Erick de Sousa Almeida **Prontuário: BI3008444**

Processamento Digital de Imagens(PDI)

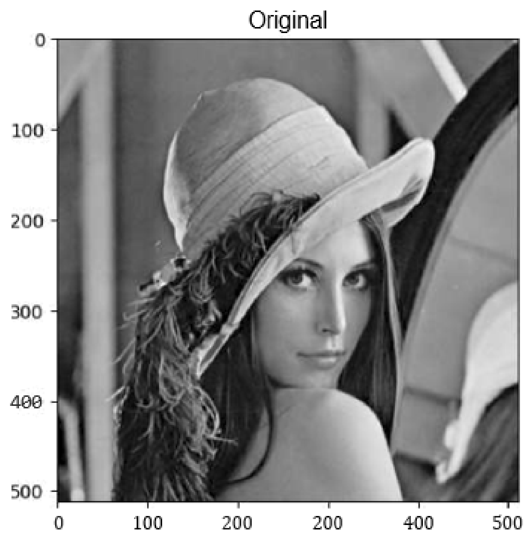
Exercícios- Fundamentos

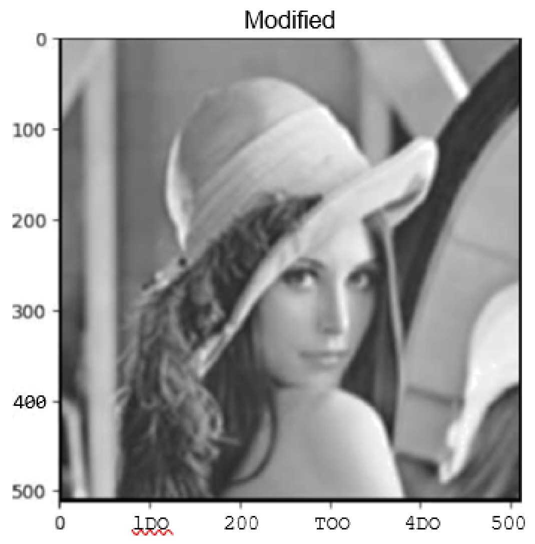
Operação por Vizinhança

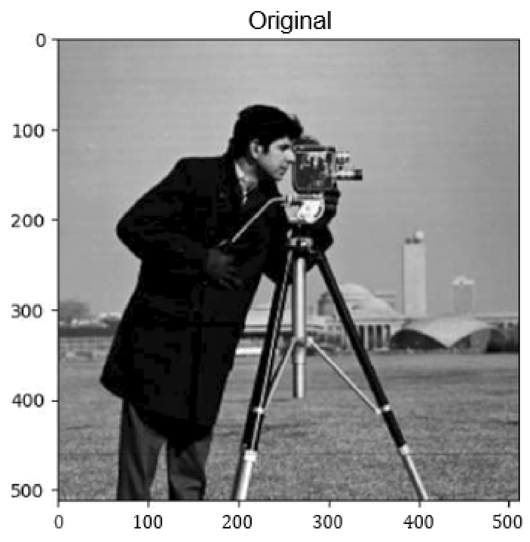
1. Filtro da Média

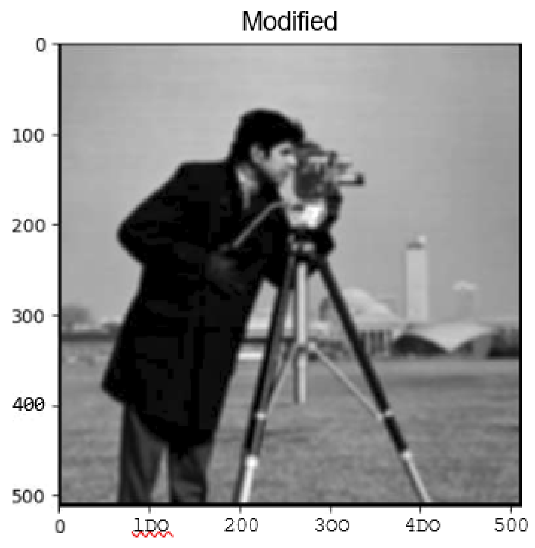
```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image, ImageFilter
import cv2

paths = ['./lena_gray_512.tif', './cameraman.tif', './house.tif']
images = [Image.open(path) for path in paths]

kernel= 3

def process(img, cb: callable):
     np.ndarray: inverted_img = cb(img.copy())

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

plt1 = plt.subplot(1, 2, 1)
plt1.imshow(img, cmap='gray')
plt1.set_title('Original')

plt2 = plt.subplot(1, 2, 2)
plt2.imshow(inverted_img cmap='gray')
plt2.set_title('Modified')

def for_each_imgs(cb: callable):
    for img in images:
        process(img, cb)

def mean_numpy(img):
    np.ndarray= img_nd = np.array(img)
    lines = img_nd.shape[0]
    columns = img_nd.shape[1]
    image_nd = np.zeros((lines, columns), dtype=np.uint8)

    for x in range(kernel, lines - kernel):
        for y in range(kernel, columns - kernel):
             s_xy = img_nd[x - kernel: x + kernel + 1, y - kernel:
y + kernel + 1]
            mean = np.mean(s_xy).astype(int)
            image_nd[x, y] = mean

    return image_nd

for_each_imgs(mean_numpy)
```

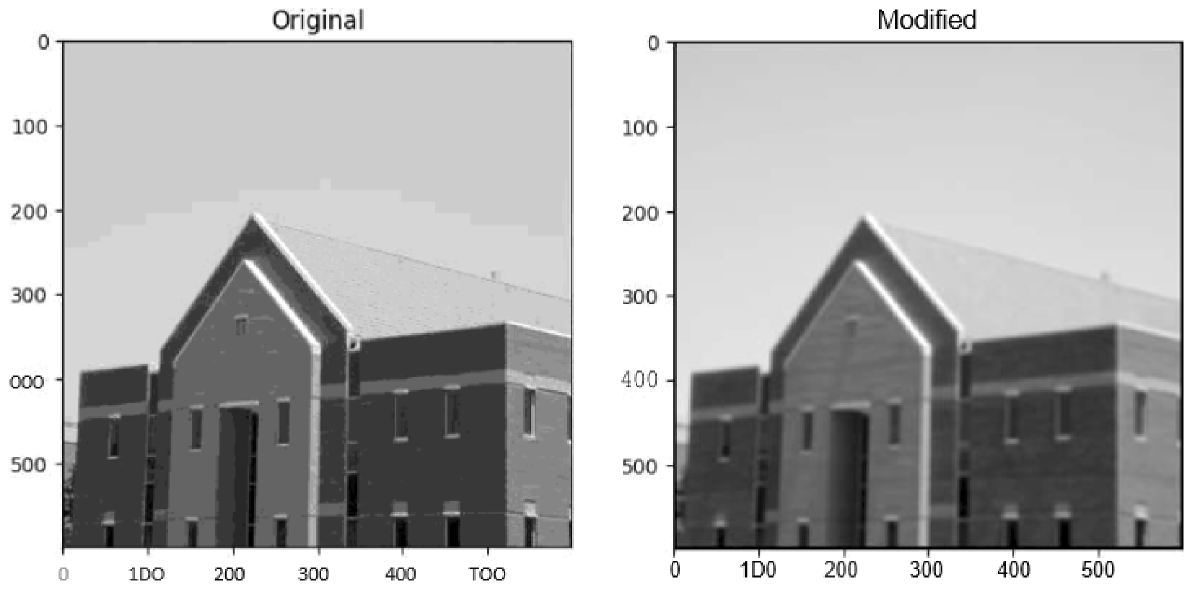
RESULTADOS:





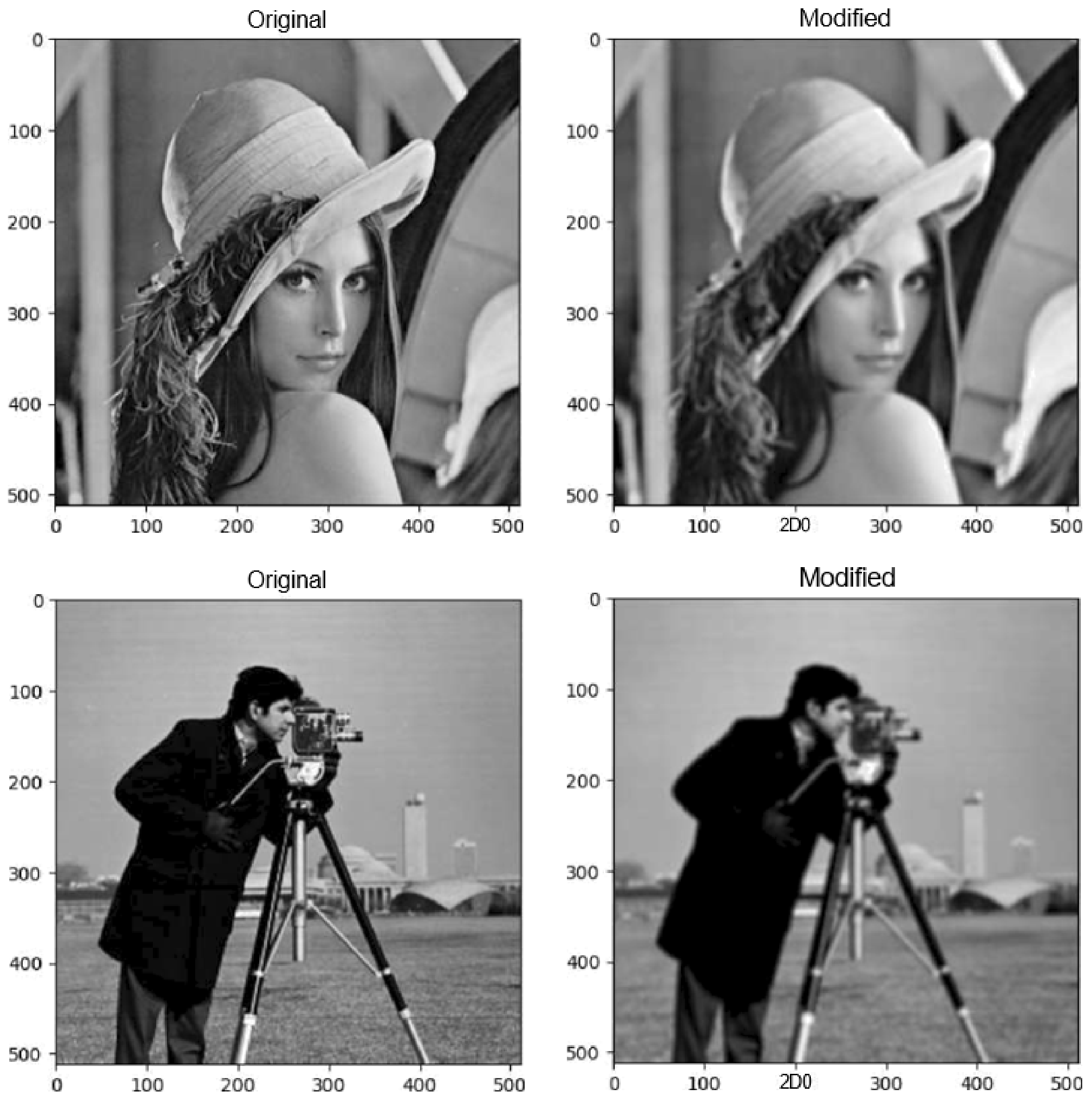


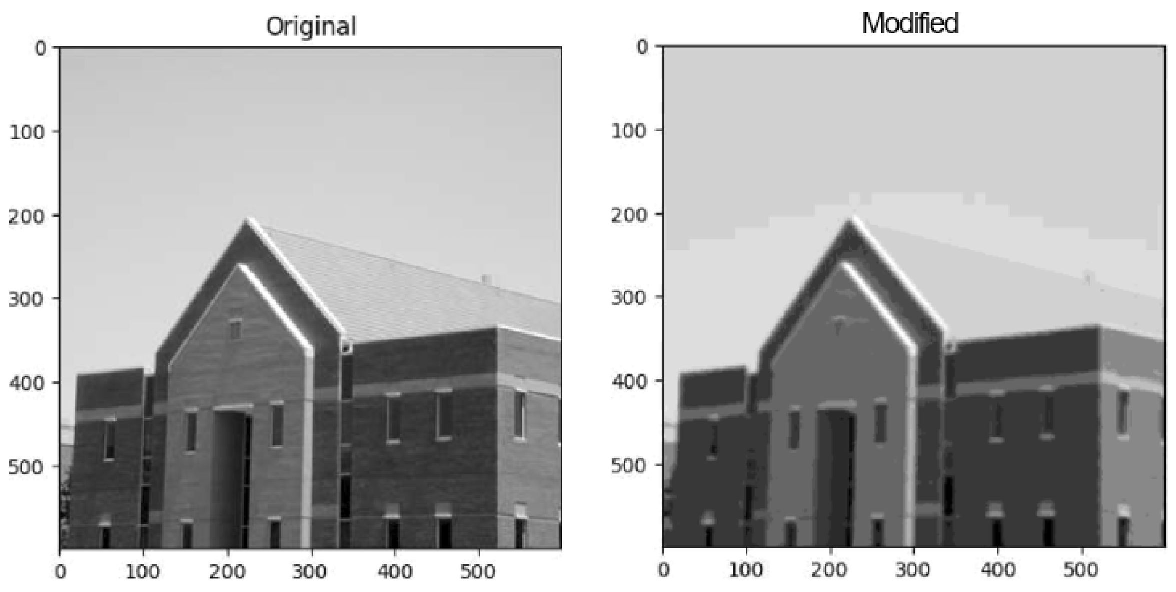




Utilizando o Pillow

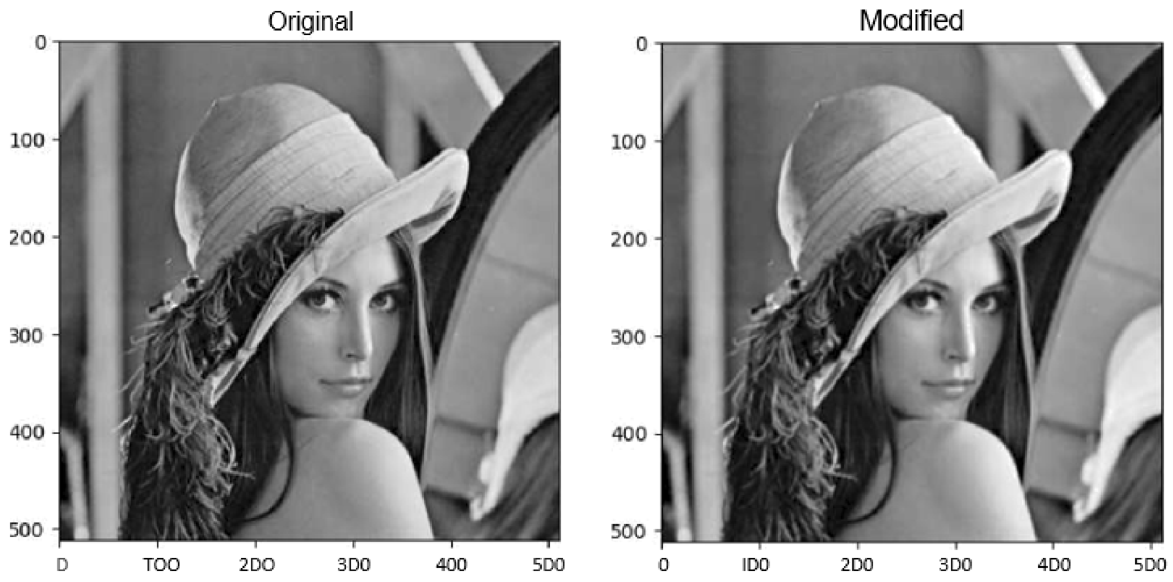
```
def mean_pillow(img) -> np.ndarray:
return img.filter(ImageFilter.BoxBlur(kernel))

for_each_imgs(mean_pillow)
```

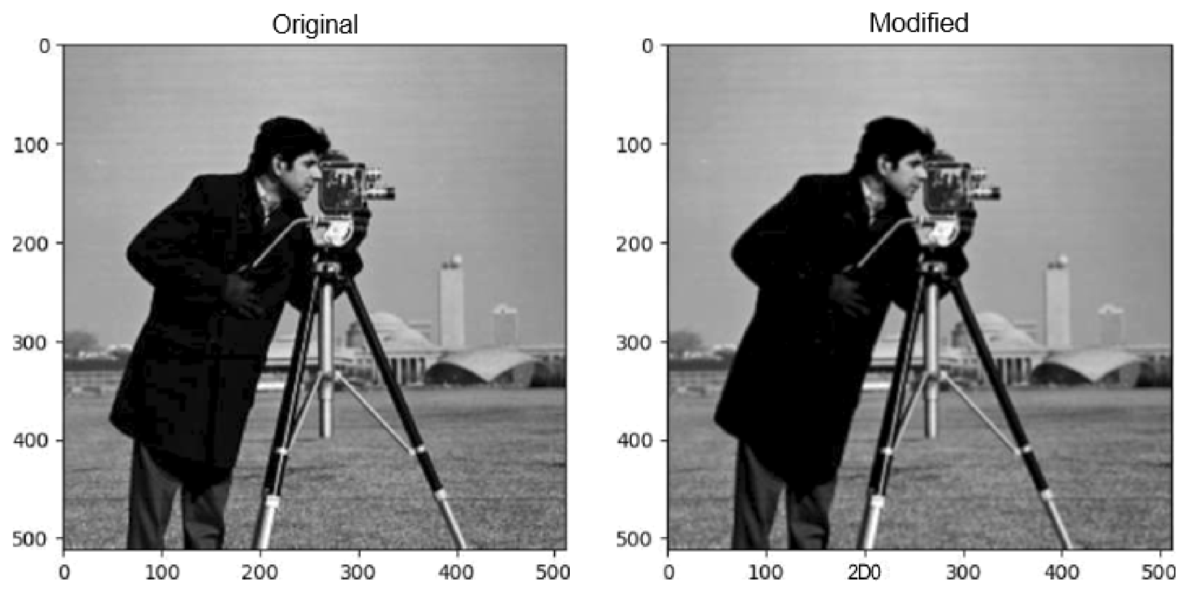

Original
Modified
Original
Modified

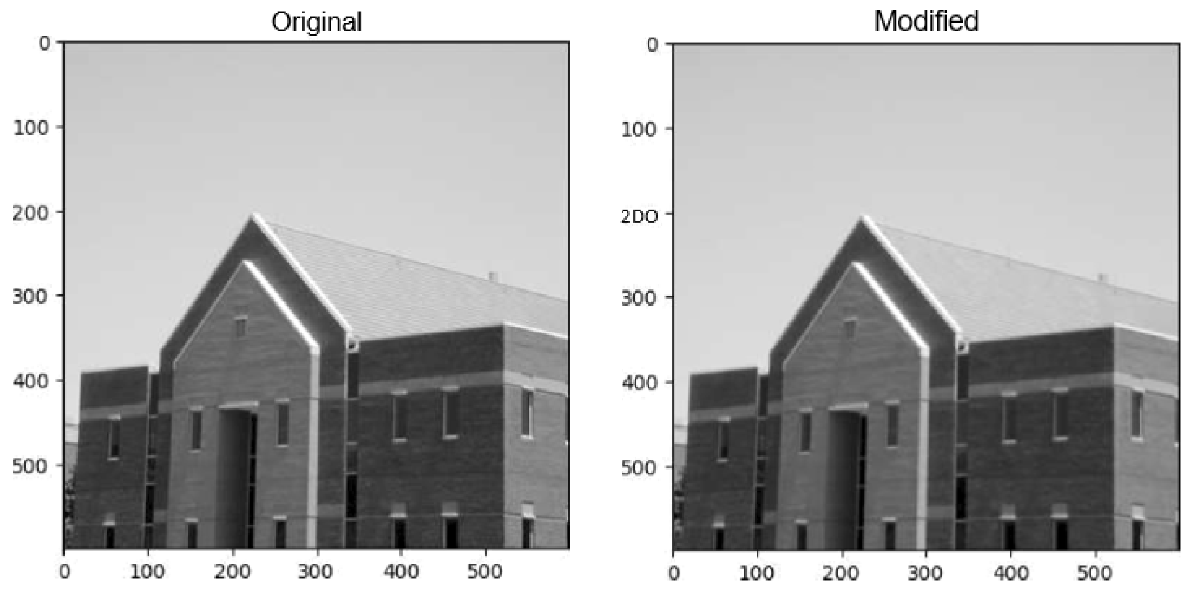


Using OpenCV

```
import cv2

def mean_opencv(img) -> np.ndarray: img_nd = np.array(img)
return cv2.blur(img_nd, (kernel, kernel))


for_each_imgs(mean_opencv)
```





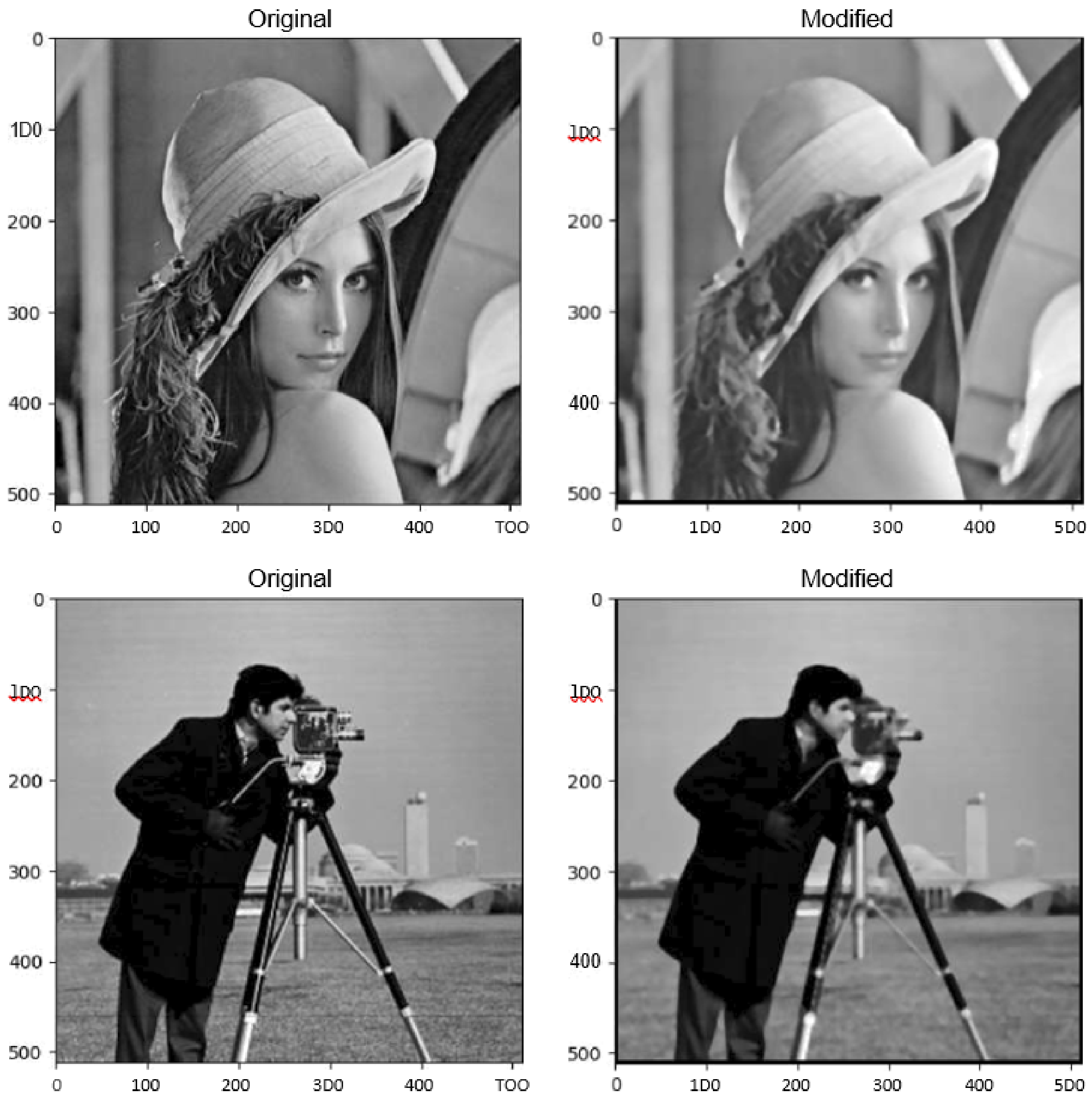


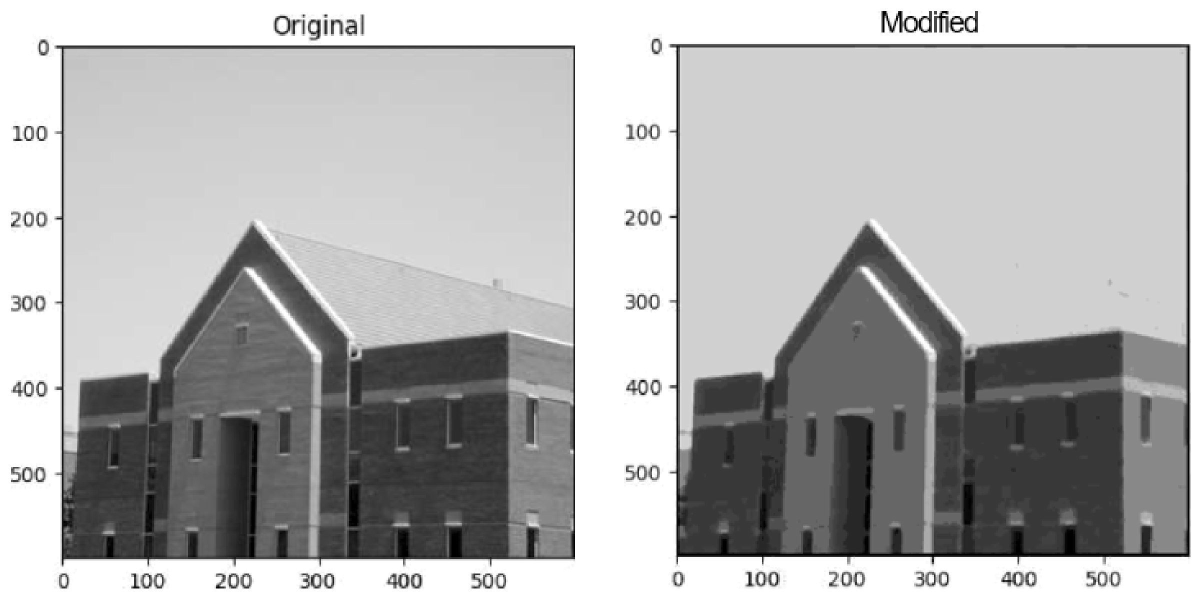
2. Calcular o filtro da mediana

Usando o Numpy

```
def median_numpy(img) -> np.ndarray:
    img_nd = np.array(img)
    lines = img_nd.shape[0]
    columns = img_nd.shape[1]

    image_nd = np.zeros((lines, columns), dtype=np.uint8)

    for x in range(kernel, lines - kernel):
        for y in range(kernel, columns - kernel):
            s_xy = img_nd[x - kernel: x + kernel + 1, y - kernel:
y + kernel + ]
```

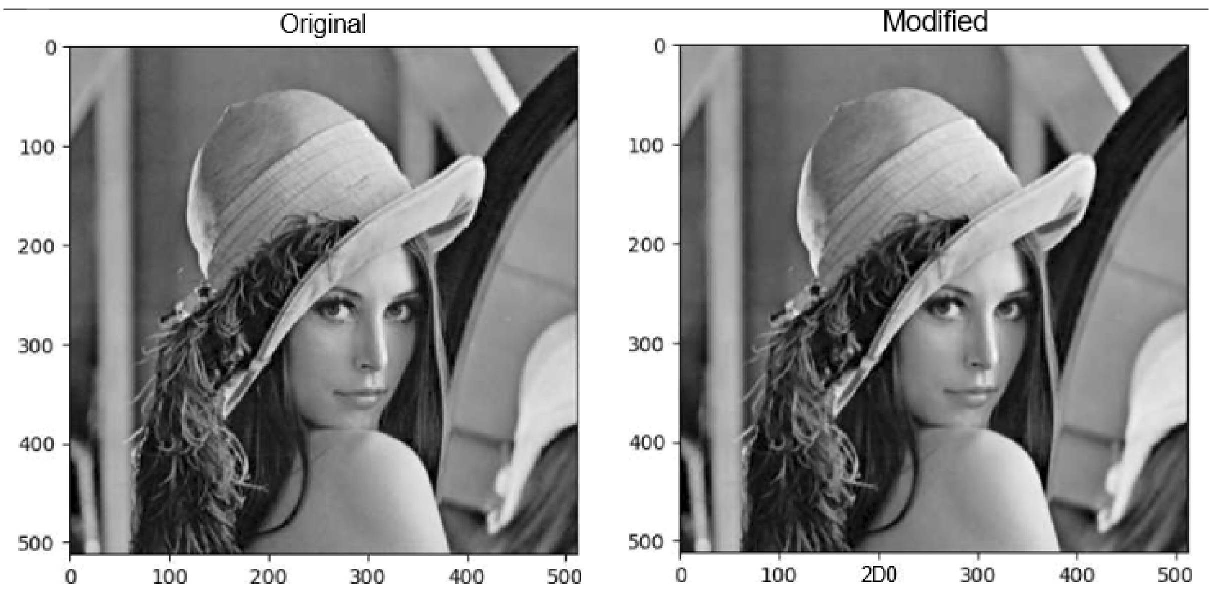
```
            median = np.median(s_xy).astype(int)


        image_nd[x, y] = median

    return image_nd

for_each_imgs(median_numpy)
```

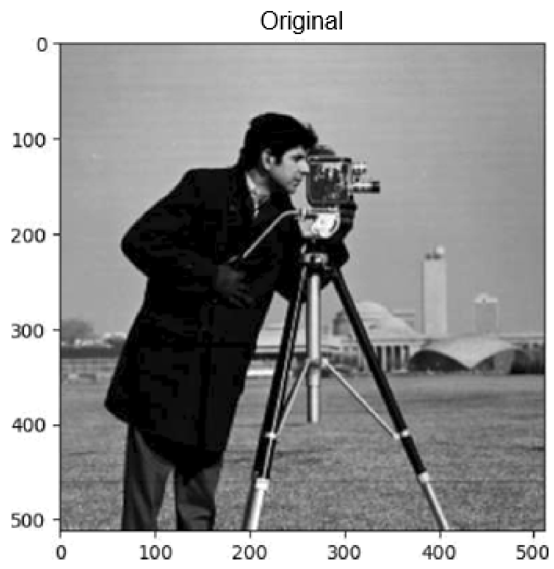


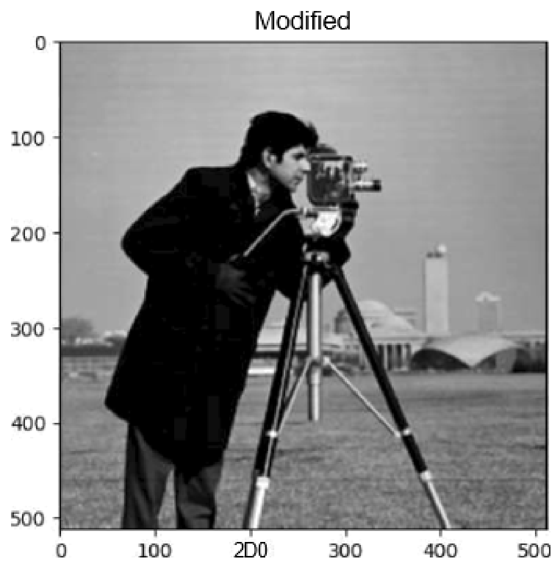


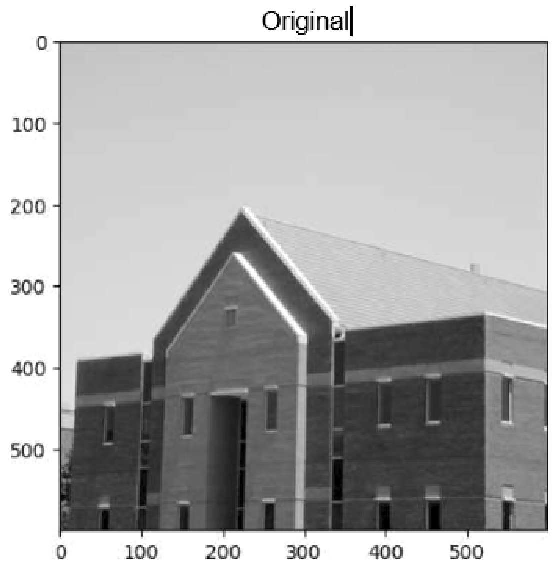
Pillow

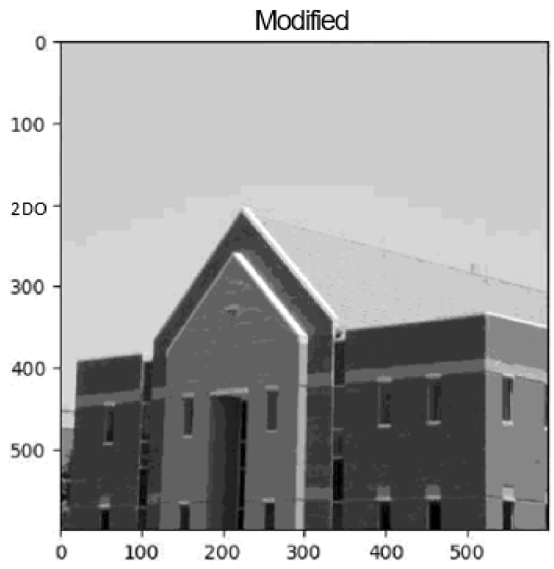
```
def median_pillow(img) -> np.ndarray:
    return img.filter(ImageFilter.MedianFilter(kernel))


for_each_imgs(median_pillow)
```





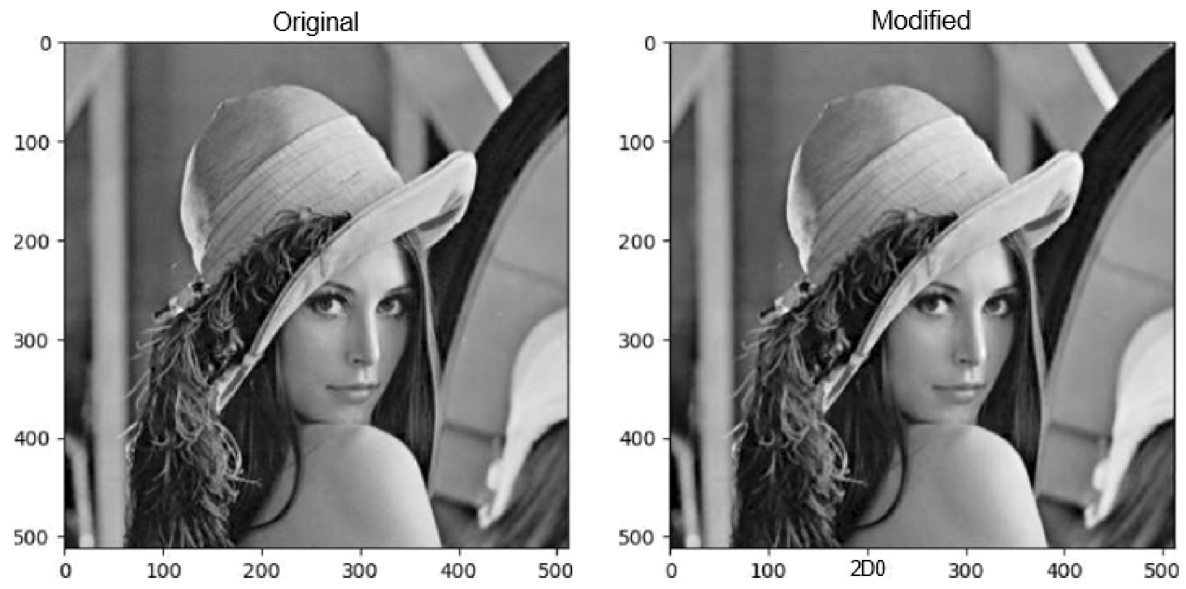


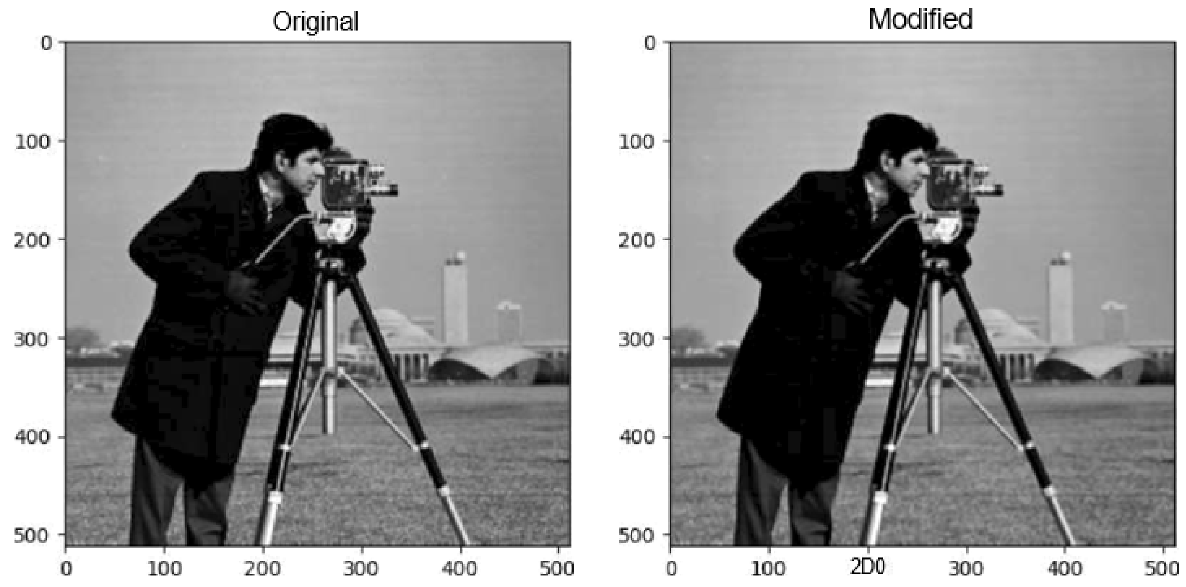


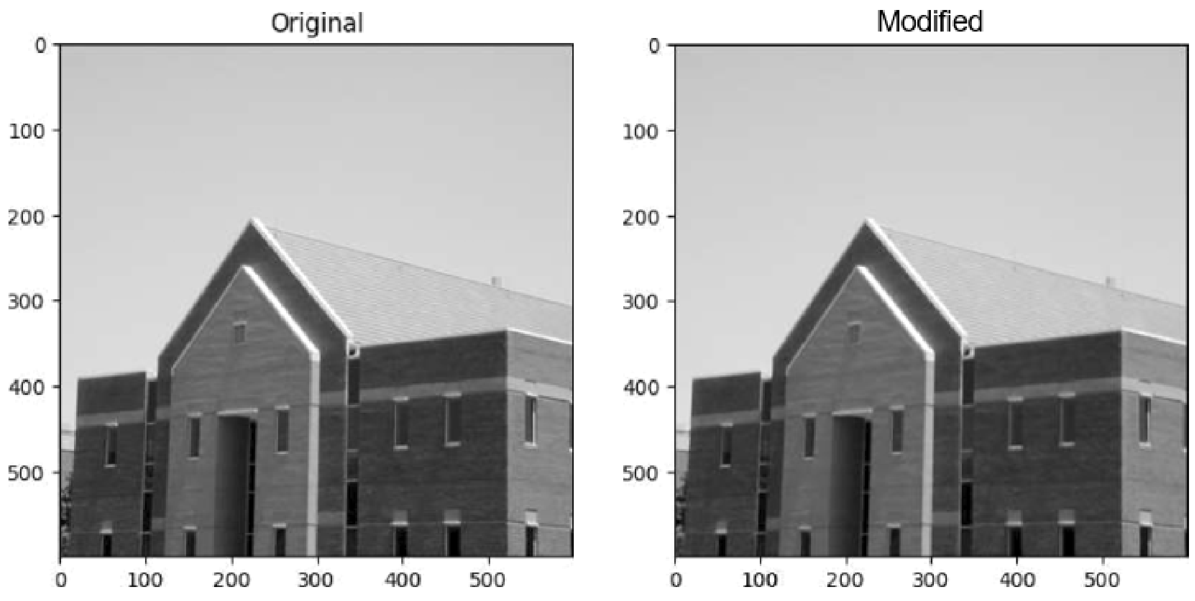


OpenCV

```
def median_opencv(img) -> np.ndarray:
    img_nd = np.array(img)
    return cv2.medianBlur(img_nd, kernel)

for_each_imgs(median_opencv)
```


Original
Modified
0
100
200
300
400
500
2D0


Original
Modified
0
100
200
300
400
500
2D0


Original
Modified
0
100
200
300
400
500