Compressione di immagini tramite DCT
Attraverso l'utilizzo di Python

`https://github.com/nicoripa/MCS_Project/`

Lorenzo Rovida, Nicolò Ripamonti
l.rovida1@campus.unimib.it, n.ripamonti@campus.unimib.it
817151, 816171

Dipartimento di Informatica, Sistemi e Comunicazione, Universitá degli Studi di Milano-Bicocca, Milano, Italia

---

Abstract

Lo scopo di questo progetto è quello di utilizzare l'implementazione dell'operazione matematica denominata **DCT2** in un ambiente *open source* e di studiare gli effetti di un algoritmo di compressione di tipo jpeg (senza utilizzare una matrice di quantizzazione) sulle delle immagini in toni di grigio di formato *bitmap*. Il progetto si divide in tue macro parti:

Parte 1 Implementare la DCT2 in un ambiente open source a scelta e confrontare i tempi di esecuzione con la DCT2 ottenuta usando la libreria dell'ambiente utilizzato. In particolare, bisogna utilizzare array quadrati $N \times N$ con N crescente e rappresentare su un grafico i tempi di esecuzione dei due algoritmi.

Parte 2 Scrivere un software in grado di far scegliere all'utente un'immagine in formato bitmap e successivamente comprimere tale immagine utilizzando la DCT2, tagliare le frequenze che l'utente sceglie di eliminare, utilizzare l'inversa della DCT e far visualizzare a schermo l'immagine originale con quella ottenuta dopo aver modificato le frequenze.

---

1. Introduzione

1.1. Discrete Cosine Transform

La trasformata discreta del coseno o DCT, è la più diffusa funzione che provvede alla compressione spaziale, capace di rilevare le variazioni di informazione tra un'area e quella contigua di un'immagine digitale trascurando le ripetizioni.

È una trasformata simile alla trasformata discreta di Fourier (DFT), ma fa uso solo di numeri reali.

La variante più comune della trasformata discreta del coseno è la DCT tipo II che è spesso chiamata semplicemente DCT; la sua inversa, la DCT tipo III è, in corrispondenza, chiamata spesso DCT inversa o IDCT.

La DCT, e in particolare la DCT-II, è spesso usata nell'elaborazione dei segnali e delle immagini, specialmente per la compressione con perdita (compressione di tipo **lossy**). L'algoritmo JPEG è basato sulla Trasformata discreta del coseno bidimensionale (DCT2), che viene applicata su blocchi di 8x8 pixel, i cui risultati sono poi quantizzati e compressi con tecniche basate sull'entropia (come la Codifica di Huffman o la Codifica aritmetica).

La formula che descrive il funzionamento della DCT bidimensionale è la seguente:

$$c_{kl} = a_{kl} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f_{ij} \, cos\left(k\pi \frac{2i+1}{2N}\right) \, cos\left(l\pi \frac{2j+1}{2M}\right)$$

Dove $a_{kl} = a_k^N a_l^M$:

$$a_{00} = \frac{1}{\sqrt{NM}}, \; e \; a_{k0} = a_{0l} = \sqrt{\frac{2}{NM}}, \; a_{kl} = \frac{2}{\sqrt{NM}}, \; k, l \geq 1.$$

Mentre f_{ij} sono gli indici bidimensionali:

$$\mathbf{f} = (f_{ij}), \; i = 0, ..., N-1, j = 0, ..., M-1.$$

1.2. .Formato bitmap

Windows bitmap è un formato dati utilizzato per la rappresentazione di immagini **raster** sui sistemi operativi Microsoft Windows. Noto soprattutto come formato di file, fu introdotto con Windows 3.0 nel 1990. Le bitmap,

come sono comunemente chiamati i file d'immagine di questo tipo, hanno generalmente l'estensione **.bmp**.

Sono state sviluppate tre versioni del formato bitmap. La prima e più comunemente utilizzata è la versione 3: non esistono versioni antecedenti. Le versioni successive 4 e 5 si incontrano piuttosto raramente.

Il formato di file Windows bitmap nella versione 3 permette operazioni di lettura e scrittura molto veloci e senza perdita di qualità, ma richiede generalmente una maggior quantità di memoria rispetto ad altri formati analoghi.

Le immagini bitmap possono avere una profondità di 1, 4, 8, 16, 24 o 32 bit per pixel. Le bitmap con 1, 4 e 8 bit contengono una tavolozza per la conversione dei (rispettivamente 2, 16 e 256) possibili indici numerici nei rispettivi colori. Nelle immagini con profondità più alta il colore non è indicizzato bensì codificato direttamente nelle sue componenti cromatiche RGB; con 16 o 32 bit per pixel alcuni bit possono rimanere inutilizzati.

Nel caso in esame le immagini in formato *.bmp* sono state scelte con colori in scala di grigi, ovvero immagini con una profondità di 8 bit. Tali bit servono per convertire gli indici numerici nei rispettivi bit, in questo caso da 0, che identifica il colore nero, a 255, che identifica il colore bianco. Tutti i valori tra l'1 e il 254 sono tutte le possibili varianti di grigio.

1.3. Calcolatore utilizzato

Per effettuare i calcoli sugli script è stato utilizzato un MacBook Pro 2016 con le seguenti caratteristiche hardware:

Processore 2 GHz Intel Core i5 dual-core

Architettura CPU x64

Memoria 8 GB 1867 MHz LPDDR3

Scheda grafica Intel Iris Graphics 540 1536 MB

2. Ambiente di sviluppo

L'ambiente di sviluppo scelto per il progetto, sia per la parte 1 che per la parte 2, è stato **Python**.

Versione: 3.7.3

Referenze: Sito - Documentazione

Python è un linguaggio di programmazione di più "alto livello" rispetto alla maggior parte degli altri linguaggi. È orientato a oggetti, ma non in maniera ferrea come ad esempio Java; adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

È un linguaggio multi-paradigma che ha tra i principali obiettivi: dinamicità, semplicità e flessibilità. Supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programmazione funzionale e riflessione.

Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche.

3. Librerie

Nello specifico sono state utilizzate diverse librerie sia per l'implementazione del codice della parte 1 del progetto, sia per la parte 2.

Esse nello specifico sono:

TkInter Sito - Versione 3.8.3

È una libreria che permette di creare interfacce grafiche nella programmazione con Python, molto conosciuta e utilizzata per la sua leggerezza e stabilità.

SciPy Sito - Versione 1.19.0

È una libreria open source di algoritmi e strumenti matematici. Contiene moduli per l'ottimizzazione, per l'algebra lineare, elaborazione di segnali ed immagini e altri strumenti comuni nelle scienze e nell'ingegneria. Trova utilizzo in quei programmatori che usano anche MATLAB. All'interno di SciPy sono presenti diversi pacchetti tra cui **fftpack**, che contiene tutti gli algoritmi della trasformata discreta di Fourier e anche classi per l'implementazione della trasformata discreta del coseno (**DCT**);

NumPy Sito - Versione 1.18.4

È un pacchetto per l'elaborazione scientifica con Python. In particolare questo pacchetto è stato utilizzato per effettuare operazioni su array bidimensionali;

MathPlotLib Sito - Versione 3.2.1

È una libreria completa per la creazione di visualizzazioni statiche, animate e interattive. Nel nostro interesse è stata usata per stampare le immagini nella conclusione dello script della seconda parte del progetto;

Random Sito - Versione 3.0

Questo modulo implementa generatori di numeri pseudo-casuali per varie distribuzioni. È stata utilizzata questa libreria per la generazione di matrici casuali per lo studio dei tempi di esecuzione delle DCT nella prima parte del progetto;

Time Sito - Versione 3.7

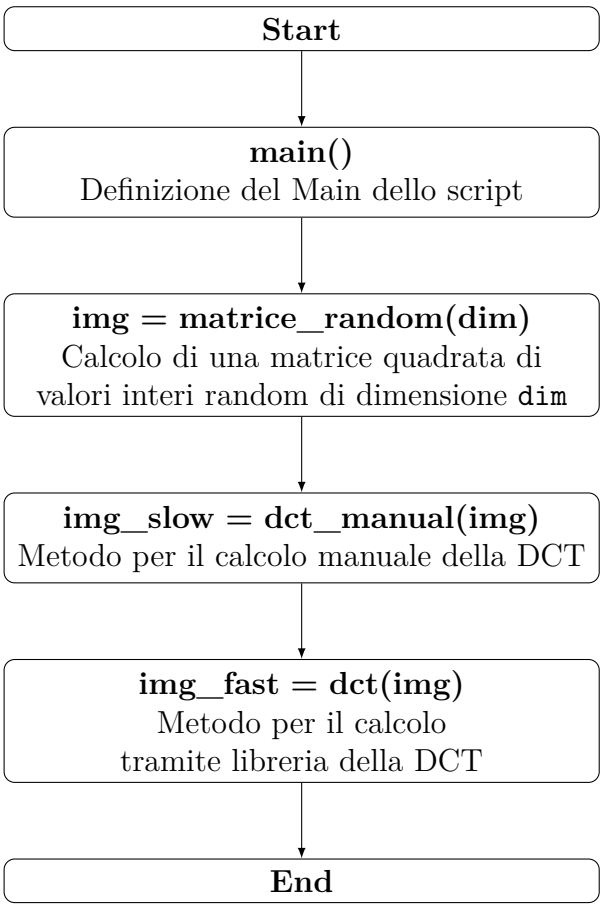
Questo modulo offre varie funzioni correlate al tempo. Utilizzata per la misurazione dei tempi di esecuzione delle CDT nella prima parte del progetto;

CV2 Sito - Versione 4.3.0

È un modulo facente parte della libreria OpenCV. CV2 è stata utilizzata all'interno dell'interfaccia grafica per poter far scegliere all'utente il file .bmp.

4. Implementazioni

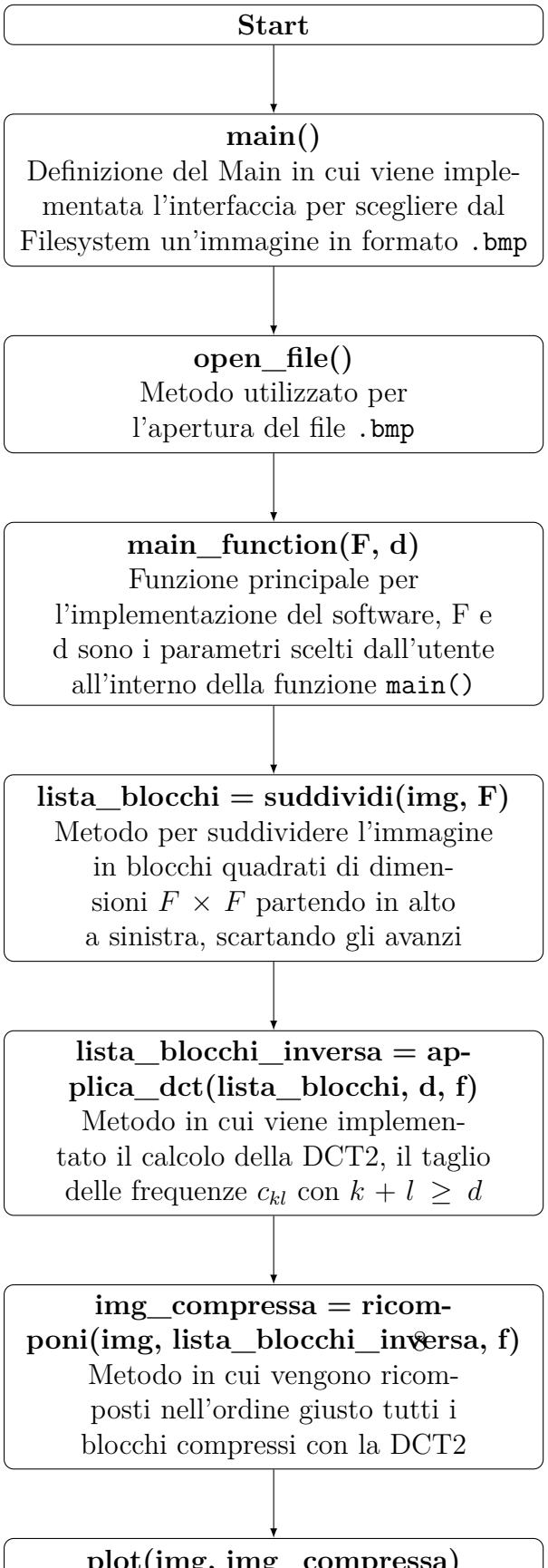
4.1. Parte 1



4.1.1. Codice

L'implementazione su ambiente Python è stata effettuata tramite questo script:

SCRIVERE SCRIPT.


Start
main()
Definizione del Main in cui viene implementata l'interfaccia per scegliere dal Filesystem un'immagine in formato .bmp
open_file()
Metodo utilizzato per l'apertura del file .bmp
main_function(F, d)
Funzione principale per l'implementazione del software, F e d sono i parametri scelti dall'utente all'interno della funzione main()
lista_blocchi = suddividi(img, F)
Metodo per suddividere l'immagine in blocchi quadrati di dimensioni $F \times F$ partendo in alto a sinistra, scartando gli avanzi
lista_blocchi_inversa = applica_dct(lista_blocchi, d, f)
Metodo in cui viene implementato il calcolo della DCT2, il taglio delle frequenze c_{kl} con $k + l \geq d$
img_compressa = ricomponi(img, lista_blocchi_inversa, f)
Metodo in cui vengono ricomposti nell'ordine giusto tutti i blocchi compressi con la DCT2
plot(img, img_compressa)

4.2.1. Codice

L'implementazione su ambiente Python è stata effettuata tramite questo script:

SCRIVERE SCRIPT

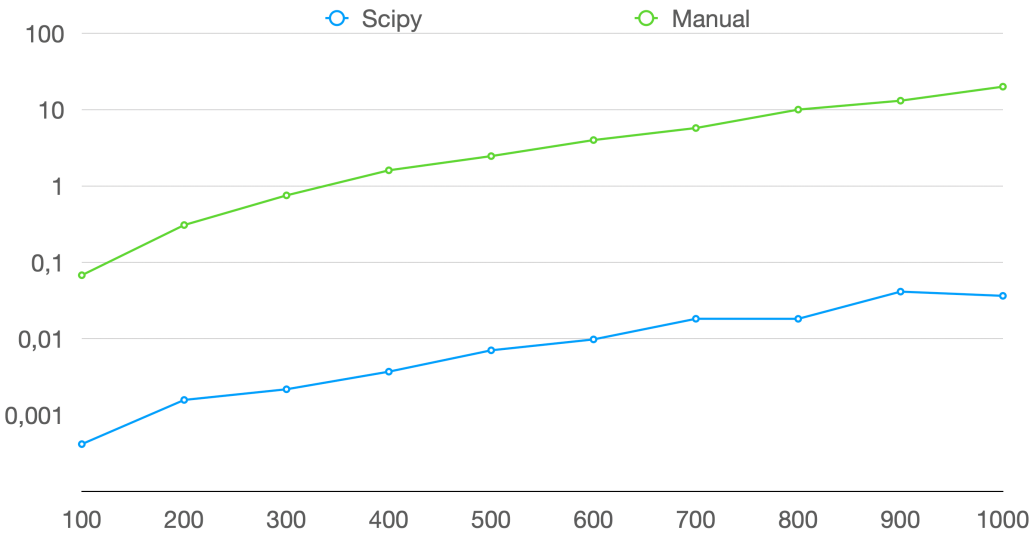
5. Elaborazioni e risultati

5.1. Parte 1

I risultati della prima parte del progetto riguardano il confronto dei tempi d'esecuzione della DCT2. In particolare il confronto è stato fatto tra una DCT implementata a mano e la funzione DCT del pacchetto SciPy.fftpack.

Gli array bidimensionali utilizzati di dimensione $N \times N$ sono stati creati in maniera casuale e sono state scelte dimensioni che variano da 100×100 a 1000×1000. Non sono state scelte dimensioni maggiori, sia per motivi di eccessivo tempo di calcolo, sia perché un'array 1000×1000 rispecchia un'immagine di 1000×1000 pixel, definizione già abbastanza alta, paragonabile in termini di numero di pixel al formato HD a 720p.

Durante l'implementazione abbiamo fatto molta attenzione che il nostro algoritmo venisse implementato correttamente confrontando sempre i risultati degli array dopo il calcolo di entrambe le DCT.



Nella figura sopra si possono notare come i tempi della DCT implementata a mano (Manual) siano nettamente superiori rispetto alla DCT implementata dal pacchetto di Python. Questo è del tutto normale in quanto le librerie sono sicuramente implementate meglio e più efficienti.

Nella figura sotto sono riportati in una tabella tutti i tempi di esecuzione della DCT.

		Dimensione array bidimensionale [Pixel]									
		100	200	300	400	500	600	700	800	900	1000
Tempo [Sec]	Scipy	0,00041484	0,00157308	0,00217103	0,0036931	0,00703287	0,00976991	0,01821494	0,01818513	0,04131293	0,03638101
	Manual	0,06779313	0,30789494	0,75378012	1,6057541	2,46435403	3,99684786	5,74928379	10,0226428	13,1171091	20,0206999

Si ricorda che i tempi di un algoritmo per DCT2 implementato da una libreria sono di circa $N^2 log N$, mentre i tempi di esecuzione di una DCT2 manuale si aggirano intorno a N^3, dove con N si intende ovviamente la dimensione della matrice.

Detto ciò, i risultati finali dei tempi rispecchiamo a pieno tale regola, quindi possiamo tenerci soddisfatti circa la prima parte del progetto.

5.2. Parte 2

Nella sezione corrente verranno presentati i risultati delle varie elaborazioni su grafici.