

Article

Semantic Segmentation of a Printed Circuit Board for Component Recognition Based on Depth Images

Dongnian Li , Changming Li, Chengjun Chen * and Zhengxu Zhao

School of Mechanical & Automotive Engineering, Qingdao University of Technology, Qingdao 266525, China; dongnianli@qut.edu.cn (D.L.); changmingli168@163.com (C.L.); zhaozhengxu@qut.edu.cn (Z.Z.)

* Correspondence: chencj@qut.edu.cn

Received: 19 August 2020; Accepted: 14 September 2020; Published: 17 September 2020



Abstract: Locating and identifying the components mounted on a printed circuit board (PCB) based on machine vision is an important and challenging problem for automated PCB inspection and automated PCB recycling. In this paper, we propose a PCB semantic segmentation method based on depth images that segments and recognizes components in the PCB through pixel classification. The image training set for the PCB was automatically synthesized with graphic rendering. Based on a series of concentric circles centered at the given depth pixel, we extracted the depth difference features from the depth images in the training set to train a random forest pixel classifier. By using the constructed random forest pixel classifier, we performed semantic segmentation for the PCB to segment and recognize components in the PCB through pixel classification. Experiments on both synthetic and real test sets were conducted to verify the effectiveness of the proposed method. The experimental results demonstrate that our method can segment and recognize most of the components from a real depth image of the PCB. Our method is immune to illumination changes and can be implemented in parallel on a GPU.

Keywords: PCB; component recognition; semantic segmentation; pixel classification; random decision forest; depth image

1. Introduction

The printed circuit board (PCB) is an important part of modern electronic products. The quality of the PCB determines the quality of the product. In particular, the quality of the assembly of the components on the PCB directly influences the performance and service life of electronic equipment. Traditional manual inspection of PCBs is inefficient, and has a high error rate, which makes it difficult to adapt to industrial situations. Automatic inspection based on machine vision is increasingly applied in fields related to PCB inspection. Thus, automatic visual inspection of PCBs has become a popular research topic in the area of industrial inspection. In addition, the flexibility of modern automatic production systems could be improved significantly if the production equipment was able to effectively and automatically recognize and inspect the PCB.

In recent years, many researchers have carried out studies on PCB inspection based on machine vision. Among them, a lot of studies [1–11] have focused on the inspection of solder joints, which is both important and challenging. Researchers have proposed a number of methods for solder joint inspection that use neural networks [1,2], fuzzy rules [3], Boolean rules [4], deep learning [5], support vector machines [6], decision trees [7], principle component analysis [8], modal analysis [9], etc. Component placement inspection is another significant and challenging problem, and it is the basis of other PCB inspections, such as the inspection of solder joints. Many defects are caused by errors in PCB component placement, such as missing or misaligned components, or the incorrect rotation of the components. In addition, the techniques and methods derived from the research on component placement inspection

can also be used in automated PCB recycling. With regard to the inspection of component placement, research needs to be conducted on segmentation, detection, and recognition of the components in the PCB to locate and identify the components. Many studies [12–24] have been carried out to solve this problem. Crispin et al. [12] proposed a genetic algorithm template-matching method to locate and identify PCB components from images to the component placement. They used grey-model templates for PCB components. Zeng et al. [13] proposed a method to locate PCB components according to the color distribution of solder joints. By using three layers of ring-shaped LEDs as illumination, they recognized and located the solder joints based on their color distribution patterns. Then, they located each component by clustering all the solder joints of the component. The method uses PCB color images as input. Herchenbach et al. [14] proposed a framework for segmentation and classification of through-hole components on PCBs to enable the automatic recycling of PCBs by using both RGB (Red Green Blue) and depth frames from the Microsoft Kinect sensor as input. They used a multi-threshold approach to create segmentation hypotheses of components and searched for the best hypothesis using an optimization step. However, they misclassified a great number of PCB segments as components. Li et al. [15] divided the surface mounted devices (SMDs) mounted on PCBs into two groups: small devices and integrated circuits (ICs). They used assembly print-based segmentation for small devices and color distribution-based segmentation for ICs, respectively. Lim et al. [16] used a convolution neural network (CNN) for classification of SMDs mounted on PCBs. They extracted the device regions from device images for classification. However, the segmentation of device images from PCB images was not presented in [16].

In this paper, we propose a PCB semantic segmentation method based on depth images, to segment and recognize components in the PCB through pixel classification. Machine vision based on depth images has high robustness since it is not influenced by the illumination and the texture of the target object [25]. Pixel classification methods based on depth images have been applied in the field of human body pose recognition [26–28] and hand gesture recognition [29–32]. Shotton et al. [26,27] constructed a large synthetic depth image database with automatic pixel labeling by using motion capture data of human bodies. They used the database to train a random forest and estimated the approximate poses of different parts of the human body through pixel classification based on depth images. Keskin et al. [29,30] estimated hand gestures from depth images by using a pixel classification method with random forests.

In this paper, a random forest pixel classifier based on depth images is applied to the semantic segmentation of a PCB, which segments and recognizes the components in the PCB through pixel classification. We first set up an image training set and an image test set for the PCB with graphic rendering. Then, depth difference features were extracted from depth images in the training set, and the random forest pixel classifier was trained by the depth difference features, thus establishing a mapping relationship from depth difference features to pixel classification labels. Finally, by using the constructed random forest pixel classifier, we carried out semantic segmentation for the PCB to segment and recognize components in the PCB through pixel classification. The proposed method can be further used to inspect the placement of the components in the PCB. In addition, the proposed method can be used in automated PCB recycling [14,15,33].

The main contributions of this paper are as follows:

- A method for segmenting and recognizing components on a PCB is proposed by using a random forest pixel classifier based on depth images.
- A technique is developed to synthesize the PCB training image set automatically based on the graphic rendering engine OpenSceneGraph (OSG).
- A mechanism for extracting depth difference features is constructed by using a series of concentric circles centered at the depth pixel to be predicted.

This paper is organized as follows: Section 2 outlines the framework of the proposed method. Section 3 describes the construction of the image training set and test set. Section 4 describes the

extraction of depth difference features. The training and predicting process of the random forest pixel classifier is presented in Section 5. The experimental results and discussions are provided in Section 6. Section 7 concludes the paper and presents the future work.

2. The Framework of the Method

In this paper, semantic segmentation of the PCB was performed with a pixel classification method using random forests to achieve segmentation and recognition of the components on the PCB. The technological framework of the method is shown in Figure 1.

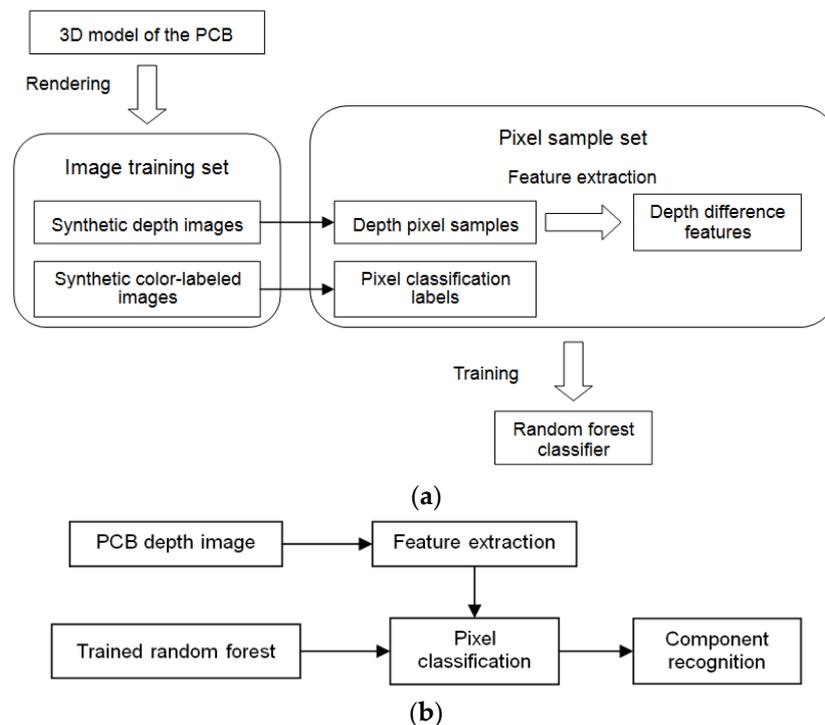


Figure 1. The framework of the proposed method. (a) Training. (b) Testing.

First, we set up an image training set and an image test set for the PCB, which covered the depth image samples and the corresponding color-labeled image samples. The image training set was synthesized by computer graphic rendering based on a prebuilt 3D model of the PCB. The image test set includes both synthetic image samples and real image samples. Pixels in the training images were randomly selected as pixel training samples. The depth difference features of the selected pixels were extracted from the depth images in the training set, and the classification labels of the pixels were extracted from the corresponding color-labeled images.

Then, a random forest classifier was trained by the extracted depth difference features, which established a mapping relationship from the depth difference features to the pixel classification labels. The trained random forest classifier was tested by predicting the depth images in the test set, and the optimal configuration of parameters for the random forest were acquired by repeated training and testing. Finally, semantic segmentation of the PCB was performed by the established random forest pixel classifier to segment and recognize the components in the PCB. Thus, the placement of the components in the PCB could be inspected.

3. Construction of the Image Training Set and Test Set

In this paper, 3D computer graphic rendering was used to synthesize the image sample set required to train the random forest classifier. First, we constructed a 3D model of the PCB by using a CAD modeling software, which was then put into the visualization modeling software MultiGen-Paradigm

Creator through the .obj middle file format. In the Creator, components denoted $\{P_i | i = 1, \dots, N\}$ of the PCB were labeled in different colors. Each color corresponds to one classification label l_i . N is the number of components in the PCB. Next, the PCB model with color labels was loaded into the open-source graphic rendering engine, OSG. The pose states of the PCB were sampled within a certain range in the pose space that the PCB lies in, thus obtaining the pose sample set $\{s_j | j = 1, \dots, M\}$. M is the number of pose samples of the PCB. For each pose sample s_j , a depth image (Figure 2a) and its corresponding color-labeled image (Figure 2b) of the PCB under this pose were synthesized with OSG off-screen rendering through a frame buffer object (FBO). In this way, we constructed a depth image sample set and a corresponding color-labeled image sample set, which are required to train the random forest classifier.

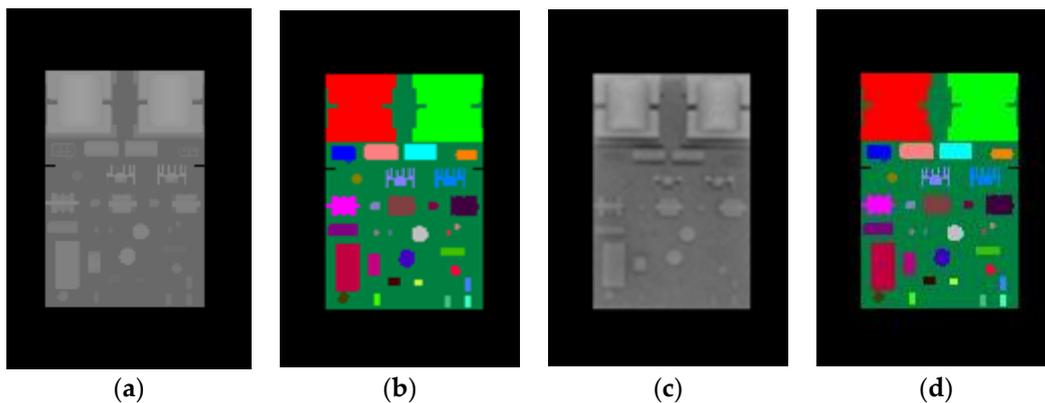


Figure 2. Depth images and corresponding color-labeled images of the printed circuit board (PCB). (a) Synthetic depth images. (b) Synthetic color-labeled images. (c) Real depth images. (d) Real images labeled with colors.

The synthetic depth images were generated by data stored in the GPU depth buffer, the Z-buffer, while the synthetic color-labeled images were generated by data stored in the GPU color buffer. The image test set for the random forest included both synthetic image samples and real image samples. Like the training images, the synthetic test images were also generated through computer graphic rendering. To collect the real images for testing, we acquired the depth images of the real PCB by a Kinect 2.0 depth sensor (Figure 2c) and manually obtained the corresponding color-labeled images by using an image processing software to label the acquired depth images with colors (Figure 2d).

4. Extraction of Depth Features

During feature extraction, a certain number of pixels were selected randomly from each depth image in the image training set to generate the pixel sample set. Depth difference features of the pixel samples were used to train the random forest pixel classifier. Given a depth pixel x , the depth difference feature f_θ is defined as

$$f_\theta = d\left(x + \frac{u_1}{d(x)}\right) - d\left(x + \frac{u_2}{d(x)}\right) \quad (1)$$

where the parameters $\theta = (u_1, u_2)$ describe two 2D pixel offset vectors (u_1 and u_2) relative to the depth pixel x , and $d(x)$ is the corresponding depth value of depth pixel x in the depth image. Two offset vectors u_1 and u_2 are normalized by dividing with $d(x)$, ensuring depth invariance of the feature value, f_θ . In other words, f_θ is not influenced by the distance from the object to the camera. If the offset pixel lies in the background region or beyond the image boundaries, the corresponding depth value is set to $C + d(x)$, where C is a large constant and $d(x)$ is the depth value of the original depth pixel x . These depth difference features only provide weak classification signals, but the combination of these features with a random forest classifier will enable the recognition of different parts in an object.

In this paper, as shown in Figure 3, a set of concentric circles centered at depth pixel x were set up for depth feature extraction. A set of offset vectors were selected on each concentric circle, according to Equations (2) and (3).

$$\mathbf{A} = \left\{ \alpha \mid \alpha = \frac{2i\pi}{N}, i = 1, 2, \dots, N, N = 8n \right\} \quad (2)$$

$$\mathbf{U} = \{ \mathbf{u} \mid \mathbf{u} = M(\cos \alpha, \sin \alpha), \alpha \in \mathbf{A}, M = mn \} \quad (3)$$

where \mathbf{A} is a set of N equal angles on the n th concentric circle, and \mathbf{U} is a set of offset vectors that are selected on the n th concentric circle. $M = mn$ is the modulus of the offset vector \mathbf{u} , which is also the radius of the n th concentric circle and m is a constant basis of modulus M . In this paper, eight concentric circles were set up, so $n = 1, 2, \dots, 8$. A total of 288 offset vectors were selected according to Equations (2) and (3). The extracted depth difference features include unitary features and binary features. When extracting the unitary depth difference features, we let $u_2 = 0$ and calculated the depth difference between each offset point and the depth pixel x . The 288 offset vectors on the eight concentric circles provided 288 unitary features. When the binary depth difference features were extracted, 1800 pairs of offset vectors were chosen randomly from the 288 offset vectors, and the depth difference between two offset points was calculated. Therefore, a total of 2088 depth difference features were extracted for each depth pixel x for classification.

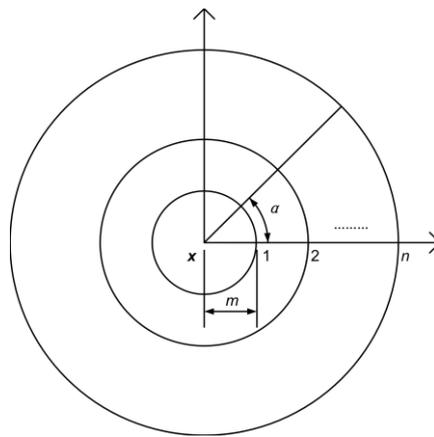


Figure 3. The concentric circles for depth feature extraction.

5. Random Decision Forest

In this paper, semantic segmentation of the PCB was performed by using a random forest pixel classifier to segment and recognize components on the PCB. The random forest classifier is trained by depth difference features extracted from synthetic depth images. Random decision forest [34,35] is a fast and effective multiclass classifier that comprises multiple decision trees. As shown in Figure 4, each decision tree is composed of many branch nodes and leaf nodes. Each branch node corresponds to a feature f_θ and contains a threshold τ . Each leaf node contains a probability distribution of classification label l_i . When pixel x is predicted by the t th decision tree for classification, the depth difference feature of pixel x is compared against the corresponding threshold at each branch node of decision tree t . Based on the results of the comparison, pixel x is classified to the left or right sub-node and finally assigned to a leaf node. The probability distribution $p_t(l_i|x)$ corresponding to this leaf node makes a prediction for pixel x . The probability distributions gained by all the decision trees in the forest are averaged at the end to make the final decision for pixel x according to Equation (4).

$$p(l_i|x) = \frac{1}{T} \sum_{t=1}^T p_t(l_i|x) \quad (4)$$

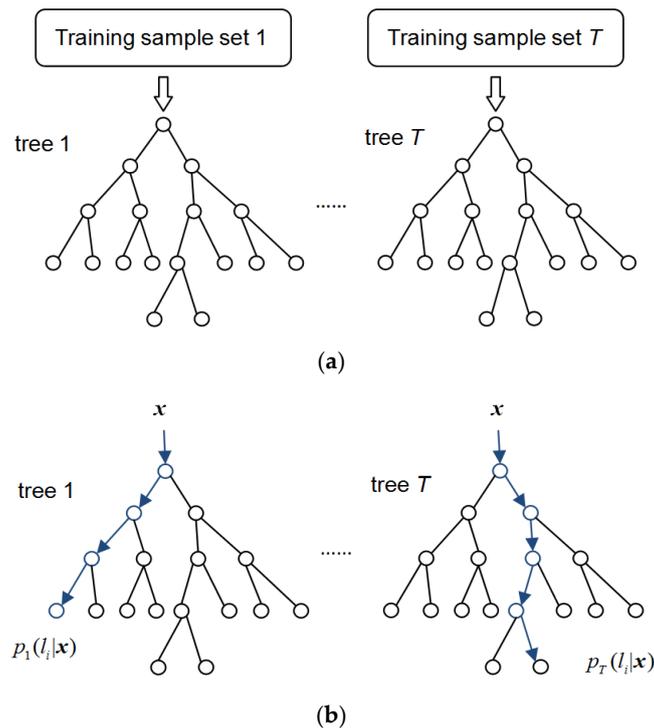


Figure 4. The training and predicting process of the random decision forest. (a) The training process. (b) The predicting process.

A random forest trains and predicts samples by several random decision trees, which can effectively avoid the over-fitting problem that happens frequently for single decision tree. The training samples used in each tree are sampled randomly with replacement from the general training sample set. The features used for the training of each branch node in the decision trees are sampled randomly with a certain proportion from all features without replacement. In this paper, the optimal parameters for random forest were determined by repeated training and testing. The parameters of concern here mainly include the maximum depth of the decision trees and the number of decision trees. The training and predicting processes of the random forest classifier are summarized as follows:

Training:

For tree $t = 1 - T$:

1. The training samples are selected randomly with replacement from the general training set, constructing a training sample set for the current decision tree. Start training from the root node.
2. For the current branch node, n features $\{f_\theta\}$ are selected randomly from all features without replacement. For each selected feature f_θ , samples in the sample set S of the current node are divided into the left subset S_L and the right subset S_R , according to the comparison between f_θ and the corresponding threshold τ , as shown in Equation (5). S_L and S_R are allocated to the left subnode and right subnode of the current node, respectively.

$$\begin{cases} S_L(\theta, \tau) = \{x | f_\theta(x) < \tau\} \\ S_R(\theta, \tau) = \{x | f_\theta(x) \geq \tau\} \end{cases} \quad (5)$$

From the n selected features, search for the feature f_θ and threshold τ that split the samples best according to Equations (6) and (7).

$$Gini(S) = 1 - \sum_{i=1}^N p_i^2 \quad (6)$$

$$(\theta, \tau)^* = \underset{\theta, \tau}{\operatorname{argmax}} (Gini(S) - \frac{|S_L|}{|S|} Gini(S_L) - \frac{|S_R|}{|S|} Gini(S_R)) \quad (7)$$

where $Gini(S)$ is the Gini coefficient of sample set S , and p_i is the proportion of class i samples in the sample set.

3. If the current node fits the termination criterion, it is set as a leaf node. This leaf node stores a probability distribution $p(l_i)$ of classification label l_i , which is calculated by the proportion of l_i samples in the current leaf node.
4. If some nodes have not been split or set as a leaf node, turn to Step 2.

Prediction:

For tree $t = 1 - T$:

1. Start from the root node of the current decision tree, classify sample x into the left subnode or right subnode of the current node according to the comparison between the corresponding feature f_θ and its threshold τ .
2. Repeat Step 1 until arriving at a leaf node of the decision tree. Output the corresponding probability distribution $p_t(l_i|x)$ to make the prediction of sample x for the current decision tree.

Finally, the probability distributions gained by all T trees are averaged, and the class label with the maximum probability is made as the output of the random forest.

6. Experiments and Discussions

The effectiveness of the proposed method was verified by semantic segmentation and component recognition of a typical PCB, an ultrasonic power board. A 3D model of the PCB was constructed with SolidWorks and then loaded into Multigen-Paradigm Creator to label its components with different colors. Then, an image training set for random forest training and the synthetic images for test were generated through the rendering of the PCB model with OSG. The real depth images of the PCB were acquired by a Kinect 2.0 depth sensor. The color-labeling of the acquired real depth images was achieved manually with an image processing software. The color labels of the components on the PCB are shown in Figure 5. The training program of the random forest pixel classifier was developed on the Linux platform by using the g++ compiler and OpenCV. The classifier was trained on a workstation with two Intel Xeon 10-core E5-2630 v4 2.2GHz CPUs, 64GB memory, and two Nvidia Titan X GPUs.

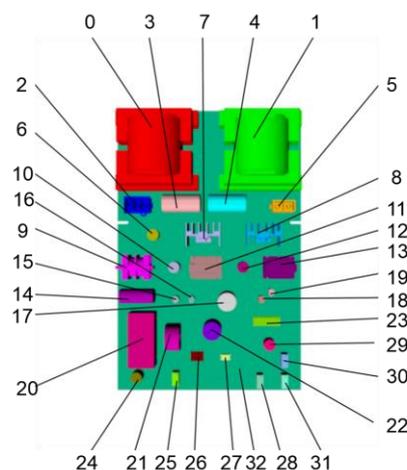


Figure 5. Color labels of the PCB components.

6.1. Selection of Training Parameters

To select the optimal training parameters for the random forest classifier, we designed the following four groups of experiments. In these experiments, the training samples were totally selected from the

synthetic image set. A total of 2000 pixel samples were randomly selected from each depth image in the image training set. The image test set was divided into a synthetic test subset and a real test subset. The synthetic test set is composed of 40 depth images that were randomly selected from the synthetic image set. The real test set is composed of 10 images that were randomly selected from the real image set. The experimental results are shown in Figure 6.

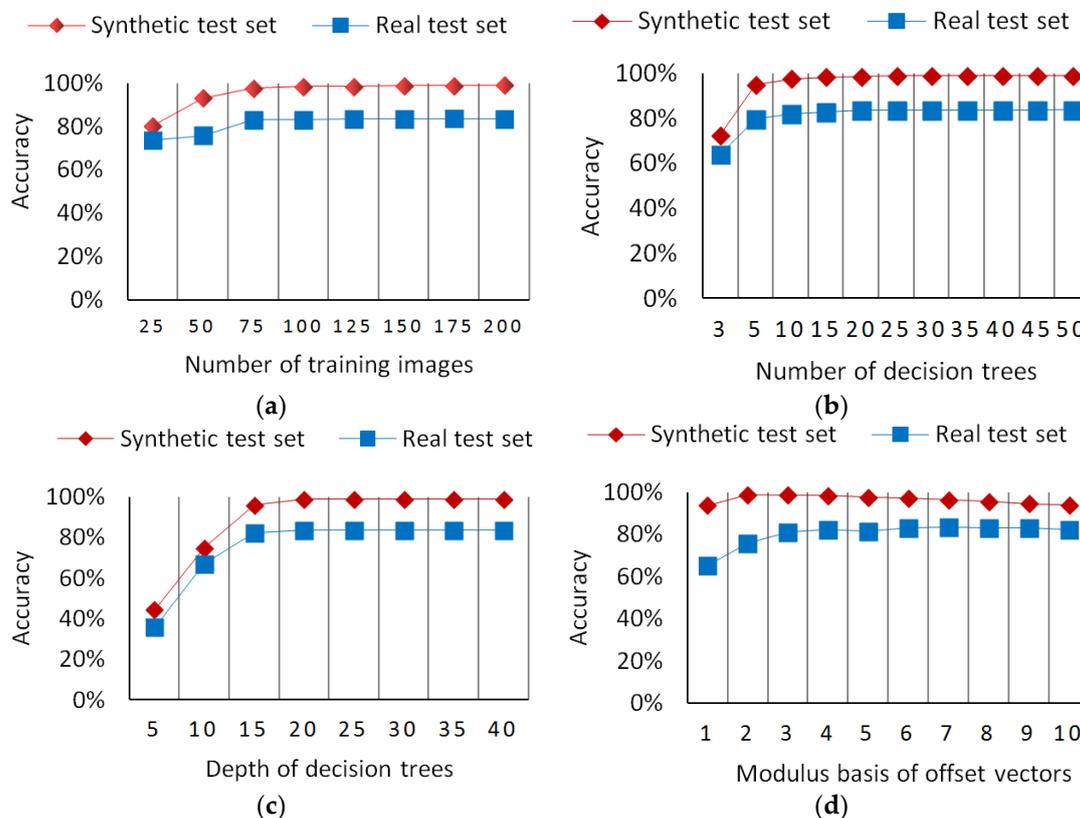


Figure 6. Training parameters vs. pixel classification accuracy. (a) Number of training images. (b) Number of decision trees. (c) Depth of decision trees. (d) Modulus basis of offset vectors.

6.1.1. Number of Training Images

A certain number of training images provide the basis by which the classifier obtains recognition capabilities. However, the more training images, the greater the required memory and training time, thus it is necessary to determine the optimal number of training images by experimenting with the random forest classifier. In these experiments, the number of decision trees and the depth of each decision tree were both set to 30. The number of training images was increased from 25 to 200 gradually in increments of 25 for each experiment. The experimental results are shown in Figure 6a. In the beginning, the pixel classification accuracy increases quickly with the increase in the number of training images. After 75 training images, the pixel classification accuracy of the trained random forest becomes stable.

6.1.2. Number of Decision Trees

The random decision forest has multiple random decision trees, which makes it capable of solving the problem of overfitting. The classification result of the random forest is determined by votes from all random decision trees. Thus, the number of decision trees can influence the performance of the random forest. If there are only a few decision trees, the random forest classifier is easily susceptible to overfitting. However, the more decision trees, the greater the training time and training cost. In this paper, we designed two groups of experiments on synthetic and real test images to determine the

optimal number of decision trees. In these experiments, a total of 200 synthetic depth images were used for training, and the depth of the decision trees was set to 30. The number of decision trees was increased gradually from 3 to 50. The experimental results are shown in Figure 6b. With the increase in the number of decision trees, the pixel classification accuracy of the trained random forest increases quickly in the beginning and then becomes stable after 10 decision trees.

6.1.3. Depth of Decision Trees

The depth of the decision trees can significantly influence the pixel classification accuracy of a random forest. Therefore, it is necessary to determine the optimal depth of the decision trees to achieve the best classification accuracy. Two groups of experiments on synthetic and real test images were designed to determine the optimal depth of decision trees. In these experiments, a total of 200 synthetic depth images are used for training, and the number of decision trees was set to 30. The depth of the decision trees was increased gradually from 5 to 40 in increments of 5 for each experiment. The experimental results for the depth of the decision trees are shown in Figure 6c. With the increase in the depth of the decision trees, the pixel classification accuracy of the trained random forest increases quickly in the beginning and becomes stable after the depth reaches 20.

6.1.4. Modulus Basis of Offset Vectors

The modulus values of the offset vectors reflect the scale of the offset vectors. Therefore, it is necessary to determine the optimal values of each offset vector modulus through experiments. The offset vector moduli $M = mn$, which are also the radii of the concentric circles used for extracting depth difference features, are integral multiples of the modulus basis m . Thus, we only designed experiments to determine the optimal value of the modulus basis m . Two groups of experiments were conducted in this study to test the influences of m on the pixel classification accuracy of the synthetic test set and the real test set. In these experiments, a total of 200 synthetic depth images were used for training. The number of decision trees and the depth of each decision tree were both set to 30. The value of m was increased from 1 to 10 in increments of 1 for each experiment. The experimental results are shown in Figure 6d. It can be seen from the results that the value of m influences the pixel classification accuracy of the synthetic test set slightly, but it could significantly influence the pixel classification accuracy of the real test set. For the synthetic test set, with an increase in m , the pixel classification accuracy of the random forest first increases and then decreases, and reaches a peak when m is set to 2. However, for the real test set, the pixel classification accuracy of the random forest obtains its highest value when m is set to 7.

6.2. PCB Semantic Segmentation and Component Recognition

In this paper, semantic segmentation of the PCB was performed through pixel classification by a random forest to segment and recognize each component in the PCB. To verify the effectiveness of the proposed method, we designed two experiments on synthetic and real test images. Specifically, 40 depth images and the corresponding label images were selected randomly from the synthetic image set as the synthetic test set. Ten depth images and the corresponding label images were selected randomly as the real test set from the real image set. In these experiments, a total of 200 synthetic depth images were used for training. For experiments on the synthetic test set, we constructed the image training set with the yaw angles of the PCB model sampled uniformly in the range of 360° and the pitch angles were sampled uniformly in the range of $(-30^\circ, 30^\circ)$. However, for experiments on the real test set, to counter the effects of the image noises on the accuracy of our method, we narrowed down the variety of the sampled training images and the yaw angles and the pitch angles of the PCB model were both sampled uniformly in the range of $(-15^\circ, 15^\circ)$. In all of the experiments, the number of decision trees and the depth of each decision tree were both set to 30. For the synthetic test set, the modulus basis m of the offset vectors was set to 2, whereas for the real test set, m was set to 7. The test program runs on a laptop with a 4-core Core i5 2.9GHz CPU, 8.0GB memory, and a Nvidia GTX 950M GPU.

Our method takes about 0.9s on average to process one PCB image, which involves the time for feature extraction and the time for the predicting process of the random forest. The average accuracy of the pixel classification for the proposed method is 98.96% on the synthetic test images and 83.64% on the real test images. The pixel classification accuracy of each component is shown in Figure 7, which is calculated according to Equation (8).

$$AC_i = \frac{TP_i}{TP_i + FP_i} \quad (8)$$

where AC_i is the pixel classification accuracy of component i , TP_i is the number of true-positive pixel classification results for component i and FP_i is the number of false-positive pixel classification results for component i .

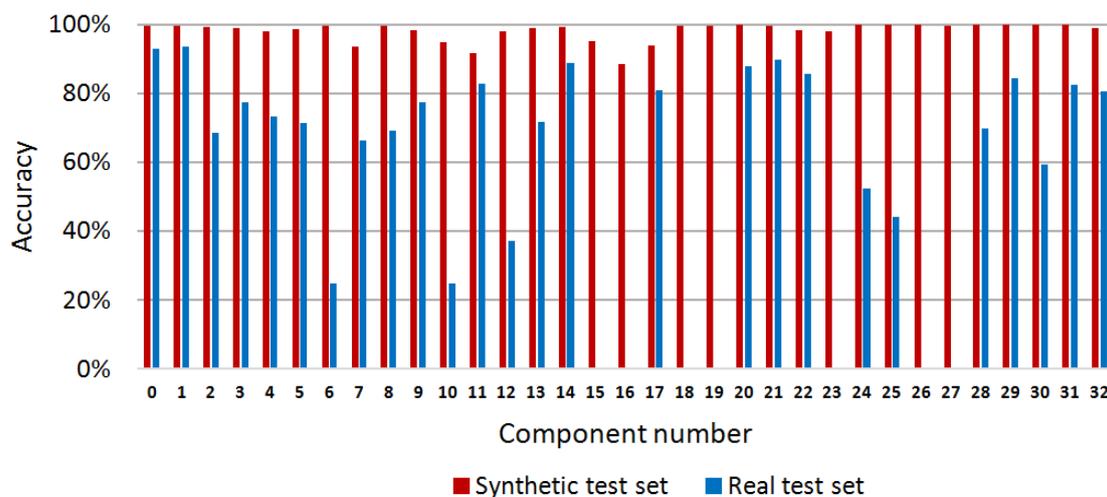


Figure 7. Pixel classification accuracy of the random forest classifier for each PCB component on synthetic and real data.

Sample results of the synthetic test set are shown in Figure 8 whereas the results of the real test images are shown in Figure 9. It can be seen from Figure 8 that for synthetic depth images all of the components on the PCB can be accurately segmented and recognized using the proposed method. For real depth images of the PCB, most of the components can also be segmented and recognized using the proposed method, as shown in Figure 9. However, Kinect 2.0 is a low-cost consumer-level electronic sensor. Due to the limited precision of the Kinect 2.0 depth sensor, several small, thinner components, such as component 6, 15, 16, 18, 19, 26, and 27, cannot be correctly detected from the real depth image. Future research will apply a high-precision industrial depth sensor to capture real depth images.

The goal of our system is to inspect the placement of the components mounted on the PCB. We need to ensure that the correct components are mounted in the right positions. Thus, the output labels of the PCB do not need to be the component instance labels. So, we labeled the same type of components (component 0 and 1, 3 and 4, 7 and 8, etc.) rather than the component instance with a same color, and conducted some tests. Figure 10 shows our results for synthetic and real images with component type labels used as the output labels. The average accuracy of pixel classification was 99.26% on the synthetic test images and 83.84% on the real test images, slightly higher than their counterparts when using component instance labels as the output labels. It can be seen that the results in Figure 10 are nearly equal in quality to the results in Figures 8 and 9 where component instance labels are used as the output labels. In this paper, we used the depth difference features that were extracted based on a set of concentric circles for training and predicting. These features make use of the information from the neighboring depth pixels to predict the given depth pixel and to decide which component instance or component type the given depth pixel belongs to. Component type

segmentation fits better with the goal of our system, whereas component instance segmentation allows us to analyze the accuracy of each specific component instance.

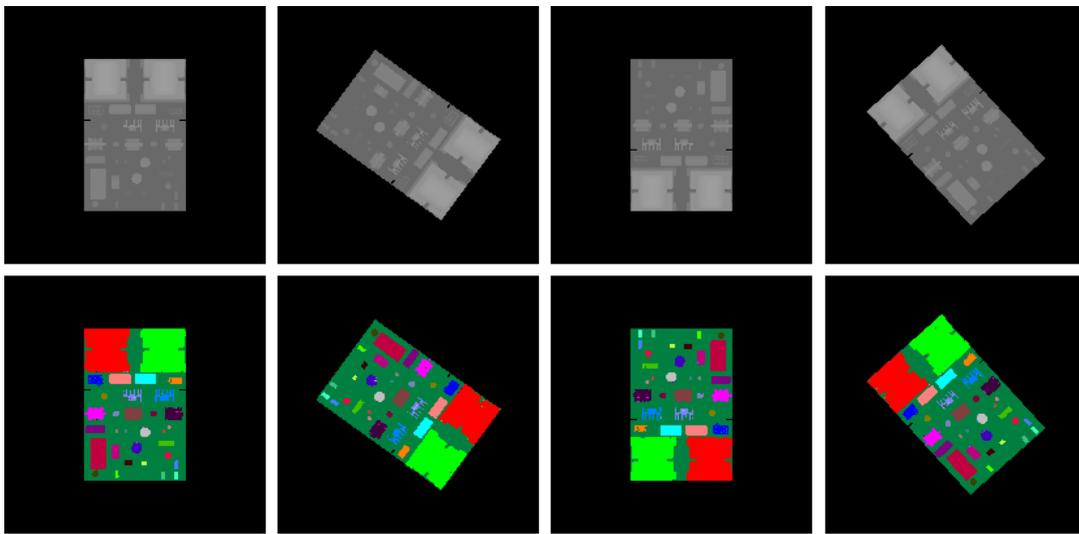


Figure 8. Synthetic depth images and the corresponding pixel classification images. The first row: synthetic depth images for test. The second row: the corresponding pixel classification images generated by the trained random forest pixel classifier.

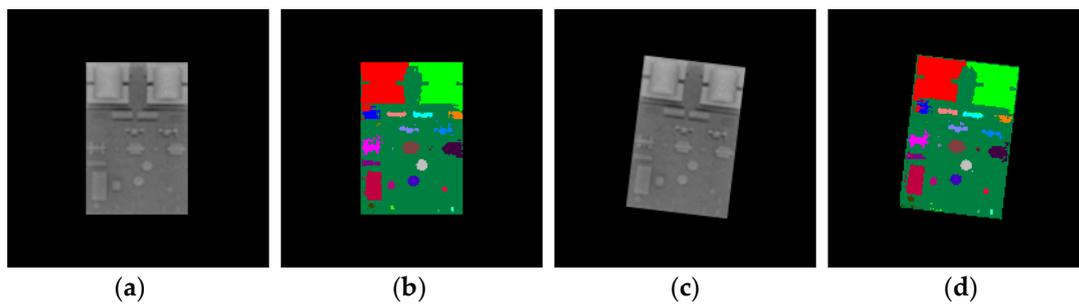


Figure 9. Real depth images and the corresponding pixel classification images. (a,c) Real depth images for test. (b,d) The corresponding pixel classification images generated by the trained random forest pixel classifier.

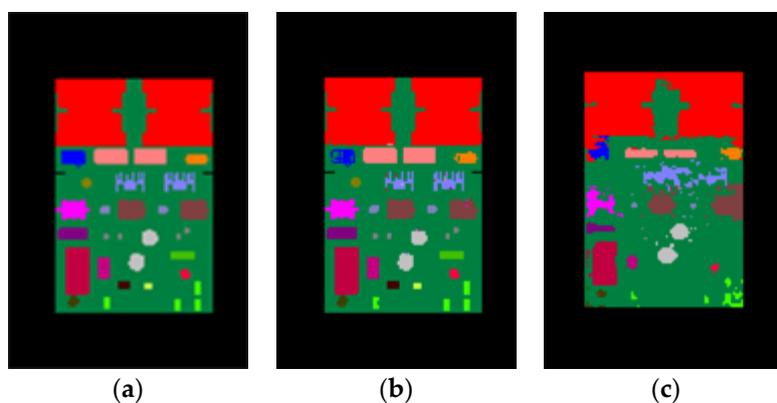


Figure 10. Component type labels used as the output labels. (a) The ground truth color-labeled image. (b) The results for the synthetic depth image. (c) The results for the real depth image.

6.3. Tests with Missing or Misplaced Components

To evaluate the performance of our method for component placement inspection, we conducted experiments on PCBs with missing or misplaced components. We used the synthetic images of the correctly mounted PCB to train the random forest pixel classifier. During the inspection, we used the trained classifier to predict the depth images of the PCBs with missing or misplaced components. Then, we compared the resulting pixel classification images of these PCBs with the color-labeled image of the correctly mounted PCB (as shown in Figure 2b) and calculated the accuracy of each component according to Equation (8). By analyzing the accuracies of the components, the missing or misplaced components could be detected. Since the real images acquired by the Kinect 2.0 depth sensor have limited depth precision, we used synthetic depth images for testing in these experiments. Figure 11 shows a PCB with component 22 misplaced, whereas Figure 12 shows a PCB with component 22 missing. Figure 13 shows the accuracy of each component on these PCBs calculated based on the color-labeled image of the correctly mounted PCB. It can be seen from Figure 13 that the accuracy of component 22 is abnormal in both situations. In this way, component 22 can be identified as the missing or misplaced component.

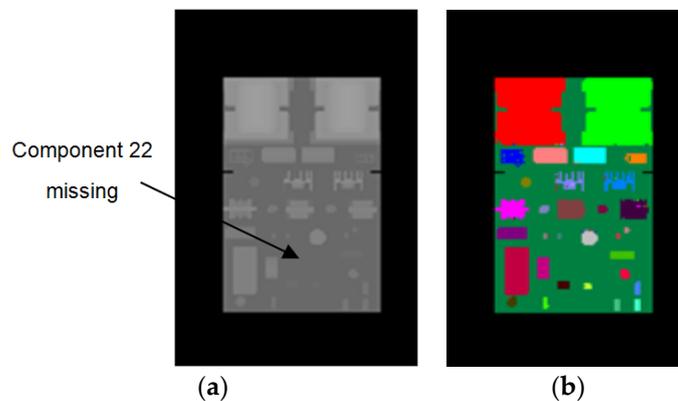


Figure 11. PCB with component 22 missing. (a) Synthetic depth image. (b) The resulting pixel classification image of (a).

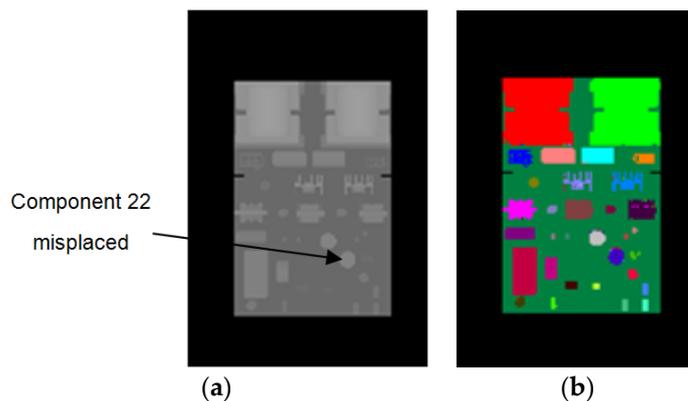


Figure 12. PCB with component 22 misplaced. (a) Synthetic depth image. (b) The resulting pixel classification image of (a).

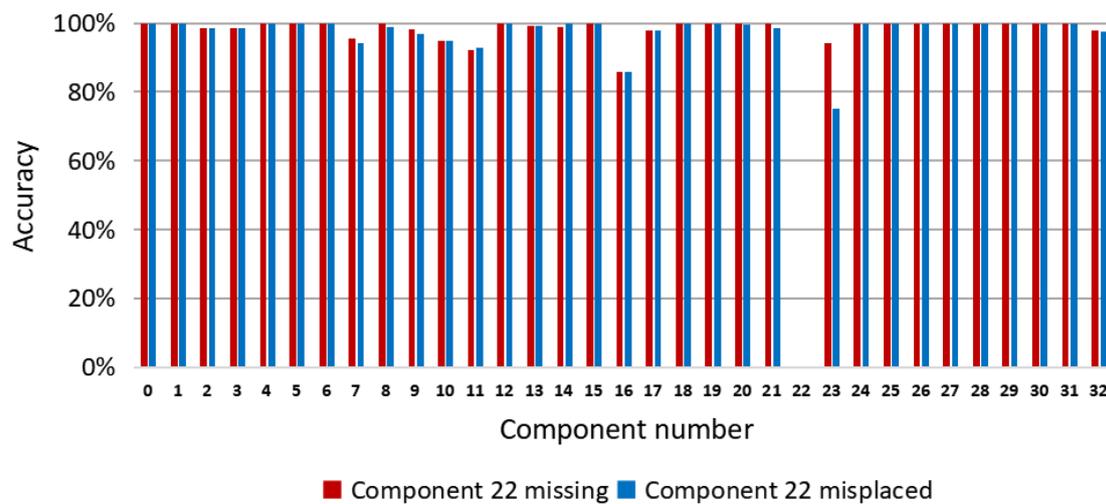


Figure 13. Pixel classification accuracy of each component on PCBs with component 22 missing and misplaced, which is calculated based on the color-labeled image of the correctly mounted PCB.

6.4. Tests with Other PCBs

To further verify the effectiveness of our method, we also conducted experiments on other PCBs: a television motherboard and a controller board. Since the real images acquired by the Kinect 2.0 depth sensor have a limited depth precision, the synthetic depth images were used for testing in these experiments. In these experiments, 200 synthetic depth images were used for training. The number of decision trees and the depth of each decision tree were both set to 30. The modulus basis m of the offset vectors was set to 2.

The average accuracies of pixel classification for our method for these two boards were both above 98%. In Figures 14 and 15, a synthetic depth image for testing and the corresponding pixel classification image are shown for these two boards, respectively. It can be seen from Figures 14 and 15 that our method was also effective for the semantic segmentation of other PCBs.

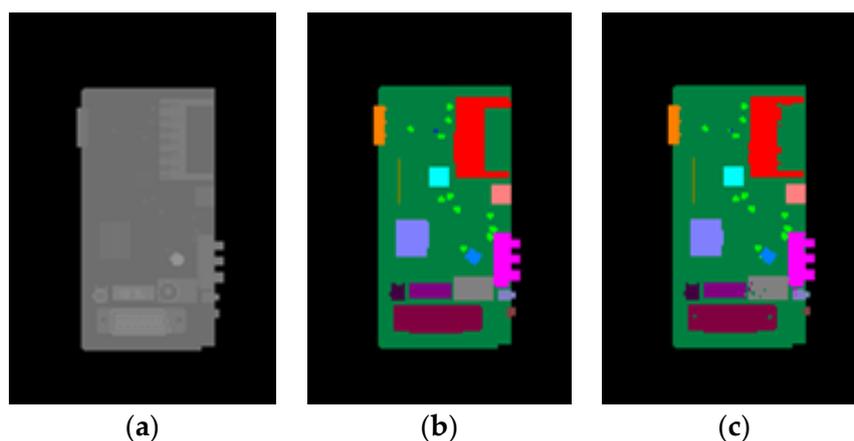


Figure 14. The results of our method for a television motherboard. (a) Synthetic depth image. (b) Color-labeled image. (c) The resulting pixel classification image of (a).

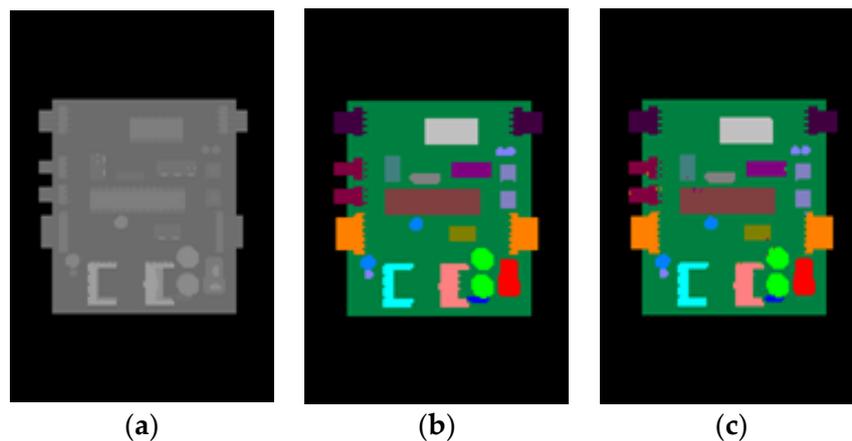


Figure 15. The results of our method for a controller board. (a) Synthetic depth image. (b) Color-labeled image. (c) The resulting pixel classification image of (a).

7. Conclusions

This paper proposes a PCB semantic segmentation method based on depth images to segment and recognize the components on the PCB. An image training set and image test set of the PCB were constructed. Depth difference features were extracted from the synthetic depth images in the training set to train a random forest classifier. The trained random forest pixel classifier was applied for semantic segmentation of the PCB. Components in the PCB were segmented and recognized through pixel classification. According to experimental results for the synthetic and real test sets, the proposed method can achieve effective segmentation and recognition of PCB components. The average pixel classification accuracy of the proposed method achieves 98% for synthetic depth images and 83% for real depth images. The proposed method can be used to inspect the placement of the components in the PCB. In addition, the proposed method can also be used in automated PCB recycling. During the recycling of the PCBs, the reusable components and precious materials need to be recovered and the toxic substances should be separated. The segmentation of the components mounted on a PCB is a first step for automated PCB recycling that is based on a comprehensive analysis of the PCBs.

The proposed method can segment and recognize most of the components from a real depth image of the PCB. However, due to the limited precision of the Kinect 2.0 depth sensor, some small, thinner components may not be correctly detected from the real depth image. Kinect 2.0 is a low-cost, consumer-level electronic sensor. In future research, we will change to a high-precision industrial depth sensor for capturing real depth images. In this paper, the pixel classifier is trained by synthetic data. The image training set is synthesized automatically by computer graphic rendering based on the 3D model of the PCB, but color labeling of the 3D model of the PCB and its components requires manual work. In future research, we plan to achieve automatic color labeling through secondary development of the 3D CAD software systems. On this basis, the image training set could be generated completely automatically in industrial production. The proposed method is not a real-time method for prediction. However, the feature extraction and random forest can both be implemented in parallel on a GPU. Future research will accelerate the system by GPU programming.

Author Contributions: Conceptualization, D.L. and C.C.; Methodology, D.L. and Z.Z.; Software, D.L. and C.L.; Investigation, D.L. and C.L.; Validation, C.L. and D.L.; Formal analysis, C.L. and D.L.; Writing—original draft, D.L.; Writing—review and editing, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by National Natural Science Foundation of China (Grant No.51705273).

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

Symbols

l_i	Classification label for component i
s_j	Pose sample j of the PCB
x	Pixel coordinates of the given depth pixel to be predicted
u	Pixel offset vectors relative to depth pixel x
$d(x)$	The corresponding depth value of depth pixel x
θ	A pair of pixel offset vectors (u_1 and u_2) for depth feature extraction
f_θ	Depth difference feature at parameter θ
τ	The threshold corresponding to f_θ at the branch node
C	A large constant
α	The angle of the offset vector
n	Concentric circle number
m	A constant basis for offset vector moduli
A	A set of N equal angles on the n th concentric circle for depth feature extraction
U	A set of offset vectors that are selected on the n th concentric circle
t	Decision tree number
S_L	The left sample subset of the current branch node
S_R	The right sample subset of the current branch node
AC_i	The pixel classification accuracy of component i
TP_i	The number of true-positive pixel classification results for component i
FP_i	The number of false-positive pixel classification results for component i

References

1. Acciani, G.; Brunetti, G.; Fornarelli, G. Application of neural networks in optical inspection and classification of solder joints in surface mount technology. *IEEE Trans. Ind. Inform.* **2006**, *2*, 200–209. [\[CrossRef\]](#)
2. Hao, W.; Xianmin, Z.; Yongcong, K.; Gaofer, O.; Hongwei, X. Solder joint inspection based on neural network combined with genetic algorithm. *Optik* **2013**, *124*, 4110–4116. [\[CrossRef\]](#)
3. Ko, K.W.; Cho, H.S. Solder joints inspection using a neural network and fuzzy rule-based classification method. *IEEE T. Electron. Pack.* **2000**, *23*, 93–103.
4. Wu, F.; Zhang, X. An inspection and classification method for chip solder joints using color grads and Boolean rules. *Robot. Cim. Int. Manuf.* **2014**, *30*, 517–526. [\[CrossRef\]](#)
5. Dai, W.; Mujeeb, A.; Erdt, M.; Sourin, A. Soldering defect detection in automatic optical inspection. *Adv. Eng. Inform.* **2020**, *43*, 101004. [\[CrossRef\]](#)
6. Wu, H.; Zhang, X.; Xie, H.; Kuang, Y.; Ouyang, G. Classification of solder joint using feature selection based on Bayes and support vector machine. *IEEE Trans. Comp. Pack. Man.* **2013**, *3*, 516–522.
7. Hongwei, X.; Xianmin, Z.; Yongcong, K.; Gaofer, O. Solder joint inspection method for chip component using improved AdaBoost and decision tree. *IEEE Trans. Comp. Pack. Man.* **2011**, *1*, 2018–2027. [\[CrossRef\]](#)
8. Cai, N.; Zhou, Y.; Ye, Q.; Liu, G.; Wang, H.; Chen, X. IC solder joint inspection via robust principle component analysis. *IEEE Trans. Comp. Pack. Man.* **2017**, *7*, 300–309. [\[CrossRef\]](#)
9. Liu, J.; Shi, T.; Wang, K.; Tang, Z.; Liao, G. Defect detection of flip-chip solder joints using modal analysis. *Microelectron. Reliab.* **2012**, *52*, 3002–3010. [\[CrossRef\]](#)
10. Cai, N.; Lin, J.; Ye, Q.; Wang, H.; Weng, S.; Ling, B.W.K. A new IC solder joint inspection method for an automatic optical inspection system based on an improved visual background extraction algorithm. *IEEE Trans. Comp. Pack. Man.* **2015**, *6*, 161–172.
11. Loh, H.H.; Lu, M.S. Printed circuit board inspection using image analysis. *IEEE Trans. Ind. Appl.* **1999**, *35*, 426–432.
12. Crispin, A.J.; Rankov, V. Automated inspection of PCB components using a genetic algorithm template-matching approach. *Int. J. Adv. Manuf. Tech.* **2007**, *35*, 293–300. [\[CrossRef\]](#)
13. Zeng, Z.; Ma, L.; Suwa, M. Algorithm of locating PCB components based on colour distribution of solder joints. *Int. J. Adv. Manuf. Tech.* **2011**, *53*, 601–614. [\[CrossRef\]](#)

14. Herchenbach, D.; Li, W.; Breier, M. Segmentation and classification of THCs on PCBAs. In Proceedings of the 2013 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, 29–31 July 2013; pp. 59–64.
15. Li, W.; Esders, B.; Breier, M. SMD segmentation for automated PCB recycling. In Proceedings of the 2013 11th IEEE International Conference on Industrial Informatics (INDIN), Bochum, Germany, 29–31 July 2013; pp. 65–70.
16. Lim, D.U.; Kim, Y.G.; Park, T.H. SMD classification for automated optical inspection machine using convolution neural network. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 395–398.
17. De Oliveira, T.J.M.; Wehrmeister, M.A.; Nassu, B.T. Detecting modifications in printed circuit boards from fuel pump controllers. In Proceedings of the 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Niterói, Brazil, 17–18 October 2017; pp. 87–94.
18. Lin, S.C.; Chou, C.H.; Su, C.H. A development of visual inspection system for surface mounted devices on printed circuit board. In Proceedings of the IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society, Taipei, Taiwan, 5–8 November 2007; pp. 2440–2445.
19. Mar, N.S.S.; Yarlagadda, P.K.D.V.; Fookes, C. Design and development of automatic visual inspection system for PCB manufacturing. *Robot. Cim. Int. Manuf.* **2011**, *27*, 949–962. [[CrossRef](#)]
20. Guerra, E.; Villalobos, J.R. A three-dimensional automated visual inspection system for SMT assembly. *Comput. Ind. Eng.* **2001**, *40*, 175–190. [[CrossRef](#)]
21. Leta, F.R.; Feliciano, F.F. Computational system to detect defects in mounted and bare PCB based on connectivity and image correlation. In Proceedings of the 2008 15th International Conference on Systems, Signals and Image Processing, Bratislava, Slovakia, 25–28 June 2008; pp. 331–334.
22. Zervakis, M.E.; Goumas, S.K.; Rovithakis, G.A. A Bayesian framework for multilead SMD post-placement quality inspection. *IEEE Trans. Syst. Man Cybern. B* **2004**, *34*, 440–453. [[CrossRef](#)] [[PubMed](#)]
23. Mashohor, S.; Evans, J.R.; Erdogan, A.T. Automatic hybrid genetic algorithm based printed circuit board inspection. In Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06), Istanbul, Turkey, 15–18 June 2006; pp. 390–400.
24. Rehman, S.U.; Thang, K.F.; Lai, N.S. Automated PCB identification and defect-detection system (APIDS). *Int. J. Electr. Comput. Eng.* **2019**, *9*, 2088–8708. [[CrossRef](#)]
25. Li, R.; Liu, Z.; Tan, J. Exploring 3D human action recognition: From offline to online. *Sensors* **2018**, *18*, 633.
26. Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; Blake, A. Real-time human pose recognition in parts from single depth images. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, CO, USA, 20–25 June 2011; pp. 1297–1304.
27. Shotton, J.; Girshick, R.; Fitzgibbon, A.; Sharp, T.; Cook, M.; Finocchio, M.; Moore, R.; Kohli, P.; Criminisi, A.; Kipman, A.; et al. Efficient human pose estimation from single depth images. *IEEE Trans. Pattern. Anal.* **2013**, *35*, 2821–2840. [[CrossRef](#)]
28. Girshick, R.; Shotton, J.; Kohli, P.; Criminisi, A.; Fitzgibbon, A. Efficient regression of general-activity human poses from depth images. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 415–422.
29. Keskin, C.; Kırac, F.; Kara, Y.E.; Akarun, L. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision, Advances in Computer Vision and Pattern Recognition*; Fossati, A., Gall, J., Grabner, H., Ren, X., Konolige, K., Eds.; Springer: London, UK, 2013; pp. 119–137.
30. Keskin, C.; Kırac, F.; Kara, Y.E.; Akarun, L. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In Proceedings of the 2012 European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; pp. 852–863.
31. Sharp, T.; Keskin, C.; Robertson, D.; Taylor, J.; Shotton, J.; Kim, D.; Rhemann, C.; Leichter, I.; Vinnikov, A.; Wei, Y.; et al. Accurate, robust, and flexible real-time hand tracking. In Proceedings of the 33rd Annual Acm Conference on Human Factors in Computing Systems (CHI'15), Seoul, Korea, 18–23 April 2015; pp. 3633–3642.
32. Tompson, J.; Stein, M.; Lecun, Y.; Perlin, K. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graphic.* **2014**, *33*, 169. [[CrossRef](#)]

33. Pramerdorfer, C.; Kampel, M. PCB recognition using local features for recycling purposes. In Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP), Berlin, Germany, 11–14 March 2015; pp. 71–78.
34. Breiman, L. *Classification and Regression Trees*, 1st ed.; Routledge: New York, NY, USA, 1984.
35. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).