**Documentation: Image Processing App**

**Overview:**

This script is an image processing application built using Tkinter (a Python GUI toolkit) and Pillow (PIL) for image manipulation. It supports several image editing features like grayscale conversion, cropping, resizing, flipping, and combining images. The application also provides advanced features like drag-and-drop functionality and JPEG compression level control when saving images. It is also implemented in MATLAB.

The app allows users to:

1. Load images.
2. Perform various image manipulations (crop, resize, flip, etc.).
3. Convert an image to black-and-white (with a dynamic threshold slider).
4. Combine images (side-by-side or overlay).
5. Save images in different formats (JPEG, PNG, TIFF, BMP) with custom compression levels for JPEG format.

**Code Structure and Components:**

The main components of the application are:

1. **Tkinter Window:** The GUI framework used to create buttons, sliders, and other interface elements.
2. **Pillow (PIL):** Used for image loading, saving, and manipulation (e.g., cropping, resizing, flipping, etc.).
3. **TkinterDnD2:** A drag-and-drop extension for loading images into the application via drag-and-drop functionality.

**\_\_init\_\_(self, root)**

This is the constructor method of the ImageProcessingApp class. It initializes the application window, sets its title and size, and calls various methods to set up the layout, load the background image, and enable drag-and-drop functionality.

* **Parameters:**
  + **root:** The root window of the Tkinter application.
* **Key Initializations:**
  + **self.save\_format**: Holds the format in which the user will save the image (default is JPEG).
  + **self.bg\_img\_tk:** A placeholder for the background image to prevent garbage collection.
  + **self.original\_img:** Stores the original image for operations like black-and-white conversion (with threshold adjustment).

**load\_background\_image(self)**

This method loads a background image for the application's window. It resizes the image to fit the window size and uses ImageTk.PhotoImage to display it within a Tkinter Canvas.

* **Key Operations:**
  + The image is loaded from a local file ip.jpg.
  + The image is resized to fit the window dimensions (600x700).
  + The resized image is placed on a canvas in the root window.

**create\_widgets(self)**

This method sets up the main components of the application interface, including:

* Title label at the top.
* Button frame containing buttons for image manipulations.
* Image display area, where the loaded image is shown.
* A slider for adjusting the threshold during black-and-white conversion.

**center\_widgets(self)**

This method is responsible for creating and placing buttons for various image operations (browse, save, crop, resize, etc.) and combining these buttons into a button frame.

* **Key Widgets:**
  + **Browse:** Loads an image from the filesystem.
  + **Save:** Saves the currently loaded image (with optional compression).
  + **Convert to B/W:** Converts the image to black and white, using a dynamic threshold.
  + **Crop/Resize:** Allows users to crop or resize the image.

**create\_save\_format\_radiobuttons(self, parent)**

This method creates radio buttons to allow the user to choose the image format in which they want to save the image (JPEG, PNG, BMP, TIFF). These formats are stored in **self.save\_format.**

**load\_image(self)**

This method allows the user to browse and load an image from their local file system. It uses **filedialog.askopenfilename** to open a file dialog. After the image is selected, it calls **display\_loaded\_image()** to process and display the image.

**display\_loaded\_image(self, img\_path)**

This method loads the image from the provided path and sets it as the current image **(self.img)**. It also stores a copy of the original image **(self.original\_img)** to preserve it for future manipulations (such as adjusting threshold in black-and-white conversion).

**display\_image(self, img)**

This method resizes and displays the loaded image in the application's image area.

* **Key Steps:**
  + The image is resized to fit within a 400x300 area while maintaining aspect ratio.
  + The resized image is displayed using ImageTk.PhotoImage in a Tkinter Label.

**show\_image\_info(self)**

This method displays the basic information about the loaded image (dimensions, format, and file size). It is shown in a pop-up window using **messagebox.showinfo.**

**save\_image(self)**

This method handles saving the currently loaded and processed image. It provides the user with options for:

* Custom compression levels: When saving in JPEG format, the user is prompted to input a compression quality level (1-100).
* Format selection: The user can save the image in the format selected via radio buttons (JPEG, PNG, TIFF, BMP).
* **Logic:**
  + If the format is JPEG, the method prompts the user for a compression level.
  + If the compression level is specified, the image is saved with the corresponding quality.
  + For other formats, the image is saved without compression.

**convert\_to\_bw(self)**

This method converts the loaded image to black and white (grayscale). It also displays a slider below the save format radio buttons for dynamically adjusting the threshold for the black-and-white conversion.

**update\_threshold(self, threshold)**

This method dynamically adjusts the threshold for black-and-white conversion based on the value selected using the slider.

* **Steps:**
  + The original image is converted to grayscale.
  + A threshold is applied, turning pixels above the threshold white and pixels below the threshold black.

**crop\_image(self)**

This method allows the user to crop the image. The user is prompted to input the crop dimensions (left, top, right, bottom), and the image is cropped accordingly.

**resize\_image(self)**

This method allows the user to resize the image. The user is prompted to input the new dimensions (width, height), and the image is resized using the specified dimensions.

**flip\_horizontal(self) and flip\_vertical(self)**

These methods flip the image horizontally or vertically, respectively, using the Image.transpose function from Pillow.

**combine\_images(self)**

This method allows the user to combine two images, either side-by-side or overlaying them. The user is prompted to select two images and specify the combination method (side or overlay).

**enable\_drag\_and\_drop(self)**

This method enables drag-and-drop functionality using the TkinterDnD2 library. Users can drag an image from their file explorer and drop it onto the application window, and the dropped image will be loaded.

**on\_drop(self, event)**

This method handles the drag-and-drop event by loading the dropped image file into the application.

**Usage:**

1. Loading an Image: Click on the Browse button to select an image from your computer.
2. Manipulating the Image:
   * Use the Convert to B/W button to convert the image to black and white and adjust the threshold using the slider.
   * Use the Crop, Resize, Flip, and Combine buttons to perform the respective image manipulations.
3. Saving the Image: Click the Save Image button to save the edited image. If saving as JPEG, you will be prompted to select a compression level.

**Conclusion:**

This ImageProcessingApp provides a simple yet powerful interface for basic image processing operations. It integrates several features, including drag-and-drop functionality, real-time threshold adjustments, and custom JPEG compression. The code is built using Tkinter and Pillow, making it lightweight and easy to extend or modify for additional features.