INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus Birigui	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA Campus Birigui** **Bacharelado em Engenharia de Computação**	
Disciplina: Processamento Digital de Imagem	Relatório – Transformações básicas	
Professor: Murilo Varges		**Data:** 21/08/2023
Nome do Aluno: Raul Prado Dantas		**Prontuário:** BI3007669

Operação De Vizinhança

Birigui 2023

Operação-vizinhança

August 29, 2023

[15]:
```
import numpy as np
from numpy import asarray
from PIL import Image, ImageFilter
import cv2
from scipy.signal import convolve2d, medfilt
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
```

1 [OPERAÇÃO VIZINHANÇA]

1.1 Utilizar kernel 3x3 pixels e desconsiderar pixels das extremidades

1.2 Calcular o filtro da média

1.3 Calcular o filtro da mediana

1.3.1 print_final_result() :: void

função responsável por mostrar o resultado final da imagem através da biblioteca matplotlib e aplicar as transformações geométricas na imagem original.

[16]:
```
def print_final_result(img_path1, img_path2, img_path3, title, function,lib,␣
 ↪kernel_size = 3):

    imgLena = Image.open(img_path1)
    f_imgLena = asarray(imgLena)
    imgCameraman = Image.open(img_path2)
    f_imgCameraman = asarray(imgCameraman)
    imgHouse = Image.open(img_path3)
    f_imgHouse = asarray(imgHouse)

    print(title)

    plt1 = plt.subplot(1,3,1)
    plt2 = plt.subplot(1,3,2)
    plt3 = plt.subplot(1,3,3)

    plt1.set_title('Lena')
    plt2.set_title('Cameraman')
```

```
    plt3.set_title('House')

    if(lib.lower()=='numpy'):


        if(title.find('MEDIA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        elif(title.find('MEDIANA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        else:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)


    elif (lib.lower() == 'pillow'):
        if(title.find('MEDIA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        elif(title.find('MEDIANA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        else:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)


    elif (lib.lower() == 'opencv'):
        if(title.find('MEDIA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        elif(title.find('MEDIANA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        else:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
```

```
    elif (lib.lower() == 'scipy'):
        if(title.find('MEDIA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        elif(title.find('MEDIANA')) != -1:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
        else:
            plt1.imshow(function(imgLena), cmap='gray')
            plt2.imshow(function(imgCameraman), cmap='gray', vmin=0, vmax=255)
            plt3.imshow(function(imgHouse), cmap='gray', vmin=0, vmax=255)
    else:
        print('Biblioteca não encontrada')

    plt.show()
    plt.figure()
    print()
```

2 Funções de tranformação de vizinhança utilizando a biblioteca Numpy

2.1 Funções auxiliares

2.1.1 neighborhoodMedian(kernelSize, f_img) :: g_image

função responsável por calcular a mediana dos pixels da imagem de entrada em torno de um kernel de tamanho kernelSize e retornar a imagem resultante.

2.1.2 neighborhoodMean(kernelSize, f_img) :: g_image

função responsável por calcular a média dos pixels da imagem de entrada em torno de um kernel de tamanho kernelSize e retornar a imagem resultante.

2.1.3 numpyFilter() :: void

função responsável por aplicar os filtros de média e mediana na imagem de entrada e mostrar o resultado final da imagem através da biblioteca matplotlib.

[17]:
```
def neighbourhoodMedian(kernelSize, f_img):
    l = f_img.shape[0]
    c = f_img.shape[1]
    g_img = np.zeros(f_img.shape)

    for x in range (kernelSize, l-kernelSize):
        for y in range (kernelSize, c-kernelSize):
```

```
            g_img[x,y] = np.median(f_img[x-kernelSize:x+kernelSize+1,␣
 ↪y-kernelSize:y+kernelSize+1]).astype(int)

    return g_img

def neighbourhoodMean(kernelSize, f_img):
    l = f_img.shape[0]
    c = f_img.shape[1]
    g_img = np.zeros(f_img.shape)

    for x in range (kernelSize, l-kernelSize):
        for y in range (kernelSize, c-kernelSize):
            g_img[x,y] = np.mean(f_img[x-kernelSize:x+kernelSize+1,␣
 ↪y-kernelSize:y+kernelSize+1]).astype(int)

    return g_img

def numpyFilter():
    imgLena = Image.open('lena_gray_512.tif')
    f_imgLena = np.array(imgLena)

    imgCameraman = Image.open('cameraman.tif')
    f_imgCameraman = np.array(imgCameraman)

    imgHouse = Image.open('house.tif')
    f_imgHouse = np.array(imgHouse)

    print("FILTROS COM NUMPY")
    print()
    print()
    print_final_result('lena_gray_512.tif', 'cameraman.tif', 'house.tif',␣
 ↪'Imagens originais', lambda x: x, 'numpy')

    k = 5

    g_imgLena = neighbourhoodMedian(k, f_imgLena)
    g_imgCameraman = neighbourhoodMedian(k, f_imgCameraman)
    g_imgHouse = neighbourhoodMedian(k, f_imgHouse)

    print("Imagens com filtro de MEDIANA")

    plt1 = plt.subplot(1,3,1)
    plt2 = plt.subplot(1,3,2)
    plt3 = plt.subplot(1,3,3)

    plt1.title.set_text("Lena")
    plt2.title.set_text("Cameraman")
```

```
    plt3.title.set_text("House")

    plt1.imshow(g_imgLena, cmap='gray')
    plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)

    plt.show()
    plt.figure()

    g_imgLena = neighbourhoodMean(k, f_imgLena)
    g_imgCameraman = neighbourhoodMean(k, f_imgCameraman)
    g_imgHouse = neighbourhoodMean(k, f_imgHouse)

    print("Imagens com filtro de MEDIA")

    plt1 = plt.subplot(1,3,1)
    plt2 = plt.subplot(1,3,2)
    plt3 = plt.subplot(1,3,3)

    plt1.title.set_text("Lena")
    plt2.title.set_text("Cameraman")
    plt3.title.set_text("House")

    plt1.imshow(g_imgLena, cmap='gray')
    plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)

    plt.show()
    plt.figure()
```

3 Funções de tranformação de vizinhança utilizando a biblioteca Pillow

3.1 Funções auxiliares

3.1.1 pillowFilter() :: void

função responsável por aplicar os filtros de média e mediana na imagem de entrada e mostrar o resultado final da imagem através da biblioteca matplotlib.

[18]:
```
def pillowFilter():
    f_imgLena = Image.open('lena_gray_512.tif')

    f_imgCameraman = Image.open('cameraman.tif')

    f_imgHouse = Image.open('house.tif')
```

```
print("FILTROS COM PILLOW")
print()
print()
print_final_result('lena_gray_512.tif', 'cameraman.tif', 'house.tif',␣
↪'Imagens originais', lambda x: x, 'numpy')

# neighbourhood operation
# kernel size
k = 5

g_imgLena = f_imgLena.filter(ImageFilter.BoxBlur(5))
g_imgCameraman = f_imgCameraman.filter(ImageFilter.BoxBlur(5))
g_imgHouse = f_imgHouse.filter(ImageFilter.BoxBlur(5))

print("Imagens com filtro de MEDIA")

plt1 = plt.subplot(1,3,1)
plt2 = plt.subplot(1,3,2)
plt3 = plt.subplot(1,3,3)

plt1.title.set_text("Lena")
plt2.title.set_text("Cameraman")
plt3.title.set_text("House")

plt1.imshow(g_imgLena, cmap='gray')
plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)

plt.show()
plt.figure()

g_imgLena = f_imgLena.filter(ImageFilter.MedianFilter(size=k))
g_imgCameraman = f_imgCameraman.filter(ImageFilter.MedianFilter(size=k))
g_imgHouse = f_imgHouse.filter(ImageFilter.MedianFilter(size=k))

print("Imagens com filtro de MEDIANA")

plt1 = plt.subplot(1,3,1)
plt2 = plt.subplot(1,3,2)
plt3 = plt.subplot(1,3,3)

plt1.title.set_text("Lena")
plt2.title.set_text("Cameraman")
plt3.title.set_text("House")

plt1.imshow(g_imgLena, cmap='gray')
```

```
    plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)
    plt.show()
    plt.figure()
```

4 Funções de tranformação de vizinhança utilizando a biblioteca OpenCV

4.1 Funções auxiliares

4.1.1 opencvFilter() :: void

função responsável por aplicar os filtros de média e mediana na imagem de entrada e mostrar o resultado final da imagem através da biblioteca matplotlib.

[19]:
```
def openCVFilter():
    imgLena = cv2.imread('lena_gray_512.tif', 0)

    imgCameraman = cv2.imread('cameraman.tif', 0)

    imgHouse = cv2.imread('house.tif', 0)

    print("FILTROS COM OPENCV")
    print()
    print()
    print_final_result('lena_gray_512.tif', 'cameraman.tif', 'house.tif',␣
 ↪'Imagens originais', lambda x: x, 'numpy')

    # neighbourhood operation
    # kernel size
    k = 9

    g_imgLena = cv2.medianBlur(imgLena, k)
    g_imgCameraman = cv2.medianBlur(imgCameraman, k)
    g_imgHouse = cv2.medianBlur(imgHouse, k)

    print("Imagens com filtro de MEDIANA")

    plt1 = plt.subplot(1,3,1)
    plt2 = plt.subplot(1,3,2)
    plt3 = plt.subplot(1,3,3)

    plt1.title.set_text("Lena")
    plt2.title.set_text("Cameraman")
    plt3.title.set_text("House")
```

```
    plt1.imshow(g_imgLena, cmap='gray')
    plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)

    plt.show()
    plt.figure()

    k=10

    g_imgLena = cv2.blur(imgLena, (k,k))
    g_imgCameraman = cv2.blur(imgCameraman, (k,k))
    g_imgHouse = cv2.blur(imgHouse, (k,k))

    print("Imagens com filtro de MEDIA")

    plt1 = plt.subplot(1,3,1)
    plt2 = plt.subplot(1,3,2)
    plt3 = plt.subplot(1,3,3)

    plt1.title.set_text("Lena")
    plt2.title.set_text("Cameraman")
    plt3.title.set_text("House")

    plt1.imshow(g_imgLena, cmap='gray')
    plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
    plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)
    plt.show()
    plt.figure()
```

5 Funções de tranformação de vizinhança utilizando a biblioteca Scipy

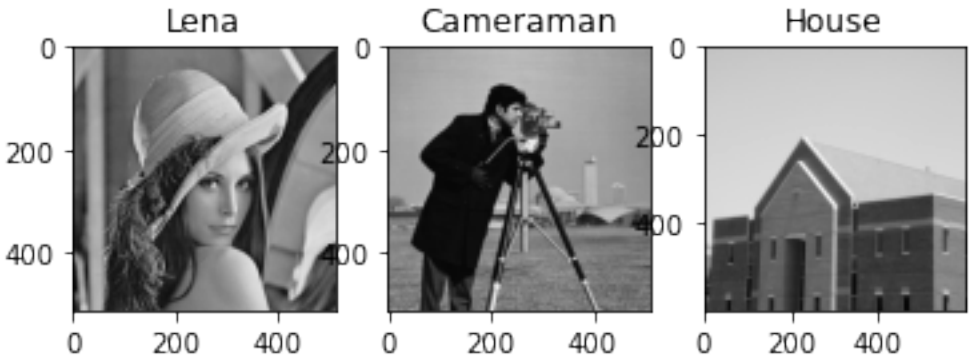
5.1 Funções auxiliares

[20]:
```
def scipyFilter():
    imgLena = cv2.imread('lena_gray_512.tif', 0)

    imgCameraman = cv2.imread('cameraman.tif', 0)

    imgHouse = cv2.imread('house.tif', 0)

    print("FILTROS COM SCIPY")
    print()
    print()
    print_final_result('lena_gray_512.tif', 'cameraman.tif', 'house.tif',␣
 ↪'Imagens originais', lambda x: x, 'numpy')
```

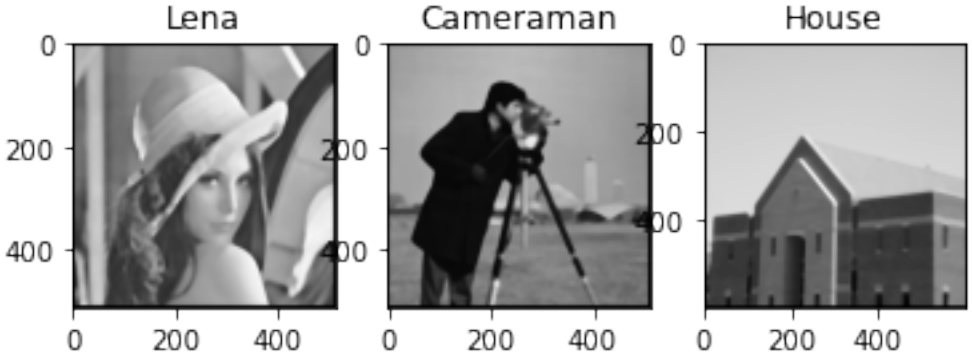
```
# neighbourhood operation
# kernel size
k = 9

g_imgLena = medfilt(imgLena, k)
g_imgCameraman = medfilt(imgCameraman, k)
g_imgHouse = medfilt(imgHouse, k)

print("Imagens com filtro de MEDIANA")

plt1 = plt.subplot(1,3,1)
plt2 = plt.subplot(1,3,2)
plt3 = plt.subplot(1,3,3)

plt1.title.set_text("Lena")
plt2.title.set_text("Cameraman")
plt3.title.set_text("House")

plt1.imshow(g_imgLena, cmap='gray')
plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)

plt.show()
plt.figure()

k=10

g_imgLena = convolve2d(imgLena, np.ones((k,k))/k**2, mode='same')
g_imgCameraman = convolve2d(imgCameraman, np.ones((k,k))/k**2, mode='same')
g_imgHouse = convolve2d(imgHouse, np.ones((k,k))/k**2, mode='same')

print("Imagens com filtro de MEDIA")

plt1 = plt.subplot(1,3,1)
plt2 = plt.subplot(1,3,2)
plt3 = plt.subplot(1,3,3)

plt1.title.set_text("Lena")
plt2.title.set_text("Cameraman")
plt3.title.set_text("House")

plt1.imshow(g_imgLena, cmap='gray')
plt2.imshow(g_imgCameraman, cmap='gray', vmin=0, vmax=255)
plt3.imshow(g_imgHouse, cmap='gray', vmin=0, vmax=255)
plt.show()
plt.figure()
```

[21]:
```
if __name__ == "__main__":
    numpyFilter()
    pillowFilter()
    openCVFilter()
    scipyFilter()
```

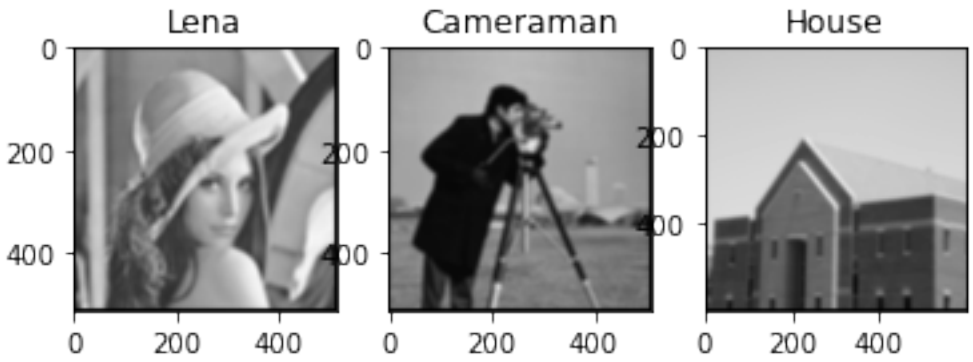
```
FILTROS COM NUMPY


Imagens originais
```




```
Imagens com filtro de MEDIANA
```

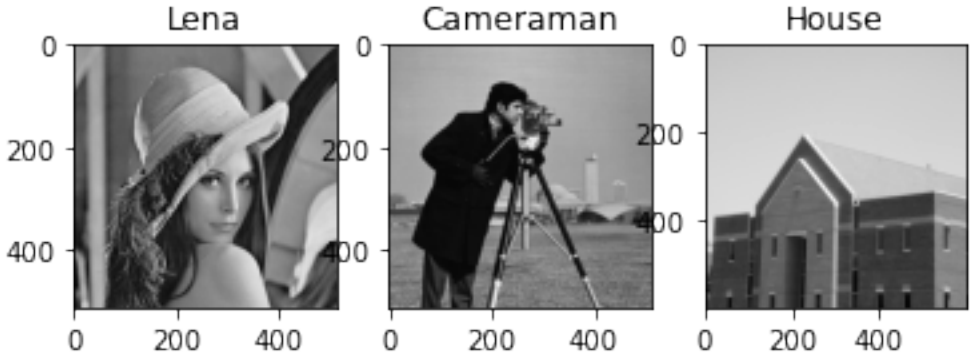



```
Imagens com filtro de MEDIA
```

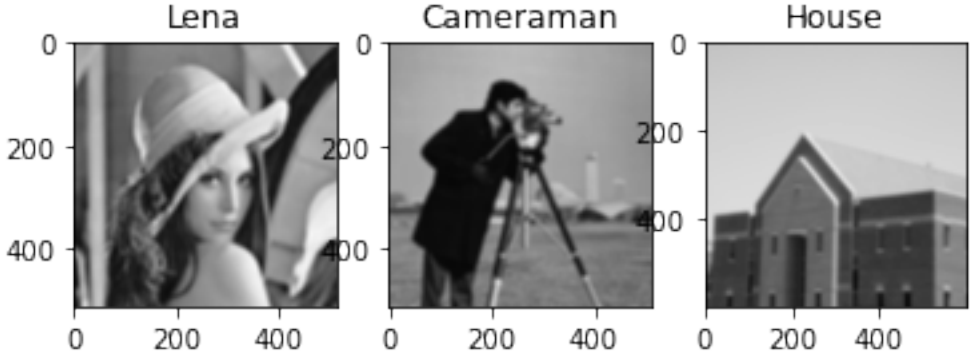


FILTROS COM PILLOW

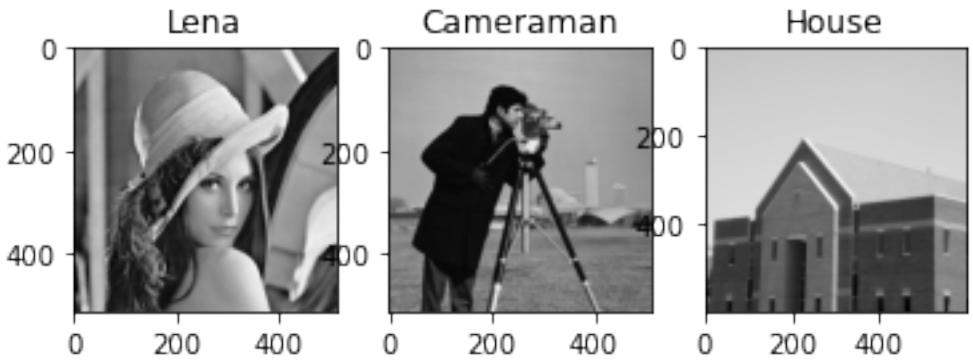
Imagens originais



Imagens com filtro de MEDIA

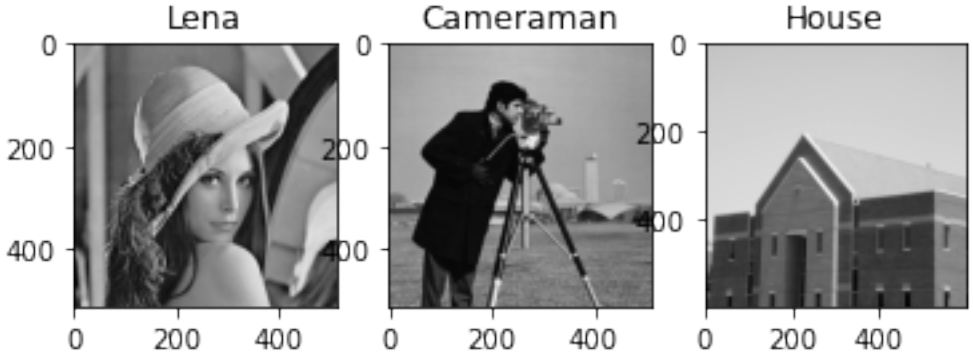


Imagens com filtro de MEDIANA

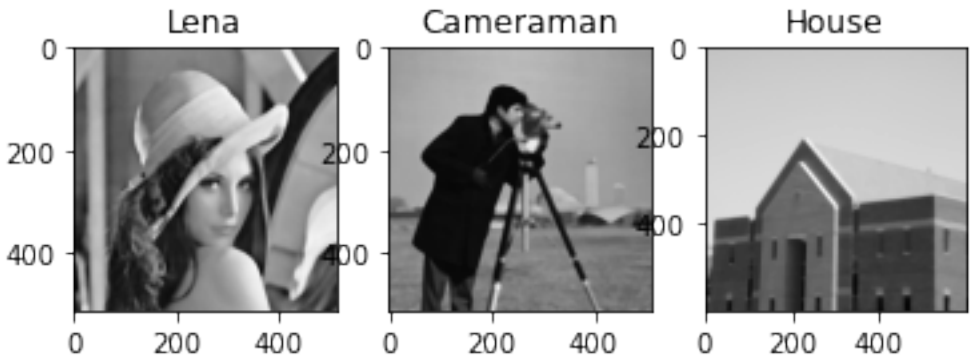


FILTROS COM OPENCV

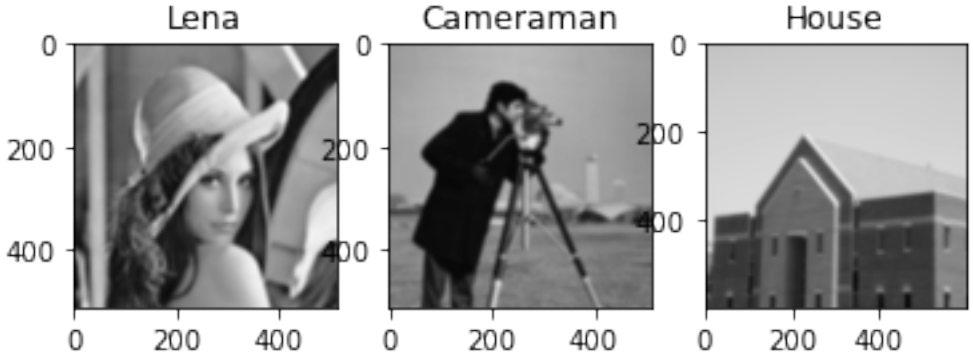
Imagens originais



Imagens com filtro de MEDIANA

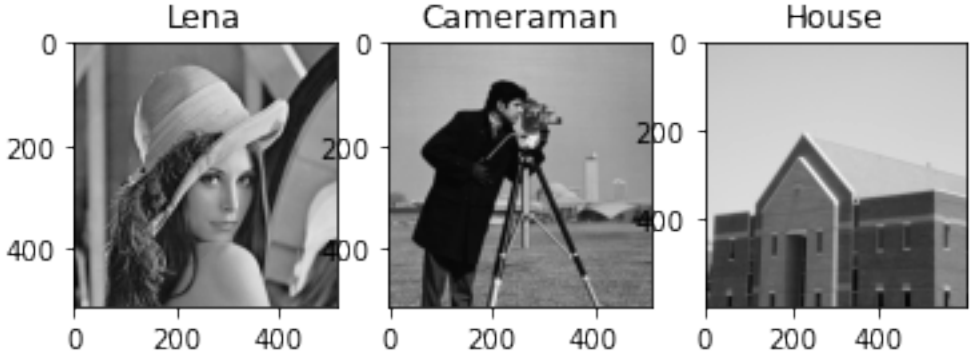


Imagens com filtro de MEDIA

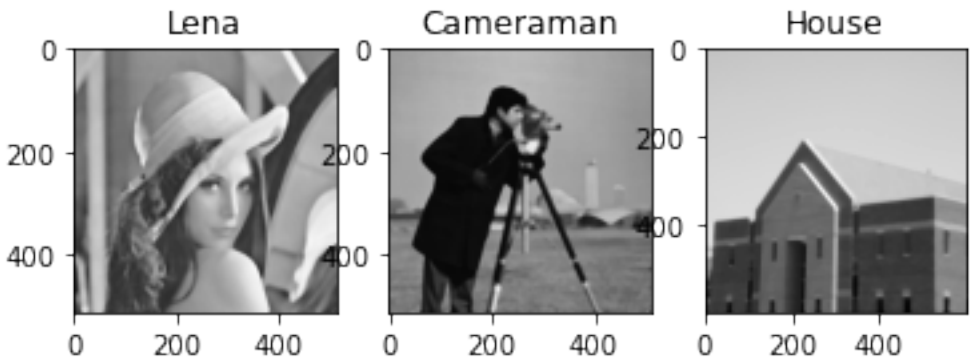


FILTROS COM SCIPY

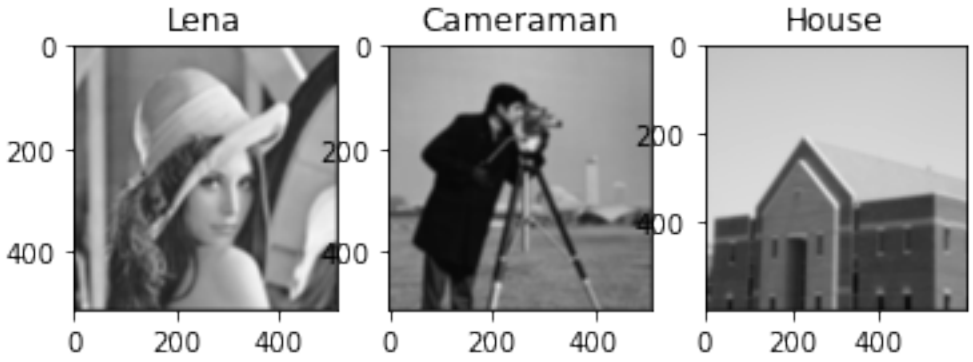
Imagens originais



Imagens com filtro de MEDIANA



Imagens com filtro de MEDIA



<Figure size 432x288 with 0 Axes>