Aluno: Carlos Vinicius Baggio Savian, BI3002217

Atividades - Filtragem Espacial

Aula - 6

- Implementar a operação de convolução.
- Utilizando OPENCV, scipy função convolve e implementaçãomanual.
- Implementar seguintes máscaras:
  - Média
  - Guassiano
  - Laplaciano
  - Sobel X
  - Sobel Y
  - Gradiente (Sobel X + Sobel Y)
  - Laplaciano somado a imagem original
- Utilizar as imagens já disponibilizadas: biel, lena, cameraman, etc

Codigo

Bibliotecas

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

Abriremos a imagem com o comando

```
img = Image.open("\Images\lena_gray_512.tif")
```

Media:

```
img = Image.open("lena_gray_512.tif")

npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel =  np.array([[0.0625,0.125,0.0625], [0.125,0.25,0.125],
[0.0625,0.125,0.0625]], dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

resultado:

- Guassiano:

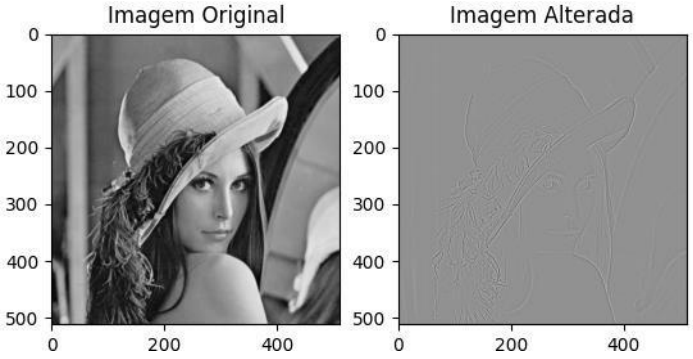
```
npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel =  np.array([[0.0625,0.125,0.0625], [0.125,0.25,0.125],
[0.0625,0.125,0.0625]], dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

Resultado:

- Laplaciano:

```
npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]], dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

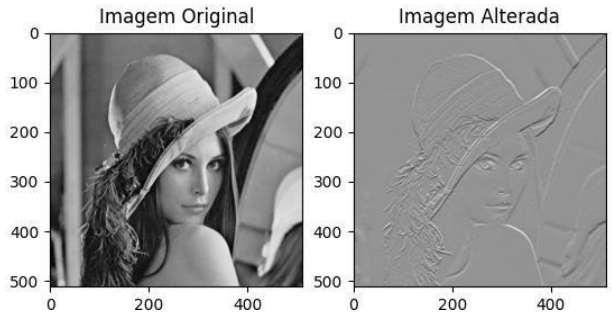
Resultado:



- Sobel X

```
npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel =  np.array([[-1,-2,-1], [0,0,0], [1,2,1]],
dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

Resultado

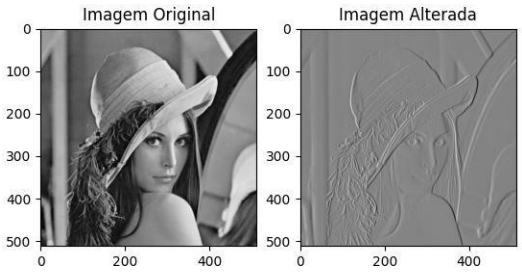


:

- Sobel Y

```
npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel =  np.array([[-1,0,1], [-2,0,2], [-1,0,1]],
dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

Resultado:



- Laplaciano somado a imagem original

```
npImgOriginal = np.array(img)
width, height = npImgOriginal.shape
npNewImg = np.zeros((width, height))
Kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]], dtype=np.float)
for row in range(1, height - 1):
    for col in range(1, width - 1):
        Sub_img = npImgOriginal[row-1:row+2,col-1:col+2]
        npNewImg[row,col] = np.sum(np.multiply(Sub_img,Kernel))
```

Resultado: