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

Surface Defect Detection with 2D + 3D Fusion

Tech Stack: Mask RCNN, PCL, Python, OpenCV

Problem:

Surface **holes** were often **missed by standard 2D Mask RCNN detection**, leading to false negatives and quality concerns.

Solution:

Developed a detection algorithm based on integration of **point cloud** and 2D images using Mask RCNN, checking suspected areas with **2D features, 3D features and registered areas depth**.

Results:

Reduced manual inspectors from 12 to 0; achieved **0% missed defects and <0.4% false alarms** across 200k+ products.

My Contribution:

Independently handled **data preprocessing**, model design and training, **PCL-based postprocessing**, and deployment scripting.

Electrode Orientation Filtering using YOLO + DBSCAN

Tech Stack: YOLO, OpenCV, Scikit-learn (DBSCAN), Python

Problem:

Electrodes **misaligned** in orientation were difficult to detect **using angle thresholds alone**, causing **defects to slip through** automated QA filters.

Solution:

Used **YOLO** to detect electrode **masks** and evaluated their alignment via overlap with its ROI. Applied DBSCAN clustering on overlap scores to **dynamically determine threshold** boundaries and filter out outliers.

Results:

Enabled reliable detection of orientation anomalies under varying production configurations. **Reduced human checking workload** and stabilized process quality.

My Contribution:

Modified the orientation filtering algorithm, tuned **DBSCAN parameters** for dynamic thresholding, and implemented the full postprocessing logic in Python.

Optical Spot Detection System (Real-Time GUI)

Tech Stack: PyQt, OpenCV, Python, Multithreading

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 PyQt integrated with OpenCV processing. Enabled **live defect detection**, dynamic **parameter tuning**, and immediate **MES** integration for anomaly upload.

Results:

Reduced end-to-end inspection delay **from hours to seconds**. The tool processed 200+ images within 6 seconds, maintaining a **false positive** rate of **<0.6% 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.

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 3×** 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.