UNIVERSITÁ DEGLI STUDI MILANO-BICOCCA

Corso di Laurea Magistrale in Informatica

Metodi del Calcolo Scientifico

A.A. 2022/2023

Progetto 2

Repository del codice:

https://github.com/dadegrande99/ProgettoMCS

Report di:

Davide Grandesso 852078

Fabio Marini 851977

Sommario

1 Introduzione

La DCT (acronimo di Discrete Cosine Transform ~ Trasformata Cosinusoidale Discreta), è una tecnica matematica utilizzata principalmente per la compressione dei dati, infatti è ampiamente impiegata in applicazioni come la compressione di immagini JPEG. La DCT converte un segnale discreto, in un insieme di coefficienti che rappresentano le componenti di frequenza presenti nel segnale.

1.1 Obiettivi progetto

Lo scopo di questo progetto è di analizzare la DCT2 tramite la sua implementazione e la sua applicazione vera e propria per quanto riguarda la compressione di immagini in toni di grigio.

Lo svolgimento si divide principalmente in due parti:

- La prima consiste nell'implementare su linguaggio di programmazione la DCT sia nella sua versione monodimensionale che in quella bidimensionale, verificandone la sua correttezza e mettendo a confronto le sue tempistiche con quelle della sua versione fast (FFT) già implementata in un'altra libreria.
- La seconda invece consiste nell'applicare i passaggi necessari ad effettuare la conversione delle immagini sfruttando l'algoritmo della libreria a cui vengono passati come parametri un'immagine e dei coefficienti di compressione inseriti tramite un'interfaccia sviluppata ad hoc.

2 Costruzione libreria Discrete Cosine Transform

Di seguito saranno spiegati i passaggi eseguiti dal codice da noi scritto per implementare le funzioni DCT e IDCT.

2.1 DCT monodimensionale

La funzione DCT da noi implementata prende in input un array monodimensionale e restituisce i coefficienti DCT dei pixel come un nuovo array. La funzione, innanzitutto, crea un array di zeri della stessa dimensione di quello di input, dopodiché, iterando da 0 a N-1 (N dimensione dell'array di input), calcola il coefficiente alla posizione n con la formula specifica della DCT.

$$X_k = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{\mathrm{N}-1} x_n \cdot \cos\left(\frac{\pi}{\mathrm{N}} \cdot n \cdot \left(n + \frac{1}{2}\right)\right)$$

Dove X_k rappresenta il coefficiente DCT alla frequenza k, x_n rappresenta il valore del segnale di input al tempo n e N è la lunghezza del vettore dato in input. Per n=0 però, il coefficiente viene moltiplicato per un aggiuntivo $\sqrt{\frac{1}{2}}$ che garantisce una corretta normalizzazione dei coefficienti. Infine, viene restituito l'array dei coefficienti calcolati.

2.2 DCT2

Avendo già implementato la DCT monodimensionale, la DCT2 risulta di facile implementazione, la funzione prende in input una matrice e crea un'altra matrice di pari dimensioni che sarà quella corrispondente ai coefficienti DCT.

Essendo la DCT una trasformata ortogonale si ha la garanzia che le righe e le colonne delle DCT monodimensionali siano indipendente l'una dall'altra, perciò, quello che viene fatto è semplicemente iterare sulle colonne della matrice di input e applicare ad esse la DCT precedentemente implementata. Una volta finite queste iterazioni, viene ripetuto il processo sulle righe della matrice dei coefficienti. Infine, tale matrice sarà restituita dalla funzione.

2.3 Inverse DCT monodimensionale

La funzione IDCT serve per ripristinare il vettore originale a partire dalla sua rappresentazione nel dominio delle frequenze, ovvero nel dominio di arrivo della funzione DCT. A livello di codice quello che viene fatto una volta creato un array di zeri di lunghezza pari a quello di input è applicare la seguente formula iterando sull'array creato.

$$x_n = \sqrt{\frac{2}{N}} \cdot \left(\frac{1}{\sqrt{2}} \cdot X_0 + \sum_{n=1}^{N-1} X_k \cdot \cos\left(\frac{\pi}{N} \cdot \left(n + \frac{1}{2} \right) \cdot n \right) \right)$$

Dove X_k rappresenta il coefficiente DCT alla frequenza k dato in input, x_n rappresenta il valore del segnale ricostruito al tempo n e N è la lunghezza del vettore dato in input.

2.4 Inverse DCT2

Come per la DCT2 è possibile calcolare l'IDCT2 applicando la sua versione monodimensionale prima alle righe e successivamente alle colonne (l'opposto di DCT2), infine verrà restituita dalla funzione la matrice che contiene i coefficienti riportati al dominio iniziale. Questo ragionamento, come per la DCT2, è possibile applicarlo grazie all'ortogonalità che ci permette di applicare la IDCT in maniera indipendente tra righe e colonne.

3 Controlli

Dopo aver implementato l'algoritmo della DCT e DCT2 su Python, sono stati fatti essenzialmente due controlli su di essi:

- **Controlli tempistici**: in questa fase sono stati confrontati i tempi di esecuzione della DCT2 costruita nel progetto con quella di una libreria già presente in Python che viene costruita nella versione veloce (FFT ~ Fast Fourier Transform).
- **Verifica di correttezza:** in questa fase è stato controllato che gli algoritmi implementati eseguissero dei casi test in modo corretto. In particolare, si richiede per la DCT2 che data in input una matrice 8×8 questa restituisca i risultati attesi, stessa cosa per la DCT con la sola differenza che al posto di avere in input una matrice abbiamo un vettore di 8 elementi

3.1 Confronto con FFT

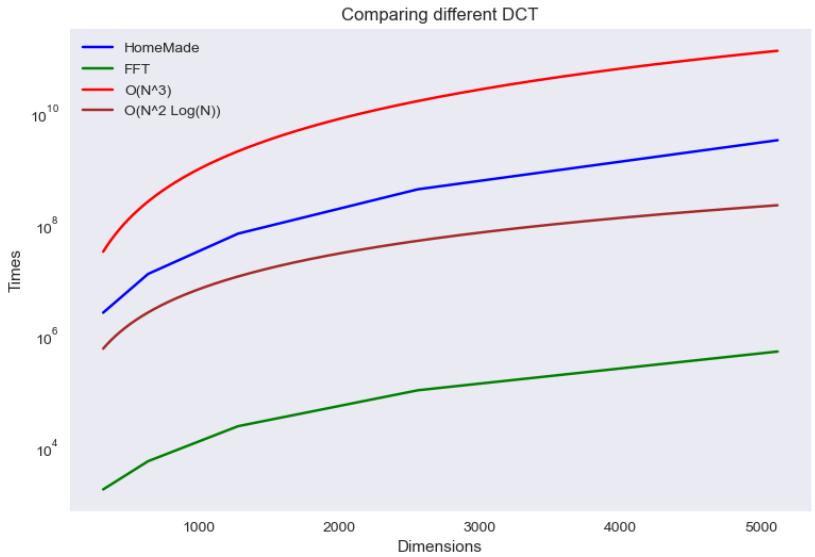
La DCT2 costruita nel progetto viene confrontata con quella presente nella libreria **cv2**.

Si cerca di dimostrare che i tempi per la DCT2 fatta in casa siano proporzionali a N^3, mentre per la versione fast di cv2 siano proporzionali a $N^2 \log(N)$.

Il confronto è stato fatto misurando i tempi ottenuti dai due metodi su 5 matrici inizializzate con valori interi casuali compresi tra 0 e 255, delle seguenti dimensioni:

$$[320 \times 320 \quad 640 \times 640 \quad 1280 \times 1280 \quad 2560 \times 2560 \quad 5120 \times 5120]$$

Per ognuna di esse vengono applicati i metodi prima citati calcolando così le tempistiche in modo da poterle confrontare. Dopo aver acquisito i dati sulle tempistiche essi sono stati messi a confronto nel grafico seguente. Il diagramma è su scala logaritmica nelle ascisse (dimensioni delle matrici) e ordinate (tempistiche).



Come è possibile notare le tempistiche della DCT2 "fatta in casa" ha dei tempi al di sotto della curva teorica di $O(N^3)$ e quelle della libreria costruita nella versione fast si trova anche questa sotto la curva teorica di $O(N^2 \log(N))$. Le tempistiche di quest'ultimo sono anche influenzate dal fatto che l'algoritmo FFT, a differenza di quello custom, sfrutta alla base un linguaggio di programmazione nettamente più veloce a python.

3.2 Verifica correttezza

3.2.1 Correttezza DCT:

- Array dato in input:

$$[231 \quad 32 \quad 233 \quad 161 \quad 24 \quad 71 \quad 140 \quad 245]$$

- Array atteso:

$$[4.01e+02 \quad 6.60e+00 \quad 1.09e+02 \quad -1.12e+02 \quad 6.54e+01 \quad 1.21e+02 \quad 1.16e+02 \quad 2.88e+01]$$

- Array calcolato:

```
[ 401.9902051     6.60001991  109.16736544 -112.78557857   65.40737726  121.83139804  116.65648855   28.80040722]
```

3.2.2 Correttezza DCT2:

- Matrice data in input:

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

- Matrice attesa:

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

- Matrice calcolata:

```
[[ 1.11875000e+03  4.40221926e+01  7.59190503e+01 -1.38572411e+02   3.50000000e+00  1.22078055e+02  1.95043868e+02 -1.01604906e+02]
 [ 7.71900790e+01  1.14868206e+02 -2.18014421e+01  4.13641351e+01   8.77720598e+00  9.90829620e+01  1.38171516e+02  1.09092795e+01]
 [ 4.48351537e+01 -6.27524464e+01  1.11614114e+02 -7.63789658e+01   1.24422160e+02  9.55984194e+01 -3.98287969e+01  5.85237670e+01]
 [-6.99836647e+01 -4.02408945e+01 -2.34970508e+01 -7.67320594e+01   2.66457750e+01 -3.68328290e+01  6.61891485e+01  1.25429731e+02]
 [-1.09000000e+02 -4.33430857e+01 -5.55436908e+01  8.17347083e+00   3.02500000e+01 -2.86602437e+01  2.44149822e+00 -9.41437025e+01]
 [-5.38783591e+00  5.66345009e+01  1.73021519e+02 -3.54234494e+01   3.23878249e+01  3.34576728e+01 -5.81167864e+01  1.90225615e+01]
 [ 7.88439693e+01 -6.45924096e+01  1.18671203e+02 -1.50904840e+01  -1.37316928e+02 -3.06196663e+01 -1.05114114e+02  3.98130497e+01]
 [ 1.97882438e+01 -7.81813409e+01  9.72311860e-01 -7.23464180e+01  -2.15781633e+01  8.12999035e+01  6.37103782e+01  5.90618071e+00]]
```

3.2.3 Risultati

Come possiamo notare, sia per quanto riguarda entrambi gli algoritmi calcoliamo dei risultati al pari di quelli attesi con la dovuta approssimazione, possiamo quindi dire che la verifica dei due casi test è andata a buon fine.

Per entrambi i test sono state anche testate anche le funzioni inverse, rispettivamente IDCT e IDCT2, verificando che dalla matrice calcolata la funzione inversa calcola quella data in input, facendo poi i dovuti confronti è stato possibile verificare che anche le funzioni inverse effettuano correttamente i test.

4 Compressione delle immagini tramite DCT

Il campo in cui viene applicata maggiormente la DCT è quello della compressione delle immagini, è infatti usato come standard per la compressione in JPEG.

È stata quindi costruita una applicazione che permette di comprimere un'immagine in toni di grigio in base a dei coefficienti sfruttando la DCT, sia l'immagine che i coefficienti vengono inseriti dall'utente.

L'applicazione permette di selezionare l'algoritmo di DCT da usare per i calcoli. Dato che le immagini possono essere anche molto grandi, con la libreria custom si rischia di avere dei tempi di computazione eccessivamente alti, per ovviare a ciò è possibile scegliere la versione Fast della DCT presenta nella libreria cv2.

4.1 Processamento

La compressione verrà fatta a partire da un'immagine in formato **.bmp** (bitmap), questo perché i file di tipo bitmap memorizzano l'immagine pixel per pixel senza compressione ed è quindi il punto di partenza per effettuare la procedura di compressione.

La prima procedura eseguita per effettuare il processamento è quella di suddividere l'immagine in dei blocchi quadrati di pixel a seconda del parametro **F** specificato nell'interfaccia, esso indica l'ampiezza dei blocchi in cui si andrà ad effettuare la DCT2. Dato che non tutte le immagini possono essere formate da blocchi quadrati FXF questo porta a dover scartare parte dell'immagine in modo da poter effettuare la compressione correttamente (senza eccessi).

```
image = cv2.imread(image_path, 0)
height, width = image.shape
bw = width // F  # block width
bh = height // F  # block height
image = image[:bh*F, :bw*F]

blocks = np.split(image, bh, axis=0)
blocks = [np.split(block, bw, axis=1) for block in blocks]
blocks = np.array(blocks)
```

Dopodiché, per ognuno dei blocchi si applica la DCT2 (nella versione scelta dall'utente) e da ogni blocco della matrice restituita si eliminano le frequenze a destra della diagonale individuata grazie al parametro **d** specificato nell'interfaccia.

```
# Applica la DCT2
if fast:
    c = cv2.dct(np.float32(block))
else:
    c = custom_dct2(np.float32(block))

# Taglia le frequenze
c = threshold_cutoff(c, d)
```

La funzione per il taglio delle frequenze è definita come segue:

```python
def threshold_cutoff(coefficients, threshold):
    size = coefficients.shape[0]
    for i in range(size):
        for j in range(size):
            if i + j >= threshold:
                coefficients[i, j] = 0
    return coefficients
```

Dopo aver effettuato questa fase di eliminazione dell'informazione si esegue la DCT inversa (o IDCT) per ricostruire l'immagine in maniera tale che sia una rappresentazione fedele dell'originale, seppur con una certa quantità di dati persa per la compressione. Se non si applicasse questo processo, l'immagine compressa sarebbe rappresentata solo dai coefficienti DCT ridotti, senza la rappresentazione dei pixel necessaria per visualizzare l'immagine. Per ricondurci ai valori ammissibili dei pixel però è necessario anche arrotondare all'intero più vicino i valori ottenuti e portare a 0 i valori negativi e a 255 i valori maggiori di questo numero.

```python
# Applica l'inversa della DCT2
if fast:
    ff = cv2.idct(c)
else:
    ff = custom_idct2(c)

# Arrotonda e normalizza i valori
ff = np.round(ff)
ff = np.clip(ff, 0, 255)
```

Fatte tutte queste operazioni è possibile assegnare ai singoli blocchi il risultato ottenuto

```python
# Aggiorna il blocco ricostruito
blocks[i, j] = ff
```

e successivamente, finite le iterazioni per ogni blocco ottenuto dalla matrice, ricostruire l'immagine con i nuovi blocchi e restituirla nel formato necessario per poterla visualizzare.

```python
# Ricompone l'immagine a partire dai blocchi
reconstructed_image = np.block(
    [[block for block in row] for row in blocks])

# Converte l'immagine in formato byte
reconstructed_image = reconstructed_image.astype(np.uint8)
return reconstructed_image
```

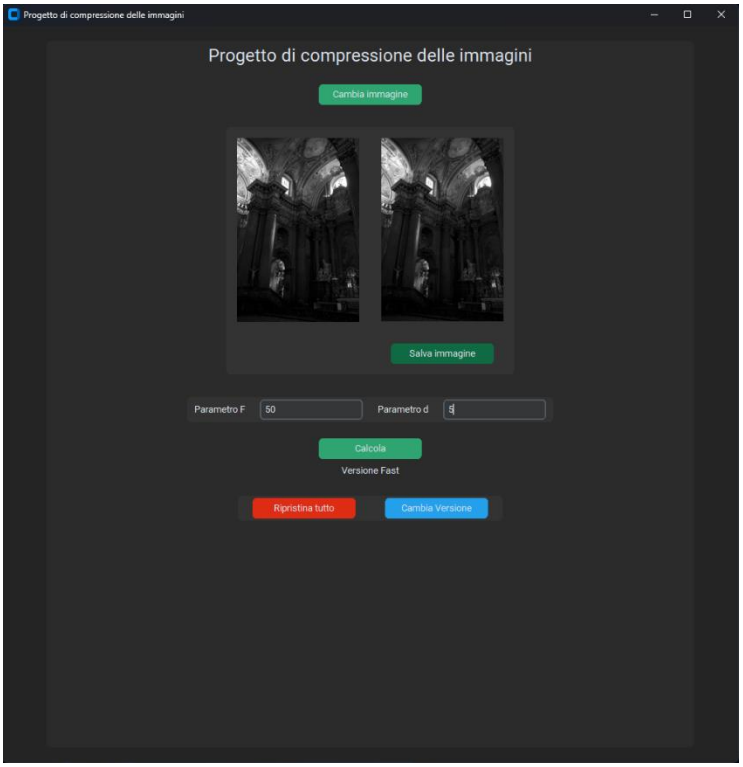
4.2 Interfaccia

L'interfaccia dell'applicazione è stata costruita con la libreria CustomTkinter. Da essa è possibile scegliere dal filesystem un'immagine .bmp in toni di grigio tramite il bottone verde in alto "**Carica immagine**", una volta inserita l'immagine questa viene visualizzata a schermo, l'applicazione permette di cambiare l'immagine scelta tramite il bottone "**Cambia Immagine**". Al di sotto dell'immagine scelta si trova la zona in cui inserire i parametri richiesti **F** (ampiezza dei macro-blocchi) e **d** (soglia di taglio delle frequenze) e richiamare il calcolo di compressione tramite il bottone "**Calcola**", la computazione avviene solo quando tutti i dati sono corretti, altrimenti vengono stampati gli errori che hanno impedito la compressione.

Una volta calcolata l'immagine questa viene mostrata alla destra di quella originale, così da poter osservare meglio le differenze tra le due. Al di sotto dell'immagine compressa è presente un bottone che permette di salvarla sul proprio pc.

È possibile decidere anche di ripulire tutti parametri inseriti tramite il bottone rosso "**Ripristina tutto**". È possibile, inoltre, cambiare la versione di DCT da usare per il calcolo della compressione tramite il bottone blu "**Cambia versione**". Sotto al bottone "**Calcola**" viene presentata la versione che verrà usata, di default è selezionata la versione fast.

Di seguito viene riportato uno screenshot di esempio su come si presenta l'interfaccia:



4.3 Influenza dei parametri F e d sulla compressione delle immagini

Nella compressione delle immagini con la DCT (Discrete Cosine Transform), i parametri di grandezza dei macro-blocchi (F) e quello della soglia di taglio delle frequenze (d) influiscono sia sull'efficienza a livello computazionale della compressione che sulla qualità dell'immagine ricostruita.

4.3.1 Parametro F

Il parametro F è un intero che rappresenta l'ampiezza delle finestrelle di pixel in cui verrà suddivisa l'immagine a blocchi ai quali verrà applicata la DCT2 ad ognuno di essi.

Maggiore è il valore di F e più grandi saranno i blocchi di pixel utilizzati per la trasformazione, l'utilizzo di macro-blocchi più grandi può portare ad una maggiore compressione, in quanto la correlazione tra i pixel all'interno di un blocco più grande può essere sfruttata per eliminare le ridondanze. Tuttavia, l'aumento della dimensione dei macro-blocchi può comportare anche una minore precisione nella rappresentazione delle frequenze.

Se invece F è più piccolo allora i blocchi su cui verrà applicata la DCT saranno di conseguenza più piccoli comportando maggiori vantaggi per quanto riguarda i dettagli delle immagini a discapito però di ottenere un file compresso di dimensioni maggiori

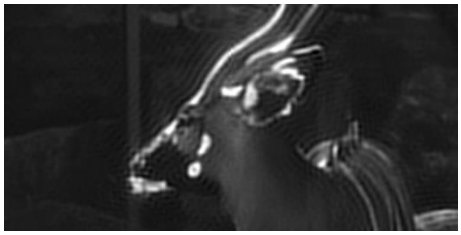
4.3.2 Parametro d

Il parametro d è un intero compreso tra 0 e 2F −2, esso rappresenta la soglia di taglio delle frequenze

Riducendo il valore di d, si tagliano più coefficienti DCT, riducendo così la quantità di informazione memorizzata e, quindi, la dimensione del file compresso. Tuttavia, una riduzione eccessiva di d può portare a una perdita significativa di dettagli nell'immagine ricostruita.

In base al parametro delle frequenze di taglio, è possibile che si manifesti il fenomeno di Gibbs che si mostra come una serie di "scalini" o di anelli scuri o luminosi che circondano gli oggetti o i dettagli nell'immagine. Questo è legato alle caratteristiche matematiche delle trasformate utilizzate durante la compressione delle immagini. Queste trasformate separano l'immagine in frequenze diverse e, durante la successiva ricostruzione, possono verificarsi errori di approssimazione o distorsioni che portano alla comparsa degli anelli di Gibbs.

Di seguito riportiamo un caso di esempio che mostra in modo significativo l'effetto del parametro d. A sinistra abbiamo l'immagine originale mentre le due successive sono state compresse con il parametro F pari a 500, il parametro d pari a 80 per l'immagine in mezzo e 230 per quella a destra. Possiamo notare come per l'immagine in mezzo la qualità dei dettagli sia molto bassa e l'effetto di Gibbs sia facilmente visibile dalle molteplici e spesse onde attorno che circondano il cervo, mentre in quella più a destra la qualità dei dettagli è decisamente più elevata e l'effetto di Gibbs si nota a malapena sopra la testa del cervo.


Anche in questo esempio abbiamo a sinistra l’immagine originale e a destra quella compressa con parametro F pari a 200 e d pari a 5. In questo caso l’effetto di Gibbs si mostra sotto forma di scalini.

4.3.3 Riflessioni

In sintesi, l'uso di macroblocchi più grandi (F) può aumentare l'efficienza di compressione, ma potrebbe influire sulla qualità dell'immagine. La soglia di taglio delle frequenze (d) determina il compromesso tra dimensione del file compresso e qualità dell'immagine ricostruita, con valori più bassi che portano a una maggiore compressione ma anche a una potenziale perdita di dettagli. È importante trovare un equilibrio tra questi parametri per ottenere il risultato desiderato nella compressione delle immagini con la DCT.

5 Conclusioni

Grazie a questo progetto siamo riusciti ad analizzare meglio la logica della compressione delle immagini, con particolare riguardo verso l'algoritmo di DCT studiandone sia l'implementazione pratica che la teoria su cui si basa. Lo scopo di questo progetto era poi quello di verificare la correttezza di tale implementazione con anche dei controlli tempistici e costruire un'applicazione in modo da poter provare a effettuare la compressione di un'immagine tramite interfaccia grafica.

Studiando meglio il procedimento della compressione delle immagini ed effettuando i dovuti test, è stato possibile venire a conoscenza di come la scelta dei parametri di compressione sia fondamentale per avere un giusto compromesso tra qualità dei dettagli e spazio occupato in memoria dell'immagine.

In conclusione, possiamo dire che grazie a questo progetto è stato possibile studiare meglio la compressione delle immagini dal punto di vista del calcolo matematico e come realizzare una applicazione matematica per un linguaggio di programmazione.