Documentación Técnica de la Clase `ImageViewer`

Desarrollo en Python

9 de abril de 2025

1. Introducción

La clase `ImageViewer` está diseñada para actuar como un visor e interactuador de imágenes utilizando la biblioteca `CustomTkinter` sobre `tkinter`, además de emplear funciones del módulo `procesamientoImagenes` (abreviado como `pi`) para realizar diversas transformaciones y ajustes de imágenes. Su implementación está orientada a la carga, visualización, procesamiento y transformación de imágenes, permitiendo operar sobre una o dos imágenes simultáneamente (por ejemplo, para realizar fusiones).

2. Estructura y Configuración Inicial

2.1. Herencia y Configuración de la Ventana Principal

La clase hereda de `ctk.CTk`, lo que le permite aprovechar una interfaz con apariencia moderna. En el método `__init__`, se configuran parámetros globales de apariencia, el título y las dimensiones de la ventana:

- Se establece el modo de apariencia (`dark`) y el tema predeterminado (`blue`).
- La ventana se dimensiona a 1000x600 píxeles.

2.2. Variables y Almacenamiento de Imágenes

La clase define varias variables para gestionar las imágenes:

- **`self.image`:** Imagen principal cargada.
- **`self.second_image`:** Segunda imagen para tareas de fusión.
- **`self.current_image`:** Imagen actualmente visualizada, reflejando las transformaciones aplicadas.
- **`self.image_original` y `self.second_image_original`:** Copias originales de las imágenes para efectos de restauración.
- **`self.typed_image`:** Imagen procesada según el tipo seleccionado (Original, Escala de Grises, Negativo, Binarizada).
- **`self.channel_image`:** Imagen resultante de la extracción o combinación de capas de canales (RGB o CMY).

3. Organización de la Interfaz Gráfica

La interfaz se estructura en tres zonas principales:

3.1. Topbar

Un **frame** superior que aloja los botones principales:

- `Cargar Imagen`: Llama a `load_image()`.
- `Guardar Imagen`: Invoca `save_image()`.
- `Restaurar`: Ejecuta `restore_image()`.
- `Eliminar Imagen`: Ejecuta `remove_image()`.

3.2. Sidebars

Se utilizan dos barras laterales para organizar controles:

- **Sidebar Izquierda:** Se implementa mediante un `CTkScrollableFrame` y agrupa:
  - Visualización del histograma de la imagen.
  - Sliders para ajustar **brillo** y **contraste** (con rangos de -100 a 100 y de 0.01 a 3, respectivamente).
  - Selección de zonas (claras u oscuras) mediante *RadioButtons* (`self.zona_var`).
  - Menú de opciones (*OptionMenu*) para definir el tipo de imagen (Original, Escala de Grises, Negativo, Binarizada).
  - CheckBoxes para la selección exclusiva de canales RGB y CMY, permitiendo extraer y visualizar únicamente las capas indicadas.
  - Controles para la fusión de imágenes, que incluyen un slider para definir el factor de transparencia y un botón para aplicar la fusión mediante `apply_fusion()`.
- **Sidebar Derecha:** También implementada con un `CTkScrollableFrame`, esta zona agrupa controles para la transformación espacial y geométrica de la imagen:
  - **Rotación:** Slider para ajustar el ángulo (entre -360° y 360°) que invoca `on_rot_slider_change(`
  - **Traslación:** Entradas para los desplazamientos en `dx` y `dy`, con botón que llama a `apply_translation()`.
  - **Recorte:** Entradas para las coordenadas (x1, y1) y (x2, y2) y botón que activa `recortar()`.
  - **Cambio de Tamaño:** Entradas para ancho y alto junto con un botón que ejecuta `apply_resize()` utilizando la función `pi.cambiarTamaño()`.
  - **Zoom:** Entradas para definir el punto central y un slider para el factor de zoom, cuyo valor se actualiza en tiempo real mediante `update_zoom_label()`; se aplica con el botón `apply_zoom()`.

3.3. Área Principal

Dividida en dos sub-*frames*, se utiliza para la visualización de:

- **Imagen 1:** Visualización de la imagen principal.
- **Imagen 2:** Visualización de la segunda imagen en caso de estar cargada (por ejemplo, para fusión).

4. Funcionalidades de Carga y Visualización

4.1. Carga de Imágenes

El método `load_image()` permite:

- Seleccionar una imagen mediante un cuadro de diálogo.
- Si no existe una imagen cargada, se establece la imagen principal (`self.image`), se guarda una copia original (`self.image_original`) y se inicializa el tipo de visualización.
- Si ya hay una imagen cargada y la variable `self.second_image` es nula, se carga la imagen como segunda imagen, almacenándose además una copia original.

4.2. Visualización y Guardado

El método `display_image()` actualiza los paneles de imagen:

- Convierte la imagen actual a RGB y la muestra en el panel izquierdo.
- Si existe una segunda imagen, se actualiza el panel derecho.
- Se actualizan las etiquetas de tamaño para reflejar las dimensiones de las imágenes.

El método `save_image()` despliega un diálogo que permite guardar la imagen actualmente mostrada en distintos formatos (PNG, JPEG, etc.).

5. Transformaciones y Procesamiento de Imágenes

Se disponen de varios métodos para aplicar transformaciones a la imagen:

5.1. Ajustes Globales

- **`apply_changes()`:** Utiliza la función `cv2.convertScaleAbs` para modificar brillo y contraste sobre la imagen original.

5.2. Transformaciones Geométricas

- **Traslación:** `apply_translation()` recoge los valores de desplazamiento (dx, dy) y utiliza `pi.trasladar`.
- **Recorte:** `recortar()` obtiene las coordenadas de recorte y llama a `pi.recortar`.
- **Cambio de Tamaño:** `apply_resize()` utiliza `pi.cambiarTamaño` para redimensionar la imagen.
- **Zoom:** `apply_zoom()` aplica zoom usando las coordenadas indicadas y el factor seleccionado del slider, apoyándose en la función `pi.zoom`.
- **Rotación:** `on_rot_slider_change()` rota la imagen principal mediante la función `pi.rotar`, actualizando la etiqueta de rotación.

5.3. Fusión de Imágenes

El método `apply_fusion()` permite combinar la imagen principal con la segunda imagen:

- Se valida la existencia y la congruencia en tamaño entre ambas imágenes.
- Se aplica la fusión usando el factor de transparencia obtenido del slider y la función `pi.fusionarImagenesConEq`.
- Tras la fusión, la segunda imagen se elimina para evitar inconsistencias.

5.4. Ajustes de Brillo y Contraste

- **Brillo:** `on_brillo_slider_change()` ajusta el brillo de la imagen procesada (`typed_image`) utilizando la función `pi.ajustarBrillo`.
- **Contraste:** `on_contraste_slider_change()` modifica el contraste. Dependiendo de la zona seleccionada (claras u oscuras, mediante `self.zona_var`), se invoca a `pi.contrastarZonasClara` o `pi.contrastarZonasOscuras`.

5.5. Cambio de Tipo de Visualización

El método `on_tipo_change()` permite modificar el procesamiento de la imagen de acuerdo al tipo solicitado:

- **Original:** Se mantiene la imagen sin alteraciones.
- **Escala de Grises:** Se transforma la imagen a escala de grises utilizando `pi.grisesConAverage`.
- **Negativo:** Se aplica la función `pi.negativa`.
- **Binarizada:** Se utiliza `pi.binarizar` con un umbral fijo (0.5).

5.6. Procesamiento de Canales (RGB y CMY)

El método `on_channel_check()` gestiona la selección exclusiva de canales:

- Se permite la activación de un único canal a la vez, deseleccionando en caso de múltiples selecciones.
- Se extraen las capas correspondientes llamando a las funciones `pi.extraerCapaRoja`, `pi.extraerCapaVerde`, `pi.extraerCapaAzul` para RGB, y `pi.extraerCapaCian`, `pi.extraerCapaMa` y `pi.extraerCapaAmarilla` para CMY.
- La imagen resultante se asigna a `self.current_image` para su visualización.

6. Gestión y Restauración de la Interfaz

- **`reset_ui()`:** Restablece los valores de los sliders, entradas y opciones (por ejemplo, brillo, contraste, rotación, zoom, selección de canales y tipo de imagen).
- **`restore_image()`:** Restaura la imagen principal y, si corresponde, la segunda imagen utilizando las copias originales.
- **`remove_image()`:** Elimina las imágenes actuales y limpia los paneles de visualización, reiniciando la interfaz.

7. Visualización del Histograma

El método `see_histogram()` verifica que la imagen principal esté cargada y, de ser así, muestra su histograma RGB mediante la función `pi.histograma`.

8. Conclusiones

La implementación actual de `ImageViewer` integra una amplia gama de funcionalidades: desde la carga y visualización básica de imágenes hasta complejos procesos de transformación, ajuste de parámetros y fusión de imágenes. La organización de la interfaz en paneles superiores, laterales e área principal permite una interacción intuitiva. Además, la integración con el módulo `procesamientoImagenes` proporciona las herramientas necesarias para aplicar efectos y transformaciones, adaptándose a las necesidades de procesamiento de imágenes en aplicaciones basadas en Python.