

Compressione di immagini tramite DCT Attraverso l'utilizzo di Python

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

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

*Dipartimento di Informatica, Sistemi e Comunicazione, Università 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 due 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 $a_{kl} = a_k^N a_l^M$:

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

Mentre f_{ij} sono gli indici bidimensionali:

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

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](#)

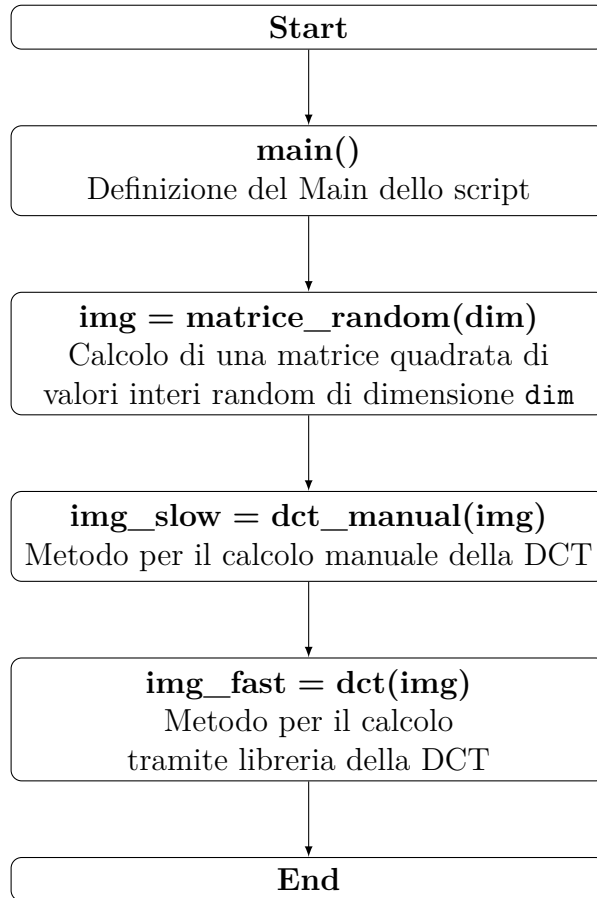
Python è un linguaggio di programmazione di più "alto livello" rispetto alla maggior parte degli altri linguaggi. È orientato a oggetti, ma non in maniera ferrea come ad esempio Java; adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing.

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.

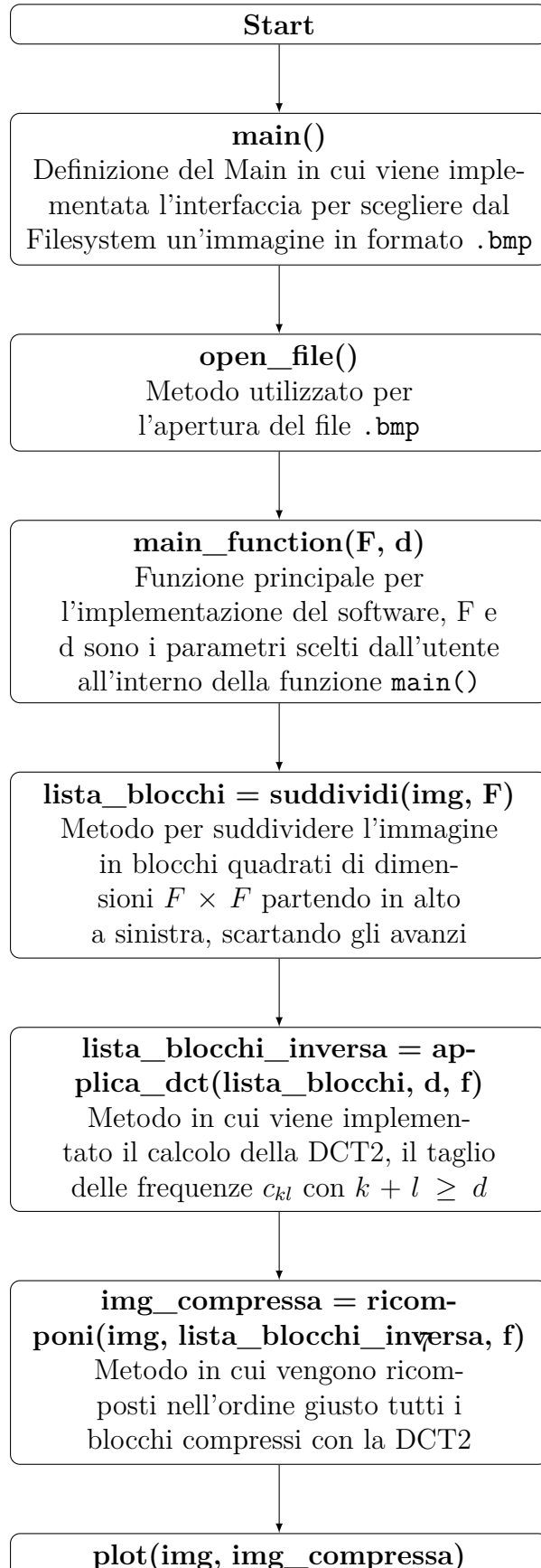
Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche.

3. Implementazioni

3.1. Parte 1



3.2. Parte 2



3.3. Codice

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

SCRIVERE SCRIPT.

4. Elaborazioni e risultati