**A program to separate text and visual elements from images.**

(Internship Assignment Report)

Submitted by Submitted to

Ayush Pareek TestLine

(IIIT Dharwad)

**Introduction**

Functionality of this program leverages the Tesseract OCR library to analyze images. The program performs text extraction, visual element segmentation.

I also tried to implement a webpage to take image input but that didn’t work well. Still I included the templates, implementation details and other files.

**Approach**

The program employs a systematic approach to image analysis by integrating the Tesseract OCR library with OpenCV for preprocessing and segmentation. Initially, it sets up Tesseract for OCR operations. The program then preprocesses the input image by converting it to grayscale and applying thresholding to enhance text region detection. Contours are identified to locate these regions, and OCR is performed on each extracted region to retrieve text. Subsequently, the program removes the detected text from the image to isolate visual elements, which are saved separately. Finally, the extracted text and visual elements are compiled into an HTML file, providing a structured and accessible representation of the image content. This methodical approach ensures efficient text extraction and visual element segmentation, facilitating comprehensive image analysis.

**Technologies Used**

* Python
* Tesseract for OCR
* OpenCV
* OS module
* Html and Flask
* NumPy

**Implementation Details of Each Function**

**1. inst()**

The inst() function is responsible for setting up the Tesseract OCR library. It configures the path to the Tesseract executable, ensuring that it is correctly installed and accessible for the program to use its OCR capabilities.

**2. preprocess\_image(image\_path)**

This function handles the initial preprocessing of the input image. It performs the following steps:

Reads the image from the specified path using OpenCV.

Converts the image to grayscale to simplify further processing.

Applies thresholding to create a binary image, which makes the text regions more distinct by turning the background white and the text regions black.

The detect\_text\_regions() function is used to identify potential text regions within the binary image. Find contours in the thresholded image using OpenCV. Contours are the outlines or boundaries of shapes in the image.

Returns a list of these contours, which are potential text regions.

**3. detect\_text\_regions(thresh)**

The detect\_text\_regions() function is used to identify potential text regions within the binary image. It finds contours in the thresholded image using OpenCV. Contours are the outlines or boundaries of shapes in the image.

returns a list of these contours, which are potential text regions.

**4. extract\_text(image, contours)**

This function extracts text from the identified text regions.

It Iterates over each contour to extract the region of interest (ROI) containing potential text.

Uses Tesseract OCR to perform text recognition on these ROIs.

Stores the extracted text and its corresponding bounding box coordinates in a list.

Returns the list of extracted text data.

**5. remove\_text(image, contours)**

The remove\_text() function removes the detected text regions from the image to help isolate visual elements.

It Iterates over each contour. Fills the bounding box of each contour with white color, effectively removing the text from the image.

returns the modified image with the text removed.

**6. save\_visual\_elements(image, contours)**

This function isolates and saves visual elements from the image. It Iterates over each contour to extract the region of interest (ROI) corresponding to each visual element.

Saves each ROI as an individual image file in a specified directory.

Returns a list of file paths to the saved visual element images.

**7. generate\_html(text\_data, visual\_elements)**

The generate\_html() function creates an html file that displays the extracted text and visual elements. It Initializes an html string with basic html structure.

Adds each piece of extracted text as a paragraph in the html body.

Embeds each visual element image as an image tag in the html body.

Completes the html structure and returns the html content as a string.

**8. help(image\_path)**

The help() function orchestrates the entire workflow of the program. It will make a call to the desired functions and as an output it saves the generated HTML content to a file named output.html and returns the extracted text data.

**Challenges encountered**

The primary challenge I encountered while developing this program was obtaining specific images to test the Python script effectively. Having access/reference to a dataset or a collection of standard images would have been immensely beneficial for validating and refining the functionality of the program.

**References**

1. ChatGpt
2. <https://github.com/tesseract-ocr/tesseract>
3. <https://www.youtube.com/watch?v=WwNuvGLblJU>
4. <https://medium.com/poatek/understanding-ocr-your-entertaining-guide-to-optical-character-recognition-5891d31159c6>
5. <https://www.youtube.com/watch?v=2kWvk4C1pMo&pp=ygUWaG93IHRvIHNldHVwIHRlc3NlcmFjdA%3D%3D>