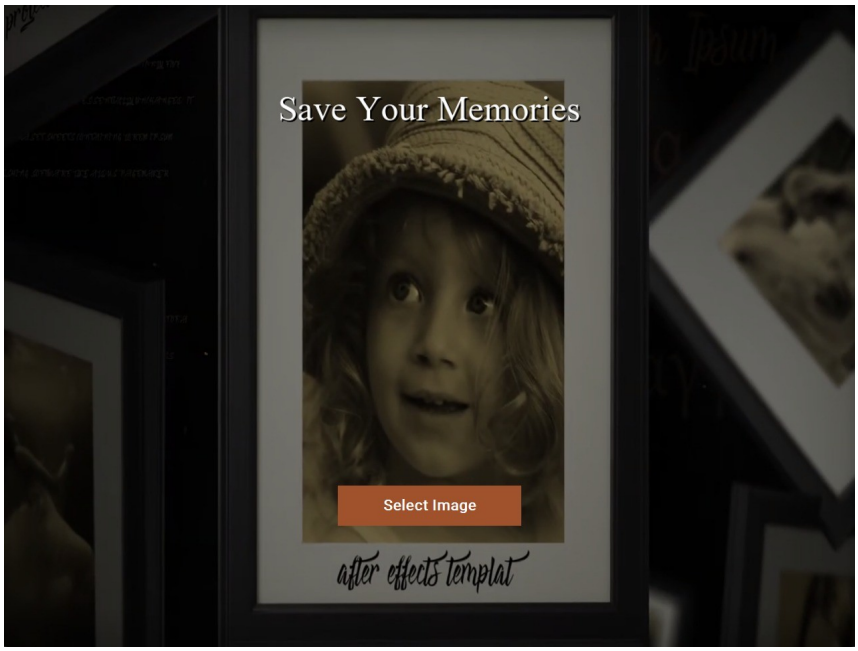
PyImgScan GUI

A Desktop Application for Document Image Scanning and Analysis



Presented to
Eng. Elias Zaghrini

Made by
Ghady Chouity
Elias Ishak

Course Name

Multimedia

Introduction

Document digitization has become an essential part of modern multimedia systems. Physical documents are frequently converted into digital form for archiving, sharing, and long-term preservation. However, images captured using cameras or mobile phones often suffer from several quality issues such as perspective distortion, glare, uneven lighting, and compression artifacts.

The **PyImgScan GUI** project aims to solve these problems by providing a desktop application that assists users in scanning and analyzing document images. The application integrates multiple image processing techniques into a simple graphical user interface, allowing users to enhance document images without requiring advanced technical knowledge.

This project focuses on practical multimedia concepts, including image preprocessing, enhancement, glare removal, and compression analysis. By combining these techniques into one system, PyImgScan GUI provides an effective solution for improving the visual quality and usability of scanned or photographed documents.

Table of Contents

Project Objectives

The PyImgScan GUI project was developed with the goal of creating a practical and easy-to-use document scanning and analysis tool. The main objectives of this project are listed below.

- To design a simple and intuitive graphical user interface for document image processing
- To allow users to load and visualize document images easily
- To automatically detect and crop documents from photographed images
- To provide a manual cropping option for better accuracy and user control
- To reduce glare and specular highlights that commonly appear in document photos
- To analyze the effect of image compression on visual quality
- To compute objective image quality metrics such as PSNR, SSIM, and MSE
- To enable users to save the enhanced document images for digital use

These objectives reflect core multimedia concepts such as image enhancement, quality evaluation, and user-centered interface design.

System Overview

PyImgScan GUI is a desktop-based multimedia application developed using Python. The system is designed to process document images and improve their quality through several image processing techniques. It provides an integrated environment where users can perform document cropping, glare removal, and compression analysis using a graphical interface.

The application is organized into multiple functional modules. Each module is responsible for a specific task, and together they form a complete document image processing pipeline. The modular design makes the system easier to understand, maintain, and extend in future versions.

Application Workflow

The workflow of PyImgScan GUI follows a clear sequence of steps that guides the user through the image processing process.

1. The user selects a document image from the local computer
2. The selected image is displayed in the main editor window
3. The user applies document cropping using automatic or manual mode
4. Optional glare removal techniques are applied
5. Compression analysis may be performed if required
6. The final processed image is saved

Figure: Application workflow of PyImgScan GUI

Graphical User Interface Design

The graphical user interface of PyImgScan GUI was designed to be simple, clear, and easy to use. The main goal of the interface is to allow users to process document images efficiently without requiring advanced knowledge of image processing techniques.

All tools are organized logically, enabling users to follow a natural workflow from image selection to saving the final result. The interface minimizes complexity while still offering advanced processing options.

Main Window

The main window is the central part of the application. It displays the currently loaded image and provides access to all processing tools through a side panel.

The left sidebar contains buttons for document cropping, glare removal, compression analysis, and image replacement. The bottom section of the window includes controls for undoing and redoing actions, as well as saving the processed image.

This layout allows users to clearly see the effect of each operation applied to the image in real time.

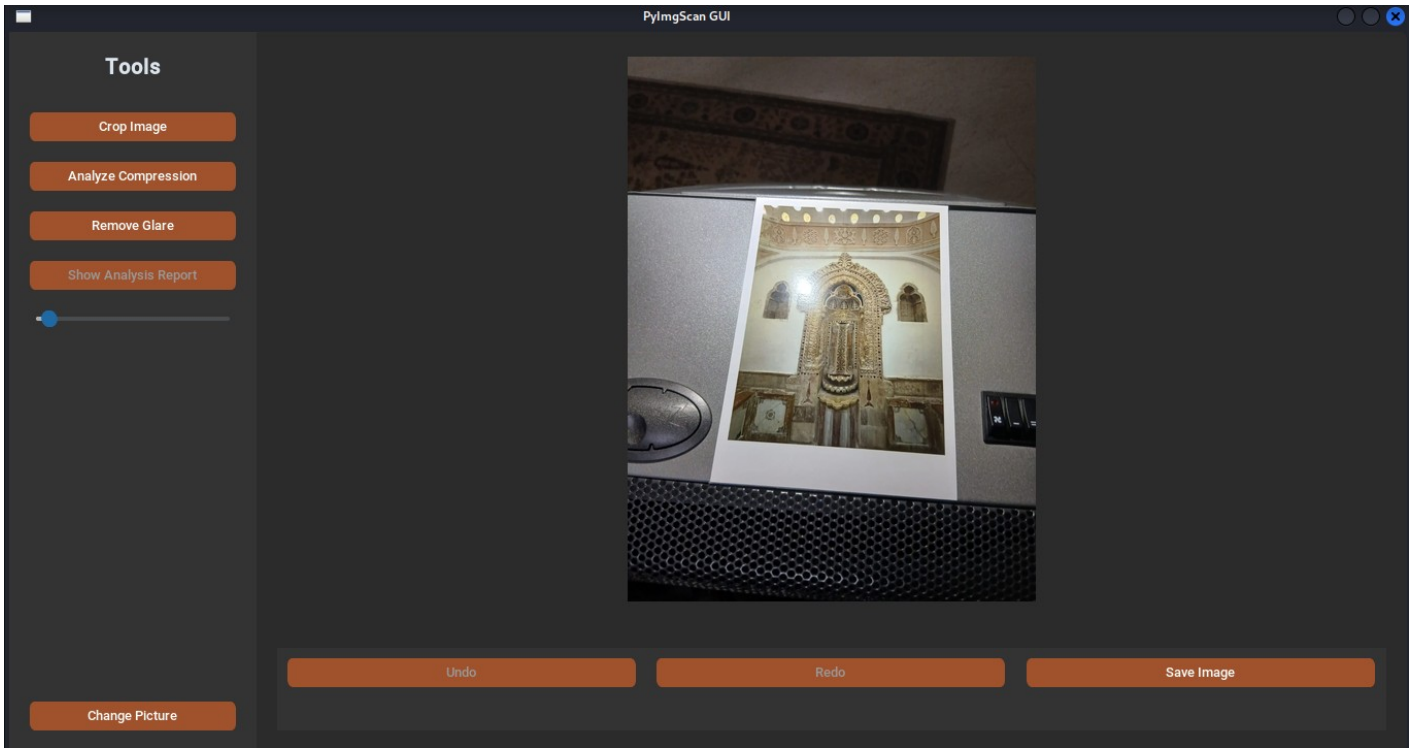


Figure: Main editor window of PyImgScan GUI

Document Cropping Module

Document cropping is a critical step in the document digitization process. The goal of this module is to accurately isolate the document area from the background and correct perspective distortions caused by camera angles.

PyImgScan GUI provides two cropping modes: automatic cropping and manual cropping. This flexibility ensures reliable results across different types of document images.

Automatic Cropping

In automatic mode, the cropping process follows a **two-stage detection approach**. In the first stage, the system detects the outer boundary of the captured image or scanned page in order to localize the general region of interest. In the second stage, the algorithm focuses on identifying the actual document region within this boundary.

To achieve this, the input image is processed to enhance structural information, and edge-based techniques are used to highlight strong intensity transitions corresponding to document borders. From these detected edges, contours are extracted, and the most significant contour is selected as the document candidate. This contour is then approximated to a quadrilateral, allowing the four corner points of the document to be estimated accurately.

This two-stage strategy improves detection reliability, particularly when documents are photographed in complex or cluttered environments. Automatic cropping is efficient, requires minimal user interaction, and produces accurate results for most standard document images.

Manual Cropping

Manual cropping allows the user to define the document boundaries interactively. A new window opens with a draggable rectangular frame where the user can adjust the four corner points of the document.

This mode is particularly useful when automatic detection does not produce accurate results or when precise control is required.

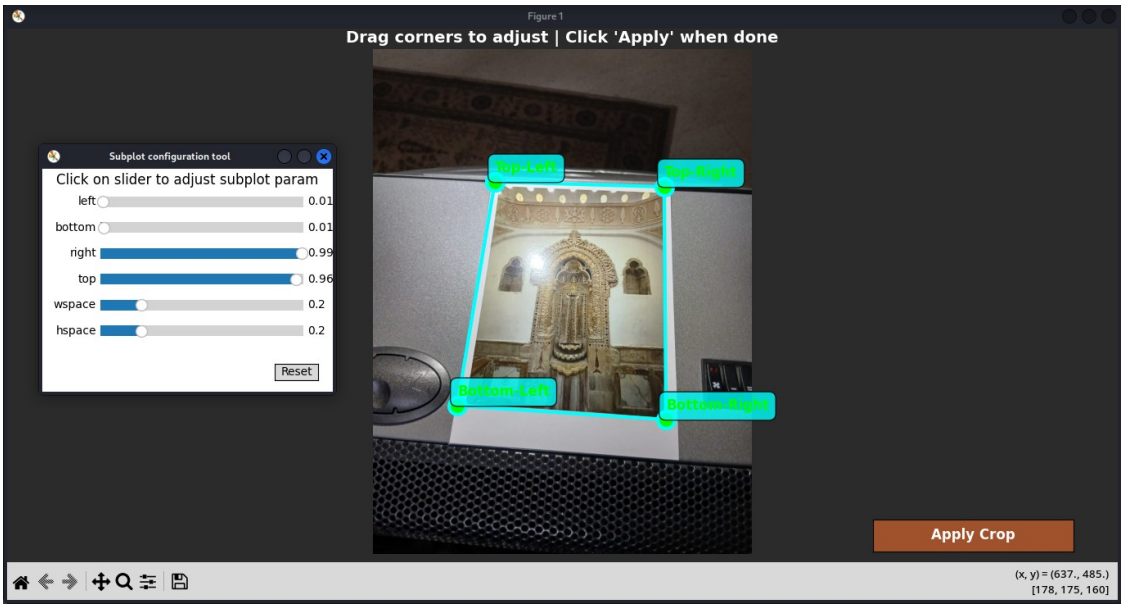


Figure: Manual document cropping interface

Glare Removal Techniques

Glare and specular reflections are common problems when documents are photographed under strong lighting conditions. These reflections can obscure text and reduce overall image quality.

The PyImgScan GUI application offers multiple glare removal techniques to handle different glare patterns effectively.

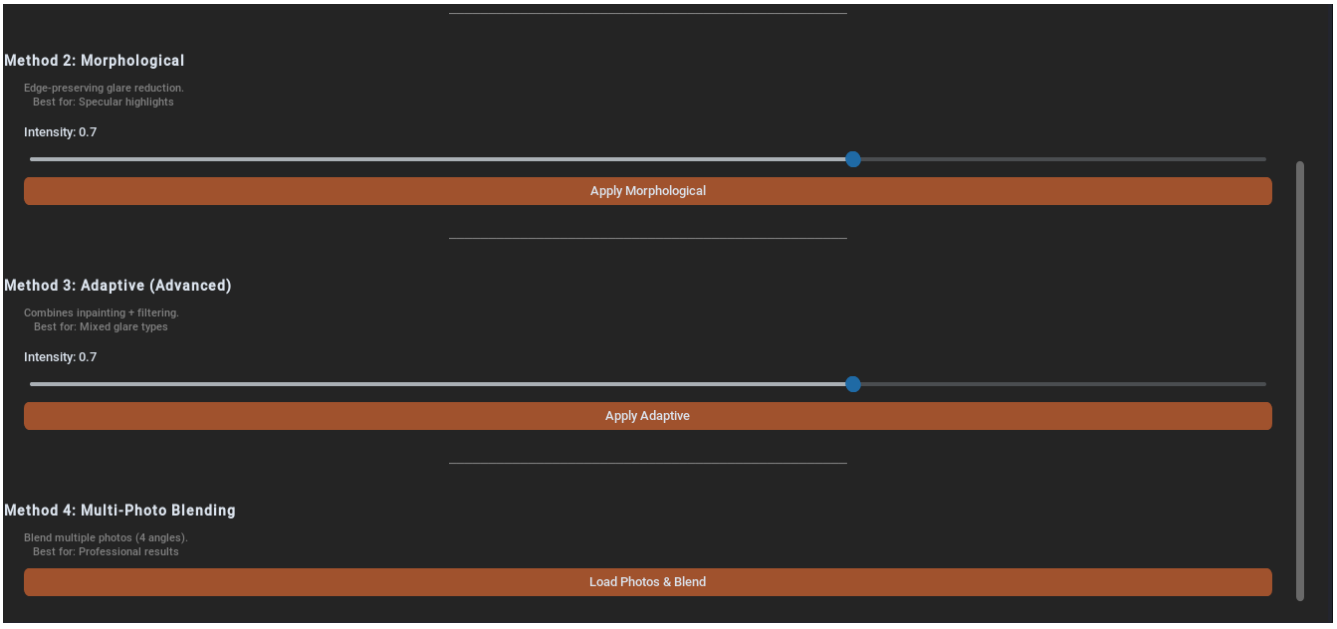


Figure: Glare removal techniques

Single-Image Glare Removal

Single-image glare removal methods operate on one image only and are suitable for most common scenarios.

The available methods include inpainting, morphological processing, and an adaptive approach. Inpainting reconstructs glare regions by sampling surrounding pixels, while morphological processing reduces bright reflective areas. The adaptive method combines multiple techniques to achieve more robust results.

Multi-Photo Blending

Multi-photo blending is an advanced glare removal technique that requires four images of the same document taken from different angles. These images are captured from the top, bottom, left, and right perspectives.

The algorithm analyzes each image and blends the regions with minimal glare to generate a final glare-free image. This method produces professional-quality results but requires additional input images.

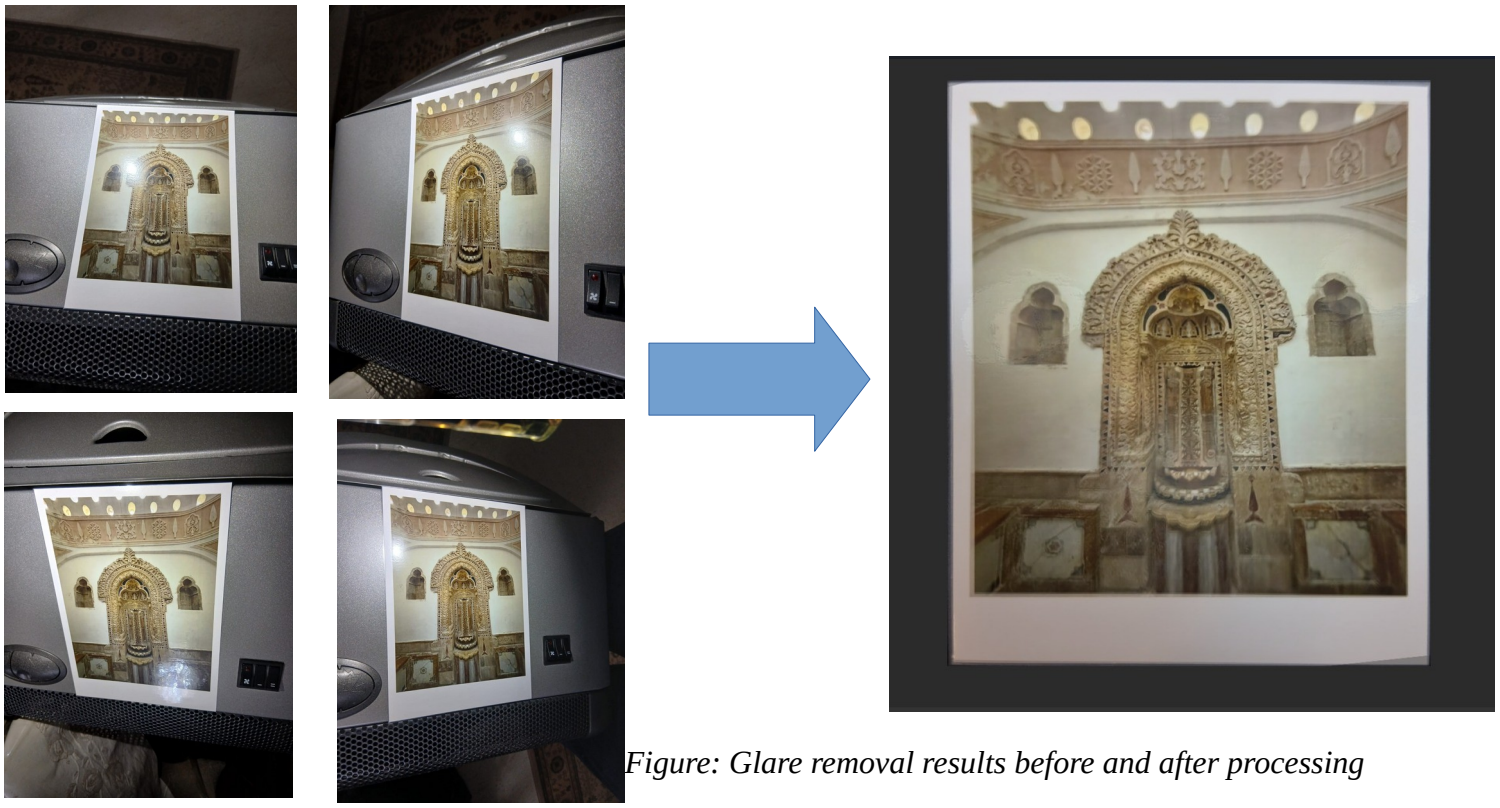


Figure: Glare removal results before and after processing

Compression Analysis Module

Compression is an essential step in digital document management, especially for storage and sharing. PyImgScan GUI includes a compression analysis module that evaluates the effects of JPEG compression on image quality.

The module allows users to compress images to specific target sizes (e.g., 30KB, 100KB, 500KB, 1MB) and analyze the resulting quality. This helps users make informed decisions about balancing file size and visual quality.

Quality Metrics

The system computes objective image quality metrics to assess compression effects:

- **PSNR (Peak Signal-to-Noise Ratio):** Measures the ratio between the maximum possible power of a signal and the power of corrupting noise. Higher values indicate better quality.
- **SSIM (Structural Similarity Index):** Evaluates the similarity between the original and compressed images in terms of structure and luminance.
- **MSE (Mean Squared Error):** Quantifies the average squared difference between original and compressed pixel values.

These metrics provide quantitative insight into the trade-offs caused by compression.

Rate-Distortion Analysis

The compression module also generates a **rate-distortion plot**, showing the relationship between file size and image quality. Users can easily identify optimal compression points that preserve quality while reducing storage requirements.

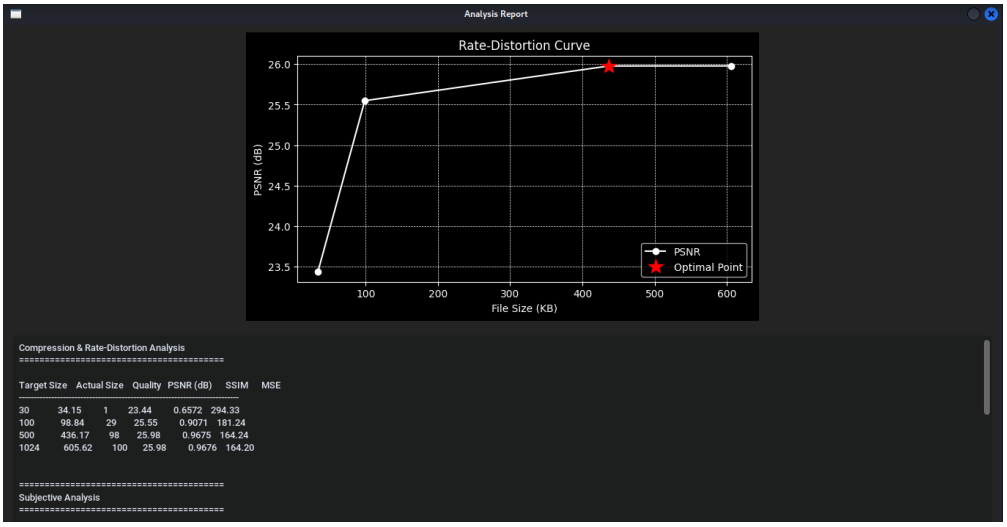


Figure: Rate-distortion curve for JPEG compression

Undo/Redo and Image Management

To improve workflow efficiency, PyImgScan GUI includes an **undo/redo system**. This allows users to revert or re-apply any recent changes to the image, providing flexibility during editing.

In addition, the **Change Picture** feature lets users load a new image into the editor without restarting the application. This feature is particularly useful when processing multiple documents sequentially.

The **Save Image** button ensures that the final processed image can be stored on the user's computer for further use or archiving.

Image Acquisition Setup

The document images used in this project were captured using a **Samsung Galaxy A16 smartphone camera** in order to simulate real-world usage by everyday users rather than professional scanning devices. The camera provides a **12-megapixel color resolution** with adequate dynamic range for document imaging. All images were taken under **normal indoor room lighting conditions**, without the use of specialized lighting equipment, and with a **moderate distance** between the camera and the document. This setup ensures that the system is evaluated under practical and realistic conditions.

Tools and Technologies Used

- **Programming Language:** Python 3
- **Libraries:** OpenCV, NumPy, PIL, scikit-image, Tkinter
- **Development Environment:** Linux / Windows
- **Application Type:** Desktop GUI Application

Conclusion

The PyImgScan GUI project demonstrates a complete solution for document digitization and enhancement. By integrating automatic and manual cropping, glare removal, compression analysis, and quality evaluation in a single graphical interface, the system provides a practical and user-friendly tool for improving document image quality.

Future improvements could include:

- Integrating OCR for text extraction
- Adding batch processing for multiple documents
- Supporting additional image formats
- Enhancing multi-photo glare removal with more sophisticated blending techniques

PyImgScan GUI highlights the application of multimedia and image processing concepts in real-world scenarios, providing both educational value and practical utility.

References

1. Gonzalez, R. C., & Woods, R. E. *Digital Image Processing*. Pearson.
2. OpenCV Documentation.
3. Scikit-image Documentation.