Dipartimento di Informatica
Corso di Laurea Magistrale in Informatica
Esame di Metodi del calcolo scientifico

Custom DCT2
&
Taglio frequenze con DCT

Di Capua Marco - 878295
Gherardi Alessandro - 817084
Pallini Vincenzo - 907303

Anno Accademico 2022 - 2023

Indice

1 Introduzione

Il presente progetto si focalizza sulla compressione di immagini in toni di grigio utilizzando la Trasformata Discreta del Coseno (DCT2) in un ambiente open source. L'obiettivo principale è studiare gli effetti di un algoritmo di compressione simile a JPEG, implementato senza l'uso di una matrice di quantizzazione, sulle immagini e confrontare i tempi di esecuzione tra l'implementazione della DCT2 sviluppata e quella fornita dalla libreria dell'ambiente utilizzato. Il confronto dei tempi è costruito sulla base di array quadrati di dimensioni $N \times N$ dove al variare di N verranno confrontati i tempi impiegati dai due algoritmi per eseguire la DCT2. Nella seconda parte del progetto è stato implementato un software che, tramite una semplice interfaccia, consente all'utente di selezionare un'immagine in formato.bmp in toni di grigio e, in base ai parametri F e d forniti in ingresso, esegue una compressione dell'immagine applicando la DCT2 e fornisce in output una stampa dell'immagine originale e una stampa dell'immagine compressa per poter confrontare i risultati.

2 Teoria e funzionamento DCT

La Discrete Cosine Transform (DCT) è una trasformata matematica utilizzata nel campo della compressione dei dati e dell'elaborazione del segnale che converte una sequenza di dati in input in un set di coefficienti di trasformazione, che rappresentano la componente di frequenza del segnale originale. La 2-D Discrete Cosine Transform (DCT2) è una variante della DCT, che consiste nell'applicare la DCT alle righe e alle colonne di una matrice bidimensionale, particolarmente adatta per la compressione delle immagini in quanto consente di separare le componenti a bassa frequenza da quelle ad alta frequenza. La DCT2 viene applicata a blocchi rettangolari o quadrati di un'immagine e genera un set di coefficienti di trasformazione per ciascun blocco; questi coefficienti rappresentano la distribuzione spettrale delle frequenze all'interno del blocco. In tal modo la DCT2 è in grado di separare le componenti di bassa frequenza, come le sfumature di colore e le transizioni graduali, da quelle ad alta frequenza, come i dettagli e gli spigoli e, grazie a questa separazione, è possibile eseguire una compressione dell'immagine senza perdere eccessivamente la qualità visiva.

3 Librerie Python

Python è un linguaggio di programmazione ad alto livello, interpretato e dinamicamente tipizzato. Ciò significa che il codice sorgente Python viene eseguito direttamente senza la necessità di una fase di compilazione e che le variabili non devono essere esplicitamente dichiarate con un tipo. Python è multi piattaforma, il che significa che può essere eseguito su diversi sistemi operativi come Windows, macOS e Linux. Inoltre, Python è open source, il che implica che il suo codice sorgente è disponibile gratuitamente e può essere modificato e distribuito liberamente. Python contiene nativamente metodi per implementare la DCT2, ma esistono diverse altre librerie open source di cui è possibile avvalersi per la soluzione del medesimo problema. Nello specifico sono state scelte la libreria open source *SciPy* 1.9.0 e la libreria *NumPy* 1.22.4. *SciPy* è una libreria open source di Python che fornisce funzioni e strumenti per lavorare con algoritmi matematici e scientifici. È costruita sulla base di *NumPy*, una libreria per la manipolazione di array multidimensionali, e fornisce numerose funzionalità aggiuntive per l'ottimizzazione, l'integrazione, l'interpolazione, l'algebra lineare, le trasformate di Fourier, il trattamento dei segnali e delle immagini, e molto altro. Di queste due librerie sono stati utilizzati i due sotto pacchetti *scipy.fftpack* e *numpy.fft*. Entrambi i pacchetti forniscono funzioni per eseguire operazioni di trasformata di Fourier veloce (FFT) e altre trasformate nel dominio delle frequenze.
Per quanto riguarda la metodologia risolutiva vera e propria sono state utilizzate le funzioni *dct* e *idct* del sotto pacchetto *scipy.fftpack* per implementare la DCT tramite libreria open source, mentre per l'implementazione della funzione custom è stato utilizzata la funzione *rfft* del sotto pacchetto *numpy.fft* . Nello specifico queste funzioni svolgono le seguenti elaborazioni:

- ***scipy.fftpack.dct***: funzione che implementa la DCT per un array unidimensionale. La DCT multidimensionale viene implementata applicando la DCT monodimensionale prima per righe

e poi per colonne. Tale funzione restituisce un array contenente i coefficienti DCT calcolati dalla trasformata.

- ***scipy.fftpack.idct***: funzione che implementa l'inversa della DCT per un array unidimensionale o multidimensionale. La DCT inversa converte una sequenza di coefficienti nel dominio della frequenza nella sequenza originale nel dominio del tempo. La funzione restituisce un array contenente la sequenza originale ricostruita dalla DCT inversa.

- ***numpy.fft.rfft***: funzione che calcola la FFT di un segnale reale unidimensionale. Tale funzione restituisce un array contenente i coefficienti della FFT per il segnale reale.

Altre librerie di supporto utilizzate sono:

- ***NumPy*** 1.22.4: libreria essenziale poiché offre funzioni matematiche complete, routine di algebra lineare e gestione e creazione di matrici nonché la gestione e la manipolazione di grandi quantità di dati numerici in Python, grazie alla sua elevata efficienza computazionale.

- ***PIL*** 1.1.7: libreria open source per l'elaborazione delle immagini in Python. Fornisce una vasta gamma di funzionalità per aprire, manipolare e salvare immagini in diversi formati, come JPEG, PNG, BMP, TIFF e molti altri. In particolare di questa libreria si è fatto uso del modulo *Image* per la manipolazione ed elaborazione delle immagini.

- ***Pandas*** 1.4.3: libreria open source per la gestione e l'analisi dei dati in Python, che fornisce strutture dati flessibili e potenti come i dataframe per la manipolazione di dati strutturati e non strutturati. Pandas è particolarmente utile per eseguire operazioni di preprocessing, pulizia, elaborazione e analisi di dati provenienti da fonti diverse, come file CSV, fogli di calcolo Excel, database SQL e molto altro.

- ***Matplotlib*** 3.5.1: libreria open source per la visualizzazione dei dati in Python, che fornisce un'ampia gamma di funzionalità per la creazione di grafici, istogrammi, diagrammi a barre, diagrammi a dispersione, mappe di calore e molti altri tipi di visualizzazioni. In particolare di questa libreria si è fatto uso del modulo *pyplot* per la generazione dei grafici di confronto delle prestazioni.

- ***PySimpleGUI*** 4.50.0: libreria che offre un'interfaccia semplice per creare applicazioni desktop grafiche (GUI). La libreria fornisce una vasta gamma di widget e opzioni di personalizzazione, consentendo di creare bottoni, caselle di testo, etichette, menu, tabelle e molti altri elementi di interfaccia grafica. Inoltre, PySimpleGUI offre funzionalità come la gestione degli eventi degli elementi dell'interfaccia, la validazione degli input dell'utente e la gestione delle finestre di dialogo.

A questo punto è necessario controllare che la funzione scelta sia effettivamente efficace nel calcolare correttamente la DCT di una matrice. É stata scelta una matrice 8x8 (Tab.1) ed è stato controllato che la DCT2 venisse applicata correttamente, ottenendo come risultato la matrice in Tab.2).

231	32	233	161	24	71	140	245
247	40	248	245	124	204	36	107
234	202	245	167	9	217	239	173
193	190	100	167	43	180	8	70
11	24	210	177	81	243	8	112
97	195	203	47	125	114	165	181
193	70	174	167	41	30	127	245
87	149	57	192	65	129	178	228

Tabella 1: Matrice 8x8 per il test

1.11e+03	4.40e+01	7.59e+01	-1.38e+02	3.50e+00	1.22e+02	1.95e+02	-1.01e+02
7.71e+01	1.14e+02	-2.18e+01	4.13e+01	8.77e+00	9.90e+01	1.38e+02	1.09e+01
4.48e+01	-6.27e+01	1.11e+02	-7.63e+01	1.24e+02	9.55e+01	-3.98e+01	5.85e+01
-6.99e+01	-4.02e+01	-2.34e+01	-7.67e+01	2.66e+01	-3.68e+01	6.61e+01	1.25e+02
-1.09e+02	-4.33e+01	-5.55e+01	8.17e+00	3.02e+01	-2.86e+01	2.44e+00	-9.41e+01
-5.38e+00	5.66e+01	1.73e+02	-3.54e+01	3.23e+01	3.34e+01	-5.81e+01	1.90e+01
7.88e+01	-6.45e+01	1.18e+02	-1.50e+01	-1.37e+02	-3.06e+01	-1.05e+02	3.98e+01
1.97e+01	-7.81e+01	9.72e-01	-7.23e+01	-2.15e+01	8.13e+01	6.37e+01	5.90e+00

Tabella 2: Matrice trasformata dalla DCT2

Inoltre, viene controllato che la prima riga della matrice venga trasformata correttamente da una DCT monodimensionale, come mostrato in Tab3.

Prima riga	231	32	233	161	24	71	140	245
Val.Atteso	4.01e+02	6.60e+00	1.09e+02	-1.12e+02	6.54e+01	1.21e+02	1.16e+02	2.88e+01
Libreria	4.02e+02	6.60e+00	1.09e+02	-1.13e+02	6.54e+01	1.22e+02	1.17e+02	2.88e+01

Tabella 3: Trasformata DCT prima riga

Come si può notare, la libreria ottiene risultati identici ai valori attesi, se non per i valori evidenziati in blu che peccano di una unità, probabilmente a causa di un errore di approssimazione più che di scaling o di non correttezza dell'algoritmo. Per comodità è stata mostrata solo la trasformazione della prima riga, ma anche la DTC2 crea una matrice trasformata praticamente identica a quella attesa, se non per errori di approssimazione.

4 Risultati

In questa sezione verranno presentati i risultati ottenuti, ovvero una analisi dei tempi sulla trasformazione di matrici quadrate da parte di una funzione custom contro i tempi della funzione di libreria. Una seconda parte, invece, verterà sull'analisi di un algoritmo di compressione simile a JPEG, implementato senza l'uso della matrice di quantizzazione.

5 Analisi dei tempi custom e di libreria

Per analizzare i tempi di analisi è stata, in primis, creata una funzione custom che applica la DCT monodimensionale. La DCT monodimensionale viene applicata prima alle righe e successivamente, sulla matrice risultante da questa prima applicazione di DCT, essa viene riapplicata alle colonne. A questo punto, vengono create delle matrici quadrate con valori randomici compresi tra 0 e 255. Per la scelta random di valori viene applicato un seed, 42[1] per essere precisi, in maniera da garantire la riproducibilità dei risultati. Le matrici hanno dimensioni a scalare come mostrato nella Tab.4. Da questa analisi ci si aspetta che la funzione custom performi con tempi dell'ordine $O(n^3)$, mentre dalla libreria, la cui funzione dovrebbe essere in versione fast, ci si attende un miglioramento dei tempi, nell'ordine di $O(n^2)$. Di seguito una tabella e un grafico che mostrano l'analisi svolta:

Dimensione	Dct2(s)	Scipy(s)
500	0,033	0,002
750	0,068	0,007
1000	0,115	0,011
2500	0,673	0,081
5000	2,608	0,374

Dimensione	Dct2(s)	Scipy(s)
7500	5,910	0,910
10000	10,636	1,662
12500	18,002	2,714
15000	26,885	4,065

Tabella 4: Tempi della funzione custom e della libreria

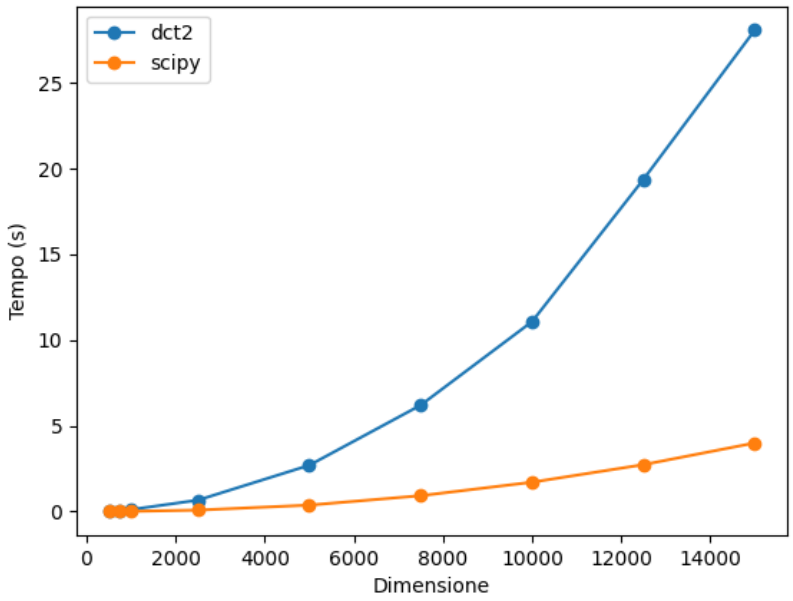


Fig. 1: Tempi di calcolo custom e libreria a confronto

Come si può notare in Fig.1 l'andamento delle due funzioni è in linea con i valori attesi: i tempi sono maggiori per la funzione custom all'aumentare della dimensione dell'input.

[1] La risposta

6 Risultati Compressione

In questa sezione si andrà ad analizzare i risultati ottenuti con una compressione di una immagine in maniera semplice. Il procedimento è il seguente:

- Si sceglie una immagine in toni di grigio, in formato BMP
- Si sceglie un intero F. FxF sarà la dimensione dei blocchetti in cui andremo a suddividere l'immagine. Per poter ottenere un risultato sensato è stato chiesto di inserire un F con dimensione massima minore della metà del lato più corto dell'immagine.
- Viene scelto un intero D compreso tra 1 e 2F-2. Questo sarà il grado di compressione, spiegato meglio seguito, che definisce una diagonale dei blocchetti atta ad individuare le frequenze da eliminare
- Partendo dal primo pixel in alto a sinistra si suddivide l'immagine in blocchetti FxF e ad ogni blocchetto si applica la DCT2
- Ottenuta la matrice dalla DCT2 si sostituiscono a zero tutte le frequenze per gli indici i, j tali che $i + j > d$ come mostrato in Tab.5
- Dopo aver rimosso le frequenze si ritrasforma la matrice tramite IDCT2, ovvero la funzione inversa, per ritornare ad un'immagine tramite una fase di decompressione dei blocchetti

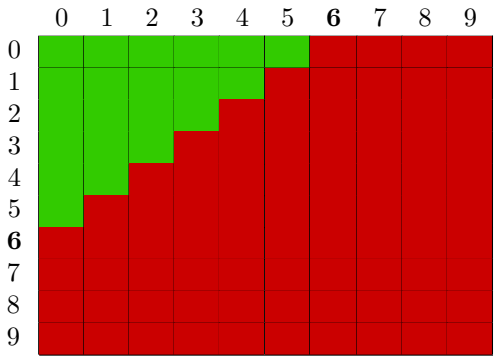


Tabella 5: Esempio di celle messe a zero con F=10 e D=5

6.1 Compressione con F costante

Di seguito verranno mostrate alcune compressioni sull'immagine di un cervo. Essa è di dimensioni circa 1000x600 pixel, ed è stata scelta per la quantità di dettagli fini, come manto e peli, e per l'elevato contrasto di alcuni punti con lo sfondo, come sulle corna o sotto il naso, zone in cui, inoltre, la variabilità dei colori è alta. Questi punti dovrebbero dare una idea della compressione nei casi di minor eliminazione frequenze. Ad alte compressioni invece l'effetto si vede su tutta l'immagine a prescindere dal tipo (tralasciando casi di immagine monocolore).

(a) Immagine originale

(b) F=50 D=20

(c) F=50 D=10

(d) F=50 D=3

Fig. 2: Esempi di compressione con F=50 fisso

In Fig.2 sono state applicate 3 diverse compressioni mantenendo la dimensione dei blocchetti costante a 50x50. Si può notare come nell'immagine (b), che ha subito una leggera compressione, inizino a crearsi dei lievi fenomeni attorno a corna e muso, più accentuati nell'immagine (c) a cui è stata applicata una maggiore compressione. La figura (d) mostra come una compressione pesante, su blocchetti relativamente grandi, porti ad un deterioramento dell'immagine, lasciando un idea delle forme dell'animale ma senza permettere di distinguerne i dettagli. In più si creano dei distacchi evidenti tra i quadrati dove è stata effettuata la compressione, visibili sopratutto sul muso. Da qui si deduce che una compressione con F grandi non produce risultati accettabili.

6.2 Compressione con D/F costante

Un altro modo per testare vari livelli di compressione potrebbe essere quello di mantenere costante il rapporto D/F. Nel caso specifico è stato scelto di applicare una forte compressione, scegliendo il rapporto D/F = 0.2

(a) F=10 D=2

(b) F=50 D=10

(c) F=100 D=20

(d) F=300 D=60

Fig. 3: Esempi di compressione con rapporto D/F = 0.2

In Fig.3 si può vedere come una compressione della stessa quantità porti a risultati molto diversi man mano che la dimensione dei blocchi aumenta. L'immagine (a) a cui è stata applicata una compressione con F=10 e D=2 riesce a mantenere l'immagine quasi intatta, se non per la perdita minima di dettaglio. Man mano che la dimensione dei blocchi aumenta, come in (b), (c) e (d), gli effetti di compressione risultano sempre più preponderanti, sempre sulle zone a più alta variabilità. Quindi nel caso specifico, un F=10 risulta essere un buon valore per la scelta della dimensione dei blocchetti.

6.3 Test di compressione con F=10

Compresa l'importanza di avere un F basso, si può procedere a testare quale livello di compressione sarebbe meglio non superare per mantenere una qualità d'immagine accettabile. In particolare verranno testate compressioni con D = 1, 2, 3 e 5.

(a) D=5

(b) D=3

(c) D=2

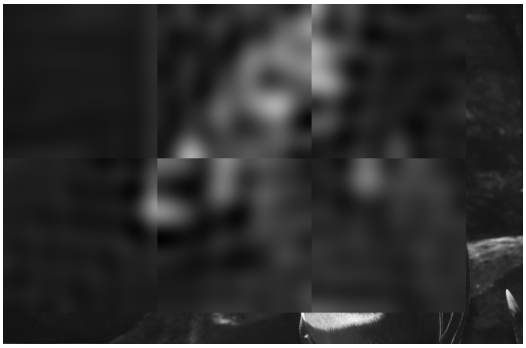
(d) D=1

Fig. 4: Compressione a vari livelli con F=10

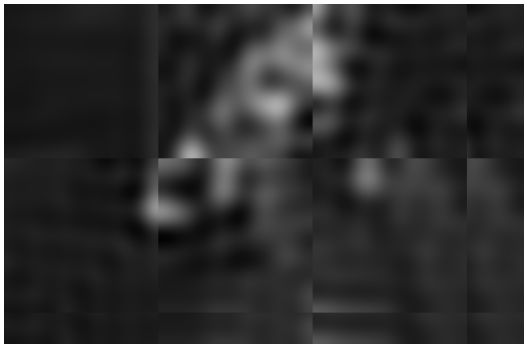
In Fig.4 si può vedere come vari livelli di compressioni su blocchetti 10x10 portino a risultati diversi, ma nessuno di questi vada a denaturare troppo l'immagine. Con compressioni D=5 e 3 l'immagine risulta praticamente identica all'originale,mentre con D=2 si inizia ad intravedere qualche fenomeno particolare seppur lieve. Utilizzando la compressione massima D=1 l'immagine viene leggermente denaturata, mantenendo comunque le caratteristiche principali. Dai risultati ottenuti è possibile dire che quindi, mantenendo i valori di F bassi, per ottenere una compressione non denaturante è consigliabile non scendere sotto 0.2 per il rapporto D/F.

6.4 Completamento immagine

La compressione, come detto, viene applicata su blocchetti di dimensione FxF. Ovviamente i blocchetti, nella maggior parte dei casi, non potranno essere estesi a tutta l'immagine, poiché le dimensioni delle stesse sono irregolari e non divisibili per F. É stato deciso quindi di duplicare i pixel compressi e completare così l'immagine. Per primo vengono duplicate le ultime righe compresse e sostituite a quelle non compresse. Lo stesso procedimento viene poi applicato alle colonne, in maniera da completare anche il blocco nell'angolo in basso a destra che sarebbe rimasto escluso. A seguire un esempio di estrema compressione con F=300 e D=10 che permette di distinguere i blocchi di compressione e che serve solo per mostrare che effettivamente i pixel vengono sostituiti:



(a) Immagine compressa senza funzione di fill



(b) Immagine compressa con funzione di fill

Fig. 5: F=300 D=10 con e senza fill

7 Conclusioni

Concludendo sono state testate le velocità di calcolo di una libreria che applica la DCT contro una funzione custom. É emerso che, come ci si aspettava, la funzione da libreria, essendo in versione fast, ottiene delle performance migliori.
Dopo di che è stato implementato un algoritmo che applica una compressione utilizzando la DCT e un taglio di frequenze su un'immagine. É emerso che, al fine di non denaturare l'immagine, è consigliabile mantenere la dimensione dei blocchetti bassa e il rapporto tra valore di taglio e dimensione dei blocchetti non inferiore a 0.2.