Milestone-1 Dataset Preparation & Image Processing

Duration: Week 1 – Week 2

---

1. Objective of Milestone-1

The primary objective of Milestone-1 was to carry out the initial and crucial stage of the project — preparing the dataset in a way that enables accurate learning and defect classification in the upcoming deep-learning model. Printed Circuit Board (PCB) defect classification involves identifying *which specific type of defect is present* in a PCB image. The project focuses on six internationally recognized PCB defect classes: **Missing_hole, Mouse_bite, Open_circuit, Short, Spur, and Spurious_copper**. Since raw PCB images contain a large number of non-defective regions, direct model training is inefficient. The learning process becomes effective only when defect-specific regions are isolated and labelled.

This milestone focused on two major components:

(1) Dataset Setup & Image Subtraction

(2) Contour Detection & ROI Extraction.

The first component involved aligning the template (defect- independent) and second one (defect-present) images and using image subtraction to highlight variations caused by defects. This step ensured that any fault locations were made visually prominent and observable. The second component concentrated on identifying the exact spatial regions containing defects using contour detection, after which the defect regions were cropped and stored as Region of Interest (ROI) patches for deep learning. The completion of this milestone forms the foundation for accurate classification in the next stage, where a YOLO-based model will be trained.

---

2. Tasks Completed

The tasks performed during this milestone were sequential and logically interconnected to accomplish defect localization from PCB images. A summary of all completed tasks is provided below:

- Downloaded and inspected the PCB dataset containing template and defect-present image pairs with annotations in xml.
- Verified dataset integrity and standardized file naming for easier automation and processing.

- Ensured correct image alignment between template and test images through feature matching.
- Performed pixel-wise subtraction to produce defect difference maps, highlighting variations between the two.
- Applied Otsu's thresholding technique to convert grayscale difference maps into binary defect masks.
- Performed noise reduction using Gaussian Blur and morphological operations to achieve smooth defect boundaries.
- Detected contours from the filtered binary masks to identify defect boundaries.
- Extracted bounding boxes around contours and cropped individual Region of Interest (ROI) defect patches.
- Labeled the cropped patches to prepare a well-structured dataset for model training.

These tasks ensure that the dataset is both clean and representative, enabling the machine learning model to focus specifically on defect-relevant information.

---

3. Tools / Libraries Used

To streamline the automation and maximize accuracy of image preprocessing and defect isolation, the following tools and libraries were employed:

- **Python:** Primary programming language used for the entire pipeline.
- **OpenCV:** Utilized for image processing tasks such as alignment, subtraction, thresholding, contour detection, and ROI extraction.
- **Keras:** Used for understanding dataset structuring and for future model training inspirations.
- **TensorFlow:** Integrated for compatibility with deep learning components and for future training phases.
- **Matplotlib:** Used for visualizing intermediate outputs such as masks, contours, and ROIs.
- **DeepPCB Dataset:** The foundational dataset for test–template image pair-based PCB defect detection.

The combination of these tools provided both flexibility and high performance during processing.

---

4. Process Summary (Step-wise)

The defect extraction pipeline followed during Milestone-1 is described in a step-by-step manner below:

Step 1: Loading Dataset

The PCB dataset was downloaded and loaded into the working directory. Image inspection was performed to understand naming conventions, resolution consistency, and corresponding test–template image pairs.

Step 2: Image Alignment

Since even a slight misalignment between template and test images can lead to false subtraction results, keypoint feature matching followed by homography-based alignment was applied to ensure pixel-accurate positioning.

Step 3: Image Subtraction

Absolute pixel-level subtraction was performed (`abs(template – test)`) to highlight deviations caused by potential defects. This generated grayscale defect difference maps where brighter regions represent anomalies.

Step 4: Thresholding and Noise Reduction

Otsu's automatic thresholding technique converted difference maps into sharply defined binary masks. Gaussian blur and morphological operations were then used to smooth and refine the regions for more consistent contour detection.

Step 5: Contour Detection

With a noise-free binary mask, `cv2.findContours()` was applied to identify defect boundaries. Contours were filtered based on their area to avoid extremely small artifacts being wrongly considered as defects.

Step 6: ROI Extraction and Storage

Bounding boxes were drawn around significant contours. Each bounding box was cropped to extract defect region patches. These ROI images were then resized and saved in an organized directory format that supports deep learning workflows.

This pipeline ensures clean, well-segmented, and defect-specific ROI samples corresponding to the six PCB defect classes: Missing hole, Mouse bite, Open circuit, Short, Spur, and Spurious Copper enabling the deep learning model to learn classification patterns accurately in the next milestone.

5. Challenges & How I Solved Them

Although the milestone execution was smooth overall, a few real and practical challenges were encountered during experimentation. The solutions implemented helped improve the accuracy and reliability of the defect detection pipeline.

Challenge Faced	Observation	Solution Applied
Noise after thresholding	Background variations were misidentified as defects	Applied Gaussian blur and morphological opening to suppress noise
Merged contours for nearby small defects	Two or more defect areas merged into one contour	Applied area-based filtering and convex hull segmentation to ensure distinct contour separation
Uneven ROI dimensions	Cropped patches had inconsistent resolution	Resized ROI images to 640×640 while maintaining aspect ratio

Summary of Milestone-1

Milestone-1 successfully completed the most essential stage of the PCB defect detection project. The dataset was cleaned, aligned, and processed using a systematic image subtraction and contour-based pipeline. All defect regions were localized precisely and exported as labeled ROI patches corresponding to the six defect types: Missing hole, Mouse bite, Open circuit, Short, Spur, and Spurious copper. With the dataset now prepared and structured, the project is ready to transition into Milestone-2 Model Training and Evaluation, where a YOLO-based deep learning model will be trained using the extracted ROIs.