INSTITUTO FEDERAL
São Paulo
Câmpus Birigui

Nome: Lucas De Paiva Lima, Turma:201, Prontuário: Bi3008371

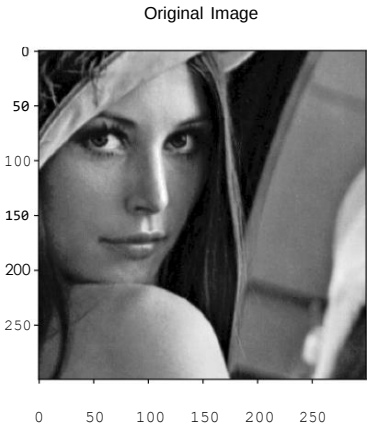
Nome: Carlos Vinicius Baggio Savian, Turma:181, Prontuário: Bi3002217

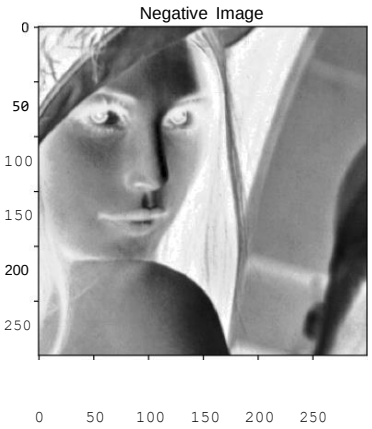
1. **[OPERAÇÃO PONTO A PONTO]:**
    1. Calcular o negativo das imagens;
    2. Diminuir pela metade a intensidade dos pixels;
    3. Incluir 4 quadrados brancos 10 x 10 pixels em cada canto das imagens;
    4. Incluir 1 quadrado preto 15X15 no centro das imagens

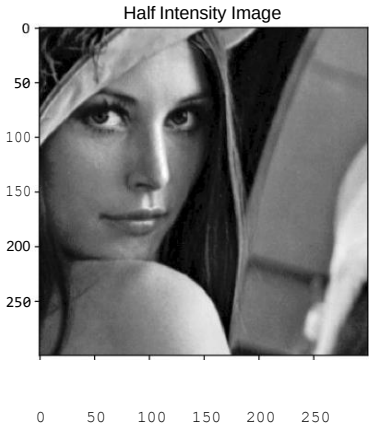
Código

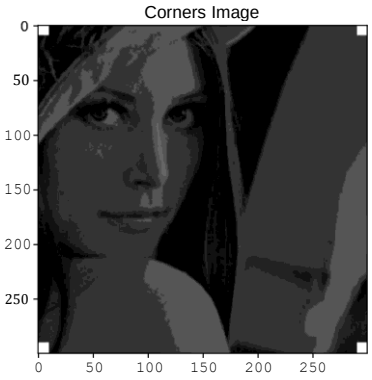
```
# OPERAÇÃO PONTO A PONTO
import numpy as np
from PIL import Image

def main():
    # Open image
    image = Image.open(r'C:\Users\Usuario\OneDrive\Área de
Trabalho\tudo\lena.tif')

    # Convert img to numpy array
    npImage = np.array(image)

    # Calculate the negative of the image
    npImageNegative = 255 - npImage

    # Reduce pixel intensity by half
    npImageHalfIntensity = (npImage / 2).astype(int)

    # Add white squares in each corner
    npImageCornerSquares = npImageHalfIntensity.copy()
    square_size = 10
    npImageCornerSquares[:square_size, :square_size] = 255
    npImageCornerSquares[:square_size, -square_size:] = 255
    npImageCornerSquares[-square_size:, :square_size] = 255
    npImageCornerSquares[-square_size:, -square_size:] = 255

    # Add black square in the center
```

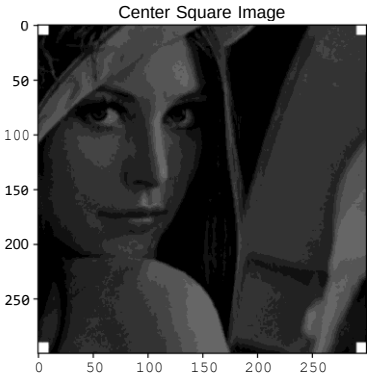
```
    npImageCenterSquare = npImageCornerSquares.copy()
    center_x, center_y = npImageCenterSquare.shape[0] // 2,
npImageCenterSquare.shape[1] // 2
    square_size = 15
    npImageCenterSquare[center_x - square_size // 2:center_x +
square_size // 2,
                        center_y - square_size // 2:center_y +
square_size // 2] = 0

    # Convert ndarray image to Pillow images
    image_negative = Image.fromarray(npImageNegative)
    image_half_intensity = Image.fromarray(npImageHalfIntensity)
    image_corner_squares = Image.fromarray(npImageCornerSquares)
    image_center_square = Image.fromarray(npImageCenterSquare)

    # Display the processed images
    image_negative.show()
    image_half_intensity.show()
    image_corner_squares.show()
    image_center_square.show()

if __name__ == "__main__":
    main()
```

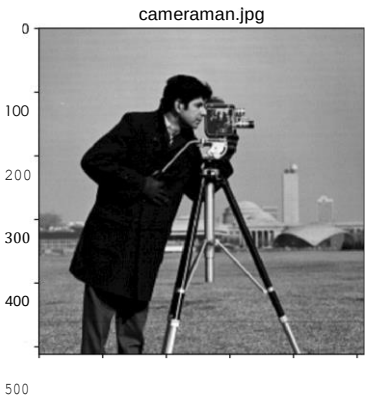

Original Image
0
50
100
150
200
250
0 50 100 150 200 250

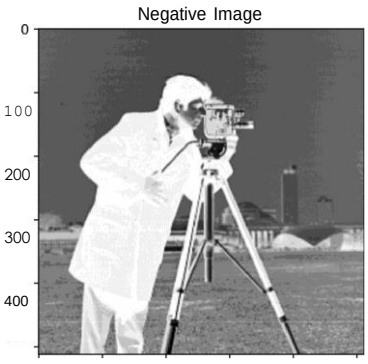

Negative Image
0
50
100
150
200
250
0 50 100 150 200 250

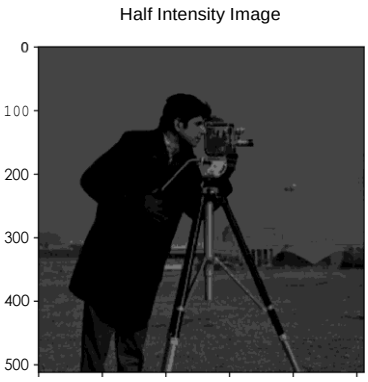

Half Intensity Image
0
50
100
150
200
250
0 50 100 150 200 250

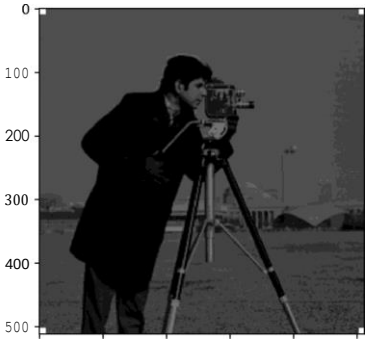

Corners Image
0
50
100
150
200
250
0 50 100 150 200 250

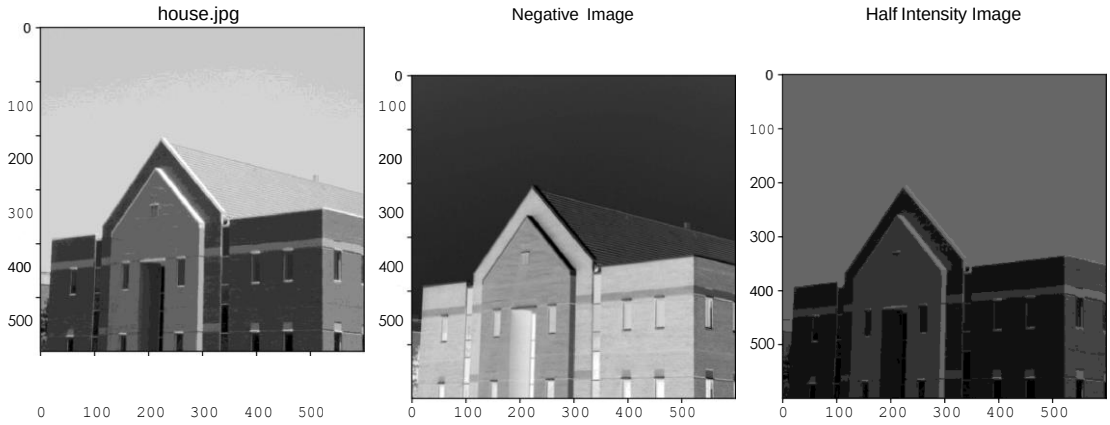

Center Square Image
0
50
100
150
200
250
0 50 100 150 200 250

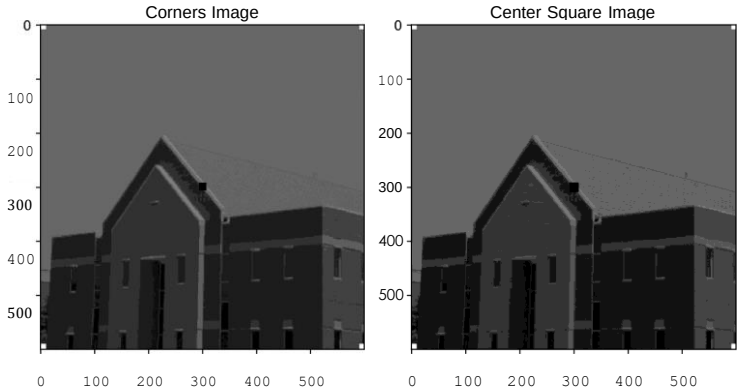

cameraman.jpg
0
100
200
300
400
500


Negative Image
0
100
200
300
400


Half Intensity Image
0
100
200
300
400
500


0
100
200
300
400
500



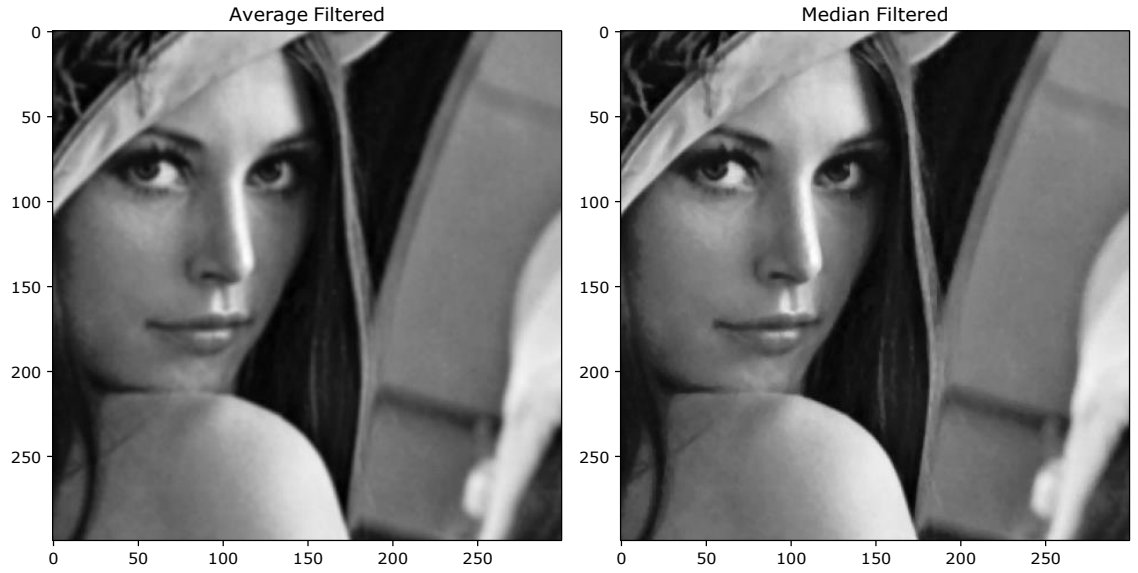


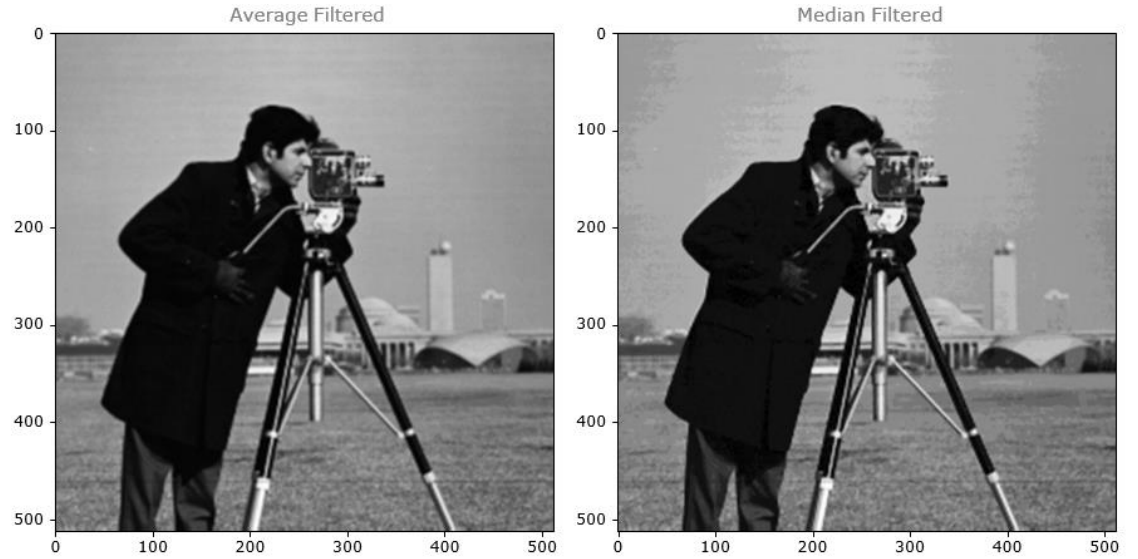
2. **[OPERAÇÃO POR VIZINHANÇA]:** Utilizar kernel 3x3 pixels e desconsiderar pixels das extremidades. Para cada filtro implementar utilizando apenas numpy, utilizando pillow, utilizando opencv e utilizando scipy.
    1. Calcular o filtro da média;
    2. Calcular o filtro da mediana;

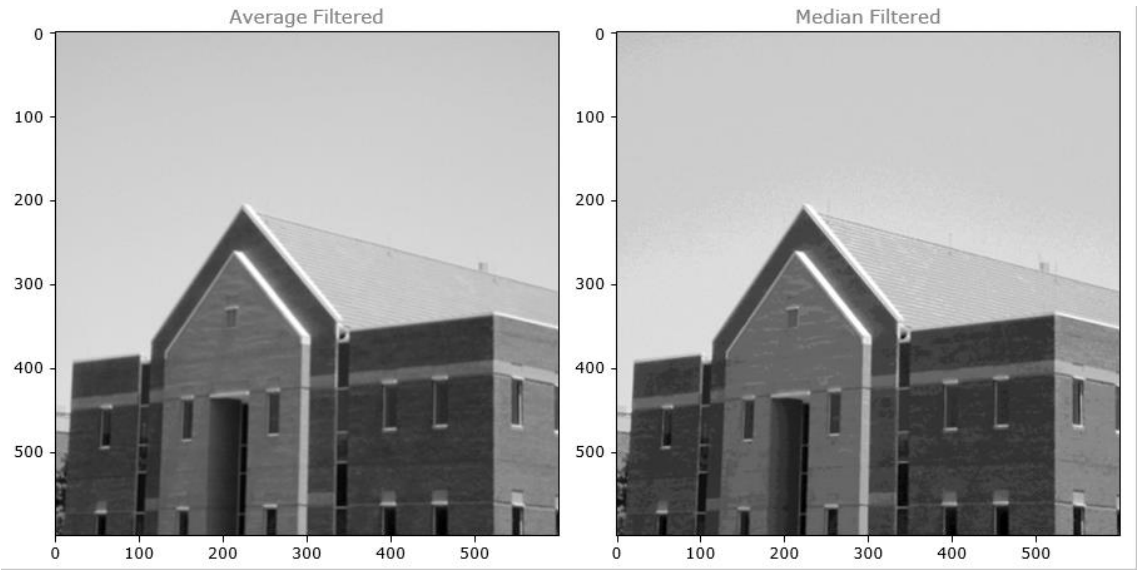
OpenCV

```
import cv2
import matplotlib.pyplot as plt  # Add this import
from matplotlib.backends.backend_pdf import PdfPages

def apply_filters_to_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    average_filtered_image = cv2.blur(image, (3, 3))
    median_filtered_image = cv2.medianBlur(image, 3)

    return average_filtered_image, median_filtered_image

def generate_pdf(image_paths):
```

```
    pdf_filename = 'opencv_ex2.pdf'
    pdf_pages = PdfPages(pdf_filename)

    for image_path in image_paths:
        average_filtered_image, median_filtered_image =
apply_filters_to_image(image_path)

        plt.figure(figsize=(10, 5))

        plt.subplot(1, 2, 1)
        plt.imshow(average_filtered_image, cmap='gray')
        plt.title('Average Filtered')

        plt.subplot(1, 2, 2)
        plt.imshow(median_filtered_image, cmap='gray')
        plt.title('Median Filtered')

        plt.tight_layout()

        pdf_pages.savefig(plt.gcf())
        plt.close()

    pdf_pages.close()
    print(f'PDF saved as {pdf_filename}')

def main():
    image_paths = [
        r'C:\Users\ifsp\Downloads\lena.tif',
        r'C:\Users\ifsp\Downloads\cameraman.tif',
        r'C:\Users\ifsp\Downloads\house.tif'
    ]

    generate_pdf(image_paths)

if __name__ == "__main__":
    main()
```


Average Filtered
Median Filtered

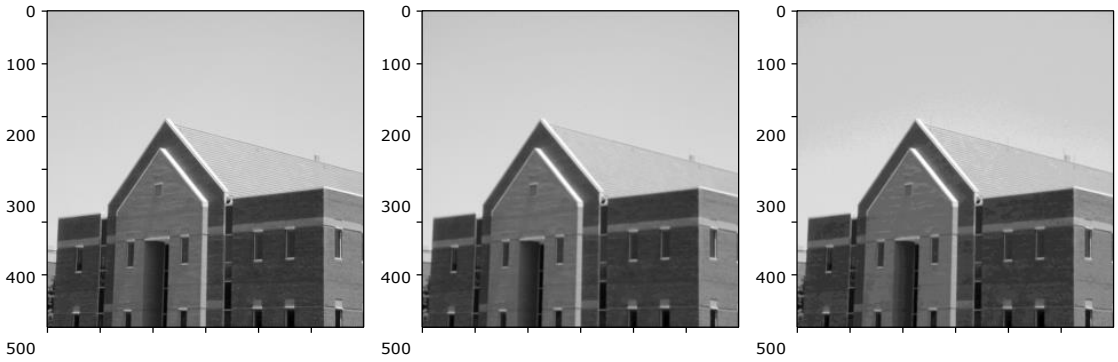

Average Filtered
Median Filtered


Average Filtered
Median Filtered

Numpy

```python
import numpy as np
from PIL import Image
from scipy.ndimage import convolve
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

def apply_average_filter_np(npImage):
    kernel = np.array([[1, 1, 1],
                       [1, 1, 1],
                       [1, 1, 1]]) / 9
    return convolve(npImage, kernel, mode='constant')

def apply_median_filter_np(npImage):
    result = npImage.copy()
    for x in range(1, npImage.shape[0] - 1):
        for y in range(1, npImage.shape[1] - 1):
            window = npImage[x-1:x+2, y-1:y+2]
            result[x, y] = np.median(window)
    return result

def generate_image_plots(image_path, npImage):
    average_filtered = apply_average_filter_np(npImage)
    median_filtered = apply_median_filter_np(npImage)

    plt.figure(figsize=(10, 5))

    plt.subplot(1, 3, 1)
    plt.imshow(npImage, cmap='gray')
    plt.title('Original')

    plt.subplot(1, 3, 2)
    plt.imshow(average_filtered, cmap='gray')
    plt.title('Average Filter')

    plt.subplot(1, 3, 3)
    plt.imshow(median_filtered, cmap='gray')
    plt.title('Median Filter')

    plt.tight_layout()

    return plt

def main():
    image_paths = [
        r'C:\Users\ifsp\Downloads\lena.tif',
        r'C:\Users\ifsp\Downloads\cameraman.tif',
        r'C:\Users\ifsp\Downloads\house.tif'
    ]
```

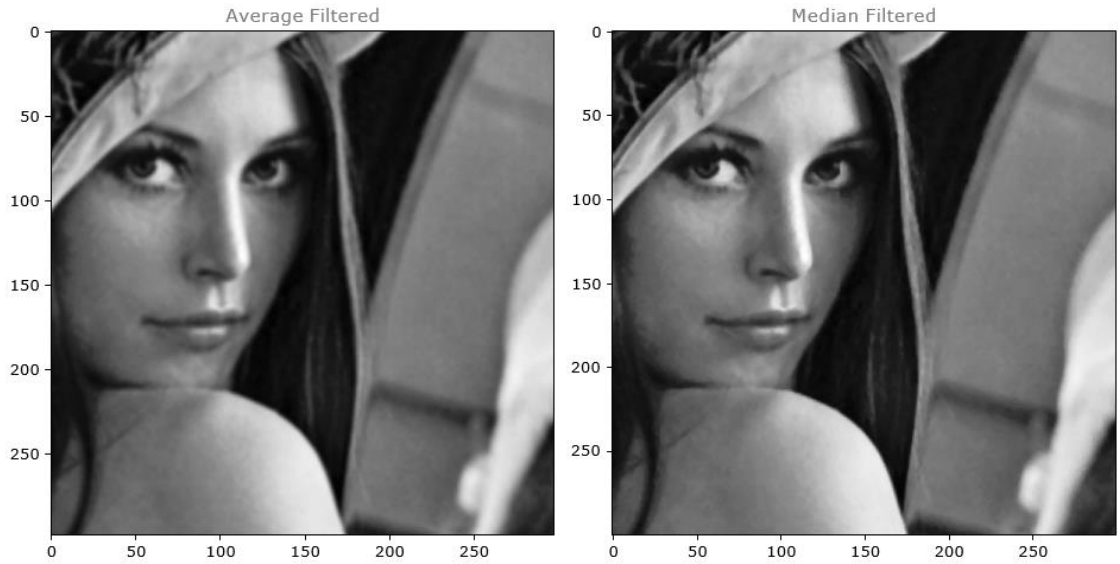
```
    pdf_filename = 'teste_do_2.pdf'  # Set the PDF filename here
    pdf_pages = PdfPages(pdf_filename)

    for image_path in image_paths:
        image = Image.open(image_path).convert('L')  # Convert to
grayscale
        npImage = np.array(image)

        plt = generate_image_plots(image_path, npImage)
        pdf_pages.savefig(plt.gcf())
        plt.close()

    pdf_pages.close()
    print(f'PDF saved as {pdf_filename}')

if __name__ == "__main__":
    main()
```

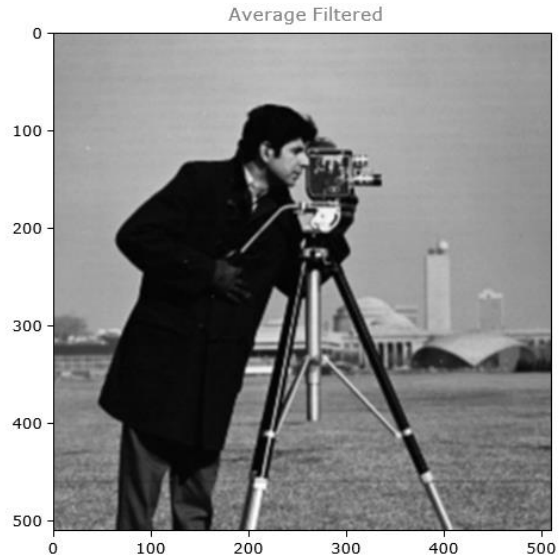


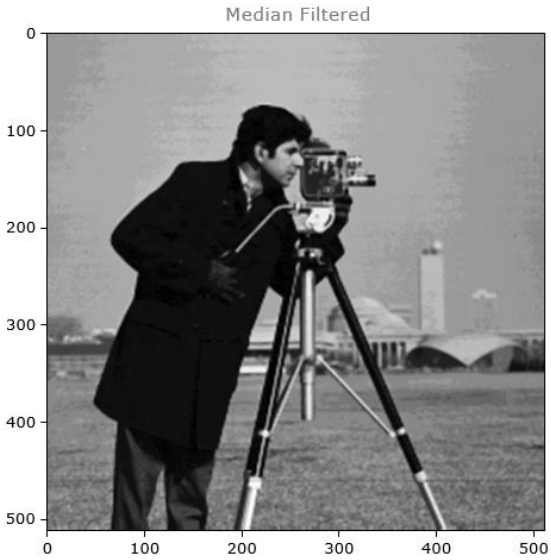
Scipy

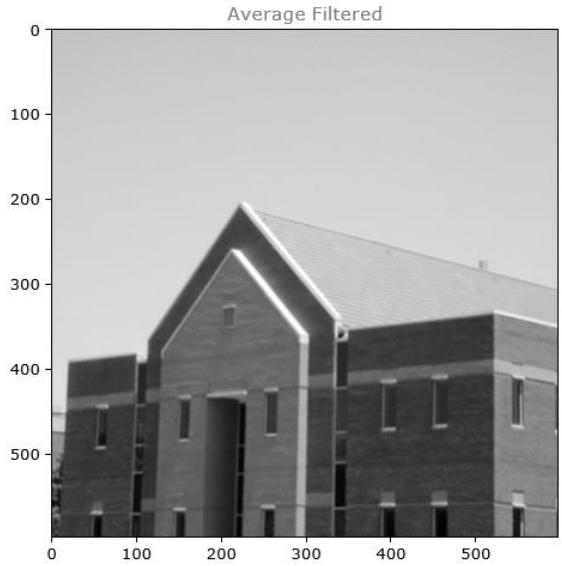
```
import numpy as np
from PIL import Image
from scipy.signal import convolve2d
from scipy.ndimage import median_filter
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.pyplot as plt

def apply_average_filter_scipy(npImage):
    kernel = np.array([[1, 1, 1],
                       [1, 1, 1],
                       [1, 1, 1]]) / 9
    return convolve2d(npImage, kernel, mode='valid')

def apply_median_filter_scipy(npImage):
    return median_filter(npImage, size=3)

def generate_pdf(image_paths):
    pdf_filename = 'scipy_ex2.pdf'
    pdf_pages = PdfPages(pdf_filename)

    for image_path in image_paths:
        image = Image.open(image_path).convert('L')  # Convert to
grayscale
        npImage = np.array(image)

        average_filtered = apply_average_filter_scipy(npImage)
        median_filtered = apply_median_filter_scipy(npImage)

        average_filtered_image =
Image.fromarray(average_filtered.astype(np.uint8))
        median_filtered_image =
Image.fromarray(median_filtered.astype(np.uint8))

        plt.figure(figsize=(10, 5))
```

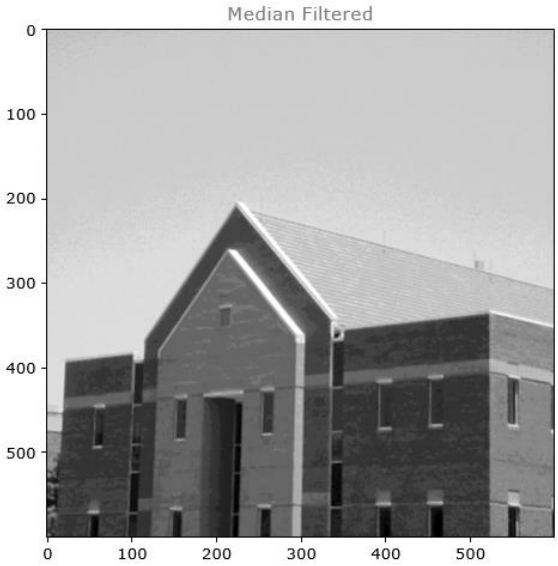
```
        plt.subplot(1, 2, 1)
        plt.imshow(average_filtered_image, cmap='gray')
        plt.title('Average Filtered')

        plt.subplot(1, 2, 2)
        plt.imshow(median_filtered_image, cmap='gray')
        plt.title('Median Filtered')

        plt.tight_layout()

        pdf_pages.savefig(plt.gcf())
        plt.close()

    pdf_pages.close()
    print(f'PDF saved as {pdf_filename}')

def main():
    image_paths = [
        r'C:\Users\ifsp\Downloads\lena.tif',
        r'C:\Users\ifsp\Downloads\cameraman.tif',
        r'C:\Users\ifsp\Downloads\house.tif'
    ]

    generate_pdf(image_paths)

if __name__ == "__main__":
    main()
```







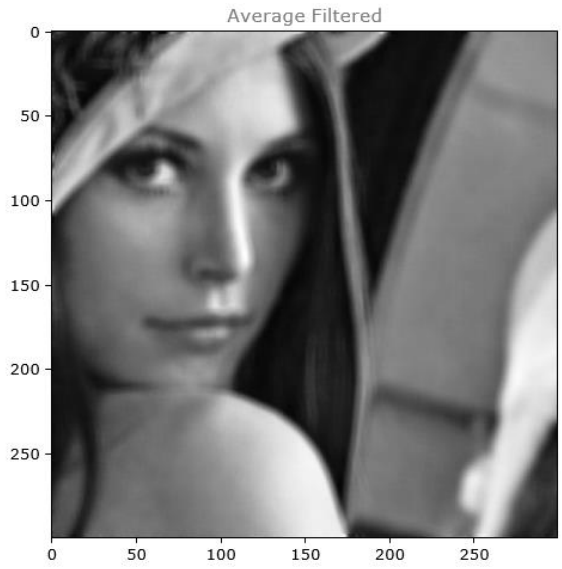


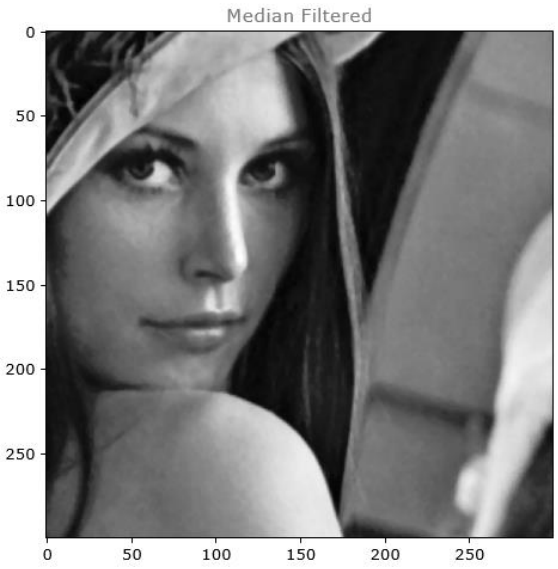


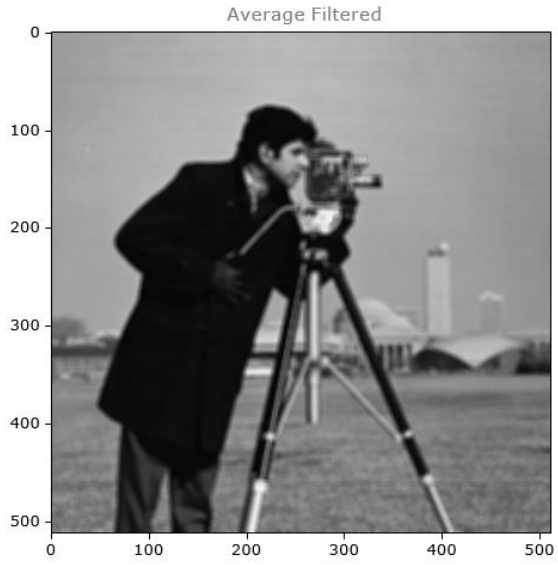
Pilow

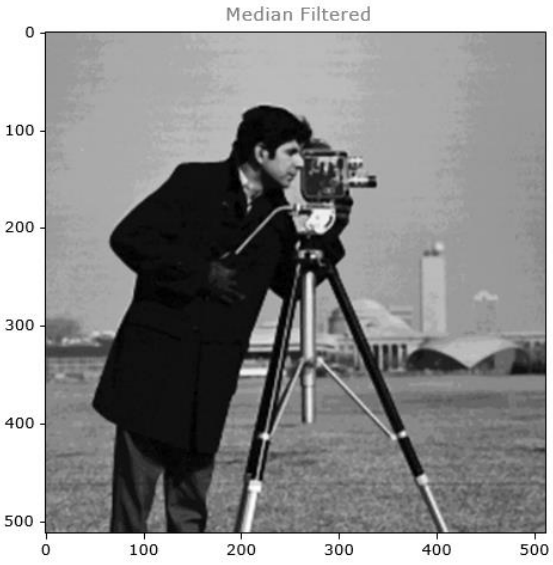
```
from PIL import Image, ImageFilter
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.pyplot as plt

def apply_filters_to_image(image_path):
    image = Image.open(image_path).convert('L')  # Convert to grayscale

    average_filtered_image = image.filter(ImageFilter.BLUR)
    median_filtered_image =
image.filter(ImageFilter.MedianFilter(size=3))

    return average_filtered_image, median_filtered_image

def generate_pdf(image_paths):
    pdf_filename = 'pilow_ex2.pdf'
```

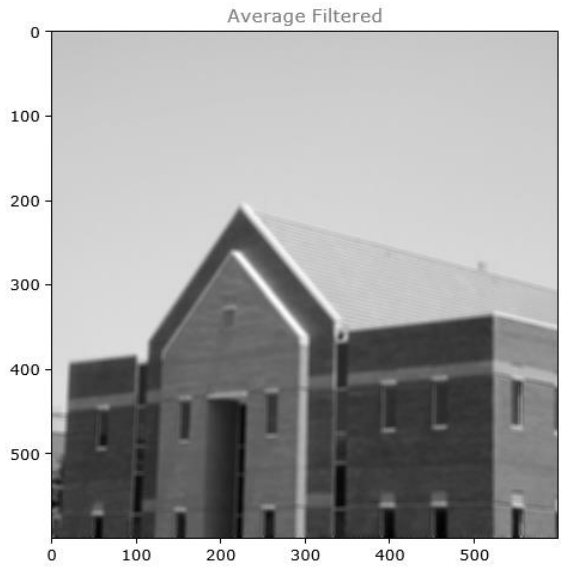
```
    pdf_pages = PdfPages(pdf_filename)

    for image_path in image_paths:
        average_filtered_image, median_filtered_image =
apply_filters_to_image(image_path)

        plt.figure(figsize=(10, 5))

        plt.subplot(1, 2, 1)
        plt.imshow(average_filtered_image, cmap='gray')  # Set colormap
to grayscale
        plt.title('Average Filtered')

        plt.subplot(1, 2, 2)
        plt.imshow(median_filtered_image, cmap='gray')  # Set colormap to
grayscale
        plt.title('Median Filtered')

        plt.tight_layout()

        pdf_pages.savefig(plt.gcf())
        plt.close()

    pdf_pages.close()
    print(f'PDF saved as {pdf_filename}')

def main():
    image_paths = [
        r'C:\Users\ifsp\Downloads\lena.tif',
        r'C:\Users\ifsp\Downloads\cameraman.tif',
        r'C:\Users\ifsp\Downloads\house.tif'
    ]

    generate_pdf(image_paths)

if __name__ == "__main__":
    main()
```

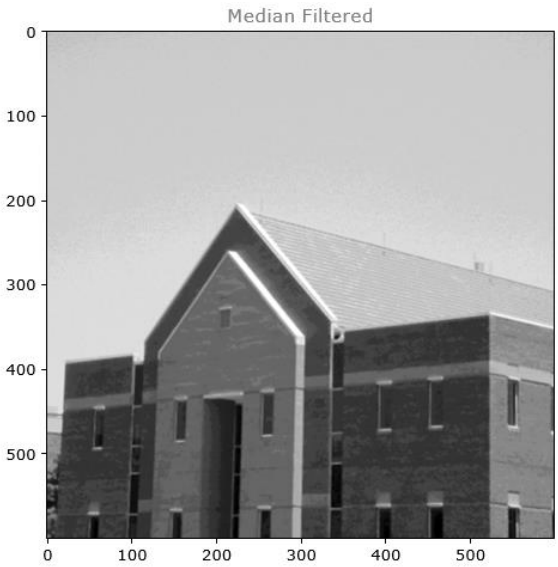

Average Filtered
0
50
100
150
200
250
0
50
100
150
200
250


Median Filtered
0
50
100
150
200
250
0
50
100
150
200
250

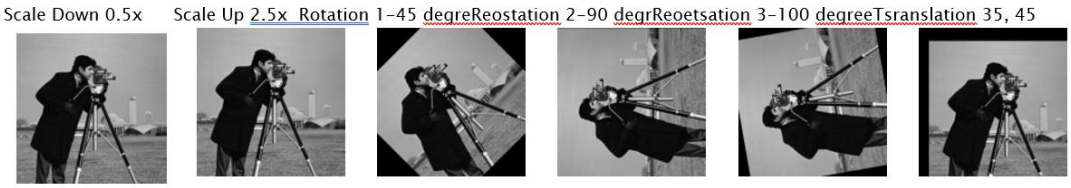

Average Filtered
0
100
200
300
400
500
0
100
200
300
400
500

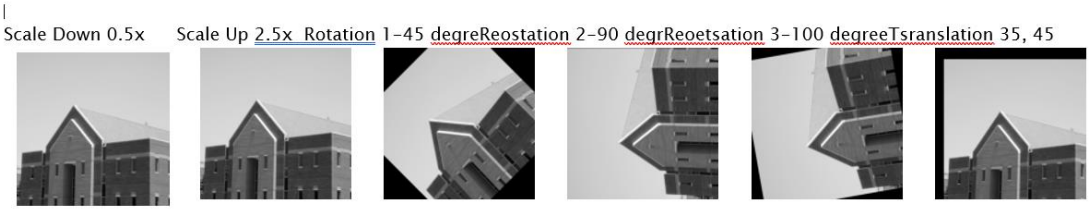

Median Filtered
0
100
200
300
400
500
0
100
200
300
400
500


Average Filtered
0
100
200
300
400
500
0
100
200
300
400
500


Median Filtered
0
100
200
300
400
500
0
100
200
300
400
500

3. **[TRANSFORMAÇÕES GEOMÉTRICAS]:** Para cada filtro implementar utilizando apenas numpy, utilizando pillow, utilizando opencv e utilizando scipy.
    1. Escala: Redução em 1.5x e aumentar em 2.5x;
    2. Rotação em 45º, 90º e 100º;
    3. Translação utilizar os parâmetros que quiser nas coordenadas x e y;
    4. Translação em 35 pixel no eixo X, 45 eixo Y;






```
import numpy as np
from PIL import Image
import cv2
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

# Your utility functions
```

```
        # ... (previous code)

def main():
    image_paths = [
        r'C:/Users/Usuario/OneDrive/Área de Trabalho/tudo/lena.tif',
        r'C:/Users/Usuario/OneDrive/Área de Trabalho/tudo/cameraman.tif',
        r'C:/Users/Usuario/OneDrive/Área de Trabalho/tudo/house.tif'
    ]

    images = []
    for path in image_paths:
        print(f"Loading image from path: {path}")
        image = cv2.imread(path)
        if image is None:
            print(f"Error: Image at path '{path}' could not be loaded.")
        else:
            images.append(image)

    if not images:
        print("No valid images were loaded. Exiting.")
        return

    titles = ['Lena', 'Cameraman', 'House']

    transformed_images = []
    for image in images:
        scaled_down_numpy_image = scale_numpy(image, 0.5)
        scaled_up_numpy_image = scale_numpy(image, 2.5)
        rotated_45_numpy_image = rotate_numpy(image, 45)
        rotated_90_numpy_image = rotate_numpy(image, 90)
        rotated_100_numpy_image = rotate_numpy(image, 100)
        translated_35_45_numpy_image = translate_numpy(image, 35, 45)
        transformed_images.append([
            scaled_down_numpy_image,
            scaled_up_numpy_image,
            rotated_45_numpy_image,
            rotated_90_numpy_image,
            rotated_100_numpy_image,
            translated_35_45_numpy_image
        ])

    pdf_filename = 'geometric_transformations.pdf'
    with PdfPages(pdf_filename) as pdf:
        for title, transformed_set in zip(titles, transformed_images):
            plt.figure(figsize=(12, 8))
            plt.suptitle(title)
```

```
            for i, transformed_image in enumerate(transformed_set,
start=1):
                plt.subplot(1, len(transformed_set), i)
                plt.imshow(cv2.cvtColor(transformed_image,
cv2.COLOR_BGR2RGB))
                if i == 1:
                    plt.title("Scale Down 0.5x")
                elif i == 2:
                    plt.title("Scale Up 2.5x")
                elif i <= 5:
                    plt.title(f"Rotation {i-2} {(i-2)*45} degrees")
                else:
                    plt.title("Translation 35, 45")
                plt.axis('off')

            pdf.savefig()
            plt.close()

if __name__ == "__main__":
    main()
```