applied sciences

MDPI

Article

Feature-Learning-Based Printed Circuit Board Inspection via Speeded-Up Robust Features and Random Forest

Eun Hye Yuk [1], Seung Hwan Park [2], Cheong-Sool Park [1] and Jun-Geol Baek [1,*]

[1] Department of Industrial Management Engineering, Korea University, Seoul 02841, Korea; eunhyeyuk@korea.ac.kr (E.H.Y.); dumm97@gmail.com (C.-S.P.)
[2] Data Science, SK Hynix Semiconductor, Icheon 17336, Korea; seunghwan1.park@sk.com
* Correspondence: jungeol@korea.ac.kr; Tel.: +82-2-3290-3396

Received: 14 May 2018; Accepted: 31 May 2018; Published: 5 June 2018

Featured Application: The main contribution of this work is to propose an inspection method using image data generated at the actual manufacturing process. This proposed method can help printed circuit board (PCB) manufacturers more effectively detect defects, such as scratches and improper etching, in an automated optical inspection (AOI). Moreover, the proposed method of this work can be also applied to the field of dermatology, where it has to detect skin diseases, as well as in PCB inspection.

Abstract: With the coming of the 4th industrial revolution era, manufacturers produce high-tech products. As the production process is refined, inspection technologies become more important. Specifically, the inspection of a printed circuit board (PCB), which is an indispensable part of electronic products, is an essential step to improve the quality of the process and yield. Image processing techniques are utilized for inspection, but there are limitations because the backgrounds of images are different and the kinds of defects increase. In order to overcome these limitations, methods based on machine learning have been used recently. These methods can inspect without a normal image by learning fault patterns. Therefore, this paper proposes a method can detect various types of defects using machine learning. The proposed method first extracts features through speeded-up robust features (SURF), then learns the fault pattern and calculates probabilities. After that, we generate a weighted kernel density estimation (WKDE) map weighted by the probabilities to consider the density of the features. Because the probability of the WKDE map can detect an area where the defects are concentrated, it improves the performance of the inspection. To verify the proposed method, we apply the method to PCB images and confirm the performance of the method.

Keywords: image inspection; non-referential method; feature extraction; fault pattern learning; weighted kernel density estimation (WKDE)

1. Introduction

Because the era of the Internet of Things (IoT) has been accompanied by the rapid development of the semiconductor industry and communication technologies, the use of high-tech products, such as mobile phones and wearable devices, has been spreading widely in our daily lives. The printed circuit board (PCB) is one of the key components of such electronic devices. A PCB is a thin plate made by printing an electrically conductive circuit on an insulator. Figure 1 shows an image of a generic PCB. As shown in Figure 1, a PCB connects to different parts electrically through a point-to-point wiring process.

In the past, handwork wiring led to frequent failures of the wire junctions and a short circuit as the wire insulation began to age. Furthermore, this type of work was conducted manually for inner-connecting components within the board. Such wiring, therefore, required a significant amount of time and effort. With advancements in technology, PCBs were developed for efficient and automated production. Because PCBs allow for the use of mass production, they allow devices to be smaller and lighter. In addition, they allow a high level of reliability at low production costs. Because of these advantages, most electronic devices use PCBs.

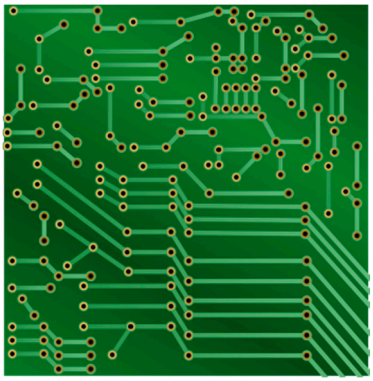


Figure 1. An illustrative image of a printed circuit board (PCB).

According to the recent trend of miniaturized and high-performance electronic devices, the demand for PCBs has increased substantially. The current PCB market has shown an average annual growth rate of 4%. Based on increasing demand, electronics manufacturers require perfect quality and a high level of accuracy from their PCB inspections to assure their competitive edge. To meet these quality requirements, PCB manufacturers conduct an inspection before proceeding to the main process. The PCB manufacturing process consists of many steps: cutting, inner layer etching, an automatic optical inspection (AOI), lay-up, lamination, etching, drilling, solder masking, routing, a bare board test (BBT), quality control, packing, and shipping, in that order. Before proceeding to the main processes, such as lamination and etching, faults in a PCB are detected during the AOI stage. An AOI is an automated visual inspection using an image comparison method. Most PCB manufacturers inspect their PCBs through the AOI process. The types of defects detected during the AOI process include scratches, improper etching, and open circuits. In particular, scratches are fatal defects because they have the potential to change the electrical properties and can result in a malfunction of the completed product.

There are typically three methods applied during the AOI process: a comparison reference (CR), non-reference verification (NV), and a hybrid approach (HA). The CR method compares an inspected image with a reference image. It measures any existing dissimilarities between the reference image and the inspected image. Thus, this method requires a reference image, and inspection is difficult without such an image. The NV approach tests the design rule of the PCB for detecting faults. It essentially verifies the widths of the insulators and conductors. However, this approach makes it difficult for users to design the rules as restrictions on the image features, and it cannot detect faults without such rules [1]. The HA approach combines various types of CR and NV methods. That is, the approach utilizes a reference image or design rules to inspect a PCB image. However, it still does not solve the limitation of being unable to conduct an inspection without reference information. Because the circuits used in a PCB are diverse and complex, the reference information increases significantly, thereby decreasing the inspection's efficiency. Therefore, additional studies are required to detect faults without reference information.

Numerous algorithms have been proposed to improve the accuracy of a PCB inspection. Such algorithms can be categorized into two approaches: referential and non-referential methods.

A referential method is based on a comparison between the inspected image and a reference image. To measure the dissimilarity between these two images, image subtraction and template matching techniques are used. Wu et al. proposed an inspection method based on a subtraction method [2]. The image subtraction method compares both images using an XOR logic operator [3]. The resulting image, which is obtained after this operation, contains only portions of a fault [4]. Therefore, the reference image and the inspected image should be placed in a fixed position to compare both images. In addition, a reference image of the same size is required. Template matching is a technique for identifying the parts of the image that match the reference image. This method extracts the features of both the reference and inspected images. It then calculates the similarities of these features. One of the major disadvantages of template matching is that a large amount of information regarding the reference image must be used. Therefore, this method requires a mass storage device that can store all of the information. Moreover, the inspected images also have to be precisely matched for a comparison with the reference image [5]. Acciani et al. suggested an inspection algorithm that extracts the wavelet and geometric features, and then detects a defect after learning the fault pattern using a neural network and k-nearest neighbors [6]. When extracting the features of a PCB image, this method uses the maximum value of the correlation coefficients between the features of the reference image and the inspected image. This approach of applying machine-learning algorithms is called the learning-based model. The aim of this approach is to automatically detect faults through pattern recognition [7]. In [8], the authors proposed a defect detection method using feature matching for non-repetitive patterned images. The method first extracts features of both the reference image and the inspected image using a modified corner detector. It then detects a fault by finding a correspondence between two feature sets. Thus, the methods [6,8] still require a reference image to detect a fault.

A non-referential method, on the other hand, does not require a reference image. However, this method uses design rules. If the inspected image does not conform to the design specification standards, it is considered defective. Ye and Danielson suggested a verifying algorithm for minimum conductive and insulator trace widths [9]. This algorithm uses morphological techniques, which are methods for processing binary and grayscale images based on shape. Morphological techniques do not require a predefined model of a perfect pattern because we can construct specific shapes in an image by choosing an appropriate neighborhood shape [10]. However, this method has a disadvantage in that we should apply different pre-processing algorithms to check for faults in a PCB. In addition, it automatically increases the inspection time [5]. Tsai et al. proposed a non-referential defect detection approach for bond pads [11]. This approach restores the shape of the bond pads using Fourier image reconstruction. The method then evaluates the similarities and differences in the pad shape. However, the method has difficulty in reconstructing an original image of a PCB in the absence of a reference image because the shape of a circuit is quite varied and complex. Thus, the method is not suitable for detecting faults on a PCB. As mentioned earlier, a non-referential image analysis has certain limitations.

Rosten and Drummon proposed a very fast and high-quality corner detector using machine-learning techniques [12]. This algorithm uses a decision tree to learn the properties of the corner points and determine which point is a corner point instead of directly counting the number of consecutive points of the same type. Pernkopf proposed an approach for the detection of three-dimensional faults on scale-covered steel surfaces [13]. After extracting features, the approach uses a Bayesian network to learn the feature property and classify the faults. Classification research, which is learning the patterns of the extracted features through image processing, is actively carried out to diagnose skin cancer not only in manufacturing but also in dermatology [14]. It is indispensable to extract meaningful features that can explain the properties of shape and color in order to effectively detect a melanoma [15]. Roberta suggested a method to find meaningful features by combining three characteristics of shape, texture, and color [16].

Based on the concept of a learning property, we propose a new non-referential method for fault detection by extracting features based on an image-processing method and learning the fault information using random forests. The proposed method utilizes features obtained through

speeded-up robust features (SURF) to describe the fault information. Therefore, the method can detect a fault quickly and accurately without a reference image and without being affected by environmental changes, such as the size, rotation, and location of the PCB. The proposed method first extracts robust features using an image-processing technique and learns the fault pattern using an efficient classification technique utilizing high-dimensional data. We then calculate the probability and draw a weighted kernel density estimation (WKDE) map weighted by the probability and identify whether a fault has occurred.

The remainder of this paper is organized as follows: In Section 2, the background of the proposed method is briefly reviewed. Section 3 describes the procedure of the proposed inspection algorithm. Section 4 presents the experimental results through a comparison of the receiver operating characteristic (ROC) curves. Finally, some concluding remarks are given in Section 5.

2. Background of the Proposed Method

This section describes the two algorithms used in the proposed method. We first introduce SURF, which is used for extracting the features of the PCB. These features include important information in the image, such as the size, angles, coordinates, and color. By digitizing the image, they enable calculations related to the pattern recognition to be applied during image processing. We then describe the random forests used to learn the fault pattern and calculate the probability.

2.1. Speeded-Up Robust Features (SURF)

SURF first digitizes a PCB image into a vector to enable computational calculations through image processing for learning the fault pattern. Among the various types of image processing techniques, including scale invariant feature transform (SIFT) and orientation by intensity centroid (ORB), we use SURF, which is the most robust algorithm with regard to environmental factors such as size, shape, and color. The performance of SIFT does not differ significantly compared to that of SURF. However, SIFT is not able to extract the features in a small defective area. ORB is specialized in extracting the features of a rounded or curved edge, rather than a straight line, in a circuit. Because a PCB is mainly composed in a straight formation, ORB is inappropriate for extracting PCB features. On the other hand, SURF is robust and useful for feature detection. It also improves the computational speed compared to SIFT [17]. In addition, SURF provides robust features and distinctive descriptors to the size and rotation transform. Thus, it results in a strong performance regardless of the fault size and shape of the circuit. For this reason, the proposed method applies SURF for the feature extraction. The following subsections describe the SURF algorithm, which has two phases. First, a point of interest is detected in an image using a Hessian matrix. In the second phase, SURF generates descriptors with 64 dimensions, which describe the characteristics of the features.

2.1.1. Interest Point Detection Based on a Hessian Detector

SURF uses a Hessian matrix for detecting interest points, which represent the characteristics of the image and include useful information for identification, including corner points. An integral image is used in an approximated Hessian matrix to improve the calculation speed of box-type convolution filters. The integral image $J(x, y)$ at coordinate $x = (x, y)$ $x = (x, y)$ is the sum of all pixels within a rectangular area formed by the origin and coordinate x in input image I, the equation of which is as follows:

$$J(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j). \tag{1}$$

Given point $x = (x, y)$ in input image I, a Hessian matrix $H(x, \sigma)$ at a scale of σ is defined through Equation (2).

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}, \tag{2}$$

where $L_{xx}(x,\sigma)$, $L_{xy}(x,\sigma)$, and $L_{yy}(x,\sigma)$ represent the convolution of the Gaussian second-order derivative space with the image at coordinate $x = (x, y)$.

Bay et al. proposed the use of 9×9 box filters to approximate the second-order Gaussian partial derivatives and rapidly compute the image convolutions using integral images [17]. An approximation of the second-order derivatives is denoted by $D_{xx}(x,\sigma)$, $D_{xy}(x,\sigma)$, and $D_{yy}(x,\sigma$. Bay et al. suggested using a σ of 1.2, which is the lowest scale [17]. Therefore, the determinant of an approximated Hessian matrix is calculated using Equation (3).

$$\det\left(H_{approx}(x,\sigma)\right) = D_{xx}(x,\sigma)D_{yy}(x,\sigma) - \left(0.9D_{xy}(x,\sigma)\right)^2. \quad (3)$$

To obtain the robust features to scale, SURF forms a pyramid scale space with various scales on the original image. The size of the box filter expands along with the pyramid scale space while the size of the original image remains fixed. Consequently, we can have an effect of enlarging or reducing the size of the image without scaling the original. Moreover, we can apply box filters of any size at the same speed directly to the original image, which can improve the computational speed.

After the determinant of the approximated Hessian matrix is calculated at each scale, non-maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied to find the maxima, which describe the edge of the features best. The maxima are then interpolated in terms of both the scale and image space [18]. Finally, we can detect the stable location of the feature through one of the maxima.

2.1.2. Descriptor Generation

After the features are detected as mentioned above, SURF generates a descriptor, which describes the characteristics of the features, such as the shape and color, using the sum of the Haar wavelet responses [19]. The Haar wavelets enable SURF to increase the robustness and decrease the computational time. To generate the descriptor, the first step is constructing a square region around the points of interest and assigning a reproducible orientation based on the orientation-selection method introduced in [17]. The region is then split equally into 4×4 sub-regions to retain some of the spatial information as shown in Figure 2. In each sub-region, we compute the Haar wavelet responses at regularly 5×5 spaced sample points. Bay et al. suggested that the level of performance is best at the each of the above sizes [17]. The wavelet responses are calculated in the x- and y-axis directions (d_x and d_y) and summed over each sub-region. The wavelet responses are then weighted with a Gaussian centered on the point of interest to increase the robustness toward geometric deformations and localization errors. Moreover, to obtain information regarding the polarity of the intensity changes, the absolute values of the responses $|d_x|$ and $|d_y|$ are also summed. Thus, each sub-region has a four-dimensional descriptor vector as shown in Equation (4).

$$v = |\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y||. \quad (4)$$

As a result, the overall descriptor has a 64-dimension vector because 4 vectors are created for each 4×4 sub-region. Figure 2 shows a descriptor vector obtained by summing the wavelet response in a sub-region.

Finally, the descriptor vectors for each sub-region are normalized to reduce the impact on the environment, such as an external light or illumination. Eventually, we can obtain the robust features and descriptors of the PCB image using SURF. The descriptor for a single feature is generated, as shown in Figure 3, where m is the number of extracted features.

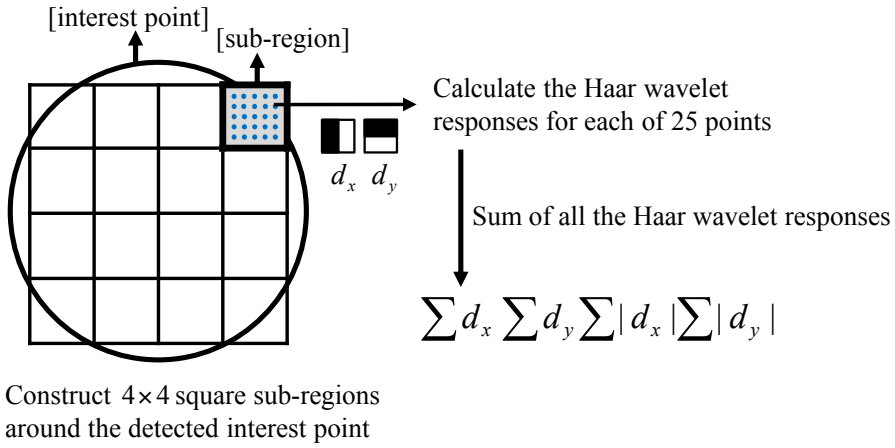


Figure 2. Diagram for generating the descriptors.

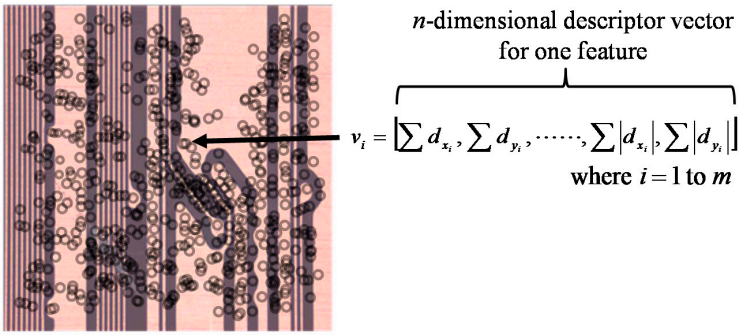


Figure 3. Features and descriptors obtained in a PCB image using speeded-up robust features (SURF).

2.2. Random Forests

Random forests is a classification method proposed by Breiman and is a type of ensemble learning method that constructs a multitude of decision trees and combines the predictions from them [20]. The random forests method is relatively accurate and fast for high-dimensional data. Furthermore, it prevents overfitting problems by voting on multiple trees and shows good predictive performance for noisy data. Because the descriptor vector obtained from SURF is composed of high-dimensional data, the algorithm is efficient for feature classification. Therefore, we use random forests to learn the fault pattern and calculate the probability of a fault feature being classified.

This algorithm begins by drawing many bootstrap samples from the training set. Classification trees are then built for each bootstrap sample using a decision tree learning method. After the forests are formed, the new data are put into each classification tree for classification. Each tree votes on the result of the tree's decision regarding the class of data. The forests then predict the class by taking the majority vote from the classification trees [21]. Because the random forests method generates many classification trees using bootstrapping rather than pruning, we can obtain low variation trees. It can therefore avoid an overfitting and thereby improve the performance. However, the performance can decrease when the data are extremely imbalanced. The class of PCB features is mostly normal, and the number of fault features is low. Thus, the performance inevitably decreases. To complement this disadvantage, we suggest utilizing the probability of being classified as a fault rather than predicting the class using the random forests and by considering the density of the features. By considering the density, the probability of being a defective feature increases, whereas the probability of being a normal feature decreases. Therefore, the proposed method detects faults more effectively than a method using only random forests. We next describe the process for calculating the probability of the

test data being classified as a fault when using random forests. Then, in Section 4, we describe how to use the calculated probability to detect a fault.

The probability $p(y = k)$ of being classified as k is $p(y = k) = \pi_k$, where $0 \leq \pi_k \leq 1$. Here, n is the number of classes. The density function $f_k(x)$ for the features in class k is $f_k(x) = f(x|y = k)$. Thus, the density function $f(x)$ for all features is calculated through Equation (5) [22].

$$f(x) = \sum_{k=1}^{n} \pi_k f_k(x). \tag{5}$$

Finally, we can estimate the probability that the class of new data x_{new} will be predicted as k using a Bayesian probability. The equation of this probability is denoted through Equation (6).

$$p(y = k|x = x_{new}) = \frac{\pi_k f_k(x_{new})}{f(x_{new})} \tag{6}$$

Thus, the probability p_i that a new feature x_i will be classified as a fault or as normal is calculated using Equation (7).

$$p_i = \begin{cases} \frac{\pi_F f(x_i)}{f(x_i)}, & p(y = Fault|x = x_i) \\ 1 - \frac{\pi_F f(x_i)}{f(x_i)}, & p(y = Normal|x = x_i) \end{cases} \tag{7}$$

3. Proposed Method

Based on the algorithms described in Section 2, we propose a scratch fault detection method, which is depicted in Figure 4. First, SURF extracts the features of the PCB image. The extracted features are composed of a multivariate vector representing the properties found in the PCB image, such as an edge, blob, or curve. Because SURF also extracts the features in a scratch-defective edge, it is difficult to distinguish between a normal edge and a fault edge using only these features.

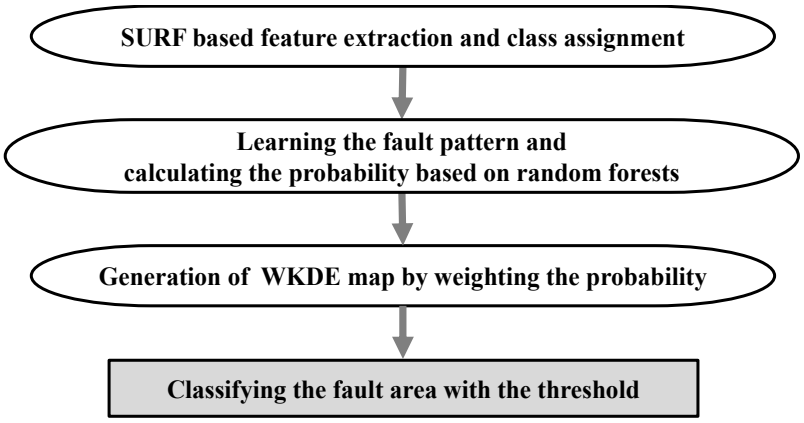


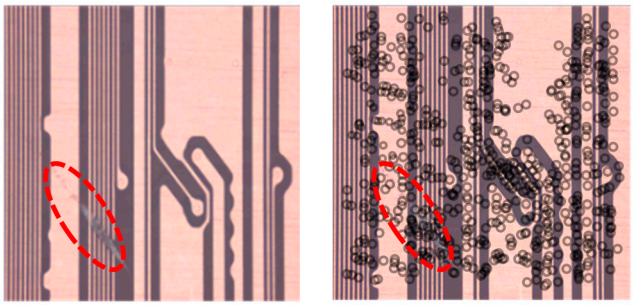
Figure 4. Flowchart of the proposed PCB inspection method. WKDE, weighted kernel density estimation.

Hence, we assign a normal or fault class to the features using the coordinates of the features within a defective or normal area. We then learn the normal and fault properties using the multivariate vector and feature class and calculate the probability of each feature being classified as a fault class. We then generate a new weighted kernel density estimation (WKDE) map. Kernel density estimation (KDE) is a technique for estimating the density based on the coordinates. That is, KDE is a technique that considers the density of the features. To maximize the effectiveness of the features, the probability of affecting the spatial density is given as a weight to the KDE. Thus, we consider not only the properties

but also the densities of the features. After generating a WKDE map, we can predict the fault area based on the WKDE value. The proposed method is described in more detail in the following subsection.

3.1. SURF-Based Feature Extraction and Class Assignment

As the first step of the proposed method, SURF digitizes a PCB image as a vector. SURF then detects the features and extracts the descriptors in the form of a vector as mentioned in the previous section. Figure 5a shows an example of an original PCB image. The circuits in the area inside the dotted line have a defective edge, whereas the other circuits have a normal edge. The extracted features are shown in the circles in Figure 5b. As Figure 5b illustrates, the features are obtained identically without discriminating between a defective edge and a normal edge. In other words, SURF cannot distinguish between a defective and a normal edge. Thus, we should learn the fault pattern of the features by setting the normal and fault areas and assigning classes to each feature.



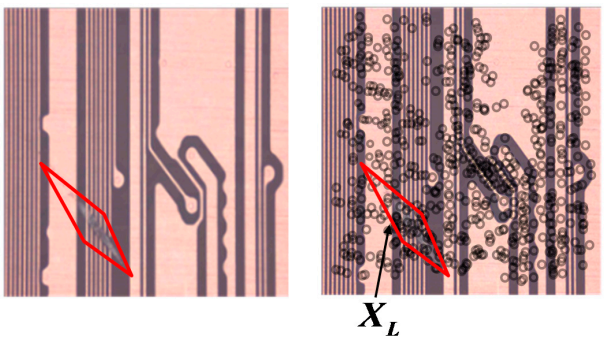
(**a**) (**b**)

Figure 5. An example of SURF-based feature extraction (**a**) Original image; (**b**) A resultant image after extracting the features of the PCB.

As shown in Figure 6a, we define a lozenge area with four points so that the fault area can be visually represented in the clearest manner. We can then consider the fault occurring in the lozenge area. We assign the features within the lozenge area in Figure 6b to the fault class. The remaining features from areas other than the lozenge area are designated as being of a normal class through Equation (8).

$$c(x_i) = \begin{cases} Fault\ class,\ if\ x_i \in X_L \\ Normal\ class,\ otherwise \end{cases}, \tag{8}$$

where x_i indicates each feature composed using a multivariate vector and X_L is a set of features within the lozenge area.



(**a**) (**b**)

Figure 6. An area selection for a class assignment: (**a**) A fault occurring in the lozenge area; (**b**) The features within the lozenge area are the fault class and the remaining features are the normal class.

3.2. Learning the Fault Pattern and Calculating the Probability Based on Random Forests

Given the class-assigned features, the following step is to handle fault patterns using a machine-learning algorithm. This method is based on the random forests method, which is a well-known ensemble learning model for classification. We first learn the fault pattern of the features. We use a 136-dimension input vector, X_i, composed using extended descriptors and the information of each feature, such as its location and class, for a more precise description. The extended descriptors are computed by summing d_x and $|d_x|$ separately for $d_y < 0$ and $d_y \geq 0$. Similarly, $\sum d_y$ and $\sum |d_y|$ are split for $d_x < 0$ and $d_x \geq 0$. Thus, the number of vector dimensions of the descriptors doubles to 128. After building the learning model by training the properties of the features, we can find the probability p_i that each feature of an inspected image will be classified as a fault, which is calculated using Equation (7). Because the classes of the features are extremely imbalanced, most features are classified as normal. That is, the probability that the features will be estimated as a fault is considerably less than the probability that they will be predicted as normal. Consequently, because it is difficult to detect a fault area using only the results of the random forests, we should consider the density in addition to the map of the probability of being predicted as a fault.

3.3. Generation of WKDE Map by Weighting the Probability

This section describes how to apply the probability calculated in the previous section to generate a weighted kernel density estimation (WKDE) map, a technique that reflects the kernel density estimation (KDE). KDE is a probability density estimation method using a kernel function. In addition, it is a nonparametric method that is even applicable to high-dimensional data [23]. The representative kernel functions include Gaussian, uniform, and Epachenikov functions [24]. This paper uses a Gaussian kernel function to estimate the density.

After calculating the probability, we generate a WKDE map by giving the weight, w_i, to the kernel function in order to complement any disadvantages as mentioned earlier. Because the PCB features have x- and y-axis coordinates, this method uses the weighted multivariate kernel density estimation.

Let $\{x_1, y_1\}, \{x_2, y_2\}, \cdots, \{x_m, y_m\}$ be the coordinates of the PCB features. The WKDE value is denoted through Equation (9) [25].

$$\hat{f}(x, y) = \frac{1}{mh} \sum_{i=1}^{m} w_i K\left\{ \frac{(x, y) - (x_i, y_i)}{h} \right\}, \tag{9}$$

where m is the total number of PCB features, h is a scaled kernel that controls the smoothness of the estimate, and w_i is the probability of being classified as a fault. In addition, K is a Gaussian kernel function.

KDE and WKDE maps were drawn using the features extracted from the PCB image in Figure 5 as shown in Figure 7. The x and y axes in the KDE and WKDE maps represent the coordinates of the features. The z axis of the KDE map indicates the KDE value, which is calculated based on the coordinates of each feature, whereas the z axis of the WKDE map is the WKDE value obtained by weighting the probability to the KDE. Because the KDE considers only the density, it has a high KDE value in areas where many features are extracted regardless of the defective area as shown in Figure 7a.

Consequently, it is difficult to distinguish between a normal area and a fault area on the KDE map. On the other hand, the WKDE value, which reflects the probability, increases when the probability of each feature being classified as a fault is higher than the probability of the other features. Otherwise, the WKDE value decreases because there is a relative difference between the probabilities. The area marked with a circle in Figure 7b has a relatively high WKDE value owing to the weight. That is, because the marked area is densely concentrated with fault features, the WKDE value of this area is higher than the KDE value. In addition, the KDE values in the remaining areas are randomly distributed, but decrease significantly after being weighted. Thus, we can predict the marked area as a fault area.

Figure 8 shows a three-dimensional (2-D) WKDE map overlapping the original image (Figure 5a). In the 3-D WKDE map, it is difficult to intuitively recognize the coordinates of a feature. Thus, we convert the 3-D image into a two-dimensional (2-D) image to prove that the marked area in Figure 7b is consistent with the real fault area. The area within the dotted line in Figure 8 indicates where the actual faults occur, whereas the shaded area is an area predicted as a fault because the WKDE value exceeds the threshold. As Figure 8 shows, we can confirm that the shaded area matches the dotted area. Therefore, we monitor the WKDE value and predict the features as being a fault if the WKDE value exceeds the threshold. In this paper, we set a WKDE value of higher than 1% as the threshold. However, the threshold changes according to the process yield or to past experience.

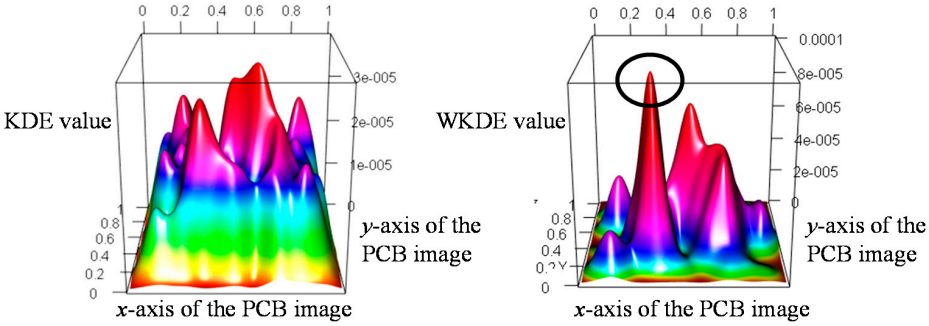


Figure 7. The kernel density estimation (KDE) and WKDE map in three-dimensions (3-D) for a PCB image: (**a**) KDE map; (**b**) Proposed WKDE map.

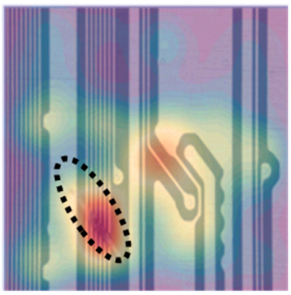


Figure 8. Two-dimensional (2-D) WKDE map on an original image.

4. Experimental Results

The present experiment used PCB image data to verify the proposed method. PCB images were collected from a battery manufacturer in Korea. We obtained 10 PCB images by selecting a fault image with a particular scratch property. As shown in Figure 9, the scratch fault, which is indicated by the dotted line, has a fine and thin form, indicating that the circuit may be broken. Because a scratch defect may prevent a current flow within the circuit, it has a significant impact on the PCB quality. Therefore, this paper analyses PCB images with a scratch defect.

We first extracted the features in the 10 PCB images using SURF. We then assigned a class to each feature by designating the defective area and organized the 136-dimension input vector including information such as the coordinates and class. The data were then divided into the test set and training set for learning purposes. This experiment applied a 10-fold cross validation using the 10 images. For example, nine images are used for training, and the remaining image is used for testing. Next, we obtained the WKDE value for each of the 10 PCB images as mentioned in Section 3.

To diagnose a fault area, statistical process control (SPC) can be applied to monitor the WKDE value. SPC is a method of quality control that uses statistical methods based on the distribution

of data [26]. However, there is no assumption regarding the distribution of the features. For this reason, this research cannot utilize SPC. Hence, a binary classification was applied based on a single continuous WKDE value to examine whether a feature is a fault. The features were classified by comparing the WKDE value with a threshold x^*, which is called the cut-off value. A feature was classified as a fault if $\hat{f}(x, y) \geq x^*$ and was classified as normal otherwise.

The appropriate threshold can be chosen based on various criteria [27]. Thus, we conducted the experiments using a variety of thresholds, and adopted an appropriate threshold based on a receiver operating characteristic (ROC) curve, which is a tool for demonstrating the performance of a classifier. An ROC curve is a threshold-independent technique for evaluating the performance of a model and represents the relationship between the model's sensitivity and specificity [27]. In this study, the ROC curve was drawn based on the true- and false-negative rates. The ROC curve for a good model achieves a high true-negative rate whereas the false-negative rate is relatively small. Therefore, the performance of a model can be evaluated by comparing the area under the ROC curve (AUROC). Robust models have an AUROC of close to 1.0, whereas poorer models have an AUROC near 0.5, and worthless models have a value of less than 0.5 [28].

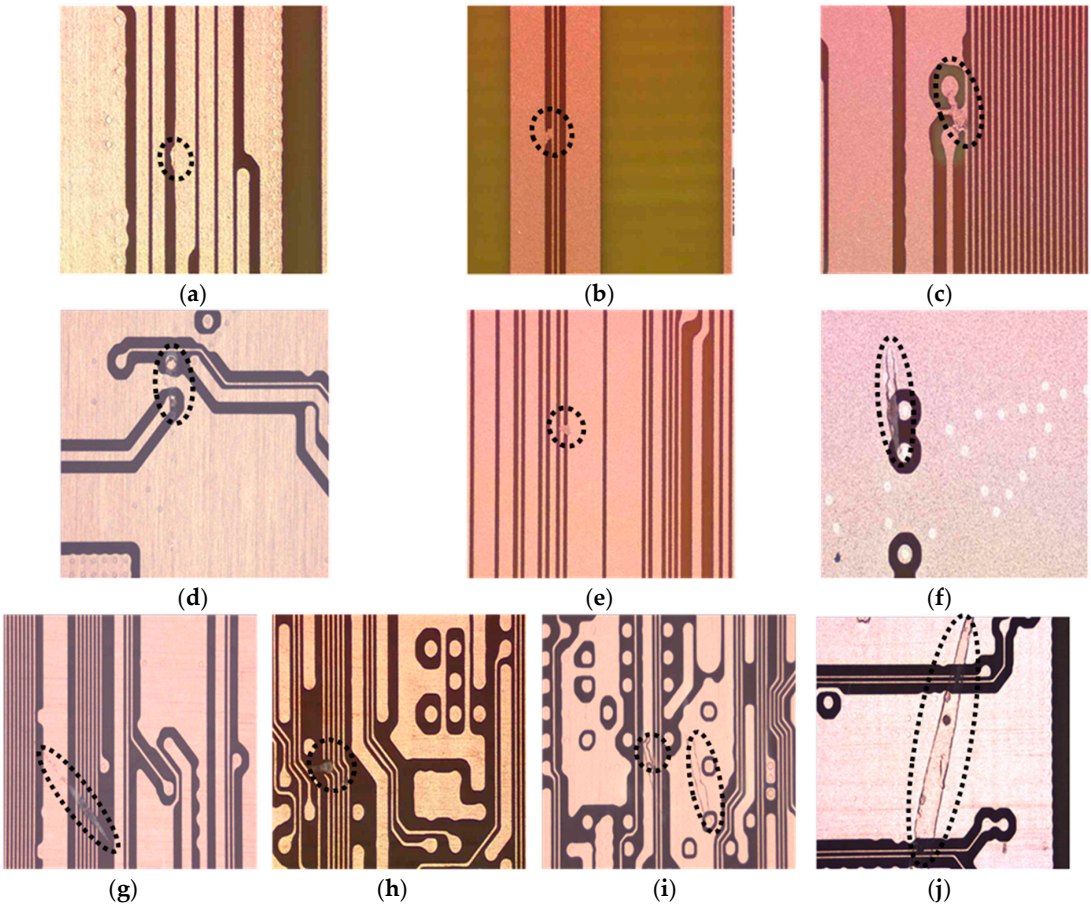


Figure 9. Example images of the 10 PCB images used: (**a**)–(**j**) The area marked with a circle or an ellipsoid on the dotted line is where the scratch-fault occurred.

To demonstrate that both the probability and density of the features have to be considered for detecting a fault, this paper compares the performances of three different methods: (1) monitoring only the probability (Method 1), (2) using a KDE map (Method 2), and (3) using the proposed WKDE map (Method 3). Each method applies binary classification to detect a fault. Method 1 detects a fault area by monitoring only the probability obtained by the random forests. An area is determined as a fault area when the probability of the feature being a fault exceeds a certain threshold. That is,

this method does not consider the density of the features. Method 2 identifies a defect by considering only the density of the features without taking into account the probability. Thus, it monitors the KDE value. Finally, Method 3, the proposed method, detects a fault by monitoring the WKDE value. This method considers both the properties of the features and their density. Finally, we draw an ROC curve to compare the performances of the three methods. Figure 10 shows the ROC curves for the three methods. A method in which the AUROC is close to 1.0 detects a PCB fault more precisely. As can be seen, the AUROC of Method 1 is larger than that of Method 2 for 6 of the 10 images. However, the AUROC of Method 1 is less than that of Method 3 for all 10 images. Thus, the performance of Method 1 is not proper for detecting faults. Although Method 2 outperforms Method 1 for certain images, it has a poorer level of performance than Method 3. On the other hand, Method 3 outperforms the other methods, and its AUROC is close to 1.0 for all images. Therefore, Method 3 is appropriate for detecting a scratch fault on a PCB.

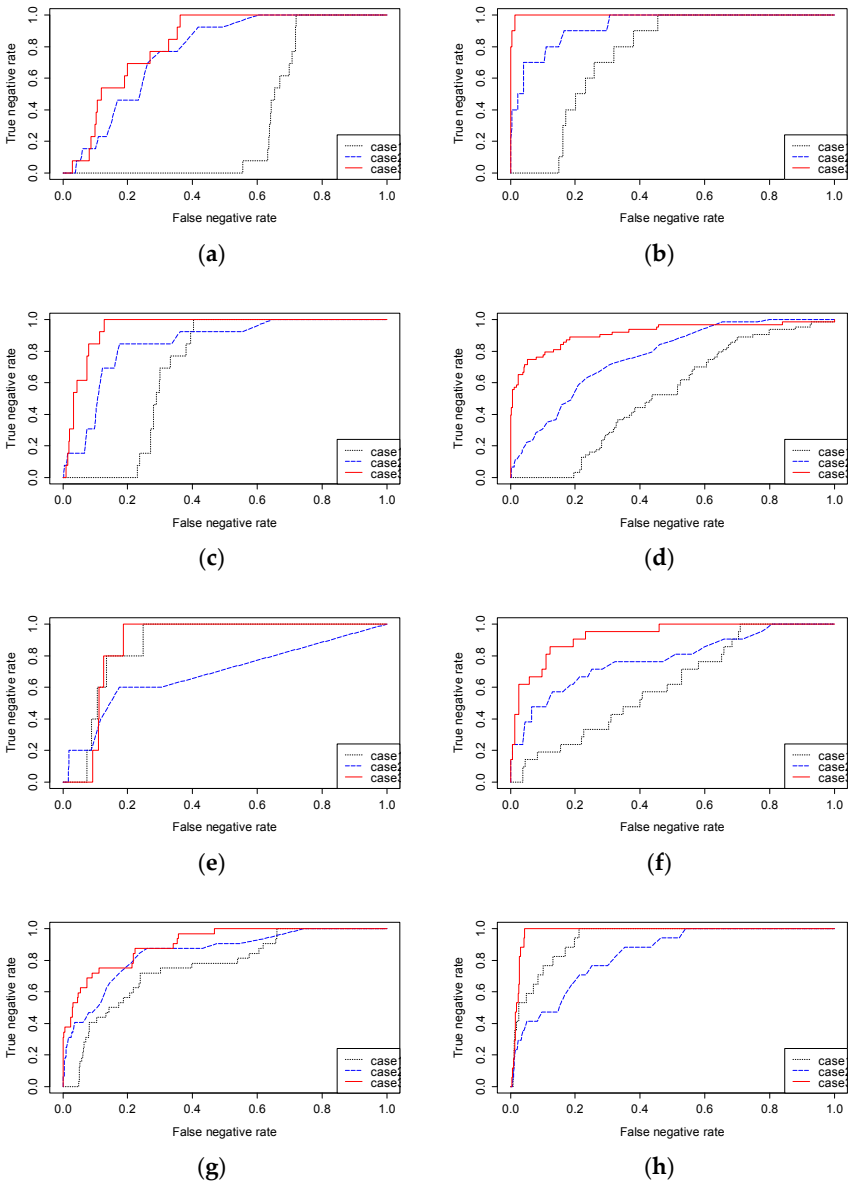


Figure 10. *Cont.*

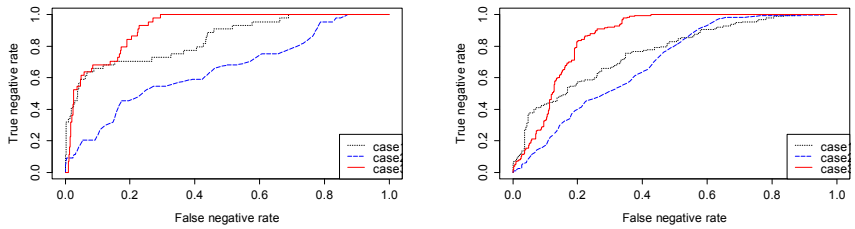


Figure 10. Performance comparison of three different methods for 10 scratch-fault images: (**a**)–(**j**) For the 10 PCB images, the mean AUROC of the Method 1 is 0.70, the Method 2 is 0.78, and the Method 3 is 0.91. The Method 3 considering both probability and density has better detection performance in all 10 images than the other method considering only one property.

5. Conclusions

In this paper, we proposed a new non-referential method by learning the fault pattern and generating a WKDE map. This method can learn various fault patterns regardless of the type of defect. Thus, the proposed method allows for a flexible PCB inspection without limitations regarding the specific type of fault. Furthermore, it can be extended to deal with unknown faults for PCB inspection. The performance of the proposed method is demonstrated by comparing the ROC curve for the three methods presented above. In addition, it was found that considering both the probability and density of the features is effective for detecting a scratch fault. Thus far, our method has dealt only with the 10 scratch faults due to security issues, and it needs to be applied to other PCB images with more data and various fault patterns. In addition, using a clustering technique, further work needs to be conducted to redefine the fault type when it is not specified, and an appropriate detection method for each type of defect should be studied. Although this paper used SURF to detect robust features, other image processing techniques can be applied. In other words, the classification accuracy can be further enhanced using another algorithm that includes more detailed edge information. The classification performance of the proposed algorithm is proved by the ROC curve. In the future, it is necessary to study the threshold selection method and its performance evaluation method for practical application in the future.

Author Contributions: E.Y. designed and implemented the algorithm to solve the defined problem. S.H.P. and C.-S.P. performed the experiments and data analyses. J.-G.B. validated the proposed algorithm and guided the whole research. All authors read and approved the final manuscript.

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2016R1A2B4013678). This work was also supported by BK21 Plus (Big Data in Manufacturing and Logistics Systems, Korea University).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Malge, P.S.; Nadaf, R.S. A survey: Automated visual PCB inspection algorithm. *Int. J. Eng. Res. Technol.* **2014**, *3*, 1. Available online: https://www.ijert.org/browse/volume-3-2014/january-2014-edition?start=20 (accessed on 2 June 2018).
2. Wu, W.Y.; Wang, M.J.J.; Liu, C.M. Automated inspected of printed circuit boards through machine vision. *Comput. Ind.* **1996**, *28*, 103–111. [CrossRef]
3. Sundaraj, K. PCB inspection for missing or misaligned components using background subtraction. *WSEAS Trans. Inf. Sci. Appl.* **2009**, *6*, 778–787. Available online: https://dl.acm.org/citation.cfm?id=1558809 (accessed on 2 June 2018).
4. Chauhan, A.P.S.; Bhardwaj, S.C. Detection of bare PCB defects by image subtraction method using machine vision. In Proceedings of the World Congress on Engineering, London, UK, 6–8 July 2011; Volume 2, pp. 103–111.

5. Moganti, M.; Ercal, F.; Dagli, C.H.; Tsunekawa, S. Automatic PCB inspection algorithms: A survey. *Comput. Vis. Image Understand.* **1996**, *63*, 287–313. [CrossRef]
6. Acciani, G.; Brunetti, G.; Fornarelli, G. Application of neural networks in optical inspection and classification of solder joints in surface mount technology. *Int. IEEE Trans. Ind. Inform.* **2006**, *2*, 200–209. [CrossRef]
7. Huang, S.H.; Pan, Y.C. Automated visual inspection in the semiconductor industry: A survey. *Comput. Ind.* **2015**, *66*, 1–10. [CrossRef]
8. Kim, H.W.; Yoo, S.I. Defect detection using feature point matching for non-repetitive patterned images. *Pattern Anal. Appl.* **2014**, *17*, 415–429. [CrossRef]
9. Ye, Q.-Z.; Danielsson, P.E. Inspection of printed circuit boards by connectivity preserving shrinking. *IEEE Trans. Pattern Anal. Mach. Intell.* **1988**, *10*, 737–742. [CrossRef]
10. Elbehiery, H.; Hefnawy, A.; Elewa, M. Surface defects detection for ceramic tiles using image processing and morphological techniques. *WEC* **2007**, *1*, 1488–1492. Available online: https://scholar.waset.org/1307-6892/15176 (accessed on 2 June 2018).
11. Tsai, D.M.; Su, Y.J. Non-referential, self-compared shape defect inspection for bond pads with deformed shapes. *Int. J. Prod. Res.* **2009**, *47*, 1225–1244. [CrossRef]
12. Rosten, E.; Drummond, T. Machine learning for high-speed corner detection. In Proceedings of the 9th European Conference on Computer Vision-ECCV, Graz, Austria, 7–13 May 2006; pp. 430–443.
13. Pernkopf, F. Detection of surface defects on raw steel blocks using Bayesian network classifiers. *Pattern Anal. Appl.* **2004**, *7*, 333–342. [CrossRef]
14. Oliveira, R.B.; Papa, J.P.; Pereira, A.S.; Tavares, J.M.R. Computational methods for pigmented skin lesion classification in images: Review and future trends. *Neural Comput. Appl.* **2018**, *29*, 613–636. [CrossRef]
15. Ma, Z.; Tavares, J.M.R. Effective features to classify skin lesions in dermoscopic images. *Expert Syst. Appl.* **2017**, *84*, 92–101. [CrossRef]
16. Oliveira, R.B.; Pereira, A.S.; Tavares, J.M.R. Computational diagnosis of skin lesions from dermoscopic images using combined features. *Neural Comput. Appl.* **2018**. [CrossRef]
17. Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. Speeded-up robust features (SURF). *Comput. Vis. Image Understand.* **2008**, *110*, 346–359. [CrossRef]
18. Su, J.; Xu, Q.; Zhu, J. A scene matching algorithm based on SURF feature. In Proceedings of the International Conference on Image Analysis and Signal Processing (IASP), Huangzhou, China, 9–11 April 2010; pp. 434–437.
19. Strang, G.; Nguyen, T. *Wavelets and Filter Banks*, 2nd ed.; Wellesely-Cambridge Press: Wellesley, MA, USA, 1996.
20. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
21. Ohn, S.Y.; Chi, S.D.; Han, M.Y. Feature Selection for Classification of Mass Spectrometric Proteomic Data Using Random Forest. *J. Korea Soc. Simul.* **2013**, *22*, 139–147. [CrossRef]
22. Li, C. Probability Estimation in Random Forests. Master's Thesis, Utah State University, Logan, UT, USA, 2013.
23. Thurstain-Goodwin, M.; Unwin, D. Defining and delineating the central areas of towns for statistical monitoring using continuous surface representations. *Trans. GIS* **2000**, *4*, 305–317. [CrossRef]
24. Kurata, E.; Mori, H. Short-term load forecasting using informative vector machine. *Electr. Eng. Jpn.* **2009**, *166*, 23–31. [CrossRef]
25. Anderson, T.K. Kernel density estimation and K-means clustering to profile road accident hotspots. *Accid. Anal. Prev.* **2009**, *41*, 359–364. [CrossRef] [PubMed]
26. Fadel, H.K.; Holloway, L.E. Using SPC and template monitoring method for fault detection and prediction in discrete event manufacturing systems. In Proceedings of the IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, Cambridge, MA, USA, 17 September 1999.
27. Kang, S.; Cho, S.; An, D.; Rim, J. Using wafer map features to better predict die-level failures in final test. *IEEE Trans. Semicond. Manuf.* **2015**, *28*, 431–437. [CrossRef]
28. Freeman, E.A.; Moisen, G.G. A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa. *Ecol. Model.* **2008**, *217*, 48–58. [CrossRef]