Processamento de Imagem

Esse script foi desenvolvido com o objetivo de fornecer uma forma simples, mas com eficiência para explorar as técnicas de processamento de imagem. Com um menu interativo e flexível, nosso script aplica modificações básicas a uma imagem, como:

1. **Equalização de Histograma**:
   - Calcula e normaliza o histograma da imagem para melhorar o seu contraste, utilizando uma função de distribuição acumulativa para a alteração dos níveis de intensidade da imagem.
2. **Remoção de Cor**:
   - Permite que o usuário altere as cores dos canais azul, vermelho e verde da imagem, modificando a imagem conforme desejado.
3. **Aplicação de Filtro Blur**:
   - Esta função aplica um desfoque na imagem utilizando um Kernel, suavizando a imagem. O Kernel pode ser ajustado pelo usuário.
4. **Filtro de Detecção de Bordas**:
   - Utiliza um Kernel que detecta bordas na imagem e destaca suas transições, indicando os contornos e formas da imagem.

Para que tenha um maior entendimento vou explicar as partes principais do scpript:

1. **Função de calcular o histograma:**

```
def calcular_histograma(imagem):
    histograma = np.zeros((256, 3))
    for canal in range(3):
        for pixel in imagem[:, :, canal].flatten():
            histograma[pixel, canal] += 1
    return histograma
```

Esta Função calcula o histograma da imagem para que possamos fazer a equalização da imagem. Cada canal de cor e processado separadamente, acumulando seu valor de intensidade de 0 a 255

Com o “histograma = np.zeros((256,3))” é um array 2D onde suas linhas vão ter os valores de intensidade de (0-255) e suas colunas representam os canais de cores contendo (3)

2. **Função de Normalizar o Histograma:**

```
def normalizar_histograma(histograma, num_pixels):
    return histograma / num_pixels
```

Essa função vai normalizar o histograma dividindo o seu valor pelo total de pixels e convertendo contagens absolutas para proporções e uma parte importantíssima para a criação da função de Distribuição acumulativa

3. **Função para calcular o CDF:**

```
def calcular_cdf(histograma):
    cdf = np.cumsum(histograma, axis=0)
    cdf_normalizado = cdf / cdf[-1, :]
    return cdf_normalizado
```

Essa função vai calcular a função acumulativa (CDF) que foi normalizada do histograma.

“`cdf = np.cumsum(histograma, axis=0)`” *\\calcula a soma acumulada ao longo de cada canal.*

“`cdf_normalizado = cdf / cdf[-1, :]`” *\\aqui e feita a normalização dividindo pela última soma que foi acumulada de cada canal.*

4. **Função para equalizar o histograma:**

```
def equalizar_histograma(imagem, cdf_normalizado):
    imagem_equalizada = np.zeros_like(imagem)
    for canal in range(3):
        imagem_equalizada[:, :, canal] = np.interp(imagem[:, :,
canal].flatten(), range(256), cdf_normalizado[:, canal] *
255).reshape(imagem[:, :, canal].shape)
    return imagem_equalizada.astype(np.uint8)
```

Essa função aplica a equalização na imagaem usando o CDF que foi normalizada.

```
“imagem_equalizada[:, :, canal] = np.interp(imagem[:, :,
canal].flatten(), range(256), cdf_normalizado[:, canal] *
255).reshape(imagem[:, :, canal].shape)” \\Vai interpolar os valores
da imagem original para os novos valores Baseados no CDF
```

5. **Função para Remoção de Cor:**

```
def remover_canais(imagem, remover_r=False, remover_g=False,
remover_b=False):
    imagem_modificada = imagem.copy()
    if remover_r:
        imagem_modificada[:, :, 0] = 0
    if remover_g:
        imagem_modificada[:, :, 1] = 0
    if remover_b:
        imagem_modificada[:, :, 2] = 0
    return imagem_modificada
```

Essa função e uma das mais simples seu intuito e apenas remover os canais de cores vermelho, verde e azul que foi especificado pelo usuário configurando-os para zero

6. **Função para Aplicar Blur:**

```
def aplicar_filtro_blur(imagem, tamanho_kernel=5):
```

```
    kernel = np.ones((tamanho_kernel, tamanho_kernel)) / (tamanho_kernel
* tamanho_kernel)
    imagem_blur = np.zeros_like(imagem)
    for canal in range(3):
        imagem_blur[:, :, canal] = convolve(imagem[:, :, canal], kernel)
    return imagem_blur
```

Essa função que vai deixar a imagem com desfoque utiliza o kernel do tamanho especificado pelo usuário.

```
kernel = np.ones((tamanho_kernel, tamanho_kernel)) / (tamanho_kernel *
tamanho_kernel)\\O kernel é uma matriz onde todos os elementos
são iguais a sua soma é 1 oque deixa a imagem suavizada.
```

7. **Função para detecção de bordas:**

```
def aplicar_filtro_deteccao_bordas(imagem):
    kernel_sobel = np.array([[-1, -2, -1],
                             [0, 0, 0],
                             [1, 2, 1]])
    imagem_bordas = np.zeros_like(imagem)
    for canal in range(3):
        imagem_bordas[:, :, canal] = convolve(imagem[:, :, canal],
kernel_sobel)
    return imagem_bordas
```

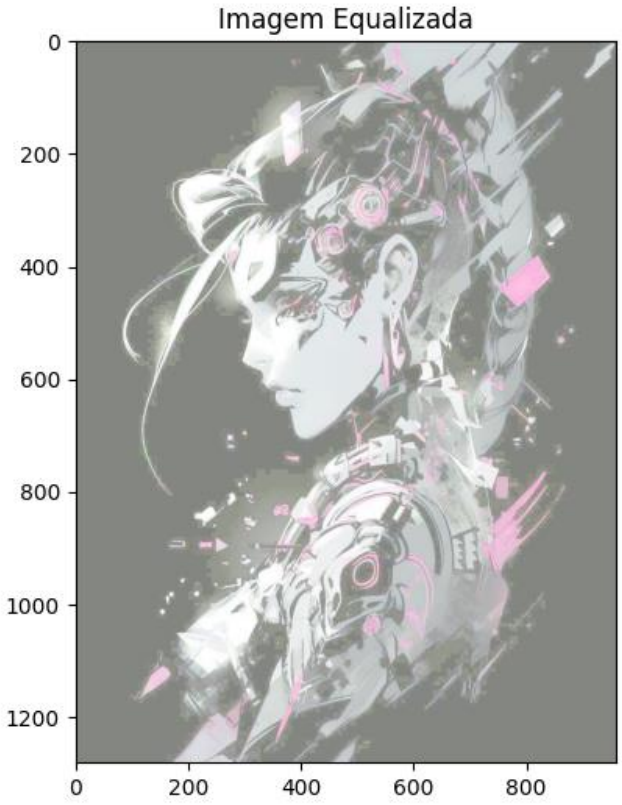
Essa função que vai aplicar o filtro de detecção de bordas usando o kernel sobel

Retorno das Funções

1. **Menu:**

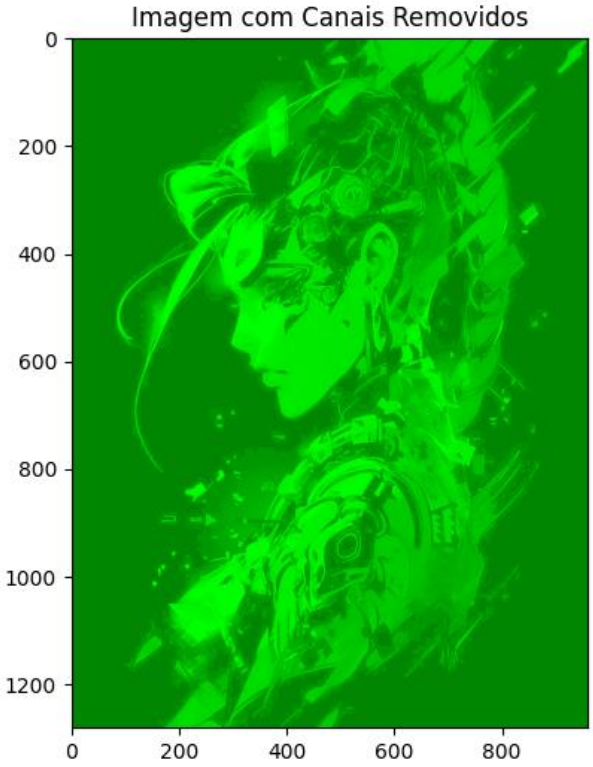
```
+==================+MENU+====================+
1. Equalização de Histograma
2. Remover Cor
3. Aplicar Filtro Blur
4. Aplicar Filtro de Detecção de Bordas
5. Sair
+=============================================+
Digite o número da operação desejada:
```

2. **Equalização de histograma:**

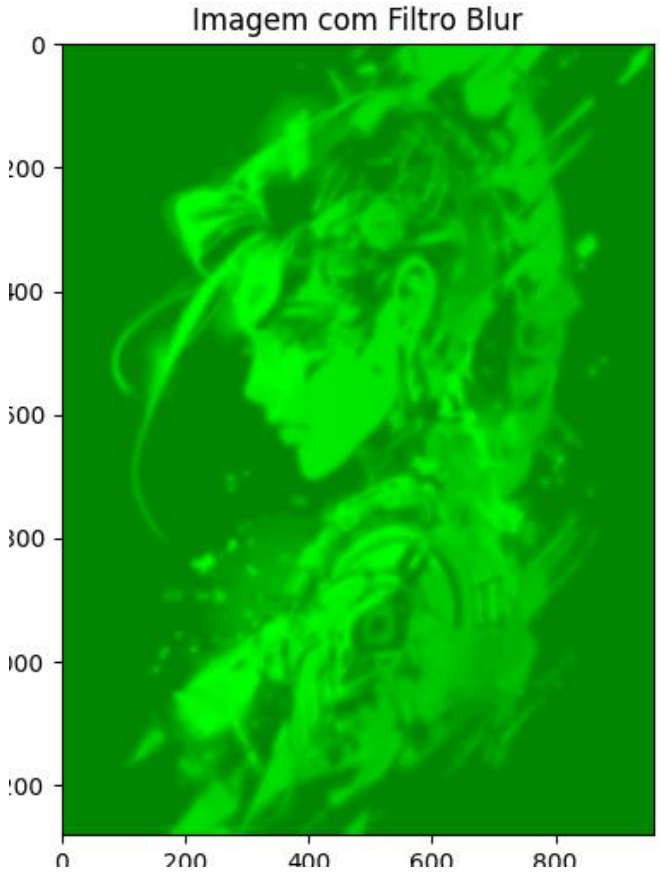


3. **Remover Cor:**

```
==============Opçoes de remoção de cor=====================
Remover canal vermelho? (S/N): s
Remover canal verde? (S/N): n
Remover canal azul? (S/N): s
```



4. **Aplicar Filtro de Blur (12):**



5. **Filtro de Detecção de bordas:**

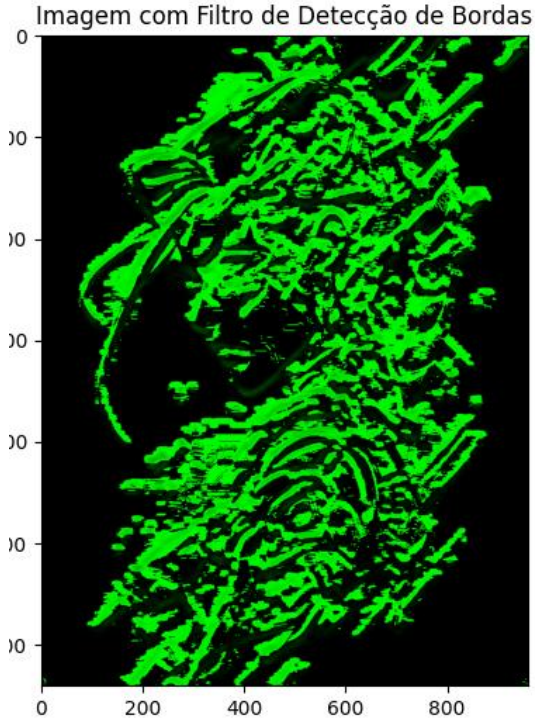


Imagem com Filtro de Detecção de Bordas
0
200
400
600
800