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	**Atividade**
Professor: Prof. Dr. Murilo Varges da Silva	**Data:** 04/09/2023
Nome do Aluno: Henrique Akira Hiraga	**Prontuário:** BI300838X

RELATÓRIO: PROCESSAMENTO DE IMAGEM COM MÁSCARAS DE CONVOLUÇÃO

Introdução

Este relatório descreve duas abordagens para o processamento de imagens com máscaras de convolução em Python. A primeira implementação utiliza bibliotecas como OpenCV, NumPy e Matplotlib para realizar a convolução e exibir os resultados. A segunda implementação é feita manualmente, sem o uso de bibliotecas de processamento de imagem, e também exibe os resultados usando Matplotlib.

Implementação Utilizando Bibliotecas

Passos do Código

1. Carregamento da Imagem

A implementação começa solicitando ao usuário o caminho da imagem que deseja processar. A biblioteca OpenCV é usada para carregar a imagem. Certifique-se de que a imagem escolhida seja válida e esteja no formato correto.

2. Definição das Máscaras

Sete máscaras de convolução são definidas para diferentes fins, como suavização, realce de bordas e detecção de características:

1. **Identity (Identidade):** Mantém a imagem inalterada.

2. **Mean (Média):** Suaviza a imagem por meio de uma média ponderada dos pixels vizinhos.

3. **Gauss (Gaussiana):** Aplica um filtro Gaussiano para suavizar a imagem e reduzir o ruído.

4. **Laplacian (Laplaciano):** Usado para detecção de bordas, realçando diferenças de intensidade.

5. **SobelX:** Detecta gradientes horizontais, destacando bordas verticais.

6. **SobelY:** Detecta gradientes verticais, destacando bordas horizontais.

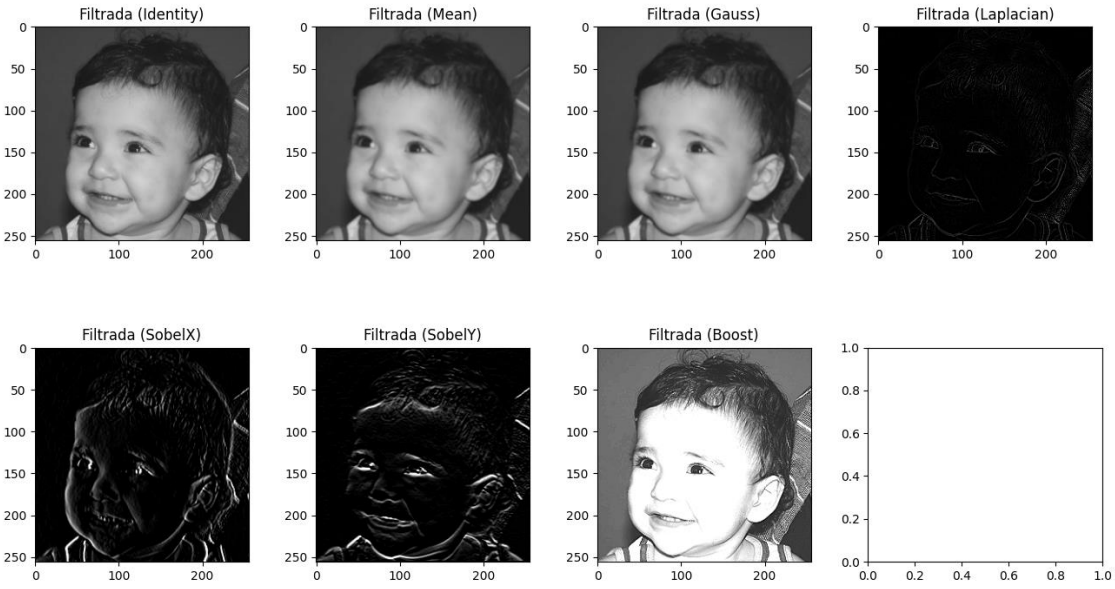
7. **Boost (Realce):** Realça características específicas na imagem.

3. Aplicação das Máscaras

Para cada máscara definida, a implementação utiliza a função `cv2.filter2D` do OpenCV para aplicar a convolução. O resultado da convolução é uma nova imagem que destaca as características da imagem original com base na máscara aplicada.

4. Exibição dos Resultados

Os resultados das máscaras são exibidos em uma única janela do Matplotlib. Uma grade de 2x4 é criada para acomodar as imagens resultantes de cada máscara. Cada imagem é exibida em uma célula da grade junto com um título que indica qual máscara foi aplicada.



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

def apply_mask(image, mask):
  return cv2.filter2D(image, -1, mask)

def main():

  # Passa o caminho da imagem
  image = cv2.imread("biel.png")

  masks = {
        "Identity": np.array([[0, 0, 0],
                              [0, 1, 0],
                              [0, 0, 0]], dtype="int"),

        "Mean": np.array([[0.1111, 0.1111, 0.1111],
                          [0.1111, 0.1111, 0.1111],
                          [0.1111, 0.1111, 0.1111]],
dtype="float"),

        "Gauss": np.array([[0.0625, 0.125, 0.0625],
                           [0.125, 0.25, 0.125],
                           [0.0625, 0.125, 0.0625]],
dtype="float"),

        "Laplacian": np.array([[0, 1, 0],
                               [1, -4, 1],
                               [0, 1, 0]], dtype="int"),

        "SobelX": np.array([[-1, 0, 1],
                            [-2, 0, 2],
                            [-1, 0, 1]], dtype="int"),

        "SobelY": np.array([[-1, -2, -1],
                            [0, 0, 0],
                            [1, 2, 1]], dtype="int"),

        "Boost": np.array([[0, -1, 0],
                           [-1, 5.7, -1],
                           [0, -1, 0]], dtype="float")
```

```
  }

  # Prepara a janela do Matplotlib para plotar as imagens
  fig, axes = plt.subplots(2, 4, figsize=(16, 8))
  plt.subplots_adjust(wspace=0.2, hspace=0.5)

    # Aplica cada máscara e exibe a imagem resultante
  for i, (mask_name, mask_value) in
enumerate(masks.items()):
        filtered_image = apply_mask(image, mask_value)
        row = i // 4
        col = i % 4

        axes[row, col].imshow(cv2.cvtColor(filtered_image,
cv2.COLOR_BGR2RGB))
        axes[row, col].set_title(f'Filtrada ({mask_name})')

  plt.show()

if __name__ == "__main__":
    main()
```

Implementação Manual

Passos do Código

1. Carregamento da Imagem

Assim como na implementação com bibliotecas, a imagem de entrada é carregada usando o OpenCV.

2. Definição das Máscaras

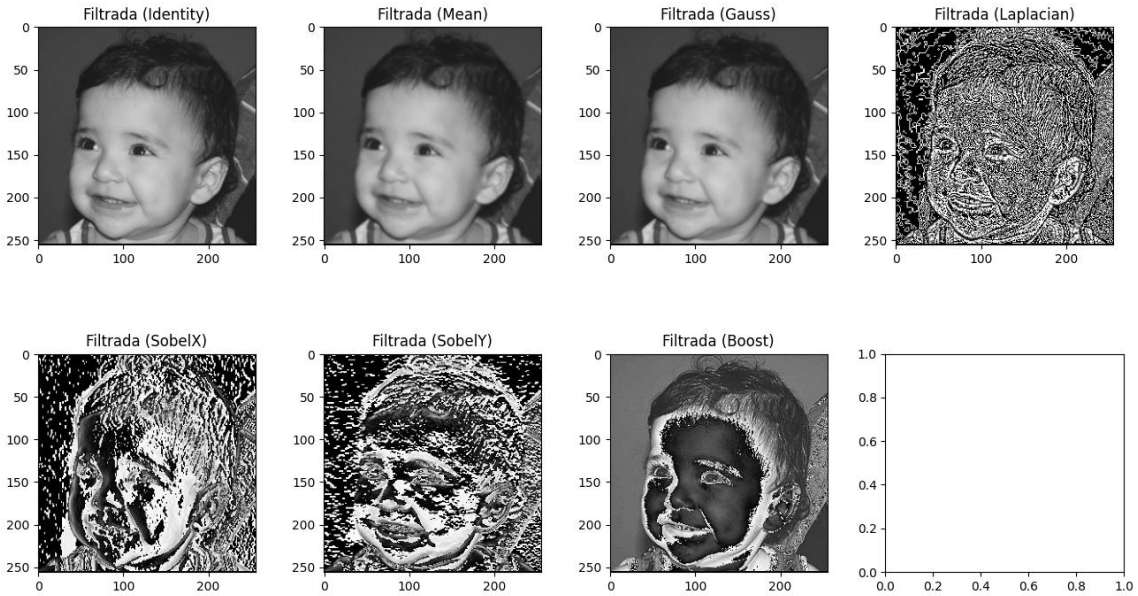
As mesmas sete máscaras de convolução definidas na implementação com bibliotecas são utilizadas na implementação manual.

3. Aplicação das Máscaras

Nesta abordagem, a convolução é realizada manualmente, utilizando loops aninhados para calcular o novo valor de cada pixel na imagem resultante.

4. Exibição dos Resultados

Os resultados das máscaras são exibidos em uma única janela do Matplotlib, da mesma forma que na implementação com bibliotecas.




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

def apply_mask(image, mask):
    height, width, channels = image.shape
    mask_height, mask_width = mask.shape
    margin_y = mask_height // 2
    margin_x = mask_width // 2

    result_image = np.zeros((height, width, channels),
dtype=np.uint8)

    for y in range(margin_y, height - margin_y):
        for x in range(margin_x, width - margin_x):
            for c in range(channels):
                result_pixel = 0
                for my in range(-margin_y, margin_y + 1):
```

```
                    for mx in range(-margin_x, margin_x +
1):
                        result_pixel += image[y + my, x +
mx, c] * mask[my + margin_y, mx + margin_x]
                result_image[y, x, c] =
np.uint8(result_pixel)

    return result_image

def main():

  # Passa o caminho da imagem
  image = cv2.imread("biel.png")

  masks = {
        "Identity": np.array([[0, 0, 0],
                              [0, 1, 0],
                              [0, 0, 0]], dtype="int"),

        "Mean": np.array([[0.1111, 0.1111, 0.1111],
                          [0.1111, 0.1111, 0.1111],
                          [0.1111, 0.1111, 0.1111]],
dtype="float"),

        "Gauss": np.array([[0.0625, 0.125, 0.0625],
                           [0.125, 0.25, 0.125],
                           [0.0625, 0.125, 0.0625]],
dtype="float"),

        "Laplacian": np.array([[0, 1, 0],
                               [1, -4, 1],
                               [0, 1, 0]], dtype="int"),

        "SobelX": np.array([[-1, 0, 1],
                            [-2, 0, 2],
                            [-1, 0, 1]], dtype="int"),

        "SobelY": np.array([[-1, -2, -1],
                            [0, 0, 0],
                            [1, 2, 1]], dtype="int"),

        "Boost": np.array([[0, -1, 0],
```

```
                            [-1, 5.7, -1],
                            [0, -1, 0]], dtype="float")
  }

  # Prepara a janela do Matplotlib para plotar as imagens
  fig, axes = plt.subplots(2, 4, figsize=(16, 8))
  plt.subplots_adjust(wspace=0.2, hspace=0.5)

    # Aplica cada máscara e exibe a imagem resultante
  for i, (mask_name, mask_value) in
enumerate(masks.items()):
        filtered_image = apply_mask(image, mask_value)
        row = i // 4
        col = i % 4

        axes[row, col].imshow(cv2.cvtColor(filtered_image,
cv2.COLOR_BGR2RGB))
        axes[row, col].set_title(f'Filtrada ({mask_name})')

  plt.show()

if __name__ == "__main__":
    main()
```

Conclusão

Ambas as implementações permitem a aplicação de várias máscaras de convolução em uma imagem de entrada e exibem os resultados de forma conveniente. As máscaras definidas podem ser usadas para várias tarefas de processamento de imagem, como suavização, detecção de bordas e realce de características.

A escolha entre as duas implementações depende das necessidades e preferências do usuário. A implementação com bibliotecas oferece uma solução mais eficiente e concisa, enquanto a implementação manual permite um maior controle sobre o processo de convolução. Em ambos os casos, é possível obter resultados de qualidade em processamento de imagem.

Espero que este relatório seja informativo e auxilie no entendimento das duas abordagens para o processamento de imagens com máscaras de convolução. Se houver dúvidas ou necessidade de mais informações, não hesite em perguntar.