Kshetra Consulting Assignment

**Brief Report: Image Processing and Text Extraction Workflow**

**Introduction**

This report outlines a comprehensive workflow for image processing and text extraction. The process involves importing necessary libraries, reading images, displaying images, and extracting data from images using various image processing techniques. The workflow is designed to handle structured images with multiple data boxes, applying preprocessing techniques to improve text extraction accuracy.

**Libraries and Setup**

I imported various libraries to facilitate the image processing and text extraction tasks: **Pytesseract** used for optical character recognition (OCR), **PIL** for handling and processing images, **numpy** for numerical operations and image array handling, **cv2 (OpenCV)** for advanced image processing techniques., **matplotlib.pyplot** for visualizing images and results.I set the path to Tesseract OCR to executable path. This configuration ensures that pytesseract can locate and utilize the Tesseract OCR engine for text extraction.

**Reading Images**

I read the images into the program using OpenCV.

**cv2.imread()** loads images from specified file paths into NumPy arrays for processing.

**read\_img1** and **read\_img2** are variables that store the loaded image data, making it ready for further manipulation.

**Display Function**

I developed the display() function to visualize images using Matplotlib. The function reads an image from a file path, determines its dimensions, and displays it using Matplotlib. The function handles both grayscale and color images by determining the image dimensions and adjusting the display parameters accordingly. This approach ensures that images are shown at their actual size and resolution. The display() function provides a straightforward method to visualize images while handling different formats and maintaining resolution. It calculates the appropriate figure size, adjusts the display settings, and ensures the image is shown clearly without additional axis clutter.

**Contour Detection and Image Processing**

The getContours() function processes images to detect and analyze contours. This function is aimed at extracting contours (boundaries of objects) from images that contain structured boxes, processing the region of interest (ROI) for each contour, and performing Optical Character Recognition (OCR) to extract text. This section of code detects and processes contours in an image, extracts regions of interest (ROIs), and performs OCR to extract text.

Steps Involved:

1. **Contour Detection and Sorting**: Finds and organizes contours
2. **ROI Extraction**: Crops significant regions based on contour size.
3. **Grayscale Conversion**: Simplifies the image for contour detection.
4. **Blurring**: Reduces noise to improve thresholding results.
5. **Thresholding and Dilation**: Enhances contours.
6. **Image Transformation**: Applies inversion, grayscale conversion, thresholding, and noise removal to prepare the image for OCR.
7. **Inversion:** The image is inverted to facilitate better OCR results.
8. **Grayscale Conversion:** The ROI is converted to grayscale, which is a necessary step for many image processing techniques.
9. **Thresholding:** Converts the grayscale ROI to a binary image.
10. **Noise Removal:** A series of morphological operations are applied to reduce noise in the binary image.
11. **OCR and Post-Processing**: Extracts text and cleans up the result.

**Challenges and Solutions**

1. **Handling Different Image Formats:**
   * **Challenge:** The function needs to work with both grayscale and color images.
   * **Solution:** I used the number of dimensions (ndim) to differentiate between grayscale (2D) and color (3D) images. Adjusted the colormap accordingly.
2. **Maintaining Image Resolution:**
   * **Challenge:** Ensuring that the displayed image maintains its original resolution.
   * **Solution:** Calculated the figure size based on the image dimensions and dpi, which preserves the aspect ratio and resolution.
3. **Contour Sorting:**

* **Challenge:** The contours needed to be processed in a specific order corresponding to the grid structure.
* **Solution:** Implemented sorting based on the bounding rectangle coordinates to ensure the contours were processed in a top-to-bottom, left-to-right order.

1. **Inconsistent Font Thickness:**

* **Challenge:** Variability in text thickness within the boxes led to inconsistent OCR results.
* **Solution:** Introduced font thinning and thickening techniques to standardize text appearance, improving OCR consistency.

I tried to extract data from the images but couldn’t extract properly because my code couldn’t detect the text exactly which made it difficult for me to clean the data according to name, relative name, …. Etc.

So I am submitting how much ever I have done, which is preprocessing and OCR.

Thank you for the opportunity. I actually learnt a lot of new things.