

Contacts Detection and Machine Learning for Classification of Defects in PCB Images

ROMAN MELNYK, OLEH IVANIUK

Software Department,
Lviv Polytechnic National University,
Lviv,
UKRAINE

Abstract: - The presented work considers three approaches for detecting the coordinates of contacts in printed circuit boards (PCBs). The objects of the approaches are contacts with holes, contacts without holes, and mixed contacts. The central procedure of the method is the flood-filling algorithm, which detects the component edges simultaneously. Overlaying the found coordinates of contacts with the standard PCB creates an image, which allows the determination of features characterizing defects, such as connectivity, redundant or missing components. Machine learning based on these identified image features is applied to classify printed circuit boards. Artificial neural networks (ANNs), which are available on the Internet, were used in the work. The developed approach combines two components: software for image processing and Machine Learning.

Key-Words: - printed circuit board, defects, flood-filling, classification, neural network, machine learning.

Received: March 8, 2026. Revised: June 9, 2026. Accepted: July 11, 2026. Available online: October 30, 2026.

1 Introduction

The number of publications on detecting defects in printed circuit boards is substantial, and most of them are collected in surveys. The review in [1] focuses on methods, while [2] focuses more on technologies. The survey [3] contains references to publications paying attention to Machine Learning and Deep Learning. The survey [4] contains works considering the application of Machine Learning for the detection of anomalies on surfaces.

The work [5] presents a segmentation-based deep-learning architecture for the detection and segmentation of surface anomalies, combining image processing and Machine Learning. In the work [6], due to the lack of available data, the technique extends the network knowledge with regard to defect classes with a few samples. Instead of thousands of images of defects as in [7], an unsupervised deep learning model is proposed using a smaller amount of representative defect-free samples (tens rather than thousands). In the work [8], a model uses nominal (non-defective) example images only. The work [9] describes artificial visual processing techniques to present the captured scenery in a mathematical/logical way. In [10] anomaly detection is based on the idea that the entropy of the latent distribution for normal data should be lower than the entropy of the anomalous distribution.

The works [11], [12] contain image processing algorithms, including subtraction algorithms. For inspection of PCB production, machine learning-based methods are widely used. A large part of these publications is in the survey [13]. A lot of publications consider Neural networks of various architectures and the approaches on their basis. In the article [14], a developed Neural network is applied to flexible PCBs. Machine learning exists in combination with a full application, as demonstrated in [15]. In the works [16], [17], a deep learning algorithm based on the you-only-look-once (YOLO) approach is proposed. Convolutional neural networks (CNNs) were considered in the publication [18]. Machine learning algorithms are also used to process data and predict results in many other technical applications [19], [20].

2 Types of defects in PCB images

Contacts are very important elements of PCBs. They are the end and intermediate points for all traces. Many PCB defects are caused by broken connections between traces and contacts, or within traces. The density of placement of elements is high, and the distances between tracks and contacts are small. There is a high probability of touching (short circuit) and breaking traces.

These types of defects can be called connection defects. Another group includes defects associated

with the absence of design details and the presence of unplanned parts of the metal coating. This group can be called coating defects. The third group combines various inaccuracies in the shape and size of the traces or contacts themselves. This group can be called filling defects.

Representatives of these three groups are shown in Figure 1 and are circled in different colors: red-connection defects; blue- coating defects; green-filling defects.

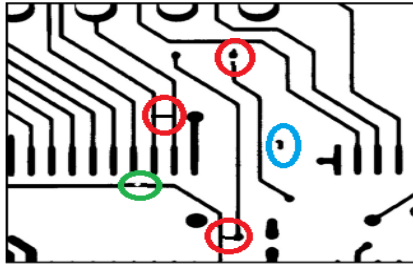


Fig. 1: A fragment of PCB with defects
Source: created by the authors

The task “Determination of defects in printed circuit board images” is generalizing and involves formulating a sequence of simpler tasks: “Determination of the presence of any defect”, “Determination of the presence of a defect of a specific type”, “Determination of the localization of a defect of a specific type”, etc. In some cases, solving the next task requires solving the previous ones.

In the presented article, the approach for the detection of defects from the first two groups is considered. For better understanding, it is necessary to recall that all contacts are divided into groups belonging to different chains. For every PCB there is a set of chains $C=\{c_1, c_2, \dots, c_N\}$ of N chains to be connected, where c_j is a chain's identifier, a set of contacts $K=\{K_1, K_2, \dots, K_N\}$, where every element K_j is a set of contacts $K_j=\{k_{j1}, k_{j2}, \dots, k_{jkj}\}$, belonging to the j -th chain and connected within this chain.

These theoretical designations are unknown when the PCB image is submitted for processing.

3 Preprocessing Algorithms

3.1 Thinning Algorithm

A discrete three-dimensional image is an array of points $P = \{p(x,y,z)\}$, each of which is assigned an intensity value in a three-dimensional space defined by Cartesian coordinates $x,y,z \in Z$. This applies to

both a binary image and the skeleton generated from it.

The skeleton of a digital image is a compact representation of its topology, geometry, and scale. The skeleton has features that can help identify images and their components. The properties of the skeleton are as follows:

- the line thickness is one pixel,
- all segments of one object are connected if all pixels of this object are connected,
- lines are in the center of the objects.

An example of a skeleton for the fragment in Figure 1 (without defects) is shown in Figure 2.

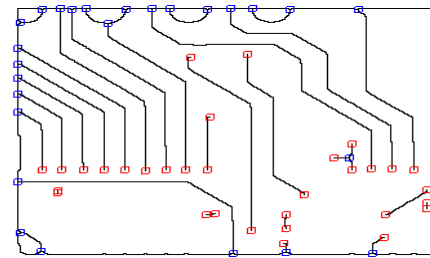


Fig. 2: A skeleton with specific points
Source: created by the authors

The skeleton was built by the thinning algorithm, which iteratively scans a 3x3 matrix of pixel positions and replaces black pixels with white ones until none can be removed without altering the connectivity.

A 2-D skeleton consists of pixels of three types: end-pixels with only one neighboring pixel in the line, normal pixels having two neighbors in the line, and branch pixels with more than two neighbors.

Building a skeleton involves converting an image of one type to an image of another kind. The operator S denotes the algorithm's actions. There are two types of results: a pure skeleton without specific points defined and a skeleton with outlined end- or branch-points. Two operators take the following form:

$$\begin{aligned} S(s, 0, 0): Im_o &\rightarrow Im_s, \\ S(s, e, b): Im_o &\rightarrow Im_{ss}, \end{aligned} \quad (1)$$

where Im_s is a skeleton, Im_{ss} is a skeleton with marked points, e is to define the end-points, b to define branch-points.

3.2 Flood-filling and Edge Detection Algorithm

The flood fill algorithm replaces the internal color Io of the object with the fill color in . When there are no more pixels of the original interior color, the algorithm terminates. This algorithm is based on a

four-join or eight-join method for pixel filling. It searches for all adjacent pixels that are part of the interior.

The next step is to modify its basic algorithm formula for application to PCB images with non-uniform component colors. For the available intensity value I_o and the target intensity value I_n , the algorithm implements modified formulas for the lower and upper tolerances:

$$\text{If } I_o - (I_o \cdot Tol) \leq I(x_i, y_i) \leq I_o + (I_o \cdot Tol) \quad (2)$$

where Tol is a relative tolerance regulating the rules for changing (filling) surrounding pixels.

Figure 3 shows two examples of the images filled by Paint and the modified algorithm with an acceptable tolerance of 100. In the first case, the black surface and some non-filled components remain after filling.

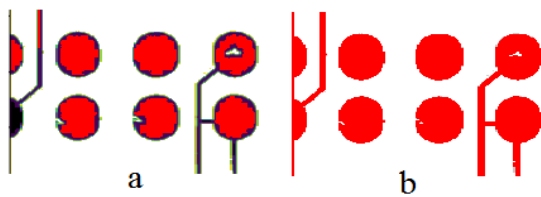


Fig. 3: Traces and contacts filled by Paint (a) and by the filling algorithm with tolerances (b)
Source: created by the authors

The third improvement of the algorithm involves a control for filling either a separate connected component or all components simultaneously that have the same pixel intensity.

In practice, this is implemented by four operators that convert images from one type to another type of presentation:

$$\begin{aligned} F(C, 1, tol^+, tol^-): Im_o &\rightarrow Im_1, \\ F(C, all, tol^+, tol^-): Im_o &\rightarrow Im_n, \end{aligned} \quad (3)$$

$$\begin{aligned} F(I, 1, tol^+, tol^-): Im_o &\rightarrow Im_1, \\ F(I, all, tol^+, tol^-): Im_o &\rightarrow Im_n, \end{aligned}$$

where Im is an image, C is a cursor as initial control, I is the intensity value for the starting point, all is a marker for filling of all possible components, tol^+ , and tol^- are tolerances for the target pixels.

The results of the first two operators are shown in Figure 4: one square (b) and three squares (c) are filled with red.

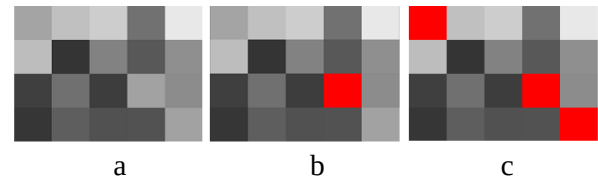


Fig. 4: Test images: non-filled (a), one filled component (b), three filled components (c)
Source: created by the authors

The latter modification of the filling algorithm is a procedure for building edges of the filled areas. The scanning 3×3 area checks the border pixels and fills all 9 pixels of the area around every border pixel. This process is illustrated in Figure 5 by the blue area.

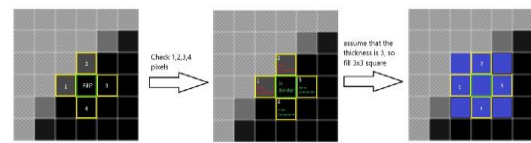


Fig. 5: Border pixels and border areas
Source: created by the authors

The results of the filling and edge detection in the filled areas are shown in Figure 6. The width of the circle edge is three pixels, and one pixel for the rectangle.

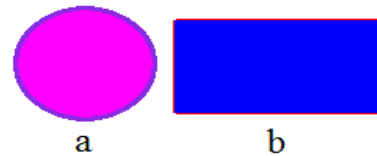


Fig. 6: Three (a) and one (b) pixel edges
Source: created by the authors

Four operators are implemented to control the process, transforming an image without edges into an image with edges. They take the form of the previous four operators with E as the edge, for example:

$$F(C, 1, E tol^+, tol^-): Im_o \rightarrow Im_1(E). \quad (4)$$

3.3 Overlaying Images

The detection of discrepancies in the structural components of reference and manufactured printed circuit boards is possible due to their mathematical superposition. This method displays the differences and common components in the two boards. The superposition of the pair of images is carried out using the following formula, which is applied to the three channels of each pixel:

$$I_r(x, y) = \alpha I_s(x_i, y_i) + (1 - \alpha) I_{rs}(x_i, y_i)_i(r), \quad (5)$$

where I_r , and I_s are the pixel intensity of the resulting and sample images, I_{rs} , is the pixel intensity of the reference image, and α is the coefficient of transparency (independent variable).

Both input images are binary: with black ($B, I=0$) and white ($W, I=255$) colors. To separate positive and negative defects in the resulting image, the transparency coefficient is accepted as $\alpha=0.25$. So, the overlaying image is characterized by the following colors:

$$\begin{aligned} W &= 0.25\text{white}(1) + 0.75\text{white}(2), \\ G(l) &= 0.25\text{black}(1) + 0.75\text{white}(2), \\ G(d) &= 0.25\text{white}(1) + 0.75\text{black}(2), \\ B &= 0.25\text{black}(1) + 0.75\text{black}(2) \end{aligned}$$

where (1), (2) denote the input images.

An example of two overlaying images is shown in Figure 7.

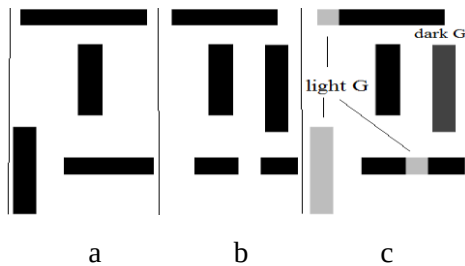


Fig. 7: Test images: a reference for $\alpha=0.25$ (a), a sample for $\alpha=0.75$ (b), overlaying (c)

Source: created by the authors

There are four colors in the resulting image: white W for the background, black B for black pixels reflecting black pixels in the reference and sample images, light gray $G(l)$, and dark gray $G(d)$ for positive and negative defects. Light gray marks black components belonging to the image with the transparency coefficient $\alpha=0.25$, and dark gray for $\alpha=0.75$ during overlaying with white pixels. In total, all types of defects and inaccuracies are automatically marked during overlay operations.

Overlaying is the conversion of two images into a single image with new features. To run this conversion, the following operator is used:

$$O(Im_1, Im_2, \alpha): Im_1, Im_2 \rightarrow Im_f \quad (6)$$

4 Detection of Contact Coordinates

4.1 Binary Images with Contacts without Holes

A majority of PCB images contain contacts and traces of the same color, and the contacts are without holes. In the case that these contacts are the endpoints of traces, their coordinates are determined as the endpoints in the skeleton. If they are located within a trace, the thinning algorithm does not notice them, accepting them as fragments of the trace. An example of such contacts is shown on the PCB fragment image in Figure 8.

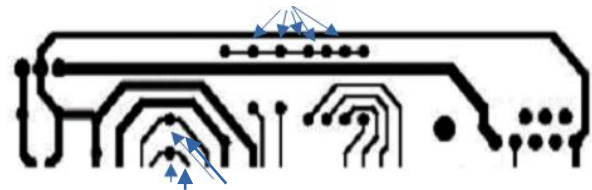


Fig. 8: A PCB image with contacts without holes

Source: created by the authors

The skeleton for this PCB image does not contain any round pins, and the algorithm cannot find any endpoints or branch points. This case with multiple pins is demonstrated in Figure 9. When checking the connections in all circuits, the absence of certain points does not prevent the correctness of the circuit connections from being verified. However, it does result in the absence of corresponding pins in some circuits for accounting and testing:

$$K_j(\text{found}) \subseteq K_j(\text{reference}).$$

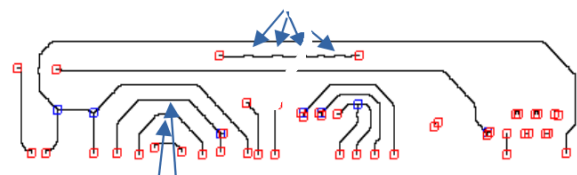


Fig. 9: Missing points (pixels) of contacts on the skeleton of the PCB image indicated by arrows

Source: created by the authors

Thus, a new approach is needed to determine the coordinates of all contacts. It is based on the method of finding and constructing edges of objects. The edges have a width of one to three pixels, and taking up space on the track with contacts reduces the thickness of the track to zero, leaving the central parts of the contacts to determine their coordinates.

This process is iterative with the filling operator F and a width W of the trace:

$$W^{k+1}=F(W^k), k=0, 1, \dots, z, W^z=0. \quad (7)$$

To illustrate the steps of determining the coordinates of contacts without holes, a small PCB image was chosen, as shown in Figure 10.

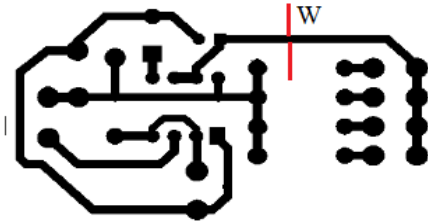


Fig. 10: A PCB image with contacts without holes
Source: created by the authors

The edge detection and filling operator (4) is applied to the input image three times: one from the white background and two from the traces.

Figure 11 shows three steps of the image transformation. The colors of the edges and filled parts of traces are chosen arbitrarily to illustrate wiping out traces. Black isolated islands of contacts remain in the image on the last step.

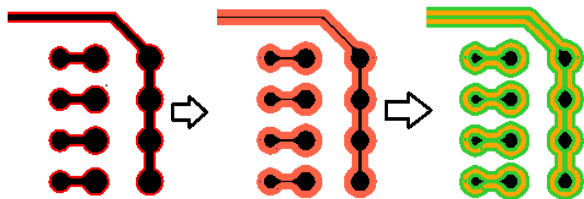


Fig. 11: Three steps of filling and edge detection in the PCB image fragment
Source: created by the authors

The full PCB image after three steps of transformation is shown in Figure 12.

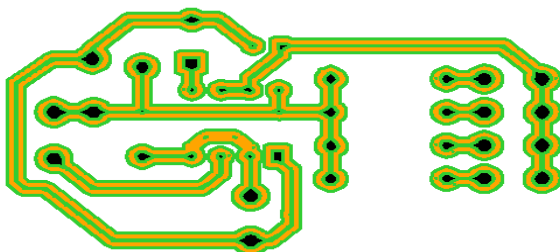


Fig. 12: The resulting PCB image with black isolated islands
Source: created by the authors

For further processing, the main result of the transformations is the presence of black islands of contact centers. Applying a sequence of operators to fill all non-black pixels with the white target color,

only black islands remain in the image, as shown in Figure 13.

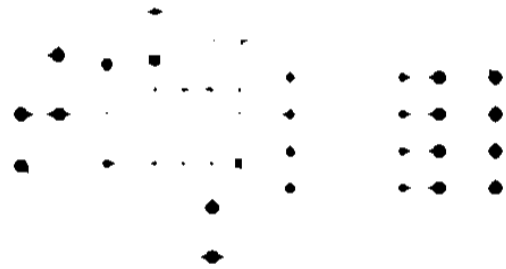


Fig. 13: Black islands of contacts after deleting other colors
Source: created by the authors

The thinning and specific points finding operators (1) transform the resulting image into the skeleton with marked contact coordinates. They are shown in Figure 14.

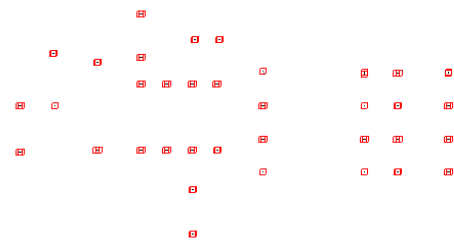


Fig. 14: Skeleton of contact islands and end-points
Source: created by the authors

It should be noted that even the smallest islands of the small contacts received their coordinates. The found coordinates of contacts are accepted as input data for defect detection in manufactured PCB images.

4.2 Transformation of Three-Color PCB Images into Images with Three Uniform Colors

In some cases, PCB images have three or more colors. They are not usually uniform or homogeneous. The intensity of the image pixels increases or decreases gradually. The image in Figure 15 has four colors: a dark green background, light green signal traces, light gray contacts, and a yellow kernel of contacts.



Fig. 15: A PCB image with four colors
Source: created by the authors

Therefore, dividing the pixels into three or four groups, as was done in the previous work [24], is possible, but with no desired result. Solder pixels in the middle of the contacts are mixed with traces, and the contacts receive holes. The needed result can be achieved using the K-means clustering algorithm to divide the image into two segments: background and signal elements. The resulting image is shown in Figure 16.

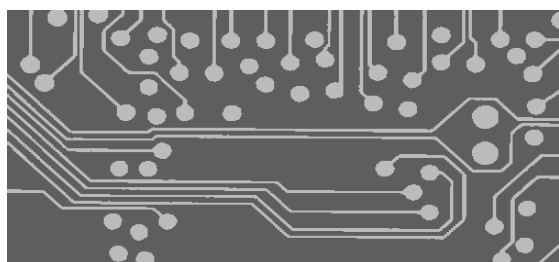


Fig. 16: A clustered PCB image
Source: created by the authors

Then, the three previous steps are repeated with the clustered image: wiping traces to separate contacts. The erased and separated contact traces are shown in Figure 17 in red color.

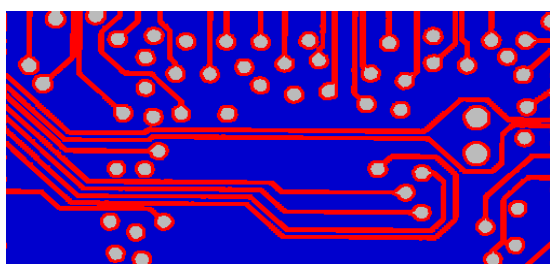


Fig. 17: Traces, contacts, and background of different colors
Source: created by the authors

Then, filling the contacts with black and the rest of the pixels with white, the contacts become isolated as shown in Figure 18a. Finally, thinning to build a skeleton and selecting specific points are performed. The pixels with contact coordinates after their finding are shown in Figure 18b.

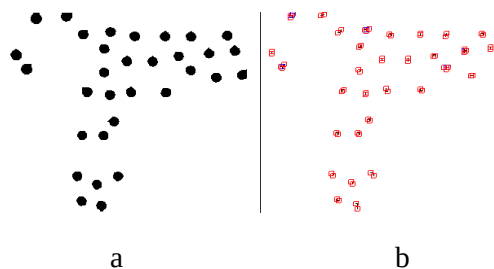


Fig. 18: Contacts (a) and coordinates of their specific points (b)
Source: created by the authors

Now, the final encircled pixels from islands simultaneously represent contacts and chains. The last image in Figure 18b is ready to be used as input data for defect detection in PCB images.

4.3 PCB Images with Holes in Contacts

Many printed circuit board images contain contacts and traces of the same color, and the circuit contacts have holes. For them, the determination of contact coordinates is carried out by a slightly different sequence of auxiliary algorithms. An example of such a printed circuit board image is shown in Figure 19.

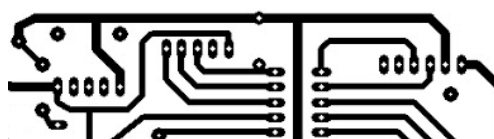


Fig. 19: A PCB image with one black color and holes in the contacts
Source: created by the authors

There are two ways to determine contact coordinates: based on a single application of the thinning algorithm and based on fill and thinning algorithms. The first method is simpler, but in addition to contact coordinates, it finds the coordinates of the branching points of three or more traces. The second method is more complex, but it finds only contact coordinates.

In the first method, two steps are implemented on the input image: thinning to build a skeleton and finding specific points. To illustrate how specific points belong to contacts, the skeleton with the coordinates of the specific points marked and the original input image are superimposed and shown in Figure 20 (the reference image is yellow).

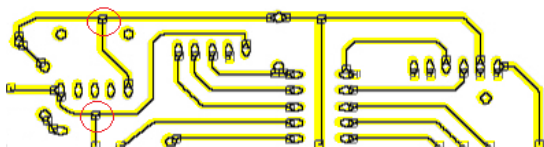


Fig. 20: The overlaid PCB image with its skeleton

Source: created by the authors

The pixels of the found special points are circled with squares. The figure also shows two points that do not belong to the contacts.

The task of the second method is to mark and separate all holes from other elements of the board, primarily from the background. The filling algorithm in the first step changes the white background to yellow. In the second step, the white holes change to red. To do this, two operators based on the filling operator (3) are executed for the input image in Figure 20: first, the white background is changed to yellow, and second, the white holes are changed to red. As a result, the white holes and the background are separated by colors. The former are marked in red and the background in yellow, as shown in Figure 21.

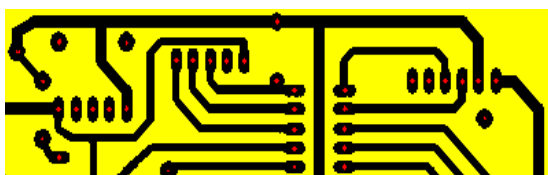


Fig. 21: Yellow background and red holes in the PCB fragment image

Source: created by the authors

Then the following two operators are executed to return the background to white and mark the black traces as white. After these two operations, only red holes remain on the board. They are shown in Figure 22a and Figure 22b.

In the final stage, the red holes or their black copy are accepted as input data for the thinning (skeleton) algorithm, which determines the contact coordinates. The task of detecting the contact coordinates is completed. They are located in computer memory and in the resulting image shown in Figure 22b.

In conclusion, contacts in PCBs are the primary constructive elements that must be correctly connected for the circuit to function properly. Their coordinates help to detect all inaccuracies linked with connectivity defects.

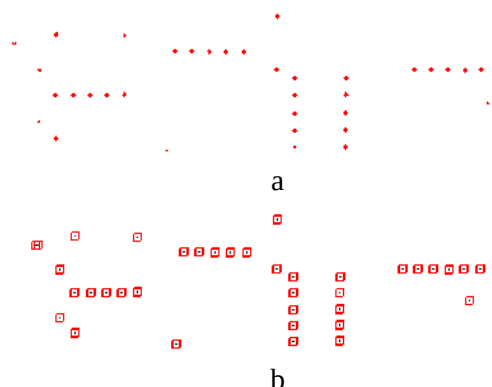


Fig. 22: PCB holes in contacts (a) and their coordinates (b)

Source: created by the authors

5 Defects Detection

5.1 Assigning Contact Pixels to Chains

The coordinates of contacts determined in the previous stage of the approach have no identifiers; that is, their affiliation with all chains is not known. The iteration procedure based on the flood-fill algorithm solves this problem. It assigns the set of contacts $K=\{K_1, K_2, \dots, K_N\}$ to the set of chains $C=\{C_1, C_2, \dots, C_N\}$ where every element K_j is a set of found contacts $K_j=\{K_{j1}, K_{j2}, \dots, K_{jkj}\}$. To accomplish this, the procedure utilizes the coordinates of contacts and the reference PCB image to fill the corresponding chain (trace), as shown in Figure 23.



Fig. 23: A filled chain with encircled contact pixels

Source: created by the authors

A filled chain and its contact pixels receive an identifier $ID=1$. Then, the process continues until all chains and contact pixels receive their identifiers. The final image is shown in Figure 24. The holes contain small white areas filled with contact pixels and squares without found black islands. In practice, for processing, they are filled with these black islands received after wiping.

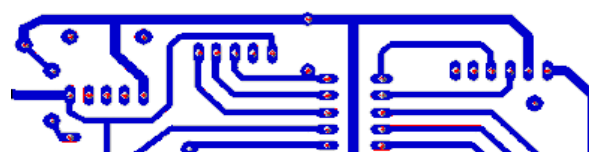


Fig. 24: Filled chains with encircled contact pixels

Source: created by the authors

After such operations, every chain has an identifier and contacts (contact pixels) assigned to it:

$$\begin{aligned} C_1(id_1) &= \{K_{11}(id_1), K_{12}(id_1), \dots, K_{1k_1}(id_1)\}, \\ &\dots, \\ C_N(id_N) &= \{K_{N1}(id_N), K_{N2}(id_N), \dots, K_{Nk_2}(id_N)\}. \end{aligned} \quad (8)$$

where $id_1, \dots, id_N$ are identifiers of the chains, $k_1, k_2, \dots, k_N$ are the indexes of contact pixels in the chains.

The number of identified pixel contacts and their coordinates are input data for the next steps of defect detection.

5.2 Calculation of Component Features

Simple operations are performed with the image with pixel contacts and the manufactured sample image. The first image is black, and the second has a white background covered by four-pixel islands marking the contact pixels.

The overlaying experiment with $\alpha=0.3$ (for black) is held with the following pixels:

$$\begin{aligned} W &= 0.3\text{white}(1) + 0.7\text{white}(2) = \text{white}, \\ LR &= 0.3\text{white}(1) + 0.7\text{red}(2) = \text{light red}, \\ LG &= 0.3\text{black}(1) + 0.7\text{white}(2) = \text{light grey}, \\ DR &= 0.3\text{black}(1) + 0.7\text{red}(2) = \text{dark red}. \end{aligned}$$

The resulting image in Figure 25 has four colors.

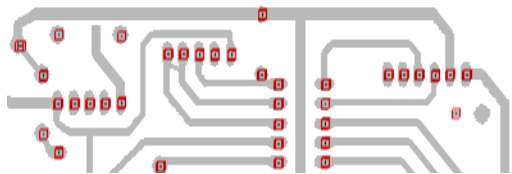


Fig. 25: The overlaid manufactured image with pixel contacts of the reference image
Source: created by the authors

They represent the following objects: missing metal in light red, traces and contacts in light grey, and contact pixels in dark red. These colors help to identify features that describe different types of defects.

Visually, the types of defects can be determined by the following rules. If the gray body does not contain the complete set $C_i = \{K_{i1}, K_{i2}, \dots, K_{ik_2}\}$ of contact pixels belonging to the i -th chain, there is an open defect. If the gray body contains contact pixels from two (or more) sets C_i, C_j of contact pixels, there is a short defect between the chains. A gray body without any contact pixels defines an additional pad, and a contact pixel on a white background defines a missing pad.

A simple approach for defect measurement involves a filling algorithm and comparison operations on contact pixels in the sets of chains $C_1, C_2, \dots, C_N$. Using starting points from the set $C(C_1, C_2, \dots, C_N)$, a certain chain changes its color. Then, the contact pixels are checked to see how many of them have also changed their color. Examples of filled chains with defects are shown in Figure 26, where they are illustrated in different colors. In practice, they are not. The number of filled contact pixels and their affiliation indicate the type of defect or the absence of defects.

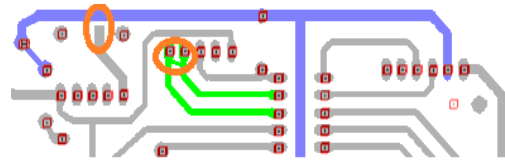


Fig. 26: One blue chain with an open defect and two green chains with a short defect
Source: created by the authors

When the current i -th chain has changed its color (for example, to blue), the array A of contact pixels is checked to obtain a list L of pixels that have also changed their color (to blue). After the check, three answers are possible. The first list contains the complete set of contact pixels belonging to this chain. The weighting factor is taken as $w_i=1$. The second list contains a part of the contact pixels describing this chain. The weighting factor is $w_i=n'_i/n_i$, where n'_i is the real number of contact pixels in a gray body, and n_i is the number of contact pixels in the reference chain.

The blue chain in Figure 26 is due to an open defect, and the absence of one contact has $w_i=4/5$. The third list contains pixels belonging to two or more chains of the sample image. The weighting factor is $w_i=n'_i/n_i+n'_j/n_j$, where j is the number of another chain connected to the current one. The green chain in Figure 27 has $w_i=2/2+2/2=2$.

As a result, the weighting factor indicates the type of defect: $w_i=1$ for the correct chain, $w_i > 1$ for a short defect, $w_i < 1$ for an open defect. Such estimation is made for all gray chains in the image. If defects are repeated, the summary estimation is averaged for the same type of defect.

The weighting factor $w_i=-1$ (lack) is used for the empty position, and $w_i=0$ (extra) for the gray chain without points. In the first case, all contacts not considered and not included in some gray chain form the weighting factor $w_i=-1$. The presence of any number of gray components without an enabled contact forms a weighting factor $w_i=0$. Clarification

of the number and location of components is carried out by additional procedures.

Figure 27 shows the distribution of characteristics of different types of defects on the number axis. Correct, missing, and extra components correspond to the exact values 1, -1, and 0. For open and short defects, their values are in two different intervals.

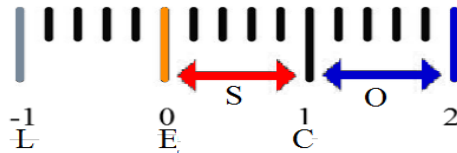


Fig. 27: Values for defects of different types

Source: created by the authors

As a result, four types of weight coefficients for four kinds of defects in the printed circuit board are ready to be passed to the neural network to obtain a classification of board defects without user intervention.

6 Application of Machine Learning

Recognition of defects directly in PCB images or derivative images of differences of cumulative histograms by Deep learning in TensorFlow [21] requires a test dataset of nearly 3-10 thousand images. Similar experiments in [22] used the PyTorch and NumPy libraries with 3 thousand samples and flood-filling algorithms [23]. One of the simplest ways to apply Machine Learning to classify defects is the JavaScript library *Brain.js* [24] with a forward neural network (FNN). Many control parameters are accepted by default. If necessary, the user can set them to study the effectiveness of the neural network modes. The main object of the library is *NeuralNetwork()* with two methods to form the model: *train()* and *run()*. The training model is based on data values located on the numerical axis.

In the given approach, the close numerical values adopted for training give the correct probabilities of the types of defects and circuits. However, these values are not very contrasting. This is explained by the proximity of the numbers in the input data. To distinguish the designations of short-circuit and open-circuit defects, the corresponding increased numbers replace their single number: for the first -100 and the second 100. Then the five defects for training are selected as follows: correct chains—(1.0); short defect—(-100); open defect—(100); shortage defect—(-1.0); extra defect—(0.0).

Then these data are accepted as input in the JavaScript training methods:

```
// Create a Neural Network
const net = new brain.NeuralNetwork();
// Train the Network with 5 input objects
net.train([
  {input:[-100],output:{short:1}},
  {input:[100],output:{open:1}},
  {input:[0.0],output:{extra:1}},
  {input:[-1.0],output:{lack:1}},
  {input:[1.0],output:{correct:1}},
]);
// What is the output of [-100]?
let result = network.run([-100]);
```

The classification result is as follows:

```
short: 0.9618812799453735
open: 7.157166237448109e-7
extra: 0.013281581923365593
correct: 0.000020949071767972782
lack: 0.03759550675749779
```

For the correct circuit, the classification result is as follows:

```
let result=net.run([1.0]);
short:0.006235572509467602
open:0.060157112777233124
extra:0.13287942111492157
correct:0.8401375412940979
lack: 0.002182509284466505
```

In the first case, the probability of the short defect is much greater than other values due to the great distance from the value of -100 to other training values. In the last case, the non-decisive probability of an extra defect is slightly higher due to the closer values of 0 and 1.

Additionally, another method for diagnosing printed circuit boards has been developed. A series of data is generated. The state of the entire PCB is encoded as a 16-element long vector.

The data set was generated for 5 classes (states):

1. Correct: The vector consists exclusively of 1.0 values.
2. Missing: A value of -1.0 appears in random positions.
3. Extra (Extra metal/Additional contact pad): the positions of the vector contain the value 0.0.
4. Short: Values in the range [1.2 ... 2.0] are generated at random positions.

5. Open: Values in the range [0.5 ... 0.7] are generated in random positions.

Then, 3000 (1500 per class) samples for training and 600 for validation were generated.

The ANN architecture is chosen as: [14] → [24] → [16] → [2]. The network demonstrated the target error (0.005) in 222 iterations.

The following examples show possibilities of developed ANN (N = 16):

Correct. Entered the series: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1. The network identifies the series as correct with 89.8% confidence.

Missing. Entered the series: 1, 1, -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1. The network classifies the series as missing with 100.0% confidence.

Open. The series is input: 1, 1, 1, 1, 1, 1, 1, 0.6, 1, 1, 1, 1, 0.55, 1, 1, 1. The network classifies the series as open with 99.49% confidence.

In conclusion, the developed software, combined with machine learning, classifies four types of PCB defects and images with these defects. To increase the accuracy of the output result, the input data can be expanded and supplemented with values from intervals on the number axis, as well as scaled to distance their value from each other.

7 Conclusion

Overlaying, thinning, and filling algorithms are programmed in the developed software to detect the coordinates of defects: short, open, excess, and lack of metal. The flood-filling and edge detection algorithms separate contacts from connecting traces, allowing you to analyze each chain in the printed circuit board.

The newly obtained image has additional features of the PCB components. Their separation and measurement provide the input data for training and testing artificial neural networks to apply machine learning to the defect detection and classification process.

The multi-layer ANN program for training and testing is based on the well-known Brain.js tool, with many input parameters for choosing the number of layers and accuracy. This ANN classifies PCB images under the criteria that are accepted within the tolerance range.

The developed combination of tools classifies defects in PCB images into defective and correct samples without involving the user.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

- [1] 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, 2020, Vol. 8, pp.183192–183271, DOI: 10.1109/ACCESS.2020.3029127.
- [2] Elena Jasiuniene, Renaldas Raišutis, Vyktas 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, 2024, 24, 1719, DOI: 10.3390/s24061719.
- [3] Ling Q., Isa N.A.M., Printed Circuit Board Defect Detection Methods Based on Image Processing, Machine Learning and Deep Learning: A Survey. IEEE Access, 2023, Vol. 11, pp. 15921–15944, DOI:10.1109/ACCESS.2023.3245093.
- [4] Raghavendra Chalapathy, Sanjay Chawla, Preprint. Deep Learning for Anomaly Detection: A Survey. January, 2019. DOI:10.48550/arXiv.1901.03407.
- [5] Domen Tabernik, Samo Šela, Jure Skvarč, Danijel Skočaj, Segmentation-based deep-learning approach for surface-defect detection. Journal of Intelligent Manufacturing, Springer, 2020, Vol. 31(3), pp. 759-776, DOI: 10.1007/s10845-019-01476-x.
- [6] Balzategui, Eciolaza L, Few-shot incremental learning in the context of solar cell quality inspection. Expert Systems with Applications, 2023, Vol. 228, 120382. DOI:10.48550/arXiv.2207.00693.
- [7] Volkau, I., Abdul, M., Dai, W., Erdt, M., and Sourin, A., Detection defect in printed circuit boards using unsupervised feature extraction upon transfer learning. In 2019 International Conference on Cyberworlds (CW), October 02, 2019, Kyoto, Japan, IEEE, pp. 101–108. DOI: 10.1109/CW.2019.00025.

- [8] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., Gehler, P., Towards total recall in industrial anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18-24 June 2022, New Orleans, LA, USA. pp. 14318-14328. DOI: 10.48550/arXiv.2106.08265.
- [9] Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., and Dario, P., Visual-based defect detection and classification approaches for industrial applications—a survey. *Sensors*, 2020, 20(5), p. 1459. DOI: 10.3390/s20051459.
- [10] Ruff, L., Vandermeulen, R. A., Gornitz, N., Binder, A., Muller, E., Muller, K.-R., and Kloft, M., Deep semi-supervised anomaly detection. 2020, Preprint arXiv:1906.02694. DOI: 10.48550/arXiv.1906.02694.
- [11] Cai L., Li J., PCB defect detection system based on image processing. *Journal of Physics: Conference Series*, 2022, Vol. 2383, No. 1, pp. 012077, IOP Publishing, DOI: 10.1088/1742-6596/2383/1/012077.
- [12] Sanli Tang, Fan He, Xiaolin Huang, Jie Yang, Online PCB Defect Detector On A New PCB Defect Dataset. arXiv: Computer Vision and Pattern Recognition. DOI: 10.48550/arXiv.1902.06197.
- [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*, 2021. 21(4), 040801. DOI: 10.1115/1.4049535.
- [14] Wenjie Xiong, Guangzhu Chen, Xiaojuan Liao, Runji Chen, Adaptive dual teacher incremental learning for defect detection of flexible printed circuit, *Computer and Electrical Engineering*, 2024, Vol. 118, 109337. DOI: 10.1016/j.compeleceng.2024.109337.
- [15] S. McClure, Building an End-to-End Defect Classifier Application for Printed Circuit Boards, Extracting and Classifying Circuit Board Defects using Image Processing and Deep Learning, Feb. 4, 2020, [Online], <https://medium.com/towards-data-science/building-an-end-to-end-deep-learning-defect-classifier-application-for-printed-circuit-board-pcb-6361b3a76232> (Accessed Date: April 7, 2026).
- [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*, 2020, Vol. 9, No. 9, pp. 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, <https://www.nature.com/articles/s41598-022-16302-3>.
- [18] Jungsuk Kim, Jungbeom Ko, Hojong Choi and Hyunchul Kim, Printed Circuit Board Defect Detection Using Deep Learning via A Skip-Connected Convolutional Autoencoder, *Sensors*, 2021, No.21(15), 4968, DOI: 10.3390/s21154968.
- [19] C. L. Perera, S. C. Premaratne, An Ensemble Machine Learning Approach for Forecasting Credit risk of Loan Applications, *WSEAS Transactions on Systems*, 2024, Vol. 23, pp. 31-46, DOI: 10.37394/23202.2024.23.4.
- [20] Atul Kachare, Mukesh Kalla, Ashutosh Gupta, Visual Question Generation Answering (VQG-VQA) using Machine Learning, *WSEAS Transactions on Systems*, 2023, Vol. 22, pp. 663-670, DOI: 10.37394/23202.2023.22.67.
- [21] An end-to-end platform for machine learning, [Online]. <https://www.tensorflow.org/> (Accessed Date: April 7, 2026).
- [22] Melnyk Roman, Vorobii Vitalii, PCB Image Defects Detection by Artificial Neural Networks and Resistance Analysis. *WSEAS Transactions on Circuit and Theory*, 2023, 22(1), pp. 35–42, DOI: 10.37394/23201.2024.23.7.
- [23] Melnyk, R.A., Kalychak, Y.I Detection of defects in printed circuit boards by flood-fill algorithm and distributed cumulative histogram, *Proceedings of the Conference “International Scientific and Technical Conference on Computer Sciences and Information Technologies”*, 2018, 1, pp. 155–159, Date of Conference: 11-14 September 2018, Conference Location: Lviv, Ukraine, DOI: 10.1109/STC-CSIT.2018.8526733.
- [24] Machine Learning in JavaScript, [Online]. https://www.w3schools.com/ai/ai_what_is.asp (Accessed Date: April 7, 2026).

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

The authors equally contributed 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 that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US