IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 April 2024, accepted 20 April 2024, date of publication 29 April 2024, date of current version 6 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3394524

RESEARCH ARTICLE

Synthetic Data Generation for Text Spotting on Printed Circuit Board Component Images

WEI JIE BRIGITTE LIAU[1], SHIEK CHI TAY[1], AHMAD SUFRIL AZLAN MOHAMED[1], MOHD NADHIR AB WAHAB[1], (Member, IEEE), LAY CHUAN LIM[2], BENG KANG KHAW[2], AND MOHD HALIM MOHD NOOR[1]

[1]School of Computer Sciences, Universiti Sains Malaysia (USM), Pulau Pinang, Gelugor 11800, Malaysia
[2]SSD Platform Technology Development and Integration, Western Digital, Simpang Ampat, Penang 14110, Malaysia

Corresponding author: Mohd Halim Mohd Noor (halimnoor@usm.my)

This work was supported in part by the Collaborative Research in Engineering, Science and Technology (CREST) Industry Graduate Research Assistant Scholarship Program (CREST i-GRASP), Universiti Sains Malaysia (USM); and in part by Western Digital (Sandisk Storage Malaysia Sdn. Bhd.) under CREST Research and Development Project "Image Data Analytics for Industry 4.0" under Grant P08C1-20.

ABSTRACT Machine vision systems with programmed text detection and text recognition features are useful in manufacturing process to automatically locate and read text markings on mounted printed circuit board (PCB) components. To better handle input images with varying image quality, text quality, and text variations, the robustness of deep learning approach for end-to-end text spotting on PCB component images is worth exploring. However, limitations of public PCB component datasets for such research and imbalance of data in actual collected PCB component datasets hinder the training of deep learning text spotting model, and consequently necessitate the generation of synthetic data. In this study, a synthetic PCB component dataset is generated using our synthetic data generator that adds synthetic text with random character sequences on manually edited PCB component images to elevate the realism of the synthetic images. The synthetic dataset covers 66 character classes while providing synthetic text with diverse text variations in font, style, size, and color. We train an existing text spotting model called Text Perceptron using both real and synthetic datasets to detect and recognize arbitrary-shaped text markings on PCB components. Our synthetic PCB component dataset has improved the text spotting performance of Text Perceptron. The trained model achieves promising text detection result and encouraging end-to-end text spotting F-score on real PCB component images. It also meets an acceptable average inference time per image. Still, the text spotting performance of the trained model needs improvement to realize deployment for PCB component inspection.

INDEX TERMS Deep learning, printed circuit board component, synthetic data, text spotting.

I. INTRODUCTION

Throughout the assembly of electronic components on printed circuit boards (PCBs), various inspections and tests have to be carried out to ensure that the final assembled PCBs are of high quality and free from PCB failure. Common visual inspection of PCBs and the components requires machine vision systems like automated optical inspection (AOI) machines which screen for failures or defects such as missing components, shifted components, poor soldering, and so on. Only good PCBs flow to subsequent stages, while rejected ones require a rework. Some AOI machines are programmed with text detection and text recognition features to automatically locate and read text markings on the surface of PCB components as shown in Fig. 1. Indeed, these features support not only inspection of PCBs but also recording of useful information (e.g., region of text marking and part number) into the system.

Ideally, PCB component images as text detection and text recognition input are of good image quality and text quality for correct detection and recognition outputs, which can be stored as data or compared with references retrieved from the system for inspection. Nonetheless, their image quality is often affected by factors like noise, sharpness, or reflective

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

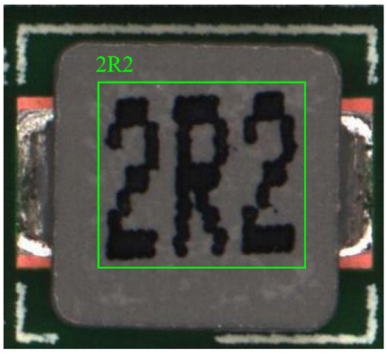

FIGURE 1. Performing text spotting on a PCB component image returns the detected region of interest that contains a text marking (Green box) and the recognized character sequence "2R2" (Green text).

surface of PCB components as shown in Fig. 2; plus, the text markings on PCB components that have no strict standards on printing method, layout, and font settings could be defective as shown in Fig. 3. For example, blurry markings have no clear edges whereas faded markings have low contrast with the background. Poor image quality and text quality in addition to text variations have made it more challenging for AOI machines to correctly detect and recognize printed text markings on PCB components. As a result, false failures made by AOI machines may increase, at the same time, it forces manual checking on the input image and the output character sequence to reduce the number of misjudgments. The worst case is when the text markings are unrecognizable even by human operators. For open-source OCR engines that are not designed for PCB component images, [1] shows that they may struggle to read small text markings on components. Hence, a more reliable text spotting solution is expected to support the management of PCB components in manufacturing.

Together with the advent of artificial intelligence, text spotting research remains active with more challenging text images to achieve accuracy like optical character recognition (OCR) for documents. It will be advantageous if such text spotting solutions can be applied to PCB component images; however, many research [2], [3], [4], [5], [6], [7], [8], [9] use scene text images [10], [11], [12], [13], [14], [15] but not PCB component images [16] for end-to-end text spotting task. One reason is that the former can be easily obtained from public datasets, whereas the latter requires permissions from factories regarding image data collection for confidentiality purposes. Nevertheless, PCB component images that come from the same site usually contain highly duplicated text markings and the number of samples in each character class may be imbalanced [17], [18], [19], [20]. Additionally, accessible PCB datasets that provide information for text markings on PCB components [21], [22], [23], [24] usually need data preprocessing such as removing human mistakes from the ground truth text annotations. These PCB datasets differ from scene text datasets in that the latter can be used directly for model training and as a benchmark for evaluation. Although the dataset in [16] can be used for text spotting research, the majority of its samples are defective characters that are targeted for character-level aesthetic assessment task. In other words, it is laborious to gather PCB component images that meet both diversity and quantity requirements to train a deep learning model for text spotting on PCB component images.

Instead of relying entirely on real data, synthetic data can be created to support the development of text spotting methods for PCB component images. There are some limitations of existing methods in generating synthetic PCB component data, for example, generating synthetic images by solely applying data augmentation methods on actual PCB component images is ineffective for character classes that have not collected any sample [18]. A large dataset could be obtained by creating fully-synthetic text images [17], [18], [19]; however, these synthetic images do not consider the actual backgrounds of PCB components during data generation. Although the work in [20] generates partially-synthetic images by replacing actual PCB component characters with synthetic characters, the synthetic images are less "real" due to the edited regions that can be easily identified by human eyes. Furthermore, excluding lowercase alphabets and symbols during synthetic text generation is less sensible. The character classes of synthetic text typically include digits and uppercase alphabets, with no symbols [17], [18], [19], [20], while lowercase alphabets are either omitted [19] or selectively disregarded [17], [18].

Hence, we present a synthetic data generation method to facilitate the training of text spotter for PCB component images. The contributions of this paper are as follows compared to the existing literature:

- We propose a practical synthetic PCB component data generation method for supporting the training of a deep learning text spotting model. It is a viable approach to generate a sufficient amount of synthetic images from a small set of real PCB component images.
- Our proposed method generates verisimilar synthetic text data that cover 66 character classes for 10 digits, 52 Latin alphabets, and 4 symbols (period, underscore, slash, and dash) to ensure the deep learning model generalizes well on unseen data.
- The synthetic data generator is capable of generating synthetic text data with diverse characteristics based on 42 fonts in 4 styles with different sizes and colors. This flexibility ensures that the synthetic dataset encompasses a wide range of text variations.
- To elevate the realism of the synthetic data, the proposed method leverages real PCB component images to create a more realistic and representative dataset.
- To evaluate the performance of the proposed method, we conduct the experiments using real-world PCB images collected from an actual production line and an existing Text Perceptron deep learning text spotting model [2]. The synthetic PCB component dataset has improved the end-to-end text spotting F-score of Text Perceptron. Generally, the trained Text Perceptron

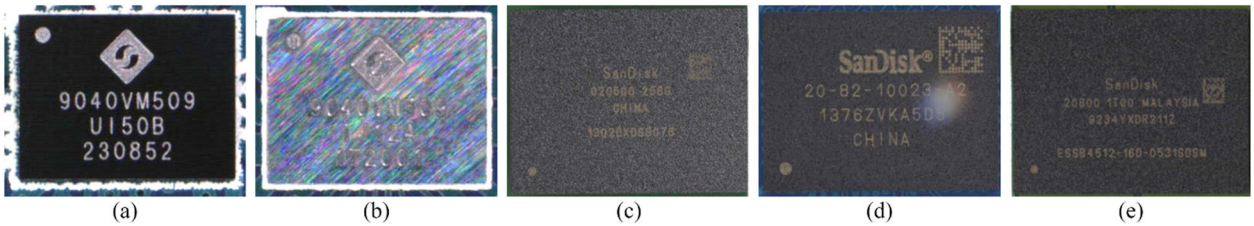

FIGURE 2. Actual PCB component images. (a) A well-focused PCB component image. (b) Reflection or uneven illumination on component surface. (c) Image with noise. (d) Lens flare or spot. (e) Low image sharpness. Images are cropped and brightness is adjusted for better visualization.

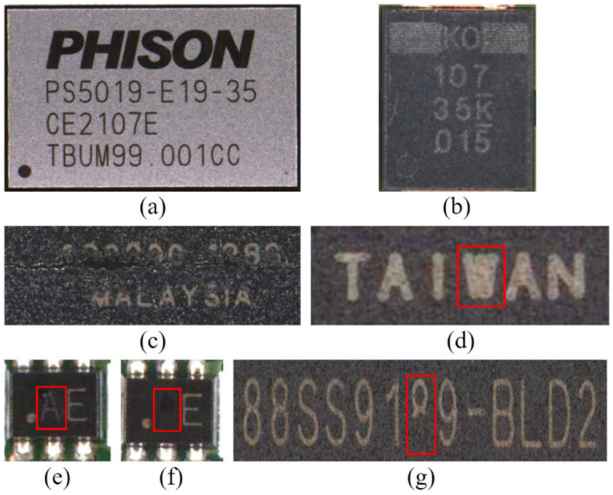

FIGURE 3. Actual PCB component text markings. (a) Clear marking. (b) Blurry marking. (c) Damaged marking. (d) Unrecognizable uppercase alphabet "W" (Red box) that is judged based on its neighboring characters. (e) Faded marking (Red box). (f) Missing character (Red box). (g) Broken character (Red box). Another type of text marking defect that is not included in this figure is the wrong character printed on a PCB component. Images are cropped and brightness is adjusted for better visualization.

shows promising text detection performance on PCB component images. Its text spotting results on real test data are encouraging and have room for improvement for synthetic images while meeting acceptable average inference time per image. The results also show that the text spotting model has the potential for inspection on PCB component images.

The remainder of this paper is structured as follows: Section II reviews previous methods related to the work in this paper; Section III describes the generation of synthetic PCB component data and the overview of Text Perceptron for text spotting on PCB component images; Section IV reports the experiments on text spotting followed by discussion on experiment results and comparison with state-of-the-art methods; and Section V concludes the paper and suggests potential improvements for future research.

II. RELATED WORK

This section briefly describes existing methods that generate synthetic PCB component data for deep learning model training. Then, it includes related work on text detection and recognition for PCB component images together with some existing scene text spotting models. A text detection task is to locate all text regions in a text image; a text recognition task is to transcribe a text image into character sequence; and a text spotting task accepts a text image and outputs the position of each detected text instance along with its corresponding character sequence via an end-to-end text detection and recognition model.

A. METHODS OF GENERATING SYNTHETIC DATA

PCB inspection tasks such as defect detection [25] or component classification [26] usually have more public PCB datasets that can be used for training or as a standard to compare different methods. Among the available PCB datasets, a few of them provide information for text markings on PCB components [16], [21], [22], [23], [24], as described in Table. 1. A common limitation of [21], [22], [23], and [24] is the need for data cleaning before feeding the text data for model training, particularly due to unstandardized annotation format or human mistakes in the ground truth text annotations, whereas the majority of samples in [16] are defective character samples that are targeted for character-level aesthetic assessment task. Alternatively, PCB component images can be collected from actual production lines; however, data that come from the same site usually have imbalanced distributions within a character class and between character classes [17], [18], [19], [20]. The former happens when a character class highly contains samples with similar structure, whereas the latter means that some characters appear very frequently while other character classes may have little or no samples at all. To introduce diversity in the dataset, the collected PCB component character images in [18] are augmented in four ways: 1) color reversal; 2) random rotation between $+5$ and -5 degrees; 3) image resizing (32×32 pixels); and 4) random Gaussian noise. Still, data augmentation methods are rendered useless for character classes that are underrepresented in the training set, and this can hinder the deep learning model's ability to learn and generalize well on the minority classes.

A more practical approach involves the generation of synthetic images. Synthetic character images with random colors for text and background are generated in [17] and [18] based on empirically-defined color tones that the authors claim to be frequently used on actual PCBs. First, a generated image with black character and white background is converted from RGB (red, green, blue) to HSV (hue,

TABLE 1. Description of accessible PCB datasets with text information.

Dataset	Description
PCB DSLR [21]	• It contains 748 images and 9,313 annotated ICs by rotating each of the 165 PCBs for 3 to 5 different orientations. It has 1,740 IC text labels. • Text markings on ICs are annotated at block level, in which each text block has a rectangular bounding box with orientation of text, and words or lines of text are delimited by space character. However, not all text markings on the same IC are annotated, even with similar legibility, while other PCB components with text markings are ignored. Also, the ground truth annotations are case-insensitive (e.g., "intel" is annotated as "INTEL"). • URL: https://zenodo.org/records/3886553
[22]	• It contains 48 PCB images with over 8,000 annotated instances and over 10,000 text annotations [24]. • Text instances on PCB and components are annotated at block level, in which each text block has a rectangular bounding box, and words or lines of text are delimited by space character. However, there is no information on the orientation of text, although text instances with different orientations are observed in a text block. Also, the ground truth text annotations contain unstandardized text annotation formats (e.g., elements in the XML files). • URL: https://sites.google.com/view/chiawen-kuo/home/pcb-component-detection
FICS-PCB [23]	• It contains 9,912 images and 77,347 annotated components from 31 PCBs, and challenging cases in terms of imaging modality, image scale, and illumination are considered to facilitate the development of PCB automated vision inspection methods. • Text markings on PCB components are annotated at block level, in which words are delimited by space character, and lines of text are delimited by semicolon. However, there is no information on the text bounding box and orientation of text. Also, the ground truth text annotations contain human errors (e.g., missing delimiter) and unstandardized text annotations (e.g., some circled text are annotated, but some are ignored). • URL: https://trust-hub.org/#/data/fics-pcb
ICText [16]	• It is a controlled-access dataset that contains 10,000 component training images and 3,000 validation images collected under different lighting conditions. The majority of annotated characters are defective as the dataset is used for both text spotting and aesthetic assessment tasks for each character on components. • Text instances on microchip components are annotated at character level with tight bounding boxes, together with the character's orientation, legibility, and aesthetic classes (i.e., low contrast, blurry, and broken). • URL: https://github.com/chunchet-ng/ICText-AGCL
FPIC [24]	• It is a controlled-access dataset that contains 271 images and over 71,000 annotated instances from 97 PCBs. It has 23,218 text instances for a subset of selected images. • Text instances on PCBs, components, stickers, and text outside the PCBs are provided with rectangular bounding boxes and orientation of text. However, a mixture of line-level and word-level text annotations is observed, in which words are delimited by space character. The ground truth text annotations also contain human errors (e.g., missing text annotation) and unstandardized text annotations (e.g., some aesthetic text logos are ignored, but some are annotated). • URL: https://trust-hub.org/#/data/pcb-images

saturation, value) color model, and then, each pixel is set with random values of H (hue), S (saturation), and V (value) within their predefined ranges for text and background pixels respectively. Before saving the character image, it is converted back to RGB and the four sides are cropped to be close to the character. Additionally, some of the synthetic images in [18] are applied with random noise. Due to character-level classification tasks in [17] and [18], uppercase and lowercase alphabets with similar shapes but different in size, e.g., 'V' and 'v' have to be combined into one class, in which the lowercase ones are ignored during the generation of synthetic images. As a result, only 10 digits, 26 uppercase alphabets, and 16 lowercase alphabets are included in the character classes.

Unlike synthetic RGB character images in [17] and [18], 8-bit grayscale synthetic text string images are generated in [19] to train a word-level text recognizer for PCB components. Each synthetic image with height and width of 32 and 128 pixels respectively, contains a synthetic text with two to eight digits or uppercase alphabets using one of the seven selected fonts that the author states to be commonly found on PCB components. The generated background pixels have random values between 50 and 150, whereas the text pixels are brighter than the background by a value of 40. Random data augmentation methods such as speckle noise, rotation, horizontal/vertical scaling, and image shearing are applied to the synthetic images. To ensure the synthetic images follow fixed height and width, the remaining regions, if any, are filled with black pixels. The steps to create fully-synthetic text images are straightforward; nevertheless, these images do not consider the actual PCB component background during the generation of synthetic images.

In the Robust Reading Challenge on Integrated Circuit Text Spotting and Aesthetic Assessment (RRC-ICText) 2021 competition, actual PCB component images with varying component structures, text orientations, image quality, and text quality are gathered as a private dataset called ICText, whereby the training images and annotations are released to the participants for model training [16]. The leading team that proposed the YOLOv5s-based text spotting model [20] has generated partially-synthetic images to balance the training samples for lowercase alphabets in ICText which are much less than that for uppercase alphabets. The original printed characters of the sampled ICText images are replaced with three types of synthetic characters: 1) white color lowercase alphabets in Windows fonts; 2) colorful lowercase alphabets in Windows fonts; and 3) lowercase alphabets generated using text style transfer method in [27]. Nonetheless, after removing the original printed characters and overlaying the synthetic characters, the resulting partially-synthetic PCB component images in [20] appear less realistic due to noticeable edited regions.

Based on the previous works, it can be concluded that data augmentation methods alone are not able to resolve the problem of an imbalanced dataset, especially if there are character classes with very minimal or absolutely zero samples. While fully-synthetic text images are quantitatively beneficial to model training, the generation of synthetic data does not incorporate actual background information on PCB components. Yet, partially-synthetic PCB component images with clearly visible edited regions will reduce their resemblance to real PCB component images. It is also impractical to exclude lowercase alphabets and symbols from synthetic text generation merely because of their relatively low occurrence count in actual PCB component markings.

Thus, in this study, a synthetic PCB component data generation method is proposed. It can generate synthetic text data that cover a total of 66 character classes, including 10 digits, 52 Latin alphabets, and 4 symbols (period, underscore, slash, and dash) for the deep learning model to generalize well on unseen data. It is also flexible in generating synthetic text data with diverse text variations in terms of font, style, size, and color. Moreover, the proposed method creates a more realistic synthetic dataset by leveraging real PCB component images, which are collected from an actual production line and grouped by part number. We generate the synthetic PCB component dataset through: 1) random sampling on real images from each image group; 2) manual removal of original PCB component markings on sampled images; 3) manual labeling of valid synthetic text region for edited images; and 4) generation of synthetic text with random character sequence, synthetic images, and ground truth annotations using our synthetic data generator. Since real PCB component text markings do not follow a specific lexicon, each synthetic image may have one or more synthetic texts with random character sequences.

B. TEXT SPOTTING MODELS

Unlike scene text spotting, the detection and recognition of PCB component text markings may involve various customized intermediate steps. In [1], a combination of image filters and binarization methods is used to improve the character recognition accuracy of Tesseract OCR [28] for PCB components with light-colored text and dark-colored backgrounds. However, finding a universal set of image preprocessing steps for actual PCB components and markings that have various colors is a nontrivial problem. A conventional way to locate the positions of component markings is by calculating the project profile of the component image which is converted to binary or grayscale [29]. In [30], the region that only contains the component markings is specified in the inspection system. Then, the row-sum of text (e.g., white) pixels are counted to determine the horizontal lines that separate a marking from the background or between rows of markings, in which the obtained line-level markings are considered as the detected text region. If there is more noise in the input component images, these methods are likely to fail in locating the correct regions of markings. In addition,

FIGURE 4. Text markings (Green box) and non-text markings (Yellow box) that are printed on a PCB component.

these image processing techniques are unable to distinguish between text and non-text markings. A proper text detection result should only include text while excluding other non-text markings as shown in Fig. 4.

For character-level text recognition, image processing techniques such as projection profile [29], counting the column-sum of text (e.g., white) pixels [30], and text contour border detection adopted from Suzuki's border-following algorithm [31] are used to segment PCB component text markings into individual characters before the character recognition stage. Yet, these methods are ineffective in segmenting a character that is not well-connected or multiple characters that are close to one another. Customized algorithms using morphological operations are needed to combine a fractured character or to segment joined characters [31]. Also, in [17] and [18], programmed inspection software may segment printed characters on PCB components but manual corrections are often needed, especially for fonts that have short distances in between characters. After getting the segmented character images, a straightforward character recognition method is by matching each of them with the stored patterns and checking for the degree of mismatch according to a selected threshold [29]. Another way is to apply feature extraction methods like projection profile, zoning, moments, and contour profile on the segmented characters to classify them into different character classes [30]. Since neural network character classifiers are more robust to variations of text such as font, style, and size, various neural networks are used to recognize segmented characters on PCB components, including the modified LeNet [31], ResNet and EfficientNet family [17], [18], and ShuffleNetV2 [32]. In [19], character segmentation is discarded, as opposed to the previous works mentioned above which recognize markings at character level. The author has adopted an LPRNet to perform word-level text recognition, in which RNN is excluded from the network to speed up the inference [19].

Instead of using separated text detection and text recognition methods, a related work that involves text spotting on PCB component images is the RRC-ICText 2021 competition [16]. Since it is proposed to seek better solutions for quality inspection on each printed character, the text spotting task is being treated as an object detection task that uses COCO evaluation protocol for 62 character classes (10 digits and 52 Latin alphabets). Among the participating models for the text spotting task, a YOLOv5s-based model [20] is reported as the leading solution. Nevertheless, the text spotting mAP score for every participating team still has a

huge room for improvement. For example, multi-oriented characters and defective characters are the common challenges for all the participating text spotting models [16].

It is valuable to study existing text spotting models to perform text spotting on PCB component images, based on their reported performance on scene text benchmark datasets [10], [11], [12], [13], [14], [15], [33], [34], [35], [36]. Earlier text spotters with cascaded, separately-trained text detection model and text recognition model are sometimes known as two-step text spotters [37], [38], [39], [40]. To avoid the accumulation of errors between these cascaded models and, at the same time, to realize the regularization of text detection using text recognition output, and vice versa, end-to-end trainable text spotting models are proposed. There are mainly two groups of end-to-end trainable deep learning text spotting models: 1) two-staged text spotter, and 2) one-staged text spotter. Generally, two-staged text spotters sequentially detect and recognize text in an image [2], [3], [4], [5], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54]. Various tailored modules are proposed to connect the text detection and text recognition modules for end-to-end model training, such as ROI Align [3], [43], [47], ROIRotate [46], ROISlide [48], and BezierAlign [4], [51]. On the other hand, one-stage text spotters simultaneously predict text region and character sequence in an image [6], [7], [8], [9], [55], [56], [57]. These kinds of models remove customized ROI operations that usually require experts with prior knowledge for the design of modules; however, they may produce inaccurate text detection result that leads to the failure of subsequent text recognition [55]. Furthermore, existing deep learning two-staged and one-staged text spotting models may use different representations to indicate a text region, including axis-aligned horizontal rectangle [42], [49], [55], oriented rectangle or quadrangle [6], [41], [44], [45], [46], [48], [55], polygon [2], [3], [6], [7], [9], [43], [47], [49], [52], point form [5], [50], and [57], parameterized Bezier curve [4], [51], [54], kernel form [53], as well as grid row and column with rough text locations [8]. While most of these text spotters are capable of handling word-level text [2], [7], [8], [9], [44], [45], [46], [47], [50], [52], [55], [56] or line-level text [4], [5], [48], [49], [51], [53], [54], [57], there are a few text spotting models that operate at character level [6], [43]. To improve the prediction of character sequence, CTC [41], [46], [48], [51] and attention mechanisms [2], [3], [4], [8], [44], [47], [49], [50], [52], [53], [54] including 1D-attention [42] and designed 2D-attention [55] are popular approaches in designing a deep learning text spotting model.

In this paper, Text Perceptron [2] is chosen as the text spotting model for PCB component images. An end-to-end trainable deep learning text spotter is beneficial to our model training on PCB component datasets compared to a two-step deep learning text spotter with cascaded text detector and text recognizer that need separate training. This is because mutual optimization of text detector and text recognizer is possible for the former. One-staged text spotters that discard ROI operations may lead to failure in text spotting due to inaccurate text detection results; hence, a two-staged text spotter is preferred to a one-staged text spotter. Since PCB component text markings do not follow any lexicon or vocabulary, a text spotter that involves language model in model training is ill-suited for our text spotting task. To reduce labeling cost, text spotters that require character-level training data are not considered. Text spotters that only detect and recognize text lines are excluded as well. The reason is that the predicted PCB component text markings at line level may group text of interest and irrelevant text as one text instance, which requires postprocessing to correctly extract the desired output. Therefore, text spotters that operate on word-level text instances are more manageable for this study. In terms of bounding box representation for text instances, we follow polygonal ones as in [2] that can handle text markings with arbitrary shapes.

III. METHODOLOGY

This section describes the real PCB component dataset and explains the rationale behind the generation of synthetic PCB component dataset, followed by the steps to generate the synthetic dataset using our synthetic data generator. Then, we briefly describe Text Perceptron [2] to be trained and experimented on PCB component images.

A. GENERATION OF SYNTHETIC DATA

Before we generate the synthetic dataset, we prepare the real PCB component dataset by collecting PCB component images from an actual production line, followed by the grouping of images based on their part number and labeling of images for all image groups as shown in Fig. 5. The ground truth annotations record the text marking regions and text transcriptions for all the text markings on PCB components at word level. At this stage, we use a horizontal rectangular bounding box to represent the region of a text marking on PCB component. Defective text markings that are unrecognizable by human eyes are labeled with the '$' symbol to be ignored from model training and evaluation of Text Perceptron [2]. Since this study focuses on text information on PCB components, we annotate text logos as they carry information about the components' manufacturers, whereas aesthetic text logos are excluded as they usually differ from the normal structure of characters. Also, logo recognition is excluded from the scope of this study. This is to avoid making assumptions about non-text logos with unknown manufacturers during labeling.

After getting all the annotations, we convert the rectangular bounding box annotations to polygonal form that is used in [2]. Based on the width, height, and top-left coordinates of a rectangular bounding box, its corresponding polygonal bounding box with four pairs of (x, y) coordinates which represent the top-left, top-right, bottom-right, and bottom-left corners respectively, can be easily computed as shown in Fig. 5. This method is practicable in this paper because the text markings in our real dataset are mainly horizontal

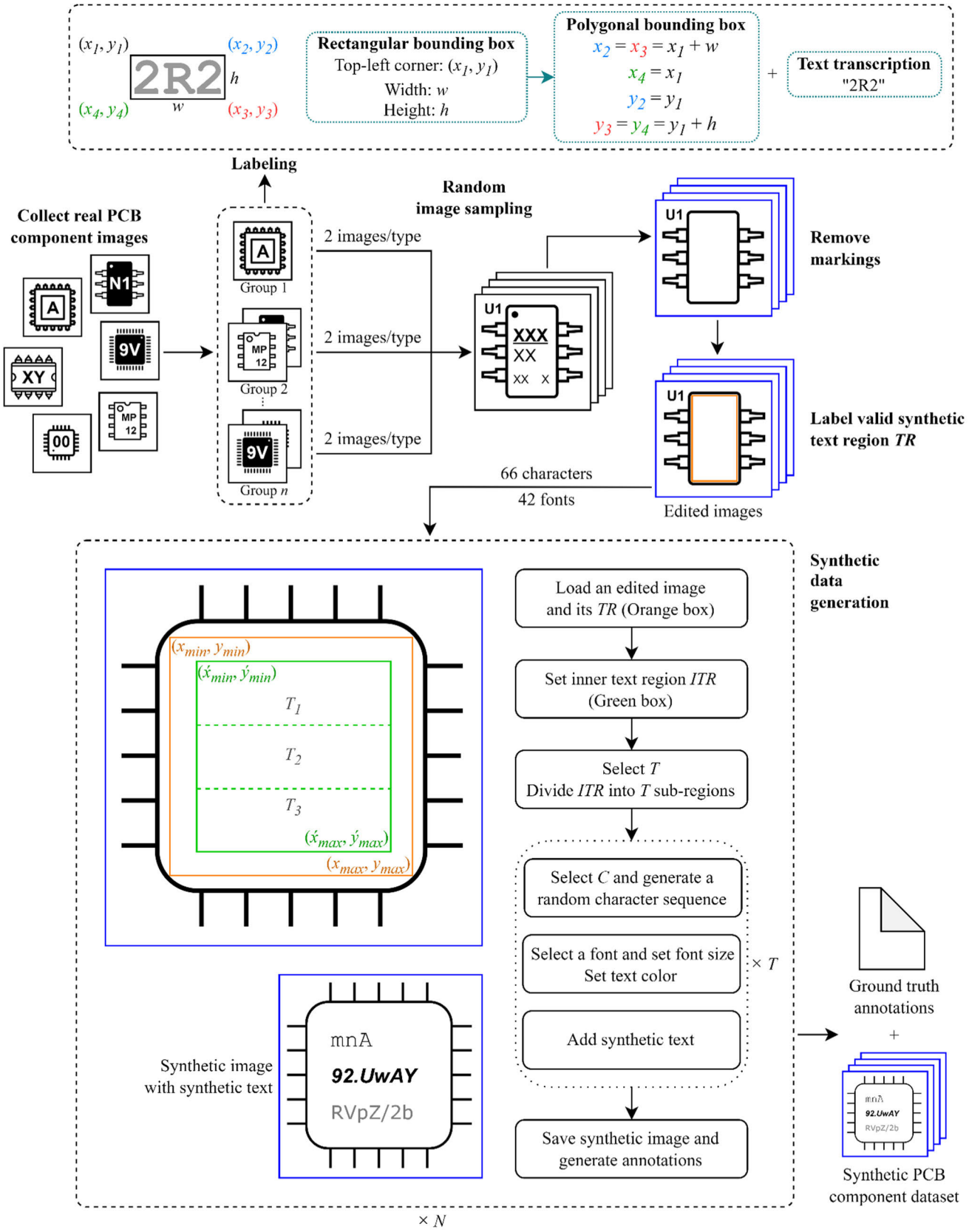

FIGURE 5. Preparation process for real PCB component dataset and synthetic PCB component dataset. T, C, and N represent the number of synthetic text per image, the length of a synthetic text, and the number of synthetic images to be generated respectively.

text without overlapping of text instances; therefore, we can treat the rectangle bounding box as a polygon that has four points. Although the small number of slightly-oriented text markings may include some background in the bounding boxes, they will not cause the text spotting model to fail right away. Hence, we can utilize Text Perceptron to detect and recognize arbitrary-shaped text markings without going through strenuous and costly labeling work to obtain tight polygonal bounding boxes for text markings on PCB components. The text transcriptions for recognizable text markings are kept the same, whereas those that need to be ignored are labeled as '###' as in [2].

Based on the ground truth annotations of real dataset, we count the number of samples by character classes as shown in Fig. 6 and we observe the imbalance of data. First, digits and uppercase alphabets appear frequently, whereas lowercase alphabets and symbols have little or no samples for deep learning model training. Second, the samples in character classes "a", "i", "k", "n", and "s" are mostly duplicates from the repeated text marking "SanDisk" which have similar appearances. We do not opt for data augmentation on the real dataset because there are seven character classes that have no samples to be augmented. Instead, we generate a synthetic PCB component dataset that covers 66 character classes and more diverse synthetic text to handle the imbalanced distribution of data in the real dataset.

As shown in Fig. 5, our proposed synthetic dataset is generated based on real PCB component images through:

1) **Random image sampling.** An image group of real dataset may contain more than one type of PCB components with dissimilar structures. For each of the component type, we randomly select two images. This step is repeated for all image groups and a total of 758 real images are sampled.

2) **Removal of PCB component markings.** We manually remove all the markings (text and non-text) on a PCB component by clone stamping the component's background pixels with no markings to the regions with markings. Hence, the edited regions retain the information of the original background pixels, including the level of noise, sharpness or blurriness, and the lighting condition. Any text instance outside of the PCB component region, e.g., reference designator on PCB is not removed. The graphics software that we used is Microsoft Paint. This step is repeated for all sampled images and 758 edited images are obtained.

3) **Labeling of valid synthetic text region.** We label the component region that can be added with synthetic text for an edited image as the valid synthetic text region *TR*. This step is repeated for all edited images and 758 annotations for valid synthetic text regions are obtained.

4) **Generation of synthetic text, synthetic images, and ground truth annotations.** The character classes for synthetic text cover 66 characters, including 10 digits, 52 Latin alphabets, and 4 symbols (period, underscore, slash, and dash), while excluding symbols that are less likely to be used in real text marking. We select 42 TrueType fonts which consist of 28 Regular styles, 9 Bold styles, 3 Italic styles, and 2 Bold Italic styles. Based on the edited PCB component images, we generate the synthetic dataset using the synthetic data generator to carry out the following steps:

a) Set N number of synthetic images to be generated.
b) Load an edited PCB component image and the annotations of its valid synthetic text region *TR*.
c) Set an inner text region *ITR* based on *TR*'s width W_{TR} and height H_{TR} in pixels by padding the sides of *TR*:

$$\begin{aligned} p_{lr} &= 0.1 \times W_{TR} \\ p_{tb} &= 0.1 \times H_{TR} \\ \acute{x}_{min} &= x_{min} + p_{lr} \\ \acute{y}_{min} &= y_{min} + p_{tb} \\ \acute{x}_{max} &= x_{max} - p_{lr} \\ \acute{y}_{max} &= y_{max} - p_{tb} \end{aligned} \tag{1}$$

where p_{lr} is the left/right padding, p_{tb} is the top/bottom padding, (x_{min}, y_{min}) and (x_{max}, y_{max}) are the top-left corner and bottom-right corner of *TR* respectively, and $(\acute{x}_{min}, \acute{y}_{min})$ and $(\acute{x}_{max}, \acute{y}_{max})$ are the top-left corner and bottom-right corner of *ITR* respectively.

d) Set the range for the number of synthetic text to be generated based on *ITR*'s height H_{ITR} in pixels:

$$\text{Minimum} = \begin{cases} 1 & 0 < H_{ITR} \leq 200 \\ 3 & 200 < H_{ITR} \leq 400 \\ 4 & 400 < H_{ITR} \leq 600 \\ 5 & \text{Otherwise} \end{cases}$$

$$\text{Maximum} = \begin{cases} 1 & 0 < H_{ITR} \leq 100 \\ 2 & 100 < H_{ITR} \leq 200 \\ 4 & 200 < H_{ITR} \leq 400 \\ 5 & 400 < H_{ITR} \leq 600 \\ 6 & \text{Otherwise} \end{cases} \tag{2}$$

The number of synthetic text T to be generated in *ITR* is randomly chosen within this range. Divide *ITR* into T sub-regions of equal height.

e) For each of the synthetic text to be generated, set the range for the number of characters based on *ITR*'s width W_{ITR} in pixels:

$$\text{Minimum} = \begin{cases} 3 & 0 < W_{ITR} \leq 200 \\ 4 & 200 < W_{ITR} \leq 400 \\ 5 & 400 < W_{ITR} \leq 600 \\ 6 & 600 < W_{ITR} \leq 800 \\ 7 & \text{Otherwise} \end{cases}$$

$$\text{Maximum} = \begin{cases} 4 & 0 < W_{ITR} \leq 200 \\ 5 & 200 < W_{ITR} \leq 400 \\ 6 & 400 < W_{ITR} \leq 600 \\ 7 & 600 < W_{ITR} \leq 800 \\ 8 & \text{Otherwise} \end{cases} \tag{3}$$

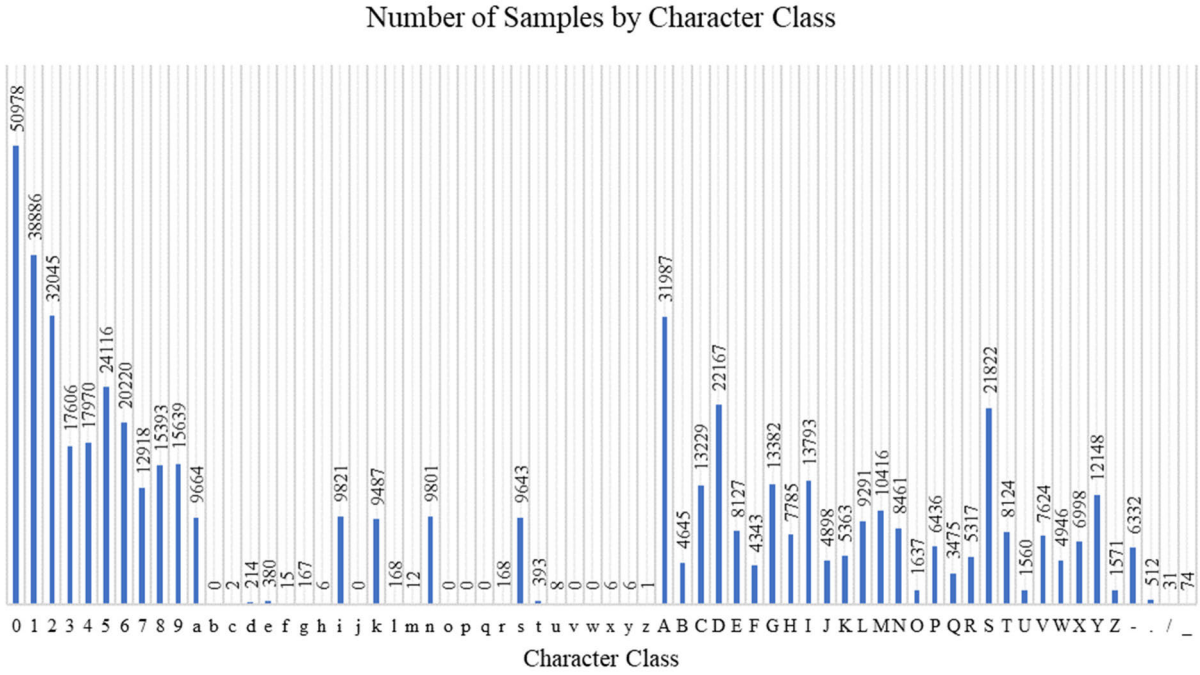

FIGURE 6. Imbalanced distribution of character classes in real dataset. Lowercase alphabets and symbols are less than digits and uppercase alphabets. Lowercase alphabets "b", "j", "o", "p", "q", "v", and "w" are character classes with no collected samples.

The number of characters C to be generated for each synthetic text is randomly chosen within this range. Based on the value of C, generate a character sequence by randomly choosing one character at a time from the 66 characters. Repetition of characters is allowed. The generated character sequence is accepted only if the first and the last character is not a symbol, and it does not contain two or more consecutive symbols. Otherwise, the generation of a new character sequence repeats until the conditions above are met.

f) For each generated synthetic text, randomly choose a font from the 42 provided fonts. If T is greater than 3, the default font size is 60, otherwise 30. Based on the selected font, compute the width W_T and height H_T of synthetic text using the default font size. The font size is applied only if the values of W_T and H_T do not exceed the values of W_{ITR} and H_{ITR} respectively. Otherwise, the computation repeats by reducing the font size by 2 until the conditions above are met or when the font size is 2. The final coordinates of the horizontal rectangular bounding box of synthetic text are recorded. Next, the text color is determined based on the background color of the recorded text region. The color of the text region is inverted and the first most frequent inverted color is set as the text color. Then, the synthetic text is added in its text region with top-left alignment.

g) After adding T number of synthetic text, the final synthetic image is saved. The ground truth annotations are recorded according to the format used in [2], including information on the image's width and height, all the synthetic text regions and their text transcriptions, in which the 'care' flag is set to 1 for all synthetic texts.

h) Repeat steps (b) to (g) for N times.

To create synthetic text that resembles text markings on real PCB components, we have carefully chosen the character classes, fonts, and styles to be used for the generation of random synthetic text. We follow the real dataset which only covers digits, Latin alphabets and some symbols, while characters in other languages are not included. Some font examples that we have selected are 8Pin Matrix Regular, Verdana Bold, Century Gothic Italic, and Century Gothic Bold Italic. The majority of the selected fonts are in Regular styles because we observe more text markings in this style on real PCB components compared to the other three styles. Instead of augmenting the synthetic images at the synthetic data generation stage, we perform random augmentations on the images during training as in the settings of [58] to further increase the diversity of the training data. The data augmentation methods include random crop, random jitter, and random rotation, in which random jitter can introduce changes in image brightness, contrast, saturation, and hue. Nevertheless, augmentation methods such as adding image noise, blurring, flare or spots are not carried out to ensure the legibility of text in the images, since the real dataset contains images that have these kinds of factors.

During the generation of synthetic text, the sides of synthetic text are padded so that they will not exceed the area of PCB components. We take into account the height and width of the valid synthetic text region when setting the number of synthetic text to be added and the length of

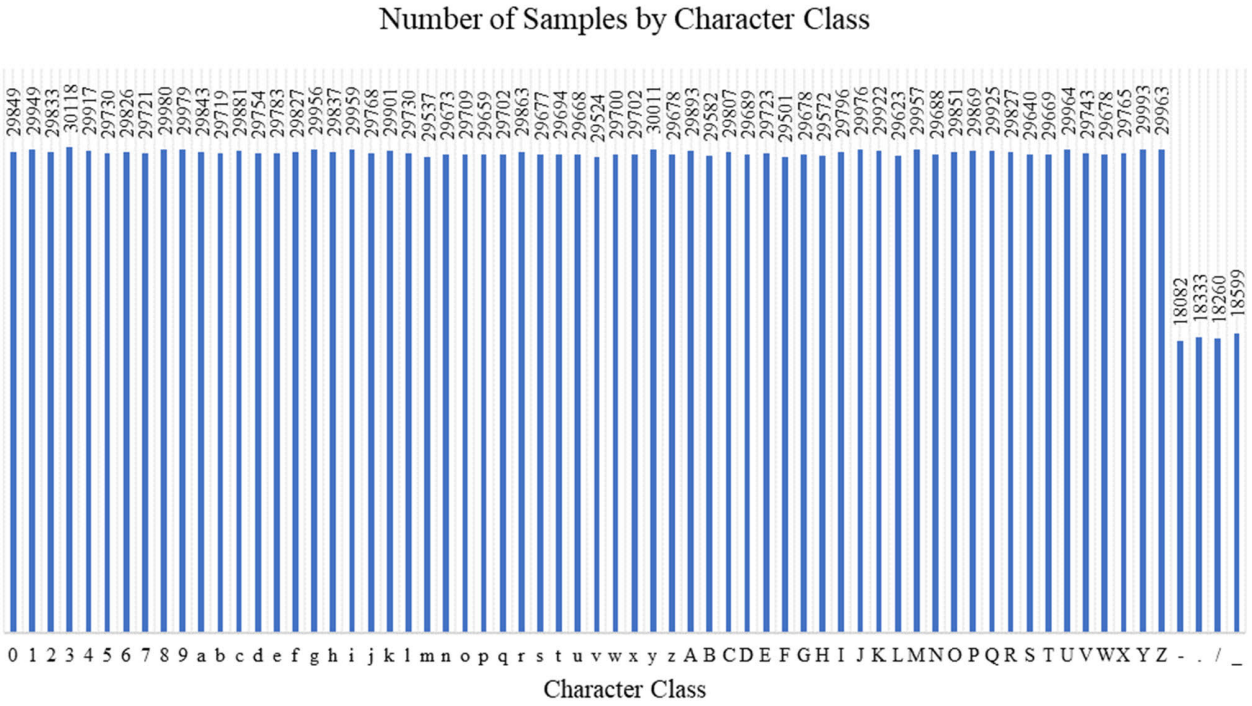

FIGURE 7. Synthetic dataset that covers 66 character classes. Digits, lowercase alphabets, and uppercase alphabets are evenly distributed, whereas symbols are less than them due to designed rules for the generation of synthetic text.

synthetic text respectively. A valid synthetic text region with greater height in pixels can be added with more rows of synthetic texts, and that with greater width in pixels can be added with a longer synthetic text. Furthermore, we design rules to validate the generated character sequence, especially on symbols based on the general pattern observed in real PCB component text markings and the logical usage of symbols. As a result, symbol classes have lesser samples compared to digits and alphabets classes. Fig. 7 shows the number of samples by character classes of the generated synthetic dataset. After a random font is selected for a synthetic text, we apply the largest possible font size within its text region, in which the text color is determined based on the inverted background color to make the synthetic text more legible.

B. DEEP LEARNING-BASED TEXT SPOTTING ON PCB COMPONENT IMAGES

In this study, we use a deep learning model, Text Perceptron [2] to perform text spotting on PCB components. It sequentially detects and recognizes each text marking in a PCB component image by predicting a word-level polygonal bounding box and a character sequence. Text Perceptron consists of a segmentation-based text detection module, a Shape Transform Module (STM), and an attention-based text recognition module [2] as shown in Fig. 8.

The text detection module can efficiently separate irregular text instances using a ResNet-50-FPN backbone which simultaneously learns three tasks: 1) order-aware semantic segmentation; 2) corner regression; and 3) boundary offset regression. The text detector learns multiclass semantic segmentation to locate the text instances, in which text boundary segmentation is used to separate different text instances. Corner and boundary offset regression tasks are learned to improve the segmentation of irregular text and to provide position information of fiducial points around the text. To handle text with arbitrary shapes, pixels around the center text region are categorized into head, tail, top and bottom boundaries as shown in Fig. 9(b). Moreover, a pair of head and tail boundaries may capture information on the reading order of a text such as from left to right or from top to bottom. The corner regression task is to regress the geometry offsets of head pixels to the two corner points (P_1 and P_{2n}) in the head region while regressing the geometry offsets of tail pixels to the two corner points (P_n and P_{n+1}) in the tail region. Meanwhile, the boundary offset regression task is to regress the vertical and horizontal geometry offsets of center pixels to their nearest boundaries. After computing the geometry offset values for pixels in head, tail, and center regions respectively, the geometry maps of these three regions are obtained. During forward processing, the predicted segmentation maps are obtained by overlaying the segmented head, tail, top and bottom, and center feature maps, where center pixels are the text. The training process of multiclass semantic segmentation uses Dice coefficient-based loss [59], whereas each regression task uses Smooth-L1 loss.

STM plays an important role in uniting the text detection and text recognition modules to make Text Perceptron an end-to-end trainable text spotter. It takes the predicted segmentation maps and geometry maps produced by the

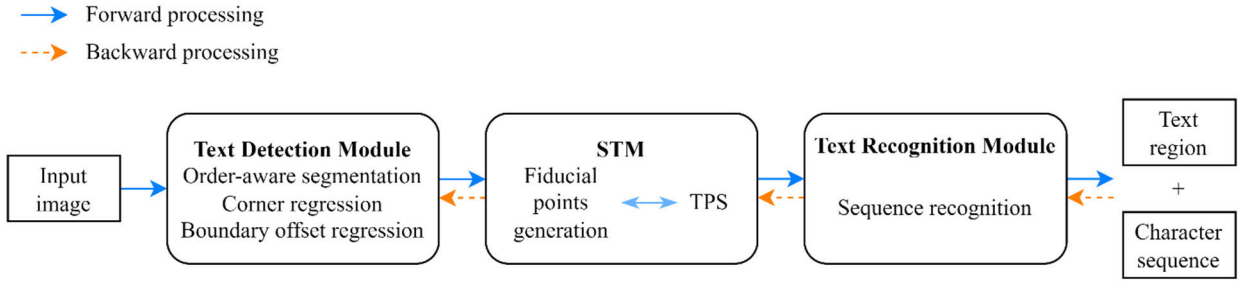

FIGURE 8. Workflow of Text Perceptron [2]. Blue arrows and orange arrows indicate forward processing and backward processing respectively. The text detector with ResNet-50-FPN backbone locates the text markings on PCB components by producing segmentation maps and geometry maps. Based on the predicted text regions, STM iteratively generates fiducial points around each text instance and rectifies irregular text using TPS. The attention-based FAN [61] is adopted as the text recognizer to output the character sequence.

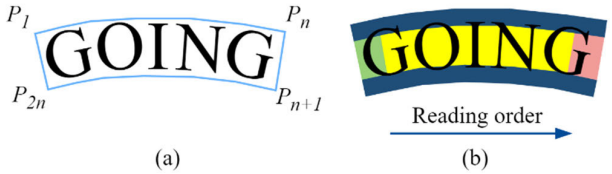

FIGURE 9. (a) Ground truth polygonal bounding box. The value of n is set to 7, which is the same value used in [2]. (b) Predicted segmentation map by the text detector. Pixels around the center text region (Yellow) are categorized into head (Green), tail (Pink), top and bottom (Blue) boundaries. The reading order information may be captured by the head and tail boundaries.

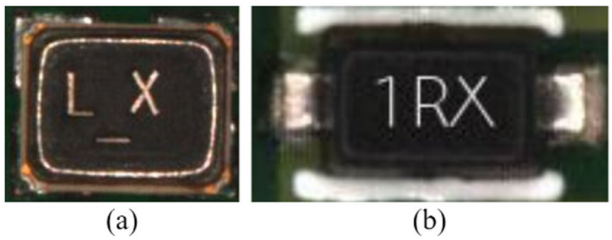

FIGURE 10. (a) Real PCB component image. (b) Synthetic PCB component image generated using the synthetic data generator.

text detection module to iteratively generate fiducial points around the text and rectifies the irregular text feature regions into regular form using thin-plate splines (TPS) [60] based on the supervision of text recognition module. Each text instance has at least four corner fiducial points being generated, i.e., two in head region and another two in tail region before the generation of the remaining fiducial points using dichotomous method that also applies to arbitrary-shaped text. We follow the implementation in [2] which uses 14 fiducial points for a text instance, meaning that each top and bottom side of the text has 7 points. After transforming the text feature regions into regular shapes, they are passed to the text recognition module.

The text recognition module in Text Perceptron [2] directly adopts the Focusing Attention Network (FAN) in [61] to produce the final character sequence. The main modules of FAN are the attention-based RNN decoder called Attention Network (AN) and the Focusing Network (FN) with a focusing mechanism, in which AN recognizes the target characters from the extracted features and FN automatically adjusts the deviated attention of AN back to the proper target character areas [61]. During the backward processing of Text Perceptron, the differences between the predictions and ground truths are back-propagated to all pixel values in the irregular text feature regions through STM to make adjustments on the fiducial points. Then, the adjustment values are back-propagated to the corresponding geometry maps in the head, tail, and center regions. Standard cross-entropy loss is used in the training of text recognition module.

IV. EXPERIMENT AND RESULTS

A. DATASET

Two private PCB component datasets are used in this paper, i.e., a real dataset and a synthetic dataset. Fig. 10 shows examples of real and synthetic PCB component images respectively. The real dataset is split into 16,795 train images, 1,793 validation images, and 8,817 test images. These PCB component images are collected at the actual production line of SanDisk Storage Malaysia Sdn. Bhd., a company of Western Digital. While excluding non-text markings, defective text markings, and vertical text markings, only horizontal text markings on real PCB components will be trained and contribute to the performance of Text Perceptron.

In the synthetic dataset, 50,000 train images, 5,555 validation images, and 23,810 test images are generated using our synthetic data generator based on a small sample of edited real PCB component images. All the original markings on these sampled PCB components are removed, and then, horizontal left-aligned synthetic texts with random character sequences are overlaid on the images. A total of 66 character classes are covered during the generation of synthetic text, including 10 digits, 52 Latin alphabets, and 4 symbols that are present in the real dataset, i.e., period, underscore, slash, and dash. Hence, the synthetic dataset alleviates the imbalanced data in the real dataset and removes the hindrance to the goal of having a text spotter that can detect and recognize digits, alphabets, and specific symbols on PCB components. Furthermore, it provides synthetic text with diverse text variations in terms of font, style, size, and color.

In both datasets, the region of each ground truth text instance is represented by a word-level polygonal bounding

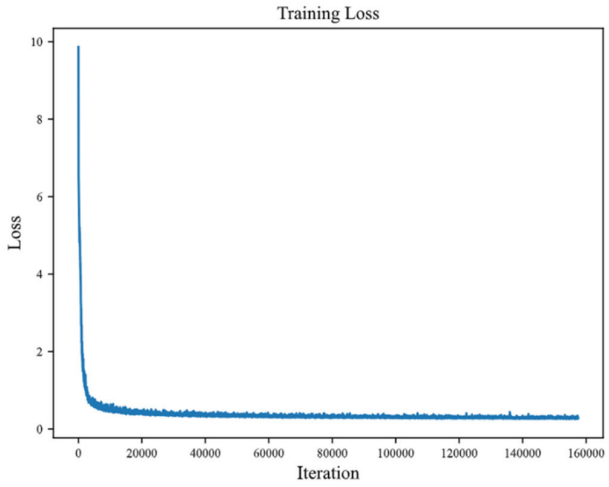

FIGURE 11. Training loss curve of text perceptron.

box with four pairs of (x, y) coordinates that represent the top-left, top-right, bottom-right, and bottom-left corners respectively, and its corresponding ground truth text transcription is labeled as a word-level text. During training, the mixture of real train set and synthetic train set follows the ratio of 1:3. At inference stage, Text Perceptron is tested on real test set and synthetic test set separately.

B. EVALUATION METRICS

In [2], the performance of Text Perceptron on Total-Text [13] is evaluated for both text detection and end-to-end text spotting tasks. We use the same evaluation protocol to evaluate our trained Text Perceptron on the PCB component datasets. The precision, recall, and F-score of text detection task are calculated based on the selected IoU threshold value, which is 0.5. For end-to-end text spotting task, F-score is calculated using the same IoU threshold value of 0.5 based on the 'None' metric that is lexicon-free. This metric is more relevant to the text spotting task in this paper compared to the 'Full' metric that needs a lexicon for the evaluation of text spotting model. Given a ground truth polygon and a predicted polygon, if the IoU is below 0.5, the predicted polygon will be suppressed to reduce false detection. Otherwise, the predicted text within this predicted polygon is compared with the corresponding ground truth text transcription. Only an exact match contributes to the text spotting performance. If not, it is considered a false positive.

Another aspect to be evaluated is the speed of Text Perceptron. In this paper, inference time refers to the elapsed time for model prediction, which excludes elapsed time to read the input image and postprocess on output. For each resized test image, out of the five recorded inference time, the minimum and maximum time are discarded, and the average of the remaining three is calculated. Then, an averaging of all these averages is carried out to obtain the average inference time of Text Perceptron.

C. IMPLEMENTATION DETAILS

We follow the original work of Text Perceptron [2] based on the implementation provided by DavarOCR [58], which is an open-source OCR toolbox. We train the Text Perceptron in an end-to-end manner from scratch without using pretrained weight for 150 epochs with AdamW optimizer and fixed learning rate at 3×10^{-4}. Fig. 11 shows the training loss curve of Text Perceptron on the mixed train sets which generally decreases with the number of iterations.

Throughout the training of Text Perceptron, the checkpoint with the highest end-to-end text spotting F-score on the validation set is recorded as the best checkpoint. In this paper, we use the final best checkpoint to evaluate the performance of Text Perceptron. Fine-tuning using the real train set is not performed because the real dataset does not have full coverage on the 66 character classes as mentioned in Section III-A. The data augmentation steps applied on the real and synthetic PCB component images during training follow the settings in [58], including random crop, random jitter, and random image rotation within the range of $[-15°, 15°]$. For both training and inferencing, the PCB component images are resized by setting its longer side to 384 pixels while keeping its original aspect ratio. Training of Text Perceptron runs in PyTorch with 2 NVIDIA Tesla V100 SXM2 32GB GPUs, whereas inferencing runs on single GPU.

D. RESULT AND DISCUSSION

After training and evaluating Text Perceptron using PCB component images, the precision, recall, and F-scores for text detection task, as well as the end-to-end text spotting F-scores are recorded as shown in Table. 2. The experimental results show that the model achieves satisfactory text detection F-scores under all settings, which is above 99%. The model is trained on slightly rotated PCB component images; thus, it can detect most text regions in both test sets which are mainly horizontal text. With data augmentation, the text detection F- scores for real test set increase. Generally, we say that the text detection module of Text Perceptron serves the purpose of finding text marking regions on both real and synthetic PCB component images.

For end-to-end text spotting task, the model achieves an encouraging F-score of 90.16% on the real PCB component test images, which is also the highest F-score obtained by training on both datasets with data augmentation. Fig. 12 shows examples of correct text spotting predictions made by Text Perceptron on PCB component images. Still, there is room for improvement for end-to-end text spotting F-scores on synthetic test images as shown in Table. 2. The. performance gap of end-to-end text spotting between real images and synthetic images is possibly due to the randomness of character sequences in the synthetic dataset that do not follow any string pattern, unlike real text markings that have duplicated character sequences. It is also possible that the text variations in terms of font, styles, size, and color are more than that in the real dataset, making the

TABLE 2. Result on synthetic test set and real test set.

Train Set	Validation Set	Data Augmentation (Training)	Synthetic Test Set				Real Test Set			
			Text Detection			End-to-end Text Spotting (None metric)	Text Detection			End-to-end Text Spotting (None metric)
			P	R	F	F	P	R	F	F
Synthetic	Synthetic	×	99.98	99.52	99.75	64.38	–	–	–	–
		✓	99.99	98.19	99.08	**69.53**	–	–	–	–
Real	Real	×	–	–	–	–	99.82	99.46	99.64	88.18
		✓	–	–	–	–	99.86	99.73	99.80	88.08
Synthetic + Real		×	99.99	99.44	99.71	63.38	99.76	99.52	99.64	82.95
		✓	100.00	98.45	99.22	66.17	99.88	99.71	99.79	**90.16**

Data augmentation methods that are disabled/enabled during training include random crop, random jitter (brightness, contrast, saturation, hue), and random rotation on training images. "P", "R", and "F" represent precision, recall, and F-score respectively.

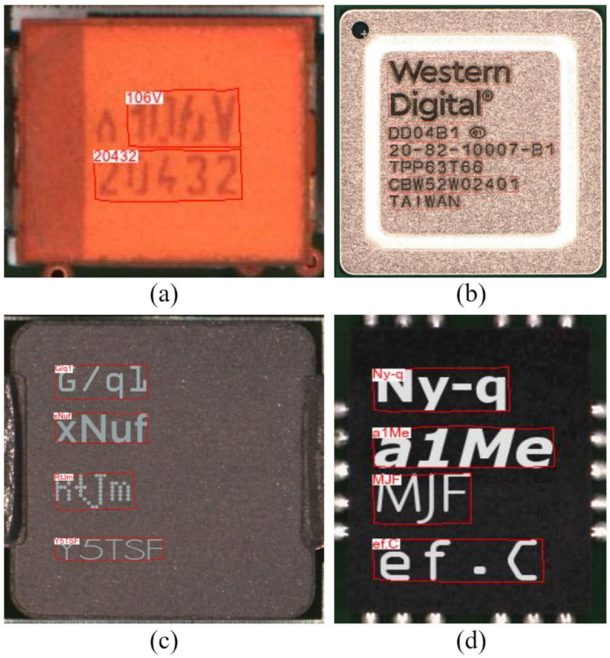

FIGURE 12. Correct text spotting results predicted by the trained model for (a) text markings on real PCB component images and (b) synthetic text on synthetic images.

FIGURE 13. Examples of failure cases for real PCB component test images. (a) Digit "9" is predicted as "8". (b) Alphabet "D" is predicted as digit "0".

text recognition task on synthetic PCB component images to be more challenging.

From another aspect, the text spotting performance of Text Perceptron reveals the limitations of real and synthetic datasets. Firstly, the repetition of text markings, e.g., "SanDisk", "MALAYSIA", or part numbers of PCB components in the real train set can be a double-edged sword. On the one hand, at inference stage, the trained model may give a correct prediction for test input that has the same character sequence as in train data, though not every character of the test input is clear and free from printing defects. It is beneficial to the inferencing on long text markings which may contain some slightly blurred or faded characters. On the other hand, a test input with one or more blurred characters is possible to be predicted incorrectly as other similar character sequence that occurs repeatedly in train data. Secondly, the synthetic data generator is designed to generate character sequences in a completely random manner. It does not take into account the common patterns of text markings in real dataset. For example, the synthetic data generator may generate a synthetic text such as "g.S_6/T" which fulfills the designed rules for symbols; nevertheless, it is less likely to observe this kind of pattern in real text markings on PCB components with 3 symbols out of 7 characters. Moreover, despite the number of training samples generated for each character class, the trained model fails to distinguish characters that look alike as shown in Fig. 13.

To further evaluate the performance of our synthetic dataset, we study the predicted text sequences of the real test set that have equal length with their corresponding ground truths. Assuming that a pair of ground truth and predicted text sequences can be compared character-wise, the precision of the predicted results is calculated by character group, including digit, lowercase and uppercase alphabets, and symbols as shown in Table. 3. Note that we have excluded the pairs of ground truth and predicted text sequences that have different lengths from the calculation of precision by character group because such pairwise sequence alignment may result in one or more optimal pairings. Based on Table. 3, when synthetic data is used for model training, it has increased the precision for digit and uppercase alphabet character groups by 1.17% and 0.41% respectively, which demonstrates the benefit of our proposed method since the actual PCB component text markings usually contain more digits and uppercase alphabets. Additionally, from the character-wise comparison, we observe that the synthetic data has reduced the total number of incorrect character pairs (e.g., "A-B", if an alphabet "A" is being predicted as "B") by 26% from 4257 to 3150 pairs. Meanwhile, the drop in precision for the symbol character group is possibly due to fewer samples in the synthetic dataset compared to other character groups as described in Section III-A.

TABLE 3. Precision of predicted text sequences (real test set).

Synthetic Train Set	Character Group			
	Digit	Alphabet (Lowercase)	Alphabet (Uppercase)	Symbol
×	96.68	99.97	97.30	99.77
✓	**97.85**	99.95	**97.71**	99.37

Data augmentation methods are enabled during training.

The average inference time of Text Perceptron on the real test set is 87.75 milliseconds per image. This result shows that the model can detect and recognize text markings in PCB component images at an acceptable average inference time per image. In future research, the accuracy-speed trade-off needs to be tackled, for example, the network size may be deeper to achieve better end-to-end text spotting performance while maintaining the inference speed.

E. COMPARISON WITH THE STATE-OF-THE-ART

Brief descriptions for the state-of-the-art methods [17], [18], [19], [20] and our proposed method along with the respective text recognition or text spotting performance on PCB component images are summarized in Table. 4. In comparison with the methods in [17], [18], [19], and [20], the proposed method appears promising in several regards. First, its performance is comparable if not better than the previously proposed methods. As shown in the table, the increase in performance of the deep learning model for our proposed method is about 2.08%. In [17], a grid-based sampling method [62] for data reduction, data augmentation methods, and a synthetic data generation method are applied to produce a balanced dataset consisting of 10 digits and 42 Latin alphabets. Lowercase alphabets that have similar forms to their capital letters are ignored, for example, ''c'' and ''C''. For each character class, the mean and standard deviation of pixel values in the V color channel are visualized through a 2D data distribution graph. After dividing the graph into small grids, the grid algorithm randomly samples a percentage of data, e.g., 10%, from each of the grids to maintain the overall distribution of the original dataset. Furthermore, augmented images are generated through random slight rotation (left and right), image resizing, and random Gaussian noise, whereas synthetic character images are created using 35 fonts with empirically-defined color tones to provide different character images for the training of classification model. The proposed method is trained and evaluated on different combinations of two real datasets and two generated datasets. A mixture of real and synthetic images containing 8,000 images per class without augmented images gives the highest improvement of 20.84% in the model performance. In other words, the data augmentation methods in [17] are less effective in generating a representative dataset for model training compared to synthetic data generation.

The work in [18] follows the grid-based sampling method and synthetic data generation method in [17] for generating a balanced dataset while using data augmentation methods such as color reversal, random rotation, image resizing, and random Gaussian noise. Still, lowercase alphabets are conditionally excluded as in [17]. The combination of two real datasets and one generated dataset sampled using the grid-based method increases the model performance by 17.84%. In addition, an n-pick ($n = 3$) algorithm that samples n data from each of the grids in the data distribution graphs is proposed to reduce the size of dataset. The proposed methods in [17] and [18] handle the text on PCB components at character level, which allows the methods to reduce redundant data in original datasets while generating new synthetic data. Also, a short inference time is needed to classify each character image. However, these character-based methods have limited application whereby the methods cannot be used to support the training of deep learning models that operate at a higher semantic (word or line) level to capture contextual information between characters.

As mentioned in Section I, the quality of printed characters is often affected throughout the production process. This could make character segmentation before character classification to be more difficult. Hence, the proposed method in [19] opts for word-level text recognition on PCB components. Since the real data in [19] is very limited, fully-synthetic 8-bit grayscale text images at word level are created. For synthetic text generation, seven fonts are included but all lowercase alphabets are ignored. Each synthetic image contains a synthetic text with two to eight characters, in which the remaining regions for shorter text are filled with black pixels to maintain the fixed size of all synthetic images. Random data augmentation methods such as speckle noise, rotation, horizontal/vertical scaling, and image shearing are applied to the synthetic images. After using the synthetic dataset and real dataset to pretrain and fine-tune the model respectively, the evaluation using the remaining real data results in text recognition accuracy of 93.80%. From Table. 4, we can see that the inference time of [19] is longer compared to [17] and [18] as it increases with the number of characters in a single test image. On the one hand, large datasets can be obtained by creating fully-synthetic images to support model training when it is hard to collect real PCB component data [17], [18], [19]. On the other hand, these synthetic images do not incorporate actual background information on PCB components.

The generation of partially-synthetic images in [20] for balancing the dataset involves character-level replacement of original characters on real PCB component images with synthetic lowercase alphabets. The synthetic characters can be white or colorful in Windows fonts, or are generated through text style transfer [27]. Results in [20] show that the synthetic dataset increases the model performance by 1.80%. In terms of inference speed, it is unsurprising that more time is needed in [20] like our work to process all characters in each PCB component image compared to [17], [18], and [19]. Though real images are used in [20] for synthetic data generation, the noticeable edited regions on

TABLE 4. Description of state-of-the-art methods and their performance.

Relevant Study	Description	Text Recognition/Text Spotting Performance
S. M. Gang and J. J. Lee [17]	A grid-based sampling method [62] that leverages the data distribution of original dataset, working together with data augmentation methods, and a character-level synthetic data generation method for producing a balanced dataset. Different combinations of real, augmented, and synthetic images are sampled and mixed for training the deep learning model to identify the representative dataset.	The performance of the deep learning model trained on the representative dataset compared to that which is trained on original datasets are 97.76% and 76.92% respectively. The increase in performance is 20.84%. The computational time per character image on GPU is 0.28 milliseconds.
S. Gang et al. [18]	A grid-based sampling method [62] that leverages the data distribution of original dataset, along with data augmentation methods, and a character-level synthetic data generation method for producing a balanced and representative dataset. Then, a smaller base set for future learning is sampled using *n*-pick sampling method.	The performance of the deep learning model trained on the representative dataset compared to that which is trained on original datasets are 97.76% and 79.92% respectively. The increase in performance is 17.84%. The computational time per character image on GPU is 0.28 milliseconds.
T. H. Cho [19]	A segmentation-free text recognition method for real-time PCB component inspection. Word-level synthetic images are generated due to very limited real data for training the deep learning model.	The performance of the deep learning model pretrained on synthetic dataset and fine-tuned on real dataset is 93.80%. The inference time for a text image with 7 characters on GPU is around 3 milliseconds.
Q. Wang et al. [20]	A leading character-level text spotting method in the RRC-ICText 2021 competition. Synthetic dataset is generated through character-level replacement of original characters on real PCB component images with synthetic lowercase alphabets for balancing the dataset and training the deep learning model.	The performance of the deep learning model trained on the dataset with and without synthetic data are 59.60% and 57.80% respectively. The increase in performance is 1.80%. The inference speed per PCB component image on GPU is at 31.04 fps.
Proposed Method	A synthetic PCB component data generation method that leverages real PCB component images through removal of original markings on real PCB components, followed by overlaying of word-level synthetic text for balancing the dataset and training the deep learning model.	The performance of the deep learning model trained on the dataset with and without synthetic data are 90.16% and 88.08% respectively. The increase in performance is 2.08%. The training takes around 5 to 6 days and the average inference time on GPU is 87.75 milliseconds per PCB component image.

the resulting synthetic images are making them look less realistic.

Our proposed method involves the removal of original markings on real PCB component images and overlaying of word-level synthetic text on the edited images, which leaves no noticeable edited regions. This study differs from the state-of-the-art methods in [17], [18], [19], and [20] as it leverages real PCB component images for generating more realistic partially-synthetic images that are overlaid with word-level synthetic text data. Furthermore, it is the only work that considers not only all lowercase alphabets but also some reasonable symbols as part of the character classes, regardless of their occurrence count in actual PCB component markings. Last but not least, the synthetic dataset demonstrates the flexibility of proposed method in synthetic text generation, which creates a wide range of text variations using 42 fonts with different styles, sizes, and colors.

V. CONCLUSION

This paper generated a synthetic PCB component dataset based on real PCB component images using a synthetic data generator. The synthetic dataset has covered 66 character classes and provided synthetic text data with diverse text variations in terms of font, style, size, and color. Hence, it alleviated the imbalance of samples between character classes and within a character class observed in the real dataset. We trained Text Perceptron with real and synthetic datasets to perform text spotting on PCB component images. It achieved promising text detection F-score on PCB component images while showing encouraging end-to-end text spotting F-score on real images. Moreover, it met the acceptable average inference time per image. Still, effort is needed to improve its end-to-end text spotting performance such as using a deeper text recognition model while maintaining the inference speed. A possible improvement on the synthetic dataset is to generate synthetic text data based on the patterns of actual text markings on PCB components.

REFERENCES

[1] K. Maliński and K. Okarma, "Analysis of image preprocessing and binarization methods for OCR-based detection and classification of electronic integrated circuit labeling," *Electronics*, vol. 12, no. 11, p. 2449, May 2023, doi: 10.3390/electronics12112449.

[2] L. Qiao, S. Tang, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "Text perceptron: Towards end-to-end arbitrary-shaped text spotting," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 1609, vol. 34, no. 7, pp. 11899–11907, doi: 10.1609/aaai.v34i07.6864.

[3] M. Liao, G. Pang, J. Huang, T. Hassner, and X. Bai, "Mask textspotter v3: Segmentation proposal network for robust scene text spotting," in *Proc. 16th Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 706–722, doi: 10.1007/978-3-030-58621-8_41.

[4] Y. Liu, C. Shen, L. Jin, T. He, P. Chen, C. Liu, and H. Chen, "ABCNet v2: Adaptive Bezier-curve network for real-time end-to-end text spotting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8048–8064, Nov. 2022, doi: 10.1109/TPAMI.2021.3107437.

[5] Y. Liu, J. Zhang, D. Peng, M. Huang, X. Wang, J. Tang, C. Huang, D. Lin, C. Shen, X. Bai, and L. Jin, "SPTS v2: Single-point scene text spotting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 12, pp. 15665–15679, Dec. 2023, doi: 10.1109/TPAMI.2023.3312285.

[6] L. Xing, Z. Tian, W. Huang, and M. Scott, "Convolutional character networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9125–9135, doi: 10.1109/ICCV.2019.00922.

[7] P. Wang, C. Zhang, F. Qi, S. Liu, X. Zhang, P. Lyu, J. Han, J. Liu, E. Ding, and G. Shi, "PGNet: Real-time arbitrarily-shaped text spotting with point gathering network," in *Proc. AAAI Conf. Artif. Intell.*, May 1609, vol. 35, no. 4, pp. 2782–2790, doi: 10.1609/aaai.v35i4.16383.

[8] L. Qiao, Y. Chen, Z. Cheng, Y. Xu, Y. Niu, S. Pu, and F. Wu, "MANGO: A mask attention guided one-stage scene text spotter," in *Proc. AAAI Conf. Artif. Intell.*, May 1609, vol. 35, no. 3, pp. 2467–2476, doi: 10.1609/aaai.v35i3.16348.

[9] X. Zhang, Y. Su, S. Tripathi, and Z. Tu, "Text spotting transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 9509–9518, doi: 10.1109/CVPR52688.2022.00930.
[10] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazàn, and L. P. de las Heras, "ICDAR 2013 robust reading competition," in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 1484–1493, doi: 10.1109/ICDAR.2013.221.
[11] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, "ICDAR 2015 competition on robust reading," in *Proc. 13th Int. Conf. Document Anal. Recognit. (ICDAR)*, Aug. 2015, pp. 1156–1160, doi: 10.1109/ICDAR.2015.7333942.
[12] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "COCO-text: Dataset and benchmark for text detection and recognition in natural images," 2016, *arXiv:1601.07140*.
[13] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 01, Nov. 2017, pp. 935–942, doi: 10.1109/ICDAR.2017.157.
[14] L. Yuliang, J. Lianwen, Z. Shuaitao, and Z. Sheng, "Detecting curve text in the wild: New dataset and new solution," 2017, *arXiv:1712.02170*.
[15] C. K. Chng, Y. Liu, Y. Sun, C. C. Ng, C. Luo, Z. Ni, C. Fang, S. Zhang, J. Han, E. Ding, J. Liu, D. Karatzas, C. S. Chan, and L. Jin, "ICDAR2019 robust reading challenge on arbitrary-shaped text-RRC-ArT," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 1571–1576, doi: 10.1109/ICDAR.2019.00252.
[16] C. C. Ng, A. K. B. Nazaruddin, Y. K. Lee, X. Wang, Y. Liu, C. S. Chan, L. Jin, Y. Sun, and L. Fan, "ICDAR 2021 competition on integrated circuit text spotting and aesthetic assessment," in *Proc. 16th Int. Conf. Document Anal. Recognit.*, 2021, pp. 663–677, doi: 10.1007/978-3-030-86337-1_44.
[17] S. M. Gang and J. J. Lee, "Coreset construction for character recognition of PCB components based on deep learning," *J. Korea Multimedia Soc.*, vol. 24, no. 3, pp. 382–395, 2021, doi: 10.9717/KMMS.2020.24.3.382.
[18] S. Gang, N. Fabrice, D. Chung, and J. Lee, "Character recognition of components mounted on printed circuit board using deep learning," *Sensors*, vol. 21, no. 9, p. 2921, Apr. 2021, doi: 10.3390/s21092921.
[19] T. H. Cho, "Recognition of characters printed on PCB components using deep neural networks," *J. Semicond. Display Technol.*, vol. 20, no. 3, pp. 6–10, 2021.
[20] Q. Wang, P. Li, L. Zhu, and Y. Niu, "1st place solution to ICDAR 2021 RRC-ICTEXT end-to-end text spotting and aesthetic assessment on integrated circuit," 2021, *arXiv:2104.03544*.
[21] C. Pramerdorfer and M. Kampel, "A dataset for computer-vision-based PCB analysis," in *Proc. 14th IAPR Int. Conf. Mach. Vis. Appl. (MVA)*, May 2015, pp. 378–381, doi: 10.1109/MVA.2015.7153209.
[22] C.-W. Kuo, J. D. Ashmore, D. Huggins, and Z. Kira, "Data-efficient graph embedding learning for PCB component detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 551–560, doi: 10.1109/WACV.2019.00064.
[23] H. Lu, D. Mehta, O. Paradis, N. Asadizanjani, M. Tehranipoor, and D. L. Woodard, "FICS-PCB: A multi-modal image dataset for automated printed circuit board visual inspection," *Cryptol. ePrint Arch.*, vol. 2020, Jul. 2020. [Online]. Available: https://eprint.iacr.org/2020/366
[24] N. Jessurun, O. P. Dizon-Paradis, J. Harrison, S. Ghosh, M. M. Tehranipoor, D. L. Woodard, and N. Asadizanjani, "FPIC: A novel semantic dataset for optical PCB assurance," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 19, no. 2, pp. 1–21, Apr. 2023, doi: 10.1145/3588032.
[25] F. Ulger, S. E. Yuksel, A. Yilmaz, and D. Gokcen, "Solder joint inspection on printed circuit boards: A survey and a dataset," *IEEE Trans. Instrum. Meas.*, vol. 72, 2023, Art. no. 2515121, doi: 10.1109/TIM.2023.3277935.
[26] A. Mantravadi, D. Makwana, S. C. T. R, S. Mittal, and R. Singhal, "Dilated involutional pyramid network (DInPNet): A novel model for printed circuit board (PCB) components classification," in *Proc. 24th Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2023, pp. 1–7, doi: 10.1109/ISQED57927.2023.10129388.
[27] L. Wu, C. Zhang, J. Liu, J. Han, J. Liu, E. Ding, and X. Bai, "Editing text in the wild," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 1500–1508, doi: 10.1145/3343031.3350929.
[28] Tesseract. *Tesseract Open Source OCR Engine*. Accessed: Mar. 10, 2024. [Online]. Available: https://github.com/tesseract-ocr/tesseract
[29] Z. Bien, S.-R. Oh, J. Won, B.-J. You, D. Han, and J. O. Kim, "Development of a well-structured industrial vision system," in *Proc. 16th Annu. Conf. IEEE Ind. Electron. Soc.*, 1990, pp. 501–506, doi: 10.1109/IECON.1990.149191.
[30] R. Nagarajan, S. Yaacob, P. Pandian, M. Karthigayan, S. H. Amin, and M. Khalid, "A real time marking inspection scheme for semiconductor industries," *Int. J. Adv. Manuf. Technol.*, vol. 34, nos. 9–10, pp. 926–932, Sep. 2007, doi: 10.1007/s00170-006-0669-1.
[31] C.-H. Lin, S.-H. Wang, and C.-J. Lin, "Using convolutional neural networks for character verification on integrated circuit components of printed circuit boards," *Appl. Intell.*, vol. 49, no. 11, pp. 4022–4032, Nov. 2019, doi: 10.1007/s10489-019-01486-5.
[32] C. Jiang, S. Chen, Z. Zhang, and R. Li, "Energy meter patch resistance and welding spot anomaly detection method based on machine vision," *J. Phys., Conf. Ser.*, vol. 2428, no. 1, Feb. 2023, Art. no. 012045, doi: 10.1088/1742-6596/2428/1/012045.
[33] R. Gomez, B. Shi, L. Gomez, L. Numann, A. Veit, J. Matas, S. Belongie, and D. Karatzas, "ICDAR2017 robust reading challenge on COCO-text," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 1435–1443, doi: 10.1109/ICDAR.2017.234.
[34] N. Nayef, Y. Patel, M. Busta, P. N. Chowdhury, D. Karatzas, W. Khlif, J. Matas, U. Pal, J.-C. Burie, C.-l. Liu, and J.-M. Ogier, "ICDAR2019 robust reading challenge on multi-lingual scene text detection and recognition—RRC-MLT-2019," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 1582–1587, doi: 10.1109/ICDAR.2019.00254.
[35] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," 2014, *arXiv:1406.2227*.
[36] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2315–2324, doi: 10.1109/CVPR.2016.254.
[37] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2014, pp. 512–528, doi: 10.1007/978-3-319-10593-2_34.
[38] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "TextBoxes: A fast text detector with a single deep neural network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, pp. 4161–4167, 2017, doi: 10.1609/aaai.v31i1.11196.
[39] F. Naiemi, V. Ghods, and H. Khalesi, "A novel pipeline framework for multi oriented scene text image detection and recognition," *Expert Syst. Appl.*, vol. 170, May 2021, Art. no. 114549, doi: 10.1016/j.eswa.2020.114549.
[40] M. Liao, B. Shi, and X. Bai, "TextBoxes++: A single-shot oriented scene text detector," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3676–3690, Aug. 2018, doi: 10.1109/TIP.2018.2825107.
[41] M. Busta, L. Neumann, and J. Matas, "Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2231, doi: 10.1109/ICCV.2017.242.
[42] H. Li, P. Wang, and C. Shen, "Towards end-to-end text spotting with convolutional recurrent neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5248–5256, doi: 10.1109/ICCV.2017.560.
[43] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai, "Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 71–88, doi: 10.1007/978-3-030-01264-9_5.
[44] T. He, Z. Tian, W. Huang, C. Shen, Y. Qiao, and C. Sun, "An end-to-end TextSpotter with explicit alignment and attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5020–5029, doi: 10.1109/CVPR.2018.00527.
[45] C. Bartz, H. Yang, and C. Meinel, "SEE: Towards semi-supervised end-to-end scene text recognition," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, vol. 32, no. 1, pp. 6674–6681, doi: 10.1609/aaai.v32i1.12242.
[46] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan, "FOTS: Fast oriented text spotting with a unified network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5676–5685, doi: 10.1109/CVPR.2018.00595.
[47] M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai, "Mask TextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 532–548, Feb. 2021, doi: 10.1109/TPAMI.2019.2937086.
[48] W. Feng, W. He, F. Yin, X.-Y. Zhang, and C.-L. Liu, "TextDragon: An end-to-end framework for arbitrary shaped text spotting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9075–9084, doi: 10.1109/ICCV.2019.00917.
[49] S. Qin, A. Bissaco, M. Raptis, Y. Fujii, and Y. Xiao, "Towards unconstrained end-to-end text spotting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4703–4713, doi: 10.1109/ICCV.2019.00480.

[50] H. Wang, P. Lu, H. Zhang, M. Yang, X. Bai, Y. Xu, M. He, Y. Wang, and W. Liu, "All you need is boundary: Toward arbitrary-shaped text spotting," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 7, pp. 12160–12167, doi: 10.1609/aaai.v34i07.6896.
[51] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "ABCNet: Real-time scene text spotting with adaptive Bezier-curve network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9806–9815, doi: 10.1109/CVPR42600.2020.00983.
[52] Y. Baek, S. Shin, J. Baek, S. Park, J. Lee, D. Nam, and H. Lee, "Character region attention for text spotting," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 12374, 2020, pp. 504–521, doi: 10.1007/978-3-030-58526-6_30.
[53] W. Wang, E. Xie, X. Li, X. Liu, D. Liang, Z. Yang, T. Lu, and C. Shen, "PAN++: Towards efficient and accurate end-to-end spotting of arbitrarily-shaped text," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5349–5367, Sep. 2022, doi: 10.1109/TPAMI.2021.3077555.
[54] S. Fang, Z. Mao, H. Xie, Y. Wang, C. Yan, and Y. Zhang, "ABINet++: Autonomous, bidirectional and iterative language modeling for scene text spotting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7123–7141, Jun. 2023, doi: 10.1109/TPAMI.2022.3223908.
[55] P. Wang, H. Li, and C. Shen, "Towards end-to-end text spotting in natural scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 7266–7281, Oct. 2022, doi: 10.1109/TPAMI.2021.3095916.
[56] Y. Kittenplon, I. Lavi, S. Fogel, Y. Bar, R. Manmatha, and P. Perona, "Towards weakly-supervised text spotting using a multi-task transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4594–4603, doi: 10.1109/CVPR52688.2022.00456.
[57] D. Peng, X. Wang, Y. Liu, J. Zhang, M. Huang, S. Lai, J. Li, S. Zhu, D. Lin, C. Shen, X. Bai, and L. Jin, "SPTS: Single-point text spotting," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 4272–4281, doi: 10.1145/3503161.3547942.
[58] DAVAR Lab. *OCR Toolbox From DAVAR-Lab*. GitHub. Accessed: Feb. 2, 2023. [Online]. Available: https://github.com/hikopensource/DAVAR-Lab-OCR
[59] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 565–571, doi: 10.1109/3DV.2016.79.
[60] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 6, pp. 567–585, Jun. 1989, doi: 10.1109/34.24792.
[61] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou, "Focusing attention: Towards accurate text recognition in natural images," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5086–5094, doi: 10.1109/ICCV.2017.543.
[62] N. Fabrice, S. Gang, and J. J. Lee, "Training data sets construction from large data set for PCB character recognition," *J. Multimedia Inf. Syst.*, vol. 6, no. 4, pp. 225–234, Dec. 2019, doi: 10.33851/jmis.2019.6.4.225.

WEI JIE BRIGITTE LIAU was born in Perak, Malaysia, in 1996. She received the B.Comp.Sc. degree (Hons.) from Universiti Sains Malaysia, Malaysia, in 2020, where she is currently pursuing the M.Sc. degree in computer science by research. Her research interests include intelligent systems, computer vision, and deep learning.

SHIEK CHI TAY received the B.Comp.Sc. degree (Hons.) from Universiti Sains Malaysia, Malaysia, in 2020, with a focus on multimedia computing, where she is currently pursuing the M.Sc. degree in computer science by research. Her research interests include machine vision and deep learning techniques.

AHMAD SUFRIL AZLAN MOHAMED received the B.I.T. degree (Hons.) from Multimedia University, Malaysia, the M.Sc. degree from The University of Manchester, U.K., and the Ph.D. degree from the University of Salford, U.K. He is currently an Associate Professor with the School of Computer Sciences, Universiti Sains Malaysia, Malaysia. His research interests include image processing, video tracking, facial recognition, and medical imaging.

MOHD NADHIR AB WAHAB (Member, IEEE) received the B.Eng. (Hons.) and M.Sc. degrees in mechatronics engineering from Universiti Malaysia Perlis, Malaysia, in 2010 and 2012, respectively, and the Ph.D. degree in robotics and automation systems from the University of Salford, U.K., in 2017. He is currently a Senior Lecturer with the School of Computer Sciences, Universiti Sains Malaysia, Malaysia. His research interests include mobile robotics, computer vision, machine learning, deep learning, artificial intelligence, optimization, navigation, and path planning.

LAY CHUAN LIM received the bachelor's degree in computer science and engineering from Monash University, in 2000. From 2001 to 2016, he was a Research and Development Engineer with Trek 2000, Stec, and Motorola, Malaysia. In 2017, he joined Western Digital in test engineering. He specializes in embedding programming, NAND storage devices, two-way radio, and computer bus interfaces. He utilizes various data analytics and machine learning techniques in the company's 4th industrial revolution and digital transformation. He is currently the inventor of six patents and three trade secrets. He received the recognition award for the "Global Lighthouse Network" when Western Digital, Batu Kawan, Penang, and he was awarded as the first "Light House" company in Asia by the World Economic Forum (WEF).

BENG KANG KHAW received the B.E. degree in electronic system engineering from Sheffield Hallam University, U.K., in 2000. From 2001 to 2004, he was a Research and Development Engineer with Renesas Semiconductor (Malaysia) Sdn. Bhd. Since 2005, he has been with Motorola Solutions Malaysia Sdn. Bhd. He is currently with Western Digital, Batu Kawan, Penang, as a Specialist in test engineering for solid-state drives. He worked on two-way radio firmware development.

MOHD HALIM MOHD NOOR received the B.Eng. degree (Hons.) from International Islamic University Malaysia, Malaysia, in 2004, the M.Sc. degree from Universiti Sains Malaysia, Malaysia, in 2009, and the Ph.D. degree in computer systems engineering from the University of Auckland, New Zealand. He is currently a Senior Lecturer with the School of Computer Sciences, Universiti Sains Malaysia. His research interests include machine learning, deep learning, computer vision, and pervasive computing.