Marc Kendal

Imaging & Computer-Vision Engineer — Python · C Interop · Embedded Real-Time Systems London, UK
marckendal@gmail.com · +44 7939 130 443 marckendal.com · github.com/MKToronto

---

Summary

Python engineer with five years inside a deployed industrial imaging product — owned the **computer-vision and image-processing layer of Fortress Technology's flagship ICON X-ray inspection machine**, shipped at scale across Fortress's installed base. The production architecture transfers cleanly into adjacent imaging domains (satellite, medical, industrial inspection): C detector code writes into SysV shared memory; Python reads via `sysv_ipc` with dtype-driven byte offsets programmatically derived from the C struct layout; **numba-JIT hot paths** process the image stream on Raspberry Pi edge hardware under real-time constraints; atomic `hardlink_to` → `.replace()` file publishing so downstream consumers never see half-written frames; nine asyncio locks coordinate disjoint state regions with a ThreadPoolExecutor / asyncio bridge bringing SHM polling into the async server. Since May 2025, solo-building an algorithmic trading platform (four broker APIs). Works spec-first with AI coding agents — plans before prompts, reviews every change before commit. **Every piece of code shipped at Fortress went out with zero post-ship bugs and no customer complaints.**

Experience

Research Engineer — Fortress Technology (Toronto) · Oct 2019 – Aug 2024

Owned the Python computer-vision and data-ingestion layers across Fortress's industrial-inspection product line. Flagship ICON X-ray inspection machine catches bone, metal, ceramic, and glass contaminants on food-production lines before packaged product reaches consumers. Deployed at scale across Fortress's installed base. **Zero post-ship bugs and no customer complaints across shipped work.**

- **Owned the real-time image-processing pipeline** on Raspberry Pi edge hardware, alongside legacy C / OpenGL / Perl already on the machine. *(product demo)*
  - **C ↔ Python interop via SysV shared memory**, with `sysv_ipc` reading and byte offsets computed programmatically from a numpy dtype mirror of the C struct; `extract_constants.py` parses the C header (`SparcXray_CP/common01.h`) for `#define` constants and caches to `constants.json` , keeping the interop contract from drifting silently when the C side evolves.
  - **numba-JIT hot paths** (`@jit(nopython=True, parallel=True, fastmath=True)` with `prange`) for the image-processing work — pre-allocated coordinate arrays, single-pass bounding-box computation, parallelised rendering.
  - **Adaptive geometry**: brush radius scaled by `sqrt(skip_factor)` under performance pressure to keep coverage consistent; recursive hole-filling with configurable stop thresholds trading aesthetic perfection against performance budget.
  - **Atomic file publishing** via `hardlink_to` → `.replace()` so the UI never sees a half-written image.
  - **Nine asyncio locks** coordinating disjoint regions of shared state (recent / past image state, binary data, batches, JSON, memory view), with a ThreadPoolExecutor / asyncio bridge bringing SHM polling into the async server.
  - **Telemetry-by-correlation** (`stats_on_production.py` uses Pearson coefficients between render time and mask characteristics) finds real bottlenecks empirically rather

than guessing; per-run metrics written to `production_speed_test.csv` for fielded-system observability.

- **Led the on-device contaminant-detection modelling work** (bone and metal classification via FastAI + PyTorch). Wrote the coordinate-auto-generation function that made object-detection training viable on production-shaped data; integrated the trained models into the on-device pipeline; this directly influenced the hardware design of the next generation of Fortress machines.
- **Made the strategic call to invest in labelled-data acquisition** at R&D-leadership level, then designed and built `image_dev_tool_fastapi` — a FastAPI + Svelte + Redis-queue data-ingestion tool that pulled time-windowed binary image batches from fielded machines over SSH (paramiko with TCP keep-alive for long sessions, SCP recursive pull, auto-save flag toggling), queued batch conversion through the C pipeline, and gave operators a Svelte SPA (Home / Create / Search / Settings / Evaluate) to categorise contaminants. That tool built the dataset every downstream anomaly-detection model relied on. Collection IDs generated deterministically by pattern `{DN}{Episode}{RandomLetters}{Date}{Product}`.
- **Built the SvelteKit operator UI** running on every shipped ICON X-ray machine (`xray_streaming`) — kiosk-mode Chromium served via `sirv` on Pi, with a full offline-deployable installation pipeline (mamba env + screen sessions + boot-time startup + field-installable packaging). Iterated through seven streaming architectures (aiortc / WebRTC / HLS / gstreamer / various queued-Python designs) before landing on the kiosk design. *(Fortress ICON product page)*
- **Founding engineer on the Knapp pill-dispensing product** — Flask control system shipped through external-customer handoff on a first Fortress project with zero post-ship bugs, including user-editable calibration for precise dispensing.
- **Owned the multilingual HMI translations pipeline (`Translations_v3`)** — bidirectional JSON ↔ Excel tool synchronising 40+ languages across every Fortress machine.
- **Mentored junior engineers** on the React + Python auth stack through pair programming and code review; 150+ bugs closed together with measurable skill growth.

Founding Engineer — Automated Trading Platform (independent R&D) · May 2025 – Present

Sole architect and engineer of a production-grade algorithmic trading platform in Python / asyncio — ~200 modules covering execution across four broker APIs, risk management, and strategy iteration. Live against real capital. *(Following paternity leave, Sep 2024 – April 2025.)* Architected alongside **two Claude Code plugins** — one published (`python-clean-architecture`, on the Claude Code marketplace + Codex CLI) and one private (`quant-finance`, operationalising Marcos López de Prado's three quant-ML books for personal trading research).

Software Engineer — British Airways (London, UK) · 2014 – 2016

- Built live-data interception code for ba.com in Python (aspect-oriented programming / AOP) — captured real production data streams so tests ran against realistic live inputs instead of synthetic stubs.
- Built Selenium / Cucumber / Concordion BDD suites for desktop and mobile.
- Co-authored a "date-rolling" Python service that kept stubbed test data valid as calendar dates advanced — adopted across all teams.

Selected Projects

- **`xray_graphics` production imaging pipeline** — SysV shared memory + numba JIT + adaptive geometry + atomic file publishing + 9-lock async coordination on Raspberry Pi. Inside Fortress's flagship ICON X-ray machine, deployed at scale.

- `image_dev_tool_fastapi` — FastAPI + Svelte + Redis-queue data-ingestion + human-labelling tool. SSH-orchestrated pull from 100+ fielded X-ray machines, queued conversion through the C pipeline, operator UI for contaminant categorisation.
- `xray_streaming` — SvelteKit operator UI + Python orchestration + offline-deployable kiosk installation pipeline on every shipped ICON X-ray.
- `python-clean-architecture` — Published Claude Code marketplace plugin (+ Codex CLI). 12 commands + skill pack for clean Python / FastAPI architecture.
- `localytics` — Published self-hosted alternative to GitHub Insights. Local FastAPI scanner (Radon function-level cyclomatic complexity + per-file churn + weekly/monthly/yearly heatmaps) pushes only aggregated metrics to a Render-hosted FastAPI + Redis dashboard; source stays local. MIT-licensed; live demo.
- **Automated trading platform** — ~200 Python / asyncio modules; four-broker execution; live.
- **MSc Data Science dissertation — Persian OCR** (2019) — PyTorch CNN for Middle-Eastern-language OCR, convolution analysis, classifier interpretation. Earlier CV work.
- **Shiurim transcription pipeline** — Whisper + SpeechBrain speaker ID + Claude Code classification over a large audio corpus.

Tech

Primary: Python · C (ctypes + shared memory) · JavaScript / Svelte / SvelteKit **Imaging / CV:** NumPy · OpenCV · numba (JIT for image hot paths) · PyTorch / FastAI · scikit-image · `sysv_ipc` (shared memory interop) · `extract_constants.py` (C-header → JSON bridge) **Systems / Infra:** FastAPI · asyncio · Docker · SQLite · Redis / RQ · WebSockets · systemd · mamba / conda · Raspberry Pi deployment · paramiko / SCP · memray profiling · kiosk Chromium + `sirv` **Working practice:** Agile · CI/CD · code review · telemetry-driven optimisation (Pearson-correlation analysis on render-time metrics) · production-observability for fielded hardware · spec-driven, agent-orchestrated development · review every change before commit · Claude Code + Codex CLI day-to-day

---

Education

MSc Data Science · City, University of London · 2016 – 2019 Dissertation (Computer Vision): *Improving the Optical Character Recognition of Middle Eastern Languages* — PyTorch CNNs, convolution analysis and heat-maps, classifier interpretation.

PGDip Philosophy · The Heiden Institute

BSc (Hons) Economics, 2:1 · University of Nottingham · 2005 – 2008
