Elaborazione di Segnali Multimediali
a.a. 2017/2018

La compressione

L.Verdoliva

L'argomento di questa esercitazione è la compressione di immagini. Simuleremo alcune possibili strategie di codifica nel dominio DCT. Infine proveremo a realizzare (trascurando la codifica entropica) i passi che caratterizzano il codec nello standard JPEG.

1 Codifica tramite DCT

In questa sezione proporremo alcuni approcci per l'elaborazione dei coefficienti trasformati mediante DCT ai fini della compressione. A rigore per progettare un adeguato schema di codifica è necessario quantizzare i coefficienti e aggiungere ai dati codificati una eventuale *side information* (es. posizione dei coefficienti che si vogliono trascurare). Per semplicità le strategie che consideriamo di seguito non prevedono queste operazioni, e il nostro obiettivo sarà solo quello di valutare l'importanza che hanno i diversi coefficienti nella ricostruzione dell'immagine.

1.1 Taglio dei coefficienti sotto soglia

Realizziamo una strategia di codifica che prevede di annullare i coefficienti inferiori ad una determinata soglia. Scriviamo dunque il codice che realizza le seguenti operazioni:

1. DCT-2D dell'immagine originale;
2. azzeramento dei coefficienti DCT in modulo minori di γ;
3. calcolo della percentuale dei coefficienti posti a zero;
4. DCT-2D inversa dei dati elaborati;
5. visualizzazione dell'immagine originale, di quella ricostruita e della maschera.

```
soglia = 20;
y=dct2(x);
y(abs(y)<soglia)=0;
xr=idct2(y);
% Calcolo della percentuale di coefficienti posti a 0
Mask = (abs(y)<soglia);
rho = sum(Mask(:))/prod(size(x)) *100;
figure(1); imshow(x);
title('Immagine originale');
figure(2); imshow(xr);
title(sprintf('Eliminato il %1.2f %% dei coefficienti', rho));
figure(3); imshow(Mask,[]);
```

Fate variare il valore della soglia γ per alcune delle immagini di prova e visualizzate l'immagine ricostruita e calcolate il PSNR al variare della soglia mostrando il grafico relativo.

Scrivete poi una funzione con il prototipo `Y=compress(X,gamma,K,L)` che effettua il taglio dei coefficienti blocco per blocco invece che su tutta l'immagine. Fissato il parametro γ, confrontate il risultato della codifica per blocchi con quello che opera su tutta l'immagine.

1.2 Zonal coding

Consideriamo adesso una diversa strategia per la codifica in cui si conservano solo i coefficienti a frequenza più bassa. In particolare, la maschera va costruita in modo tale che solo i coefficienti che sono presenti in alto a sinistra nell'immagine siano conservati. In particolare se si trattengono solo i $\frac{K_1(K_1+1)}{2}$ coefficienti a frequenza più bassa di un'immagine $K \times K$, la maschera è la seguente:

```
mask = [fliplr(triu(ones(K1))),zeros(K1,K-K1);zeros(K-K1,K1),zeros(K-K1,K-K1)];
```

Usate l'help in linea di matlab per comprendere come opera il comando `triu`. Quindi scrivete il codice, in modo analogo all'esempio precedente, con la maschera così definita. Visualizzate poi l'immagine ricostruita e confrontatela con quella ottenuta con l'approccio visto nel paragrafo 3.1. Che differenze notate?

2 Lo standard JPEG

Lo standard JPEG è il più diffuso standard per la compressione di immagini. Se x è l'immagine da codificare in cui ogni pixel è rappresentato da b bit, prevede di effettuare i seguenti passi:

1. sottrarre il valore 2^{b-1} dai dati originali (questo significa in sostanza rendere i dati a media nulla). Se i dati sono codificati su 8 bit bisogna sottrarre dunque 128;

2. effettuare la DCT su blocchi 8×8 dell'immagine. Nel caso in cui la dimensione lungo le righe (colonne) dell'immagine non sia un multiplo di 8, allora l'ultima riga (colonna) viene replicata fin quando non diventa multiplo di 8. Ovviamente in fase di decodifica queste righe (colonne) addizionali vengono rimosse;

3. quantizzare uniformemente ogni blocco con passi di quantizzazione specificati dalla matrice Q; tale matrice definisce per ogni coefficiente appartenente al blocco il passo di quantizzazione Δ da utilizzare con un quantizzatore uniforme che prevede anche lo zero come livello di restituzione:

```
Q = [ 16  11  10  16   24   40   51   61
      12  12  14  19   26   58   60   55
      14  13  16  24   40   57   69   56
      14  17  22  29   51   87   80   62
      18  22  37  56   68  109  103   77
      24  35  55  64   81  104  113   92
      49  64  78  87  103  121  120  101
      72  92  95  98  112  100  103   99  ];

y = Q.*(round(x./Q));
```

In realtà, i valori in uscita al quantizzatore dovrebbero essere gli indici relativi a ogni valore (round(x/Δ)), tuttavia trascuriamo in questa analisi il processo di codifica/decodifica.

Teniamo presente che è anche necessario introdurre un *fattore di qualità*, che consenta all'utente di scegliere il livello di compressione che si vuole raggiungere. A tale scopo la matrice Q viene opportunamente scalata, in base alla qualità che si vuole ottenere nel modo seguente:

$$Q_s = \frac{(S \cdot Q + 50)}{100}$$

dove

$$S = \begin{cases} 5000/L & \text{se} \quad 0 < L < 50 \\ 200 - 2L & \text{se} \quad 50 \leq L < 100 \\ 1 & \text{se} \quad L = 100 \end{cases}$$

L definisce il livello di qualità ed è compreso tra 1 (elevato fattore di compressione, pessima qualità) a 100 (basso fattore di compressione, elevata qualità).

Scrivete allora una funzione che produce la stessa immagine che otterreste con codifica e decodifica JPEG dell'ingresso per un assegnato fattore di qualità e che abbia il seguente prototipo: `function y=simJPEG(x,L)` (tenete presente che non ci stiamo preoccupando di codificare i valori quantizzati in binario, stiamo cioè trascurando la codifica entropica). Fate quindi degli esperimenti al variare delle immagini considerate (non usate immagini già compresse jpeg, ma quelle in formato grezzo) e valutate l'andamento di PSNR o MSE al variare di L. Potete verificare la correttezza del codice usando il comando `fwrite` di Matlab che consente di scrivere un'immagine in formato JPEG con un assegnato livello di qualità, usando il seguente codice:

```
fp = fopen('peppers.y','rb');
x  = fread(fp,[512 512],'uint8');
x  = uint8(x');
figure; imshow(x);
imwrite(x,'imdec.jpg','JPEG','Quality',10);
xq = imread('imdec.jpg');
figure; imshow(xq);
```

Per conoscere il tasso ottenuto con il fattore di qualità scelto potete guardare la dimensione del file compresso 'imdec.jpg' e confrontarla con l'originale oppure scrivere il seguente codice che vi fornisce in bpp il tasso di codifica espresso in bit/pixel dell'immagine compressa:

```
fid = fopen('imdec.jpg','r');
[TMP, bytes] = fread(fid,'int8');
bpp = bytes*8/(M*N)
```

2.1 Esercizi proposti

1. Si vuole effettuare una pseudo-codifica a basso bit-rate dell'immagine lena.y (512×512, uint8) mediante DCT. A tale scopo, dopo aver effettuato la DCT per blocchi dell'immagine, si conservano solo alcuni coefficienti, annullando tutti gli altri, e si ricostruisce l'immagine pseudo-codificata. Confrontate sia visivamente che in termini di SNR le seguenti strategie:

   (a) blocchi 8×8 e solo il coefficiente DC conservato;

   (b) blocchi 8×8 e solo i 3 coefficienti a frequenza più bassa conservati (DC + 2 AC);

   (c) blocchi 16×16 e solo i 15 coefficienti a frequenza più bassa conservati (DC + 14 AC);

   (d) blocchi 32×32 e solo i 10 coefficienti con modulo più elevato conservati.

2. Si vuole comprimere l'immagine peppers.y (512×512, uint8) usando la DCT su blocchi di 64 pixel, ma aventi diversi rapporti d'aspetto. A tal fine, scrivete una funzione `function y = codec(x,K,L)` in cui calcolate la DCT di blocchi $K \times L$ dell'immagine `x`, conservate solo i 4 coefficienti più grandi (in modulo) di ogni blocco, li sottoponete a quantizzazione uniforme con passo 16, e infine ricostruite l'immagine `y` a partire dai blocchi quantizzati. Nello script ex2.m realizzate un esperimento in cui, per ognuna delle seguenti dimensioni del blocco: 8×8, 4×16, 2×32, 1×64, effettuate la compressione di peppers.y e valutate l'MSE rispetto all'immagine originale. Commentate sinteticamente i risultati ottenuti.

   Suggerimento: in matlab è presente la funzione `sort` che realizza l'ordinamento.

3. Codifica con trasformata Wavelet. Un modo molto semplice per effettuare compressione è quello di trascurare i coefficienti con *valore assoluto* piccolo (al di sotto di una prefissata soglia). Scriviamo dunque il codice che realizza le seguenti operazioni:

   (a) Wavelet-2D dell'immagine originale;

   (b) azzeramento dei coefficienti (in valore assoluto) inferiori ad una certa soglia relativi alle sole bande dettaglio;

   (c) calcolo della percentuale dei coefficienti posti a zero;

   (d) Wavelet-2D inversa dei dati elaborati;

   Fate variare il valore della soglia γ per alcune delle immagini di prova e visualizzate l'immagine ricostruita e calcolate il PSNR al variare della soglia mostrando il grafico relativo.