

Elaborazione di Segnali Multimediali
a.a. 2017/2018

Le trasformate DCT e Wavelet

L. Verdoliva

In questa esercitazione realizziamo la trasformata coseno discreta (DCT) e la trasformata wavelet. Per la DCT visualizzeremo le funzioni base sia nel caso monodimensionale che bidimensionale e mostreremo la sua capacità di compattazione dell'energia. Per la Wavelet useremo l'implementazione mediante banco di filtri, evidenziando la capacità di localizzazione tempo-frequenza e la possibilità di fornire una gerarchia di immagini a diversa risoluzione.

1 La trasformata coseno discreta (DCT)

Visualizziamo in Matlab i vettori base della DCT monodimensionale di lunghezza $N = 4, 8, 16$ usando la funzione `dctmtx`, che fornisce in uscita proprio la matrice C :

```
C = dctmtx(N);  
figure;  
for i=1:N,  
    subplot(N/4,4,i), bar(C(i,:),0.5); axis([0 N+1 -0.5 0.5]);  
end;
```

Si noti come il segnale vari sempre più rapidamente man mano che ci si sposta lungo le righe, aumenta così la frequenza dei segnali che costituiscono la base. Supponiamo adesso di voler visualizzare le matrici base 8×8 come immagini:

```
C = dctmtx(8);  
figure;  
for k=1:8,  
    for l=1:8,  
        A = C(k,:)'*C(l,:);  
        subplot(8,8,sub2ind(size(A),k,l)); imshow(A,[]); hold on;  
        pause;  
    end;  
end;
```

Anche le matrici base variano sempre più rapidamente man mano che ci si sposta verso frequenze più alte lungo la direzione orizzontale, verticale e obliqua.

La trasformata DCT bidimensionale (diretta e inversa) può essere calcolata semplicemente in matlab usando i comandi `dct2` e `idct2`. Provate allora a calcolare la trasformata coseno discreta delle immagini

fiammiferi1.jpg e fiammiferi2.jpg e interpretate il risultato ottenuto in base alle variazioni spaziali contenute nell'immagine. Verificate, inoltre, la capacità di compattazione dell'energia nelle componenti in alto a sinistra della DCT. Ricordate inoltre di realizzare enhancement prima della visualizzazione, così come avete fatto per la trasformata di Fourier. Potete anche visualizzare la trasformata in falsi colori (inserendo anche la colorbar), dato che l'occhio umano è in grado di discriminare più facilmente le variazioni di colore piuttosto che dei livelli di grigio. O ancora potete usare il comando `mesh`.

Realizzate poi una nuova coppia di funzioni che operi *sui blocchi* di un'immagine. Il prototipo della funzione deve essere `function Y = blkdct(X,K,L)`, dove K e L sono le dimensioni dei blocchi. L'uscita Y ha le stesse dimensioni di X ed ha una struttura a blocchi, ognuno dei quali è la DCT del corrispondente blocco dell'ingresso. Provate anche a visualizzare i coefficienti di un blocco 16×16 e verificate ancora una volta la capacità di compattazione dell'energia dei coefficienti DCT.

1.1 Esercizi proposti

1. Watermarking nel dominio DCT. Si vuole realizzare il watermarking dell'immagine `cowper.bmp` nel dominio trasformato usando la seguente strategia:

- (a) si effettua la DCT bidimensionale diretta di tutta l'immagine;
- (b) si conservano i $K(K+1)/2$ coefficienti DCT alle basse frequenze e si modificano i successivi L coefficienti DCT alle medie frequenze secondo la regola: $c(i,j) = c(i,j) + \alpha |c(i,j)| w(i,j)$, dove w è il watermark ed è costituito da valori pseudo-casuali ottenuti con il comando `randn`;
- (c) si effettua la DCT bidimensionale inversa di tutta l'immagine.

Scrivete uno script dal nome `ex2.m` in cui realizzate i passi descritti e mostrate sia l'immagine originale che quella marcata nell'ipotesi in cui $K = 180$, $L = 16350$ e $\alpha = 0.2$. Mostrate a video l'immagine originale e quella marcata.

2. Filtraggio nel dominio DCT. Effettuate il filtraggio a blocchi dell'immagine `Libro.jpg` mediante la seguente strategia:

- (a) dividete l'immagine in blocchi 8×8 ;
- (b) di ogni blocco calcolate la varianza s , e trovate poi le due soglie s_1 e s_2 superate solo dal 25% e 5% rispettivamente delle varianze;
- (c) per ogni blocco effettuate la DCT: se $s \leq s_1$ conservate solo la componente continua, se $s_1 < s \leq s_2$ conservate solo i $(K*(K+1)/2)$ coefficienti a più bassa frequenza oltre alla DC, se $s > s_2$ conservate tutti i coefficienti, infine antitrasformate il blocco.

Realizzate un esperimento per K che va da 1 a 5, valutando l'MSE rispetto all'immagine originale.

2 La Trasformata Wavelet

La trasformata Wavelet può essere realizzata semplicemente ed efficacemente mediante un banco di filtri. In particolare la fase di analisi (decomposizione) prevede un filtraggio passa basso e passa alto seguiti da un'operazione di decimazione per 2, mentre la fase di sintesi (ricostruzione) elabora separatamente gli ingressi, attraverso un'espansione ed un filtraggio, per poi sommarli. Queste operazioni devono essere iterate a seconda del numero di livelli di decomposizione con cui si vuole realizzare la trasformata. A tale scopo usate le funzioni `fwt.m` e `iwt.m` che implementano la trasformata wavelet diretta e inversa monodimensionale con la base di Haar. L'estensione al caso bidimensionale risulta molto semplice, in quanto la trasformata wavelet è separabile, pertanto è possibile operare prima lungo le righe e poi lungo le colonne. Se indichiamo con lev il numero dei livelli di decomposizione, una possibile struttura del codice è:

```

W=double(imread('vaso.jpg'));
[M,N]=size(W);
for l=1:lev,
    for i=1:M                % filtraggio lungo le righe
        riga=W(i,1:N);      % riga da trasformare
        [wtriga]=fwt(riga,1); % WT monodimensionale con lev=1
        W(i,1:N)=wtriga;    % posizionamento
    end
    for j=1:N,                % filtraggio lungo le colonne
        riga = W(1:M,j);    % colonna da trasformare
        [wtriga] = fwt(riga',1);
        W(1:M,j) = wtriga'; % posizionamento
    end;
    M=M/2;                    % dimezza le dimensioni
    N=N/2;                    % per il passo successivo
end;

```

Si noti come ad ogni livello di decomposizione l'immagine che viene elaborata è la versione passa-basso di quella originale che ha dimensioni 4 volte più piccole dell'immagine di partenza (figura 2). Fate molta attenzione alla visualizzazione della trasformata wavelet, in quanto le varie sottobande hanno dinamiche anche molto diverse. In particolare le bande con i dettagli assumono generalmente valori prossimi allo zero eccetto in corrispondenza dei bordi (orizzontali, verticali o obliqui) dell'immagine (figura 1).

2.1 Esercizi proposti

1. Create due funzioni Matlab function $[w] = \text{fwt2d}(x, \text{lev})$ (Forward Wavelet Transform) e function $[x] = \text{iwt2d}(w, \text{lev})$ (Inverse Wavelet Transform) che realizzano la trasformata wavelet bidimensionale con lev livelli di decomposizione. Verificate la proprietà di perfetta ricostruzione, sia visualizzando l'immagine antitrasformata sia calcolando il rapporto segnale rumore tra immagine originale e ricostruita.
2. *Visualizzazione wavelet 2D.* Scrivete una funzione function $\text{wtview}(w, \text{lev})$ che visualizza la trasformata wavelet di un'immagine con numero di livelli di decomposizione assegnato. Per una corretta visualizzazione ogni sottobanda deve essere sottoposta ad un FSHS indipendentemente dalle altre.
3. *Watermarking.* Un modo possibile per inserire una firma digitale in un'immagine è mediante l'uso della trasformata wavelet. Si può pensare di aggiungere il watermark (che in questo caso si genera come una sequenza casuale) solo ai coefficienti wavelet significativi (che superano cioè una determinata soglia) delle bande dettaglio. In questo modo vengono alterati solo quei coefficienti che sono legati alle discontinuità dell'immagine cui l'occhio umano risulta meno sensibile (*visual masking*). Scrivete quindi il codice matlab per realizzare i seguenti passi:

- (a) trasformata wavelet diretta dell'immagine su 3 livelli di decomposizione;
- (b) modifica dei coefficienti wavelet $w(i, j)$ superiori (in modulo) ad una soglia T_1 relativi solo alle bande dettaglio usando la seguente regola (*casting del watermark*):

$$w'(i, j) = w(i, j) + \alpha |w(i, j)| \text{wm}(i, j)$$

Il watermark $\text{wm}(i, j)$ può essere generato con il comando `rand`, specificando anche il seme.

- (c) trasformata wavelet inversa dell'immagine.

Applicate l'algoritmo all'immagine `cowper.bmp`, usando $\alpha = 0.2$ e $T_1 = 40$, visualizzate l'immagine originale e quella marcata.

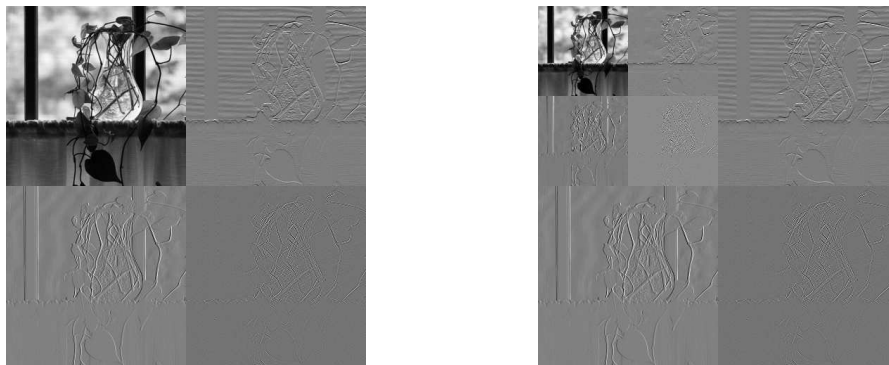


Figura 1: Trasformata wavelet a uno e due livelli di decomposizione

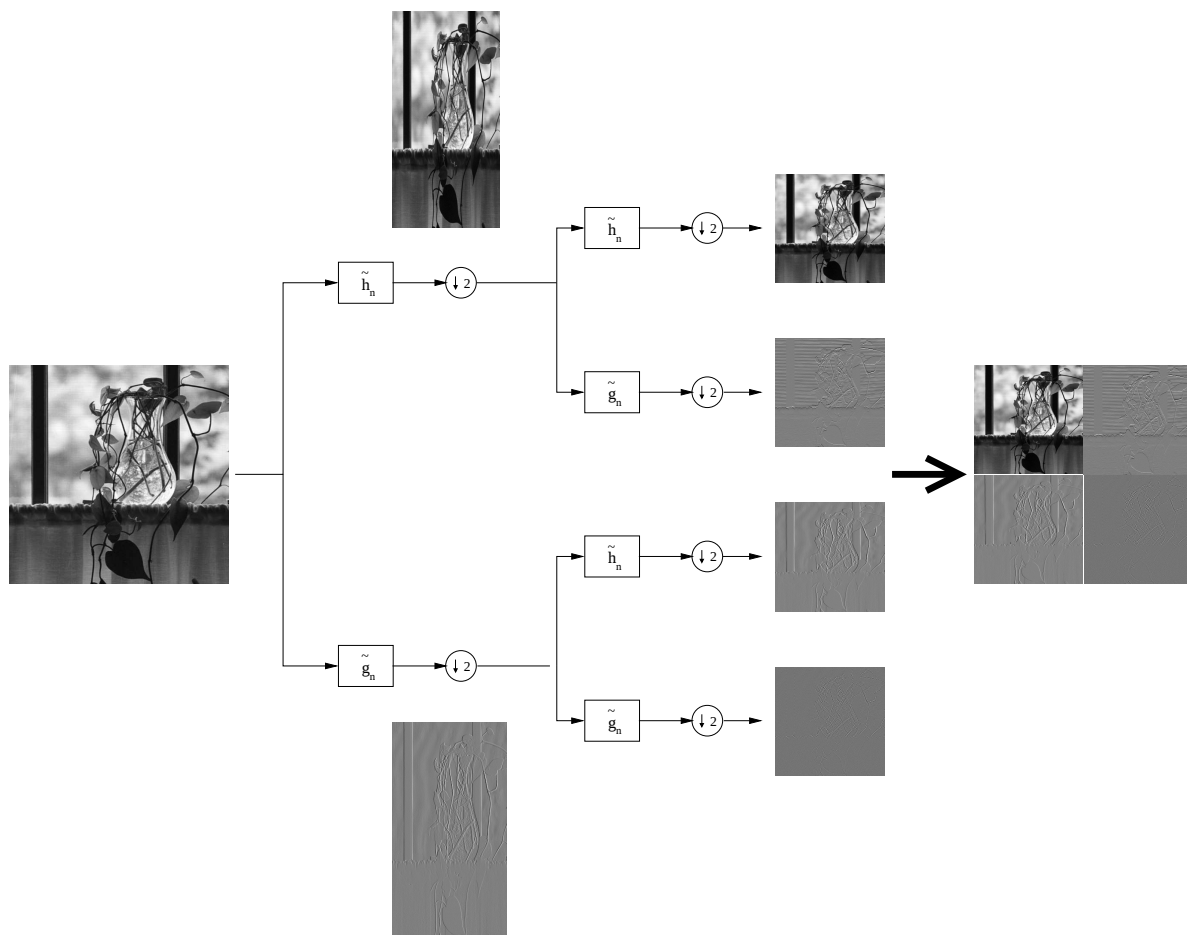


Figura 2: Implementazione wavelet 2D mediante banco di filtri