

DCT e JPEG

Metodi del Calcolo Scientifico 2020/2021 - *Progetto2*

- 829470 Di Lauro Federica
- 829827 Cozzi Davide
- 829835 De Rosa Gabriele

Parte 1: introduzione

Obiettivo: effettuare un confronto tra

- una implementazione naive della DCT vista a lezione
- la DCT presente all'interno di una libreria open-source, sfruttando la FFT

Codice:

- Jupyter Notebook per esecuzione: [Colab](#)

Linguaggi e ambienti utilizzati



Google Colab



Python3, Scipy

Custom DCT2

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

```
#definizione dct
def my_dct(f):
    c = np.zeros(f.size)
    N = f.size
    for k in range(N):
        sum = 0
        a = math.sqrt(1. / N) if k == 0 else math.sqrt(2. / N)
        for i in range(N):
            sum += f[i] * math.cos((k * math.pi * (2 * i + 1)) / (2 * N))
        c[k] = a * sum
    return c

#definizione dct2, chiamando la dct su righe e colonne
def my_dct2(m):
    c = np.zeros(m.shape)
    c = np.apply_along_axis(my_dct, axis=1, arr=m)
    c = np.apply_along_axis(my_dct, axis=0, arr=c)
    return c
```

Python: scipy.dct

Per il confronto abbiamo scelto di utilizzare la libreria [scipy.dct](#).

Tale libreria offre un wrapper della libreria [pocketfft](#), basata su [FFTPACK](#), scritta in linguaggio C e molto performante.

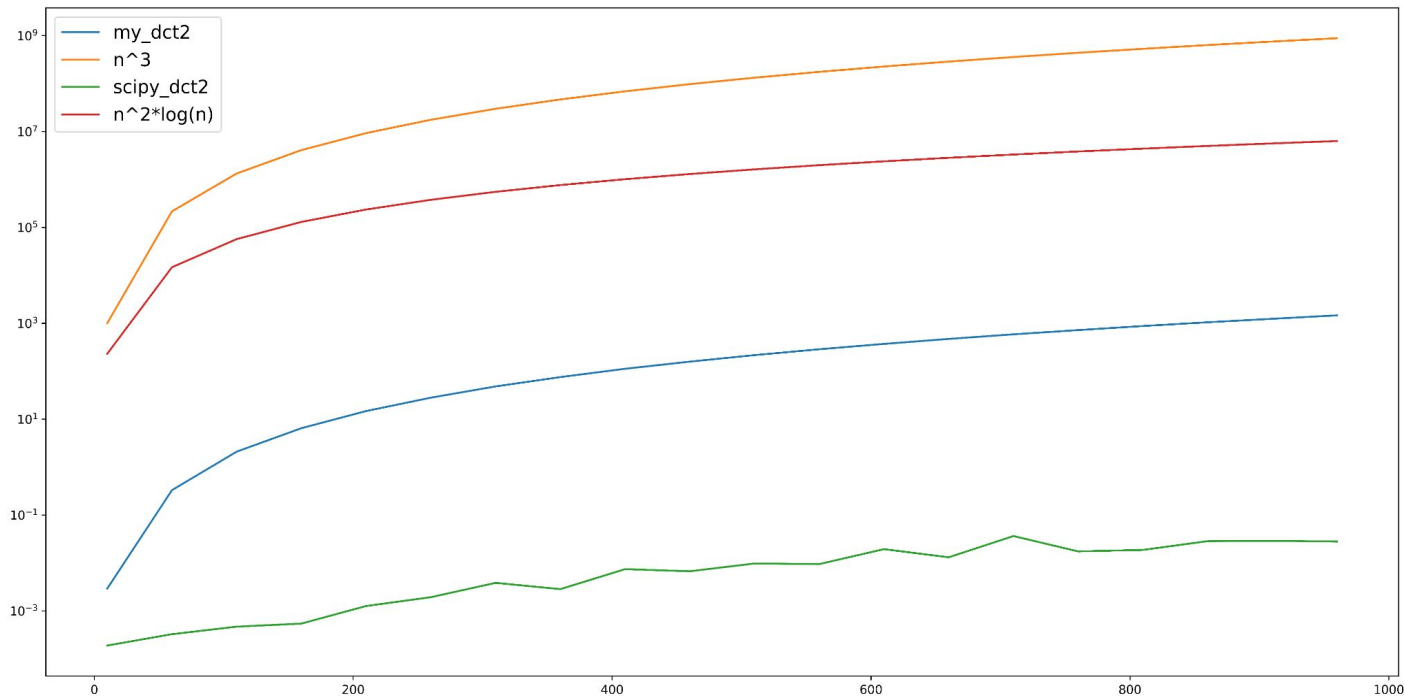
Scipy fornisce l'implementazione della DCT sia monodimensionale che n-dimensionale.



```
from scipy.fft import dctn, idctn

scipy_dct2 = dctn(mat, type = 2, norm = 'ortho')
mat = idctn(scipy_dct2, type = 2, norm = 'ortho')
```

Confronto tempi



Sono state generate 20 matrici con numeri casuali, partendo da una dimensione 10 e arrivando a una dimensione 960.

Parte 2: compressione JPEG

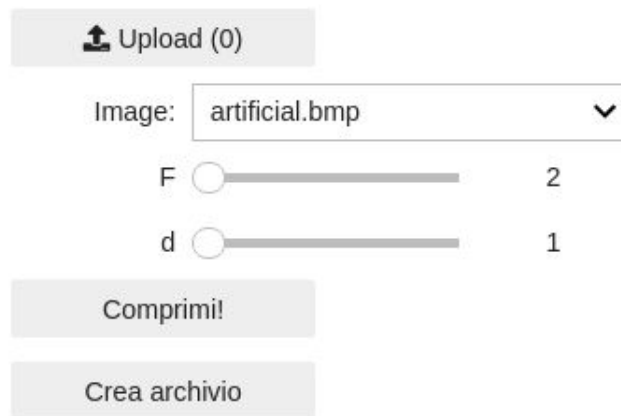
Obiettivo: effettuare la compressione delle immagini sfruttando la DCT2 e il taglio delle frequenze alte

Codice:

- Jupyter Notebook per esecuzione: [Colab](#)

Immagini compresse:

- Link Drive: [compressed_images](#)



A web interface for JPEG compression. It features an 'Upload (0)' button with an upload icon. Below it is a dropdown menu labeled 'Image:' showing 'artificial.bmp'. There are two sliders: 'F' with a value of 2 and 'd' with a value of 1. At the bottom are two buttons: 'Comprimi!' and 'Crea archivio'.

Upload (0)

Image: artificial.bmp

F 2

d 1

Comprimi!

Crea archivio

Algoritmo di compressione

```
def my_dct_compression(image, F, d):
    h = image.shape[0]
    w = image.shape[1]

    #andiamo a tagliare i pixel in eccesso rispetto al multiplo di F
    if h%F != 0:
        h = int(h/F) * F
    if w%F != 0:
        w = int(w/F) * F
    image_to_compress = image[0:h, 0:w]

    #nuovo array per l'immagine compressa
    compressed_image = np.zeros((h, w))

    # cicliamo sull'immagine a step di F
    for x in range(0,h,F):
        for y in range(0,w,F):
            cell = image_to_compress[x:x+F, y:y+F] # ampiezza e larghezza della cella = F
            cell = dctn(cell, type = 2, norm = 'ortho') # DCT
            # cancelliamo le frequenze sotto la diagonale in base a d
            for i in range(0,F):
                for j in range(0,F):
                    if i+j >= d:
                        cell[i,j] = 0

            # calcoliamo l'inversa della DCT del blocchetto
            cell = idctn(cell, type = 2, norm = 'ortho')
            #arrotondiamo all'intero più vicino, 0 per i valori negativi e 255 per i valori più grandi di 255
            for i in range(0,F):
                for j in range(0,F):
                    value = np.round(cell[i,j])
                    if value < 0:
                        value = 0
                    elif value > 255:
                        value = 255
                    cell[i,j] = value
            compressed_image[x:x+F, y:y+F] = cell

    return compressed_image
```

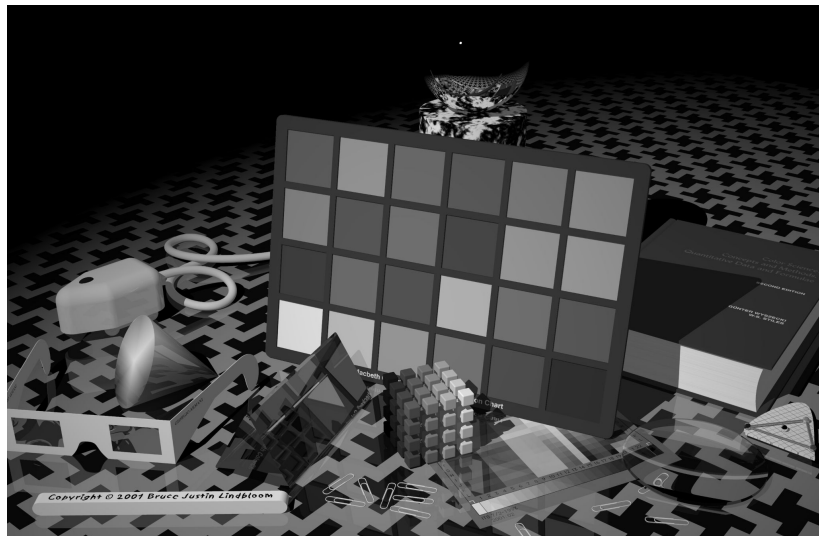

Imagine: deer ($F = 400$, $d = 30$)



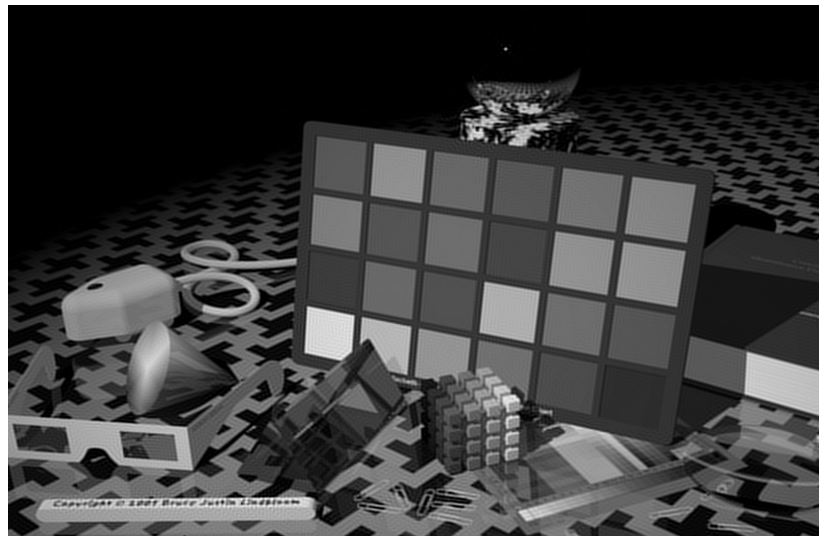
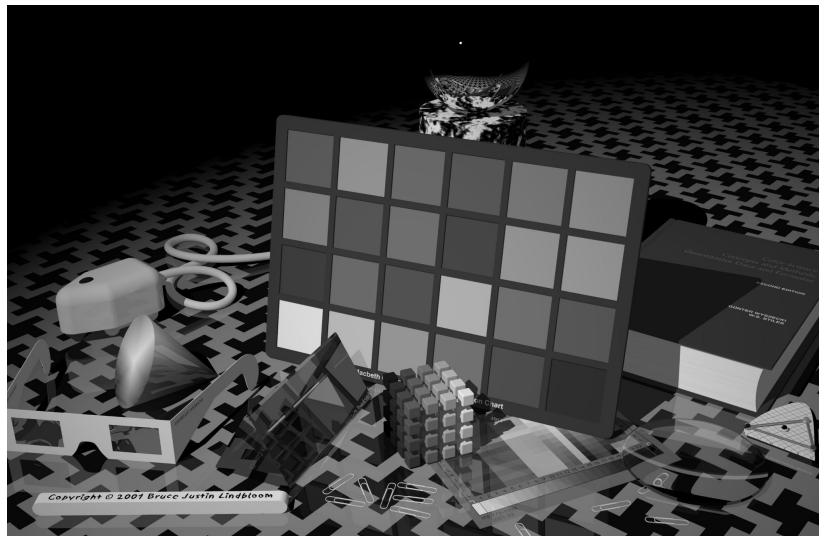
Imagine: deer ($F = 400$, $d = 20$)



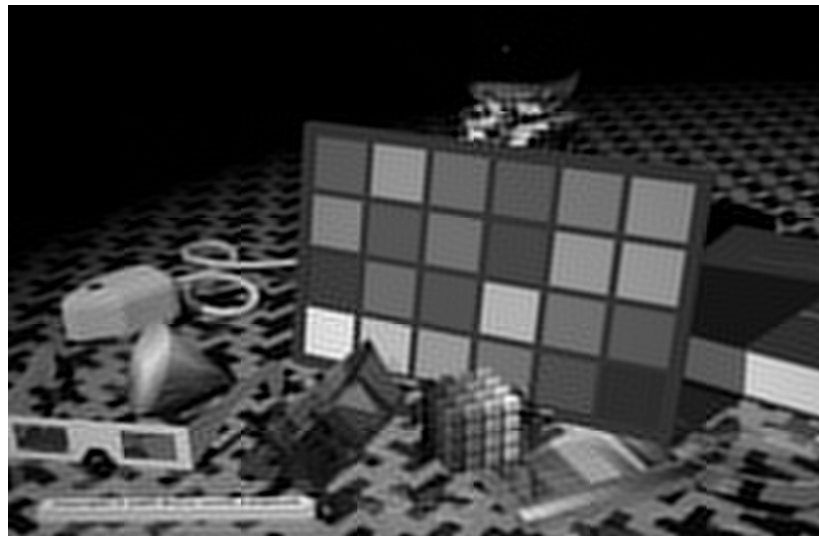
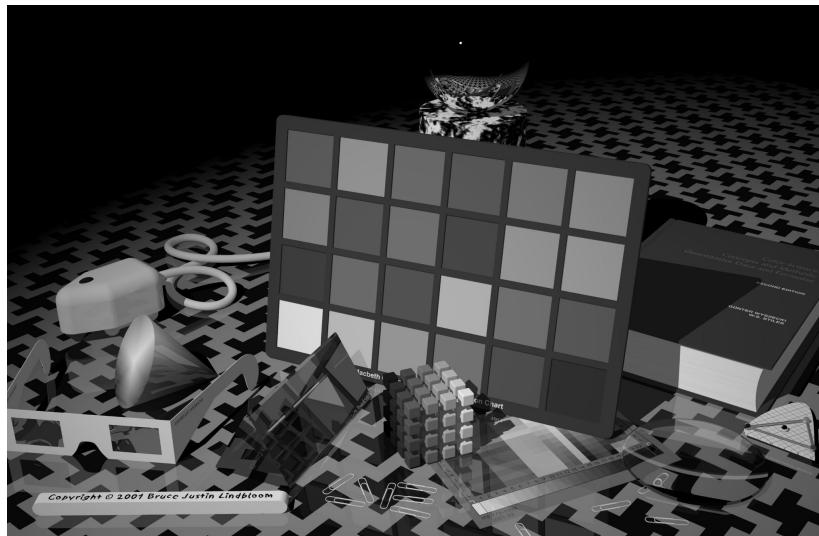
Imagine: artificial ($F = 400$, $d = 500$)



Imagine: artificial ($F = 400$, $d = 70$)



Imagine: artificial ($F = 400$, $d = 30$)



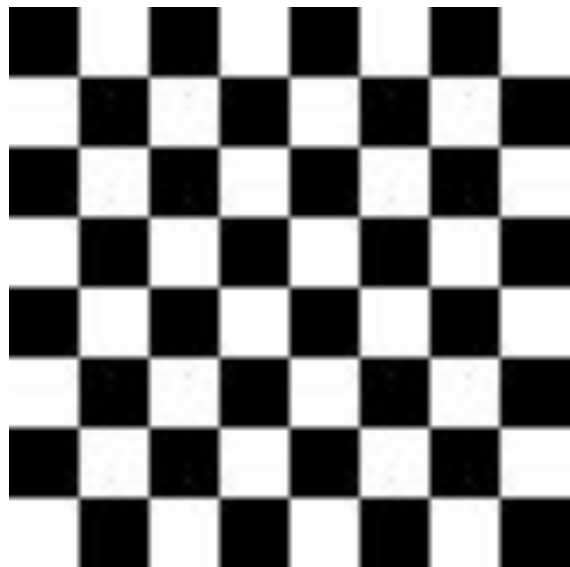
Imagine: hdr ($F = 400$, $d = 399$)



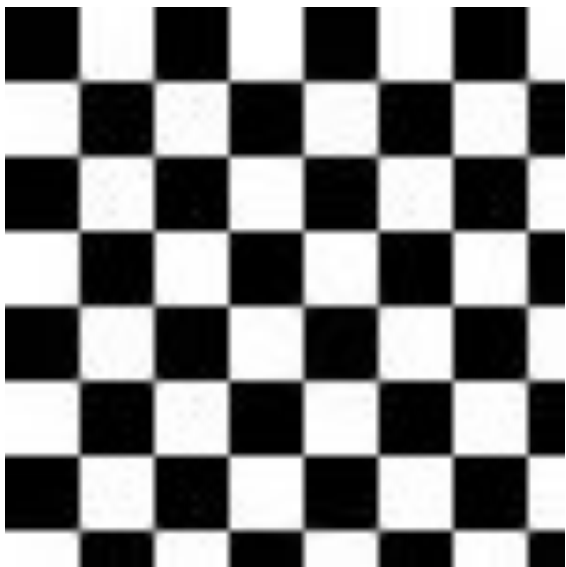
Imagine: hdr ($F = 400$, $d = 35$)



Imagine: 80x80 ($F = 15, d = 20$ / $F = 15, d = 5$)



ORIGINAL

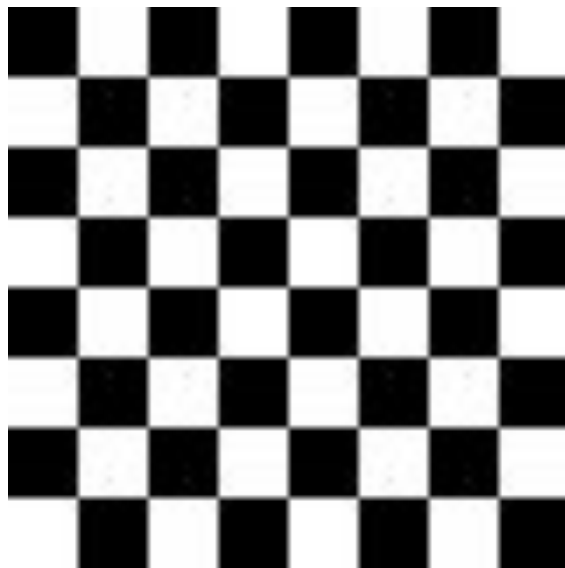


$F=15, D=20$



$F=15, D=5$

Imagine: 80x80 ($F = 10$, $d = 1$)



Imagine: jump2 ($F = 8$, $d = 3$)

