Luiz Henrique Botega Cervantes bi3008479 Eng. 201

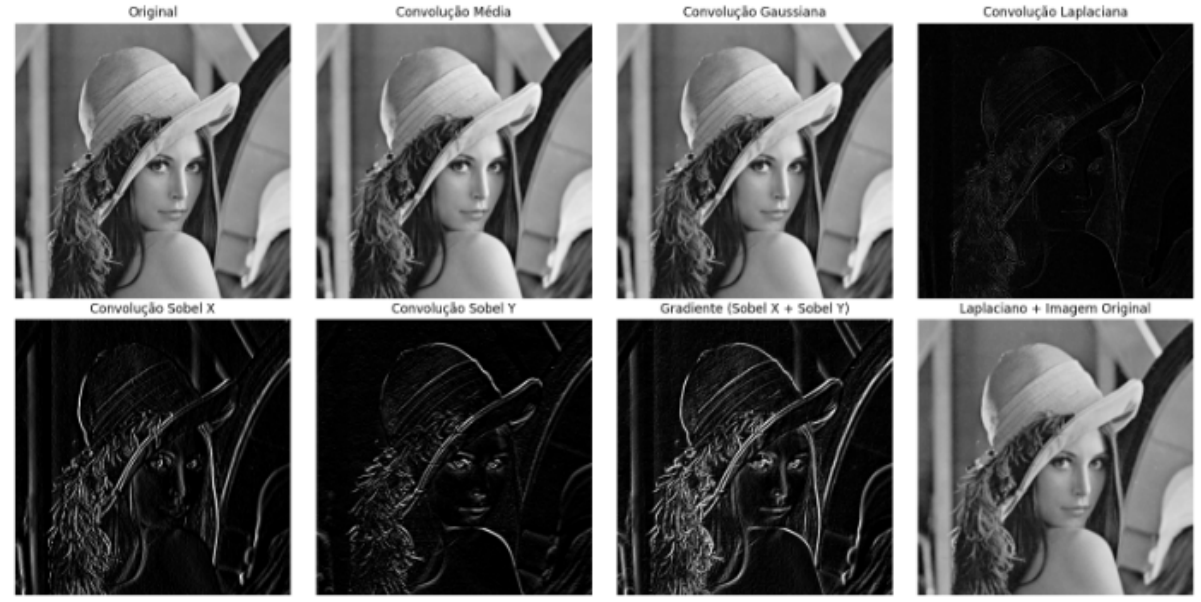
Atividade - Filtragem Espacial

Código com a implementação utilizando a biblioteca openCV

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

# Carregar a imagem de exemplo
image = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)

# Definir as máscaras
media_mask = np.ones((3, 3), dtype=np.float32) / 9.0
gaussian_mask = np.array([[1, 2, 1],
                          [2, 4, 2],
                          [1, 2, 1]], dtype=np.float32) / 16.0
laplacian_mask = np.array([[0, 1, 0],
                           [1, -4, 1],
                           [0, 1, 0]], dtype=np.float32)
sobel_x_mask = np.array([[-1, 0, 1],
                         [-2, 0, 2],
                         [-1, 0, 1]], dtype=np.float32)
sobel_y_mask = np.array([[-1, -2, -1],
                         [0, 0, 0],
                         [1, 2, 1]], dtype=np.float32)

# Aplicar convolução com OpenCV
conv_media = cv2.filter2D(image, -1, media_mask)
conv_gaussian = cv2.filter2D(image, -1, gaussian_mask)
conv_laplacian = cv2.filter2D(image, -1, laplacian_mask)
conv_sobel_x = cv2.filter2D(image, -1, sobel_x_mask)
conv_sobel_y = cv2.filter2D(image, -1, sobel_y_mask)

# Sobel Y
sobel_y_mask = np.array([[-1, -2, -1],
                         [0, 0, 0],
                         [1, 2, 1]], dtype=np.float32)
conv_sobel_y = cv2.filter2D(image, -1, sobel_y_mask)

# Gradiente (Sobel X + Sobel Y)
gradient = cv2.add(conv_sobel_x, conv_sobel_y)
```

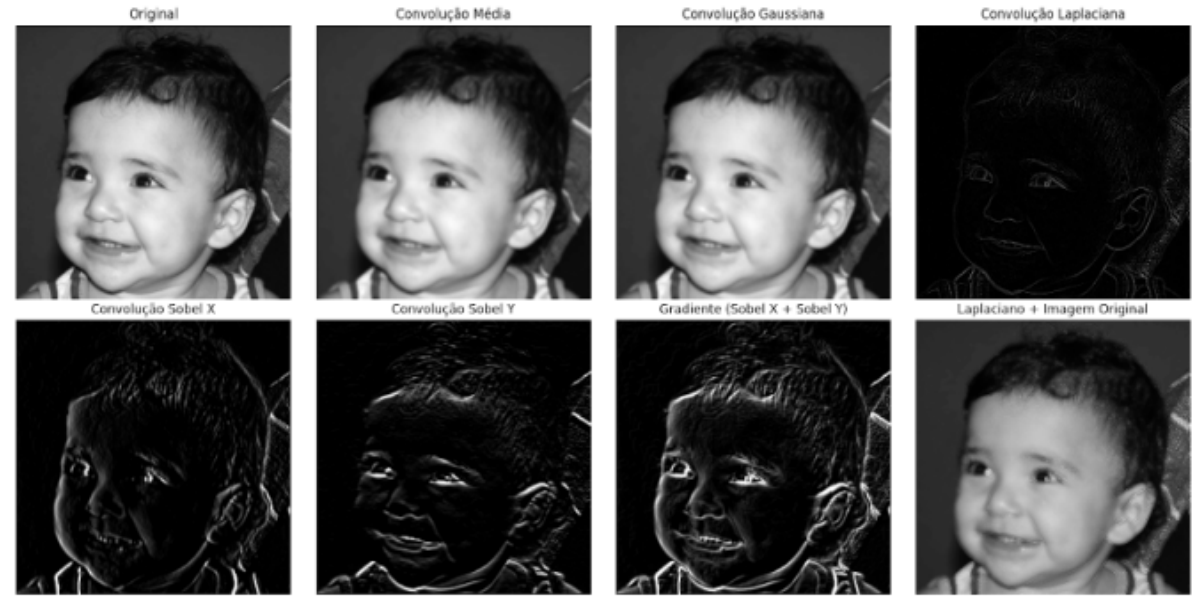
```
# Laplaciano somado à imagem original
laplacian_result = cv2.add(image, cv2.filter2D(image, -1,
laplacian_mask))

# Configurar a exibição lado a lado
fig, axs = plt.subplots(2, 4, figsize=(16, 8))

# Exibir imagens
axs[0, 0].imshow(image, cmap='gray')
axs[0, 0].set_title('Original')

axs[0, 1].imshow(conv_media, cmap='gray')
axs[0, 1].set_title('Convolução Média')

axs[0, 2].imshow(conv_gaussian, cmap='gray')
axs[0, 2].set_title('Convolução Gaussiana')

axs[0, 3].imshow(conv_laplacian, cmap='gray')
axs[0, 3].set_title('Convolução Laplaciana')

axs[1, 0].imshow(conv_sobel_x, cmap='gray')
axs[1, 0].set_title('Convolução Sobel X')

axs[1, 1].imshow(conv_sobel_y, cmap='gray')
axs[1, 1].set_title('Convolução Sobel Y')

axs[1, 2].imshow(gradient, cmap='gray')
axs[1, 2].set_title('Gradiente (Sobel X + Sobel Y)')

axs[1, 3].imshow(laplacian_result, cmap='gray')
axs[1, 3].set_title('Laplaciano + Imagem Original')

# Remover ticks dos eixos
for ax in axs.flat:
    ax.axis('off')

# Exibir as imagens lado a lado
plt.tight_layout()
plt.show()
```

Lena



Cameraman

Biel



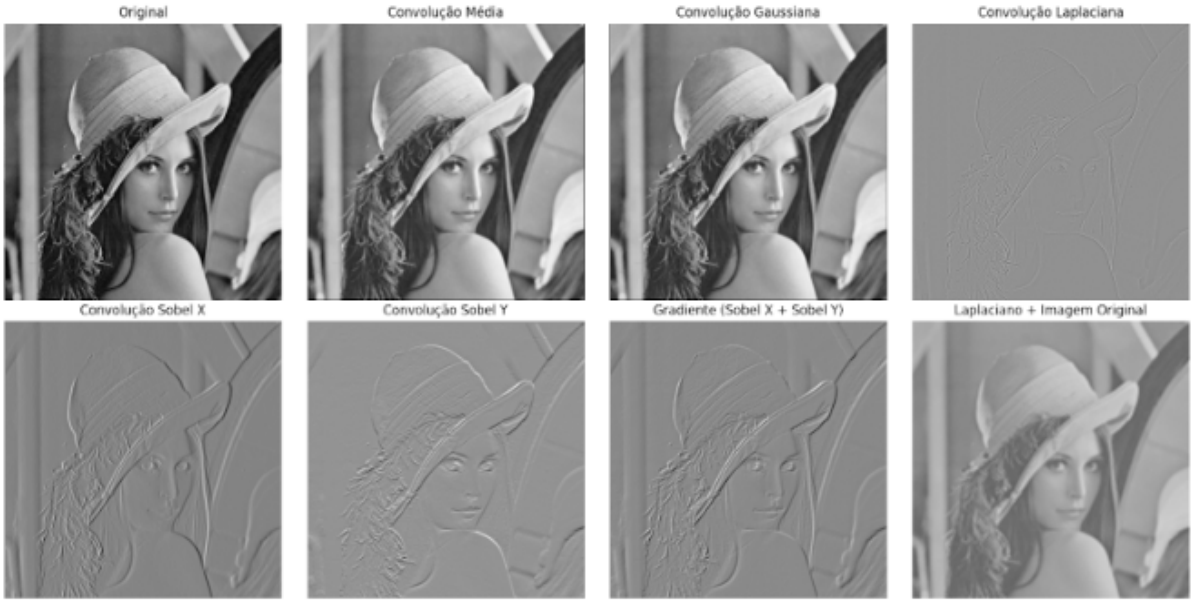
Código com a implementação manual

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


# Função para realizar a convolução manualmente
def convolution_manual(image, kernel):
    height, width = image.shape
    k_height, k_width = kernel.shape
    padding_height = k_height // 2
    padding_width = k_width // 2
    conv_result = np.zeros((height, width), dtype=np.float32)


    for i in range(padding_height, height - padding_height):
        for j in range(padding_width, width - padding_width):
            roi = image[i - padding_height : i + padding_height + 1, j
- padding_width : j + padding_width + 1]
            conv_result[i, j] = np.sum(roi * kernel)


    return conv_result
```

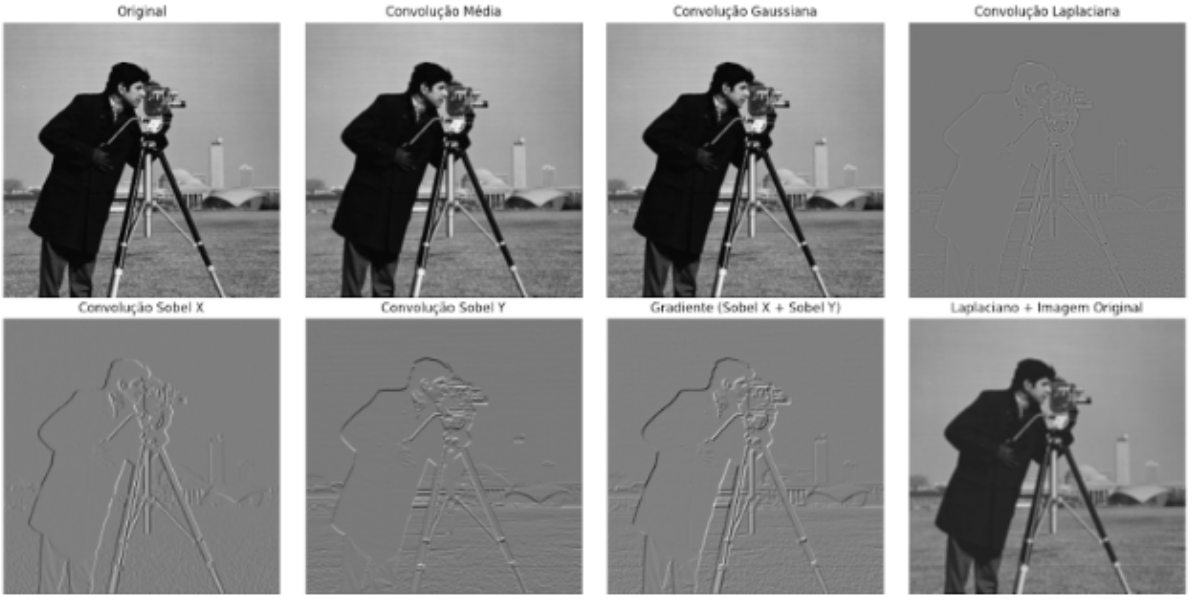
```
# Carregar a imagem de exemplo (substitua 'biel.jpg' pelo caminho da
imagem desejada)
image = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)


# Definir as máscaras
media_mask = np.ones((3, 3), dtype=np.float32) / 9.0
gaussian_mask = np.array([[1, 2, 1],
                          [2, 4, 2],
                          [1, 2, 1]], dtype=np.float32) / 16.0
laplacian_mask = np.array([[0, 1, 0],
                           [1, -4, 1],
                           [0, 1, 0]], dtype=np.float32)
sobel_x_mask = np.array([[-1, 0, 1],
                         [-2, 0, 2],
                         [-1, 0, 1]], dtype=np.float32)
sobel_y_mask = np.array([[-1, -2, -1],
                         [0, 0, 0],
                         [1, 2, 1]], dtype=np.float32)


# Aplicar convolução manual
conv_media = convolution_manual(image, media_mask)
conv_gaussian = convolution_manual(image, gaussian_mask)
conv_laplacian = convolution_manual(image, laplacian_mask)
conv_sobel_x = convolution_manual(image, sobel_x_mask)
conv_sobel_y = convolution_manual(image, sobel_y_mask)


# Gradiente (Sobel X + Sobel Y)
gradient = conv_sobel_x + conv_sobel_y


# Laplaciano somado à imagem original
laplacian_result = image + conv_laplacian


# Configurar a exibição lado a lado
fig, axs = plt.subplots(2, 4, figsize=(16, 8))


# Exibir imagens
axs[0, 0].imshow(image, cmap='gray')
```

```
axs[0, 0].set_title('Original')


axs[0, 1].imshow(conv_media, cmap='gray')
axs[0, 1].set_title('Convolução Média')


axs[0, 2].imshow(conv_gaussian, cmap='gray')
axs[0, 2].set_title('Convolução Gaussiana')


axs[0, 3].imshow(conv_laplacian, cmap='gray')
axs[0, 3].set_title('Convolução Laplaciana')


axs[1, 0].imshow(conv_sobel_x, cmap='gray')
axs[1, 0].set_title('Convolução Sobel X')


axs[1, 1].imshow(conv_sobel_y, cmap='gray')
axs[1, 1].set_title('Convolução Sobel Y')


axs[1, 2].imshow(gradient, cmap='gray')
axs[1, 2].set_title('Gradiente (Sobel X + Sobel Y)')


axs[1, 3].imshow(laplacian_result, cmap='gray')
axs[1, 3].set_title('Laplaciano + Imagem Original')


# Remover ticks dos eixos
for ax in axs.flat:
    ax.axis('off')


# Exibir as imagens lado a lado
plt.tight_layout()
plt.show()
```

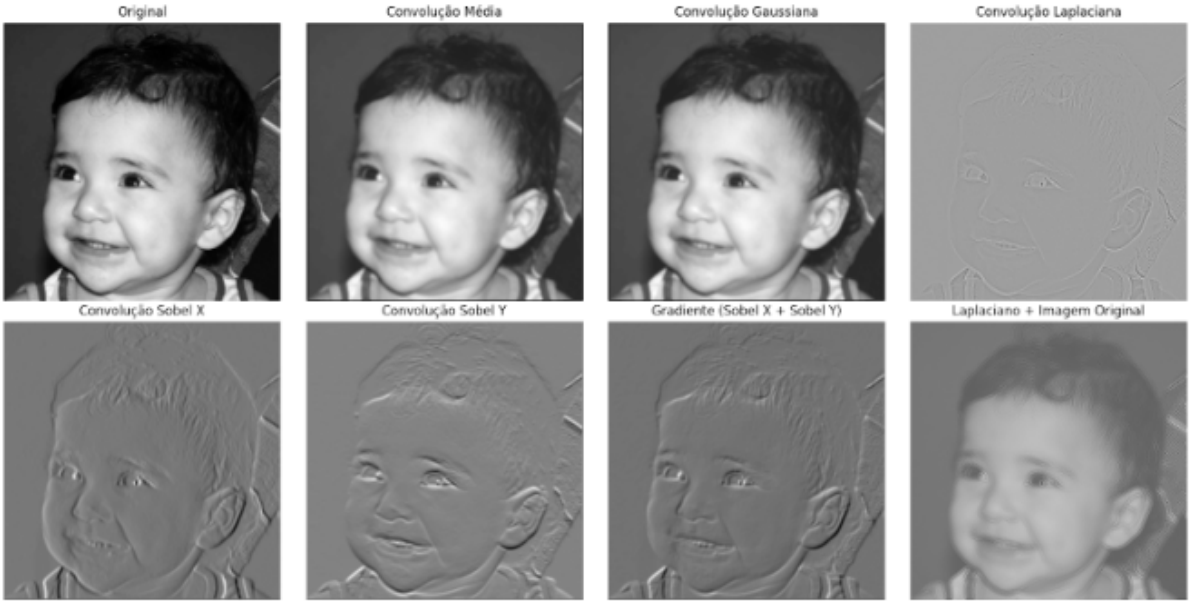
Lena



Cameraman



Biel


Original
Convolução Média
Convolução Gaussiana
Convolução Laplaciana
Convolução Sobel X
Convolução Sobel Y
Gradiente (Sobel X + Sobel Y)
Laplaciano + Imagem Original