Università degli Studi di Perugia

Dipartimento di Matematica e Informatica

Report di Laboratorio – Tecniche di Acquisizione Dati

“Trasformata 2d di Immagini”

Studenti

Filippo Notari
Nicolò Tittarelli
Leonardo Nicoletta

Anno Accademico 2023-2024

Sommario

Introduzione

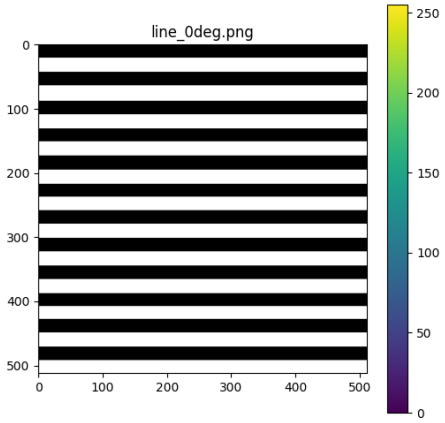
In questa relazione andremo ad illustrare l'esercitazione relativo all'uso della Trasformata di Fourier su delle immagini.
L'esercitazione è svolta completamente in python, utilizzando delle librerie per fare trasformate, grafici, funzioni matematiche e gestione immagini che sono rispettivamente: scipy, matplotlib, numpy e PIL.
L'obbiettivo del primo punto è quello di aprire delle immagini e isolando tutti i canali a parte uno, utilizzarlo per fare la sua trasformata 2D e visualizzare lo spettro di potenza.
L'obbiettivo del secondo punto è quello di aprire un'immagine, isolare tutti i canali a parte uno, utilizzarlo per vederne lo spettro e creare una maschera.
L'obbiettivo del terzo punto è quello di aprire un'immagine, isolare tutti i canali a parte uno, utilizzarlo per vederne lo spettro, creare una maschera e applicare due filtri: mediano e gaussiano.

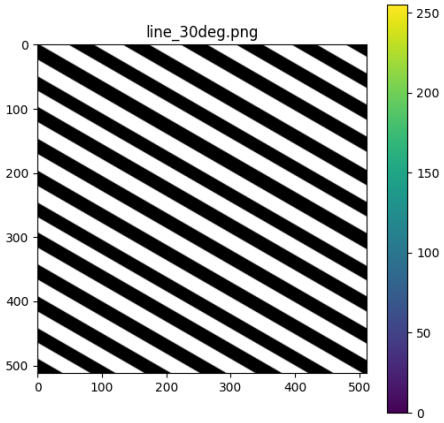
Capitolo 1. Corrispondenza tra immagini e la loro trasformata di Fourier 2D

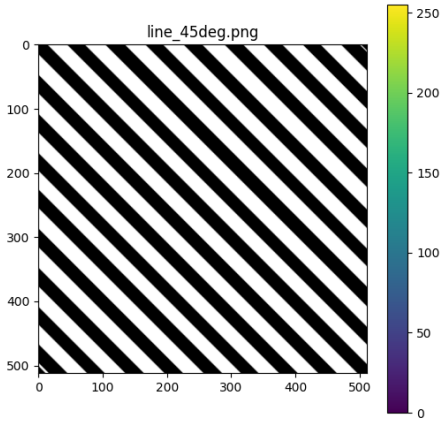
1.1 Descrizione

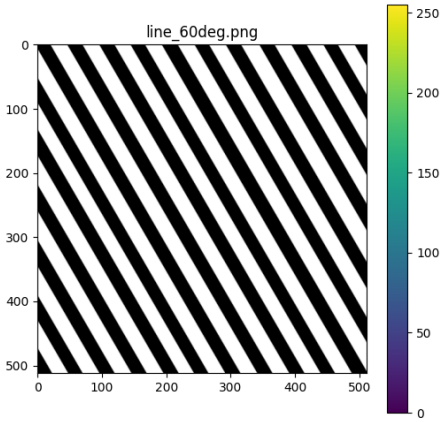
L’obbiettivo è quello di usare 5 immagini (line_0deg.png, line_30deg.png, line_45deg.png, line_60deg.png, line_90deg.png) da cui bisogna isolare i loro canali per usarne uno solo, calcolare la loro trasformata 2d e visualizzarne lo spettro di potenza

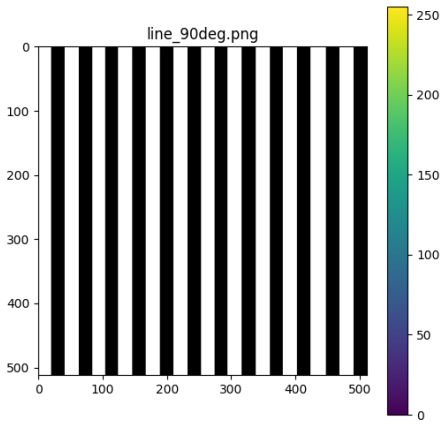
1.2 Immagini originali











1.3 Librerie utilizzate

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy import signal, fft
from scipy.ndimage import gaussian_filter, median_filter
from PIL import Image
```

Per questa esercitazione abbiamo usato una serie di librerie per poterla riusce a svolgere, tra cui: numpy per utilizzare delle funzioni matematiche di base, matplotlib per poter creare i grafici, scipy per poter calcolare la traformata e i filtri e infine PIL per poter elaborare le immagini.

1.4 Isolare un canale

```
path_to_img = "line_0deg.png"  #path_to_img = immagine

img = Image.open(path_to_img)
img=img.convert('RGB')
width, height = img.size

#Isolo i canali
red_channel, green_channel, blue_channel = img.split()
```

Come prima cosa abbiamo aperto un'immagine tramite PIL, che poi abbiamo convertito in RGB per essere sicuri di lavorare su 3 canali, dato che alcune immagini erano RGBA quindi 4 canali, mentre altre ne avevano uno solo.
Una volta convertita abbiamo salvato le sue misure, altezza e larghezza, che ci serviranno più avanti.
Arrivati a questo punto possiamo isolare i canali facendo semplicemente img.split(), il quale ci ritornerà i 3 canali di cui è composto, così possiamo salvarli e utilizzarli come vogliamo, nel nostro caso useremo nel corso dell'esercitazione il canale rosso.
Tutta questa prima esercitazione è dentro un ciclo che permette in automatico di aprire una alla volta tutte le immagini e elaborarle.

```
imgs = ["line_0deg.png", "line_30deg.png", "line_60deg.png", "line_90deg.png", "line_45deg.png"]

# Itera attraverso ogni immagine nell'elenco
for immagine in imgs:
        path_to_img = immagine #path_to_img = "line_0deg.png"
```

1.5 Trasformata 2D

```
#Trasformata 2d
fft2 = scipy.fft.fft2(red_channel)
#Shift
fft_result_shifted = scipy.fft.fftshift(fft2)

# Magnitude spectrum
magnitude_spectrum = np.log(abs(fft_result_shifted)+1)
```

Utilizzando il canale rosso che abbiamo isolato nello precedente punto, abbiamo calcolato la sua trasformata 2d utilizzando la funzione fft2 di scipy.

- **scipy.fft.fft2**: Questa funzione esegue una trasformata di Fourier bidimensionale sull'array red_channel, che rappresenta il canale rosso di un'immagine.
- **red_channel**: È un array bidimensionale che contiene i valori di intensità del canale rosso di un'immagine.

La trasformata di Fourier bidimensionale converte l'immagine dallo spazio dei pixel allo spazio delle frequenze. Ogni punto nell'output rappresenta una frequenza specifica e la sua ampiezza.

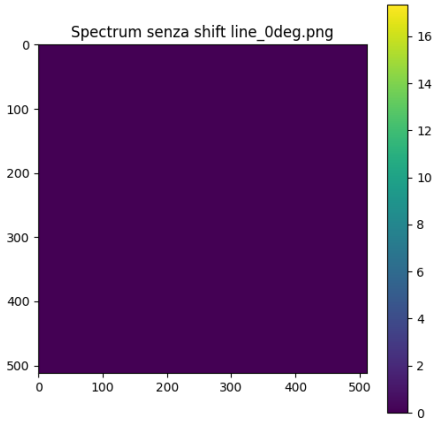
Calcolata la trasformata effettuiamo lo shift della trasformata che server per spostare la componente a frequenza zero al centro dello spettro. Questo è particolarmente utile dove l'analisi della trasformata di Fourier produce uno spettro con la componente a frequenza zero all'inizio dell'array.
Per calcolare lo spettro abbiamo trovato il valore assoluto usando abs e poi abbiamo eseguito un'operazione logaritmica +1 per poter ridurre il rumore e rendere più visibile il risultato, mentre il +1 è per evitare il logaritmo di 0.

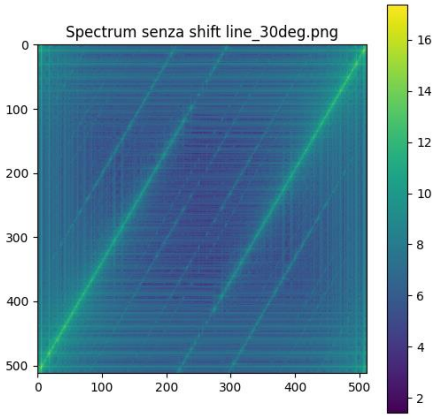
1.6 Visualizzare lo spettro di potenza

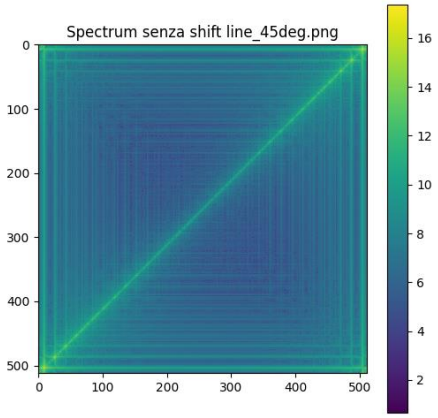
```
# Plot the magnitude spectrum
plt.figure(figsize=(6, 6))
plt.imshow(magnitude_spectrum)
plt.title('Spectrum '+path_to_img)
plt.colorbar()
```

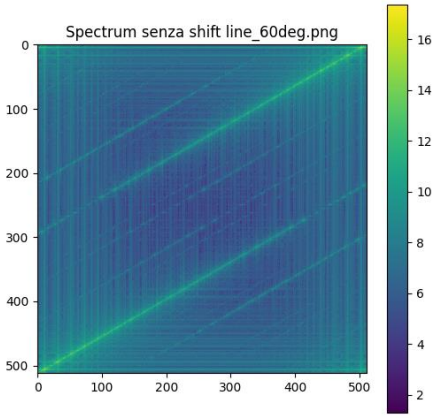
Tramite la libreria MatPlotLib possiamo plottare i nostri risuilati per visualizzarli in un grafico.

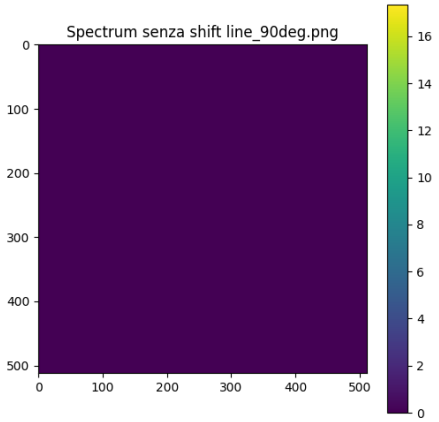
1.6.1 Spettro senza shift



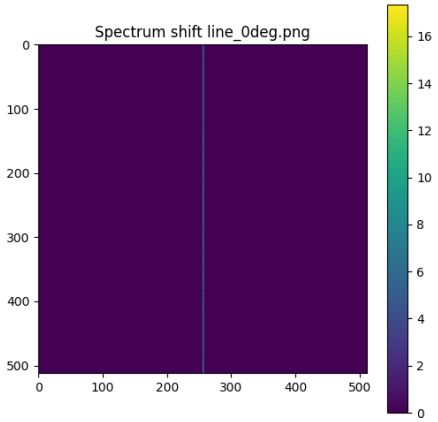


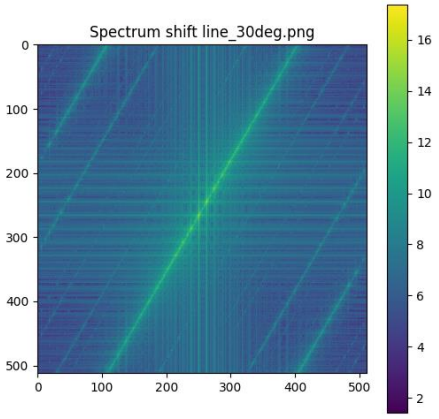


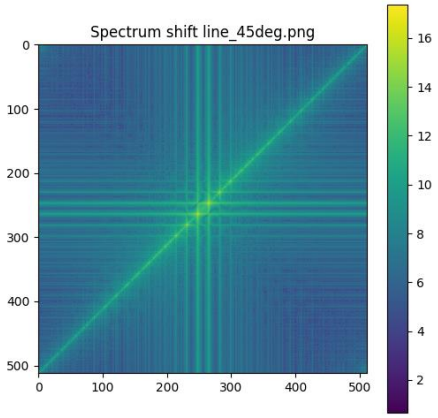


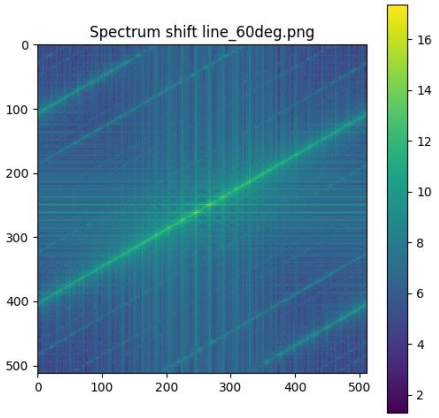


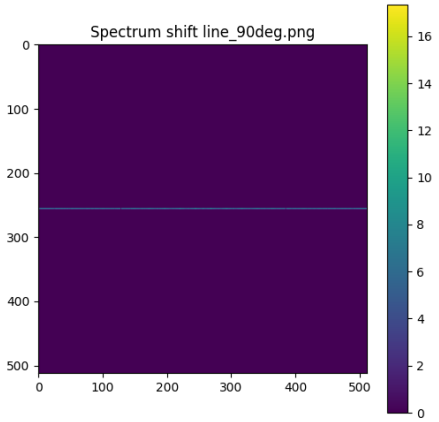
1.6.2 Spettro con Shift









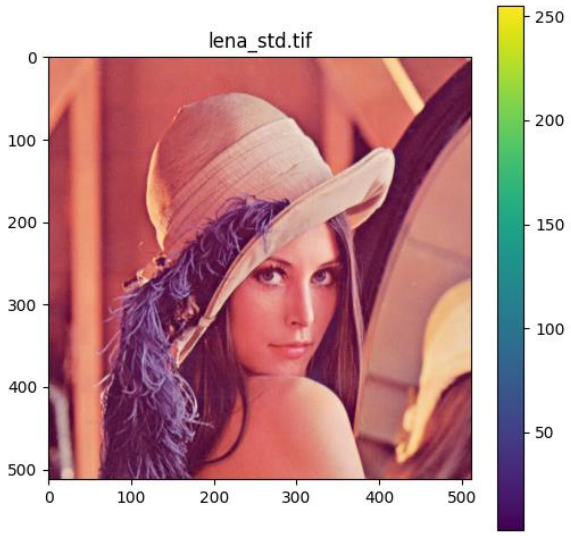


Capitolo 2. Applicare maschera a immagini

2.1 Descrizione

L'obbiettivo è quello di usare l'immagine "lena_std.tif" da cui bisogna isolare i loro canali per usarne uno solo, calcolare la loro trasformata 2d, visualizzarne lo spettro di potenza e applicare una maschera all'immagine

2.2 Immagine originale



2.3 Isolare un canale

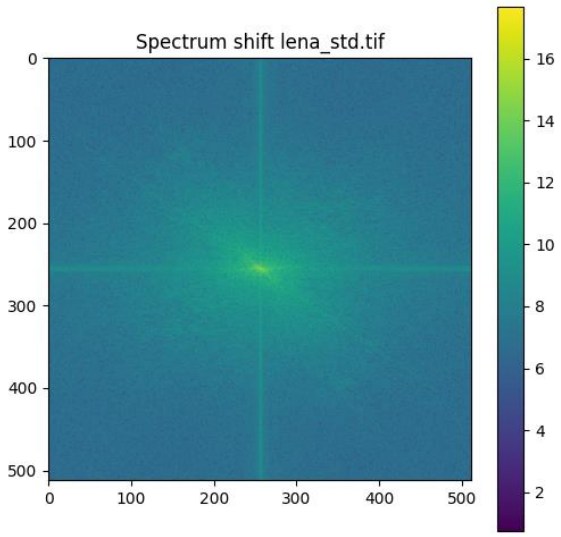
```
path_to_img = "lena_std.tif"
```

Per isolare i canali abbiamo effettuato lo stesso procedimento in 1.3, unica differenza abbiamo usato un'immagine diversa.

2.4 Visualizzare lo spettro

Per calcolare lo spettro e visualizzarlo abbiamo effettuato lo stesso procedimento in 1.4 e 1.5.

2.4.1 Spettro con shift



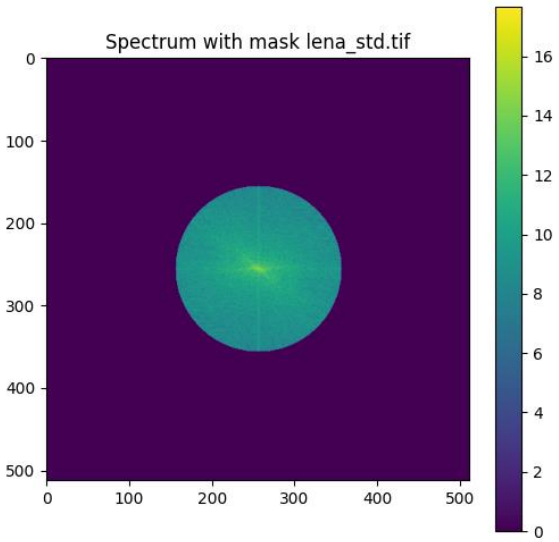
2.5 Applicare un filtro passa-basso, fatto manualmente con una maschera

```
#Mask
radius = 100
mask = np.zeros((width, height))
center_x, center_y = width // 2, height // 2
for x in range(width):
        for y in range(height):
                if (x - center_x) ** 2 + (y - center_y) ** 2 <= radius ** 2:
                        mask[x, y] = 1

spectrum_mask = magnitude_spectrum * mask

# Plot the spectrum with mask
plt.figure(figsize=(6, 6))
plt.imshow(spectrum_mask)
plt.title('Spectrum with mask '+path_to_img)
plt.colorbar()
```

Per creare la maschera non abbiamo fatto altro che creare una matrice della stessa dimensione dell'immagine e impostare a 1 o 0 le parti che vogliamo mascherare o no. In questo caso abbiamo scelto una maschera rotonda, quindi tramite l'uso di due cicli for abbiamo costruito una maschera rotonda di raggio 100.

Per applicare la maschera abbiamo moltiplicato il nostro spettro di potenza con la maschera, per poi farne il plot per visualizzarla.

2.5.1 Spettro con shift e maschera

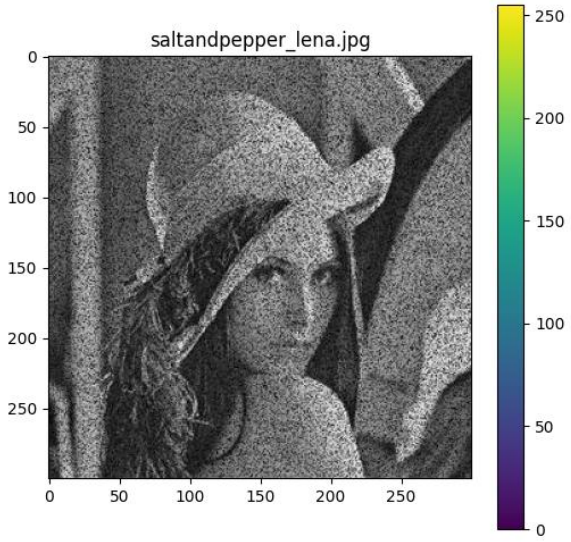


Capitolo 3. Applicare filtro a mediana o gaussiano a immagini

3.1 Descrizione

L'obbiettivo è quello di usare l'immagine "saltandpepper_lena.jpg" da cui bisogna isolare i suoi canali per usarne uno solo, calcolare la loro trasformata 2d, visualizzarne lo spettro di potenza, applicare una maschera all'immagine e applicare il filtro mediano e gaussiano

3.2 Immagine originale



3.3 Isolare un canale

```
path_to_img = "saltandpepper_lena.jpg"
```

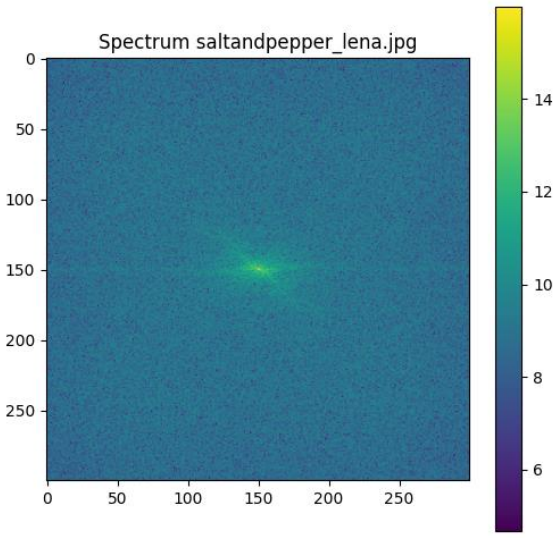
Per isolare i canali abbiamo effettuato lo stesso procedimento in 1.3, unica differenza abbiamo usato un'immagine diversa.

In questo caso l'immagine di base aveva un solo canale essendo in bianco e nero, ma noi l'abbiamo convertita in RGB come nelle precedenti esercitazioni, così abbiamo isolate sempre tre canali.

3.4 Visualizzare lo spettro

Per calcolare lo spettro e visualizzarlo abbiamo effettuato lo stesso procedimento in 1.4 e 1.5.

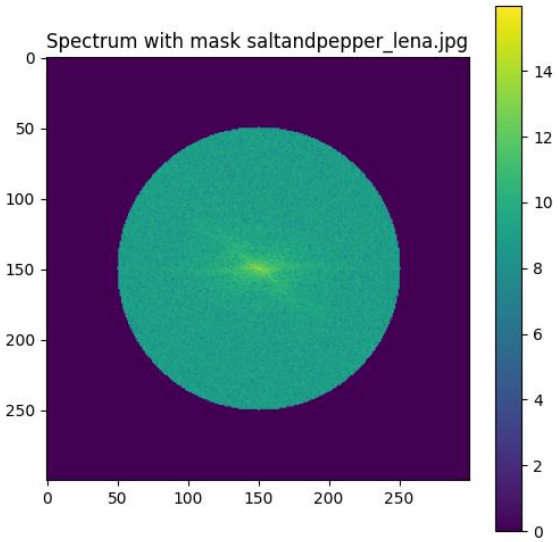
3.4.1 Spettro con shift



3.5 Applicare un filtro passa-basso, fatto manualmente con una maschera

Per creare la maschera e visualizzare lo spettro con la maschera abbiamo effettuato lo stesso procedimento in 2.4.

3.5.1 Spettro con shift e maschera



3.6 Applicare un filtro a Mediana o Gaussiano

```
# Plot the Gaussian Filter
gaussian=gaussian_filter(red_channel, sigma=1)

plt.figure(figsize=(6, 6))
plt.imshow(gaussian)
plt.title('gaussian Filter '+path_to_img)
plt.colorbar()

# Plot the Median filter
median=median_filter(red_channel, size=20)
plt.figure(figsize=(6, 6))
plt.imshow(median)
plt.title('Median filter '+path_to_img)
plt.colorbar()
```

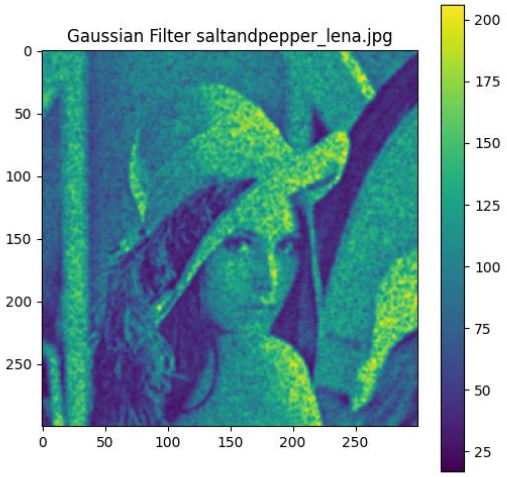
Per applicare il filtro a media e gaussiano abbiamo usato le funzioni fornite da scipy gaussian_filter e median_filter.

Il filtro gaussiano lo abbiamo applicato ad un canale dell'immagine e abbiamo inserito il parametro `sigma=1` che specifica la deviazione standard della funzione gaussiana, che determina quanto il filtro deve essere "sfocato".

Il filtro mediano lo abbiamo applicate anch'esso ad un canale dell'immagine e abbiamo inserito il parametro `size=20` che specifica la dimensione del filtro, ovvero il lato del quadrato (20x20 pixel) utilizzato per calcolare il valore mediano dei pixel.

Poi abbiamo usato sempre MatPlotLib per plottare i risultati.

3.6.1 Filtro Gaussiano



3.6.2 Filtro Mediano