Elaborazione di segnali multimediali

Elaborazioni nel dominio della frequenza

L.Verdoliva, D.Cozzolino

In questa esercitazione esamineremo la trasformata di Fourier discreta, e useremo la DFT-2D per analizzare il contenuto frequenziale di un'immagine. In oltre elaboreremo le immagini nel dominio della frequenza, in particolare realizzeremo sia filtri di smoothing che di sharpening progettandoli direttamente nel dominio di Fourier.

1 La trasformata discreta 2D

Data un'immagine $x(m,n)$ di dimensioni $M \times N$, la DFT-2D (equazione di analisi) è definita come:

$$X(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n) e^{-j2\pi(\frac{mk}{M}+\frac{nl}{N})} \qquad k = 0, \cdots, M-1 \qquad l = 0, \cdots, N-1$$

mentre l'equazione di sintesi è:

$$x(m,n) = \frac{1}{NM} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X(k,l) e^{j2\pi(\frac{mk}{M}+\frac{nl}{N})} \qquad m = 0, \cdots, M-1 \qquad n = 0, \cdots, N-1$$

In Numpy la DFT bidimensionale è implementata tramite il comando `np.fft.fft2`, che di default valuta la DFT su un numero di punti pari proprio alle dimensioni dell'immagine. Scriviamo il codice per realizzare e visualizzare il modulo della trasformata di Fourier dell'immagine costituita da un rettangolo bianco su sfondo nero:

```
x = np.float64(io.imread('rettangolo.jpg'))
X = np.fft.fft2(x)
plt.figure(); plt.imshow(np.abs(X), clim=None, cmap='gray');
```

L'immagine visualizzata è tutta nera eccetto per pochissimi valori chiari agli angoli dell'immagine. In effetti per una più corretta visualizzazione è necessario usare il comando `np.fft.fftshift` per traslare le basse frequenze della DFT al centro dell'immagine, e poi realizzare l'operazione di logaritmo per migliorare la visualizzazione dei coefficienti, che presentano una dinamica estremamente ampia.

```
Y = np.log(1+np.abs(np.fft.fftshift(X)))
plt.figure();
plt.imshow(Y, clim=None, cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
```

Inoltre con l'opzione `extent=(left, right, bottom, top)` di `plt.imshow` visualizziamo gli assi nel corretto range delle frequenze spaziali $[-1/2, 1/2] \times [-1/2, 1/2]$.

Il comando `np.fft.fft2` consente anche di specificare il numero di punti su cui calcolare la DFT, e se superiori alle dimensioni dell'immagine, realizza automaticamente lo zero-padding: `X = np.fft.fft2(x,(P,Q))`.

```
M,N = x.shape
P = 2*M; Q = 2*N
X = np.fft.fft2(x, (P,Q))
```

Un altro modo per rappresentare il modulo della trasformata di Fourier è utilizzando un grafico 3d. Per realizzare il grafico 3d è necessario prima creare la griglia di punti su cui è definita la trasformata:

```
Y = np.log(1+np.abs((np.fft.fftshift(X))))
m = np.fft.fftshift(np.fft.fftfreq(Y.shape[0]))
n = np.fft.fftshift(np.fft.fftfreq(Y.shape[1]))

from mpl_toolkits.mplot3d import Axes3D
ax = Axes3D(plt.figure());  # crea una figura per i grafici 3d

l,k = np.meshgrid(n,m)
ax.plot_surface(l,k,Y, linewidth=0, cmap='jet')
```

1.1 Esercizi proposti

1. *Spettro di ampiezza.* Analizzate lo spettro di ampiezza di alcune immagini di test (circuito.jpg, impronta.tif, anelli.tif), siete in grado di legare il contenuto in frequenza con l'andamento spaziale dell'immagine?

2. *Spettro di fase.* Per comprendere l'importanza dello spettro di fase provate a realizzare il seguente esperimento. Considerate l'immagine volto.tif e dopo averne calcolato e visualizzato spettro di ampiezza e di fase, ricostruite l'immagine con la sola informazione di ampiezza e poi solo con quella di fase e confrontate le due immagini.

   Infine, provate a ricostruire l' immagine usando una volta il suo spettro di ampiezza, ma quello di fase relativo all'immagine rettangolo.jpg e una seconda volta il suo spettro di fase, ma quello di ampiezza sempre relativo all'immagine rettangolo.jpg.

3. *Risposta in frequenza del filtro media aritmetica.* Calcolate la DFT del filtro media aritmetica di dimensione $k = 5, 10, 15$, utilizzando per la FFT un numero di punti relativamente elevato, allo scopo di ottenere un campionamento fine di $H(\nu, \mu)$. Verificate il comportamento passa-basso del filtro.

2 Uso della DFT per il filtraggio lineare

Lo studio dei segnali e dei sistemi nel dominio della frequenza è uno strumento fondamentale per l'elaborazione dei segnali, infatti nel dominio della frequenza il prodotto tra la trasformata di Fourier dell'ingresso e quella della risposta impulsiva equivale alla convoluzione realizzata nel dominio spaziale. Il filtraggio nel dominio frequenziale risulta computazionalmente molto efficiente grazie proprio ad algoritmi veloci (FFT) per il calcolo della DFT. In realtà se si realizza il prodotto delle DFT dell'immagine di ingresso, $X(k, l)$, e della

risposta impulsiva $H(k,l)$ si ottiene la convoluzione circolare delle immagini, che in generale non è uguale alla convoluzione lineare, operazione a cui siamo interessati per realizzare il filtraggio nel discreto. Tuttavia, si può dimostrare, che realizzando opportunamente uno zero padding dell'immagine di partenza la convoluzione circolare coincide con quella lineare. Quindi pur di effettuare il riempimento con zeri, la DFT può essere usata per realizzare il filtraggio lineare.

Cominciamo ad esplorare questi concetti nel caso monodimensionale per poi passare alle immagini. Proviamo allora a determinare attraverso la DFT e la IDFT la risposta del filtro con $x = [1,2,3,4]$ e $h = [1,1,1]$. In questo caso bisogna calcolare la DFT almeno su $N = 4+3-1 = 6$ punti per garantire la correttezza del risultato:

```
x = np.array([1,2,3,4], dtype=np.float64)   # vettore di ingresso
h = np.array([1,1,1]  , dtype=np.float64)   # risposta impulsiva
X = np.fft.fft(x,6)                         # realizza la DFT su 6 punti
H = np.fft.fft(h,6)
Y = X * H
y = np.fft.ifft(Y,6)
```

Verificate che l'uscita è la stessa che fornisce il comando `y = np.convolve(x,h)`, ripetete quindi l'operazione di filtraggio in frequenza calcolando la DFT su 4 punti, e verificando che il risultato è lo stesso di quello che otterreste con un'estensione periodica ai bordi.

Se si considera invece un'immagine, $x[m,n]$, di dimensioni $M \times N$ e un filtro $h[m,n]$ di dimensioni $A \times B$, è necessario calolare la DFT-2D su un numero di punti P e Q che soddisfino le condizioni $P \geq M + A - 1$ e $Q \geq N + B - 1$. Vediamo un esempio di implementazione nel dominio della frequenza del *filtro di Sobel* per la rivelazione dei bordi orizzontali. La risposta impulsiva del filtro è:

$h =$

1	0	-1
2	0	-2
1	0	-1

Il codice Python per filtrare in frequenza un'immagine contenuta nella variabile `x` con tale filtro è il seguente:

```
h = np.array([[1,0,-1],[2,0,-2],[1,0,-1]], dtype=np.float64)
M,N = x.shape
A,B = h.shape
P = M + A -1
Q = N + B -1
X = np.fft.fft2(x, (P,Q))
H = np.fft.fft2(h, (P,Q))
Y = H * X
y = np.real(np.fft.ifft2(Y))
plt.figure(); plt.imshow(y, clim=None, cmap='gray');
```

A rigore, il risultato della DFT inversa dovrebbe essere puramente reale, in quanto equivalente alla convoluzione nel dominio dello spazio di due segnali reali. In pratica, errori dovuti alla precisione limitata comportano che l'antitrasformata possieda una piccola parte immaginaria, che viene trascurata usando la funzione `np.real`.

Questa procedura risulta computazionalmente più efficiente del calcolo diretto della convoluzione, solo se il filtro è molto lungo (almeno maggiore di 32). Verificate che il risultato ottenuto nel dominio della frequenza è

praticamente uguale a quello che si può ottenere nel dominio spaziale (fate attenzione al fatto che la maschera non è simmetrica).

3 Progetto dei filtri in frequenza

Proviamo adesso a progettare un filtro direttamente nel dominio della frequenza, e consideriamo un filtro passa-basso ideale, che ha come funzione di trasferimento:

$$H(\mu,\nu) = \begin{cases} 1 & D(\mu,\nu) \leq D_0 \\ 0 & \text{altrimenti} \end{cases}$$

dove $D(\mu,\nu) = \sqrt{\mu^2+\nu^2}$ e rappresenta la distanza dal punto (ν,μ) al centro del filtro, mentre D_0 rappresenta la frequenza di cut-off. Questo filtro ideale può essere simulato al calcolatore col seguente codice:

```
M,N = x.shape
m = np.fft.fftshift(np.fft.fftfreq(M))
n = np.fft.fftshift(np.fft.fftfreq(N))
l,k = np.meshgrid(n,m)
D   = np.sqrt(k**2+ l**2)
D0  = 0.1;
H   = (D <= D0)
plt.figure();
plt.imshow(H, clim=[0,1], cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));
```

A questo punto usiamo questo filtro per elaborare in frequenza l'immagine lena.jpg, visualizzando il risultato sia nel dominio frequenziale che in quello spaziale:

```
x = np.float64(io.imread('lena.jpg'))
X = np.fft.fft2(x)
X = np.fft.fftshift(X)
plt.figure();
plt.imshow(np.log(1+np.abs(X)), clim=None, \
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));

Y = H * X
plt.figure();
plt.imshow(np.log(1+np.abs(Y)), clim=None, \
    cmap='gray', extent=(-0.5,+0.5,+0.5,-0.5));

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

Provate a cambiare la dimensione del raggio e visualizzate l'uscita. Noterete l'effetto di *ringing* che caratterizza un filtraggio ideale e che può essere sensibilmente ridotto se si usano filtri gaussiani con funzione di trasferimento:

$$H(\mu,\nu) = e^{-D^2(\mu,\nu)/2\sigma^2}$$

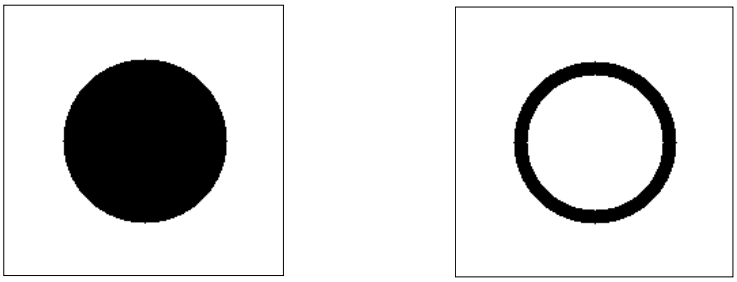

Figura 1: Esempio di filtro passa-alto ed elimina-banda (bianco è 1 e nero è 0).

dove al crescere di σ aumenta la banda del filtro. Scrivete una funzione con il prototipo `gaussLPF(x,sigma)` che implementa il filtraggio passa-basso gaussiano con un assegnato valore di σ. Quindi applicate questo filtro all'immagine `testo.tif`, che mostra un esempio di testo a bassa risoluzione che caratterizza di solito un fax o una fotocopia. Il testo in questa forma si presta poco ad essere elaborato da un calcolatore che effettua riconoscimento automatico, dato che sono presenti dei vuoti nei caratteri, un opportuno filtro passa-basso può migliorarne la visualizzazione. Questo filtro può anche essere usato per ringiovanire l'aspetto della donna raffigurata nell'immagine `volto.jpg`. Fate degli esperimenti, in entrambi i casi, al variare di σ.

Infine, implementate i filtri ideali passa-alto ed elimina-banda la cui risposta armonica visualizzata come immagine è mostrata in figura 1. Usateli per filtrare l'immagine lena.jpg e osservate il risultato al variare del raggio.

3.1 Filtraggio di rumore periodico

L'immagine contenuta nel file lenarumorosa.y (512×512, int16) mostra un disturbo periodico (righe verticali) che la rende sgradevole. Vogliamo progettare un filtro in grado di rimuovere solo la porzione dello spettro relativo alla sinusoide. A tal fine, valutiamo la trasformata di Fourier dell'immagine:

```
x = np.fromfile('lenarumorosa.y', np.int16)
x = np.reshape(x, [512,512])
x = np.float64(x)
plt.figure(); plt.imshow(x, clim=[0,256], cmap='gray');
plt.title('immagine rumorosa');

X = np.fft.fftshift(np.fft.fft2(x));
plt.figure();
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');
```

Notate che teoricamente la sinusoide dovrebbe essere rappresentata da due impulsi in frequenza. Nella pratica non abbiamo degli impulsi ideali, bensì due sinc. Questo è dovuto al fatto che il rumore sinusoidale aggiunto è limitato all'immagine, cioè risulta troncato nello spazio lungo le due direzioni attraverso una finestra rettangolare bidimensionale. Il prodotto tra la sinusoide e la finestra in frequenza è la convoluzione tra i due impulsi e la sinc bidimensionale (trasformata dell'impulso rettangolare), quindi due sinc localizzate proprio alle frequenze specificate dalla sinusoide. Un possibile filtro che rimuove la sinuoside è il seguente:

```
# Definizione del filtro
nu = 0.2; B = 0.03;
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)
D1 = np.sqrt(k**2+(l-nu)**2)
D2 = np.sqrt(k**2+(l+nu)**2)
H  = (D1>B) & (D2>B)

plt.figure();
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();
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(); plt.imshow(y, clim=[0,256], cmap='gray');
plt.title('Immagine filtrata');

# Calcolo MSE
xo = np.fromfile('lena.y', np.uint8)
xo = np.reshape(xo, [512,512])
xo = np.float64(xo)

plt.figure; plt.imshow(xo, clim=[0,256], cmap='gray');
plt.title('immagine originale');

MSE = np.mean((xo-y) ** 2)
```

In realtà, con questo filtro non si riescono a rimuovere perfettamente le righe verticale, a causa della presenza dei lobi laterali, che presentano valori non trascurabili. E' possibile progettare un filtro rettangolare molto stretto che (conservando i valori intorno all'origine che corrispondono al contenuto frequenziale rilevante del segnale) sia diretto proprio lungo l'asse orizzontale.

```
# Definizione del filtro
Bk = 0.004; Bl = 0.02
H1 = (-Bk <= k) & (k <= Bk)
H2 = (-Bl <= l) & (l <= Bl)
H = (~H1) | (H2 & H1)
plt.figure();
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
y = np.real(np.fft.ifft2(np.fft.ifftshift(Y)));

plt.figure(); plt.imshow(y, clim=[0,256], cmap='gray');
plt.title('Immagine filtrata');

MSE = np.mean((xo-y) ** 2)
```

Notate come l'errore quadratico medio tra immagine filtrata e originale (contenuta nel file lena.y, uint8) diminuisca con usando questa seconda soluzione.

3.2 Esercizi proposti

1. *Filtraggio notch.* L'immagine anelli.tif mostra una parte degli anelli che circondano Saturno. Il rumore sinusoidale è dovuto ad un segnale AC sovrapposto a quello della fotocamera prima di digitalizzare l'immagine. Tale interferenza è semplice da rimuovere se si progetta un filtro notch in grado di cancellare il contributo del rumore. Calcolate quindi la trasformata di Fourier dell'immagine, analizzatela, individuate il contributo realtivo al segnale sinusoidale e cercate di eliminarlo con il filtraggio.

2. *Pattern di Moiré.* L'immagine car.tif è caratterizzata dal pattern di moiré, un artefatto piuttosto fastidioso che può essere generato da una scansione non appropriata di una fotografia stampata su di un giornale. Dopo aver osservato attentamente la trasformata di Fourier dell'immagine, scrivete il codice per rimuovere questo disturbo attraverso un opportuno filtro ideale e mostrate l'immagine risultante.

3. *Immagini ridimensionate.* Un algoritmo che permette di rivelare se un'immagine $x(m,n)$ di dimensioni $M \times N$ è stata ricampionata, per esempio a causa di uno zoom, prevede di:

   (a) calcolare la derivata seconda lungo le righe dell'immagine:
   $d_2(m,n) = 2x(m,n) - x(m,n+1) - x(m,n-1)$;

   (b) valutare la pseudo-varianza come: $v(n) = \sum_{m=0}^{M-1} |d_2(m,n)|$ e la sua derivata prima $d(n) = v(n) - v(n-1)$;

   (c) identificare le periodicità di $d(n)$ nel dominio di Fourier tramite DFT su $N-2$ punti. A tal fine mostrare $|D(\nu)|$ nell'intervallo $(-1/2, 1/2)$;

   (d) rivelare i picchi che identificano le periodicità per stimare il fattore di scala R. In particolare, una volta identificata la frequenza ν_0 cui si trova il picco nell'intervallo $(0, 1/2)$, il fattore di scala è stimato come $R = 1/\nu_0$.

   Scrivete una funzione dal prototipo `function [d_hori,d_vert,R_hori,R_vert] = detect(x)` in cui si realizzano i passi descritti sia operando lungo le righe che lungo le colonne dell'immagine. Applicate l'algoritmo allo zoom dell'immagine cameraman contenuto in zoom.y (128×128, float), mostrate il grafico ottenuto al punto 3 (lungo le righe e le colonne) e determinate il fattore di scala dell'ingrandimento lungo le due direzioni.

4. *Immagini sintetiche.* Un modo per distinguere i volti sintetici da quelli reali è attraverso l'analisi nel dominio della frequenza. In particolare scrivete il codice per realizzare i seguenti passi:
   (a) convertite l'immagine in scale di grigi considerando la dinamica $[0, 1]$;
   (b) calcolate $H(\mu, \nu)$ applicando la seguente formula:
   $$H(\mu, \nu) = \log |X(\mu, \nu)|$$
   devo $X(\mu, \nu)$ è la trasformata di Fourier dell'immagine.
   (c) Calcolate la densità alle basse frequenze come:
   $$d = \frac{\sum_{\mu,\nu:|\mu|\leq\tau,|\nu|\leq\tau} H(\mu, \nu)}{\sum_{\mu,\nu} H(\mu, \nu)}$$
   dove τ è la soglia in frequenze pari a 0.35. Se il valore d è inferiore a 0.7, l'immagine ha una distribuzione meno concentrata alle basse frequenze rispetto alle immagini naturali e quindi risulta generata sinteticamente.

   Utilizzando tale procedura, stabilire se le immagini siano reali o sintetiche (volto1, volto2, volto3 e volto4).

5. *Immagini contraffatte.* Nell'immagine mare.png una parte della scogliera è stata duplicata. Si vuole individuare la regione copiata calcolando il secondo picco dell'autocorrelazione dell'immagine. Per ottenere $R_x(l, k)$, la la funzione di autocorrelazione dell'immagine $x(m, n)$, calcolate la trasformata di Fourier $\mathcal{F}[x(m, n)] = X(\mu, \nu)$, prendete il modulo quadro, e infine effettuate l'antitrasformata (per ridurre la complessità non considerate zero-padding):
   $$R_x(l, k) = \mathcal{F}^{-1}\left[|X(\mu, \nu)|^2\right]$$
   Realizzate quindi i seguenti passi:
   (a) Visualizzate (usando il comando `mesh`) la funzione di autocorrelazione, usando opportunamente il comando `fftshift` per avere il picco al centro dell'immagine. Le coordinate spaziali andranno così da $-N/2$ ad $N/2 - 1$, con N numero di righe/colonne. Sul grafico sarà possibile notare un secondo picco lontano dall'origine.
   (b) Per individuare i picchi, prendete per ogni punto una finestra 5×5 e marcate il punto come picco solo se è il massimo nella finestra. Il secondo di tali picchi è quello cercato.
   (c) A questo punto, note le coordinate (m_0, n_0) del secondo picco, calcolate la differenza fra l'immagine di partenza e quella traslata di tali coordinate e marcate i punti per cui tale differenza è nulla. Nell'immagine risultante sarà chiaramente visibile la regione duplicata.

6. *Liveness detection.* Per scoprire se un'impronta digitale è autentica o contraffatta si può operare nel dominio di Fourier, calcolando la frazione di energia contenuta alle medie frequenze, e dichiarando l'immagine autentica se tale frazione supera una certa soglia. Scrivete una funzione `function EM = elabora(x, r1, r2)` che calcola la DFT-2D, $X(\mu, \nu)$, di un'immagine $x(m, n)$, quindi applica il filtro con risposta in frequenza:
   $$H(\mu, \nu) = \begin{cases} 1 & r_1 \leq |\nu| \leq r_2, \;\; r_1 \leq |\mu| \leq r_2, \\ 0 & \text{altrimenti} \end{cases}$$
   ottenendo l'immagine $Y(\mu, \nu)$. A questo punto si valuta $E_M = \frac{1}{|\Omega|} \sum_\Omega |Y(\mu, \nu)|^2$ dove Ω è l'insieme dei punti in cui $Y(\mu, \nu)$ assume valori diverso da zero mentre $|\Omega|$ è la cardinalità di Ω.

   Applicate la funzione alle due immagini impronta1.tif e impronta2.tif, usando raggio interno $r_1 = 0.10$ e raggio esterno $r_2 = 0.25$ ed etichettate come vera quella che fornisce il valore maggiore di E_M/E, con E energia dell'immagine.