Relatório de Photoshop Simplificado

Introdução

Este relatório descreverá a modelagem e as funcionalidades do código do meu photoshop simplificado, que implementa um aplicativo de edição de imagens usando a biblioteca tkinter para a interface gráfica e várias bibliotecas de processamento de imagens, como OpenCV, matplotlib e numpy. O aplicativo permite carregar, editar e salvar imagens, bem como aplicar uma variedade de filtros e transformações às imagens.

GUI

- A interface gráfica é composta por menus, botões e um canvas para exibir a imagem.
- Existem dois menus: um na parte superior para funções gerais e outro na lateral direita para funções de processamento de imagem.
  - Esses menus foram feitos com grid
  - A lógica dos menus com `grid` envolve a organização dos elementos da interface gráfica em linhas e colunas da grade para criar uma disposição ordenada em L invertido.
  - Os botões na interface acionam as diferentes funções do aplicativo.
- Além disso, o código também cria um `canvas` (`self.canvas`) para exibir a imagem e um `Frame` (`self.image_frame`) para conter o `canvas`. A configuração `grid` é usada para ajustar a posição e o tamanho desses elementos dentro da janela principal.

Estrutura da Classe `EditorImagem`

O código está organizado em uma única classe chamada `EditorImagem`nela existem:

- O `root` que representa a janela principal da aplicação tkinter
- O `app` que representa a instância do aplicativo de edição de imagens criado a partir da classe `EditorImagem`.

Variáveis de Instância Globais

- `imagem_original`: Armazena a imagem original carregada.
- `imagem_atual`: Mantém a imagem original após as modificações.
- `imagem_brilho`, `imagem_negativo`, `imagem_log`, `imagem_exp`: Armazenam versões modificadas da imagem.
- `intensidade_brilho` e `intensidade_contraste`: Controlam a intensidade do brilho e contraste, respectivamente.

Inicialização (`__init__`)

O construtor da classe `EditorImagem` inicializa a interface gráfica (GUI) e define variáveis de controle.

Carregamento de Imagem (`abrir_imagem`)

- Ao clicar no botão "Abrir", é exibida uma caixa de diálogo para selecionar uma imagem no sistema de arquivos.
- A imagem selecionada é carregada e exibida no canvas.

Redimensionamento de Imagem (`redimensionar_imagem`)

- Esta função redimensiona a imagem para se ajustar ao tamanho do canvas.
- Ela mantém a proporção da imagem original e ajusta seu tamanho para caber no canvas sem distorções.

Transformações de Intensidade Pontuais

- O código implementa várias funções para aplicar transformações à imagem, como
- Filtro Negativo

    - O filtro negativo é aplicado subtraindo 255 de cada pixel na imagem original, isso inverte as intensidades de cor, transformando áreas mais claras em mais escuras e vice-versa.
- Filtro Logarítmico
    - O filtro logarítmico é uma transformação não linear que aumenta o contraste em áreas escuras da imagem.
    - A transformação envolve o cálculo do logaritmo natural de cada valor de pixel na imagem original.
    - Para ajustar a faixa de saída para o intervalo [0, 255], os valores resultantes são normalizados usando uma fórmula que envolve mínimos e máximos.
- Filtro Exponencial
    - O filtro exponencial aumenta o contraste nas áreas mais claras da imagem.
    - Cada valor de pixel na imagem original é elevado ao quadrado para aplicar a transformação exponencial.
- Aumento e Diminuição de brilho
    - O método `aumentar_brilho` é chamado quando o botão "Aumentar Brilho" é clicado na interface, ele incrementa a variável `self.intensidade_brilho` em 0.5 para aumentar o brilho.
    - O método `diminuir_brilho` é chamado quando o botão "Diminuir Brilho" é clicado na interface, ele decrementa a variável `self.intensidade_brilho` em 0.5 para diminuir o brilho.
    - O método `aplicar_brilho` é chamado quando o botão "Aplicar Brilho" é clicado na interface.
    - Ele aplica o ajuste de brilho à imagem atual usando a função `cv2.convertScaleAbs`, que multiplica cada valor de pixel pela intensidade de brilho (`self.intensidade_brilho`). O

resultado é armazenado na variável `self.imagem_brilho` e definido como a imagem atual, por fim, a nova imagem é redimensionada para se ajustar ao widget de exibição, atualizando a interface gráfica.

- Aumento e Diminuição de contraste
  - O método `aumentar_contraste` é chamado quando o botão "Aumentar Contraste" é clicado na interface, ele incrementa a variável `self.intensidade_contraste` em 0.2 para aumentar o contraste.
  - O método `diminuir_contraste` é chamado quando o botão "Diminuir Contraste" é clicado na interface, ele decrementa a variável `self.intensidade_contraste` em 0.2 para diminuir o contraste.
  - O método `aplicar_contraste`é chamado quando o botão "Aplicar Contraste" é clicado na interface.
  - Primeiro, a imagem é convertida para tons de cinza usando `cv2.cvtColor`, em seguida, a função `cv2.convertScaleAbs` é aplicada à imagem em tons de cinza para ajustar o contraste. O resultado é mesclado novamente para criar uma imagem em cores e a nova imagem é armazenada na variável `self.imagem_atual` e redimensionada para se ajustar ao widget de exibição, atualizando a interface gráfica.
- Aumento e diminuição de imagem
  - O método `aplicar_aumento` é responsável por aumentar o tamanho da imagem atual em exibição.
  - Ele utiliza a função `cv2.resize` do OpenCV para redimensionar a imagem.
  - Para o aumento, é definido um fator de escala de 2x, é utilizado a interpolação cubíca, `cv2.INTER_CUBIC`

- Após o redimensionamento, a imagem resultante é armazenada na variável `self.imagem_zoom` e é definida como a imagem atual.
- A nova imagem é então redimensionada para caber no widget de exibição (`self.redimensionar_imagem`), atualizando assim a interface gráfica com a imagem ampliada.

O método `aplicar_diminuicao` é responsável por diminuir o tamanho da imagem atual em exibição.

Assim como no aumento, ele utiliza a função `cv2.resize` do OpenCV para redimensionar a imagem.

Para a diminuição, é definido um fator de escala de 0.5x, o que significa que cada dimensão da imagem (largura e altura) será multiplicada por 0.5, reduzindo assim a imagem para metade do tamanho original.

E faz o mesmo procecimento de redimensionamento e atualiza a interface gráfica.

Histograma (`exibir_hist`)

- Esta função calcula e exibe o histograma da imagem.
- No código, o histograma é calculado para cada canal de cor (R, G, B) separadamente.
- A função `np.histogram` é usada para calcular o histograma, onde `bins` define o número de barras no histograma (geralmente 256 para 8 bits de profundidade de cor) e `range` define o intervalo de valores de intensidade de cor (0 a 255 para 8 bits).
- Os resultados são então empilhados verticalmente para criar um histograma de cores completo.
- O histograma é exibido em um gráfico usando a biblioteca `matplotlib`.

Equalização de Histograma (`hist_equalizado`)

- A equalização de histograma é usada para melhorar o contraste em uma imagem.
- No código, ela é aplicada separadamente a cada canal de cor (R, G, B).
- Para cada canal, a função `cv2.equalizeHist` é usada para equalizar o histograma.
- Isso redistribui as intensidades de cor para abranger todo o intervalo [0, 255].
- O resultado é mesclado novamente para criar uma imagem equalizada.

Carregar Imagem de Referência (`carregar_imagem_referencia`)

- Permite carregar uma imagem de referência para comparação ou outras ações específicas.
- Redimensiona a imagem de referência e exibe-a, se necessário.

Filtros de Suavização (`filtro_box`, `filtro_gaussiano`, `filtro_mediana`)

- Implementa filtros de suavização para aplicar desfoque à imagem.
- Cada filtro é aplicado a `self.imagem_original` e a imagem resultante é exibida.

Filtro de Aguçamento Laplaciano (`filtro_agucamento_laplaciano`)

- Implementa um filtro de aguçamento usando a técnica Laplaciana.
- Aplica suavização à imagem original, calcula a Laplaciana e adiciona de volta à imagem original para realçar as bordas.

Filtro Sobel (`filtro_sobel`)

- Aplica o operador Sobel para detecção de bordas.

- Converte a imagem para tons de cinza, aplica o operador Sobel horizontal e vertical e combina os resultados.

Salvamento de Imagem (`salvar`)

- Permite salvar a imagem atual com um nome de arquivo especificado pelo usuário.

Encerramento do Aplicativo (`sair`)

- Fecha o aplicativo quando o botão "Sair" é pressionado.

Funcionalidades Adicionais

- O código permite interações do usuário para ajustar a intensidade de brilho e contraste.
- Oferece opções para aumentar e diminuir o tamanho da imagem.

Conclusão

O código apresentado é uma implementação funcional de um editor de imagens simples com uma interface gráfica amigável. Ele oferece várias funcionalidades de processamento de imagem, incluindo filtros, transformações e melhorias de contraste.