Project Plan:

Automated Optical Inspection (AOI) System Prototype

Project Lead	Gregorius Karisma
Role	Technical Project Lead
Start Date	2025-11-01
Domain	Semiconductor Metrology & Yield Analysis

Background

This project involves the design, fabrication, and programming of a desktop Automated Optical Inspection (AOI) system. The initiative simulates industrial semiconductor metrology processes by utilizing computer vision to identify surface defects on printed circuit boards (PCBs). This system integrates hardware engineering (optical rig construction), software engineering (Python/OpenCV), and data analytics (SQL/Tableau) to generate actionable yield data. The primary goal is to replicate the defect detection and yield improvement workflows utilized in high-volume semiconductor manufacturing.

Project Objectives

1. Engineering → construct a stable optical rig capable of capturing consistent reference ("Golden Sample") and test images.
2. Software → develop a Python algorithm utilizing OpenCV to detect surface anomalies with greater than 90% accuracy.
3. Analytics → establish a data pipeline to log inspection results into a SQL database for automated yield calculation.
4. Application → demonstrate the practical application of precision engineering and data analysis within a manufacturing industry.

Technical Architecture & Stack

1. Hardware Components:
    a. Imaging sensor → high-resolution webcam or smartphone camera linked via USB/Wi-Fi.

b. Lighting → LED ring light to ensure uniform illumination and eliminate shadow artifacts.
c. Fixture → rigid stand or mount (3D printed or aluminum extrusion) to maintain a fixed focal distance.
d. Device under test (DUT) → Arduino Uno Board or custom PCB serving as the inspection subject.

2. Software Stack:
    a. Language → Python 3.9+
    b. Libraries:
        i. Opencv-python → image processing, subtraction, and contour detection.
        ii. Numpy → array manipulation.
        iii. sqlite3 / pandas → data logging and storage.
    c. Visualization → Tableau (Yield Dashboards).

Implementation Roadmap

1. Hardware Integration & Calibration:
    a. In this phase, the goal is to establish a controlled imaging environment.
        i. Assemble the rig and secure the imaging sensor to prevent micro-vibrations.
        ii. Calibrate lighting to prevent surface glare on the DUT.
        iii. Capture the "Golden Sample" (Reference) image for baseline comparison.
2. Algorithm Development (Python):
    a. In this phase, the goal is to detect manual defects (e.g., scratches, foreign objects).
        i. Implement image registration to align the Test image with the Reference image.
        ii. Code Image Subtraction logic: $Difference = |Reference - Test|$.
        iii. Apply thresholding and contour detection to isolate significant defects.
3. Data Pipeline Construction (SQL):
    a. In this phase, the goal is to transition from detection to analytics.
        i. Design SQL Schema containing:
            1. Inspection_ID ← primary key
            2. Timestamp
            3. Defect_Count ← integer
            4. Status ← PASS/FAIL
        ii. Develop a Python script to append inspection results to the database automatically.

4. Yield Analysis & Visualization:
    a. In this phase, the goal is to generate high-level yield metrics.
        i. Connect Tableau to the SQL dataset.
        ii. Visualize Yield Rate (Pass vs. Fail percentage) and Defect Severity (Pareto Charts).

Methodology: Defect Detection Logic

The system utilizes the Golden Sample Comparison method, a standard in fabrication manufacturing:

1. Image acquisition → capture frame of the DUT.
2. Preprocessing → convert to grayscale and apply Gaussian Blur for noise reduction.
3. Alignment → perform feature matching (ORB/SIFT) to align the DUT with the Reference.
4. Subtraction → execute pixel-wise subtraction to isolate differences.
5. Filtering → apply morphological operations to remove false positives (e.g., dust).
6. Decision → Mark as FAIL if *Defect_Area* exceeds the defined threshold.

Risk Management

1. Lighting inconsistency → false positives (shadows detected as defects) → utilize an enclosed lighting box or fixed ring light; implement image normalization.
2. Camera misalignment → failure of subtraction algorithm → implement robust image registration (software alignment) prior to subtraction.
3. Processing latency → extended inspection time per unit → resize high-resolution images to standard resolution prior to processing.

Expected Deliverables

1. Physical prototype → a functional desktop inspection station.
2. Code repository → documented Python code.
3. Yield dashboard → Tableau visualization displaying inspection statistics.
4. Technical report → documentation correlating defect size with detection accuracy.