**Assignment 5**

**1. Objective**

The purpose of this assignment is to **detect and extract text from images** using **Tesseract OCR** and **OpenCV**. This approach is widely used in document digitization, automatic number plate recognition, and information extraction from scanned images.

**2. Libraries and Tools**

* **OpenCV**: For image processing and visualization.
* **pytesseract**: Python wrapper for Tesseract OCR engine.
* **Tesseract OCR**: Open-source OCR engine for text recognition.
* **Google Colab**: Environment for running and testing the code.

**3. Methodology**

**3.1 Image Preprocessing**

Preprocessing improves OCR accuracy. Steps include:

1. **Grayscale Conversion**: Convert the input image to grayscale using cv2.cvtColor.
2. **Noise Removal**: Use cv2.fastNlMeansDenoising to reduce noise.
3. **Binarization**: Apply **Otsu’s thresholding** to convert the image into binary format for better text contrast.

**3.2 Text Detection**

* Detect text regions in the preprocessed image using pytesseract.image\_to\_data.
* Output includes bounding box coordinates, confidence levels, and recognized text.

**3.3 Drawing Bounding Boxes**

* Only text regions with confidence > 60% are considered.
* Draw rectangles around detected text using OpenCV for visualization.

**3.4 Text Extraction**

* Extract text from the detected regions.
* Only include high-confidence predictions.

**4. Implementation**

* Load the input image using OpenCV.
* Preprocess the image to enhance text clarity.
* Detect text regions and mark them on the image.
* Extract and display the detected text.
* Display both the **original image**, **image with bounding boxes**, and **extracted text**.

**5. Results**

* **Original Image**: Input image with text.
* **Text Detection Image**: Rectangles highlight text regions.
* **Extracted Text**: The detected text is printed in the console.

**Sample Output:**

Detected Text:

This is an example of text extracted from the image.

* The bounding boxes accurately capture text regions with high confidence.
* Preprocessing significantly improves the OCR accuracy by reducing noise and enhancing contrast.

**6. Conclusion**

* This project demonstrates **optical character recognition using Tesseract and OpenCV**.
* Image preprocessing (grayscale conversion, denoising, and binarization) plays a key role in improving detection and extraction accuracy.
* Applications include: document scanning, number plate recognition, invoice automation, and digitization of printed text.