

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SÃO PAULO, CÂMPUS BIRIGUI - SP
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO**

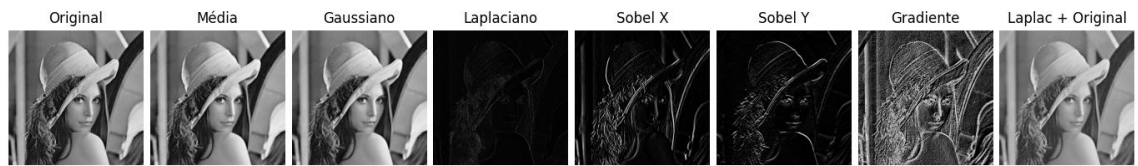
ISADORA DISPOSTI BUENO DOS SANTOS

EXERCÍCIOS – FILTRAGEM ESPACIAL

BIRIGUI - SP

2023

- Implementar a operação de convolução.



Código:

```
import cv2
import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt

import cv2
import numpy as np
import matplotlib.pyplot as plt

def convolucao_manual(imagem, kernel):
    if len(imagem.shape) == 3:
        altura, largura, canais = imagem.shape
    else:
        altura, largura = imagem.shape
        canais = 1
        imagem = imagem[:, :, np.newaxis]

    k_altura, k_largura = kernel.shape
    padding_altura = k_altura // 2
```

```

padding_largura = k_largura // 2

imagem_padded = np.pad(imagem,
((padding_altura, padding_altura),
(padding_largura, padding_largura), (0, 0)),
mode='constant')
saida = np.zeros_like(imagem)

for canal in range(canais):
    for y in range(altura):
        for x in range(largura):
            saida[y, x, canal] =
np.sum(imagem_padded[y:y + k_altura, x:x +
k_largura, canal] * kernel)

if canais == 1:
    saida = saida[:, :, 0]

return saida

media = np.ones((3, 3)) / 9
gaussiano = cv2.getGaussianKernel(5, 1) *
cv2.getGaussianKernel(5, 1).T
laplaciano = np.array([[0, 1, 0], [1, -4, 1], [0,
1, 0]])
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0,
1]])
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2,
1]])

#Carregando as imagens
imagens_info = {
    "lena": {
        "path": "lena.tif",

```

```

        "data": None
    },
    "biel": {
        "path": "biel.png",
        "data": None
    },
    "cameraman": {
        "path": "cameraman.tif",
        "data": None
    }
}
diretorio = "./imagem/"

for nome, info in imagens_info.items():
    imagem = cv2.imread(diretorio + info["path"],
cv2.IMREAD_GRAYSCALE)
    imagens_info[nome]["data"] = imagem

for nome, info in imagens_info.items():
    imagem = info["data"]

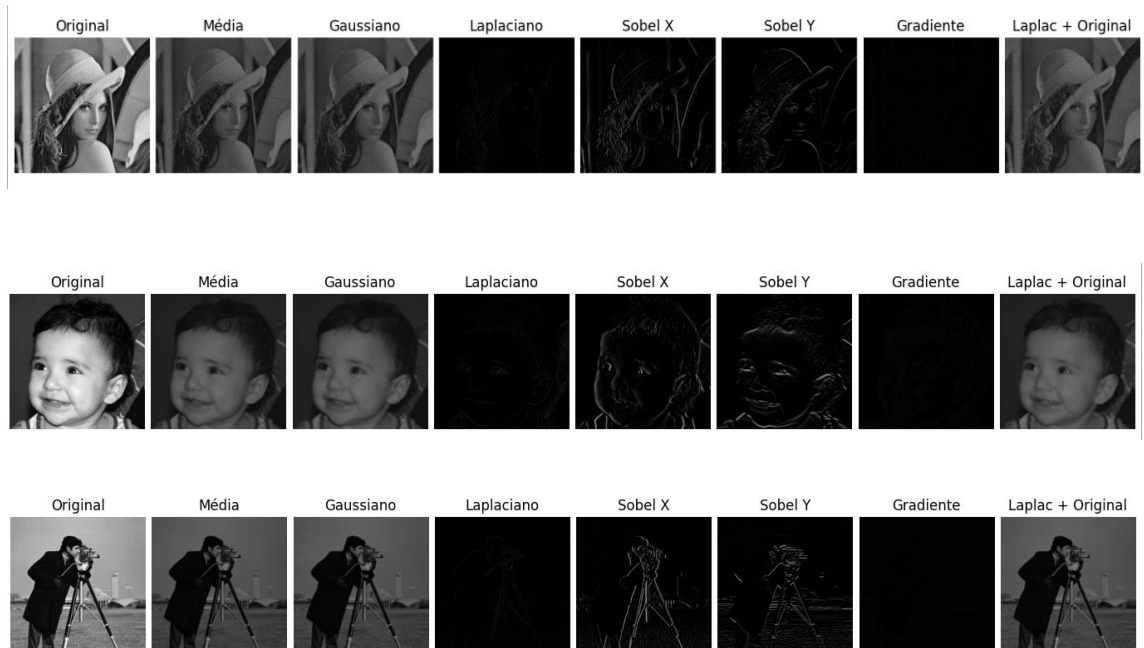
    imagem_media = convolucao_manual(imagem, media)
    imagem_gauss = convolucao_manual(imagem,
gaussiano)
    imagem_laplac = convolucao_manual(imagem,
laplaciano)
    imagem_sobel_x = convolucao_manual(imagem,
sobel_x)
    imagem_sobel_y = convolucao_manual(imagem,
sobel_y)
    imagem_gradiente = np.sqrt(imagem_sobel_x*2 +
imagem_sobel_y*2)
    imagem_laplac_original = imagem + imagem_laplac

```

```
fig, axs = plt.subplots(1, 8, figsize=(25, 5))
axs[0].imshow(imagem, cmap='gray', vmin=0,
vmax=500)
axs[0].set_title('Original')
axs[0].axis('off')
axs[1].imshow(imagem_media, cmap='gray',
vmin=0, vmax=500)
axs[1].set_title('Média')
axs[1].axis('off')
axs[2].imshow(imagem_gauss, cmap='gray',
vmin=0, vmax=500)
axs[2].set_title('Gaussiano')
axs[2].axis('off')
axs[3].imshow(imagem_laplac, cmap='gray',
vmin=0, vmax=500)
axs[3].set_title('Laplaciano')
axs[3].axis('off')
axs[4].imshow(imagem_sobel_x, cmap='gray',
vmin=0, vmax=500)
axs[4].set_title('Sobel X')
axs[4].axis('off')
axs[5].imshow(imagem_sobel_y, cmap='gray',
vmin=0, vmax=500)
axs[5].set_title('Sobel Y')
axs[5].axis('off')
axs[6].imshow(imagem_gradiente, cmap='gray',
vmin=0, vmax=500)
axs[6].set_title('Gradiente')
axs[6].axis('off')
axs[7].imshow(imagem_laplac_original,
cmap='gray', vmin=0, vmax=500)
axs[7].set_title('Laplac + Original')
axs[7].axis('off')
```

```
plt.tight_layout()
plt.show()
```

- Utilizando OPENCV, scipy função convolve e implementação manual.
 - OPENCV



Código:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Função de Convolução usando o OpenCV
def convolucao_opencv(imagem, kernel):
    return cv2.filter2D(imagem, -1, kernel)

media = np.ones((3, 3)) / 9
gaussiano = cv2.getGaussianKernel(5, 1) *
cv2.getGaussianKernel(5, 1).T
laplaciano = np.array([[0, 1, 0], [1, -4, 1], [0,
1, 0]])
```

```
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
```

```
imagens_info = {
    "lena": {
        "path": "lena_gray_512.tif",
        "data": None
    },
    "biel": {
        "path": "biel.png",
        "data": None
    },
    "cameraman": {
        "path": "cameraman.tif",
        "data": None
    }
}
```

```
diretorio = "./imagem/"
```

```
for nome, info in imagens_info.items():
    imagem = cv2.imread(diretorio + info["path"],
cv2.IMREAD_GRAYSCALE)
    imagens_info[nome]["data"] = imagem
```

```
for nome, info in imagens_info.items():
    imagem = info["data"]

    imagem_media = convolucao_opencv(imagem, media)
    imagem_gauss = convolucao_opencv(imagem,
gaussiano)
```

```
    imagem_laplac = convolucao_opencv(imagem,
laplaciano)
    imagem_sobel_x = convolucao_opencv(imagem,
sobel_x)
    imagem_sobel_y = convolucao_opencv(imagem,
sobel_y)
    imagem_gradiente = np.sqrt(imagem_sobel_x **2 +
imagem_sobel_y **2)
    imagem_laplac_original = imagem + imagem_laplac

fig, axs = plt.subplots(1, 8, figsize=(25, 5))
axs[0].imshow(imagem, cmap='gray')
axs[0].set_title('Original')
axs[0].axis('off')
axs[1].imshow(imagem_media, cmap='gray',vmin=0,
vmax=500)
axs[1].set_title('Média')
axs[1].axis('off')
axs[2].imshow(imagem_gauss, cmap='gray',
vmin=0, vmax=500)
axs[2].set_title('Gaussiano')
axs[2].axis('off')
axs[3].imshow(imagem_laplac, cmap='gray',
vmin=0, vmax=500)
axs[3].set_title('Laplaciano')
axs[3].axis('off')
axs[4].imshow(imagem_sobel_x, cmap='gray',
vmin=0, vmax=500)
axs[4].set_title('Sobel X')
axs[4].axis('off')
axs[5].imshow(imagem_sobel_y, cmap='gray',
vmin=0, vmax=500)
axs[5].set_title('Sobel Y')
axs[5].axis('off')
```



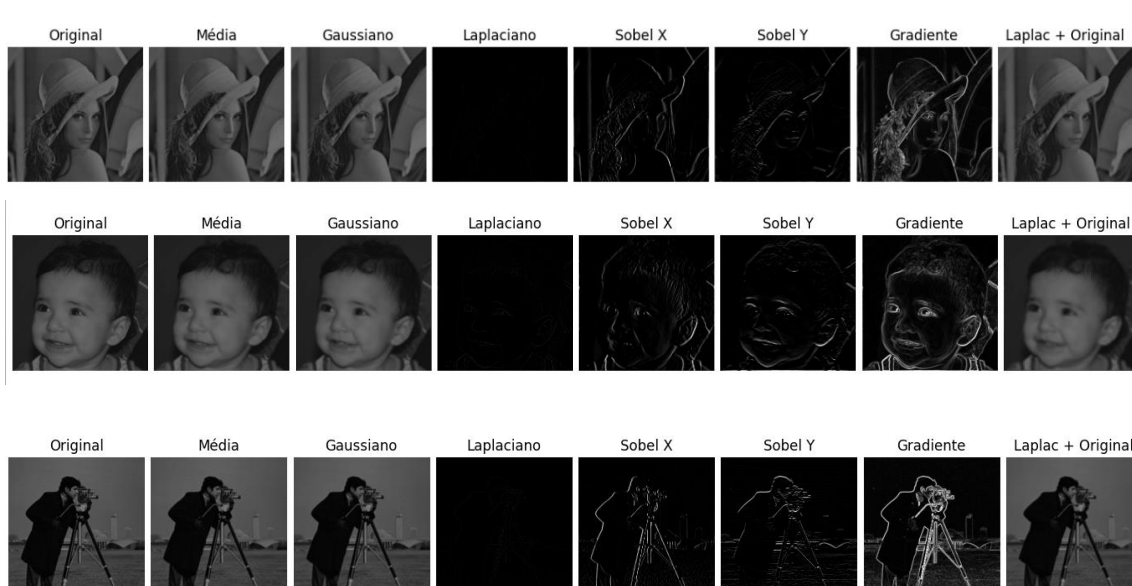
```

    axs[6].imshow(imagem_gradiente, cmap='gray',
vmin=0, vmax=500)
    axs[6].set_title('Gradiente')
    axs[6].axis('off')
    axs[7].imshow(imagem_laplac_original,
cmap='gray', vmin=0, vmax=500)
    axs[7].set_title('Laplac + Original')
    axs[7].axis('off')

plt.tight_layout()
plt.show()

```

- o SCIPY



Código:

```

import cv2
import numpy as np
from scipy.signal import gaussian, convolve2d
import matplotlib.pyplot as plt

# Média
media = np.ones((3, 3)) / 9

```

```

# Gaussiano
def gaussian_kernel(size: int, sigma: float):
    """ Retorna um kernel gaussiano usando scipy.
    """
    gauss_1d = gaussian(size, sigma)
    gauss_2d = np.outer(gauss_1d, gauss_1d)
    return gauss_2d / gauss_2d.sum()

gaussiano = gaussian_kernel(5, 1)

# Laplaciano
laplaciano = np.array([[0, 1, 0], [1, -4, 1], [0,
1, 0]])

# Sobel X
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0,
1]])

# Sobel Y
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2,
1]])

def convolucao_scipy(imagem, kernel):
    return convolve2d(imagem, kernel, mode='same',
boundary='wrap')

imagens_info = {
    "lena": {
        "path": "lena_gray_512.tif",
        "data": None
    },
    "biel": {
        "path": "biel.png",
        "data": None
    }
}

```

```

    },
    "cameraman": {
        "path": "cameraman.tif",
        "data": None
    }
}

diretorio = "./imagem/"

for nome, info in imagens_info.items():
    imagem = cv2.imread(diretorio + info["path"],
cv2.IMREAD_GRAYSCALE)
    imagens_info[nome]["data"] = imagem

for nome, info in imagens_info.items():
    imagem = info["data"]

    imagem_media = convolucao_scipy(imagem, media)
    imagem_gauss = convolucao_scipy(imagem,
gaussiano)
    imagem_laplac = convolucao_scipy(imagem,
laplaciano)
    imagem_sobel_x = convolucao_scipy(imagem,
sobel_x)
    imagem_sobel_y = convolucao_scipy(imagem,
sobel_y)
    imagem_gradiente = np.sqrt(imagem_sobel_x**2 +
imagem_sobel_y**2)
    imagem_laplac_original = imagem + imagem_laplac

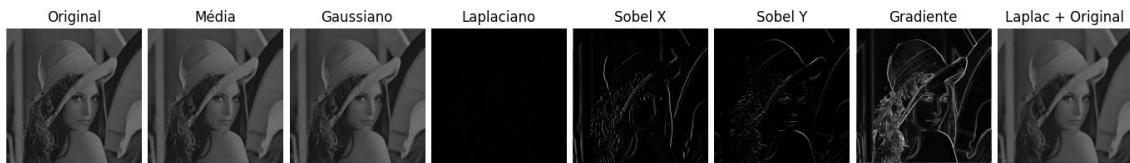
    fig, axs = plt.subplots(1, 8, figsize=(25, 5))
    axs[0].imshow(imagem, cmap='gray', vmin=0,
vmax=500)
    axs[0].set_title('Original')

```

```
    axs[0].axis('off')
    axs[1].imshow(imagem_media, cmap='gray',
vmin=0, vmax=500)
    axs[1].set_title('Média')
    axs[1].axis('off')
    axs[2].imshow(imagem_gauss, cmap='gray',
vmin=0, vmax=500)
    axs[2].set_title('Gaussiano')
    axs[2].axis('off')
    axs[3].imshow(imagem_laplac, cmap='gray',
vmin=0, vmax=500)
    axs[3].set_title('Laplaciano')
    axs[3].axis('off')
    axs[4].imshow(imagem_sobel_x, cmap='gray',
vmin=0, vmax=500)
    axs[4].set_title('Sobel X')
    axs[4].axis('off')
    axs[5].imshow(imagem_sobel_y, cmap='gray',
vmin=0, vmax=500)
    axs[5].set_title('Sobel Y')
    axs[5].axis('off')
    axs[6].imshow(imagem_gradiente, cmap='gray',
vmin=0, vmax=500)
    axs[6].set_title('Gradiente')
    axs[6].axis('off')
    axs[7].imshow(imagem_laplac_original,
cmap='gray', vmin=0, vmax=500)
    axs[7].set_title('Laplac + Original')
    axs[7].axis('off')

plt.tight_layout()
plt.show()
```

- Implementação Manual



Código:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#Média
media = np.ones((3, 3)) / 9

# Gaussiano
def gaussian_kernel(size: int, sigma: float):
    """ Retorna um kernel gaussiano usando numpy.
    """
    x, y = np.mgrid[-size//2 + 1:size//2 + 1, -size//2 + 1:size//2 + 1]
    g = np.exp(-((x**2 + y**2) / (2.0 * sigma**2)))
    return g / g.sum()

gaussiano = gaussian_kernel(5, 1)
```

```

# Laplaciano
laplaciano = np.array([[0, 1, 0], [1, -4, 1], [0,
1, 0]])

# Sobel X
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0,
1]])

# Sobel Y
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2,
1]])

def convolucao_manual(imagem, kernel):
    altura, largura = imagem.shape
    k_altura, k_largura = kernel.shape

    padding_altura = k_altura // 2
    padding_largura = k_largura // 2

    imagem_padded = np.pad(imagem,
((padding_altura, padding_altura),
(padding_largura, padding_largura)),
mode='constant')
    saida = np.zeros(imagem.shape)

    for y in range(altura):
        for x in range(largura):
            saida[y, x] = np.sum(imagem_padded[y:y
+ k_altura, x:x + k_largura] * kernel)

    return saida

imagens_info = {
    "lena": {

```

```

        "path": "lena_gray_512.tif",
        "data": None
    },
    "biel": {
        "path": "biel.png",
        "data": None
    },
    "cameraman": {
        "path": "cameraman.tif",
        "data": None
    }
}

diretorio = "./imagem/"

for nome, info in imagens_info.items():
    imagem = cv2.imread(diretorio + info["path"],
cv2.IMREAD_GRAYSCALE)
    imagens_info[nome]["data"] = imagem

for nome, info in imagens_info.items():
    imagem = info["data"]

    imagem_media = convolucao_manual(imagem, media)
    imagem_gauss = convolucao_manual(imagem,
gaussiano)
    imagem_laplac = convolucao_manual(imagem,
laplaciano)
    imagem_sobel_x = convolucao_manual(imagem,
sobel_x)
    imagem_sobel_y = convolucao_manual(imagem,
sobel_y)
    imagem_gradiente = np.sqrt(imagem_sobel_x**2 +
imagem_sobel_y**2)

```

```
    imagem_laplac_original = imagem + imagem_laplac

    fig, axs = plt.subplots(1, 8, figsize=(25, 5))
    axs[0].imshow(imagem, cmap='gray', vmin=0,
vmax=500)
    axs[0].set_title('Original')
    axs[0].axis('off')
    axs[1].imshow(imagem_media, cmap='gray',
vmin=0, vmax=500)
    axs[1].set_title('Média')
    axs[1].axis('off')
    axs[2].imshow(imagem_gauss, cmap='gray',
vmin=0, vmax=500)
    axs[2].set_title('Gaussiano')
    axs[2].axis('off')
    axs[3].imshow(imagem_laplac, cmap='gray',
vmin=0, vmax=500)
    axs[3].set_title('Laplaciano')
    axs[3].axis('off')
    axs[4].imshow(imagem_sobel_x, cmap='gray',
vmin=0, vmax=500)
    axs[4].set_title('Sobel X')
    axs[4].axis('off')
    axs[5].imshow(imagem_sobel_y, cmap='gray',
vmin=0, vmax=500)
    axs[5].set_title('Sobel Y')
    axs[5].axis('off')
    axs[6].imshow(imagem_gradiente, cmap='gray',
vmin=0, vmax=500)
    axs[6].set_title('Gradiente')
    axs[6].axis('off')
    axs[7].imshow(imagem_laplac_original,
cmap='gray', vmin=0, vmax=500)
    axs[7].set_title('Laplac + Original')
```



```
axs[7].axis('off')  
  
plt.tight_layout()  
plt.show()
```

GitHub: <https://github.com/isadoradisposti/Filtragem-Espacial2>