OCR Tool Documentation

# Introduction

This document provides an overview of the OCR (Optical Character Recognition) tool developed to extract text from images. The OCR tool utilizes Tesseract OCR, an open-source software library, and Python to process images of documents, receipts, and other scanned text, converting their contents into editable .txt files. This tool is beneficial for automating text extraction, allowing digital handling and organization of physical or scanned documents.

# Implementation

The OCR tool is implemented in Python with the following key components:

### Libraries Used:

* **pytesseract**:

A Python wrapper for Tesseract OCR that enables easy text extraction from images.

* **Pillow (PIL)**:

A Python Imaging Library used to open and handle image files in various formats.

* **os**:

Provides functionality to handle directory paths, check file existence, and create new folders.

* **json**:

Allows formatting of extracted data into a JSON object for structured output and easy viewing.

### Process Flow:

* **Folder Reading**:

The tool reads each image within a specified folder, supporting formats like .png, .jpg, .jpeg, .bmp, and .tiff.

* **OCR Text Extraction**:

For each image, the tool applies Tesseract OCR to extract the text content. The extracted text may include unnecessary spaces or artifacts based on the quality of the source image.

* **Output Storage**:

The tool creates an output folder within the specified directory (if it does not already exist) and saves each extracted text as a .txt file. Each text file is named after the original image to maintain easy referencing.

# How the OCR Tool Works

### Initialize Input and Output Paths:

* The user specifies an input folder containing images for OCR processing.
* The tool creates an output folder within the directory to save extracted text files.

### OCR Text Extraction Function:

* The main function, ocr\_text\_extraction, processes each image file by applying Tesseract OCR to convert image text into a string. Each image is opened and processed in formats such as .png, .jpg, and .jpeg.

### Saving Extracted Text:

* After extraction, the text is saved in the output folder as a .txt file, with the filename matching the original image name to ensure easy identification and access.

# Challenges Faced

During the development and testing phases, several challenges were encountered, including:

1. **Image Quality Variations**:

OCR accuracy relies significantly on the input image quality. Low-resolution or noisy images caused incorrect text extraction. To improve accuracy, additional image preprocessing techniques like grayscale conversion and noise reduction may be added.

1. **Unsupported File Formats**:

Initially, unsupported file types led to errors during processing. The tool now includes a check for compatible file extensions, ensuring only compatible image formats are passed to Tesseract OCR.

1. **Text Formatting Issues**:

The raw OCR text often contained unwanted spaces and line breaks, particularly for complex document layouts. The tool includes minimal text preprocessing to clean up redundant spaces, but additional formatting adjustments may be necessary depending on the specific requirements of the user.

# Conclusion

The OCR tool efficiently converts text within images to an editable text format, supporting the digital storage and organization of document information. With Python and Tesseract OCR, this tool provides a simple yet effective solution for basic text extraction. It also offers flexibility for future enhancements, including additional preprocessing steps to improve accuracy and more comprehensive text cleanup based on specific document types and layouts.