COMPRESSIONE DI IMMAGINI TRAMITE LA DCT

METODI DEL CALCOLO SCIENTIFICO - PROGETTO II

Mattia Pennati 793375
Francesco Prete 793389
Andrea Spreafico 793317

INTRODUZIONE

- Con l'avanzare della tecnologia, i dati sono sempre qualitativamente migliori ma anche *«più pesanti»*
- Questo ha portato ad un'inevitabile attenzione riguardo l'efficienza degli algoritmi di compressione
- Questi algoritmi sono, nella maggior parte dei casi, basati su trasformate che traslano il dominio da quello spaziale a quello delle frequenze in modo da poter operare direttamente sulle frequenze che compongono il segnale

PARTE 1 – DCT2

DCT2

- Una delle trasformate più famose è la DCT [1] (*Discrete Cosine Transform*) (DCT2 nel caso di input bidimensionali)
- La DCT trasforma un segnale digitale in input in una collezione *«ordinata»* di contributi di frequenze
- Lo scopo dell'applicazione di questa trasformata è quello di comprendere il contributo di ogni frequenza al fine di eliminare quelle con un contributo minimo, permettendo di ridurre la quantità di informazione necessaria a memorizzare il segnale, minimizzando la perdita di qualità

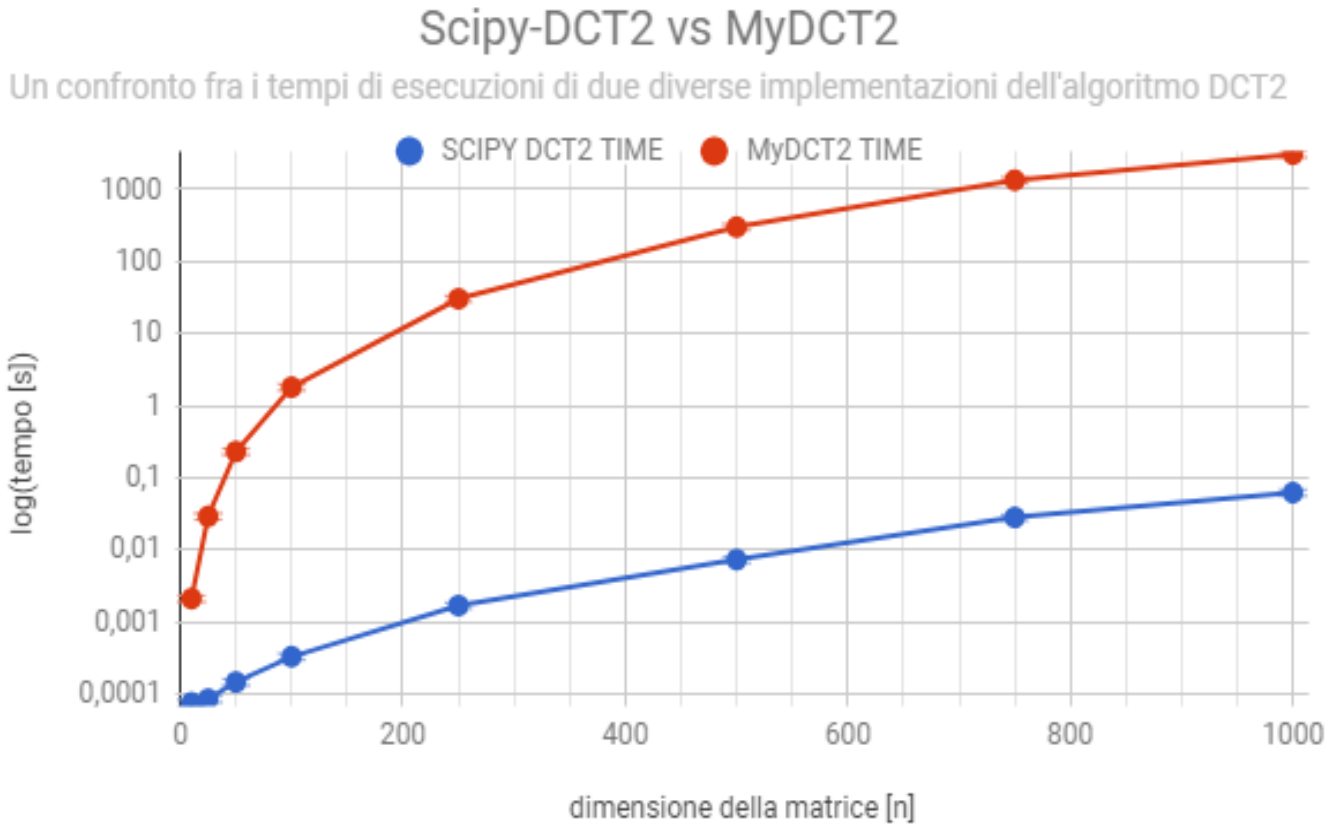
OBIETTIVI

- L'obiettivo di questa parte è quello di implementare al DCT2 e di confrontarne le performance con quelle di una versione già presente in una specifica libreria. Abbiamo deciso di:
  - Implementare in Python dell'algoritmo teorico della DCT2
  - Confrontare la nostra implementazione con DCT[2][3] della libreria in Python «Scipy.fftpack»[4]

- Lo scopo principale è di valutare quanto la versione fornita dalla libreria migliori effettivamente il tempo di esecuzione rispetto alla versione originale

CONFRONTO

MATRIX SIZE	SCIPY-DCT2 TIME	MyDCT2 TIME
10	0,0000761	0,0021173
25	0,0000855	0,0292382
50	0,0001475	0,2319072
100	0,0003318	1,8103051
250	0,0016939	30,453742
500	0,0073518	300,64348
750	0,0282837	1340,1626
1000	0,0623399	3079,0002



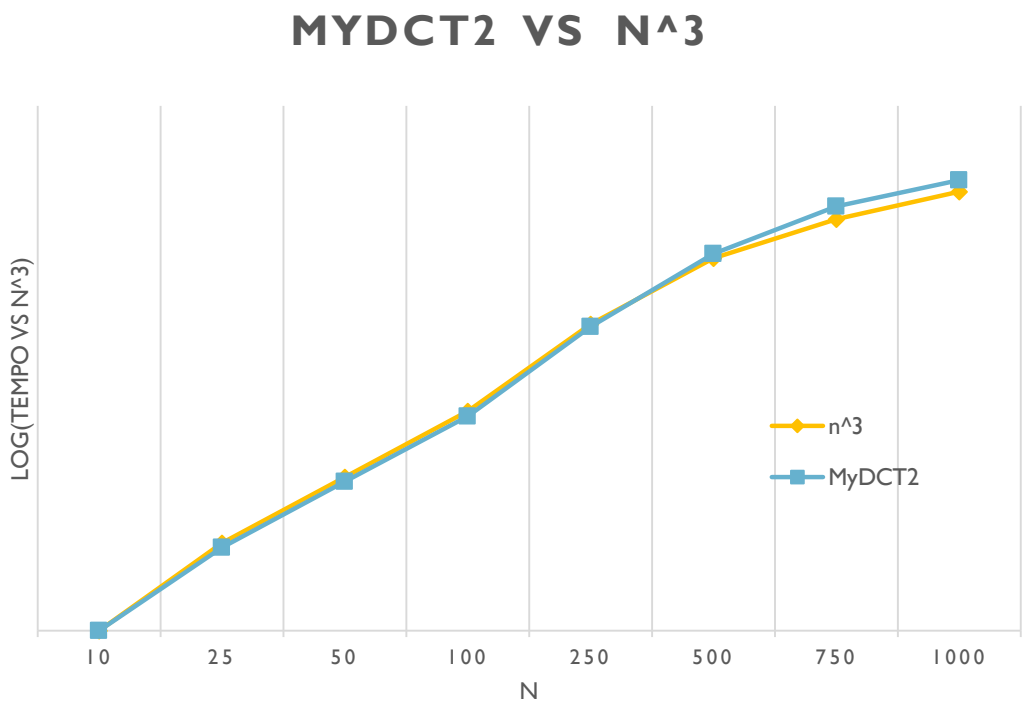
RISULTATI - I

- MyDCT2 è stata implementata applicando la DCT monodimensionale, che richiede tempo $O(n^2)$, su righe e colonne (n) dell'input per migliorare l'efficienza algoritmica da $O(n^4)$ a $O(n^3)$

- Lo stesso è stato fatto per la DCT2 di Scipy, in quanto la libreria conteneva solo la versione monodimensionale

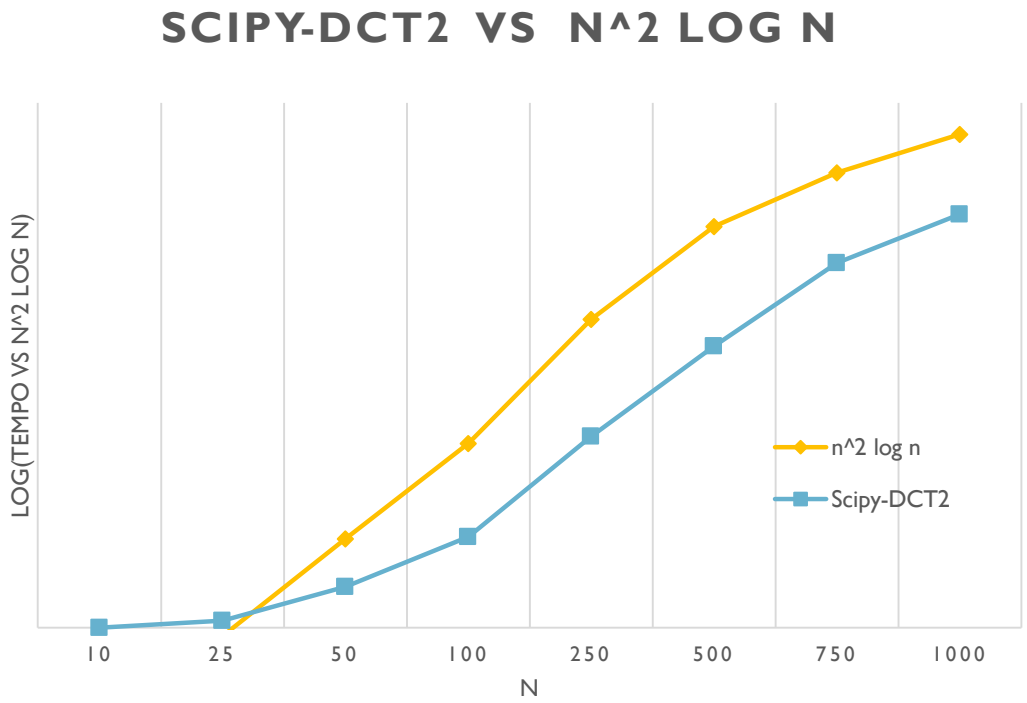
- La principale differenza tra le due implementazioni risiede nel fatto che Scipy utilizza FFT *(Fast Fourier Transform)* per eseguire DCT, abbassando così il costo teorico da $O(n^3)$ ad un costo approssimato $O(n^2 \log n)$

RISULTATI – 2



- Come possiamo notare dal grafico, i tempi raccolti nell'esecuzione della funzione MyDCT2 seguono lo stesso andamento di una curva $y = x^3$, coerentemente al suo andamento teorico $O(n^3)$
- Calcolando la cross-correlation [5] tra le due serie, infatti, essa risulta uguale a 0.9996

RISULTATI – 3



- Come possiamo notare dal grafico, inizialmente i tempi raccolti nell'esecuzione della funzione Scipy-DCT2 non seguono esattamente l'andamento teorico $O(n^2 \log n)$
- Quando n inizia a crescere, invece, gli andamenti tendono ad avvicinarsi e mantenere lo stesso trend
- Nonostante la discrepanza iniziale tra i valori, a cross-correlation tra le due serie risulta uguale a 0.9936

PARTE 2 – MANIPOLAZIONE DELLE FREQUENZE

IMMAGINI DIGITALI E FREQUENZE

- Un'immagine digitale è una matrice di pixel. Questa matrice deriva dalla digitalizzazione di un segnale analogico (l'insieme delle frequenze che compongono l'immagine)

- Attraverso l'utilizzo di trasformate (DCT) è possibile «effettuare il processo inverso», ovvero ottenere le frequenze (digitalizzate) che compongono l'immagine

- La modifica di queste frequenze ha ripercussioni sull'immagine che saranno visibili una volta effettuata la IDCT *«inverse DCT»*

OBIETTIVI

- Lo scopo principale di questa fase è quello di progettare e implementare un software che permetta di manipolare le frequenze di un'immagine a piacere in toni di grigio, in formato .bmp

- Le modifiche possibili avvengono attraverso la scelta di due parametri:
  - d : valore che influenza il numero di frequenze da alterare
    - Maggiore è il valore d, minore è il numero di frequenze selezionate
  - β : coefficiente moltiplicativo per le frequenze considerate (tramite d)
    - $\beta > 1$: le frequenze considerate risulteranno più forti (*«sharpening»* dell'immagine)
    - $\beta < 1$: le frequenze considerate risulteranno meno forti (*«smoothing»* dell'immagine)

INTERFACCIA GRAFICA

Valore del parametro d

Valore del parametro β

Immagine originale

Immagine elaborata

Tasto per la conversione:
- DCT2
- Manipolazione tramite i parametri
- IDCT2
- Visualizzazione

ESEMPIO 1 - SMOOTHING

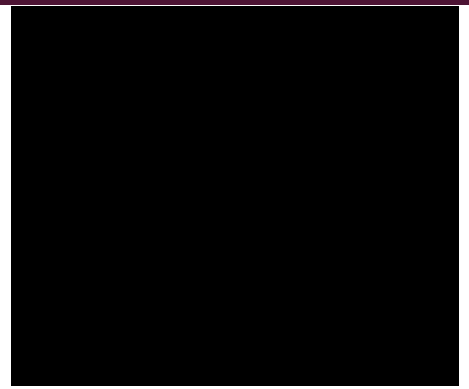
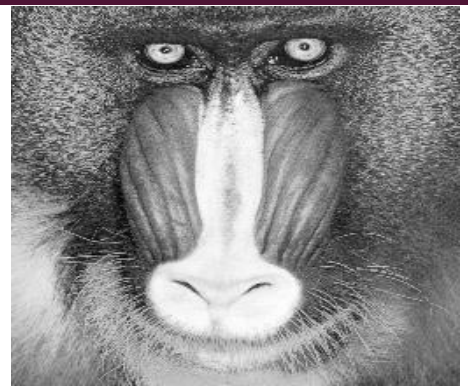
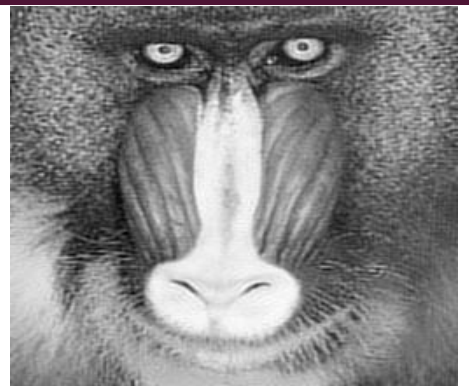
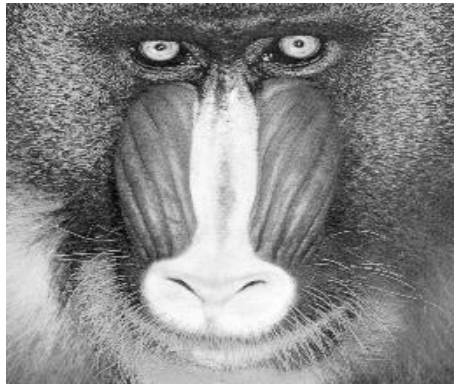
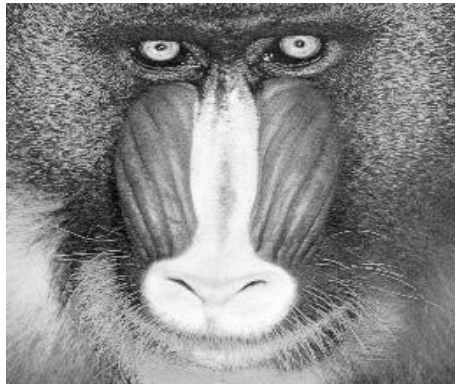
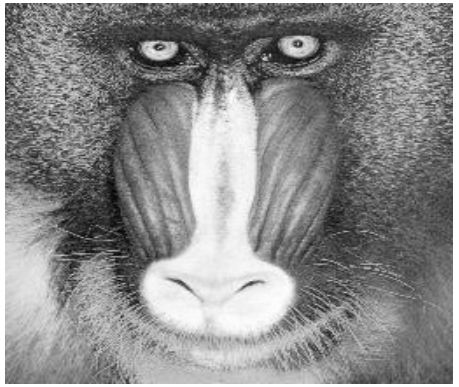
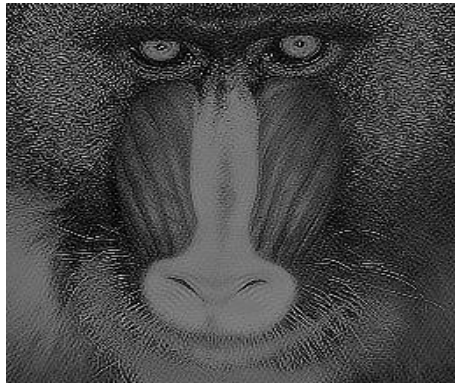
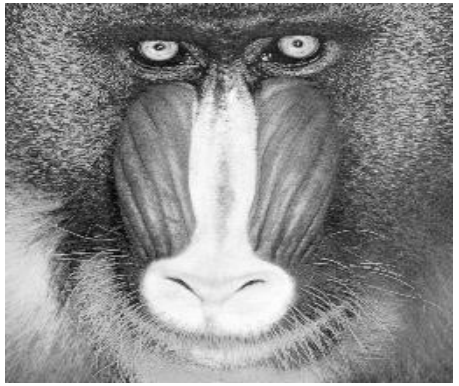
Immagine originale

Immagine con $d = 100, \beta = 0$

ESEMPIO 2 - SHARPENING

Immagine originale

Immagine con $d = 100, \beta = 10$

	D = 0	D = 200	D = 510
β = 0	 Beta = 0, d = 0	 Beta = 0, d = 200	 Beta = 0, d = 510
β = 1	 Beta = 1, d = 0	 Beta = 1, d = 200	 Beta = 1, d = 510
β = 5	 Beta = 5, d = 0	 Beta = 5, d = 200	 Beta = 5, d = 510

BIBLIOGRAFIA

- [1] "Discrete Cosine Transform« by Ahmed N., Natarajan T,. Rao K. R., IEEE *Transactions on Computers,* C–23(1), pp. 90–93, (January 1974)
- [2] A Fast Cosine Transform in One and Two Dimensions', by J. Makhoul, *IEEE Transactions on acoustics, speech and signal processing* vol. 28(1), pp. 27-34 (1980)
- [3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.dct.html#scipy.fftpack.dct, 30/05/2018
- [4] https://docs.scipy.org/doc/scipy/reference/fftpack.html, 30/05/2018
- [5] Weisstein Eric W., "Cross-Correlation." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/Cross-Correlation.html

GRAZIE DELL'ATTENZIONE