Proceedings of the 2019 IEEE/ASME
International Conference on Advanced Intelligent Mechatronics
Hong Kong, China, July 8-12, 2019

Automatic Container Code Localization and Recognition via an Efficient Code Detector and Sequence Recognition*

Chenghao Li[1], Shuang Liu[1*], Qiaoyang Xia[1], Hui Wang[1], Haoyao Chen[2]

***Abstract*—The challenge of container code localization and recognition is that the code varies in size, position, and arrangements according to the container style and environmental changes. This paper proposes an automatic container code localization and recognition system via an efficient code detector and a sequence recognizer. The code detector locates text features on the backend of the container and extracts the code through segmentation, which increases the stability of code localization. The proposed recognition algorithm regards the code as three separate sequences and recognizes them through a recurrent-convolutional neural network to improve efficiency and accuracy. To verify the validity, a test dataset is established containing about 700 high-resolution images and over 2000 gray-scale code patches. The proposed code localization and recognition algorithm is verified on the dataset in experiments and achieves overall accuracy of 93.98% at about 0.1s per-frame.**

I. INTRODUCTION

In the container transportation process, the identification of containers is critical. According to the ISO-6346 standard, the code consists of three parts: 1) Four English letters indicating the owner and container type. 2) Six digits indicating the cabinet registration code. 3) One check digit. The check digit is used to verify whether the code is recognized correctly. Even though some other texts on the backend of the container represent specific characteristics, the container code is unique for each container.

The traditional method of manually recording container numbers cannot satisfy the massive flow of container transportation. The automatic identification system includes wireless sensor identification systems, radio frequency identification systems (RFID), and automatic computer vision recognition systems (ACCR). Compared with the other two automatic identification systems, the automatic computer vision recognition system has the characteristics of low cost, convenient installation and high feasibility. It generally divides into three parts: container code localization, code segmentation, and code recognition [1]. In the process of code localization, due to the relative position variance of the container during the transportation and the variety of the container type, the localization error of the container code is significant. Many other texts cover the backend of the container that possesses the best shooting conditions crows with other texts about specific information [2]. As a result, it is difficult to distinguish them from the code. The code segmentation needs to segment the whole container code area into 11 patches [1] [3], which has high requirements on the clarity and noise conditions of shooting. Highly robust segmentation algorithms often lead to a time-consuming problem. Code recognition tends to apply a neural network or other recognition algorithms to identify the segmented patches individually, which is inefficient [4] [15].

Wu et al. [1] proposed an overall computer-vision container identification system that includes container code localization, code segmentation, and code recognition. They use the text line analysis to locate the code part, and then divide them according to the vertical projection of the image binarization. This paper suppresses the reflective effect caused by the irregular shape of the container during the segmentation stage. Kumano et al. [2] proposed a dynamic segmentation method based on the dynamic design for the backend of the container. This method calculates the optimal segmentation method based on character similarity and space. The algorithm adjusts according to the illumination condition through the light intensity sensor, which increases the robustness to the low image quality. Youngwoo et al. [3] proposed a container code identification system based on multi-view method. This system locates and OCR identifies the code distributed on the five surfaces of the container from five different angles and obtains the final recognition result by multi-sequence alignment (MSA) [4], which can improve the system's Overall recognition rate. Unfortunately, these identification systems are designed for specific scenarios, and the ability to adapt to diverse environments needs to be improved.

Goccia et al. [5] proposed a container code recognition method based on the post-fusion processing method. In each feature space of the input sample, the one with the minimum calculated loss in the training set is taken as the recognition

* This work was sponsored by the National Natural Science Foundation of China (No. 51405236), the Shanghai Pujiang Program (No. 16PJ1402300), and the Shenzhen Science and Innovation Committee (No. JCYJ20160427183958817)

1 Chenghao Li, Shuang Liu, Qiaoyang Xia, Hui Wang are with the School of Mechanical and Power Engineering, East China University of Science and Technology, Shanghai 200237, P.R China (e-mail: shuangliu@ecust.edu.cn).

2 Haoyao Chen is affiliated with the School of Mechanical Engineering and Automation in Harbin Institute of Technology Shenzhen, and the State Key Laboratory of Robotics and System, China (a (hychen5@hit.edu.cn).).

*corresponding author: Shuang Liu

result. The calculation model parameters are optimized by the gradient descent method. Igual et al. [6] proposed a set of morphological processing algorithms for container code image preprocessing, and the processed images were identified using the nearest neighbor classifier. However, these algorithms do not solve the problem of accurate container number positioning. Kim et al. [7] proposed a fuzzy logic-based noise cancellation method and a self-organizing supervised learning algorithm based on ART2 for container code recognition. The accuracy of letters and numbers reported separately is 98.5% and 97.4%, respectively.

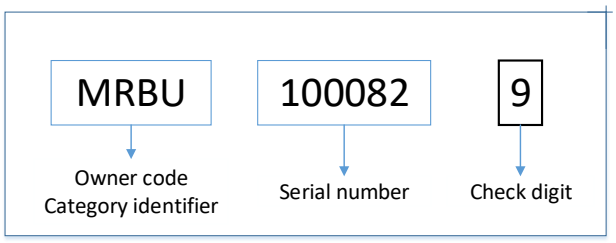

Figure 1. Container Code Components: (a) Owner code & Category identifier (b) Serial number (c) Check digit

The deep convolutional neural networks [16] [24] [25] show great performance on the feature extraction of images. As a relevant research field in computer vision, the detection and identification of scene texts have been affected. Almost all of the state-of-art text detection methods are built on the deep learning model due to its ability to extract features. This technology gives the algorithm the ability to detect and recognize text with different lengths and fuzzy occlusion, which has essential application value for the container code localization and recognition.

In terms of text localization, influenced by R-CNN [8], Faster R-CNN [9], SSD [10], and STN [11] deep convolutional neural networks, TextBoxes [12] uses the framework of SSD networks to accommodate different orientations of text and the aspect ratio. At the corner of each anchor frame, it replaces the default box with a quad, which allows the system to capture text lines more closely and reduce noise. EAST [13] proposes a simple and efficient full convolutional neural network model that can perform high-speed and accurate text detection in natural scenes. The model directly predicts quadrilateral shaped words or text in either direction of the complete image, eliminating the unnecessary steps of using a single neural network. In terms of text recognition, CRNN [14] proposed an image-based sequence recognition network to process images with arbitrary lengths, involving no character segmentation or horizontal scale normalization. Other algorithms like [26] [27] [28] convert word images into sequential HOG features for RNN or detect individual characters to recognize them separately.

Unlike the scene text recognition, the container number follows a specific pattern, with relatively fixed positions on the container according to the type and color of the container. Ankit et al. [15] proposed an end-to-end neural network model to identify container numbers. The model uses the connected area algorithm to locate the container code and then combines the space transformation network (STN) to recognize it. The training data of the model is obtained by applying affine transformations on 35 image samples, and the test set only consists of 19 samples.

To improve the efficiency of container number identification, this paper proposes a high-robust automatic container code localization and recognition system via an efficient code detector and sequence recognition in response to the variance of container types and shooting conditions with about 0.1s per-frame and 93.98% overall accuracy. The main contributions of this paper are as follows:

- A full convolutional neural network along with a container code segmentation to localize the code in 720p images. It detects the localization of the three parts of the code with the robustness of position and shooting condition variance.
- A convolutional recurrent neural network to simplify the combination of the code patch extraction and identification process. It directly identifies the three parts of the code which significantly saves the time required for the code segmentation.
- The container number identification system is easy to install. Only a camera and a processor with a graphics card are required, which dramatically reduces the system cost.

The rest of the paper is organized as follows: Section II provides the detail of the proposed efficient code detector. Section III provides the detail of the proposed sequence recognition network. Section IV presents the details of the experiments and results. Section V outlines the conclusions and future works of the proposed container code localization and recognition algorithm.

II. Efficient Code Detector

The primary purpose of the container number identification system is to identify the container number following the ISO6346 coding mode, as shown in Figure 1. These three parts are:

- Code 1: 1: 4 letters (owner code and type identification number)
- Code 2: 6 numbers (container serial number)
- Code 3: 1 bounded number (check code)

Each box number block is distributed horizontally or vertically as a whole at the upper or upper right end of the container rear end.

A. Pipeline

To decide whether a container to detect shows up in the surveillance video and efficiently localizing its code position, this paper merges multi-dimensional feature extracted by ResNet50 [16] following the methodology in EAST [13] which uses PVANet [24]. This detector localizes the text in the overall picture, and then feed the result into a code segmentation to extract the three parts of the container code. The specific pipeline is shown in Figure 2.

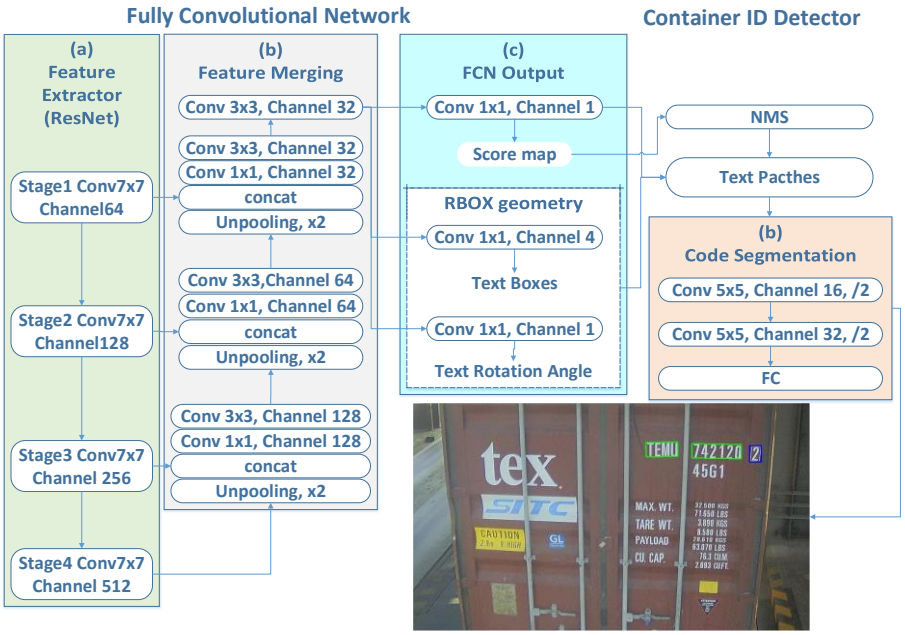

Figure 2. Proposed pipeline for Container Code Localization comprises of 4 major blocks – (a) Feature extractor (b) Feature Merging (c) FCN output and (d) Code Segmentation. The overall output of this code detector is shown as a picture inserted in this figure whit bounding boxes illustrating the code position

The container code localization network pipeline proposed in this paper follows the design of EAST [13] and ResNet [16]. Firstly, the original image is input into the feature extraction layer from Resnet50. The multi-layer convolutional neural network extracts pixel-level features of different dimensions. Those features concatenate together after unpooling followed by a bottleneck layer to reduce the number of channels. Details of the layers in the FCN are shown in Figure 2.

In this paper, the number of convolutional layer channels in the feature fusion layer is controlled to a small number, which can save the computational overhead of the network and make the network more efficient. The final output layer contains three 1x1convolution layers that map 32 convolution feature channels into a single-channel score map and a 5-channel geometry output. The score map value of the FCN output is in the interval *[0, 1]*, and the geometric position information output represents the pixel level position of the text. The value of the score map represents the confidence of the geometric result.

This paper applies the rotating text box (RBOX) in EAST [13] as the text localization output, represented by a 4-channel axis-aligned bounding box (AABB) $\boldsymbol{T}$ and a 1-channel rotation angle θ:

$$\boldsymbol{G} = \{\theta, \boldsymbol{T}\} = \{\theta, \{\tau_i | i \in \{1,2,3,4\}\}\} \quad (1)$$

Where T indicates the position of the upper left corner and the lower right corner of the text box. Figure 3 shows the generation method of the specific bounding box and rotation angle.

After the non-maximum value suppression (NMS), the FCN output score map and geometry information can locate all the texts at the backend of the container and input them into a two-layer convolutional neural network to segment the container code out. The detection result is shown in the backend image of the container in Figure 2.

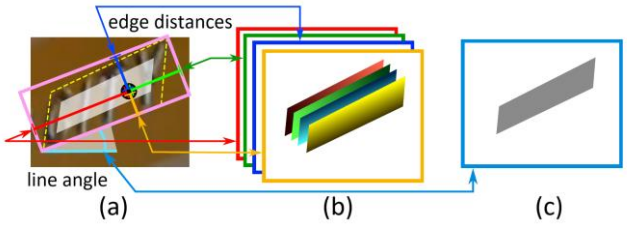

Figure 3. Generation of bounding box: (a) RBOX geometry map generation; (b) 4channels for distances of each pixel to rectangle boundaries; (c) Rotation angle.

B. *Loss Function*

The loss function of the box number positioning network composes of two parts, and the loss function of the FCN network can be expressed by the following formula:

$$L_1 = L_s + \lambda_g L_g \quad (2)$$

Where L_s and L_g represent the loss function of the score map and the geometry output RBOX, respectively. λ_g is the balance parameter between the two losses, which is set to λ_g=1 in this paper. The loss function L_2 extracted by the box number is a cross entropy function:

$$L_2(\mathrm{Y}, Y^*) = -Y^* logY + (1 - Y^*)\log(1 - Y) \quad (3)$$

Where Y is the prediction whether patches are container codes, and Y^* is the ground truth.

The score graph loss function in this paper draws on the class-balanced cross-entropy loss function in [18] as follow:

$$L_s = -\beta Y^* logY - (1 - \beta)(1 - Y^*)\log(1 - Y) \quad (4)$$

Where Y is the predicted score map, Y^* is the ground truth, and the parameter β is the balance parameter between the positive and negative samples which is formulated as:

$$\beta = 1 - \frac{\sum_{y^* \in Y^*} y^*}{|Y^*|} \quad (5)$$

Since the backend of a container has external text with scale variance, the regression with L1 or L2 loss function will bias the loss value to the larger and longer text area, which may cause instability in detection. Therefore, this paper uses the IoU loss function for the RBOX bounding box in EAST [13] and [17] to improve the adaptability of the loss function to scale variant text. The IoU loss function is formulated as:

$$L_{IoU} = -logIoU(\boldsymbol{T}, \boldsymbol{T}^*) = -log\frac{|\boldsymbol{T} \cap \boldsymbol{T}^*|}{|\boldsymbol{T} \cup \boldsymbol{T}^*|} \quad (6)$$

Where $\boldsymbol{T}$ represents the bounding box (RBOX) of the prediction and $\boldsymbol{T}^*$ represents the corresponding ground truth. The rotation loss is formulated as:

$$L_\theta(\theta, \theta^*) = 1 - \cos(\theta - \theta^*) \quad (7)$$

Where θ represents the predicted rotation angle and θ^* represents the corresponding ground truth. The overall geometry loss is formulated as:

$$L_{rbox} = L_{IoU} + \lambda_\theta L_\theta \quad (8)$$

Where λ_θ is the parameter to balance the loss between L_{IoU} and L_θ, which is set to 8 in this paper.

C. Training

Since the non-maximum suppression part of the box number location algorithm cannot apply backpropagation, this paper trains two parts of the detector separately. For the full convolutional network model, we use ADAM [22] for end-to-end training, with ICDAR 2015 [23] for initialization, and 2000 images of the container backend 720p for ultimate training. The training procedure is validated by 200-sample validation set. The learning rate of ADAM starts from 1e-3, decays to one-tenth for every ten epochs, and the lower limit is 1e-5. The network keeps training until its performance stops improving. For the code segmentation network, a training set of 8000 code patches is applied with ratio 1:3 for positive and negative samples. This training procedure also uses the ADAM optimizer with learning rate 1e-3 for 800 epochs.

III. Code Sequence Recognition

The detector in Section III outputs the localization of the container code and divides it into three parts. Prior works segment the code into single characters, ignoring the pattern container codes follows. This paper follows the pipeline of CRNN[14] to propose a method to recognize the container code in patches with variant lengths, eliminating the time for character segmentation.

A. Pipeline

The pipeline of the proposed network consists of four parts, including a convolutional layer, a sequence extraction layer, a loop layer and a text transcription layer from top to bottom, as described in Fig.3. The detail of the convolutional layer and bidirectional lstm [19] are highlighted, and output of the transcription layer is the recognition result of the network.

The convolutional layer includes multiple convolution layers and pooling layers which follows ResNet [16] for extracting convolutional feature maps from the input image. All input images are scaled to a uniform height, and high-dimensional features are extracted through the convolutional layer and fed into the sequence extractor. It maps the feature tensor to a series of feature vectors. Each one of the feature sequences is generated from left to right in columns on the convolution feature map (i.e. the i-th feature vector corresponds to the i-th column of the feature map, and the width of each column is fixed to a single pixel). Each column of the feature map corresponds to a rectangular area of the original image, and such a rectangular area is in the same order as the corresponding column on the feature map from left to right. The feature sequence extractor converts the deep features into sequences, making the network adaptable to sequence targets with variable length.

The feature sequence then is fed into a recurrent layer for recognition. The recurrent layer is composed of two layers of bidirectional long-short-term memory loop neural network (Bidirectional LSTM) [19]. A higher level of abstraction shows significant performance improvements in speech recognition tasks.

Text transcription is the process of converting the predicted output of the network into a sequence of texts (i.e. it finds the corresponding text sequence with the highest probability in the predicted output). The proposed pipeline applies the conditional probability proposed in Time Series Classification (CTC) [20]. The conditional probability of network output $\boldsymbol{y} = y_1, \ldots, y_t$ is predicted frame by frame for text sequence l. The network training process only needs the image sample and its corresponding ground truth, which significantly saves labor costs.

B. Loss Function

The training set for the recognizer is composed of code part patches and corresponding text ground truth. The loss function is defined as the negative maximum likelihood estimation function of the text condition probability, which is formulated as:

$$L = -\sum_{l_i \in l^*} \log p(l_i | y_i) \quad (9)$$

Where l^* denotes the sequence ground truth, $y_i \in \boldsymbol{y}$ is the predicted confidence sequence output by the recurrent layer. p is a text conditional probability function described as follow: The recurrent layer output sequence is $\boldsymbol{y} = y_1, \ldots, y_t$, where t is the length of the sequence. Each element represents a probability distribution over$L' = L \cup " - "$, where L' contains all the labels L (in this case 26 alphabets and 9 numbers) and a mark $-$ representing the blank in the recognition. A transcription mapping function B is defined on the sequence $\pi \in L'^t$, which π is obtained by removing blank items and duplicates (e.g. B maps "-T-E-MMU" to "TEMU"). The text conditional probability function p is formulated as:

$$p(l|y) = \sum_{\pi:B(\pi)=l} p(\pi|y) \quad (10)$$

Where the probability of π is defined as $p(\pi|y) = \prod_{i=1}^{t} y_{\pi_i}^i$, $y_{\pi_i}^i$ is the probability that i-th subsequence has the label π_i. The overall output of the recognizer is formulated as:

$$l \approx B(argmax_\pi p(\pi|y)) \quad (11)$$

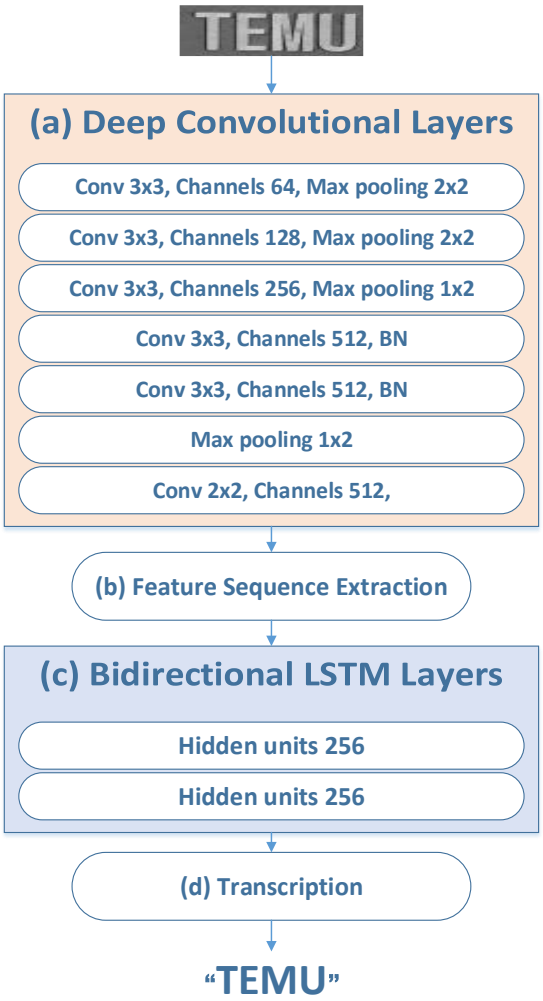

Figure 4. Sequence Recognition Network Pipeline: (a) ResNet Style Deep Convolutional Layers; (b) Feature Sequence Extraction with the ability of backpropogation; (c) Recurrent neural network with two layers of bidirectional lstm; (d) Transcription layer to transcript the raw prediction to output text

C. Training

The container code recognition network is trained on the synthetic dataset released by Jaderberg [21] for initial training, and a training set of 5000 code patches with ground truth for ultimate training. The optimizer is ADAM with a learning rate of 1e-3, which decays one-tenth for every ten epochs, and the lower limit is 1e-5. The training process is set to 800 epoch.

TABLE I. ACCURACY TEST RESULTS

Accuracy Test	Test results		
	Localization Rate	Recognition Rate	Overall Rate
Tesseract	77.22%	80.89%	62.46%
Ours	100%	93.98%	93.98%

IV. EXPERIMENT

The experimental platform is a workstation equipped with an i9-7900X processor and an Nvidia GTX 1080Ti with 11G memory. The system is implemented in Python with PyTorch. The proposed automatic container id localization and recognition system perform on 720p images for about 0.0897 seconds per frame. After multi-process optimization, it can achieve 12FPS in video processing, which has confident real-time performance. The multi-process system composes of 3 parallel processes:

- The main process: Grab a frame from the video stream to share with other processes and display detection results.
- The localization process: Localize container code and send the location information to the recognition process.
- The recognition process: Recognize the code patches extracted from the frame with the location information and send the result to the main process.

This system takes 2 frames at the same time, the later frame is in the localization process and the earlier one is in the recognition process or in the main process to display.

Figure 5. Test dataset – (a) 720p image samples of container backend shot in daylight and night (b) gray-scale code patches (c) check digit patches

A. Test Dataset

All the experiments of prior works apply different datasets to test. Ankit et al. [15] released their test set, but it was only 19 samples. The proposed automatic container-code localization method detects the backend image of the container. To test its performance properly, we create a test set with 698 samples under different situations like during days or nights, 1000 gray-scale container code patches, and 1000 check digit patches for independent check digit recognition. Fig.5 shows the proposed test dataset. The 720p images of container backend are shot once containers appear on the transportation channels. As shown in Fig.5 (c), a rectangle box bounds the check digit part of container code, which may cause unstable performance. We build a 3-layer convolutional neural network to detect the check digit separately.

B. Results

The experiment results of the proposed automatic container id localization and recognition system fall into three parts: code localization accuracy, code recognition accuracy, and overall accuracy. For benchmarking, an OCR-based

system with the tesseract-ocr [1] recognition engine is implemented to test on the dataset. Table I shows accuracy results between the proposed system and the tesseract system.

The accuracy of code segmentation is 99.80% on the test set with added random noise. The code overall accuracy of the proposed method is 93.98% and the errors are miss recognition of several code characters. Figure 6 shows the recognition accuracy of alphabets, numbers and check digits that construct the container code. The recognition of "6", "G", "D", "8" and check digit with bounding box is relatively low. This is caused by character similarity, staining, blurring, and deficiency in the letter patches.

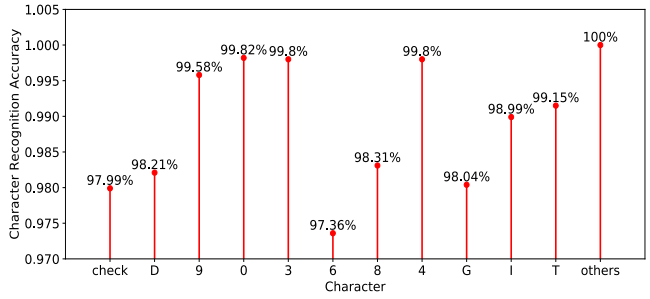

Figure 6. Character Recognition Accuracy Result

The time consumption of the proposed method is about 0.1s per-frame on 720p container backend images overall. The Localization cost 0.08835s on average and the Recognition cost 0.03926s on average, which is faster than the system in [15]. After multiprocessing optimization, a system implemented with the proposed method can reach 12 FPS on a 720p video stream.

V. Conclusion

This paper proposes an automatic container code localization and recognition algorithm for images of the container backend. The proposed algorithm includes an efficient code detector with a fully convolutional neural network and a code segmentation network and sequence recognizer with a convolutional recurrent neural network. The code detector possess the ability to locate the container code correctly. The sequence recognizer outputs the code text accurately with high speed. The system using the proposed algorithm is robust to environmental variation and is capable of performing at about 0.1s per-frame. Future work of the automatic container code localization and recognition attempts to locate and recognize the box number directly through a multi-output neural network, improve the number and reliability of the container dataset, and localize and recognize the container code on other surfaces of containers.

References

[1] Wei, Wu, et al. "An automated vision system for container-code recognition." Expert Systems with Applications 39.3(2012):2842-2855.

[2] Kumano, Shintaro, et al. "Development of container identification mark recognition system." Electronics & Communications in Japan 87.12(2010):38-50.

[3] Youngwoo Y, et al. "Automatic Container Code Recognition from Multiple Views." ETRI Journel, 2016, 38(4):767-775

[4] Döring Andreas , et al. "SeqAn An efficient, generic C++ library for sequence analysis." BMC Bioinformatics9.1(2008):11-0.

[5] M. Goccia, M. Bruzzo, et al. "Recognition of container code characters through gray-levelfeature extraction and gradient-based classifier optimization". Seventh International Conference on Document Analysis and Recognition, IEEE Computer Society, 2003.

[6] Igual, Ismael Salvador, G. A. Garcia, and Jiménez, Alberto Pérez. "Preprocessing and recognition of characters in container codes." 16thInternational Conference on Pattern Recognition IEEE Computer Society, 2002.

[7] Kim, Kwang Baek, S. Kim, and Y. W. Woo. "An Intelligent System for Container Image Recognition using ART2-based Self-Organizing Supervised Learning Algorithm." (2006).

[8] Girshick, Ross, et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) IEEE Computer Society, 2014.

[9] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." IEEE Transactions on Pattern Analysis & Machine Intelligence 39.6(2017):1137-1149.

[10] Liu, Wei, et al. "SSD: Single Shot MultiBox Detector." European Conference on Computer Vision 2016.

[11] Jaderberg, Max, et al. "Spatial Transformer Networks." (2015).

[12] Liao, Minghui, et al. "TextBoxes: A Fast Text Detector with a Single Deep Neural Network." (2016).

[13] Zhou, Xinyu, et al. "EAST: An Efficient and Accurate Scene Text Detector." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

[14] Shi, Baoguang, X. Bai, and C. Yao. "An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition." IEEE Transactions on Pattern Analysis & Machine Intelligence 39.11(2015):2298-2304.

[15] Verma, Ankit, et al. "Automatic Container Code Recognition via Spatial Transformer Networks and Connected Component Region Proposals." International Conference of Machine Learning Applications IEEE, 2016.

[16] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." (2015).

[17] Yu, Jiahui, et al. "UnitBox: An Advanced Object Detection Network." Acm on Multimedia Conference ACM, 2016.

[18] Xie, Saining, and Z. Tu. "Holistically-Nested Edge Detection." International Journal of Computer Vision 125.1-3(2017):3-18.

[19] Graves, Alex, A. R. Mohamed, and G. Hinton. "Speech Recognition with Deep Recurrent Neural Networks." (2013).

[20] Graves, Alex, Santiago Fernández, and F. Gomez. "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks." International Conference on Machine Learning ACM, 2006.

[21] Jaderberg, Max, et al. "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition." Eprint Arxiv(2014).

[22] Kingma, Diederik P., and J. Ba. "Adam: A Method for Stochastic Optimization." Computer Science (2014).

[23] Karatzas, Dimosthenis, et al. "ICDAR 2015 competition on Robust Reading." International Conference on Document Analysis & Recognition2015.

[24] Kim, Kye Hyeon, et al. "PVANET: Deep but Lightweight Neural Networks for Real-time Object Detection." (2016).

[25] Simonyan, Karen, and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." Computer Science (2014).

[26] Bissacco, Alessandro, et al. "PhotoOCR: Reading Text in Uncontrolled Conditions." 2013 IEEE International Conference on Computer Vision (ICCV) IEEE, 2013.

[27] Wang, Kai, B. Babenko, and S. Belongie. "End-to-end scene text recognition." 2011 International Conference on Computer Vision IEEE, 2012.

[28] Su, Bolan, and S. Lu. "Accurate Scene Text Recognition Based on Recurrent Neural Network." (2014).

[1] https://github.com/madmaze/pytesseract