Homework 1

Una semplice tecnica di codifica d'immagini basata sulla DCT

Giacomo Calabria - 2007964

May 2, 2023

1 Introduction

L'obiettivo di questo Homework è quello di produrre un programma che effettui una semplice tecnica di codifica basata sulla DCT.

Il codice fornito è stato scritto nell'ambiente di programmazione MATLAB® in quanto esso dispone della libreria *Image Processing Toolbox* che offre molti strumenti per manipolare ed elaborare in modo semplice le immagini, tra cui comprese la funzione DCT. Inoltre, la possibilità di visualizzare immagini e manipolarle in modo facile e intuitivo è estremamente utile per analizzare e valutare la qualità della codifica effettuata.

Per eseguire il codice, è necessario salvare l'immagine da elaborare nella stessa cartella del file MATLAB, quindi aprire il file e premere il tasto "Run" o digitare il comando `"run Project1.m"` nella finestra della console di MATLAB.

Il codice utilizza alcune funzioni matematiche standard di MATLAB per calcolare il MSE e il PSNR e tracciare le curve del PSNR in funzione di R. E richiede la libreria *Image Processing Toolbox* per gestire l'elaborazione di immagini e in particolare la funzione DCT.

Il codice richiede l'impostazione di alcuni parametri di input:

- Il path/nome del file
- Dimensione dei blocchi N
- La percentuale R di coefficienti DCT da mettere a zero

Questi parametri possono essere modificati all'interno del file MATLAB®.

2 Metodo di codifica

1. Caricamento dell' immagine
2. Cambio spazio dei colori da RGB a YCbCr
3. Per ognuna delle componenti Y,Cb e Cr si è proceduto a fare la DCT bidimensionale con blocchi di dimensione N sulla singola componente, utilizzando la funzione

   ```
   blockproc(y, [N N], dctfun);
   ```

   dove `dctfun` è la funzione per la DCT sul blocco.

   (a) Successivamente è stata calcolata la frazione R% dei coefficienti DCT dell'**intera componenete** da mettere a zero

   ```
   perc_y = prctile(abs(y_dct(:)), R);
   ```

   (b) E' stata messa a zero la frazione calcolata al punto precedente

   ```
   y_dct(abs(y_dct) < perc_y) = 0;
   ```

4. E' stata fatta la DCT inversa bidimensionale sui blocchi di lunghezza sempre N sulla singola componente, sempre utilizzando la funzione

   ```
   blockproc(y, [N N], dctinvfun);
   ```

   dove `dctinvfun` è la funzione inversa della DCT sul blocco.

5. Calcolo dell'*MSE* tra la componente originale e la componente "compressa"

   ```
   mse_y = immse(double(y(:)),double(y_compressed(:)));
   ```

Successivamente è stato calcolato l'MSE pesato, utilizzando

$$MSE_P = \frac{3}{4}MSE_Y + \frac{1}{8}MSE_{Cb} + \frac{1}{8}MSE_{Cr} \tag{1}$$

da cui abbiamo calcolato poi il PSNR pesato come

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE_P}\right) \tag{2}$$

Abbiamo proceduto poi a plottare in un grafico i diversi valori di PSNR ottenuti facendo variare opportunamente R in un intervallo da 10 a 100.

3 Risultati: grafici del *PSNR*

Riportiamo adesso i risultati ottenuti con la tecnica di codifica d'immagine basata su DCT implementata. In Figura 1 è stata riportata l'immagine originale scelta per testare la codifica.

Figure 1: Immagine originale

Dapprima abbiamo tracciato i grafici del PSNR in funzione di R per $N = 8, 16, 64$, facendo variare il parametro R da 10 a 100 in modo lineare a passi di 0.1 in modo da poter apprezzare al meglio l'andamento del grafico. Riportiamo quindi nelle Figure 2,3 e 4 i grafici.

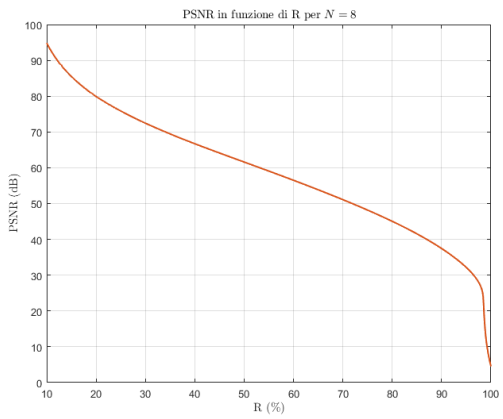

Figure 2: $N = 8$

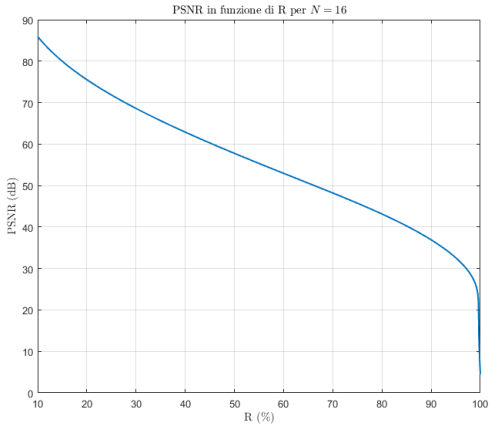

Figure 3: $N = 16$

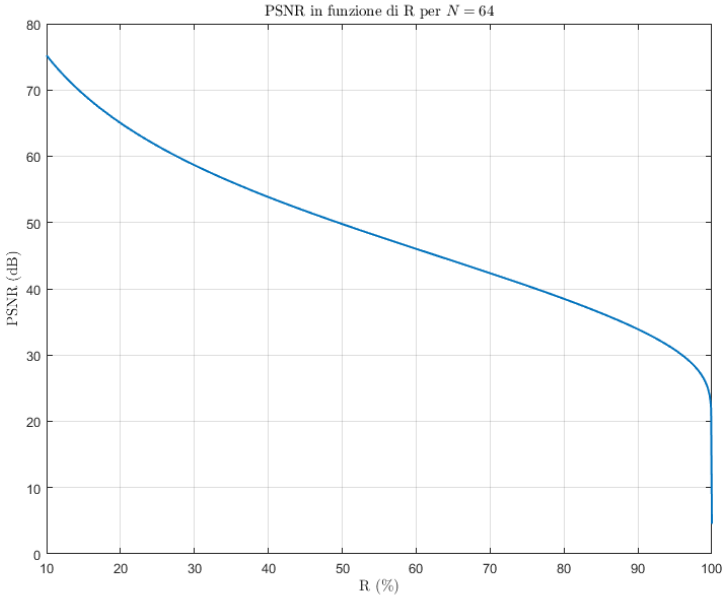

Figure 4: $N = 64$

Si è voluto aumentare la "risoluzione" di R, rispetto alla consegna, per permettere di vedere il reale andamento del PSNR in funzione di R. Viene riportato in Figura 5 l'andamento del grafico nel caso in cui il parametro R viene fatto variare a passi di 10.

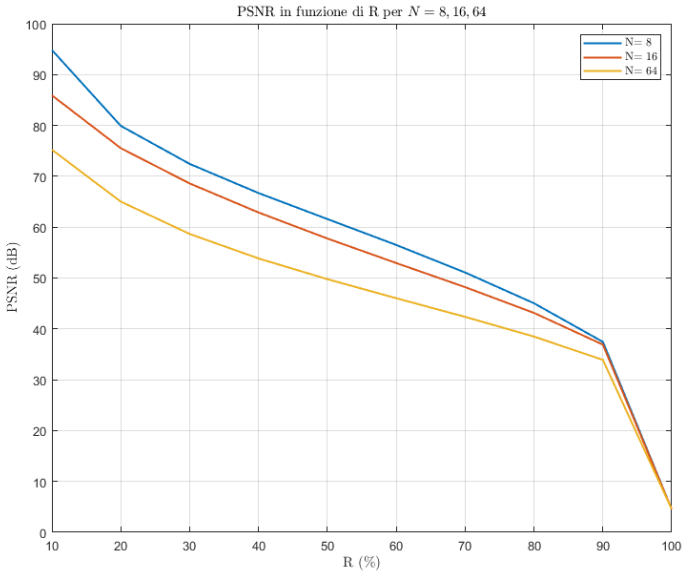

Figure 5: Andamento del PSNR per $N = 8, 16, 64$ con passo di R a 10.

Si osserva che all'aumentare della dimensione dei blocchi la compressione lavora meglio. Questo è dovuto al fatto che con blocchi più grandi è possibile raggruppare più informazioni spaziali in un unico blocco, sono meno sensibili alle variazioni locali dell'immagine, aumentando l'efficacia della codifica DCT. Si nota che per tutti i valori di N, il PSNR si riduce rapidamente per valori di R superiori al 90%

Si vuole sottolineare infine che il tempo computazionale aumenta sensibilmente diminuendo la dimensione del blocco.

4 Risultati: immagini codificate

Infine, riportiamo alcune immagini ricavate dalla nostra codifica, avendo impostato il fattore di compressione $R = 98\%$ ad un valore abbastanza elevati per poter apprezzare come l'immagine si degrada a seguito della compressione.

Si può notare come la qualità dell'immagine compressa peggiora gradualmente all'aumentare della percentuale di coefficienti DCT posti a zero, con una perdita di dettagli e una comparsa di artefatti visibili in alcune zone dell'immagine. Tuttavia, anche quando viene impostata una percentuale elevata ($R > 95\%$), l'immagine compressa rimane abbastanza riconoscibile. Una compressione troppo aggressiva può causare una perdita di informazioni.

Figure 6: $N = 8, R = 98\%, PSNR = 26.5859$

Figure 7: $N = 16, R = 98\%, PSNR = 28.6002$

Figure 8: $N = 64, R = 98\%, PSNR = 27.8918$