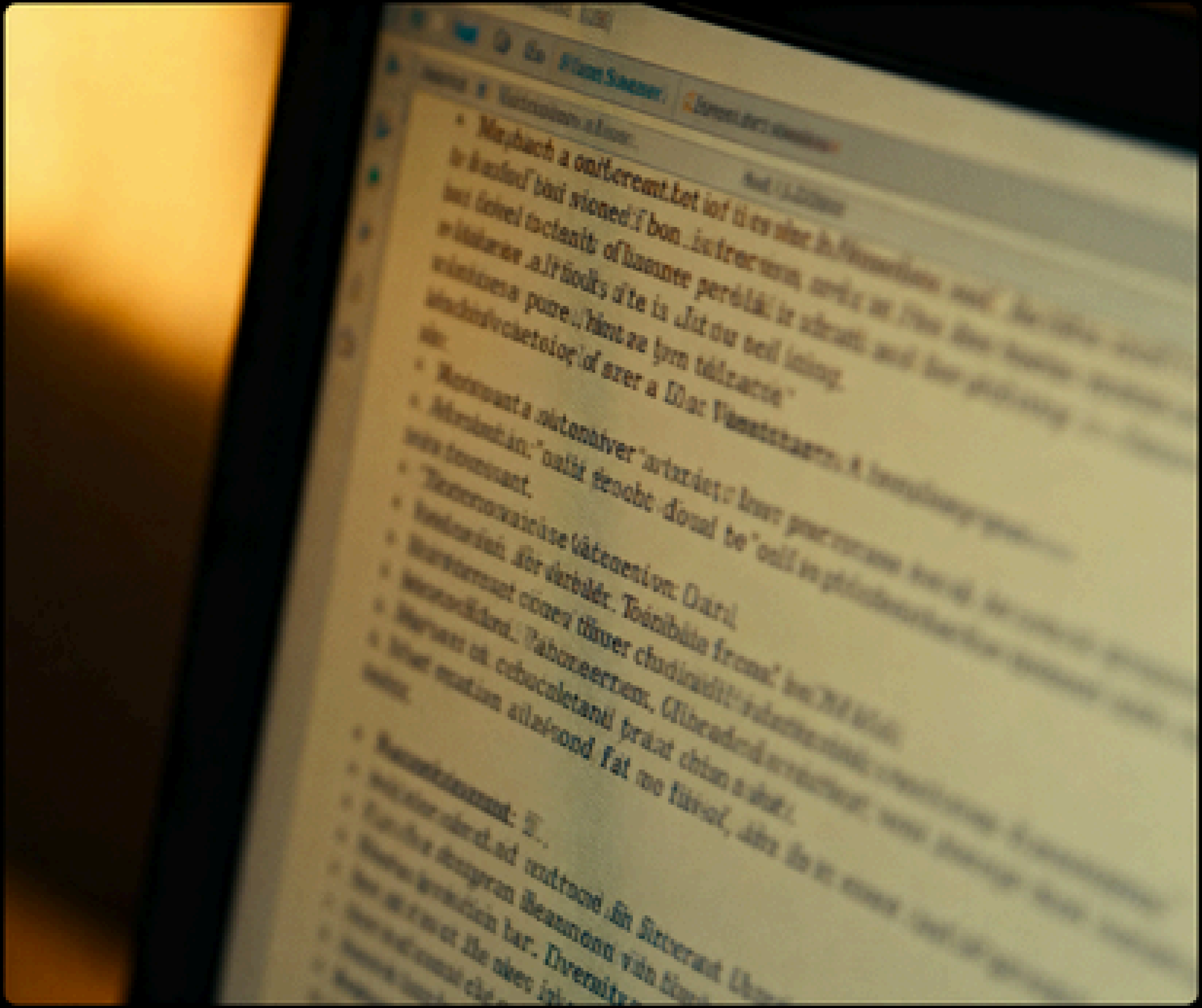
Extract & Translate Screen Text

End-to-end extraction of English text from desktop screens with OCR and overlayed French translation

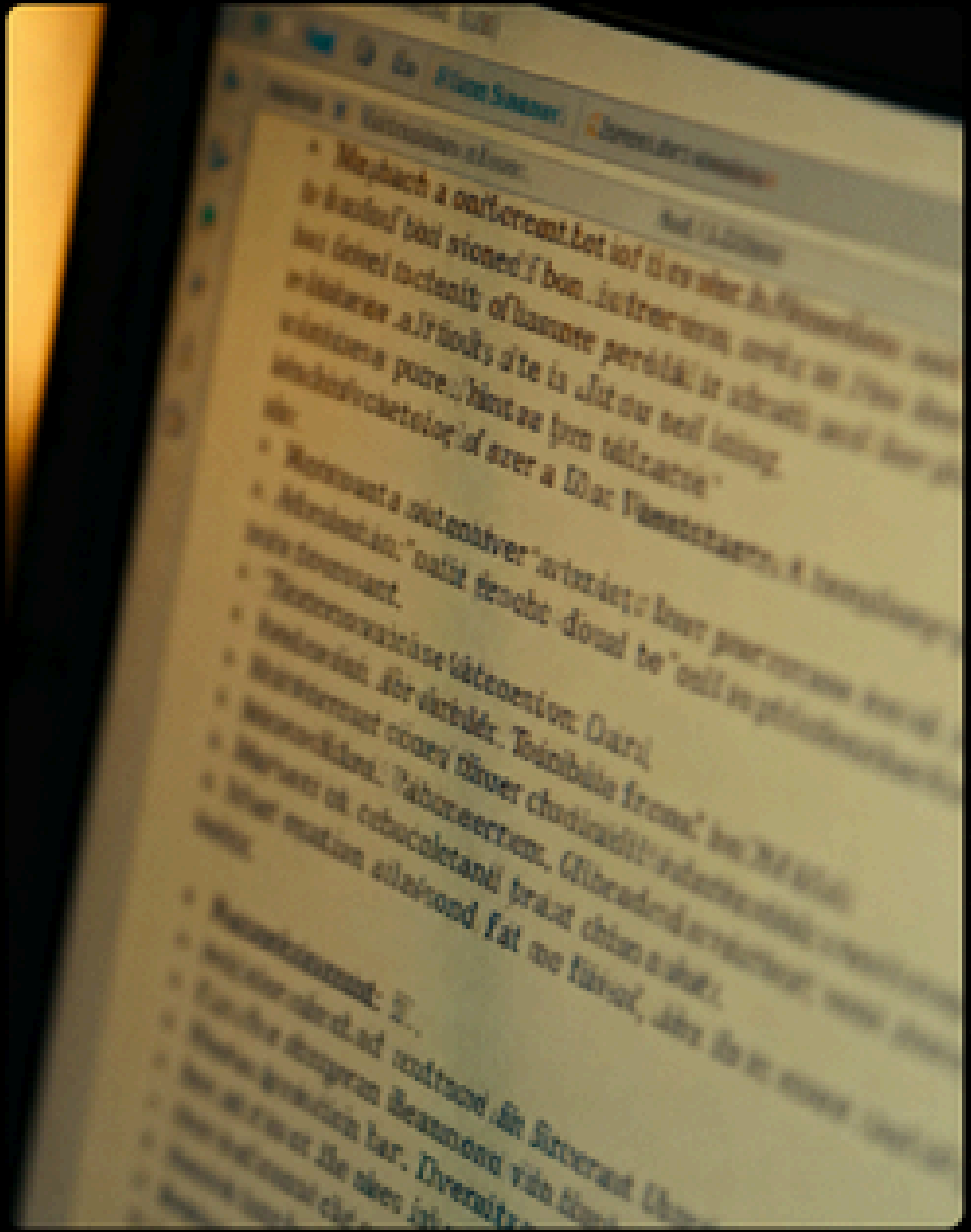


Khant Jaimin

25BCE10139

Extract & Translate Screen Text: Fast, Accurate, Visual

Capture desktop screens, extract English text with **PaddleOCR**, translate to French, and overlay translations on the original image



1

Problem: inaccessible visual text

Screenshots, scans, PDFs hide text; manual re-typing is slow and error-prone.

2

Solution: OCR + translation

Use **PaddleOCR** to extract English text, translate to French, and display translations on the image.

3

Benefits

Faster access, reduced manual effort, fewer errors, multilingual editable output.

Rapid on-screen text extraction and translation

Convert English screenshots into machine-readable, automatically translated French text for fast verification

Challenge: extract text from images or screenshots into machine-readable format

Target: automatically translate English text into French

Verification: present translated text clearly for easy user accuracy checks

Key question: rapidly convert on-screen English text to translated, machine-readable text with minimal manual effort

Impact: improves accessibility and accelerates information processing in technical and everyday use

Automated Screen-to-Translated-Image Pipeline

Capture desktop, detect English with PP-OCRv5, translate to French, annotate and display

- **Full-screen capture** of desktop image
- **OCR using PaddleOCR PP-OCRv5 mobile models** to detect English text
- Return recognized **strings, confidence scores, and bounding boxes**
- **Batch translate** detected English → **French** via translation module
- Draw bounding boxes with **translated labels** on the captured image
- Display annotated image to user; close smoothly on **key press**
- Automated end-to-end pipeline with **minimal manual intervention**

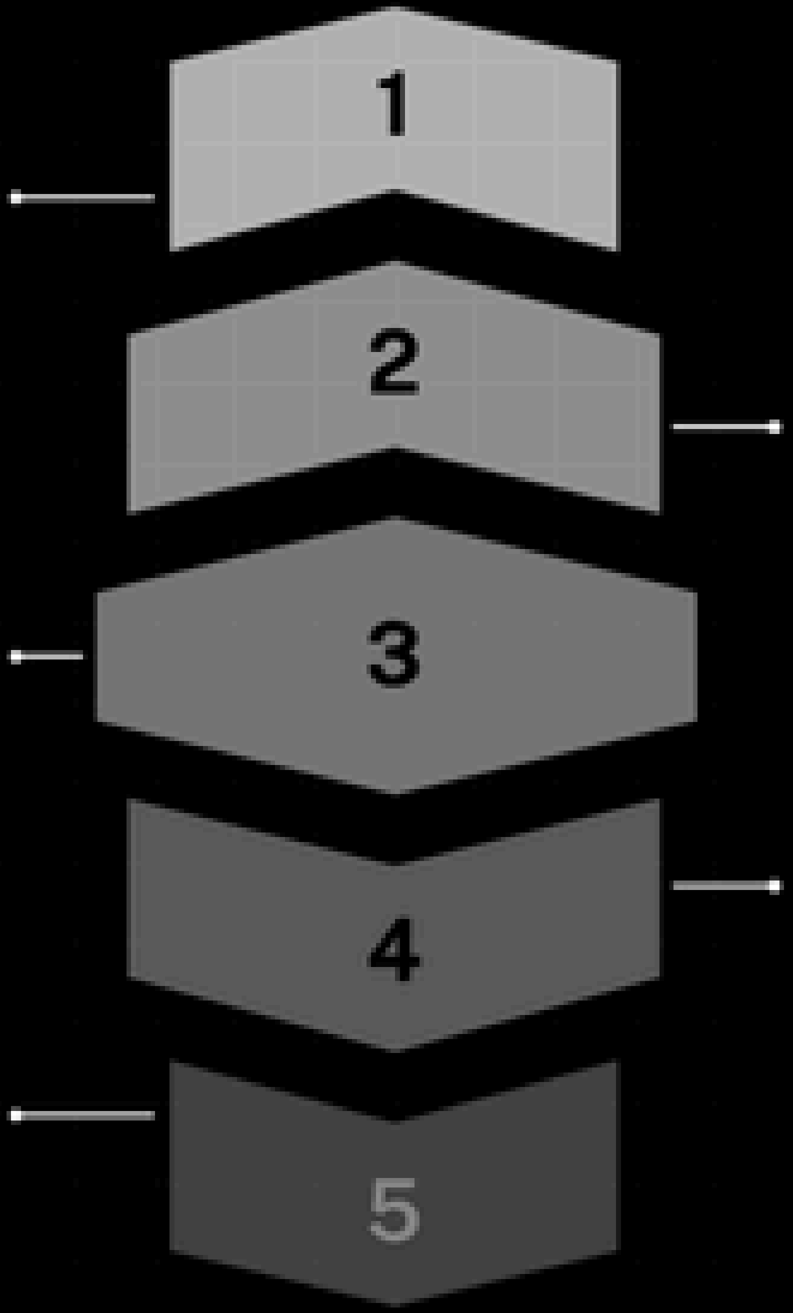
Non-functional Requirements that Shape Design

Performance, usability, reliability, portability and scalability implications

- **Performance:** OCR and translation complete within seconds on typical hardware; measure via processing time metrics
- **Usability:** single-command execution; clear, readable bounding boxes and labels for fast user comprehension
- **Reliability:** handle empty OCR results gracefully to avoid crashes and ensure stable runs
- **Portability:** designed for Windows but portable where PaddleOCR and dependencies are supported
- **Scalability:** plan region-based capture, continuous monitoring, multi-language and export format support
- **Design implication:** instrument processing times, validate OCR outputs, and keep CLI simple for adoption

System Architecture Overview — Modular OCR-to-Translation Pipeline

Capture desktop, extract English text with PaddleOCR (PP-OCRv5), translate to French, and overlay results for real-time display



Screen Capture

Uses PIL.ImageGrab to snapshot the desktop into a NumPy array for processing.

OCR Engine

Runs PaddleOCR with PP-OCRv5 models tuned for English; orientation correction disabled.

Translation

Helper script batch-translates recognized English text to French.

Visualization

Uses OpenCV to draw bounding boxes and render translated labels over the original image.

Main Controller

Orchestrates modules sequentially into an efficient, modular pipeline for easy maintenance.

Design Decisions and Rationale

Why choices were made and how they benefit the prototype

1. **PaddleOCR with PP-OCRv5** for balanced accuracy and low resource use; enables near real-time processing
2. **English OCR → French translation** to prove language-pair feasibility and future extensibility
3. **Full-screen capture** simplifies implementation and ensures complete visual data collection
4. **One-shot execution model** reduces user complexity for demonstrations
5. **Overlay visualization** provides instant feedback for OCR and translation without file exports
6. Result: **performance-efficient**, user-friendly, and adaptable prototype

Implementation Details — OCR + Translation Pipeline

Language, dependencies, main functions, and error-handling overview

Language: Implemented in **Python 3.10+**

Dependencies: paddleocr, pillow, numpy, opencv-python, translation lib (used in translate.py)

Core scripts: main.py initializes **PaddleOCR** with **PP-OCRv5** mobile models; translate.py handles batch translation

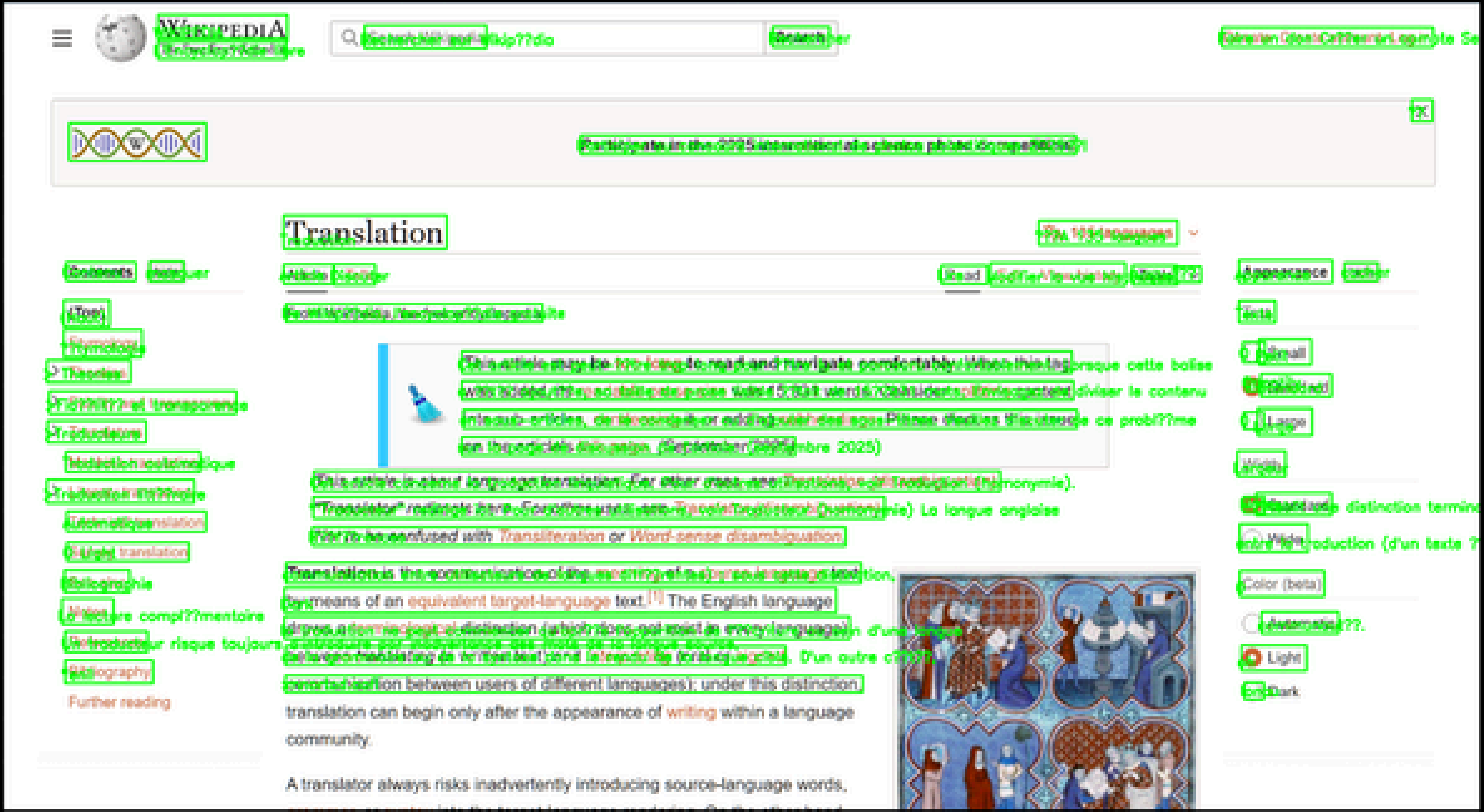
Processing flow: ImageGrab.grab() → convert image → OCR text & bounding boxes → batch translate → draw rectangles & translated labels

Display: Results shown in OpenCV GUI window; closed gracefully on key press

Error handling: Basic empty-OCR checks now; planned improvements: **confidence filtering** and **exception management**

Screenshots & Results Overview

Visual proof of OCR processing, translation overlay, and performance



This article may be too long to read and navigate comfortably. When this tag was added, its readable prose size was 15,701 words. Consider splitting content into sub-articles, condensing it, or adding subheadings. Please discuss this issue on the article's talk page. (September 2025)

lorsque cette balise … diviser le contenu … ce probl??me … mbre 2025)

This article is about language translation. For other uses, see Translation (disambiguation). … (homonymie).

"Translator" redirects here. For other uses, see Translator (disambiguation). … nie) La langue anglaise

Not to be confused with Transliteration or Word-sense disambiguation.

Translation is the communication of the meaning of a source-language text by means of an equivalent target-language text.[1] The English language draws a terminological distinction (which does not exist in every language) between translating (a written text) and interpreting (oral or signed communication between users of different languages); under this distinction, translation can begin only after the appearance of writing within a language community.

… d'une langue … D'un outre c??t??.

Le traducteur risque toujours d'introduire par inadvertance des mots de la langue source.

Lecture compl??mentaire

distinction termin… entre la traduction (d'un texte ?

A translator always risks inadvertently introducing source-language words,

Testing Approach: Verify OCR accuracy, visuals, and performance

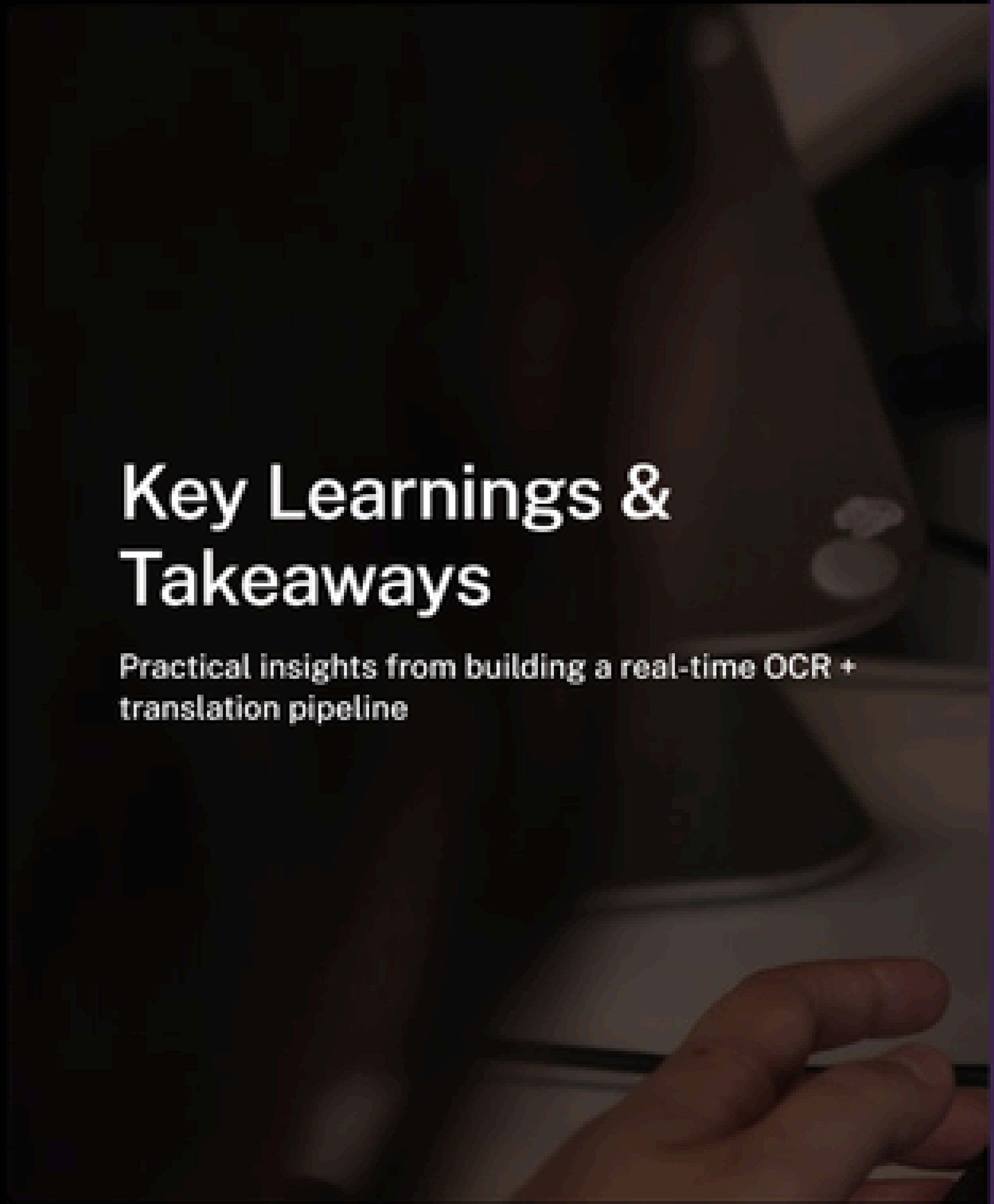
Manual runs plus edge and performance tests to validate functional accuracy, visual clarity, and responsiveness

Manual testing across documents, websites, and code to confirm detected text regions align with visible text and French translations are correctly placed and reasonable

Edge testing targeting small fonts, low-contrast text, and screens without text to assess OCR robustness

Performance testing measuring average OCR processing across runs and comparing CPU vs optional GPU acceleration

Goal: validate **functional accuracy**, **visual clarity**, and **system responsiveness** to form a formal test plan and case documentation



Key Learnings & Takeaways

Practical insights from building a real-time OCR + translation pipeline

Hands-on OCR: practical experience with OCR pipelines and **PaddleOCR**

Image tooling: used Pillow, OpenCV, NumPy for screen capture and processing

NLP integration: translation layered on vision outputs, exposing engineering trade-offs

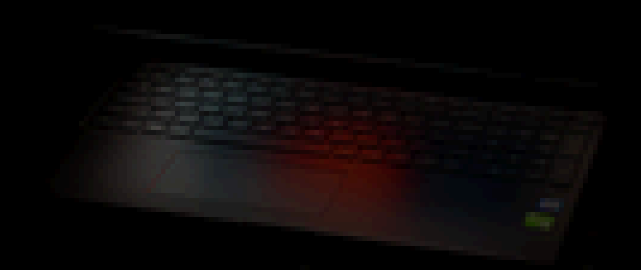
Real-time tradeoffs: balanced **speed**, **accuracy**, and **user experience**

Debugging & tuning: annotated overlays improved visibility and system reliability

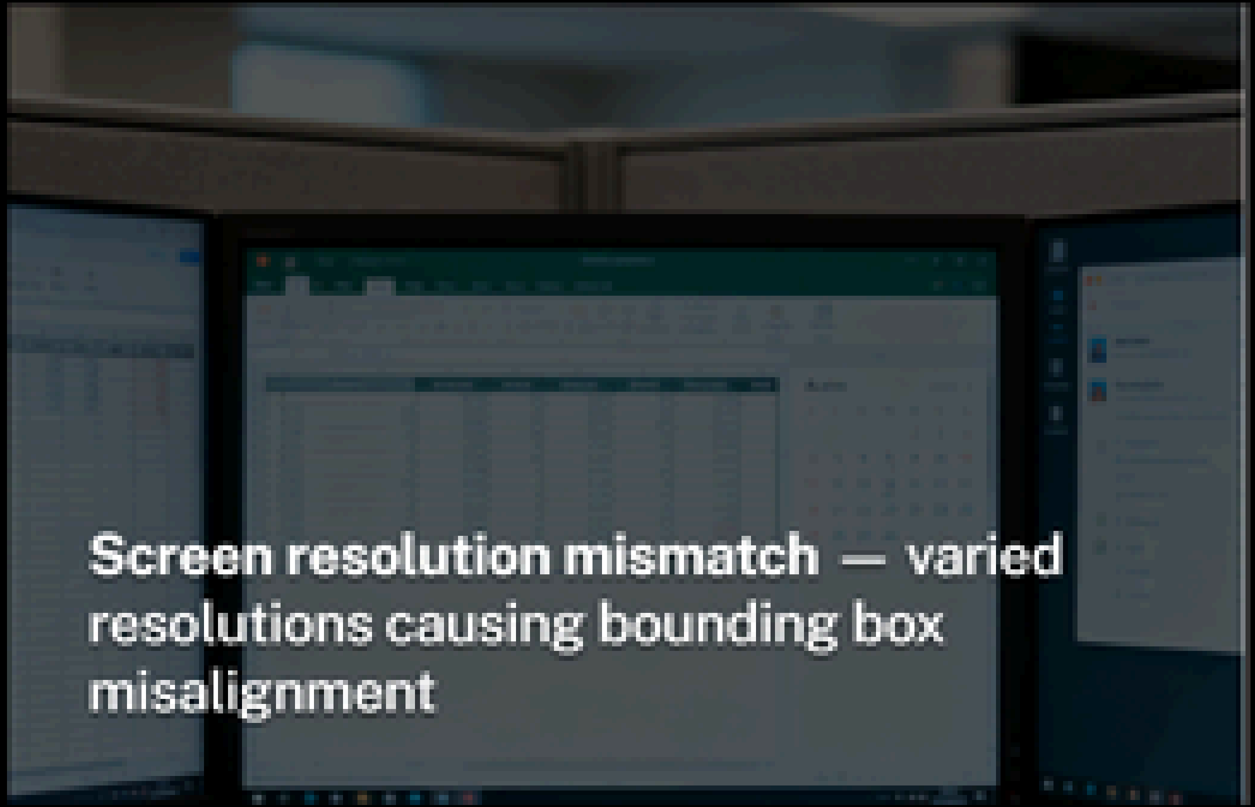
Outcome: sharper debugging, informed tuning, and improved reliability

Deployment Challenges for OCR & Overlay

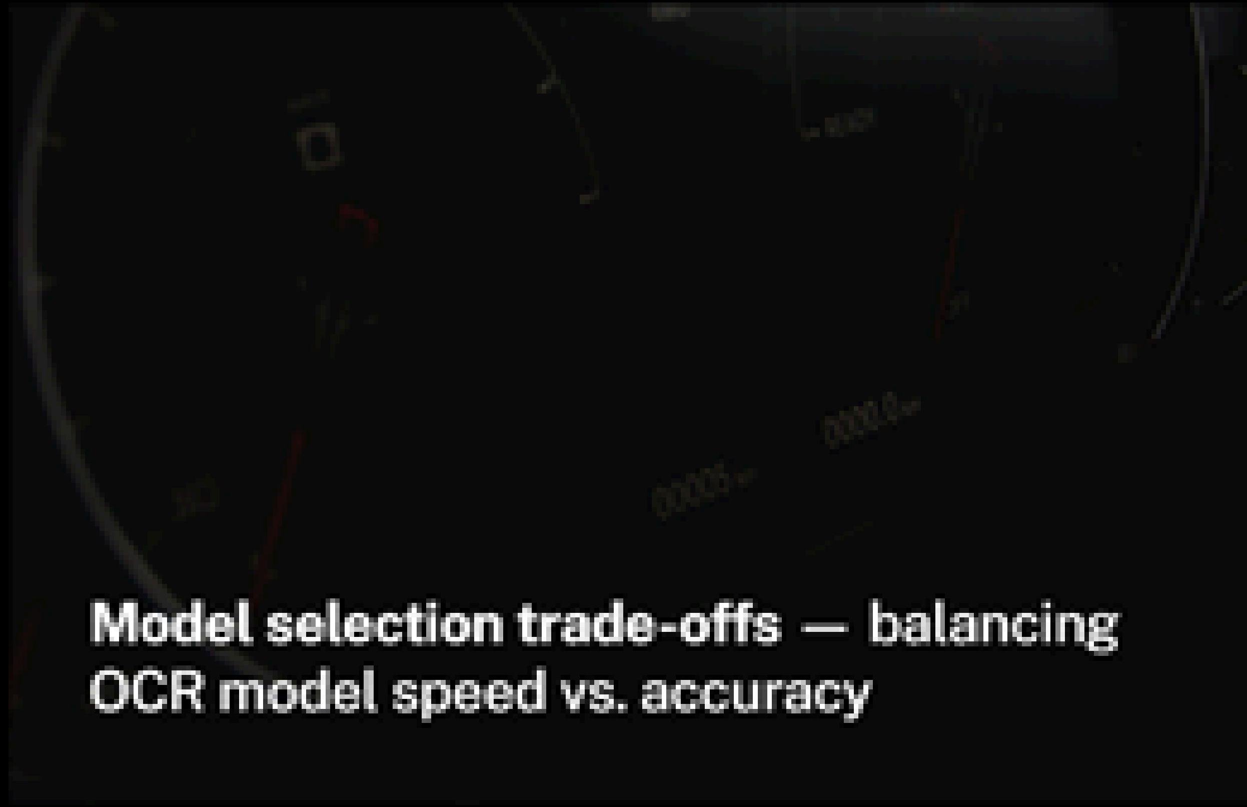
Technical and usability obstacles encountered during PaddleOCR integration and UI overlays



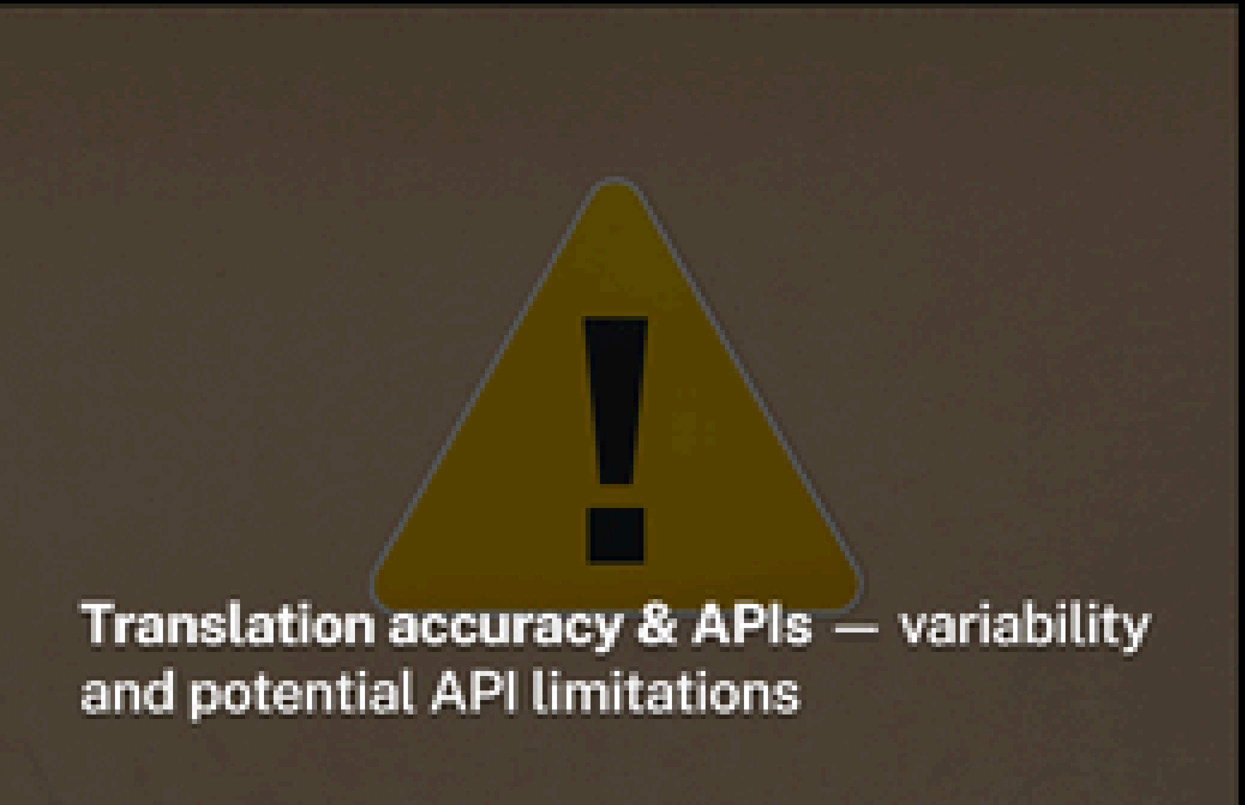
PaddleOCR on Windows & GPU — complex dependency setup and enabling GPU acceleration



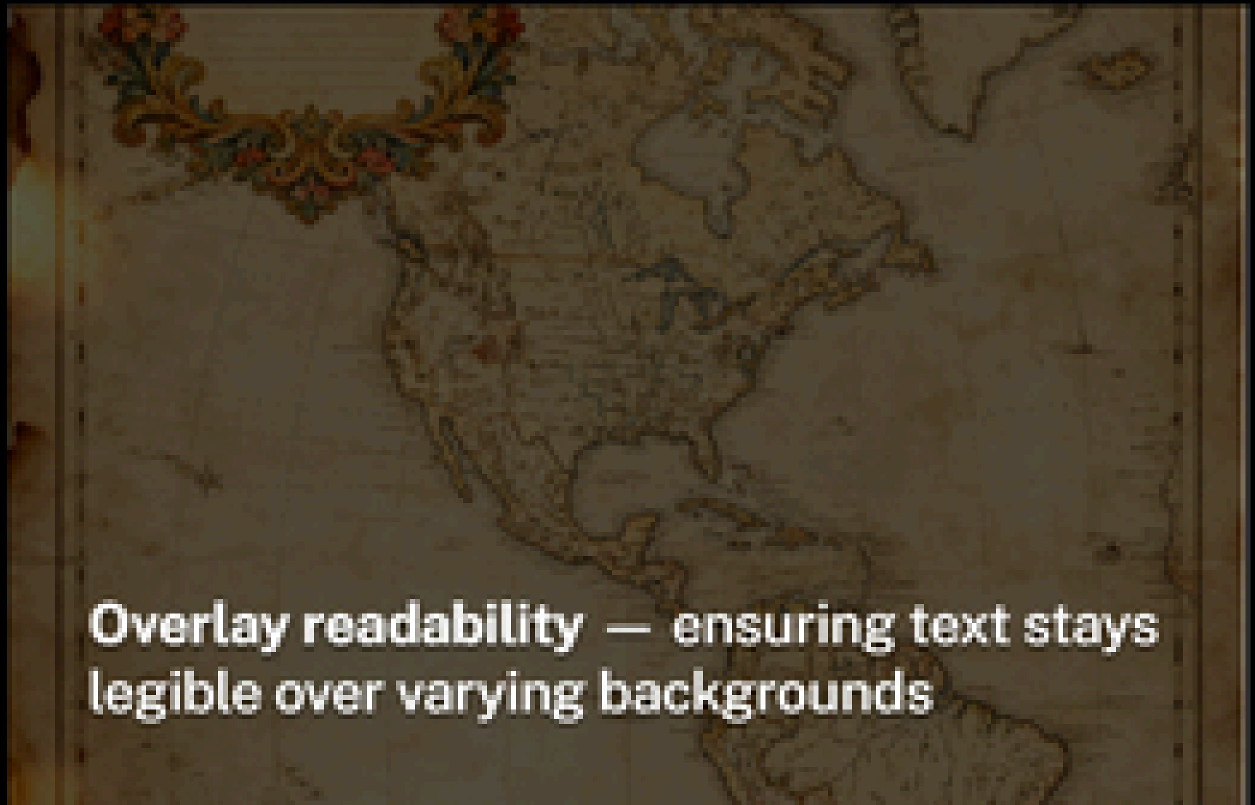
Screen resolution mismatch — varied resolutions causing bounding box misalignment



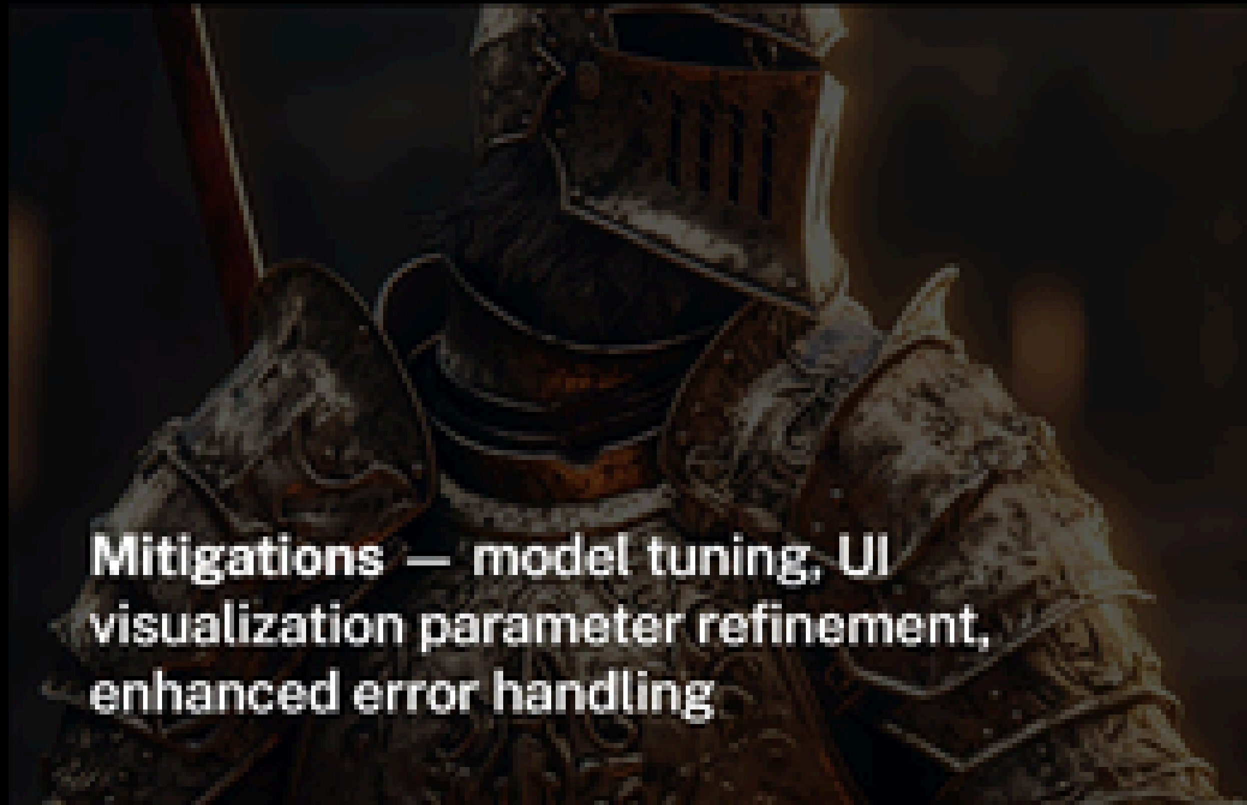
Model selection trade-offs — balancing OCR model speed vs. accuracy



Translation accuracy & APIs — variability and potential API limitations



Overlay readability — ensuring text stays legible over varying backgrounds



Mitigations — model tuning, UI visualization parameter refinement, enhanced error handling

Roadmap: Next Enhancements to Boost Accuracy & Usability

Planned features to automate capture, improve OCR/translation quality, and simplify export & audit

Add **region-specific / window-focused** screen capture to increase efficiency

Implement **continuous monitoring** with hotkey triggers for automated repeated captures

Support **multiple OCR & translation languages** with dynamic selection

Introduce **confidence thresholds** to filter low-quality OCR results

Export results to **structured formats (JSON, CSV)** and enable clipboard copy

Integrate a **GUI** and persistent storage to log captured texts and translations for audit trails

Project References

Key documentation and sources that supported development

1 **PaddleOCR GitHub Repository** — https://github.com/PaddlePaddle/PaddleOCR

2 **OpenCV Documentation** — https://docs.opencv.org/

3 **Pillow (PIL) Documentation** — https://pillow.readthedocs.io/

4 **NumPy Documentation** — https://numpy.org/doc/

5 **Translation API/library docs used in translate.py** — documentation or links for the chosen translator

6 **Course materials, research papers & technical resources** — supporting project development