Compressione di immagini tramite la DCT

Lorenzo Olearo `l.olearo@campus.unimib.it`

Alessandro Riva `a.riva86@campus.unimib.it`

A.A. 2022-2023

Lo scopo di questo progetto è di utilizzare l'implementazione della trasformata 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 immagini in toni di grigio. Comprende l'implementazione di un codice e la scrittura di una relazione da consegnare al docente.

Tutto il codice sorgente per il la realizzazione del progetto comprendente consegne, test, esperimenti, grafici e relazione è stato pubblicato al seguente repository GitHub: `https://github.com/LorenzoOlearo/cosine-image-compression`. Il repository rimarrà privato fino alla data dell'esame, per avere l'accesso, le chiediamo per ragioni tecniche di comunicarcelo via mail.

1 Confronto tra DCT diretta e veloce

Nella prima parte del progetto si richiede di confrontare le prestazioni della DCT2 come spiegata a lezione nella sua forma diretta, con quella di una libreria open source a scelta che si presuppone essere nella sua versione *fast*.

La trasformata DCT2 diretta è stata implementata in Python mentre per la sua versione *fast* è stata utilizzata la libreria Python `fftpack` di `scipy`.

La trasformata DCT che è stata implementata è la DCT di tipo II secondo la seguente definizione:

$$c_k = \alpha_k^N \sum_{i=0}^{N-1} f_i \cos\left(\pi k \frac{2i+1}{2N}\right), \qquad k = 0, \cdots, N-1$$

Dove, α_k^N è il fattore di normalizzazione ed è definito come:

$$\begin{cases} \alpha_k^N = \sqrt{\frac{1}{N}} & per \quad k = 0 \\ \alpha_k^N = \sqrt{\frac{2}{N}} & per \quad k = 1, \cdots, N-1 \end{cases}$$

in modo che le basi dello spazio dei coseni siano ortonormali.

Per calcolare la DCT2 su matrici si è calcolata la DCT, come definita sopra, prima sulle righe e poi sulle colonne della matrice.

Per confrontare i tempi di esecuzione delle due implementazioni sono state create una serie di matrici di interi di dimensioni crescenti, da 2x2 a 1024x1024, con valori casuali compresi tra 0 e 255. La scelta dell'upper-buond dei test dipende soltanto da i limiti computazionali dei calcolatori a nostra disposizine.

Ogni matrice è stata poi trasformata con entrambe le implementazioni della DCT2 tenendo traccia dei rispettivi tempi di esecuzione. I risultati sono stati riportati su un grafico che mette in relazione le dimensioni delle matrici con i tempi di esecuzione degli algoritmi in scala logaritmica.

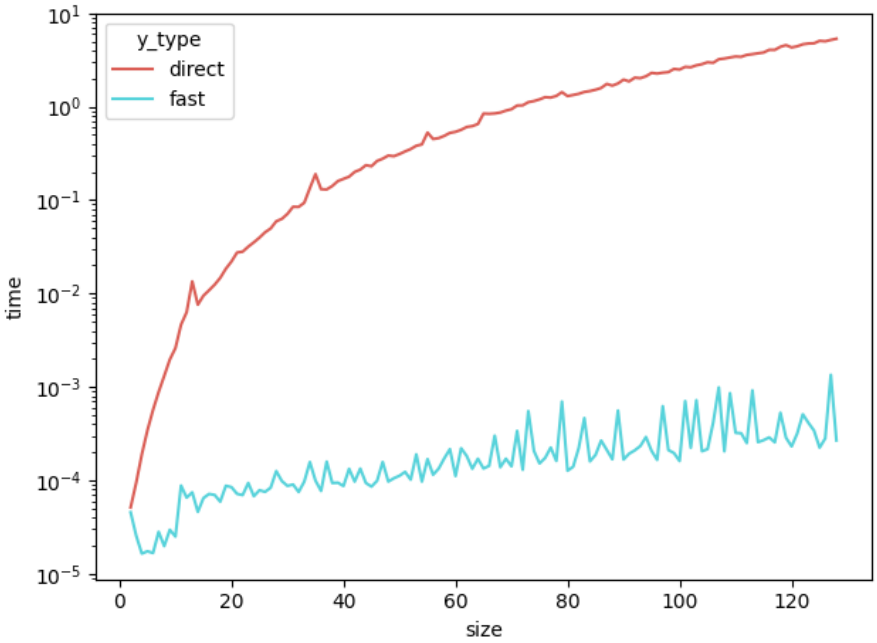

Figure 1: Confronto tra DCT diretta e veloce, sull'asse delle ascisse la dimensione delle metrici, sull'asse delle ordinate il tempo di esecuzione in scala logaritmica.

Come si può osservare dal grafico in Figura 1, il tempo computazione della DCT calcolata tramite il metodo diretto cresce in maniera decisamente più rapida rispetto a quello della DCT calcolata tramite la libreria `fft` di `scipy`.

La DCT2 calcolata tramite il metodo diretto presenta tempi proporzionali a N^3 rendendola quindi inutilizzabile per matrici di grandi dimensioni, la DCT calcolata tramite la libreria `fft` di `scipy` invece presenta tempi di esecuzione decisamente migliori.

Siccome la libreria `fft` di `scipy` utilizza la FFT, si è scelto di mettere a confronto i tempi di esecuzione della traformatata DCT2 su input di dimensioni pari a potenze di 2 crescenti.

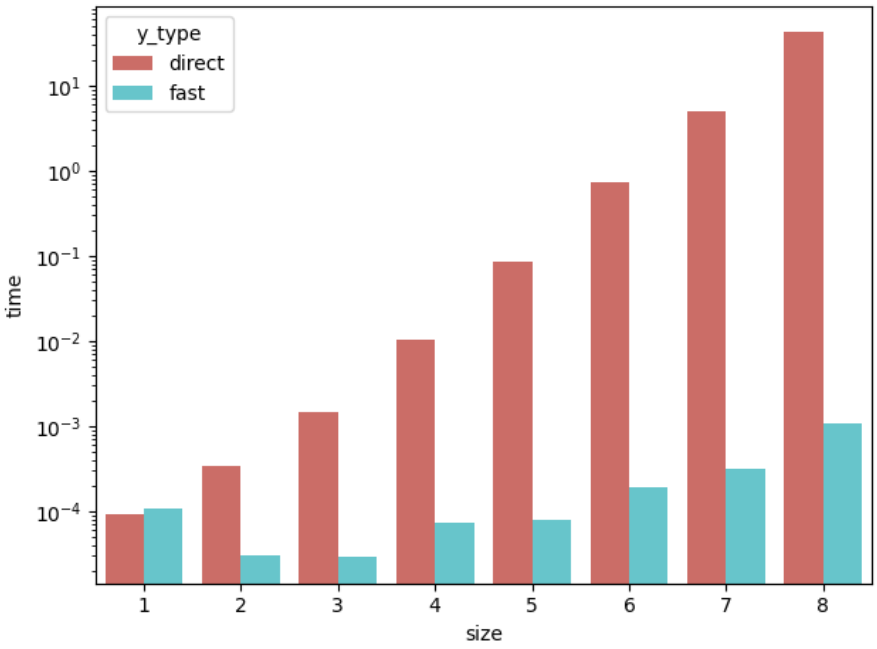

Figure 2: Confronto tra DCT diretta e veloce su input di dimensione pari a potenze di 2. Sull'asse delle ordinate il tempo in scala logaritmica, sull'asse delle ascisse le potenze di due delle dimensioni delle matrici.

Anche in questo caso, come si può osservare dal grafico in figura 2, la DCT implementata tramite il medoto diretto risulta considerevolmente più lenta rispetto a quella implementata tramite la libreria `fft` di `scipy`.

Tramite il software sviluppato è possibile ripetere i test qui presentati. Per effettuare il confronto tra la DCT diretta che è stata implementata e quella *fast* di `scipy` su matrici di dimensione incrementali comprese tra un valore di lower e upper buond, si può lanciare il programma con i seguenti argomenti:

```
␣␣python␣main.py␣--performance␣--lower
```

2 Compressione di immagini

La seconda parte della consegna richiede la realizzazione di un'interfaccia grafica che permetta di caricare un'immagine in formato `BMP` in toni di grigio e di applicare un algoritmo di compressione JPEG senza però utilizzare una matrice di quantizzazione.

Anche per la realizzazione di questa seconda parte si è scelto di utilizzare Python, in particolare, l'interfaccia grafica è stata realizzata tramite la libreria `tkinter`.

Il software implementato permette all'utente di caricare un'immagine `BMP` in scala di grigi dal proprio filesystem sfruttando il *file picker* di `tkinter`.

Una volta caricata l'immagine all'interno dell'applicazione, l'utente è in grado di specificare:

- un intero F corrispondente alla dimensione in pixel dei *macro-blocchi* su cui effettuare la DCT2.
- un intero d compreso tra 0 e $2F - 2$ rappresentante il valore di taglio delle frequenza

Una volta trasfrormata l'immagine, l'applicazione mostra fianco a fianco l'immagine originale e quella su cui è stato eseguito l'algoritmo di compressione delle frequenze. L'applicazione permette inoltre `zoom` e `pan` delle due immagini simultaneamente.

AGGIUNGERE SCREENSHOT APPLICAZIONI