Portfolio

AI application & Software Development

Tianlang Liu

Vision Algorithm Engineer | CATL, Germany

📍 Germany | 💼 Open to relocation

🔗 linkedin.com/in/tianlang-liu

✉️ tlliu_liesmars@outlook.com

Optical Spot Detection Software

Tech Stack: Qt, OpenCV, Python, Multithreading, YOLO, TensorRT

Problem:

Existing inspection processes relied on **offline tools** for optical defect detection, with significant **delays** in defect logging and limited user interactivity.

Solution:

Built a **multi-threaded real-time** GUI application in Qt integrated with OpenCV and YOLO. Enabled **live defect detection**, dynamic **parameter tuning**, and immediate **MES** integration for anomaly upload.

Results:

Reduced inspection delay **from day to product, saving 20+ hours for 20 machines.** Maintaining a **false positive** rate of **<0.2% over 200,000+ samples**.

My Contribution:

Led **full-cycle development** from UI design to backend threading. Integrated detection, user login management, parameter modules, and MES communication.

Maintenance of AI-based inspection software in Multithreading

Tech Stack: **Visual Studio, NumPy**

Problem:

Frequent crashes in **multithreading** caused **downtime** in the inspection software for **60+ devices**, impacting production.

Solution:

- **Numpy iterates all logs** to trace back the running situation before crash, e.g., in which function the crash happened.
- Used **Visual Studio Debugging Tools** for debug in **Cross-thread invocation** and and refactored the code for better synchronization and memory management.

Results:

- Saved **10h+ weekly production time** and enhanced system stability across 60+ devices.

My Contribution:

- Resolved C++ multithreading crash using **NumPy** and **Visual Studio Debugging Tools**.

Classification Result Evaluation Assistant (GUI Tool)

Tech Stack: PyQt, Python, Pandas

Problem:

Manual evaluation of classification outputs (e.g., FP/FN tagging) was **inefficient**, **error-prone**, and **lacked standardization**, especially across batches of hundreds of samples.

Solution:

Built a GUI desktop tool for efficient model evaluation with **keyboard-based tagging**, **undo operations**, and **real-time feedback**. Supported TP/FN/FP/TN tagging and automatic **export** of performance reports.

Results:

Accelerated validation process **by 85%** in model tuning cycles and enabled consistent human verification across multiple teams.

My Contribution:

Designed UI/UX flow, implemented batch logic and keyboard handlers, and deployed the tool internally for use during model iteration cycles.

Real-Time Defect Diagnosis Pipeline base on Client-Server Software

Tech Stack: PyQt, Python, Pandas, Numpy, Matplotlib

Problem:

Manual checking of defect logs **caused long delays in diagnosis** (up to 24 hours), resulting in **production bottlenecks** and unstable feedback loops.

Solution:

Built a modular **real-time defect diagnosis pipeline with a client-server** architecture. UI clients were deployed on 20+ machines for on-site visualization, while the backend automatically **parsed** shared machine outputs over LAN to extract NG results, **time cost, and heartbeat patterns, enabling continuous monitoring and early warning.**

Results:

Reduced diagnosis time by 90% (from 24h to 2h), significantly improving feedback speed and workflow stability on the production line.

My Contribution:

Designed the **full system architecture**; developed the PyQt-based client UI and Python backend parser; configured LAN-based communication and deployment across multiple devices.