Relatório do Editor de Imagens

Essa é uma implementação um editor de imagens com várias funcionalidades, incluindo abrir, salvar, aplicar transformações e filtros em imagens. Vou dividir a modelagem do código em várias seções para facilitar a compreensão:

Bibliotecas Importadas

O código começa importando as bibliotecas necessárias:

```
import customtkinter as ctk
from PIL import Image, ImageEnhance
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

1. `customtkinter` é uma biblioteca personalizada para criar interfaces gráficas.
2. `PIL` (Pillow) é uma biblioteca para manipulação de imagens.
3. `numpy` é uma biblioteca para operações numéricas em arrays multidimensionais.
4. `cv2` é a biblioteca OpenCV para processamento de imagens.
5. `matplotlib.pyplot` é usado para plotar gráficos.

Classe App

É definida uma classe chamada `App`, que herda de `ctk.CTk`, que é a classe base para a interface gráfica. A classe `App` é a principal do programa e contém todos os métodos e funcionalidades.

Método `__init__`

O construtor da classe `App` é chamado `__init__` e é responsável por inicializar a interface gráfica e configurar a janela principal. Aqui estão algumas das principais ações realizadas no método `__init__`:

1. Chama o construtor da classe pai `ctk.CTk` usando `super().__init__()`.
2. Inicializa várias variáveis de instância para armazenar diferentes versões da imagem.
3. Define a variável de instância `zoom` para controlar o zoom na imagem.

4. Configura o tema de cores padrão da interface gráfica.
5. Define o título da janela principal como "EDITOR DE IMAGENS".
6. Define o tamanho mínimo da janela principal para 800x600 pixels.
7. Configura o layout da janela principal usando `grid_columnconfigure` e `grid_rowconfigure`.

Método `abrir_imagem`

Este método é chamado quando o usuário deseja abrir uma imagem. Ele exibe um diálogo de seleção de arquivo para o usuário escolher uma imagem. Algumas das principais ações realizadas neste método incluem:

1. Usando o `ctk.filedialog`, solicita ao usuário que selecione uma imagem.
2. Se uma imagem for selecionada, as variáveis de imagem existentes são limpas.
3. A imagem selecionada é carregada e exibida no frame da imagem usando a biblioteca `Pillow`.
4. A imagem original é armazenada na variável de instância `imagem_original`.

Método `atualiza_imagem`

Este método atualiza a imagem exibida no frame da imagem com uma nova imagem. Ele recebe uma imagem como entrada e a exibe no frame.

Métodos `acao_desfazer` e `acao_refazer`

Esses métodos são chamados quando o usuário clica nos botões "Desfazer" e "Refazer" no menu. Eles restauram a imagem original ou a imagem modificada, respectivamente.

Método `salvar_imagem`

Este método permite ao usuário salvar a imagem modificada. Ele exibe um diálogo de seleção de arquivo para escolher onde salvar a imagem. A imagem é salva com base no formato original, se disponível.

Métodos de Transformação (Brilho, Contraste, Gama, Equalização, Especificação)

Esses métodos são usados para aplicar várias transformações à imagem. Eles incluem ações como ajustar o brilho, contraste, gama, equalização e especificação

do histograma. Cada método inclui a lógica para aplicar a transformação à imagem atual e atualizar a exibição.

Métodos de Zoom (Ampliar e Reduzir)

Esses métodos permitem ao usuário aumentar ou diminuir o zoom na imagem. Eles alteram a variável de instância `zoom` e chamam o método `aplica_zoom` para atualizar a exibição da imagem com o novo zoom.

Métodos para Alternar entre o Menu, Transformações e Filtros

Esses métodos permitem alternar entre diferentes telas da interface gráfica, ou seja, o menu principal, a tela de transformações e a tela de filtros. Eles usam a função `grid_forget()` para esconder um frame e mostrar outro.

Métodos de Filtros (Box, Gaussiano, Mediana, Laplaciano, Sobel)

Esses métodos são usados para aplicar vários filtros à imagem. Eles incluem a lógica para aplicar o filtro à imagem atual e atualizar a exibição.

Conclusão

O código implementa um editor de imagens com funcionalidades básicas de abrir, salvar e aplicar transformações e filtros. A interface gráfica é construída usando a biblioteca `customtkinter`, e as operações de processamento de imagens são realizadas com a ajuda das bibliotecas `Pillow` e `OpenCV`.

O código é modular e bem organizado, com métodos separados para diferentes funcionalidades, tornando-o fácil de entender e manter.