Elaborazione di Segnali Multimediali

Elaborazioni nel dominio della frequenza
Soluzioni

1 La trasformata discreta 2D

1. *Spettro di fase*. Di seguito si presenta il codice per visualizzare modulo e fase di un'immagine usando le funzioni Numpy `np.fft.fft2` e `np.fft.ifft2`:

```
x = np.float64(io.imread('volto.tif'))
X = np.fft.fft2(x)
Xf = np.fft.fftshift(X)
plt.figure(1)
Z = np.log(1+np.abs(Xf));       # enhancement per visualizzazione
plt.subplot(1,2,1);
plt.imshow(Z,clim=None,cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Spettro di ampiezza');
plt.subplot(122);
plt.imshow(np.angle(Xf), clim=[-np.pi, np.pi],
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Spettro di fase');
```

Proviamo adesso a ricostruire un'immagine con la sola informazione di ampiezza o con la sola informazione di fase:

```
ym = np.real(np.fft.ifft2(np.abs(X)))              # ricostruzione solo modulo
yf = np.real(np.fft.ifft2(np.exp(1j*np.angle(X)))) # ricostruzione solo fase
plt.figure(2)
z = np.log(ym-np.min(ym)+1);     # enhancement per la visualizzazione
plt.subplot(1,2,1); plt.imshow(z, clim=None, cmap='gray');
plt.title('Ricostruzione spettro di ampiezza');
plt.subplot(1,2,2); plt.imshow(yf, clim=None, cmap='gray');
plt.title('Ricostruzione spettro di fase');
```

Infine, proviamo a scambiare spettri di ampiezza e fase di due immagini:

```
x1 = np.float64(io.imread('volto.tif'))
x2 = np.float64(io.imread('rettangolo.jpg'))
X1 = np.fft.fft2(x1)
X2 = np.fft.fft2(x2)
y = np.real(np.fft.ifft2(np.abs(X2) * np.exp(1j*np.angle(X1))))
plt.figure(1);
plt.imshow(y,clim=None,cmap='gray');
plt.title('Modulo rettangolo, Fase volto');
y = np.real(np.fft.ifft2(np.abs(X1) * np.exp(1j*np.angle(X2))))
plt.figure(2);
plt.imshow(y,clim=None,cmap='gray');
plt.title('Modulo volto, Fase rettangolo');
```

2. *Risposta in frequenza dei filtri.* Visualizziamo la risposta in frequenza del filtro media aritmetica per $k = 5, 10, 15$.

```
P = 500; Q = 500;
for k in range(5,20,5):
   h = np.ones((k,k)) / (k**2)
   H = np.fft.fft2(h,(P,Q))
   H = np.abs(np.fft.fftshift(H))

   plt.figure();
   plt.imshow(H, clim=None, cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
   plt.title('Risposta in frequenza del filtro media aritmetica per k=%d' % k)
```

2 Progetto dei filtri in frequenza

1. *Filtraggio notch.* Questo problema richiede un po' di attenzione, infatti prima di procedere nel progetto del filtro è necessario analizzare attentamente sia l'immagine rumorosa che la sua trasformata di Fourier.

```
x = np.float64(io.imread('anelli.tif'))
plt.figure(1);
plt.imshow(x, clim=[0,255], cmap='gray');
plt.title('immagine originale');

M,N = x.shape
X = np.fft.fftshift(np.fft.fft2(x))

plt.figure(2);
plt.imshow(np.log(1+np.abs(X)), clim=None,
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Trasformata di Fourier immagine rumorosa');
```

In effetti dall'immagine si può notare come il fastidio si presenta con delle righe orizzontali, ovvero la sinusoide varia proprio lungo l'asse verticale così come si può osservare nel dominio di Fourier dai picchi di energia presenti proprio lungo l'asse verticale vicino all'origine. Un possibile filtro che rimuove il disturbo sinusoidale è il seguente:

```
# Definizione del filtro
Bl = 0.004; Bk = 0.02;

m = np.fft.fftshift(np.fft.fftfreq(X.shape[0]))
n = np.fft.fftshift(np.fft.fftfreq(X.shape[1]))
l,k = np.meshgrid(n,m)
H1 = (-Bl <= l) & (l <= Bl)
H2 = (-Bk <= k) & (k <= Bk)
H  = (~H1) | (H2 & H1)
plt.figure(3);
plt.imshow(H, clim=[0,1], cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Riposta in frequenza del filtro');
```

```
# Filtraggio
Y = X * H
plt.figure(4);
plt.imshow(np.log(1+np.abs(Y)), clim=None,
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Trasformata di Fourier immagine filtrata');

y = np.real(np.fft.ifft2(np.fft.ifftshift(Y)))
plt.figure(5);
plt.imshow(y, clim=[0,255], cmap='gray');
plt.title('Immagine filtrata');

noise = y - x;
plt.figure(6);
plt.imshow(noise, clim=None, cmap='gray');
plt.title('Rumore eliminato');
```

Un'ultima considerazione. Nel progettare un filtro non esiste una regola fissa, il tipo di problema guida la scelta dl filtro, e tale scelta non è detto che sia l'unica possibile.

2. *Pattern di Moiré.*

```
x = np.float64(io.imread('car.tif'))
M,N = x.shape

plt.figure(1);
plt.imshow(x, clim=[0,255], cmap='gray');
plt.title('immagine originale');

X = np.fft.fftshift(np.fft.fft2(x))

plt.figure(2);
plt.imshow(np.log(1+np.abs(X)), clim=None,
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Trasformata di Fourier immagine originale');

# Definizione del filtro
mu_1 = 0.15; nu_1 = +0.18; B_1 = 0.023;
mu_2 = 0.17; nu_2 = -0.16; B_2 = 0.023;
mu_3 = 0.32; nu_3 = +0.18; B_3 = 0.015;
mu_4 = 0.34; nu_4 = -0.16; B_4 = 0.015;

m = np.fft.fftshift(np.fft.fftfreq(X.shape[0]))
n = np.fft.fftshift(np.fft.fftfreq(X.shape[1]))
l,k = np.meshgrid(n,m)
H_1a = np.sqrt((k-mu_1)**2+(l-nu_1)**2) <= B_1
H_1b = np.sqrt((k+mu_1)**2+(l+nu_1)**2) <= B_1
H_2a = np.sqrt((k-mu_2)**2+(l-nu_2)**2) <= B_2
H_2b = np.sqrt((k+mu_2)**2+(l+nu_2)**2) <= B_2
H_3a = np.sqrt((k-mu_3)**2+(l-nu_3)**2) <= B_3
H_3b = np.sqrt((k+mu_3)**2+(l+nu_3)**2) <= B_3
H_4a = np.sqrt((k-mu_4)**2+(l-nu_4)**2) <= B_4
H_4b = np.sqrt((k+mu_4)**2+(l+nu_4)**2) <= B_4
H = ~(H_1a|H_1b|H_2a|H_2b|H_3a|H_3b|H_4a|H_4b)
plt.figure(3);
plt.imshow(H, clim=[0,1], cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Riposta in frequenza del filtro');

Y = X * H
plt.figure(4);
plt.imshow(np.log(1+np.abs(Y)), clim=None,
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
plt.title('Trasformata di Fourier immagine filtrata');

y = np.real(np.fft.ifft2(np.fft.ifftshift(Y)))
plt.figure(5);
plt.imshow(y, clim=[0,255], cmap='gray');
plt.title('Immagine filtrata');
```