Localization of Defects on PCBs with Diverse Acquisitions

A. Sainath Chaithanya * and L. Nirmala Devi

Department of Electronics and Communication Engineering, University College of Engineering, Osmania University, Hyderabad, Telangana, India
Email: chaitanya.aravalli@gmail.com (A.S.C.); nirmaladevi@osmania.ac.in (L.N.D.)
*Corresponding author

***Abstract*—The Printed Circuit Board (PCB) accommodates various Integrated Circuit (IC) components arranged in a specific layout of bond pads, lines, and tracks. Throughout the manufacturing process, irregularities or defects often occur during drilling, etching, stripping, and other stages, impacting the performance and functionality of the circuit board. Many of these defects are related to soldering pads and copper balance, identifying them through manual inspection is time-consuming and error-prone. This necessitates the use of Automated Optical Inspection (AOI). Practices like template matching often require two identical PCBs, which are compared using mathematical algorithms to detect differences. However, they are not resilient to viewpoint variations and non-rigid deformations. The current inspection process primarily focuses on rectifying PCB images captured with tilts ranging from 0° to 84° using homography principles. This correction process operates within a maximum run time of 7.96 s. The adjusted images then undergo analysis via a pattern-matching unit, where the system receives images of the same PCB pattern, each exhibiting different defects. Structural information mapping is performed using various spatial-domain feature-based matching algorithms. When evaluated using SSI and MSE metrics, the model achieved high matching percentages of 99.67%, 99.75%, and 99.30%, and low error rates of 0.343%, 0.358%, and 0.721% for three different types of PCB designs considered. Additionally, the model excels in precisely identifying the location of defects in the PCB images without using bounding boxes, in accordance with the description of the co-images through a segmentation approach. Overall, the proposed system effectively corrects skew, accurately detects abnormalities and outperforms traditional assessment systems.**

***Keywords*—contours, homography, image comparison techniques, image registration, image segmentation, Printed Circuit Board (PCB) defects**

I. Introduction

Many electronic devices, ranging from smartphones to personal computers, rely on a system of components integrated onto Printed Circuit Boards (PCBs) to achieve their functionalities. During the manufacturing process [1], the dimensions of insulators and conductors on PCBs may change due to various factors such as dust, over-etching, under-etching, unwanted copper, and the presence of spurious metals. These factors can result in abnormalities, including missing holes, incorrect hole sizes, and solder breakout, which are associated with soldering pad defects. Additionally, issues like open circuits, short circuits, spur connections, spurious copper, and missing conductors can be linked to copper balancing defects, as demonstrated in Fig. 1. These flaws can compromise the performance of the interconnected electronic components, thereby impacting the entire system. Therefore, prior to component assembly, it is crucial to subject PCBs to quality assurance measures.

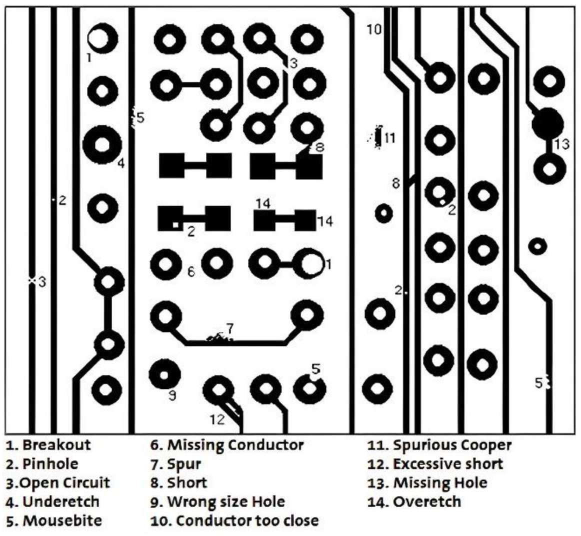


Fig. 1. Common defects on PCB.

For many years, human operators have been responsible for scrutinizing PCBs and monitoring the results. However, in the present era of ultra-large-scale integration, PCBs feature complex and sophisticated patterns. Relying solely on manual visual inspection is no longer advisable due to high scrap rates and an inability to ensure consistently high-quality assessments. Thus, the need for automated optical inspection arises.

Automatic Optical Inspection (AOI) is a non-destructive technique that consists of hardware equipment and software algorithms. The hardware setup includes image acquisition devices with proper illumination settings to capture digital images, while the software executes an inspection algorithm to extract features from the registered images and scrutinize them according to user requirements.

Manuscript received August 24, 2023; revised October 16, 2023; accepted November 16, 2023; published February 15, 2024.

Emerging technologies such as image processing [2] and computer vision, are increasingly employed to facilitate automatic optical inspection procedures, including scrutinizing PCBs. One of the most established methods in non-contact-based Automatic Optical Inspection (AOI) involves the use of reference-based approaches, which require precise alignment between the PCB test image and the PCB template image, making it a crucial step in the inspection process. During the image acquisition process of PCBs, if the object in the image is tilted or captured with variations in viewpoint, analyzing such images may lead to inconsistent results, making their utilization challenging. Hence, any discrepancies such as scaling, rotation, or translation must be estimated and corrected before proceeding to image matching analysis.

The content-based image retrieval system involves automated extraction and representation of visual features for object detection and localization. Object detection aims to identify whether one or more defects are present anywhere in the PCB image. In contrast, object localization identifies instances of object locations in the image. The practice of spotting different defects in PCBs through a bounding box is considered an established approach in addressing the image retrieval problem.

The paper is structured as follows: Section II covers extensive research contributions on skew angle correction and template comparison measures. Section III presents the proposed system architecture and analysis. Section IV provides assessments of the developed algorithm in addressing the identified problem. Finally, Section V concludes the work.

II. Literature Review

In the context of image registration, numerous approaches for detecting and correcting skew in objects are still emerging. Methods like analyzing the principal axis of the image, correlation functions, Fourier transforms, and image feature-based techniques are prevalent. Among these, the Hough transform, nearest neighbor, and projection profile methods are notable. Concerns have been raised regarding the accuracy and speed of angle detection.

In Ref. [3], paper documents often exhibit skew when they are scanned or photographed. The article compares four skew correction techniques: Hough Transformation, Cross Correlation, K-Nearest Neighbor, and Fast Fourier Transformation. The Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR) values after correcting a 75% skewed input image were measured at 23.76%, 24.12%, 23.62%, 26.74% and 25.53%, 31.45%, 24.12%, 37.17%, respectively. The corresponding execution times were observed as 3.375 s, 1.378 s, 2.070 s, and 1.359 s. For correcting the skew of document images, the Fast Fourier transform technique is found to be faster and more accurate than the other methods. While techniques like the Hough transform, cross-correlation, K Nearest Neighbors (KNN), and Fast Fourier Transform (FFT) are used, they may not be as accurate in the presence of complex image content or noise. The Hough transform may produce multiple line candidates, and cross-correlation requires multiple template images, which can be time-consuming for a wide range of skew angles.

In Ref. [4], a novel approach for skew detection in document images, utilizing bounding boxes, a probability model, and Dixon's Q test, is discussed. Bounding boxes are employed to select Eligible Connected Components (ECC), from which the slopes of the skewed document are determined using the probability model. The obtained slope is then used to estimate the skew angle through Dixon's Q test and projection profiles. The Average Error Deviation (AED) obtained with this approach, standard Hough Transform (SHT), and projection profile analysis is 0.230 s, 6.120 s, and 0.290 s, respectively, with runtime values of 0.345 s, 2.085 s, and 3.776 s. The main novelty here is the combination of the probability model, Q test, and PP method, striking a balance between computational complexity and accuracy. This method can be effective for document images with well-defined text regions and a relatively uniform background. However, it needs filters that slide over the image, counting black pixels and calculating probability measures, which can be time-consuming for a wide range of skew angles. The technique may not perform well in the presence of complex image content or significant noise.

In Ref. [5], a technique based on the Pulse-Coupled Neural Network (PCNN) and the Gabor filter is introduced. The Gabor filter is applied to both the template and rotated images to extract Gabor features. Using a PCNN model, the Gabor features of the original image are compared with those of its rotated version. The orientation features of the image are determined by identifying the global minimum of the correlation coefficients "C" between the rotated counterparts and the query image. The rotation angle errors calculated by SURF, BRISK, Fourier-Mellin, and the proposed approach for the input image "Lena" with a specified rotation of 60% are found to be 5.2738 s, 0.0538 s, 0.0060 s, and 0.0200 s, respectively. The suggested model demonstrates a strong accuracy when compared to other rotation techniques. Gabor filter-based skew estimation leverages the frequency and orientation characteristics of the image to estimate the skew angle; however, it is sensitive to noise and variations in image content. Multiple predefined Gabor filters with known orientations are generated and applied to the image, which involves time-consuming calculations of local energies.

In Ref. [6], a method for calculating the skew of documents written in the Gujarati script using the Hough Transform methodology is outlined. Before proceeding with any further processing, morphological operations, such as dilation, erosion, and thinning, are applied. Subsequently, the image is subjected to the Hough transform to determine the skew angle. This technique achieves an accuracy of 44%. It's worth noting that this approach provides accurate results within a certain range of angles. However, as the angle increases, the error between the actual and estimated angles also increases. For instance, if the actual rotation angle is 8°, the estimated rotation angle is calibrated as 8°. But for a 16% rotation, the estimated angle is reported as 12%.

Fabrizio [7] introduced a straightforward and precise approach that utilized KNN clustering to pre-process the input document image and estimate the skew angle in the frequency domain. In performance evaluation metrics, AED is observed as 0.09 for LRDE-EPITA-b, 0.08 for Ajou-SNU, and 0.07 for the present approach. The nearest neighbor method for skew estimation is a simple and computationally efficient approach. However, it may not be as accurate, especially in the presence of complex image content or noise.

In Ref. [8], The assessment of PCBs through subtraction focuses on identifying areas with imperfections. The paper presents an imperfection detection strategy based on machine vision. The model achieves accuracies of 40% and 80% with images of different resolutions, including 447×385 and 1748×931, and demonstrates a detection speed ranging from 0.856 s to 2.68 s. This indicates that image resolution directly influences accuracy. However, this approach is relatively simple and may not be the most robust choice for complex scenarios or when the differences involve substantial transformations, such as rotation, scaling, and perspective changes.

Existing object detectors that rely on IoU loss functions tend to make inaccurate predictions when dealing with densely packed objects in synthetic image data. Balanced IoU (BIoU) is proposed as a solution to address this issue in [9]. To tackle the localization problem, this loss function takes into consideration the parameterized distance between the centers and the minimum and maximum edges of the bounding boxes. The BIoU loss function demonstrated an improvement of 3.70% in Average Precision (AP) on the COCO dataset, 6.20% at AP55 on SKU110k, and 9.03% at AP80 on the custom e-scooter dataset. When it comes to handling objects of various scales and bounding boxes of different sizes, BIoU provides a fairer assessment, but there is still room for improvement in the precision of targets.

In Ref. [10], a template matching method based on correlation analysis techniques is proposed for detecting nodules in lung CT images. The model was tested on 100 CT scan medical images and achieved an accuracy of 86%. Correlation methods are employed to map patterns based on intensity similarities, enabling the detection of multiple object occurrences in images. This approach effectively handles background noise and non-rigid transformations. However, it requires the search for exact patches in images through multiple sliding, which can be time-consuming.

In Ref. [11], the detection of PCB defects in raw images has addressed practical challenges such as inadequate illumination, variations in acquisition height, and image tilting through the use of tools and image processing algorithms. The paper did not delve into a detailed methodology but provided an overview where a typical PCB image is examined for issues like poor illumination and tilt. If these issues are present, they are corrected using haze detection-correction and Hough transform techniques. Subsequently, the processed image is cropped to focus on the region of interest and then subjected to template matching for defect detection. However, the paper lacks the presentation of stage-wise implementation results.

In Ref. [12], the importance of Image Quality Assessment (IQA) in digital image processing applications is discussed, based on statistical dependence and intensity similarities between the two images being compared. The paper provides an overview of various metrics, such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Feature Similarity Index (FSIM). The former two are absolute errors, while the latter two are perception- and saliency-based errors used to evaluate image quality. The paper also emphasizes the differences between these metrics and suggests that SSIM and FSIM, which take into account human perception, offer a more accurate representation of image quality in practical applications involving noise and other visual factors.

The aforementioned studies have led to the emergence of a new challenge: during image acquisition, scene objects are captured from diverse perspectives and under varying conditions. Skewing is one such issue, where the image undergoes affine or similarity transformations [13], gets straightened, and is then applied to a pattern-spotting system. It's not feasible to design a universal metric evaluation method that applies to all registration tasks. Therefore, the proposed work focuses on identifying and correcting the cumulative deviation angle that the PCB object attains through the principles of Homography. The work also emphasizes detecting differences in the PCB images using reference-based segmentation approaches rather than conventional bounding boxes.

To add an experimental dimension to the work, the template image is also assumed to have abnormalities but is acquired without any tilt, ensuring a consistent reference. Consequently, instead of comparing defective and non-defective boards each time, we are mapping similar patterns of boards that carry different defects and tracking down these differences. This process enhances the precision of detection and reduces the number of inspection cycles.

III. Materials and Methods

On this occasion of the PCB inspection, the current system considers geometric deformation between the consecutive image acquisitions [14]. One image is captured in regular mode (non-skewed), while the other exhibits some angular deformity (skewed). Furthermore, the effort is directed towards the template matching of two PCB images, which involves mapping patterns in one image against the other, evaluating and then pinpointing one-to-one differences, also known as abnormalities, through image comparison algorithms.

The outline of the system model is illustrated in Fig. 2. The entire work is carried out within Anaconda's Jupyter notebook and executed using OpenCV in Python, with the importing of relevant libraries.

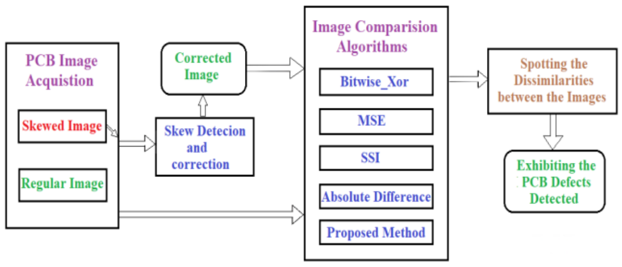

PCB Image Acquistion
Skewed Image
Regular Image
Corrected Image
Skew Detecion and correction
Image Comparision Algorithms
Bitwise_Xor
MSE
SSI
Absolute Difference
Proposed Method
Spotting the Dissimilarities between the Images
Exhibiting the PCB Defects Detected

Fig. 2. Block diagram.

A. *Object Acquisition / Image Dataset*

The model utilizes the "The Open Lab on Human-Robot Interaction of Peking University's PCB Defect Dataset". The registration task encloses ten different PCB image templates used as references. Artificial defects, such as missing holes, mouse bites, spurs, spurious copper, open circuits and shorts, are introduced at various locations on these templates, resulting in a dataset of 1,386 images for detection and classification. Additionally, there is an augmented dataset comprising 10,668 images, derived from slices of each of the 1,386 complete images, accompanied by corresponding annotation files [15]. A selection of these derived PCB image slices has been included, and to enhance the model's robustness, rotated and skewed images (which can be generated through the program) have been considered for the current experimentation.

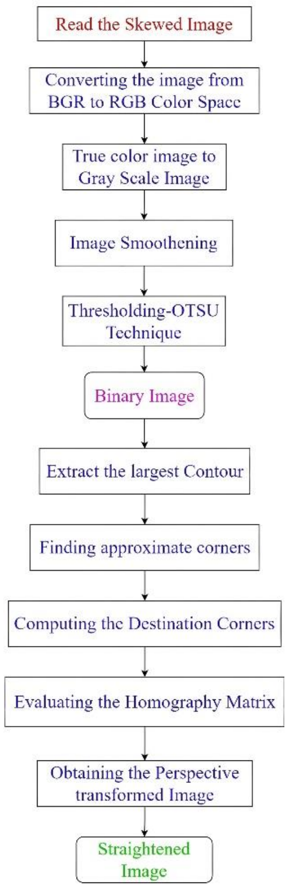

Read the Skewed Image
Converting the image from BGR to RGB Color Space
True color image to Gray Scale Image
Image Smoothening
Thresholding-OTSU Technique
Binary Image
Extract the largest Contour
Finding approximate corners
Computing the Destination Corners
Evaluating the Homography Matrix
Obtaining the Perspective transformed Image
Straightened Image

Fig. 3. Skew estimation and correction.

B. *Skew Detection and Correction*

The objective of assessing the skew of an image is achieved by finding the largest contour and discovering the active contour points that are the corners of the object. Subsequently, skew is corrected using a homography matrix. The flowchart of the proposed subsystem for skew angle estimation and correction [16] is illustrated in Fig. 3.

Sensing the image: A query image that has undergone affine transformations or similarity transformations, such as the skewed images shown in Fig. 4, is considered as input and is read using the "imread" function from the OpenCV library.

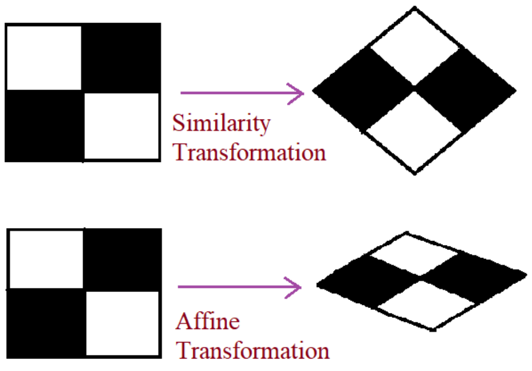


Fig. 4. Skewed images.

BGR to RGB Color Space: OpenCV reads an image in BGR format and it can be plotted using the matplotlib 'imshow' function. However, the matplotlib function interprets the image in RGB file format. Therefore, color space conversion is essential.

Gray scale conversion: Sophisticated computations can be more easily accomplished on grayscale images than on color images. This is due to the variations in human vision's sensitivity to the three channels in RGB images. The weighted method of grayscale conversion, as preferred [17], is formulated as

$$I = 0.299R + 0.587G + 0.114B.$$

where 'I' is the grayscale intensity, which is achieved by the 'BGR2GRAY' OpenCV library.

Image Smoothening: To reduce noise and maintain consistency between pixel values while preserving low-frequency information, the image should be smoothed using a low-pass filter. OpenCV offers a function, "cv2.filter2D()", which convolves the kernel with the image and performs an averaging operation on it [18].

Image Binarization: Since the image is assumed to be noise-free, uniform and bimodal in nature, whose histograms have two peaks, Otsu's Method of thresholding has been chosen to set the threshold value in the middle of those peaks. This allows the foreground object to be separated from the background, as visualized in Fig. 5.

To quickly understand this technique, let's assume the input image is noisy.

- In the first case, global thresholding is 127.
- In the second case, Otsu's thresholding is directly applied.

- In the third case, a 5×5 Gaussian kernel filter is used to remove the noise in the image, followed by Otsu thresholding.

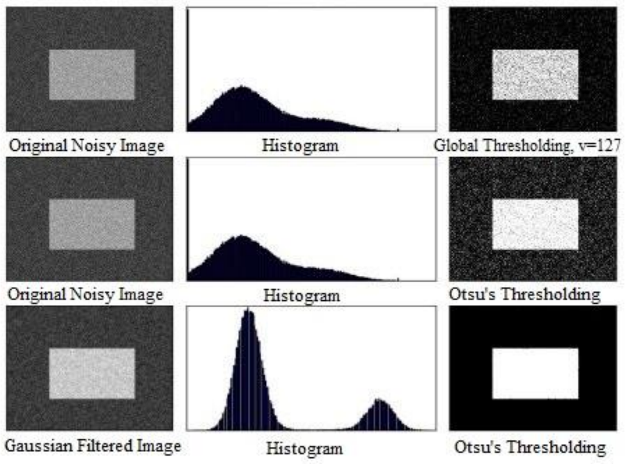


Fig. 5. Otsu's binarization.

Extracting the contour: Curves that connect all the boundary pixels with the same color or intensity are used for shape analysis and object detection. It works meritoriously for binary images. The OpenCV function 'cv2.CHAIN_APPROX_NONE' considers and connects all the boundary (x, y) coordinates as a curve.

Finding the corner coordinates: To approximate a contour [19] shape with a smaller number of vertices, the Ramer-Douglas-Peucker (RDP) algorithm is employed [20], as shown in Fig. 6.

In simple terms, this algorithm takes a curve that describes a shape and reduces the number of its vertices without distorting the original structure. This allows the preservation of shape information with fewer vertices.

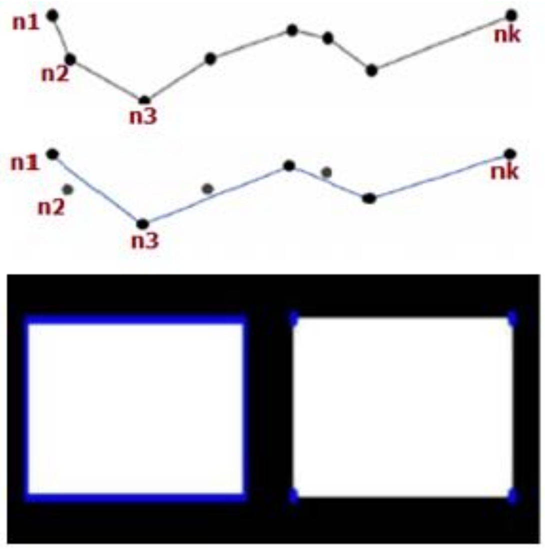


Fig. 6. Ramer-Douglas-Peucker process.

OpenCV to harness the power of RDP

- epsilon = 0.1*cv2.arcLength(cnt, True)
- approx = cv2.approxPolyDP(cnt, epsilon, True)

The "epsilon" parameter, which represents the Accuracy, signifies the maximum distance from the contour to the approximated contour.

In Fig. 7, the green arcs encircling the second and third subplots indicate the approximated curve, where the epsilon values are set at 10% and 1% of the arc length, respectively. The third argument determines whether the curve is closed or not.

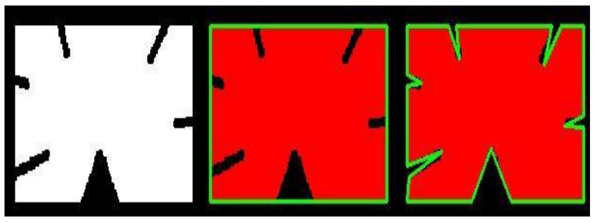


Fig. 7. Left is an isolated shape; middle and right are approximated contours.

Computing the destination corners: Upon obtaining the four corners of the skewed-warped image, the target corner point coordinates are calculated by utilizing the reference corner coordinates from the template image. By applying formulas from analytic geometry, which determine distances between two points, the length and width of the template image are found. The new destination corner points are then computed with (0, 0) as the origin. Subsequently, the height and width are added to the origin to establish the new destination corner points of the skewed image.

Homography Matrix: A sparse set of features (corner coordinates) is detected in the warped image and transformed to the destination coordinates. This transformation is necessary because the new corners in the other image are significantly different. To achieve the proper alignment of the image, a simple 3×3 matrix called homography "H" [21] is employed, as illustrated below.

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

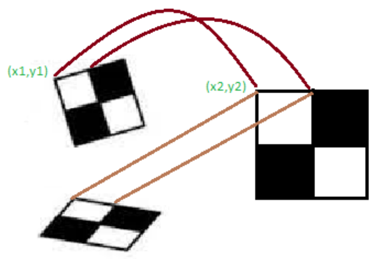


Fig. 8. Image transformation.

As depicted in Fig. 8 above, let (x_1, y_1) be one of the approximated corners in the first image, and assume that (x_2, y_2) are the destination corners of the same physical feature point in destination image. Then, the homography matrix H is associated as

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

With at least four corresponding points in the image, one can find the homography H through the OpenCV function "findHomography". Once the computed "H" is applied to all the pixels in a raw image, new pixel locations are obtained in the transformed image. Additionally, a robust line detection algorithm called Random Sample Consensus (RANSAC) [22], which iteratively generates line models using a subset of points, is also taken into account.

Perspective Transformed Image: After accurately calculating the homography, the transformation can be applied to all pixels in one image to determine their new positions in the unwarped image. This process is carried out using the "warpPerspective" function in OpenCV.

Skew correction and straightened image: The angle at which the image got exposed to tilt or skew during the image registration is being identified by the model and corrected through the necessary orientation taken care of by homography and perspective transformations.

C. *Image Comparison Algorithms*

Once the warped image is corrected and is in concord with the registration of the regular image, the two are sent to image comparison models. Based on the nature of the image, the type of template, the type of algorithm selected, and the kind of problem to be solved, there exist a number of techniques that measure similarities between two images. In a broader sense, the template matching algorithms are classified into feature-based and area-based approaches. One corresponds with respect to features and control points, and the other matches or operates directly on the bulk of intensity values among the two.

In spite of having numerous industrialized techniques, one cannot ensure the best result in all situations with a particular model. Depending on the criteria selected, we exercised some techniques and algorithms regardless of the application, whose flow charts are also presented below.

1) *Bitwise_xor*

Performs a bit-wise "exclusive or" operation on the intensities at consecutive locations on two arrays or images, letting src1 and src2 have equal size or elements, then

$$dst(I) = src1(I) \oplus src2(I); \ \ if\ mask(I) \neq 0$$

The OpenCV function cv2.bitwise_xor (src1, src2, dst, mask) is used for operation, and the arguments are:

- src1, src2: Input image arrays, which are single-channel and either 8-bit or floating-point.
- dst: Output array for the resulting image, with the same characteristics as the input image arrays.
- mask: An operation mask, which is an input/output 8-bit single-channel mask.

2) *Mean Square Error (MSE)*

Using this method [23], we compare pixel values located at the same coordinates in the two images having the same height, width, and number of channels.

Assuming that the reference image I and the template image K are of sizes m x n, then the total number of pixels in each image is mn.

The mean squared error equation:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

I (i, j) and K (i, j) are the intensity values of the template and test image, respectively.

- Convert the images from unsigned 8-bit integers to floating points.
- Then take the differences between the images by subtracting the pixel intensities.
- Finally, square these differences and sum them up. Hence the mean squared error.

The process starts by dividing the sum of squares by the total number of pixels in the image. It's important to note that a value of 0 for MSE indicates perfect similarity; a value greater than 1 implies less similarity and follows proportionally. Therefore, similar images will have a lower MSE value. The similarity, or matching, between them is inversely proportional to the MSE. Large distances between pixel intensities do not necessarily mean the contents of the images are dramatically different. The flowcharts of Bitwise_Xor and MSE are shown in Fig. 9.

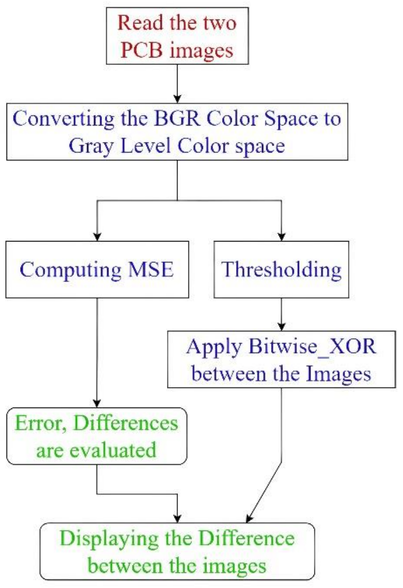

Read the two PCB images
Converting the BGR Color Space to Gray Level Color space
Computing MSE
Thresholding
Apply Bitwise_XOR between the Images
Error, Differences are evaluated
Displaying the Difference between the images

Fig. 9. Left depicts MSE, right demonstrates Exclusive-OR process flow.

3) *Structural Similarity Index Measure (SSIM)*

SSIM measures the perceptual difference between two similar images taken from the same shot that may possess different adaptations, i.e., compression or filtering. The Structural Similarity Index endeavors to model the change in structural information among the images, whereas MSE estimates the errors.

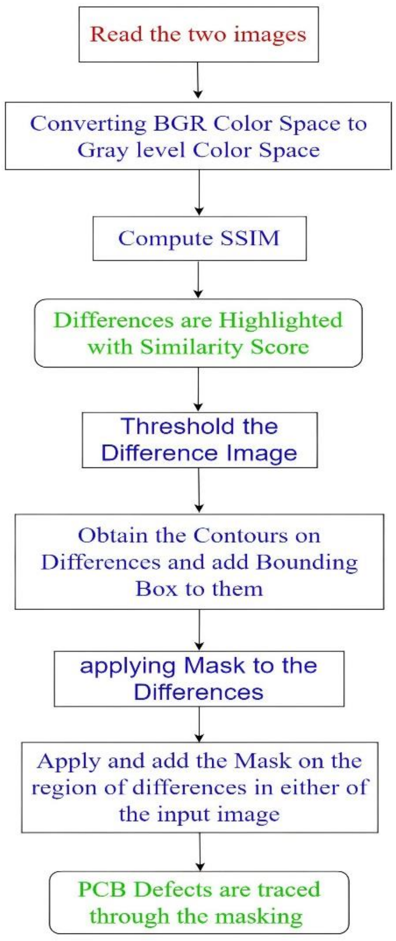
The below equation accounts for exploring the changes in the structure of the two images.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x + \sigma_y + c_2)}$$

- (x, y): pixel location in the NxN window in each image.
- μ_x μ_y: the mean of the pixel intensities in the x and y directions.
- σ_x σ_y: the variances of intensities in the x and y directions, along with the covariance σ_{xy}.

This method got standardized in the Scikit-Image library for image processing; the function "compare_ssim" [24], produces the similarity score and differences in the image.

The score can fall into the range of [–1, 1], and 1 indicates a perfect match. The diff image illustrates the places where the two input images differ.



Read the two images
Converting BGR Color Space to Gray level Color Space
Compute SSIM
Differences are Highlighted with Similarity Score
Threshold the Difference Image
Obtain the Contours on Differences and add Bounding Box to them
applying Mask to the Differences
Apply and add the Mask on the region of differences in either of the input image
PCB Defects are traced through the masking

Fig. 10. Structural Similarity Index process flow.

The workflow is highlighted in Fig. 10, and later processes are elaborated in steps.

- We then apply thresholding to our "diff" image using both normal thresholding and the OTSU method with the OpenCV functions "cv2.THRESH_BINARY_INV" and "cv2.THRESH_OTSU".
- After finding the contours in the "thresh" image, the numbers of contours are stored in a list, and rectangular boxes are used to enclose the regions identified as "different."
- These rectangular boxes iterate over the complete list of contours. The 'cv2.boundingRect' function is utilized to create bounding boxes around the contours.
- We store relevant (x, y) coordinates as "x" and "y," and the width and height of the rectangles as 'w' and "h." Then, we use these values to draw a rectangle around each contour in the image using "cv2.rectangle".

This setup effectively highlights the differences in each image when compared to others using boxes.

4) Absolute difference

This arithmetic has applications in object recognition and movement estimation for generating dissimilarity maps designed for stereo imagery.

Consider Fig. 11, which includes a two-dimensional 5×5 template "A" with intensity values at coordinates (a, b). This template is to be compared with the intensity of the input image 'B' at coordinates (a, b) as shown below.

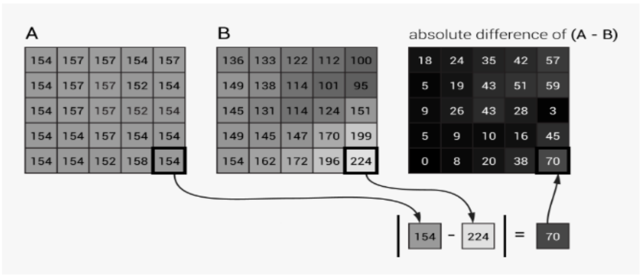


Fig. 11. Absolute difference between images.

For each pixel at locations (a, b), the difference between the gray level values at these points is calculated as

$$d(A,B) = |I_A(a,b) - I_B(a,b)|$$

As per the formula, the smaller the difference obtained via the AD function, the closer the match will be. The two images are alike if the result of the sum of all the absolute differences is zero.

The OpenCV function cv.absdiff calculates the per-element absolute difference between two arrays where they have the same size and type [25],

$$dst(I) = saturate\ (|src1(I) - src2(I)|)$$

The flow is depicted in Fig. 12 and some of the processing steps are discussed below:

Dilation: A binary image is obtained using one of the techniques discussed earlier. Following this, dilation is performed by adding pixels to the boundaries of objects in an image through the 0-padding expansion operator, which enhances the visibility of foreground objects.

Contours: Contour extraction is essential for pattern recognition and object detection, enabling the identification, classification, and analysis of objects within an image.

In OpenCV, finding contours is like locating a target object against a black background, where the object is represented in white and the background in black. The "cv2.findContours()" function takes three arguments: the first being the source image, the second for contour retrieval mode, and the third for a modified source image.

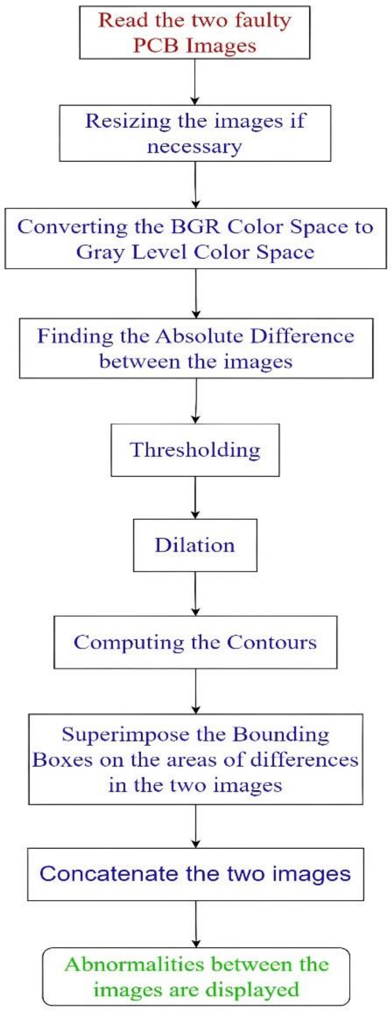

Read the two faulty PCB Images
Resizing the images if necessary
Converting the BGR Color Space to Gray Level Color Space
Finding the Absolute Difference between the images
Thresholding
Dilation
Computing the Contours
Superimpose the Bounding Boxes on the areas of differences in the two images
Concatenate the two images
Abnormalities between the images are displayed

Fig. 12. Absolute difference process flow.

Bounding boxes annotation on images: Bounding box annotation involves manually labeling an image by drawing a rectangle around a specific object or feature. This is commonly used in computer vision and machine learning, especially for object detection. An annotator marks the object, assigns a class label, and provides the coordinates of the top left and bottom right corners of the bounding box. Although it can be time-consuming, this annotation is crucial for immediate object detection or training machine learning models (see Fig. 13).

An imaginary rectangle that includes an object and a set of data points meant for object detection, one must specify four parameters for representing each object in the box,

- The top left and bottom right point coordinates.
- Center coordinates of the box and its width and height.
- Confidence: Indicates how likely it is that the object is present in that box. A confidence score of 0.9 represents a 90% likelihood that the object truly exists in that box.
- Class: What is the object inside the box? example: car, cat, truck, person, etc.

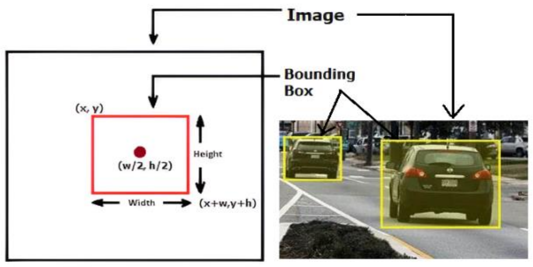


Fig. 13. Bounding box annotation in images

Data annotators [26] draw the rectangles that outline and grip each object within the underlying image by defining its attributes, which makes easier for machine learning algorithms to find what they are looking for.

Masking: OpenCV Image Masking [27] is a powerful function for manipulating images. It allows the application of effects to a single image and creates an entirely new look. It is also possible to add text and special effects, and even convert images to a different file format. It has functions like concatenate, stack, and block, providing more prevailing stacking and concatenation operations, to join images horizontally and vertically.

5) *Proposed approach*

In image processing and computer vision, image segmentation pays off by partitioning a digital image into multiple segments and assigns a label to a group of pixels that carry similar visual characteristics and textures in an image to make the context more meaningful and easier to analyze.

The process flow is shown in Fig. 14 for the present experiment and is as follows:

- As soon as the visualization of images complete, we need to inspect their shape equality; if it is found to be equal, the BGR images are converted to gray-level color space as a part of preprocessing; otherwise resizing is a must to have the same dimensions.
- Before proceeding, it’s important to improve the quality of the images. This can be achieved by enhancing contrast and filtering out noise. Salt and pepper noise can be effectively removed using a median filter with a kernel size of 7×7. Additionally, any high-intensity variations within the image can be eliminated with the use of a Gaussian filter.
- To determine the number of classes present in the image and enable multi-level thresholding, it is essential to evaluate the histograms of individual images for which the optimal threshold is adaptively determined based on the number of peaks in the histogram.
- Then we can move on to segmenting different parts of the image. The present image exhibits three peaks in the histogram, which are deciphered as three successive components in the PCB image:
    1. Soldering Pads
    2. Wire Tracks
    3. Image Background
- The segmented, well-defined patterns are extracted in each image and are subjected to an image

subtraction operation where the defects that fall either into soldering pad or copper balancing are detected; these are enclosed using masking and gripped onto any one of the two underlying images by means of different colors for immediate and precise visualization.

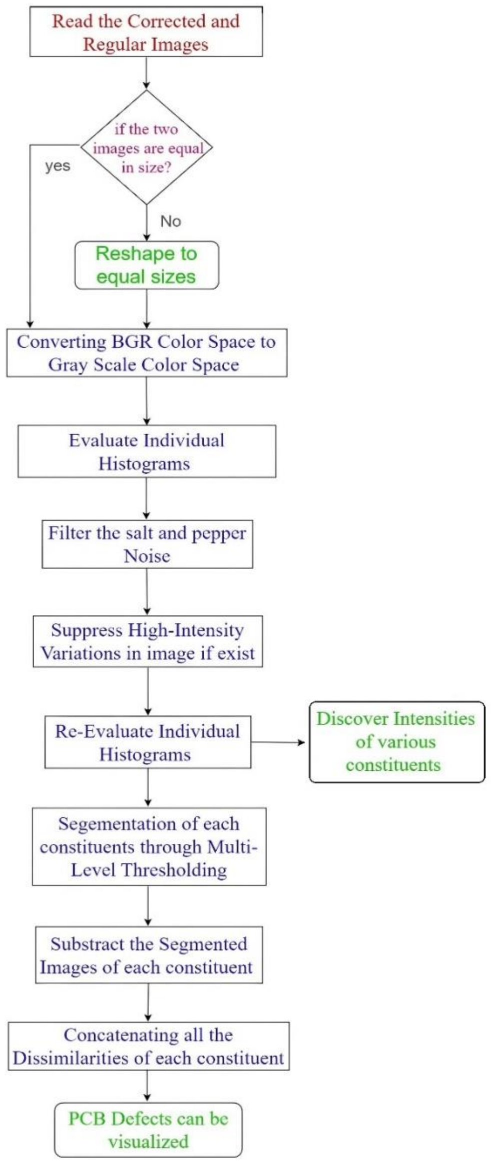

Read the Corrected and Regular Images
if the two images are equal in size?
yes
No
Reshape to equal sizes
Converting BGR Color Space to Gray Scale Color Space
Evaluate Individual Histograms
Filter the salt and pepper Noise
Suppress High-Intensity Variations in image if exist
Re-Evaluate Individual Histograms
Discover Intensities of various constituents
Segementation of each constituents through Multi-Level Thresholding
Substract the Segmented Images of each constituent
Concatenating all the Dissimilarities of each constituent
PCB Defects can be visualized

Fig. 14. Proposed technique process flow.

IV. Results and Discussions

The abnormalities are clubbed into three groups:

- Soldering pads (missing hole, wrong size hole).
- Incomplete copper (open circuit, mouse bite).
- Excessive copper (spur, short, spurious copper).

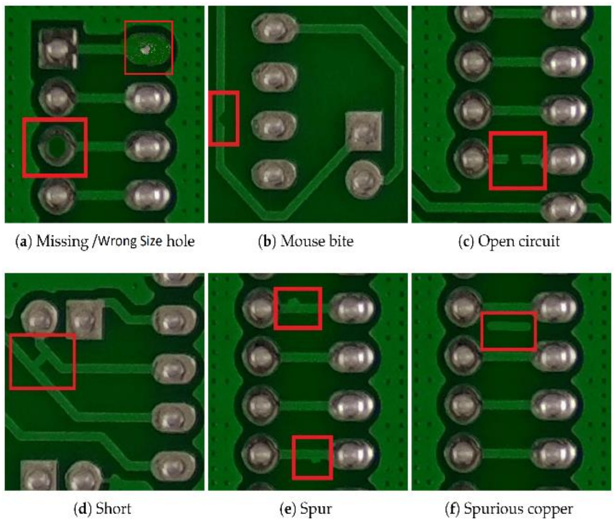


Fig. 15. PCB's with three group of abnormalities.

These defects shown in Fig. 15 are considered to be present in normal and tilted acquisitions of PCB. Instead of always appealing to conventional ideas, that is, taking defect-free template as reference and query image for the test, the system is fed with boards of the same pattern but with different defects in each, serving as template and query for each other, the reason for making this amendment is, Let's consider a scenario where we have a flawless template and two flawed images, and our task is to locate the defects in these images. The typical approach involves comparing the flawless template with the first query image and then with the second query image. This comparison process takes two system clock cycles in its entirety, but when the two query images are mapped one to other the task of locating the differences consumes only one system clock cycle under assumption that a prior knowledge is given on kind of defects a board possess.

For the best-case scenario, we take three distinct PCB designs with two units (image A, image B) for each design, in which one has one kind of defect and the other has other different defect and are subjected to skew. The initial focus is on adjusting the skew, after which proceeding with inspection through matching algorithms. This paper conducted a comparison between the proposed Homography technique and the commonly used Hough transform for line detection. The results reveal that Hough transform techniques fail to correct objects exposed to affine transformations, leading to inaccurate outcomes. Therefore, they are considered Not Applicable (NA) for such cases.

As depicted in Fig. 16, the three different designs of PCB images are skewed with similarity and affine transformations shown in subplots 1 and 2 and its corrected image produced in subplot3.

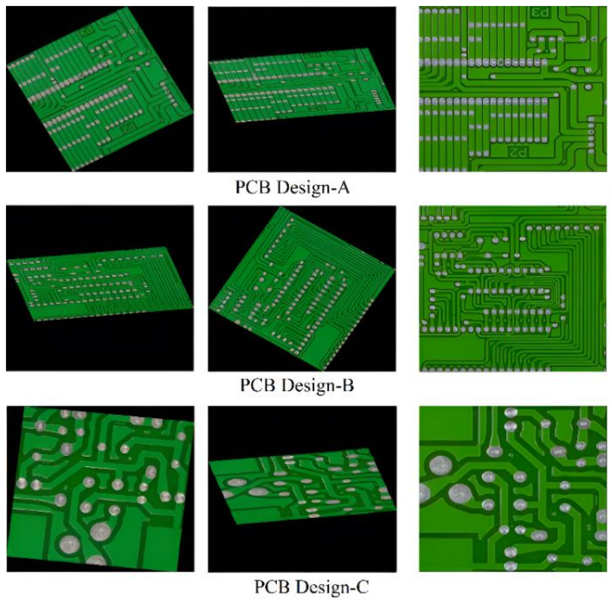


Fig. 16. PCB Designs with diverse acquisitions & skew corrections.

The actual values of tilt registered by similarity transformation for three different PCB designs are 29º, 58º and 84º. Angle of deviation calibrated in degrees by Hough transform and proposed technique are shown in Table I.

TABLE I. ESTIMATED DEVIATIONS

Technique	Design-A	Design-B	Design-C
Hough transform	28,732	57,960	83,697
Our technique	29	58	84

The execution time in seconds for correcting the skew through Hough transform and Homography technique are tabulated in Table II.

TABLE II. EXECUTION TIME FOR SKEW DETECTION AND CORRECTION

Technique	Design-A	Design-B	Design-C
Hough transform	8,017, NA	8.054, NA	7.96, NA
Our technique	6.78, 7.50	7.96, 6.57	6.73, 6.87

Once the discrepancies are corrected, the very next that will be followed is image comparison module where the process of mapping and locating the differences among the two image units for each PCB design is done. As discussed, we consider for each design of PCB, the first PCB design that is Design-A contains mouse bites in the former image unit and missing holes in the latter image unit; the Design-B holds defects open circuits in the former unit and short circuits in the latter unit; and the third Design-C succumbs to defects such as spurs in the former and spurious copper defects in the latter units.

The Bitwise_Xor image comparison algorithm, performs Xor operation between each pixel location in image A with the same pixel location in image B, if the intensities at these locations are mismatched then it marks as dissimilar with respect to each other at those locations and is shown in separate new image window as shown in Fig. 17.

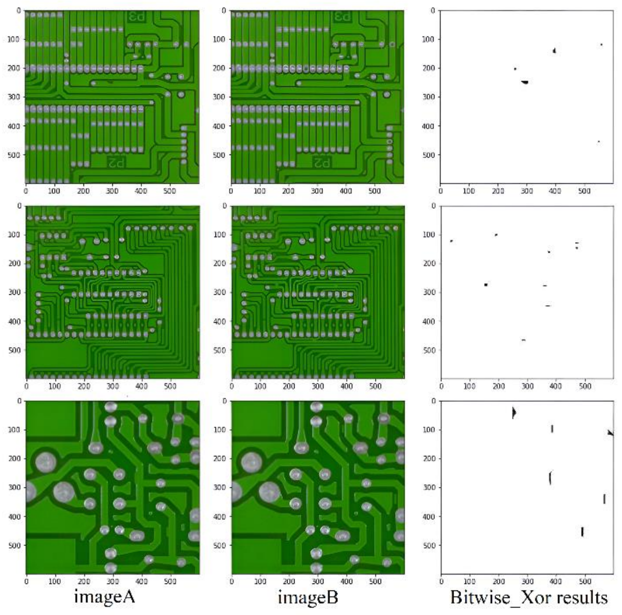


Fig. 17. Bitwise_Xor results.

With reference to Fig. 18, The values obtained in image matching through MSE (mean square error) image comparison technique between the two images in each design of PCB carrying different defects is tabulated in Table III. In ideal case if we map a query defect free PCB with a template defect free PCB the MSE will be approximated to 0% which indicates there is no error or differences exist for one to other.

TABLE III. MSE VALUE BETWEEN IMAGES IN EACH PATTERN

Class	Design-A		Design-B		Design-C	
Images	ImageA	ImageB	ImageA	ImageB	ImageA	ImageB
MSE	0.343%		0.358%		0.721%	

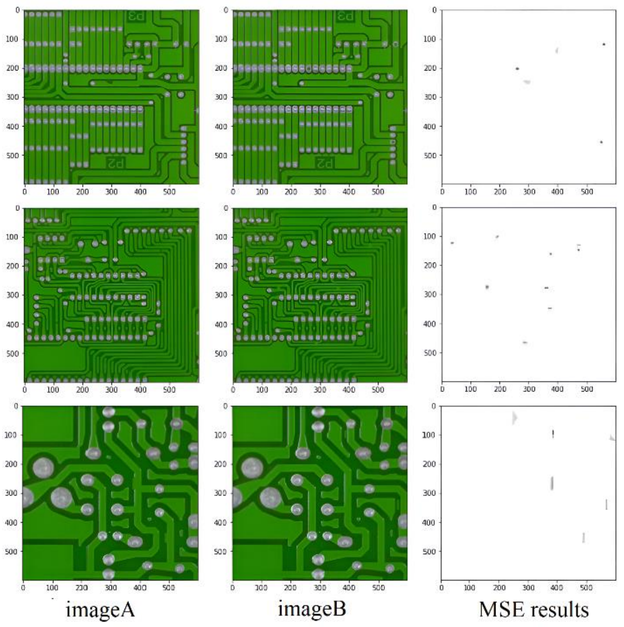


Fig. 18. MSE results.

The absolute difference image comparison algorithm results are in Fig. 19, the differences among the images in each design are portrayed by a rectangular bounding box.

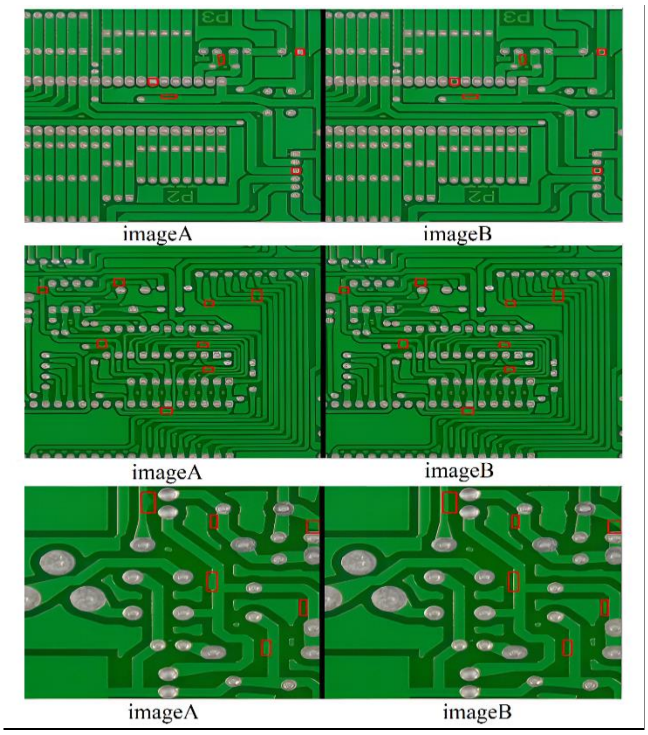
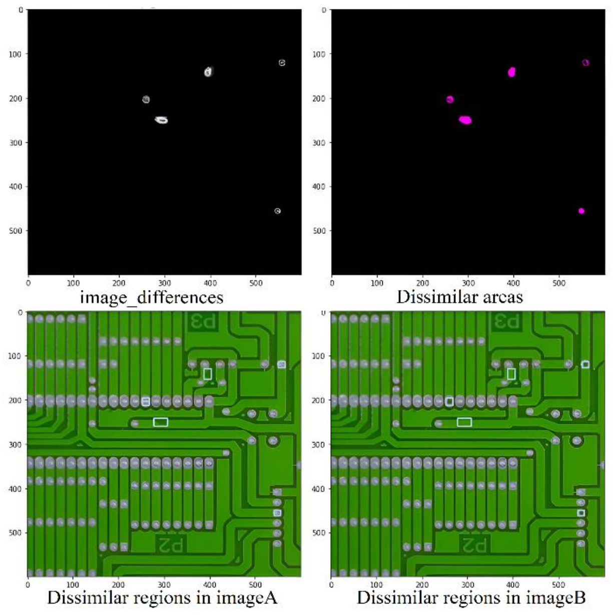


Fig. 19. Absolute differences.

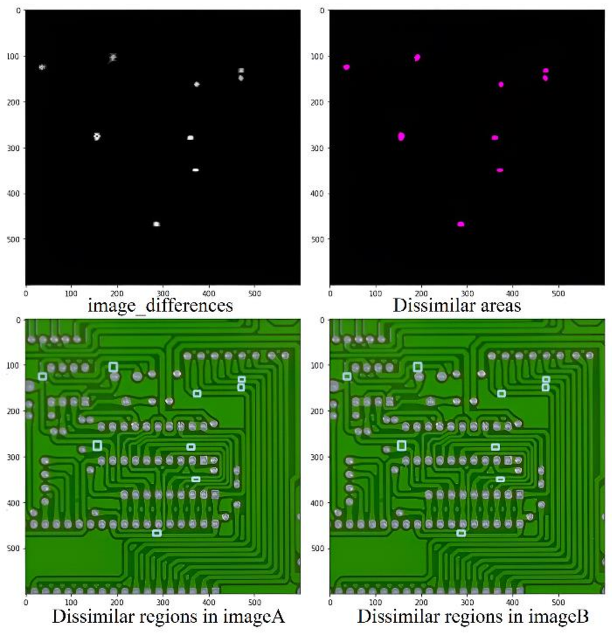
The Structure Similarity Index (SSI) Measure results are illustrated in Fig. 20 (a)–(c) for respective designs of PCBs, it is found that the Similarity Score recorded between the two images of each PCB kind is tabulated in Table IV. For an ideal case if two images are more alike and almost same the SSI turns to 100%.

TABLE IV. SSI VALUE BETWEEN IMAGES IN EACH PATTERN

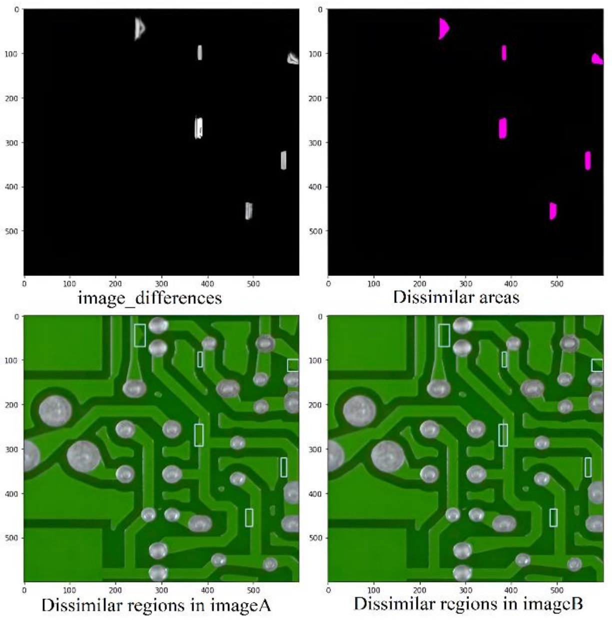
Class	Design-A		Design-B		Design-C	
Images	ImageA	ImageB	ImageA	ImageB	ImageA	ImageB
SSI	99.67%		99.75%		99.30%	



(a)



(b)



(c)

Fig. 20. Obtained SSI results; (a) SSI results for PCB Design-A; (b). SSI results for PCB Design-B; (c) SSI results for PCB Design-C.

Proposed Approach: The reason for using this segmentation-based multi-thresholding is to indicate more precisely the details (defects) in the images of each design.

After pre-processing the images, image smoothing and removing unwanted noises, re-execution of the individual histograms through multi-level thresholding is done, whose results are shown below in Fig. 21. Here, 'imageA' is a PCB with one defect, and 'imageB' is a PCB with certain another defect.

By looking at the histogram, one can figure out that each histogram has at most three peaks, which directs us the intensity variations of constituents in image and can be figured that Connecting Pads Intensity > Routing Channels Intensity > Background Intensity.

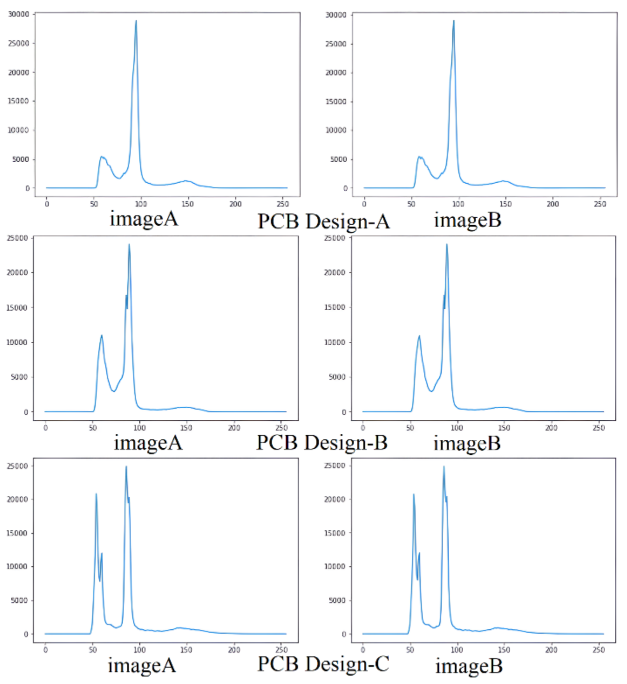


Fig. 21. Histograms of each design after preprocessing.

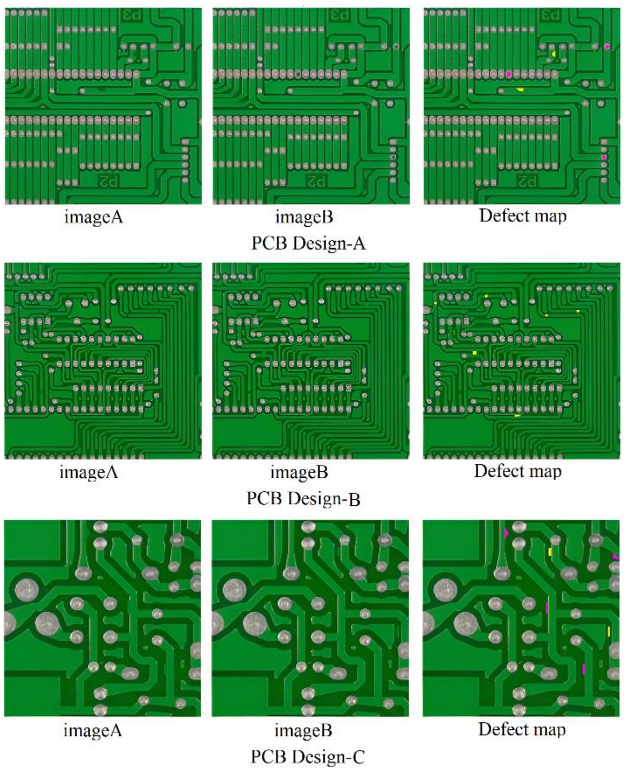


Fig. 22. Results of proposed approach.

With the subtraction operation, masks are created, and the images under consideration are enclosed by these masks using the windowing or masking technique. This involves mapping the window coordinates of the mask onto one of the scrutinized images, as depicted in Fig. 22. In the resultant image of Design-A PCB, the colors yellow and magenta indicate the locations of mouse bites (yellow) in image A and missing holes (magenta) in image B. The yellow color in the Design-B image represents open and short abnormalities. The magenta and yellow colors in the Design-C resultant image show the location of spur defects in Image A and spurious copper defects in Images B.

V. Conclusion

The present inspection work primarily focuses on identifying and correcting PCB images that are registered with a tilt using the Homography technique. The model is capable of correcting tilts ranging from 0 to 84 degrees, with a maximum execution time of 7.96 s for the considered images. Subsequently, these corrected images are processed by a pattern-matching unit where the unit receives images of the same PCB pattern with different defects in each. Structural information mapping is carried out using various spatial-domain feature-based matching algorithms. When using SSI and MSE metrics the model resulted in high match percentages of 99.67%, 99.75%, and 99.30%, and low error rates of 0.343%, 0.358%, and 0.721% for three distinct designs of PCB of two units each. When comparing the defect-free template with query image 1 and the next query image 2, it requires two system clock cycles. However, in this case, we have chosen to compare two query images with each other to identify differences, which consumes only 1 system clock cycle. Additionally, the proposed model utilizes a segmentation approach to detect and locate defects or dissimilarities in the PCB images without the need for bounding boxes, in accordance with the co-image description. From the experimentation, it is observed that the precision of abnormality detection is enhanced and the inspection runtime is decreased.

The paper primarily focuses on utilizing 2D image analysis with reference-based approaches for detecting defects in Printed Circuit Boards (PCBs). This approach can be applied in automated and intelligent quality control PCB inspections. Future research should focus on algorithm refinement to enhance the model's robustness against various deformations and environmental influences. This could be achieved by pre-training deep learning models with prior knowledge of various PCB layout defects. Ultimately, an intelligent defect detection system should be capable of scrutinizing defects in new PCB patterns without relying on reference comparisons, thereby increasing its adaptability and usability.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

A. Sainath Chaithanya performed data collection; designed and implemented the experiments and drafted the paper; L. Nirmala Devi conducted and supervised the research; all authors analyzed the results and approved the final version.

Acknowledgment

The authors would like to specially thank "The Open Lab on Human-Robot Interaction-Peking University" for PCB Defect Dataset.

REFERENCES

[1] S. Das. How to make single sided PCB. [Online]. Available: https://www.electronicsandyou.com/blog/single-sided-pcb.html

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. University of Tennessee, 2017.

[3] A. Sakila and S. Vijayarani, "Skew detection and correction in the document image," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 6, no. 8, 2017.

[4] K. Huang *et al*., "An ecient document skew detection method using probability model and Q test," *Electronics*, vol. 9, 2020.

[5] W. Tang and F. X. Jia, and X. M. Wang, "Image large rotation and scale estimation using the gabor filter," *Electronics*, vol. 11, 2022.

[6] J. Maniar, S. Patel, L. Shah, and R. Patel, "Rotation estimation of gujarati script document using hough transform," *Int. Journal of Engineering Research and Applications*, vol. 4, pp. 33–36, 2014.

[7] J. Fabrizio, "A precise skew estimation algorithm for document images using KNN clustering and fourier transform," in *Proc. 2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 2585–2588.

[8] F. Raihan and W. Ce, "PCB defect detection using opencv with image subtraction method," in *Proc. 2017 International Conference on Information Management and Technology (ICIMTech)*, 2017, pp. 204–209.

[9] N. Ravi, S. Naqvi, and M. E. Sharkawy, "BIoU: An improved bounding box regression for object detection," *J. Low Power Electron. Appl.*, 2022, vol. 12, no. 51, 2022.

[10] S. Geethapriya, K. Devaki, and V. M. Bhaskaran, "Multiple object detection in images using template matching," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 1, 2019.

[11] J. P. R. Nayaka *et al.*, "PCB fault detection using image processing," in *Proc. IOP Conf. Series: Materials Science and Engineering*, vol. 225, 2017.

[12] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through FSIM, SSIM, MSE and PSNR—A comparative study," *Journal of Computer and Communications*, vol. 7, pp. 8–18, 2019.

[13] R. Szeliski, "Computer vision: Algorithms and applications," *Texts in Computer Science*, pp. 1–26, 2010.

[14] B. Zitova and J. Flusser, "Image registration methods: A survey," *Image and Vision Computing*, vol. 21, 2003.

[15] PCB defect datasets. The Open Lab on Human Robot Interaction of Peking University's. [Online]. Available: https://robotics.pkusz.edu.cn/resources/datasetENG/

[16] Automatic skew correction using corner detectors and homography. [Online]. Available: https://github.com/ashuta03/automatic_skew_correction_using_corner_detectors_and_homography/tree/master

[17] Image processing 101 chapter 1.3: Color space conversion. [Online]. Available: https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-space-conversion/

[18] A. Mordvintsev, *OpenCV-Python Tutorials Documentation Release Beta*, 2017.

[19] P. J. Tejada, "A computational geometry approach to digital image Contour extraction," *Transactions on Computational Science*, pp 13–43, 2011.

[20] Wikipedia on ramer–douglas–peuckeralgorithm. [Online]. Available: https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

[21] S. Mallick. Feature based image alignment using OpenCV (C++/python. [Online]. Available: https://learnopencv.com/image-alignment-feature-based-using-opencv-c-python/

[22] R. Collins. Lecture 15—Robust estimation: RANSAC. [Online]. Available: https://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf

[23] U. Sara *et al*., "Image quality assessment through FSIM, SSIM, MSE and PSNR—A comparative study," *Journal of Computer and Communications*, vol. 7, pp. 8–18, 2019.

[24] A. Rosebrock. Image difference with OpenCV and python. [Online]. Available: https://pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/

[25] Kostasthanos. Spot-The-Differences. [Online]. Available: https://github.com/kostasthanos/Spot-The-Differences

[26] Drawing Bounding boxes, Annotating images using OpenCV. [Online]. Available: https://learnopencv.com/annotating-images-using-opencv/

[27] R. Holzer. (2020). OpenCV tutorial Documentation. [Online]. Available: https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html