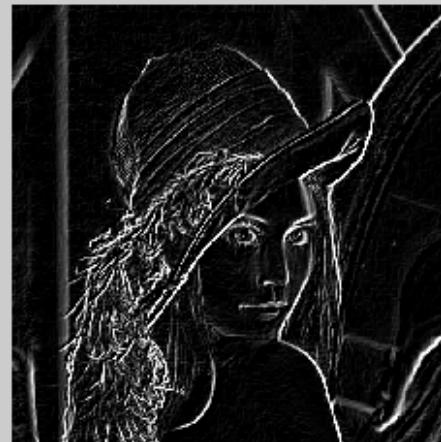
sobel vertical



sobel horizontal



sobel total



Edge detection

- Per trovare bene i contorni a volte è utile applicare una sogliatura al risultato del filtraggio
- Si tengono tutti i valori dell'immagine sopra soglia
 - `imshow(img_sobel>100,[])`

Edge detection

- Per trovare bene i contorni a volte è utile applicare una sogliatura al risultato del filtraggio
 - Si tengono tutti i valori dell'immagine sopra soglia
 - `imshow(img_sobel>100,[])`
 - In MATLAB è più comodo usare la funzione “*edge*”
-

Edge detection challenge

- Caricate *saggittal.png* o *coronal.png*
 - Cercate di ottenere la miglior separazione dei contorni:
 - Materia bianca
 - Materia grigia
 - Lo scalpo
 - Se l'immagine è troppo piccola ricampionata
 - `img = imresize(img,2);`
-

Edge detection challenge

