Relazione progetto sulla compressione di immagini tramite DCT

-

Metodi del calcolo scientifico 2024/25

Ferrario, Christian
c.ferrario30@campus.unimib.it

Lanza, Andrea
a.lanza13@campus.unimib.it

June 5, 2025

Contents

1 Introduzione

1.1 Obiettivi del progetto

Questo progetto si propone di analizzare e mettere in pratica le tecniche di compressione delle immagini tramite la Trasformata Discreta del Coseno bidimensionale (DCT), replicando un meccanismo simile a quello del formato JPEG (escludendo la quantizzazione). Gli obiettivi principali sono:

- **Implementare la DCT da zero**, secondo la definizione vista a lezione, in un ambiente open-source a scelta, al fine di comprendere a fondo la struttura matematica della trasformazione e le sue implicazioni computazionali.
- **Progettare un sistema di compressione** basato sulla DCT2, suddividendo l'immagine in blocchi quadrati di dimensione $F \times F$, applicando la trasformata e filtrando i coefficienti in frequenza in base a una soglia d.
- **Studiare l'impatto visivo della compressione** al variare dei parametri F e d, valutando la qualità delle immagini risultanti e analizzando il compromesso tra perdita di informazione e riduzione dei dati.
- **Confrontare le prestazioni** (in termini di tempo di esecuzione) tra l'implementazione manuale e quella fornita dalla libreria dell'ambiente scelto (presumibilmente ottimizzata tramite FFT). Questo confronto ha lo scopo di evidenziare le differenze di complessità algoritmica tra l'approccio naïve ($O(N^3)$) e quello ottimizzato ($O(N^2 \log N)$).

1.2 Descrizione del progetto

Nella *prima parte* del progetto, si richiede l'implementazione manuale della **DCT2**, come descritta a lezione, e la successiva comparazione in termini di tempo di esecuzione con l'implementazione ottimizzata fornita da una libreria open source, in questo caso la versione della libreria **SciPy**, in particolare del modulo `scipy.fftpack`. A tal fine, è stato utilizzato il linguaggio **Python** e la libreria **NumPy**, che mette a disposizione strumenti efficienti per il calcolo numerico e la manipolazione di array multidimensionali.

La *seconda parte* del progetto prevede la realizzazione di un'applicazione software con interfaccia grafica, che consenta all'utente di selezionare un'immagine e di eseguire una compressione tramite DCT2 su blocchi di dimensione decisa

dall'utente, e con la possibilità di impostare una soglia di taglio delle alte frequenze. L'immagine risultante viene poi ricostruita applicando l'inversa della DCT2 sui blocchi modificati, e visualizzata a fianco dell'immagine originale per un confronto qualitativo.

Tutte le componenti del progetto sono state realizzate utilizzando software open source, in linea con le specifiche richieste, ed i risultati sperimentali sono stati discussi per evidenziare vantaggi, limiti e possibili miglioramenti futuri.

1.3 Contesto teorico

La *Discrete Cosine Transform* bidimensionale (DCT2) è uno strumento fondamentale nell'elaborazione e compressione delle immagini. Essa consente di rappresentare un'immagine in termini di frequenze spaziali, permettendo la separazione tra componenti a bassa e alta frequenza. Questo principio è alla base di formati di compressione come JPEG, dove le frequenze meno rilevanti possono essere eliminate o approssimate senza un degrado percettibile dell'immagine.

Di seguito una spiegazione dettagliata della teoria dietro al metodo di compressione delle immagini tramite Trasformata Discreta del Coseno, e di come i parametri cambiano il comportamento della compressione.

2 Trasformata Discreta del Coseno

La **Discrete Cosine Transform** (DCT) è una trasformata lineare che rappresenta un segnale come combinazione di coseni a diverse frequenze. È simile alla trasformata di Fourier, ma utilizza solo funzioni coseno e fornisce coefficienti reali. La DCT è particolarmente efficace per la compressione di segnali con contenuto prevalentemente a bassa frequenza, come le immagini.

2.1 DCT monodimensionale

Data una sequenza $f = (f_0, f_1, \ldots, f_{N-1})$, la DCT di tipo II è definita come:

$$C_k = \alpha_k \sum_{j=0}^{N-1} f_j \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)k\right), \quad \text{per } k = 0, \ldots, N-1$$

dove:

$$\alpha_k = \begin{cases} \sqrt{\frac{1}{N}} & \text{se } k = 0 \\ \sqrt{\frac{2}{N}} & \text{se } k > 0 \end{cases}$$

Questa trasformata risulta ortonormale grazie alla scelta di questi coefficienti delle basi: la base della DCT è formata da vettori le cui componenti sono valori di coseni valutati a punti equispaziati. La trasformazione proietta il segnale sulla base dei coseni, ottenendo coefficienti X_k che rappresentano l'importanza della k-esima frequenza.

2.2 DCT bidimensionale (DCT2)

Nel caso bidimensionale, tipico delle immagini, la DCT2 si applica a una matrice $A \in \mathbb{R}^{N\times N}$. Essa si può definire come:

$$C_{k,\ell} = \alpha_k \alpha_\ell \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{i,j} \cos\left(\frac{\pi}{N}\left(i + \frac{1}{2}\right)k\right) \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)\ell\right)$$

per $k, \ell = 0, \ldots, N-1$, dove α_k è definito come sopra.

Alternativamente, si può esprimere in forma matriciale come:

$$C = DAD^T$$

dove D è la matrice ortogonale contenente i coefficienti dei coseni (base della DCT monodimensionale). In pratica, la DCT2 si può calcolare applicando la DCT per righe e poi per colonne (o viceversa, grazie alla simmetria dell'operazione).

Compressione JPEG

La compressione JPEG sfrutta la trasformata DCT per ridurre l'informazione contenuta nelle frequenze meno percepibili all'occhio umano. In particolare, la maggior parte dell'informazione visiva si concentra nelle frequenze più basse (variazioni lente di colore o intensità), mentre i dettagli fini, come bordi netti o pattern regolari, si trovano nelle frequenze alte.

Quando si applica una soglia per tagliare le componenti ad alta frequenza, l'effetto più visibile è la perdita di dettaglio nelle aree dell'immagine dove il colore o la luminosità cambiano bruscamente, come i contorni o i pattern ad alto contrasto. Questo tipo di compressione introduce artefatti visivi tipici del JPEG, come la sfocatura dei bordi o l'effetto "blocco".

L'efficacia del metodo risiede nel fatto che il sistema visivo umano è meno sensibile alle alte frequenze, permettendo così una riduzione significativa dei dati senza una perdita apparente di qualità in molte situazioni.

3 Implementazione della DCT e DCT2 personalizzata

Per la compressione jpeg delle immagini abbiamo realizzato una **libreria personalizzata per il calcolo della DCT** (monodimensionale e bidimensionale), basata sulla definizione teorica della trasformata discreta del coseno. L'implementazione non utilizza algoritmi ottimizzati (come quelli basati su FFT), ma si fonda sull'utilizzo esplicito di una matrice di base D contenente le funzioni coseno ed i suoi coefficienti.

3.1 Costruzione della base (matrice D)

Per un vettore di lunghezza N, la trasformata discreta del coseno può essere espressa come un prodotto matrice-vettore, dove la matrice $D \in \mathbb{R}^{N \times N}$ è definita da:

$$D_{k,j} = \alpha(k) \cdot \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)k\right) \quad \text{per } k, j = 0, \ldots, N-1$$

con:

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{se } k = 0 \\ \sqrt{\frac{2}{N}} & \text{se } k > 0 \end{cases}$$

Questa costruzione rende D una matrice ortogonale, per cui l'inversa coincide con la trasposta: $D^{-1} = D^{\top}$.

```
def dct_base(n):
    D = np.zeros((n, n))
    alpha = alphas(n)

    for i in range(n):
        for j in range(n):
            D[i, j] = alpha[i] * cos((i * pi * (2 * j + 1) / (2 * n)))
    return D

def alphas(n):
    alpha = np.zeros(n)
    alpha[0] = 1 / sqrt(n)
    for i in range(1, n):
        alpha[i] = sqrt(2 / n)
    return alpha
```

3.2 Implementazione della DCT monodimensionale

Utilizzando la matrice D, la trasformata discreta del coseno (DCT) e la sua inversa (IDCT) si implementano nel modo seguente:

```
def dct(f, D):
    return D @ f

def idct(c, D):
    return D.T @ c
```

Dove `f` è il vettore delle sequenze in ingresso e `c` il vettore dei coefficienti.

3.3 Implementazione della DCT bidimensionale (DCT2)

Nel caso bidimensionale, la DCT2 si ottiene applicando la trasformata una volta per riga e una volta per colonna. L'implementazione sfrutta nuovamente la matrice D come segue:

```
def dct2(f, D):
    return D @ f @ D.T

def idct2(c, D):
    return D.T @ c @ D
```

Dove `f` rappresenta un blocco di immagine $F \times F$ (o l'immagine stessa), e `c` è la matrice dei coefficienti trasformati.

3.4 Eliminazione delle alte frequenze

Per simulare un meccanismo di compressione simile a quello impiegato nello standard JPEG, è stato implementato un filtro che elimina le alte frequenze nella matrice dei coefficienti ottenuta dalla trasformata DCT2.

L'eliminazione viene effettuata in ciascun blocco $F \times F$ dell'immagine trasformata, applicando una maschera diagonale binaria. Questa maschera ha valore 1 nei coefficienti per cui $i + j < D$ (dove i e j sono gli indici di riga e colonna), e 0 altrove. In questo modo, vengono mantenuti solo i coefficienti a bassa frequenza, che rappresentano la parte più significativa dell'informazione visiva, mentre vengono annullati quelli a più alta frequenza, che contribuiscono in misura minore alla qualità percepita.

La funzione `cut_frequencies` applica tale maschera a ciascun blocco dell'immagine, riducendo efficacemente la quantità di dati senza alterare eccessivamente il contenuto visivo rilevante.

4 Struttura della progetto

L'implementazione del progetto è stata organizzata in modo modulare, seguendo una separazione tra la **libreria** sviluppata e i **programmi eseguibili** che ne fa uso. Di seguito si descrivono le principali componenti.

```
jpeg-converter
├── cli
│   ├── __init__.py
│   └── cli.py
├── converter
│   ├── __init__.py
│   ├── backend.py
│   └── dct.py
├── gui
│   ├── __init__.py
│   ├── gui.py
│   └── resources
├── images
│   ├── 160x160.bmp
│   └── ...
├── main.py
├── pyproject.toml
└── tests
    ├── __init__.py
    ├── benchmark.py
    ├── parameters.py
    └── test.py
```

4.1 Panoramica generale

La root della repository contiene tre directory principali:

- `converter/`: libreria contenente l'implementazione della logica della Discrete Cosine Transform e le funzioni che permettono di applicarla ad un immagine.
- `cli/`: file contenenti la logica per avviare la compressione da linea di comando
- `gui/`: file contenenti il necessario per avviare l'interfaccia grafica
- `tests/`: file contenenti test e benchmark, spiegati di seguito.

Completano il progetto i file `pyproject.toml` e `requirements.txt`, che definiscono le dipendenze del progetto e script di esecuuzione.

4.2 Verifica della correttezza

Per verificare il corretto funzionamento dell'implementazione della DCT (Discrete Cosine Transform), sono stati eseguiti due test.

È stato applicato l'algoritmo della DCT monodimensionale a un vettore di 8 valori:

```
[231, 32, 233, 161, 24, 71, 140, 245]
```

Il risultato è stato confrontato con una trasformata attesa nota:

```
[4.01e+02, 6.60e+00, 1.09e+02, -1.12e+02, 6.54e+01, 1.21e+02,
                    1.16e+02, 2.88e+01]
```

Analogamente, è stato considerato un blocco 8×8 di valori in scala di grigi. L'algoritmo DCT2 è stato applicato al blocco e i risultati ottenuti sono stati confrontati con i coefficienti attesi forniti nel testo del progetto.

Criterio di verifica

Il confronto tra i risultati ottenuti e quelli attesi è stato effettuato in termini di errore relativo massimo, confrontando elemento per elemento le due matrici. Il test è considerato superato se tutti i valori presentano un errore relativo inferiore a una soglia prefissata $\varepsilon = 10^{-2}$.

4.3 Considerazioni

La struttura modulare del progetto garantisce una **buona separazione delle responsabilità**, facilita l'estensione e favorisce la riusabilità della libreria in altri contesti. Inoltre il convertitore può essere usato sia da linea di comando in modo rapido e veloce, oppure da interfaccia grafica, rendendolo versatile in diversi contesti.

5 Risultati sperimantali

5.1 Prima parte - confronto con FFT

Per confrontare l'efficienza computazionale tra la nostra implementazione della DCT2/IDCT2 e quella fornita dalla libreria `scipy`, è stato condotto un benchmark su matrici quadrate di dimensioni crescenti (da 32×32 fino a 8192×8192).
Per ciascuna dimensione:

- È stata generata una matrice casuale con valori reali in $[0, 1]$.
- È stata applicata la trasformata discreta del coseno bidimensionale (DCT2), seguita dalla cancellazione dei coefficienti ad alta frequenza (azzeramento del blocco in basso a destra della matrice trasformata).
- È stata poi applicata la trasformata inversa (IDCT2).
- Il tempo di esecuzione complessivo per DCT2 + IDCT2 è stato misurato con `time.perf_counter()`.

Il processo è stato ripetuto sia con l'implementazione basata su `scipy.fftpack` (con normalizzazione `"ortho"`), sia con l'implementazione sviluppata manualmente, basata su prodotto matriciale e precomputazione della base DCT.

I tempi ottenuti sono stati registrati separatamente per ciascun approccio, consentendo un confronto diretto delle prestazioni al variare della dimensione della matrice, e i risultati temporali sono stati riportati su un grafico in scala semilogaritmica (ascisse lineari, ordinate logaritmiche). Come atteso, l'implementazione manuale presenta una complessità computazionale dell'ordine di $O(N^2)$, mentre la versione fornita dalla libreria è più efficiente, con una complessità stimata in $O(N^2 \log N)$.

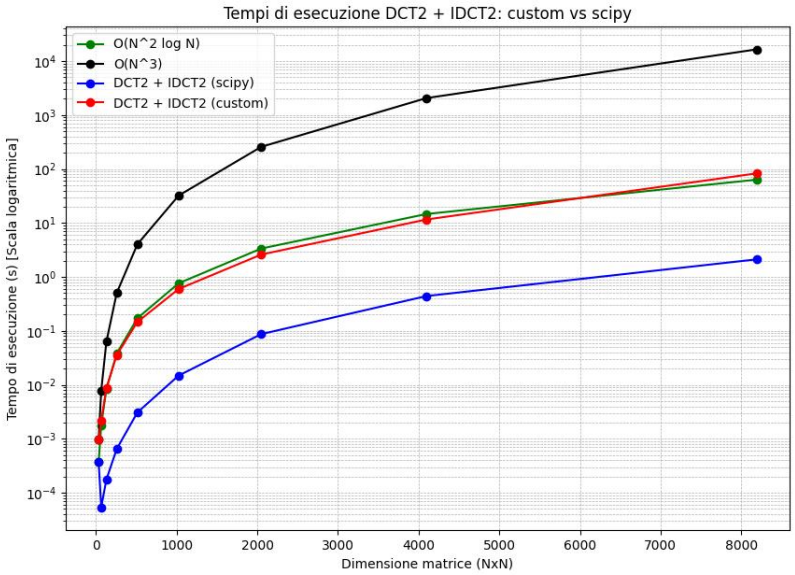


Figure 1: Confronto tra i tempi di esecuzione della DCT2 implementata manualmente e quella fornita dalla libreria `scipy`.

L'andamento dei tempi misurati è coerente con le complessità teoriche: per dimensioni moderate ($N \approx 1000$) le differenze tra i due approcci sono già significative, e tendono ad aumentare all'aumentare di N. Inoltre, l'implementazione ottimizzata mostra un comportamento più stabile e tempi sensibilmente inferiori in tutti i test effettuati, a conferma dell'efficienza dell'approccio basato su FFT per il calcolo della DCT2.

Questo confronto evidenzia l'importanza delle ottimizzazioni algoritmiche nei problemi numerici ad alta complessità, soprattutto quando si lavora con immagini di grandi dimensioni.

5.2 Seconda parte - impatto dei parametri F e D

Il progetto offre la possibilità di scegliere finestre di grandezza variabile per l'applicazione delle funzioni dct2/idct2, questo avviene tramite la scelta di un opportuno parametro F, rappresentante la dimensione dei blocchi a cui applicare la conversione; e l'implementazione della funzionalità di compressione (taglio dei coefficienti della funzione) tramite la scelta di opportuno parametro d rappresenta una soglia di taglio sulle frequenze: vengono mantenuti solo i coefficienti $c_{k,\ell}$ tali che $k + \ell < d$.

Scelta di F

- Un valore piccolo di F (ad esempio 4 o 8) porta ad una maggiore granularità. Questo comporta un numero elevato di blocchi, producendo un'immagine che risulta più dettagliata all'occhio umano, e permette di ridurre la dimensione del talio ai bordi delle immagini.
- Un valore grande di F (come 32 o 64) riduce il numero di trasformate da calcolare, ma include porzioni di immagine più eterogenee in ciascun blocco, riducendo l'efficacia della compressione e la qualità del risultato.

(a) f = 8

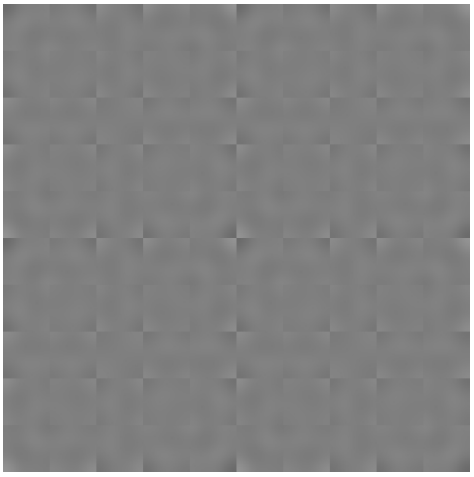
(b) f = 16

(c) f = 32

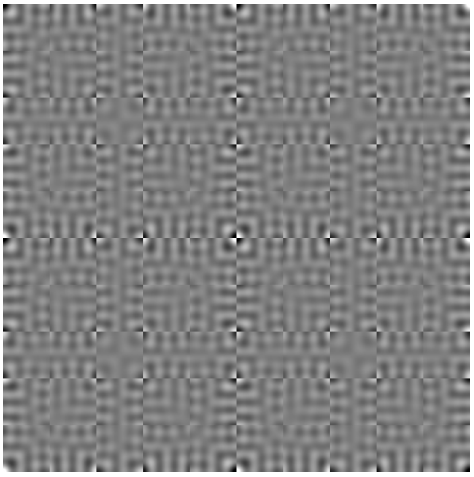
(d) f = 64

Scelta di d

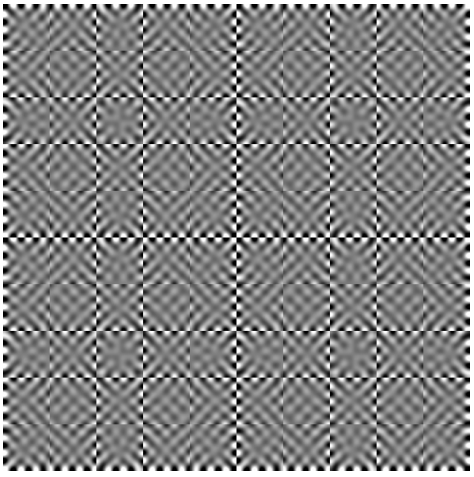
- Un valore basso di d implica un taglio più severo delle componenti in frequenza: vengono eliminate più frequenze, l'implementazione di un parametro d come quella proposta in questo progetto non permette di discriminare tra frequenze che hanno più peso nel determinare i dettagli di un'immagine; vengono quindi sempre eliminate le frequenze più alte, indiscriminatamete.
- Un valore alto di d preserva una maggiore quantità di informazione, vengono tagliati meno coefficienti, risultando in un'immagine più fedele all'originale.



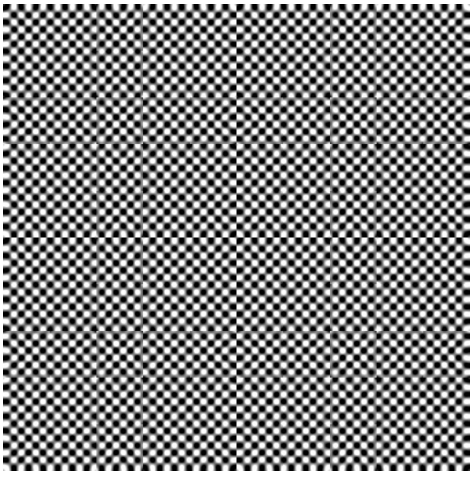
(a) d = 4



(b) d = 8



(c) d = 12



(d) d = 16