PROJECT ABSTRACT

Clipboard Image to Text Extractor (OCR)

An on-device OCR utility: paste any screenshot from the clipboard and extract text through a classical pipeline (binarization, projection-based segmentation, template matching against live-rendered glyphs), with per-character confidence, bounding boxes and a binarized debug view — nothing uploaded. Built to order with source, engine documentation, samples and complete viva kit.

01. Title

Clipboard Image to Text Extractor (OCR)

02. Introduction / Background

Students constantly need text trapped inside images — a notice-board photo, a scanned invoice, a whiteboard full of action items, a screenshot of an error message. Retyping is slow and error-prone, while cloud OCR tools upload images to someone else's server. This utility keeps everything on the device: paste any image from the clipboard, and a real, classical OCR pipeline extracts the text — grayscale conversion and binarization, line segmentation by horizontal projection profile, character segmentation by vertical projection, then template matching against glyphs rendered live from the same font. Every character carries a confidence score, low-confidence characters flag in red, and bounding boxes overlay the source image for verification.

03. Problem Statement

Useful text is locked in images, retyping is wasteful, and cloud OCR leaks potentially sensitive screenshots. Image-processing courses teach binarization, projections and template matching, but students rarely see the full pipeline running end to end. This project makes it real and private: a complete on-device OCR pipeline with visible confidence, bounding boxes and the binarized “what the engine sees” view.

04. Objectives

- Accept input via clipboard paste (Ctrl+V), file upload and three built-in samples.
- Implement the full classical pipeline: binarization, horizontal/vertical projection segmentation, 26×34 normalized glyph matching against 78 live-rendered templates.
- Score every character with an agreement-based confidence and heatmap it (amber/red).
- Overlay toggleable character bounding boxes on the source image.
- Show the per-glyph normalized bitmap and confidence in a detail grid.
- Provide a before/after binarized preview as the debugging view.
- Deliver the engine documentation, samples and full viva kit (report, PPT, Q&A;).

05. Proposed Methodology

The pasted image is drawn to a canvas, converted to grayscale and binarized with a fixed threshold. A horizontal projection profile (dark-pixel count per row) segments the bitmap into text lines; within each line, a vertical projection profile segments characters — narrow gaps merge split glyphs, wide gaps mark word spaces. Each glyph is cropped and aspect-preserved into a 26×34 normalized bitmap, then compared pixel-by-pixel against templates rendered live from the same monospace font (78 glyphs: A–Z, a–z, 0–9, punctuation). The best-matching template wins; confidence comes from the agreement margin over the runner-up. Results render as text with a confidence heatmap, bounding boxes on the source, and the per-character detail grid.

06. System Architecture / Working

A single HTML file hosts the canvas pipeline: input panel (paste/upload/samples), the binarized before/after view, the recognized text view with confidence heatmap, the bounding-box overlay on the source image, and the per-glyph detail grid showing each normalized bitmap with its confidence. One-click copy moves extracted text to the clipboard. There are no network calls at all — everything runs in the browser on the device, so sensitive screenshots never leave the machine.

07. Hardware / Software Requirements

- Any modern desktop web browser with JavaScript and the Clipboard API
- No server, no network, no libraries — single HTML file, zero dependencies
- Clean printed/monospace-style text inputs (handwriting is out of scope)
- Optional: text editor to inspect the source
- Git

08. Expected Outcomes

- A working on-device OCR pipeline with segmentation, template matching and confidence scoring.
- Visual verification tools: bounding boxes, binarized preview, per-glyph bitmaps.
- Honestly documented scope — clean printed text; handwriting and stylized fonts out of scope — and the classical-vs-neural tradeoff explained.
- Three sample images with expected outputs for demonstration and testing.
- A full viva kit: report on image-processing theory, PPT and Q&A; on binarization, projections, template matching and OCR limits.

09. Applications

- Extracting text from screenshots, notice boards, invoices and whiteboards.
- Teaching image processing: binarization, projection profiles, template matching.
- Privacy-respecting OCR where uploads are unacceptable.

10. Future Scope

- Adaptive/Otsu thresholding for uneven lighting.
- Skew detection and correction pre-processing.
- A neural-network recognizer variant for comparison.
- Multi-language glyph sets and proportional-font support.