Real-Time On-Device OCR with Confidence Filtering

Liviu Staniloiu
grigore.staniloi02@e-uvt.ro

***Abstract*—We present a fully on-device real-time optical character recognition (OCR) system that leverages OpenCV and Tesseract, augmented by confidence filtering to surface only high-quality text detections (confidence 75%). The pipeline captures frames, optionally converts to grayscale, performs OCR, filters by confidence, overlays results on the feed, and logs recognized text. We evaluated performance on a Raspberry Pi 4 under varied lighting and motion conditions, reporting latency, throughput, precision, recall, and memory usage.**

I. Introduction

On-device OCR is key for privacy-sensitive, offline applications such as assistive reading, inventory tracking, and expiration-date verification. Cloud-based offerings achieve high accuracy but introduce latency and data exposure [**?**]. Tesseract, an open-source OCR engine, can run locally but often produces spurious detections without additional filtering. In this work, we develop a lightweight pipeline that maintains real-time performance on constrained hardware and filters results based on Tesseract's confidence scores, significantly reducing false positives.

We demonstrate a single-script implementation that: First, captures camera frames at 640×480 resolution and converts them to grayscale to reduce noise. Next, it invokes `pytesseract.image_to_data` with the LSTM engine (`--oem 3`) and single-line mode (`--psm 6`). By applying a confidence threshold of 75%, the system discards low-quality words, enhancing overall reliability. Finally, the high-confidence text is overlaid on the live feed and appended to a timestamped log. We benchmark this pipeline on a Raspberry Pi 4, measuring per-frame latency, effective frames per second, word-level precision and recall against hand-annotated ground truth, and peak memory usage.

II. Related Work

Cloud OCR services such as Google Vision and AWS Recognition provide robust text detection but at the cost of network dependency and potential privacy risks [**?**]. On-device engines such as Tesseract have evolved with LSTM networks for improved accuracy [1], yet they lack built-in postfiltering, often necessitating complex preprocessing or neural scene text detectors (e.g., EAST [2], CRAFT [3]). These methods deliver high recall in challenging environments,, but require morecomputation.tion. Our approach opts for simplicity and efficiency by combining minimal preprocessing with confidence-based filtering.

III. System Design

The pipeline consists of the following steps, executed sequentially:

1) Frame acquisition via VideoCapture.
2) Optional grayscale conversion to reduce image noise.
3) OCR using `pytesseract.image_to_data` (LSTM engine, single-line mode).
4) Discarding text boxes with confidence below 75%.
5) Overlaying high-confidence words on the video frames.
6) Logging recognized text with timestamps to `ocr_output.log`.

Rather than a complex diagram, the core flow can be summarized as:

$$\text{Capture} \rightarrow \text{Preprocess} \rightarrow \text{OCR} \rightarrow \text{Filter} \rightarrow \text{Overlay} \rightarrow \text{Log}$$

This linear representation emphasizes the simplicity and modularity of the design.

For preprocessing, we found that grayscale conversion alone suffices at 640×480; adding Gaussian blur or binarization did not improve accuracy enough to justify added latency on our test hardware.

IV. Implementation

Our implementation (`pipeline.py`) relies on OpenCV for frame capture and display, and on `pytesseract` for OCR. The confidence threshold is parameterized, allowing for adaptation to different scenarios. The log file follows a CSV-like format: each line contains a timestamp and the concatenated high-confidence words detected in that frame.

Key configuration parameters include the camera source index, frame rate target, grayscale toggle, Tesseract configuration string, and confidence threshold. The script handles keyboard interruption and releases resources cleanly.

V. Experimental Evaluation

We evaluated on a Raspberry Pi 4 Model B (1.5 GHz quad core, 4 GB RAM) with a USB camera at 640×480 resolution. Test scenarios included static printed text, moving labels, and low-light environments.

Measurements show that under static conditions, the pipeline achieves an average latency of 170 ms per frame (5.9 FPS) with precision 91% and recall 85%. In moderate motion, the latency increased slightly to 185 ms (5.4 FPS), giving 88% precision and 79% recall. Under low light, performance degraded to 210 ms latency (4.8 FPS), 83% precision, and 75% recall. Memory usage remained below 250 MB throughout.

VI. Discussion

Our results indicate that confidence filtering effectively balances precision and recall, removing many false positives while preserving real-time performance. Although higher thresholds improve precision, they reduce recall; application requirements should guide threshold selection. The minimal preprocessing pipeline keeps latency low, but advanced image enhancement could benefit extreme lighting situations.

VII. Future Work

Future enhancements might include adaptive thresholding that adjusts based on recent frame confidence distributions, integration of lightweight NLP parsers for structured fields (e.g., dates), and optional on-device language model correction for critical use cases. Furthermore, selective region-of-interest cropping could focus computation on areas likely to contain text, improving throughput.

VIII. Conclusion

We have presented a concise, portable OCR pipeline that runs entirely on a device, filters outputs by confidence to ensure reliability, and achieves near-real-time performance on modest hardware. This approach offers a practical foundation for privacy-sensitive applications requiring live text recognition.

References

[1] R. Smith, "An overview of the Tesseract OCR engine," in *Proc. ICDAR*, 2007, pp. 629–633.

[2] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Li, "EAST: An efficient and accurate scene text detector," in *Proc. CVPR*, 2017, pp. 5551–5560.

[3] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proc. CVPR*, 2019, pp. 9365–9374.

[4] T.-R. Grumeza, B. Bozga, and G. L. Stăniloiu, "Integration of LLM in Expiration Date Scanning for Visually Impaired People," *Scalable Computing: Practice and Experience*, vol. 25, Oct. 2024, doi:10.12694/scpe.v25i6.4967.