Image Processing and Machine Learning for the Detection of Defects in PCB Images

ROMAN MELNYK, PAVLO VIAZOVSKYY
Software Department,
Lviv Polytechnic National University,
12 Stepan Bandera St., Lviv, 79013,
UKRAINE

Abstract: - The presented work considers three approaches to detecting defects in printed circuit boards. Each combines two components: algorithmic and software for image processing and machine learning based on image features identified in the first step. The approaches are as follows: division of boards into correct and defective boards without indicating the types of defects, determination of connection defects and redundant or missing components, and determination of defectiveness of tracks and contacts by analyzing individual components of the board. Color manipulation algorithms and additional image processing tools have been developed. Neural networks of artificial intelligence, which are available on the Internet, were used in the work.

Key Words: - Printed circuit board, chain, trace, defects, flood-filling, thinning, skeleton, tree, specific points, distributed cumulative histogram, overlay, Machine Learning.

Received: April 5, 2024. Revised: November 4, 2024. Accepted: December 7, 2024. Published: February 13, 2025.

1 Introduction

Most of the current publications devoted to the various problems of detecting defects in printed circuit boards are collected and described in fresh reviews. For example, the articles in [1], [2], [3], [4], and [5] contain references to many useful approaches, practical applications, dataset collections, original PCBs, cases of various defects, and many recommendations. This information is useful for researchers working in this direction. Therefore, they are often read and referred to.

When researchers work in a narrower field, they are interested in the individual publications noted in the surveys. In the paper [6] a generic deep-learning method and an active-learning method were proposed for the localization part and the classification part of solder joint defects in printed circuit boards (PCBs). The goal is to reduce the labeling workload when a large labeled training database is not easy. Edge detection using the Sobel operator to improve defect position is discussed in [7]. The following study in [8] proposes an algorithmic scheme for detecting and classifying many known types of PCB defects. The proposed algorithmic scheme contains fuzzy c-means clustering for image segmentation and image subtraction before defect detection. A cross-correlation pattern matching method was developed and described in [9] to meet detection requirements. The mentioned works extract defects from the difference between the gray pattern and the image samples.

Many researchers use traditional image processing algorithms based on subtraction algorithms, statistical comparison of shades of gray, and division of images into subblocks, for example, in [10], [11]. Image processing methods based on machine learning and MATLAB tools are considered in [12], [13], [14] to check defects in the manufacture of printed circuit boards.

Works that apply neural networks with different architectures to detect defects in printed circuit boards can be grouped into one large group. The paper [15] demonstrates a complete PCB defect classifier that automatically detects and classifies defects in PCBs. Machine learning is used in conjunction with a software application. A deep model detects PCB defects from an input pair of an undetected pattern and a defective tested image in paper [16]. 1500 image pairs with annotations and positions of 6 common PCB defect types are used to train the deep model.

The second group of works with artificial intelligence with a deep learning algorithm based on YOLO (you-only-look-once) is represented by publications [17], [18]. In the study [19], the authors have established a lightweight PCB-type detection model, which can perform real-time detection. In addition, in the character detection of PCB, they have established a fast and robust character recognition model.

Deep learning algorithms are widely used in related areas of research [20], [21], [22]. The paper [23] suggests a more 'intelligent' method than a traditional neural network. The image parts that it creates have more meaning and can also be put into a positional context and allow for an explainable result.

There are three options for detecting defects in the connection circuit covered by metal in the presented work. The simplest among them is the method of detecting defects in general, without specifying the types of defects and their localization. The second method indicates their types with or without their location. The most complex approach requires significant computing resources, classifies defects, and links them to specific tracks and contacts on the printed circuit board. Each of these approaches needs the implementation of both algorithmic support and machine learning tools.

For mass production, the advantage of the proposed method is the possibility of joining algorithms into three classes each of which is followed by machine learning software. No special PCB data sets are required.

2 Types of defects in PCB image

Various factors cause defects in printed circuit boards. They come in different types, sizes, and shapes. These characteristics, as well as their location, are unpredictable. They affect functionality and reliability over a wide range of ratings. Some can be tested by the user or computer software, others require electrical testing. All types and values of defects are random and do not match even on two printed circuit boards. Some examples of typical and critical defects are shown in Figure 1. Images are of different sizes and scales.

Defects of connectivity such as short and open are very critical. They destroy the functionality of a circuit. Two chains with short defects are marked with red in Figure 1(a). One chain with an open defect is marked with two colors. Various types of defects are shown in Figure 1 (b), (c). Among them extra and shortage of metal, short and open defects are available.

Because the types of defects and the questions that need to be answered are diverse, it is better to divide them into groups with common characteristics and apply different approaches to solving them. These approaches should combine methods and algorithms from image processing and machine learning. Algorithms of these approaches will not miss defects but indicate their location. Then Expert AI systems could make recommendations about what actions to take.

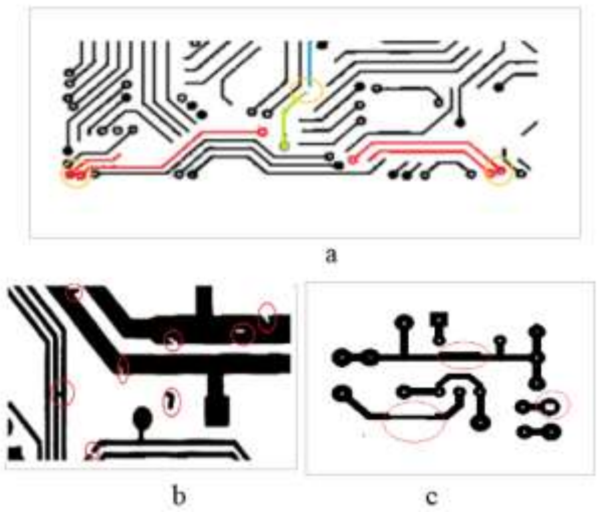


Fig. 1: Short (red) and open (blue, green) defects (a), missing and extra metal (b, c)

3 Detection of Defects of the Complete Circuit

3.1 Comparison of Samples

The subtraction operation realizes the following tasks: tracks and pins are on the board, defects are separated into two classes by the defect type, and defects are visible in their places near constructive elements.

The subtraction consists of the following operations:

0-1=*red*, 1-0=*blue*, 0-0=0, 1-1=1,

where 0 denotes white and 1 black or vice versa, *red* and *blue* are of the constant intensity values.

The printed circuit board produced is compared with the reference image. The resulting image shows excess and missing metal on the board, traces, and contacts. Red and blue pixels represent connected and isolated regions, including different forms of defects. The PCB image from [14] and subtraction operation results are shown in Figure 2.

In addition to real red and blue defects, the image in Figure 2 contains red and blue lines that reflect the displacement of individual tracks and contacts.

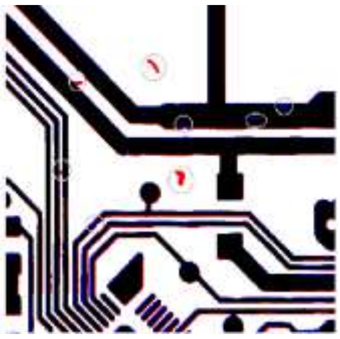


Fig. 2: The comparison difference between the two PCB images

3.2 Machine Learning for the Classification of Defective and Correct Printed Circuit Boards

In the following experiment, the normal and cumulative histograms for the resulting image of the sample difference were calculated by the following formulas:

$$h(j) = card\{(i,v)|I(i,v) \in I(0 \div 255)\},$$
$$H(j)=\Sigma h(k), k=0,\ldots,j,\ j = 0,1,..255.,$$

where *H(j)* is a cumulative histogram of the image, *h(j)* is the normal histogram, and I is the pixel intensity.

Their graphs are shown in Figure 3 in red. Figure 3 shows two superimposed images with slightly shifted scales and sizes of individual histograms. The image of the difference contains 32 percent black pixels, 3 percent red pixels, and 3 percent blue pixels. This means the defect area is about 6 percent of the image surface and 20 percent compared to the main tracks and pads. Analysis of the diagram demonstrates that the number of red and blue pixels can be used as input data in an artificial neural network.

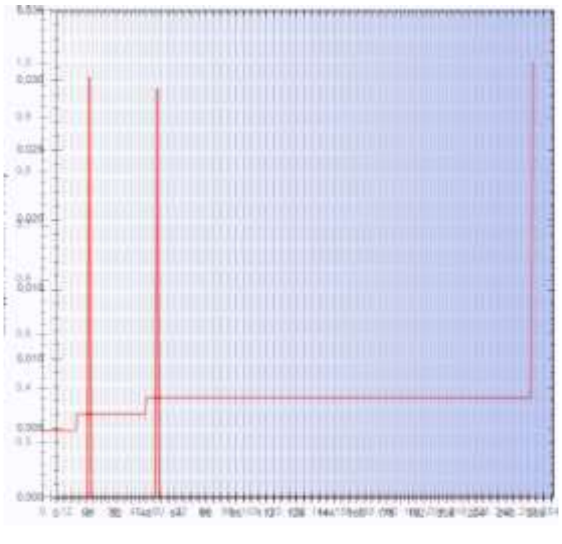


Fig. 3: Normal and cumulative histograms of the image-difference image

Modern software environments contain libraries with programmable neural networks, particularly, the Python language, which was successfully used in work [24]. More powerful architectures for broad Internet access are presented for experimentation. Among them are the libraries *TensorFlow* [25] and *Brain.js* [26] with forward neural network (FNN). JavaScript machine learning is effortless and does not require input of additional data such as a number of inputs, iterations, layers, accuracy, etc. If no parameters are specified, *Brain.js* will use the default required values.

The main object of machine learning is *NeuralNetwork*(), with two main methods: *train*() and *run*() which receive data and determine results. There are two ways to build a model and to make a decision: based on four two-component models or one four-component model. In the first case, each model is trained on input data for one defect and the correct pattern. In the second case, the model is trained with input data describing the three types of defects and the correct pattern.

The following values are selected for training the defective circuit network: the interval of values 0-0.1 for correct circuits–(0.1, 0.1, 0.1, 0.1), a value greater than 0.2 for defective circuits-(0.2, 0.2, 0.3, 0.3) as a percentage of blue and red.

Then the JavaScript code snippet looks like this:

```
// Create a Neural Network
const net = new brain.NeuralNetwork();
// Train the Network with 5 input objects
net.train([
 {input:[0.1, 0.1, 0.1, 0.1],output:{correct:1}},
{input:[0.2, 0.2, 0.0 ,0.0],output:{blue:1}},
{input:[0.0, 0.0, 0.4, 0.4],output:{red:1}},
{input:[0.2, 0.2, 0.3, 0.3],output:{blue_red:1}},
]);
// What is the expected output for the blue defect
// [0.25, 0.21, 0, 0]?
let result = network.run([0.25, 0.21, 0, 0]);
```

The classification result for the circuit with only blue defects *([0.25, 0.21, 0, 0])* is as follows:

blue defect: 0.968435525894165,
red defect: 0.00024306973500642926,
blue-red defects: 0.08996354788541794,
correct: 0.06718787550926208.

For the circuit with the blue and red inclusions (defects) ([0.25, 0.23, 0.35, 0.29]) the classification results are as follows:

blue:0.17240796983242035,
red:0.0213618241250515,
blue-red:0.9707114696502686,
correct: 0.00019091196008957922.

In conclusion, the developed software combined with learning classifies four types of PCB images: correct, blue, red, and blue-red defects. The training input data can be re-scaled for open and additional defects because their specific point coverage rates are close in the experiment. Only one PCB image participated in the classification.

4 Detection of Connectivity Defects, Missing and Extra Elements

4.1 Thinning and Flood-Filling of Elements

For maximal simplification, the process of clarifying the type of defects and their location, instead of the reference image of the printed circuit board, its topological image in the form of a skeleton is used.

Identifying the objects is possible in the PCB image with the help of a sequence of algorithms: thinning, finding specific points in the skeleton, and filling trees in the skeleton. The specific points have coordinates of the pixels belonging to traces and pads. The filling process links them together and simultaneously assigns them the identification numbers (ID). The resulting output images received by these algorithms for the PCB fragment are shown in Figure 4, the input image in Figure 4(a), skeletons in Figure 4 (b), (c).

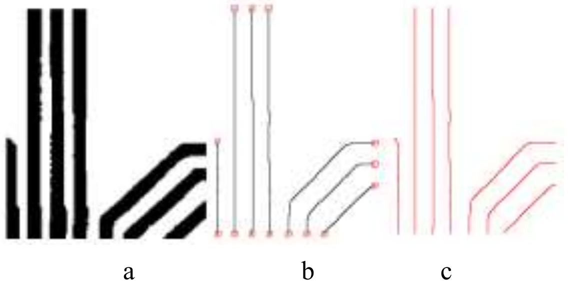


Fig. 4: A PCB fragment (a), its skeleton with endings (b) and filled skeleton (c)

After the thinning algorithm (Figure 4(b)), the specific points in the set $P(p_1, p_2,\dots,p_N)$ are characterized only by their coordinates. Their trees and chains do not have identification numbers. The flood-filling algorithm divides the set P into subsets of points belonging to only one tree (chain):

$$P(p_1, p_2,\dots,p_N)=P(P_1,P_2,\dots,P_n).$$

where $p_1, p_2,\dots,p_N$ are the coordinates of N specific points, $P_1, P_2,\dots, P_n$ are n sets of points assigned to the trees (chains), and indices of the trees are IDs.

The assigning of IDs is realized on the set of specific points $P(p_1, p_2,\dots,p_N)$. Then the first arbitrary tree of the skeleton is chosen and filled with red. This symbolically means that pixels with the chosen coordinates changed color to red, for example, p_1, p_4., and their tree obtains ID=1. Then the new operating space for the filling algorithm became $P/(p_1, p_4.)$ and the next points became red (for example, p_2, p_6.). These points and their tree obtain ID=2. The algorithm terminates when all points are red and their trees are with IDs.

The red skeleton after filling with red and identification is shown in Figure 4(c).

4.2 Overlaying of Images with Skeleton

The skeleton and the sample image are superimposed with a transparency factor of α=0.25. Twenty-five percent for the black chains and 75 percent for the skeleton. The resulting image is shown in Figure 5(a).

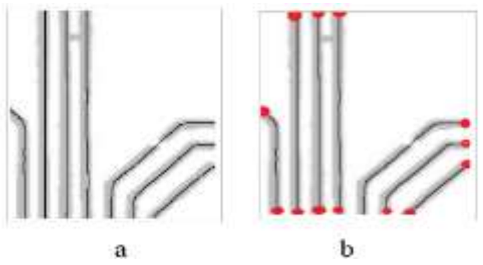


Fig. 5: Overlay of the fragment with defects and correct skeleton (a), and with endings (b)

The image of the superimposed skeleton with encircled endings and a fragment with defects is shown in Figure 5(b) on the surfaces of gray chains.

The overlaying experiment with α=0.25 is held with the following pixels:

$$W=0.25white(1)+0.75white(2)= white,$$
$$LG=0.25black(1)+0.75white(2)=light\ gray,$$
$$DG=0.25white(1)+0.75black(2)=dark\ gray,$$
$$B=0.25black(1) + 0.75black(2)= black.$$

Four colors are presented in the resulting image in Figure 5(a). One of them, dark gray indicates the incorrect element, in this case missing metal.

4.3 Detection of Defects

For an illustration of the approach, an example with wider traces missed and extra components is used. It is shown in Figure 6. To highlight the lines of the skeleton, its trees are filled with red. The resulting image in Figure 6(c) (after overlaying) contains all the necessary information for the detection of four types of defects: short, open, missing, and extra metal. Some simple algorithms are developed to process this image and obtain features characterizing these defects.

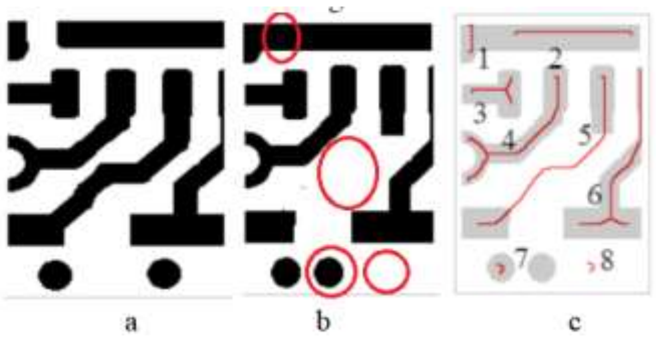


Fig. 6: The reference fragment (a), sample with four defects (b), overlay with skeleton and IDs (c)

For a short defect, in Figure 6(c) two trees are located in the gray body of the defective chain. Their ID numbers are ID=1 and ID=2. On the contrary, if the gray body does not contain the full *i*-th set P_i describing the *i*-th tree, an open defect is available. The gray body without any tree or its part defines an extra metal. A tree on the white background defines a missing metal.

To detect these defects, the overlay image in Figure 6(c) is analyzed. Each chain is separated and processed to determine how many tree inclusions it contains.

The K-means clustering algorithm can be applied to find and separate analyzed objects such as chains, wires, and encircled points. The simpler approach contains the flood-filling algorithm, calculation, and comparison operations. The first step separates the gray chains by flood-filling them from the starting points in the set $P(P_1, P_2, \ldots, P_n)$. This step is illustrated by an example of three separated chains (marked with different colors) in Figure 7(a).

Each tree on the skeleton is described by n_i specific points from the set P_i. These points are distributed on the gray component of the sample image. The number of points included in the gray chain can be different and belong to different trees. It is shown in Figure 7. In Figure 7(b) circles with the same color are surrounding points of one tree, two trees, and a half tree.

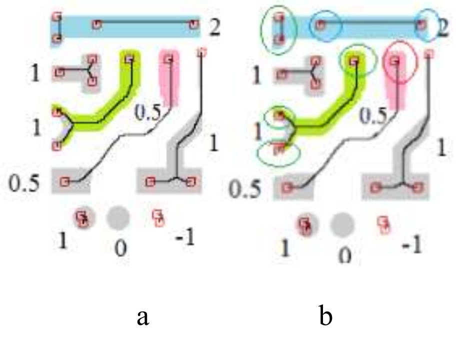


Fig. 7: Three separated (filled) chains with trees (a), encircled groups of endings (b)

Then the filled components are checked whether they include the full set P_i of the *i*-th tree, a part of the set, or a few sets. The number of so-called included trees indicates the type of defect. This number is found after interception of all coordinates of specific points from the set *P* with the separated gray chain, for example, filled with green (*g*):

$$K(g)= \Sigma\ |P_i \cap P(\text{chain}(g))|,\ \text{for}\ i=1,\ldots, N,\ P_i \in P.$$

The rules to calculate the features are following: 1) if the chain contains three points fully describing one tree its feature is 1 (correct); 2) for one point from two available, the feature is 0.5 (open); 3) if the gray chain contains two trees, the feature is 2 (short); 4) if the gray chain does not contain any tree the value of 0 (extra); 5) for the empty position the accepted value is -1 (lack). An example of the chain feature values is shown in Figure 7. If defects are repeated, the summary estimation is averaged for the same type of defects.

4.4 Application of Machine Learning

Machine Learning is used to classify printed circuit boards that contain connectivity or metal coating defects. For faulty and correct chains, five features are selected for training as follows:
Correct circuits–1, short defect–2, open defect–0.5, missing defect–(-1), excess defect–0. They form the input for learning.

Then the JavaScript code snippet looks like this:

```
// Create a Neural Network
const net = new brain.NeuralNetwork();
// Train the Network with 5 input objects
net.train([
{input:[2, 2.1, 2.5, 2.5],output:{short:1}},
{input:[0.2, 0.3, 0.4, 0.5],output:{open:1}},
{input:[0, 0, 0, 0],output:{extra:1}},
{input:[-1, -2, -1 ,-2],output:{lack:1}},
{input:[1, 1, 1, 1], output:{correct:1}},
]);
// What is the expected output of extra (0,2)?
let result=net.run([0.1, 0, 0, 0]);
```

The classification result is as follows:
Correct 0.000004523093593888916,
Open: 0.1014263927936554,
Lack: 0.03423898667097092,
Extra: 0.8860721588134766,
Short: 0.000013810598829877563.

For the correct circuit, the classification result is as follows:
let result=net.run([1.1, 1.2, 1, 1]);
Correct: 0.7048931121826172,
Open: 0.016741283237934113,

Short: 0.17118282616138458,
Lack: 0.00018151775293517858,
Extra: 0.000518892309628427.

In conclusion, the developed software combined with machine learning classifies four types of PCB defects and images. This simple experiment confirms that circuit features indicating the chains with connection defects and missing (extra) metal along with Machine Learning tools, allow us to classify PCB images into defective and correct samples without involving the user in the controlling process. Also, types of defects are output data.

5 Detection of Defects in Chains

5.1 Detection of Defects

The input data for the third stage of defects detection in printed circuit boards is a model with excluded circuits having short-circuit, open, extra, and shortage of metal defects, i.e.:

$$M_r(\text{pb})=M_o(\text{pb})-C_{cd}(\text{pb}),$$

where M_r(pb) is the reduced model, M_o(pb) is the original model, and C_{cd}(pb) is the set of chains with connection and coverage violation defects.

Consequently, each chain is selected and separated by the flood-filling algorithm based on the known ID numbers of the chain and the coordinates of specific points. As a rule, printed circuit boards are large with many long structural elements. That is why their fragments are taken as examples to save the volume of the article. Two of them are shown in Figure 8 where the images immediately present the comparison results (difference) of the reference and defective tracks.

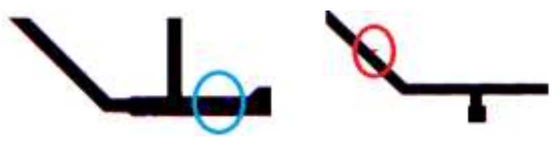


Fig. 8: PCB image fragments with shortage defects (a), and with extra defects (b)

The images in Figure 8 also contain blue and red areas representing shortage and extra material defects. The same colored lines represent the shift of the traces compared to the reference samples. Measurement of these defects and trace bodies by calculating cumulative histograms gives the results shown in Figure 9.

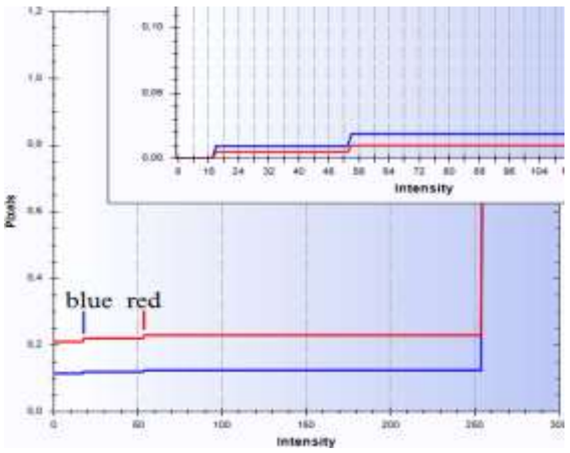


Fig. 9: Cumulative histograms of two PCB chains with marked defects

The value of the number of blue and red pixels is small compared to the black pixels of trace bodies. The difference in the image contains almost 32 percent black pixels, 3 percent red, and 3 percent blue pixels. This means the defect area is about 6 percent of the image surface and 20 percent compared to the main tracks and pads.

Segmented blue and red pixels in Figure 10 are a better presentation of defects for their measurement. However, the values are still small, as shown in Figure 9 (for a larger scale). Also, blue and red pixels can be caused by defects of uneven width of traces and displacement of contacts and traces. Therefore, the levels of blue and red colors cannot be accepted for detecting defects in trace bodies.

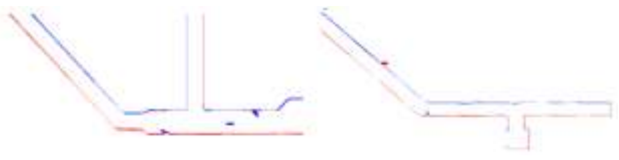


Fig. 10: Segmented blue and red pixels: withshortage defects (a), and with extra defects (b)

5.2 Amplifying Visibility of Defects

While defects of shifts are distributed along the OX and OY axes it is better to apply the mechanism of the distributed cumulative histogram. It reflects pixels along every row or column of the image matrix. The distributed histogram shows the frequency of pixel intensity values in rows Vj(r) and columns Vi(c). In the image histograms, the OX axis shows the gray level intensities in N columns (M rows) and the OY axis shows the frequency of these intensities.

Then the cumulative histograms are calculated for all rows (columns):

$$M_{ik} = \sum_{j=1}^{k} m_{ij},\ i = \overline{1, B},\ k = \overline{0, 255},$$

$$M_{kj} = \sum_{i=1}^{k} m_{ij},\ j = \overline{1, A},\ k = \overline{0, 255},$$

where m_i, m_j represent the values of the intensity histograms, M_i, M_j are the values of cumulative histograms called distributed cumulative histograms.

The following step is to draw an image of the distributed cumulative histogram (DCH). The value of each pixel of this image corresponds to a value of the cumulative histogram. The new image will have $B \times N$ pixel size (for rows) with the pixel values are calculated by the following formulas:

For columns, the new image is of $N \times A$ size and the pixel values are calculated by the similar formulas:

$$R'_{ij} = G'_{ij} = B'_{ij} = M_{ij} * \frac{255}{\max\limits_{1 \le i \le N} M_j},\ i = \overline{1, N},\ j = \overline{1, A},$$

where R,G,B are the pixel components in the DCH image, M_i is the cumulative histogram of i-row, M_{ij} is the value of j-element of the histogram, N is the number of elements in the cumulative histogram.

Two images of distributed cumulative histograms of two images of chains from Figure 9 are shown in Figure 11. The first is for the normal first trace, and the second is for the inverted second trace. The number of informative pixels is small. Therefore, dark (in the first case of a white background) and white (in the second case of a dark background) dominate all information pixels.

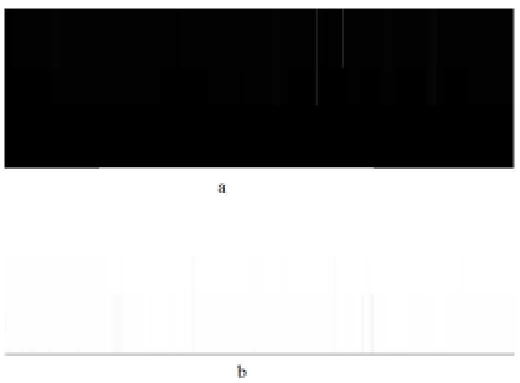


Fig. 11: Distributed cumulative histograms of traces: original first (a), inverted second (b)

To make histograms more presentable, black pixels in the first case and white pixels in the second are filled with gray. Then Figure 11 is transformed into Figure 12. All other pixels have values other than zero and 255.

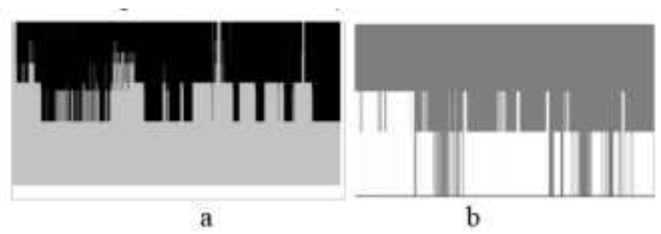


Fig. 12: Cumulative histograms of two chains with defects: first chain (a) and second chain(b)

When similar distributed cumulative histograms are constructed for reference circuits, they can be subtracted from the correspondent histograms of defective circuits:

$$I_r(x,y) = I_s(x_i,y_i) - I_{rs}(x_i,y_i),$$

If a difference $I_r(x,y)$ is greater than the given tolerance Tol the correspondent pixel changes its color to red or blue depending on the type of defects:

if $I_r(x,y) > Tol$, $I_r(x,y) = I(\text{red})$.

In the other case, pixels remain with the original color. Two image differences of two input traces with blue and red defects are shown in Figure 13.

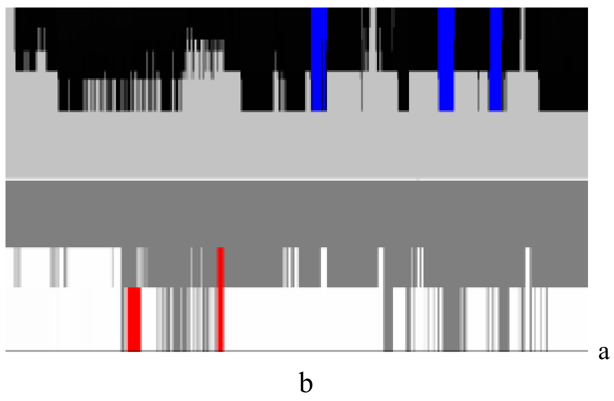


Fig. 13: Differences of distributed cumulative histograms: with blue defects (a) and red defects (b)

Figure 14 presents a view of the differences between untransformed images. Red and blue strips are also available.

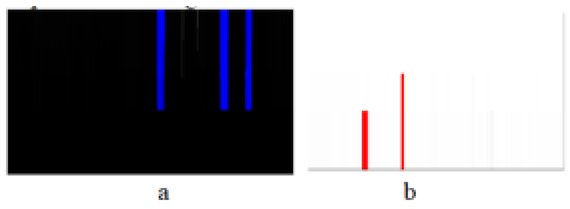


Fig. 14: Original differences of distributed cumulative histograms: with blue defects (a) and red defects (b)

An overlay of the track and the image difference is used to visualize the coordinates and locations of defects on the track. The following formula is applied to each pixel channel in both images:

$$I_r(\mathrm{x,y}) = \alpha \mathrm{I}_s(x_i, \mathrm{y}_i) + (1-\alpha) I_{\mathrm{rs}}(x_i, \mathrm{y}_i),$$

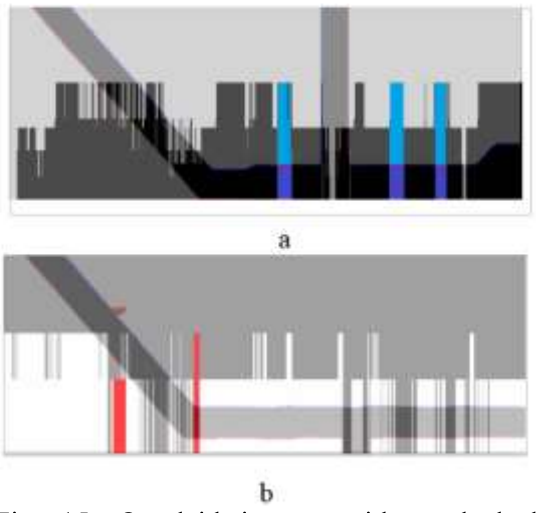
where I_{r}, and Is are the pixel intensity of the resulting and sample image, I_{rs}, is the pixel intensity of the image difference of distributed cumulative histograms, and α is the coefficient of transparency (independent variable).

The image of the trace has two colors (black and white) and the image-difference 3 colors (gray, dark, and red or blue). So, the overlaid image has more than six colors, obtained from all combinations. For example:

$$LG = 0.3\mathrm{white}(1) + 0.7\mathrm{grey}(2),$$
$$DG = 0.3\mathrm{black}(1) + 0.7\mathrm{dark}(2),$$
$$LR = 0.3\mathrm{white}(1) + 0.7\mathrm{red}(2),$$
$$DR = 0.3\mathrm{black}(1) + 0.7\mathrm{red}(2), \text{ and so on,}$$

where (1), and (2) denote the input images, and *LG*, *DG*, *LR*, and *DR* denote light and dark gray, and red.

For defects available in the trace, the two resulting images contain information about the OX coordinates of the defects, their types, and the defects themselves. Two examples of such images are shown in Figure 15.


a

b

Fig. 15: Overlaid images with marked defects: shortage defects (a), extra defects (b)

In the case of no offsets and different trace widths, the distributed cumulative histogram is simpler and does not contain pixels that reflect these inaccuracies. An example of such a case is shown in Figure 16. Vertical lines of small intensities, which are responsible for a small number of pixels, are absent.

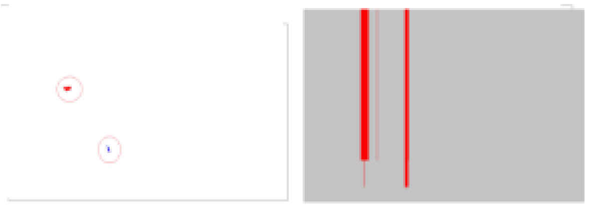

a b

Fig. 16: Two defects (a) and their difference in distributed cumulative histograms (b)

Finally, distributed cumulative histograms track small defects on the board and mark them with rectangles large enough to be visible on the board image. The color of the rectangle reflects the type of violation of drawing correctness. It is red for additional and blue for lack of metal defects.

5.3 Data for Application of Machine Learning

The proposed approach uses features extracted from comparison images for the chain classification. The rate of blue and red colors in the DCH comparison images is almost 4 times greater than the same percentage in the difference between the reference and sample images. The following values are selected for training: values 0-0.1 for correct circuits–(0.1, 0.1, 0.1, 0.1), and the interval of values 0.15-0.45 for faulty circuits-(0.2, 0.25, 0.35, 0.3). These values represent the percentage of blue (first and second) and red (third and fourth).

Then the JavaScript code snippet looks like this:

```
 // Create a Neural Network
const net = new brain.NeuralNetwork();
// Train the Network with 4 input objects
net.train([
 {input:[0.1, 0.1, 0, 0],output:{correct:1}},
{input:[0.2, 0.2, 0.0, 0.0],output:{blue:1}},
{input:[0.0, 0.0,  0.4, 0.4],output:{red:1}},
{input:[0.2, 0.2, 0.4, 0.4],output:{blue_red:1}},
]);
// What is the expected output for the red defect
// [0.0, 0.0, 0.4, 0.35]?
let result = network.run([0.0, 0.0, 0.4, 0.35]);
```

The classification result for the circuit with only red defects *([0.0, 0.0, 0.4, 0.35])* is as follows:

```
Blue: 0.00009080704330699518,
red: 0.9402382969856262,
blue-red: 0.04893253371119499,
correct: 0.07145267724990845.
```

For the circuit with the blue and red inclusions (defects) ([0.25, 0.28, 0.4, 0.35]) the classification results are as follows:
Blue: 0.07290368527173996,
red: 0.04140195995569229,
blue-red: 0.9641250371932983,
correct: 0.0000647743290755897 8.

In conclusion, software developed for circuit defect analysis combined with machine learning classifies three types of defects in PCB traces (plus correct): metal excess and metal deficiency, and both at the same time. In practice, the input data for training can be increased by accumulating data from previous processing experiments with prepared test samples.

6 Experiments

The developed algorithms were tested on the fragment of the printed circuit board with connection defects and a lack of metal. The reference image is shown in Figure 17.

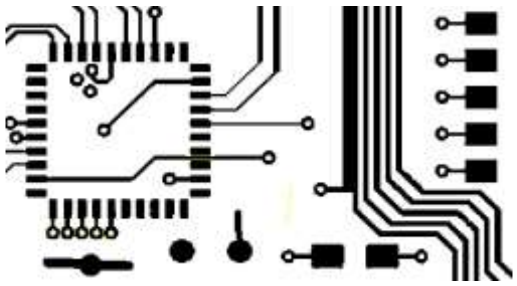


Fig. 17: Reference image without defects

Encircled defects are shown in Figure 18.

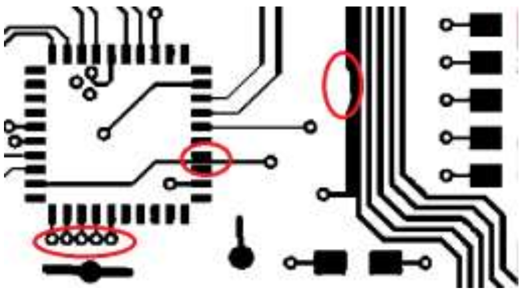


Fig. 18: PCB image with encircled defects

Thus, after thinning, flood-filling, and numbering the corresponding images are overlaid. Chains with short defects are highlighted. They are shown in Figure 19. The thinner trace and missing element are detected and marked in Figure 20.

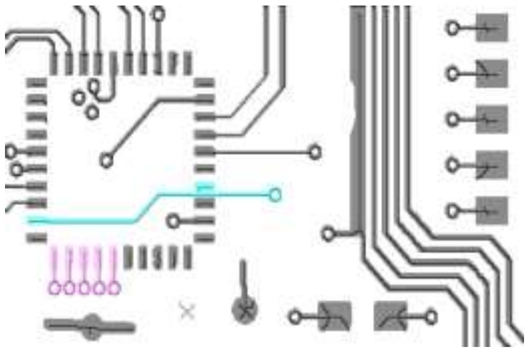


Fig. 19: Two short defects in traces

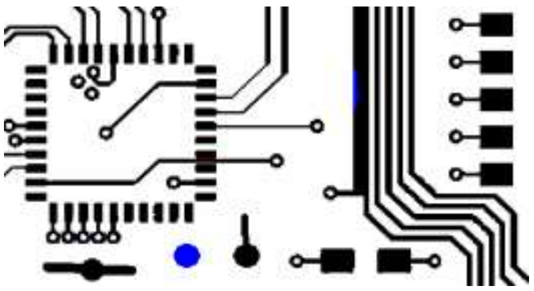


Fig. 20: Two defects of missing metal

In the experiment, the Brain.js neural network signaled the presence of defects three times: after comparing and inspecting the full images, analyzing the connection and the presence of metal, and separating the chains to examine them with distributed cumulative histograms.

7 Conclusion

Image processing algorithms such as thinning, filling, and overlaying together with the calculation of cumulative histograms are used to highlight and mark different types of defects: short, open, missing, and extra metal. The features obtained after image processing form the input data for training and testing the Brain.js neural network to classify PCBs at three different stages of defect detailing.

The simplest is the first step, which determines whether the manufactured boards are defective without specifying the type and location of the defects. At the second stage, the causes of defects are indicated, if they are caused by a short circuit, open circuit, excess or lack of metal. The third step answers, are there defects in individual tracks or pads?

In the first step, the color intensity of the defects is taken as input for machine learning represented by an artificial neural network. Additional characteristics of the PCB components, such as the number of included trees, are used as input data to train and test the artificial neural networks in the second step. In the third step, the color intensities of the defects in the auxiliary histogram images and their differences are accepted for machine learning to determine whether the traces are correct or defective with the specified defect type.

These simple experiments confirm that circuit colors and included defect-indicating trees, together with machine learning tools, enable the classification of PCB images into defective and correct samples without user involvement in the control process.

The features found are input to machine learning algorithms that can be applied to large sets of fabricated PCBs. A multi-layered ANN program for learning and recognizing PCB defect images is based on the powerful Brain.js tool, which has many control inputs for architecture selection, accuracy, loss, and processing time. This ANN can successfully classify printed circuit board images where color range and intensity are input data.

The approach involves three steps, which for large PCBs will be applied separately until one type of defect is detected. The software requires more calculation operations and can be used for boards with suspected defects.

Thus, the approach combines two mechanisms: processing by developed software and machine learning using data transmitted from the program. This reduces software development time and relieves the user from the monotonous quality control process.

The developed software tools are experimental and are intended to demonstrate the idea of combining algorithms and machine learning. They are difficult to apply to most of the images of printed circuit boards presented on the Internet. Since they are commercial with inscriptions, and additional components, and photographed for another purpose.

In the production process, the software complex receives pre-processed input images and in this case, the proposed algorithms will be useful. Therefore, further research is being conducted in the direction of reducing sensitivity to image types and universalizing machine learning not only with numerical but also with graphic data.

References:

[1] Qin Ling, Nor Ashidi Mat Isa, Printed Circuit Board Defect Detection Methods Based on Image Processing, Machine Learning and Deep Learning. A Survey, *IEEE Access*, Vol. 11, December 2022, pp. 1-24, DOI: 10.1109/ACCESS.2023.3245093.

[2] Elena Jasiuniene, Renaldas Raišutis, Vykintas Samaitis and Audrius Jankauskas. Comparison of Different NDT Techniques for Evaluation of the Quality of PCBs Produced Using Traditional vs. Additive Manufacturing Technologies. A survey, *Sensors,* 24, 2024, p. 1719, DOI: 10.3390/s24061719.

[3] Ebayyeh A.A.R.M.A., Mousavi, A., A Review and Analysis of Automatic Optical Inspection and Quality Monitoring Methods in Electronics Industry. *IEEE Access,* Vol. *8*, 2020, pp. 183192–183271, DOI: 10.1109/ACCESS.2020.3029127.

[4] K. P. Anoop, N.S. Sarath and V. V. Sasi Kumar, Review of PCB Defect Detection Using Image Processing, *International Journal of Engineering and Innovative technology (IJEIT),* Vol. 4, Is. 11, 2015, pp. 188-192.

[5] D.B. Anitha, and M. Rao, A survey on Defect Detection in Bare PCB and Assembled PCB using Image Processing Techniques, *International Conference on Wireless Communications, Signal Processing and Networking*, Chennai, India, 2017, pp. 39-43, DOI: 10.1109/WiSPNET.2017.8299715.

[6] W. Dai, A. Mujeeb, M. Erdt, and A. Sourin, “Soldering defect detection in automatic optical inspection,” *Advanced Engineering Informatics*, Vol. 43, 2020, p. 101004, DOI: 10.1016/j.aei.2019.101004.

[7] Cai L., Li J., PCB defect detection system based on image processing, *Journal of Physics: Conference Series*, Vol. 2383, No. 1, 2022, pp. 012077, IOP Publishing, DOI: 10.1088/1742-6596/2383/1/012077.

[8] O. Jakkrit and S. Jakkree, Algorithmic scheme for concurrent detection and classification of printed circuit board defects, *Computers, Materials & Continua*, Vol. 71, No. 1, 2022, pp. 355–367, DOI: 10.32604/cmc.2022.017698.

[9] M. H. Annaby, Y. M. Fouda, and M. A. Rushdi, “Improved normalized cross-correlation for defect detection in printed-circuit boards,” *IEEE Transactions on Semiconductor Manufacturing*, Vol. 32, No. 2, 2019, pp. 199–211, DOI: 10.1109/TSM.2019.2911062.

[10] Vikas Chaudhary, Ishan R. Dave and Kishor P. Upla, S. V., Visual Inspection of Printed Circuit Board for Defect Detection and Classification. *International Conference on*

Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 22-24 March, 2017, pp. 732-737, DOI**:** 10.1109/WiSPNET.2017.8299858.

[11] F. B. Nadaf and V. S. Kolkure, Detection of Bare PCB Defects by using Morphology Technique, *Morphology Technique International Journal of Electronics and Communication Engineering,* Vol. 9, No. 1, 2016, pp. 63-76. DOI: 10.23919/ChiCC.2017.8029117.

[12] J.Nayaka, K. Anitha, B.D.Parameshachari, R.Banud and P.Rashmi, PCB Fault Detection Using Image Processing, *IOP Conference Series: Materials Science and Engineering*, Vol. 225, 2017, pp. 1-5, DOI:10.1088/1757-899X/225/1/012244.

[13] Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., & Gupta, S. K. Image-based surface defect detection using deep learning: A review. *Journal of Computing and Information Science in Engineering*, August 2021, Vol. 21, Iss. 4, pp. 1-23. DOI: 10.1115/1.4049535.

[14] S. McClure, Extracting and Classifying Circuit Board Defects using Image Processing and Deep Learning, Feb. 4, 2020, [Online], https://towardsdatascience.com/building-an-end-to-end-deep-learning-defect-classifier-application-for-printed-circuit-board-pcb-6361b3a76232 (Accessed Date: December 13, 2024).

[15] Sanli Tang, Fan He, Xiaolin Huang, Jie Yang, Online PCB Defect Detector On A New PCB Defect Dataset. arXiv: Computer Vision and Pattern Recognition, [Online], https://arxiv.org/abs/1902.06197 (Accessed Date: December 13, 2024).

[16] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, Defect Detection in Printed Circuit Boards Using You-Only-Look-Once Convolutional Neural Networks, *Electronics*, Vol. 9, No. 9, 2020, p. 1547. DOI: 10.3390/electronics9091547.

[17] Abhiroop Bhattacharya, Sylvain G. Cloutier, End-to-end deep learning framework for printed circuit board manufacturing defect classification, *Scientific Reports*, 2022, [Online]. https://www.nature.com/articles/s41598-022-16302-3 (Accessed Date: December 13, 2024).

[18] J. Shen, N. Liu, and H. Sun, "Defect detection of printed circuit board based on lightweight deep convolution network," *IET Image Processing*, Vol. 14, 2020, pp. 3932-3940, DOI:10.1049/iet-ipr.2020.0841.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020*, pp. 213–229, DOI: 10.48550/arXiv.2005.12872.

[20] C. L. Perera, S. C. Premaratne, An Ensemble Machine Learning Approach for Forecasting Credit risk of Loan Applications, *WSEAS Transactions on Systems,* Vol. 23, 2024, pp. 31-46, https://doi.org/10.37394/232015.2024.20.88.

[21] Atul Kachare, Mukesh Kalla, Ashutosh Gupta, Visual Question Generation Answering (VQG-VQA) using Machine Learning, *WSEAS Transactions on Systems*, Vol. 22, 2023, pp. 663-670. https://doi.org/10.37394/23202.2023.22.67.

[22] Zurab Bosikashvili, Giorgi Kvartskhava, Using Hybrid Models of AI for Identification of Trees by UAV Images of Forests: I. Machine-learning Component of the Models, *WSEAS Transactions on Signal Processing*, Vol. 20, 2024, Art. #5, pp. 39-53, https://doi.org/10.37394/232014.2024.20.5.

[23] Kieran Greer, Recognising Image Shapes from Image Parts, not Neural Parts, *WSEAS Transactions on Signal Processing*, Vol. 19, 2023, Art. #9, pp. 77-82, https://doi.org/10.37394/232014.2023.19.9.

[24] Melnyk Roman, Vorobii Vitalii, PCB Image Defects Detection by Artificial Neural Networks and Resistance Analysis. *WSEAS Transactions on Circuit and Theory*, 22(1), 2023, pp. 35–42, https://doi.org/10.37394/23201.2024.23.7.

[25] TendorFlow. An end-to-end platform for machine learning, [Online], https://www.tensorflow.org (Accessed Date: December 13, 2024).

[26] Machine Learning in JavaScript, [Online], https://www.w3schools.com/ai/tryit.asp?filename=tryai_brain_contrast (Accessed Date: December 13, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors contributed in the ratio of 70 (Melnyk) to 30 (Viazovskyy) in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.