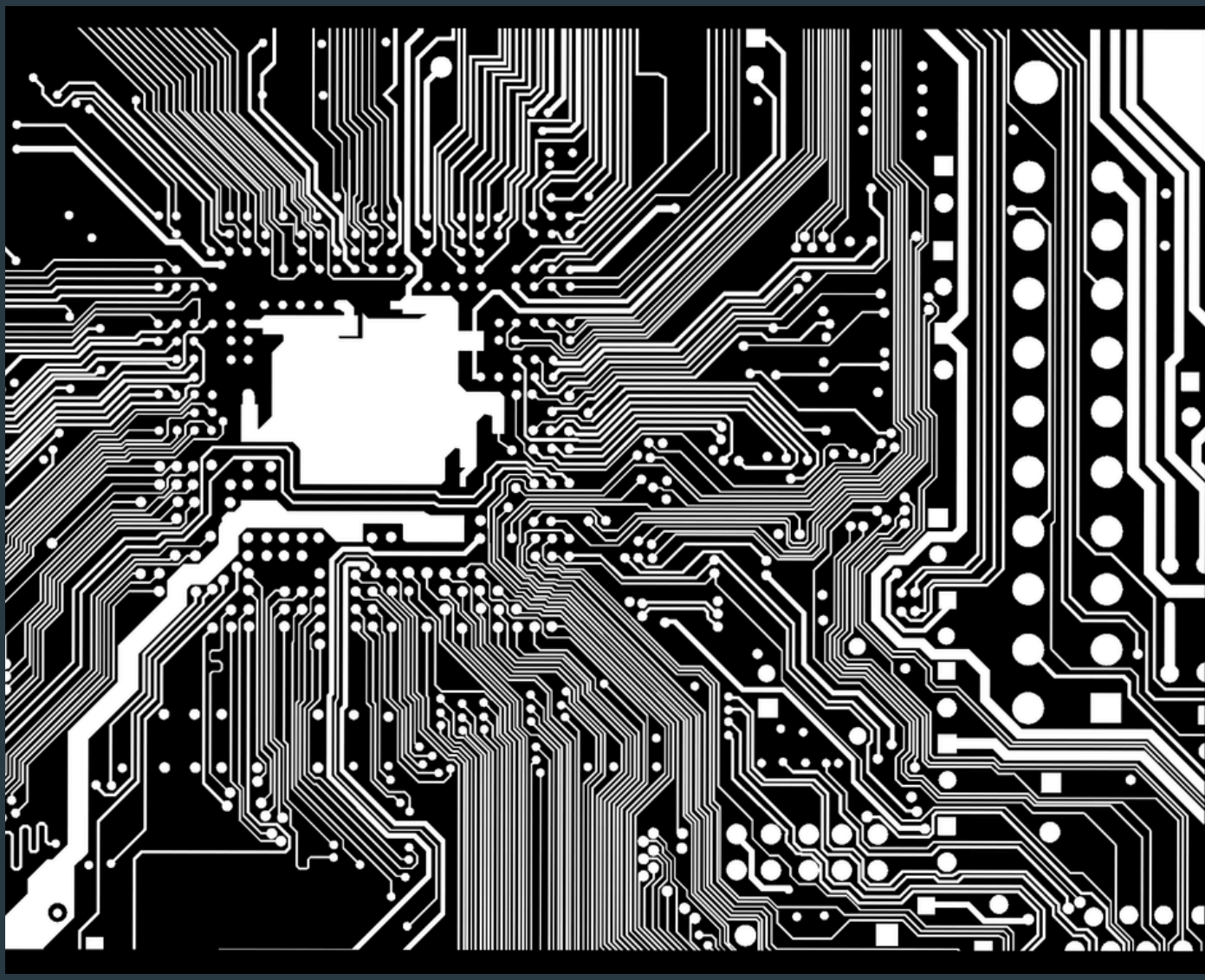
Project Report

PCB Anomaly Detection



by: Himanshu gupta

himanshu.gupta@tu-ilmenau.de

1. Approach and Methodology

The methodology of Model consists of three main stages:

1. **Preprocessing** – Image extraction, alignment, and enhancement.
2. **Defect Detection** – Feature extraction and clustering using PCA-Kmeans.
3. **Post-Processing** – Analysis of detected changes to remove false positives.

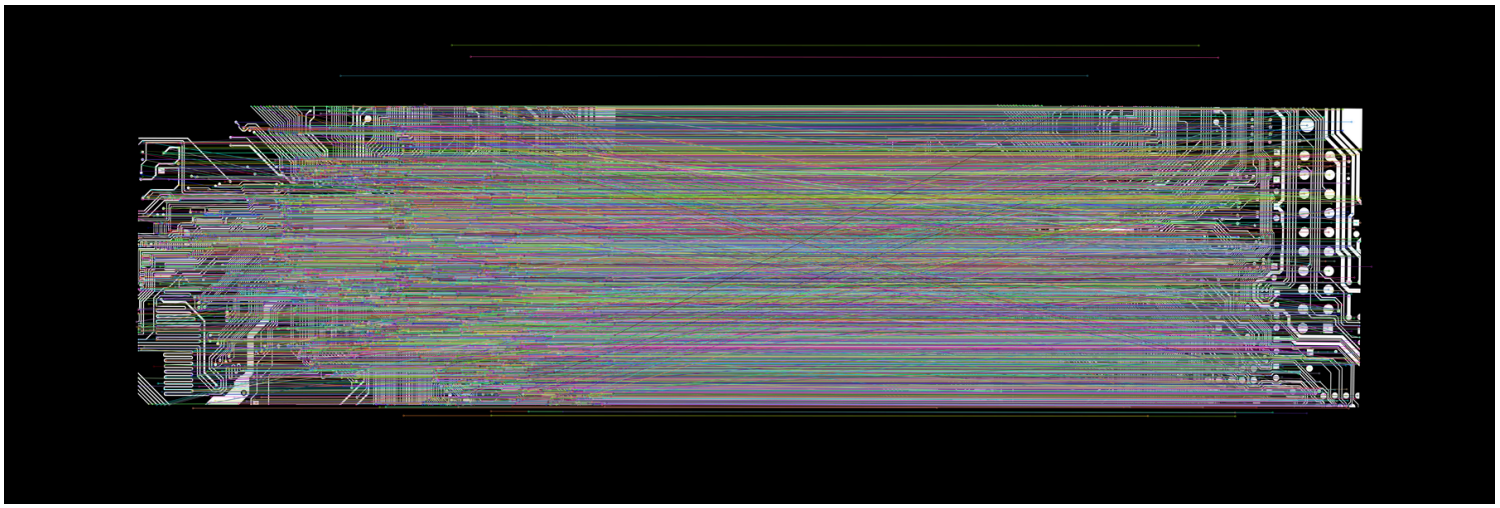
These steps enable Model to identify missing components, misalignments, and soldering defects while being robust to variations in lighting and orientation.

2. Preprocessing Steps

The preprocessing stage refines the input images to ensure reliable comparison and defect detection. The key steps include:

2.1 Image Registration using SIFT and RANSAC

- Since Golden and Defected PCB images may have slight misalignment, Model performs image registration.
- Scale-Invariant Feature Transform (SIFT) extracts key features from both images.
- Feature Matching is performed, and incorrect matches are eliminated using the Ratio Test (0.8 threshold). a value suggested by David G. Lowe **[1] [2]**.
- The best transformation matrix is computed using RANSAC **[3]** (Random Sample Consensus) to align the defected PCB image with the Golden PCB.



2.2 Histogram Matching for Illumination Correction

- Exact Histogram Specification **[4]** is applied to match color and brightness distribution.
- This reduces false defect detections caused by lighting variations.

3. Model/Algorithm Used– PCA-Kmeans Change Detection

The core defect detection algorithm is PCA-Kmeans **[5]** clustering, which is an unsupervised learning approach. It identifies changes in the PCBs by clustering change significance levels.

3.1 Feature Extraction using Change Descriptors

- Each pixel in the PCB image is represented using change descriptors, which measure differences between the Golden and defected image.
- The descriptor includes:
  - RGB channel difference
  - Grayscale difference
  - Window-based extraction (h×h region around each pixel)

3.2 Principal Component Analysis (PCA) for Dimensionality Reduction

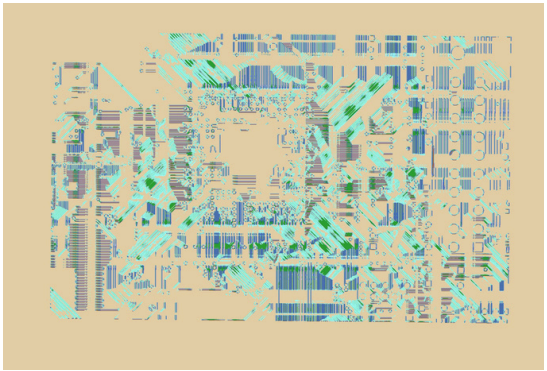
- Since raw pixel differences can be high-dimensional, PCA is applied to reduce the number of features while retaining the most important change-related information.

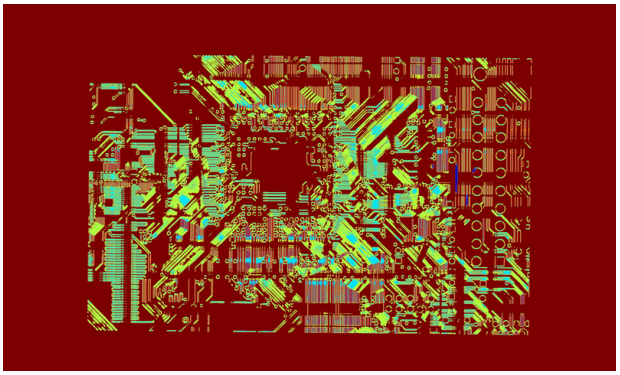
3.3 K-Means Clustering for Change Classification

- The K-Means algorithm is applied to cluster pixels into n classes based on similarity in defect-related features.
- Change significance is determined by assigning pixels to different clusters (e.g., high, medium, low defect confidence).

3.4 MSE Heuristic for Post-Processing

- After clustering, the detected change classes are analyzed using a Mean Squared Error (MSE) heuristic.
- High MSE values indicate significant changes, while low MSE values may represent noise.
- The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) **[6]** is applied to filter out false positive detections





4. Analysis of Results – Evaluation Metrics

4.1 Metric Values:

- Recall: 0.82
- Precision: 0.86

4.2 Interpretation:

- Recall (82%) – The model effectively detects most real PCB defects, minimizing missed defects.
- Precision (86%) – While some false positives are present, they are relatively minor compared to the total analyzed image area.

These metrics were computed using the **"evaluation.py"** script, which compares the model's predictions against the manually annotated ground truth images used for evaluation.

5. Challenges and Potential Improvements

5.1 Challenges:

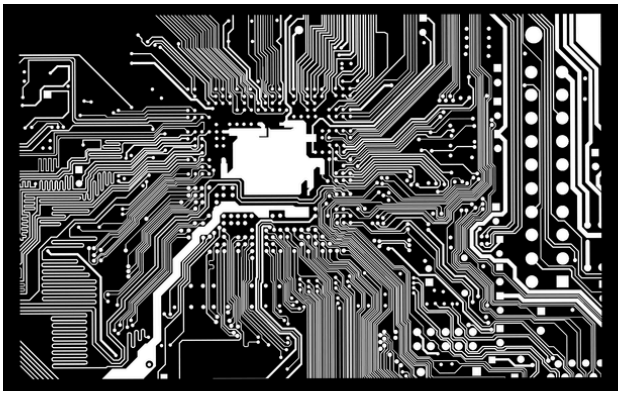
- **Imaging Variability** – Differences in lighting, angles, and reflections can impact detection accuracy.
- **Registration Sensitivity** – Misalignment in reference and target images affects defect detection.
- **Very Fine PCB Defects** – Detecting extremely small and subtle defects was a challenge, as some defects blend into the PCB structure and require high-precision analysis.
- **Time Constraints** – Due to limited time, it was difficult to explore additional techniques to further improve robustness and fine-tuning of model.

5.2 Potential Improvements:

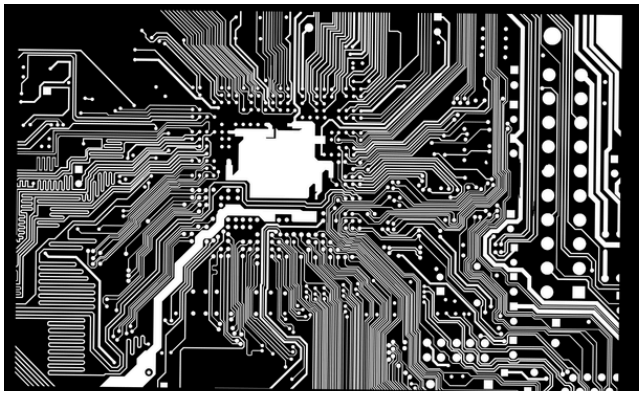
- **Improved Image Processing** – Enhance registration techniques and refine histogram matching.
- **Fine-Tuning for Accuracy** – Combine unsupervised methods with supervised fine-tuning using labeled data.
- **Given more time** – If given more time for submission approaches such as deep learning-based segmentation, domain adaptation, and self-supervised learning could be explored for a more generalized and effective solution.

Visualization

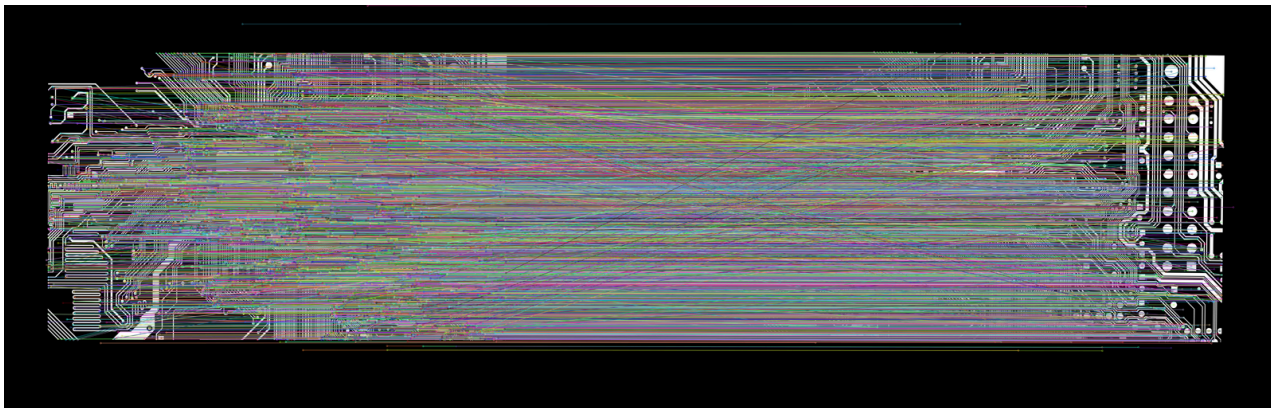
Golden Image



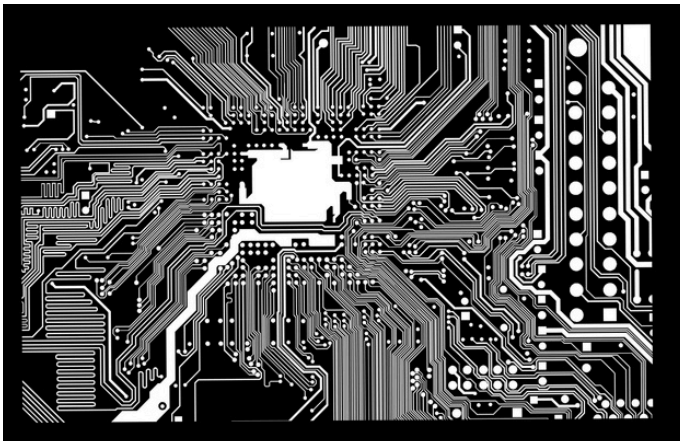
Defected Image

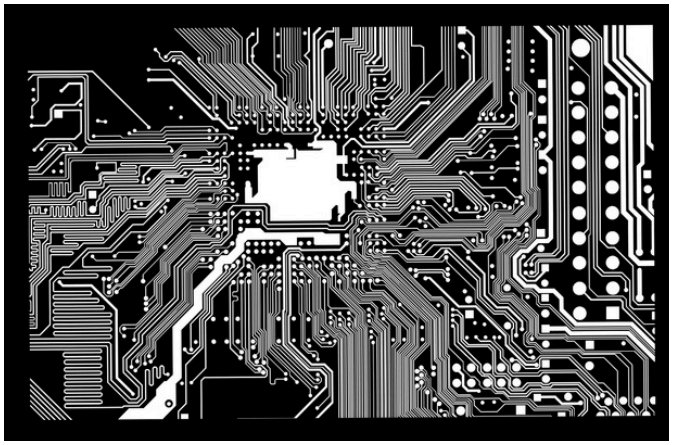


Matching feature point for registration

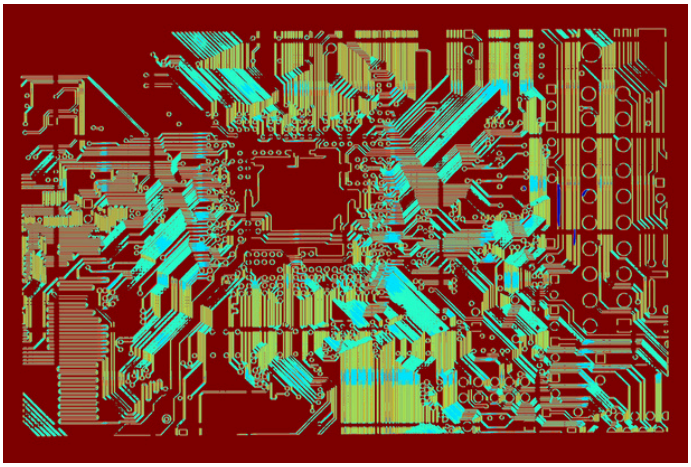


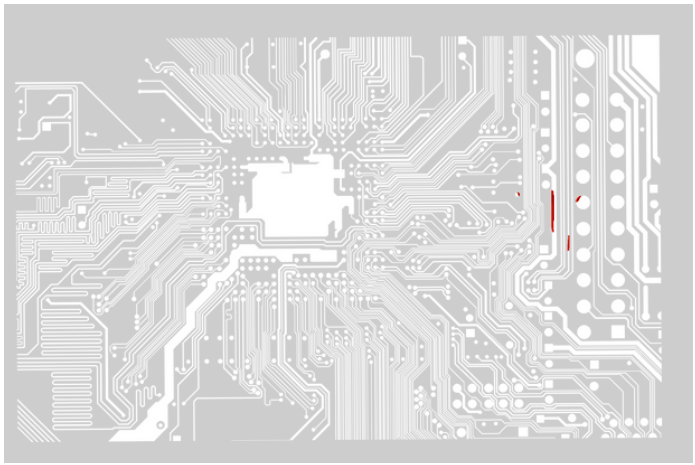
The registration and histogram matching result





The clustering map and the final detection





REFERENCES

[1] David G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the International Conference on Computer Vision, volume 2, pages 1150–1157, 1999.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. In International Journal of Computer Vision, volume 60 (2), pages 91–110, 2004.

[3] Konstantinos G Derpanis. Overview of the ransac algorithm. Image Rochester NY, 4(1):2–3, 2010.

[4] Philippe Bolon Coltuc, Dinu and J-M. Chassery. Exact histogram specification. IEEE Transactions on Image Processing, 15.5:1143–1152, 2006

[5] Turgay Celik. Unsupervised change detection in satellite images using principal component analysis and k-means clustering. IEEE Geoscience and Remote Sensing Letters, 6(4):772–776, 2009.

[6] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014), pages 232–238. IEEE, 2014.

Important Instruction's about the code

1. Number of Files:

- **main.py:** The primary script that runs the main functionality of the application.
- **registration.py:** Handles the registration process of the image**.**
- **light_differences_elimination.py:** Focuses on eliminating lighting differences in images or data for better analysis.
- **evaluation.py:** Performs evaluation of the model, using Ground truth image and result Recall, Precision values.
- **PCA_Kmeans.py:** Implements Principal Component Analysis (PCA) and K-means **c**lustering for data analysis and dimensionality reduction.
- **global_variables.py:** Contains global variables shared across different scripts for consistency and data handling.
- **histogram_matching.py:** Implements histogram matching techniques to align the distribution of pixel values in images.
- **run_example.sh:** A shell script used to execute the program or demonstrate its functionality**.**

2. Conda Environment:

Create a Conda environment with the following installations to run the scripts:

- conda install pytorch torchvision -c pytorch
- conda install numpy scipy matplotlib
- conda install opencv pillow scikit-learn scikit-image
- conda install keras tensorflow

NOTE:

1 - To run the model on any given image and detect defects, execute the **./run_example.sh** script. It contains the necessary structure and instructions for running the model.
2 - The **"clustering_data.csv"** file has been deleted by me from every PCB output directory due to its large size, making it difficult to upload via email