Image Compression GUI APP

Image Compression GUI APP Python: PyQt5

How to Get Started

Prerequisites

- Python 3.6 or higher
- PyQt5
- Pillow

How to install the required packages

1. Create a virtual environment

```
$ python3 -m venv env
```

1. Activate the virtual environment

```
$ source env/bin/activate
```

1. Install the required packages

```
$ pip install -r requirements.txt
```

How to run the app

```
$ python3 app.py
```

Project Documentation: Image Compression Application

Overview

This project implements an image compression application with a graphical user interface (GUI) using PyQt5. It allows users to load, view, and compress images by adjusting parameters such as quality and size ratio. The project comprises three primary components:

- **App.py**: The main application that defines the GUI and handles user interactions.
- **Compressor.py**: Provides image compression functionality.
- **ImageViewer.py**: A utility to display and interact with images.

- **NoiseReducer.py**: A utility to reduce noise in images.

`App.py`

This file implements the main application GUI using PyQt5.

Class: AppUI

Represents the main window of the application.

Methods

- **`__init__(self)`**
  - Initializes the `AppUI` class.
  - Sets up the GUI by calling `initUI`.
  - Creates an instance of the `Compressor` class.
  - Sets the application window properties (title, size, icon).
  - **Parameters:** None
  - **Raises:** Exception if an error occurs during initialization.
- **`initUI(self)`**
  - Constructs the GUI layout, including labels, input fields, buttons, and other widgets.
  - Connects button actions to their respective methods.
  - **Parameters:** None
- **`browse(self)`**
  - Opens a file dialog for the user to select an image file.
  - Updates the image path field and displays the selected image in the viewer.
  - **Parameters:** None
- **`compress(self)`**
  - Reads user-specified parameters and compresses the selected image using the `Compressor` class.
  - Displays compression results (file path, size before/after, and size change) in the output text box.
  - **Parameters:** None
  - **Handles:** Exceptions during compression.
- **`noise_reduction`**
  - Reduces noise in the selected image using the `NoiseReducer` class.
  - Displays the denoised image in the viewer.
  - **Parameters:** None

    - **Handles:** Exceptions during noise reduction.

`Compressor.py`

This file defines functionality for image compression.

Function: get_size_format(b, factor=1024, suffix="B")

Converts a file size in bytes to a human-readable format.

- **Parameters:**
    - `b` (int): File size in bytes.
    - `factor` (int): Conversion factor (default: 1024).
    - `suffix` (str): Suffix for the size (default: "B").
- **Returns:** A formatted string representing the file size.

Class: Compressor

Handles image compression logic.

Methods

- **`__init__(self)`**

    - Initializes the `Compressor` class.
    - **Parameters:** None

- **`_random(self)`**

    - Generates a random string of 10 digits to append to filenames.
    - **Returns:** A random string of numbers.

- **`compress_img(self, image_name, new_size_ratio=0.9, quality=90, width=None, height=None, to_jpg=True)`**

    - Compresses and resizes an image.
    - **Parameters:**
        - `image_name` (str): Path to the input image.
        - `new_size_ratio` (float): Resize ratio (default: 0.9).
        - `quality` (int): Compression quality (default: 90).
        - `width` (int): Desired width of the image (optional).
        - `height` (int): Desired height of the image (optional).
        - `to_jpg` (bool): Whether to convert the image to JPEG (default: True).
    - **Returns:** A dictionary containing:
        - `image`: Path to the compressed image.
        - `size_before`: Size of the original image.

        - `size_after`: Size of the compressed image.
        - `size_change`: Percentage size change.

`ImageViewer.py`

This file provides functionality for viewing images within the application.

Class: QImageViewer

A PyQt5-based image viewer window.

Methods

- **`__init__(self)`**

    - Initializes the `QImageViewer` class.
    - Sets up the viewer layout and menus.
    - **Parameters:** None

- **`open(self, filename)`**

    - Opens and displays an image.
    - **Parameters:**
        - `filename` (str): Path to the image file.
    - **Raises:** Information dialog if the image cannot be loaded.

- **`zoomIn(self)`**

    - Zooms in on the displayed image by a factor of 1.25.

- **`zoomOut(self)`**

    - Zooms out on the displayed image by a factor of 0.8.

- **`normalSize(self)`**

    - Resets the image to its normal size.

- **`fitToWindow(self)`**

    - Toggles between fitting the image to the window or maintaining its size.

- **`createActions(self)`**

    - Defines menu actions for opening images, zooming, and exiting the application.

- **`createMenus(self)`**
  - Creates the file and view menus, linking them to the respective actions.
- **`updateActions(self)`**
  - Enables or disables actions based on the current state (e.g., zoom availability).
- **`scaleImage(self, factor)`**
  - Scales the image by the specified factor.
  - **Parameters:**
    - `factor` (float): Scaling factor.
- **`adjustScrollBar(self, scrollBar, factor)`**
  - Adjusts the scrollbar to match the scaling.
  - **Parameters:**
    - `scrollBar` (QScrollBar): Scrollbar to adjust.
    - `factor` (float): Scaling factor.

`NoiseReducer.py`

This file provides functionality for reducing noise in images, using Median Filtering

Function:

- **`reduce_noise(image_abs_path, redcution_factor=2)`**
  - Reduces noise in an image using Gaussian blur.
  - **Parameters:**
    - `image_abs_path` (str): Path to the input image.
    - `redcution_factor` (int): Factor for Gaussian blur (default: 2).
  - **Returns:** A dictionary containing:
    - `Message`: Status message.
    - `Path`: Path to the denoised image.
  - **Raises:** Exception if an error occurs during noise reduction.

Usage Instructions

1. Launch the application by running `App.py`.
2. Use the “Browse” button to select an image.
3. Adjust compression settings as desired (quality, resize ratio, conversion to JPEG).
4. Click “Compress” to compress the image.

5. View compression results in the output box and interact with the image using the viewer.

Dependencies

- Python 3.x
- PyQt5
- PIL (Pillow)

Ensure these libraries are installed before running the application.