Image Processing Project

Project Overview

This project involves the design and implementation of a Python-based graphical user interface (GUI) program for performing various image processing operations. The key requirements include custom implementation of image processing techniques, a user-friendly interface design, and grouping similar functionalities for enhanced user experience.

Features and Functionality

General Requirements

- **Image Uploading**: Allows users to upload an image from their device and displays it within the program window.
- **Dynamic Processing**: Each operation can be applied to the uploaded image, with the results displayed dynamically in the interface.
- **Organized UI**: Operations are grouped into sections for better usability.

Detailed Function Descriptions

1. Image Color Conversion

- **Description**: Converts the uploaded image to grayscale using pixel manipulation.
- **How It Works**: Each pixel's intensity is calculated by averaging the RGB components or using weighted averages:
  Gray = 0.2989*Red + 0.5870*Green + 0.1140*Blue.

2. Thresholding

- **Basic Threshold**:
  - **Description**: Calculates the threshold based on pixel intensity averages.
  - **How It Works**: Pixels above the threshold are set to white, and those below are set to black, creating a binary image.
- **Advanced Halftoning (Error Diffusion)**:
  - **Description**: Enhances halftoning using error diffusion.

    - **How It Works**: Distributes the pixel intensity approximation error to neighboring pixels in a pattern for smoother halftones.

3. Histogram Operations

- **Generate Histogram**:
    - **Description**: Displays the frequency distribution of pixel intensities.
    - **How It Works**: Counts the number of pixels for each intensity value (0-255).
- **Histogram Equalization**:
    - **Description**: Enhances image contrast by redistributing intensity values.
    - **How It Works**: Uses the cumulative distribution function (CDF) to map intensities.

4. Edge Detection Methods

- **Simple Methods**:
    - **Sobel Operator**: Detects edges by computing intensity gradients using a 3x3 kernel.
    - **Prewitt Operator**: A simpler gradient-based edge detector using a 3x3 kernel.
    - **Kirsch Compass Masks**: Applies a directional kernel to detect edges and their orientations.
- **Advanced Methods**:
    - **Homogeneity Operator**: Detects edges by calculating intensity differences in a pixel's neighborhood.
    - **Difference Operator**: Highlights regions with significant intensity changes.
    - **Difference of Gaussians (DoG)**: Detects edges by subtracting two Gaussian-blurred images.
    - **Contrast-Based Edge Detection**: Smoothens the image and enhances edges based on contrast differences.
    - **Variance**: Identifies edges by computing intensity variance in neighborhoods.
    - **Range**: Measures intensity range (max - min) in a pixel's neighborhood to detect edges.

5. Filtering Techniques

- **High-Pass Filter**:

    - **Description**: Enhances edges and details.
    - **How It Works**: Amplifies high-frequency components while suppressing low frequencies.
- **Low-Pass Filter**:
    - **Description**: Smoothens the image.
    - **How It Works**: Reduces noise by averaging pixel values in a local neighborhood.
- **Median Filter**:
    - **Description**: Removes noise while preserving edges.
    - **How It Works**: Replaces a pixel's value with the median of its neighbors.

6. Image Operations

- **Addition**:
    - **Description**: Combines the original image and a copy.
    - **How It Works**: Increases brightness by summing pixel intensities.
- **Subtraction**:
    - **Description**: Compares the original image and a copy.
    - **How It Works**: Highlights differences by subtracting pixel values.
- **Inversion**:
    - **Description**: Converts the image to its negative.
    - **How It Works**: Flips each pixel's intensity (e.g., 255 becomes 0).

7. Histogram-Based Segmentation

- **Manual Segmentation**:
    - **Description**: Allows user-defined thresholds for segmentation.
- **Histogram Peak and Valley Methods**:
    - **Description**: Identifies intensity peaks or valleys for segmentation.
- **Adaptive Histogram**:
    - **Description**: Dynamically adjusts thresholds based on local intensity variations.