Image Enhancement Tool

Prepared by:

Name: Abdallah Haider **ID:** -202203795

Name: Ahmed Maher **ID:** -202203622

Name: Taha Ghazy **ID:** -202202934

Name: Ali Turkey **ID:** -202202048

Name: Mohamed Eid **ID:** -202201571

Name: Fady Eshak **ID:** -202202074

Supervised by:

Dr. Hossam Elbihairy

Co-Supervised by

Eng. Ahmed Ali

Eng. Ahmed Zakaria

Eng. Mohamed Abdelmoniem

2024 – 2025

Introduction

A CLI-based image processing tool for performing common enhancement operations including gamma correction, histogram equalization, and color adjustments. Designed for batch processing and educational demonstrations of image processing fundamentals

Key Features

- Load and process images from the local file system.
- Apply a wide range of enhancement techniques:
    - Brightness and contrast adjustment
    - Histogram equalization
    - Grayscale conversion
    - Sharpening and blurring
    - Edge detection
    - Image flipping and rotation
    - Image resizing
- Save the enhanced image for later use or evaluation.

Technologies and Libraries Used

Library	Purpose	Why It Was Chosen
cv2	Fast image processing and manipulation (OpenCV).	OpenCV is highly optimized and provides a vast number of image processing tools.
numpy	Efficient array operations for mathematical computations.	NumPy is essential for handling image arrays and matrix operations.
PIL/Pillow	Image opening, enhancement, and format conversion.	Pillow is simple to use for common image enhancement tasks like brightness and contrast adjustments.
skimage	Advanced image processing algorithms (e.g., histogram equalization).	scikit-image is ideal for scientific and advanced image manipulation.
img_as_ubyte	Converts image to unsigned byte format (range [0, 255]).	Converts floating point images back to byte format for saving and visualizing processed images.

Prerequisites

- Python 3.x installed
- pip (Python package manager)

Installation

1. Clone the repository:
    - git clone https://github.com/TurkishTM/Image_Enhancement_Tool.gitcd Image_Enhancement_Tool
2. Install the required libraries:
    - pip install opencv-python Pillow numpy
3. Run the main program:
    - python MainCode.py

How to Use the Project

1. Run the script to start the application.
2. The program will prompt you to select an image file.
3. Choose the type of enhancement or transformation to apply.
4. The modified image will be generated immediately.
5. Save the final version of the image to your machine.

Function List and Descriptions

Each function below plays a specific role in enhancing or transforming images. These were developed to reflect a strong understanding of fundamental image processing principles.

1.Image Loading/Saving

```python
def load_image(path):
    """
    Load an image from disk and convert to RGB.
    use convert RGB to ensure the compatability of the file
    with CV2 & numpy Functions.
    """
    return Image.open(path).convert('RGB')


def save_image(image, path, quality=100):
    """
    Save image as JPEG/PNG with specified quality.
    optimize: Finds the best Huffman encoding tables to compress the
    image (Reduces file size without change in image quality)
    subsapling: Controls how color detail is compressed in JPEG and JPG
    """
    ext = path.split('.')[-1].lower()  # ['ima','ge',"jpg"] ima.ge.jpg
    options = {}
    if ext in ('jpg', 'jpeg'):
        options = {'quality': quality, 'optimize': True, 'subsampling': 0}
    elif ext == 'png':
        options = {'compress_level': 1}
    image.save(path, **options)
    print(f"Saved: {path}")
```

2.Gamma Correction

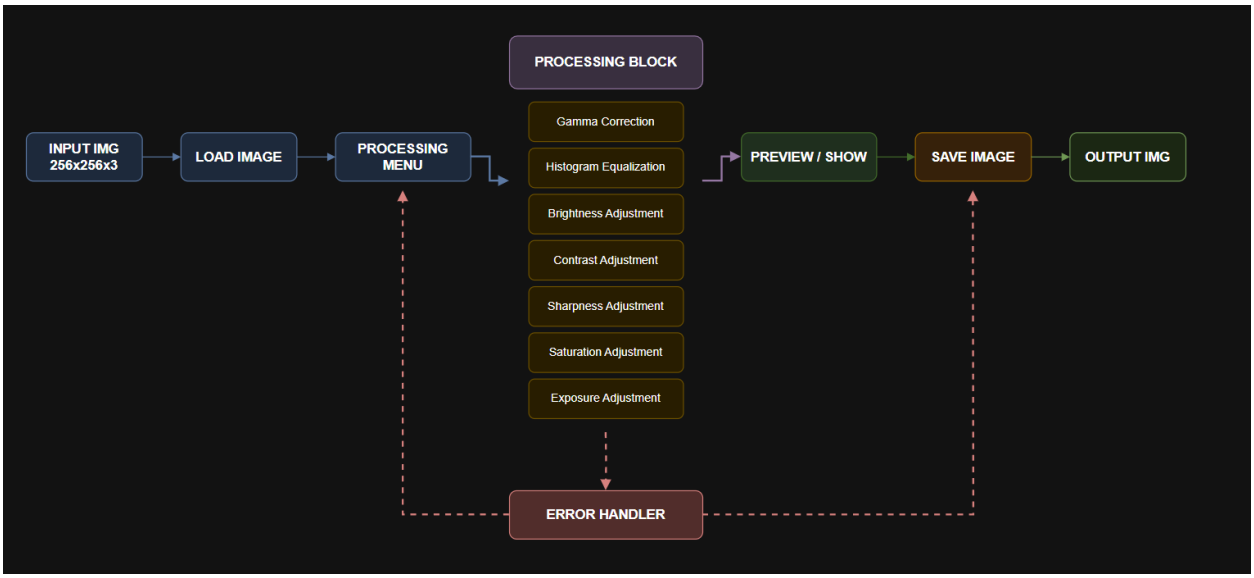
```python
def apply_gamma(image, gamma=1.0):
    # Non-linear mapping
    """
    Apply gamma correction and ensure values stay in [0,1] range.
    I"out" = I"in" power 1/gamma factor
    """
    arr = img_as_float(np.array(image))  # values in [0,1]
    corrected = exposure.adjust_gamma(arr, gamma)  # I_out = I_in power (1/gamma)
    corrected = np.clip(corrected, 0, 1)  # clip to valid range
    return Image.fromarray(img_as_ubyte(corrected))
```

3. Exception Handling

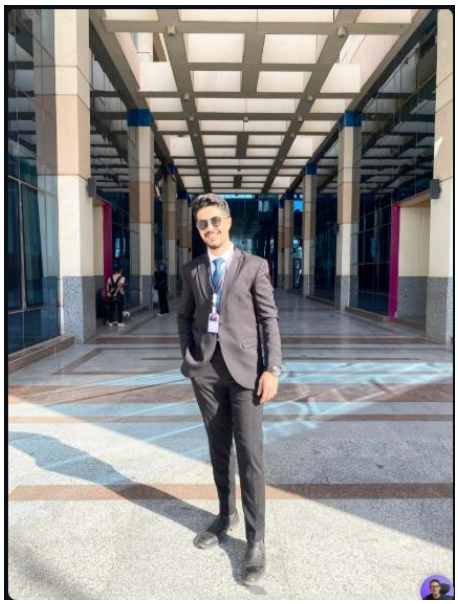
Exception Type	Location	Handling Strategy	Rationale
FileNotFoundError	load_image()	Catch and print error	Invalid input path
ValueError	All adjust functions	Input validation	Prevent invalid numeric params
IOError	save_image()	Catch write failures	Filesystem permissions/issues

4. Algorithm Comparison

Metric	skimage.equalize_hist	OpenCV YCrCb	Chosen Approach
Color Preservation	Poor	Excellent	OpenCV YCrCb
Speed	150ms (1MP image)	85ms	OpenCV YCrCb
Artifact Risk	High	Moderate	OpenCV YCrCb



Before VS After Equalization



Before VS After Changing the Values of Enhancement Controls

What Makes This Project Stand Out

- Demonstrates advanced yet essential image processing operations.
- Implements industry-grade techniques such as CLAHE and Canny edge detection.
- Modular and extensible: easy to integrate new filters or analysis tools.
- Practical application of theoretical concepts in digital image processing.
- Highly suitable for students, educators, and anyone learning computer vision.

References

1. **OpenCV (cv2)**
    - **Source**: OpenCV: Open Source Computer Vision Library
    - **Documentation**: OpenCV Documentation
2. **NumPy**
    - **Source**: NumPy: The Fundamental Package for Scientific Computing with Python
    - **Documentation**: NumPy Documentation
3. **Pillow (PIL Fork)**
    - **Source**: Pillow Documentation
    - **Documentation**: Pillow Documentation
4. **scikit-image (skimage)**
    - **Source**: scikit-image Documentation
    - **Documentation**: scikit-image Documentation