Automatic Container Code Recognition Using MultiDeep Pipeline

Duy Nguyen, Duc Nguyen, Thong Nguyen, Khoi Ngo, Hung Cao, Thinh Vuong, and Tho Quan(✉)

Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
qttho@hcmut.edu.vn

Abstract. Identification of license plates on intermodal containers (or containers) while entering and departing from the yard provides a wide range of practical benefits, such as organizing automatic opening of the rising arm barrier at the entrance and exit to and from the site. In addition, automatic container code recognition can also assist in thwarting the entrance of unauthorized vehicles into the territory. With the recent development of AI, this process is preferably automatic. However, the poor quality of images obtained from surveillance cameras might have detrimental effects on AI models. To deal with this problem, we present a pipeline dubbed as MultiDeep system, which combines several state-of-the-art deep learning models for character recognition and computer vision processes to solve problems of real camera data. We have also compared our results with other pipeline models on real data and accomplished fairly positive results. In this paper, without further references, we will only consider intermodal containers when referring to them as containers.

Keywords: Container code · Optical Character Recognition · MultiDeep pipeline

1 Introduction

The term *Industry 4.0* originates from major changes in industrial history, which is considered to be the fourth largest technological revolution. Digitization introduced by Industry 4.0 creates a vision in which an integrated process of cross-technology, technologies and systems would unite everything - production, services, logistics, personnel. And resource planning. Hence comes the concept of the digital factory, which is essentially based on four principles: technical support, automated decisions, information transparency and complete network connectivity. Thus, Industry 4.0 also requires digital identification of workpieces, tools, containers, machines and equipment. In this paper, we focus on container code identification.

M. Hernes et al. (Eds.): ICCCI 2020, CCIS 1287, pp. 139–153, 2020.
https://doi.org/10.1007/978-3-030-63119-2_12

As the manufacturing and shipping industry is growing, and also introducing some new challenges to container management, which is currently handled manually in many sites, without standards and without the ability to integrate into sophisticated computing systems. Containers sometimes are not placed properly and therefore get lost, adversely having detrimental impact on the ability to transport materials and goods. In addition to the significant lack of efficiency across the entire container management system, there is a lack of transparency in the transportation of containers within the system. As a result, the cost of transportation is boosted or management is not effective.

Currently, many logistics enterprises have applied *Artificial Intelligence* techniques in managing containers in ports and warehouses. Especially, since the optimization of management and transportation at the port is still a problem, an abundance of corporations are interested in. Much attention in this field is drawn to the problem of *container code recognition*. In practice, each container will include a unique *identification code* of ISO standards. ISO6346[1] is the international container standard used to denote the identity of that container, which is divided into 2 parts, as illustrated in Fig. 1. The first is the container identifier information including the owner code, the container list (identifier), the serial number and the check digit. The rest is the shipping size code for that container.

Businesses have set up multiple cameras at ports, on control axes to monitor the status of containers based on the above code, and a large number of machine learning algorithms have been developed for the *Optical Character Recognition (OCR)* problem, which researchers have an intention of applying them for container code recognition problem. However, when putting into use in reality, due to the harsh condition of the vast terrain, the arrangement of surveillance cameras to acquire comprehensive coverage of vehicle details faces difficulties and is intractable. Therefore, the quality of the resulting camera image is poorly affected by practical factors called *hindering*, which leads to the problems encountered identification processes, as illustrated in Fig. 2.

There are a majority of machine learning models, especially the recently developed *deep learning* ones, which have been introduced for *Optical Character Recognition (OCR)* problem. However, the practical difficulties mentioned above hinder those academic models from achieving high accuracy when applied to real data. To tackle this, we propose a pipeline architecture, named *MultiDeep*.

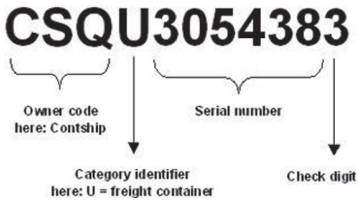

Fig. 1. Container code of the ISO6346 standard

[1] https://en.wikipedia.org/wiki/ISO_6346.

Fig. 2. Hindering factor examples (a) Different formats (horizontal/vertical letters, 2/3 lines) (b) Color of background (dark/bright), color of letters (white/black) and letters in a box (c) Inclined letters due to the rough surface (d) Different camera angles (e) Different brightness (day/night time, artificial lighting) (f) Blurry, noisy images (g) Letters' quality (rusted, deformed, discolored) (Color figure online)

This architecture takes advantage of various deep learning models and computer vision techniques to address those specific hindering factors. We have made a great deal of comparison relating to the performance of our approach with other baseline models and our method was proven to obtain encouraging initial results.

2 Related Work

Nowadays, there have been a wide range of studies related to character recognition, such as recognition in text, handwriting and especially, text boxes in reality, in which bounding boxes can be strongly affected by angles, contrast and last but not least, elasticity. Almost all models accomplish its power by aggregating the deepness, which means that by combining a bunch of layers, in particular convolutional filters, with interplayed max-pooling ones.

2.1 Scene Text Detection

2.1.1 Character Region Awareness for Text Detection

One of the most promising approaches is affinity measurement, along with region detection for every character. Particularly, the most prevalent model with this idea is [1]. As a deep learning model, CRAFT is devised from VGG-16 [2], co-operated with batch normalization. Utilizing the advantages of deep-learning and affinity assessment method allows CRAFT to deal with text regions which are escalated due to inclining or curvature, since it has realized relations among characters. Notwithstanding, as for container number images, CRAFT might have difficulties recognizing those relationships. Not only are number plates displayed vertically, in other words, each row comprises at most 1 or 2 numbers but also speeds of trucks when entering harbors might have important impacts on sensitivity of cameras. Additionally, this model is still far from approaching plates in which digits are damaged due to decaying or paint removal.

2.1.2 Efficient and Accuracy Scene Text Detector

Zhou et al. provides a time-saving method but still attains high efficiency. In fact, by putting only convolutional layers (FCN) into use, not only does EAST [3] remove intermediate steps, sometimes sorely unnecessary, for instance candidate proposal, text formation and text partition but also aggregate its time to learn a bunch of features, and propose boxes with a high likelihood of consisting of letters. The only weakness of EAST lies in text proposal phase, since it is only able to obtain rectangles or more generally, quadrangles. This might provide us with practical difficulties, yet some cameras have the capability of taking convex images with regards to wind direction and letter decaying processes in the past.

2.2 Optical Character Recognition

2.2.1 Spatial Attention Residue Network

This method has proved its benefits in assisting recognition models in avoiding noises made by image qualities. In spite of focusing on improving image input, STAR-Net [4] still includes a recognition component in its architecture in order to prove its effectiveness. Beside its strength of increasing image qualities, STAR-Net also has the ability to learn dependencies of characters as in CRAFT [1]. Despite that, it differentiates from CRAFT in the point that instead of utilizing affinity, STAR-Net seeks to learn it implicitly through LSTM cells [5]. Nonetheless, though promising, STAR-Net faces similar obstacles of CRAFT. It shows poor results when being tested with vertical container number plates. In addition, images with a wide range of contrast might cause STAR-Net troubles, especially images taken at late night.

2.2.2 Attentional Scene Text Recognizer with Flexible Rectification

ASTER's [6] core idea is at the whole analogous to STAR-Net [4]. Its main goal is to adjust directions of images after traversing through the network. Despite the same goal, ASTER's architecture is separate from STAR-Net. It applies Thin-Plate-Spline transformation and then recurrent architecture to achieve eventual output. One recognizable advantage is that modules are all differentiable, which leads to the fact that in order to train rectification network, there is no need to carry out human labelling but take advantage of back-propagated results of recognition component. However, the other side of the coin is that the problem of disqualified images remains unsolved, for it is still only concerned with tackling optical situations.

2.3 End-to-End Text Extraction from Image

2.3.1 Tesseract

Tesseract [7] is an optical character recognition engine for various operating systems. It includes many steps for preprocessing image and postprocessing result after feeding image into OCR engine. From its recent version, Tesseract adds LSTM as its primary OCR engine. Due to large dataset trained, it can score a very good result on clear document-like images. But in our experiment, Tesseract can not detect container code area because of the image quality and the ratio between text area and full image. Therefore, for a fair comparison, we only use it in text recognition step.

2.3.2 Feature-Based Local Intensity Gradient (LIG) and Adaptive Multi-threshold Methods

The combination of LIG and adaptive multi-threshold [8] gives out very fast results with minimum hardware requirements because it only relies on basic image processing. And as a result, the performance is only somewhat acceptable on clear images. Therefore, applying this method in real-life problems will cause issues.

2.3.3 Segmentation-Based and HMM-Based Method

This is a method that does not depend on machine learning but rather on probability and image processing [9]. Same as LIG, the model works fast and efficiently on datasets with horizontal clear images. However, the model accuracy is greatly reduced on blurry or tilted images. The problem of the model lies in the segmentation step on each code character, where it uses a high pass filter and scan each line of pixel from top to bottom. Thus, this model is ineffective in real-life problems.

2.3.4 Text-Line Region Location Algorithm Combined with Isolate Characters and Character Recognition

This method [10] combines image processing and machine learning. It scans each pixel from top to bottom to find lines, then uses image processing to isolate each character, and finally feeds each character into an SVM to get the result. However, the paper shows that the model only trained on one type of container image: horizontal letters on the same background. When applied with our real problem, the accuracy will be reduced on images with inclined letters, multicolor background, or vertical letters.

2.3.5 Spatial Transformer Networks and Connected Component Region Proposals

This is a newly introduced model with a high performance provided by the author [11]. The dataset used in the training are images of one or two sides of a container, with a lot of noises. However, when looking at the datasets, their images are of high resolution, and only daylight images are used to train the model. Thus, the model will definitely less effective on our dataset with blurry and nighttime images.

2.4 Summary

To conclude, almost all of the prominent models maintain their own advantages and disadvantages. Nevertheless, they are yet far from tackling a wide variety of situations which people can encounter in real world, especially in industry, in which most of the time images include inclining directions and optical obstacles. As a consequence, we propose the *MultiDeep* pipeline to address this.

3 The MultiDeep Pipeline

As stated in Related Works section, there have been a wide variety of studies related to text and digit recognition. Almost all of these models have been developed based on techniques of RNN ones. As a consequence, they will achieve efficiency when texts are demonstrated in a specified order (from left to right and line by line). Notwithstanding, in practice, container code can be demonstrated on 2, 3 lines or even a vertical one with multiple directions and optical conditions.

Consequently, applying a single model for both training and detection causes a wide range of difficulties. Moreover, because of the variety of container images, concentrating on only one model makes it hard to learn all of the features. As a result, we propose MultiDeep model which combines various Machine Learning models, computer vision techniques and some heuristics and algorithms with a view to accomplishing the most promising outcome. The main idea is that MultiDeep divides the container code recognition through two phases.

- *Phase 1:* detect regions which comprise texts.
- *Phase 2:* determine texts in those regions

In both phases, we utilize a diversity of heuristics to tackle practical obstacles. The first obstacle is that beside container codes, on container surfaces are there other kinds of text regions. The second one is that there might be errors in misclassifying texts which are caused by local recognition. Those heuristics are included in post-processing components in Phase 1 and Phase 2. In addition, with a view to attaining high accuracy in recognition, we utilize some computer vision approaches to preprocess images right before Phase 1 and Phase 2. As a result, we can handle problems such as container code images can be both horizontal and vertical, colorful, peculiar optical conditions, oversized and vacuous boxes. Details of the MultiDeep pipeline has been represented in Fig. 3, includes 6 processes. Regarding to our discussed 2 phases, Process 1 and Process 2 are conducted for environment preparation and processing. Phase 1 includes Process 3 and Process 4 and Phase 2 is involved in Process 5 and Process 6.

3.1 Process 1: Model Initiation

Two deep learning models, ASTER and CRAFT, are loaded for next processes.

3.2 Process 2: Preprocessing

The original image will be preprocessed in order to improve accuracy in the future steps. The main task includes the following ones.

1. *Increasing brightness:* In this task, we modify the value of all pixels by a scale constant α and additive one β (Eq. 1). By applying appropriate values to all of the image pixel values, the image obviously becomes brighter. The values of α and β are picked in trial-and-error method. In our dataset, $\alpha = 1.5$ and $beta = -10$ yields the best results.

$$P'(i,j) = \alpha P(i,j) + \beta(\alpha, \beta > 0) \tag{1}$$

2. *Scaling:* It has a connection with the resizing process of a digital image, comprised of Raster ImageScaling and Vector Image Scaling. Raster graphic is a bitmap image made of individual pixels. Pixels are mapped to a new grid, which may possess different dimensions from the original matrix. In

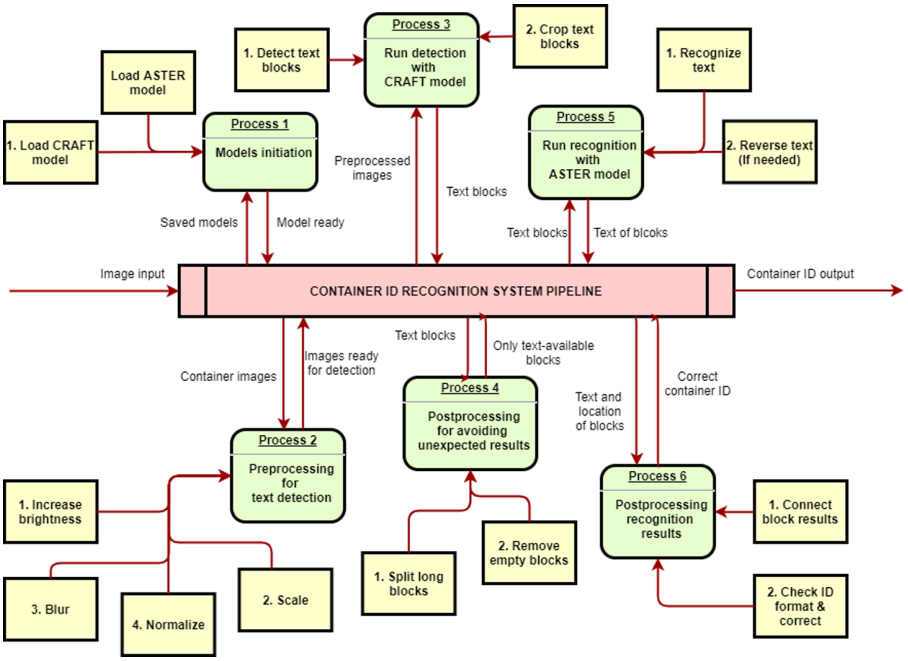

Fig. 3. MultiDeep pipeline

particular, we utilize both nearest neighbor (Eq. 2) and linear method (Eq. 3), and the results obtained are nearly analogous.

$$P'(i,j) = P(i',j') \text{ where } i',j' = \arg\min_{x,y} distance((i,j),(x,y)) \tag{2}$$

$$P'(i,j) = \sum_{k,t} w(k,t,i,j)P(k,t) \tag{3}$$

$w(k,t,i,j)$: the weighted coefficient calculated for (k,t) based upon the distances of pixels to (i,j)

3. *Blurring:* this process involves in boosting the smoothness of the image. The low pass filter, in form of a kernel, allows low frequency elements to enter and thwarts high frequency ones from dominating the whole image, being depicted in 3×3 as

$$\text{filter} = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{4}$$

4. *Normalization:* the final stage is mostly linked to the histogram stretching algorithm, which is a procedure to boost the image's contrast measures. The

adjustment for each pixel can be described as

$$P(i) = \frac{cdf(i) - cdfmin}{(width \times height) - cdfmin}(L-1) \tag{5}$$

cdf: cumulative distribution function of pixel values specified on each image
L: the range of pixel value.

The result of this process is illustrated in Fig. 4.

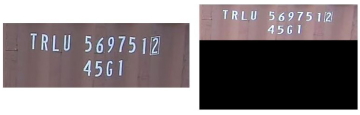

Fig. 4. Original picture and the preprocessed one.

3.3 Process 3: Text Area Recognition

In this process, we employ the CRAFT model to detect text areas on the image. Notice that the result of process 2 has significant effects on the increase of CRAFT accuracy, as illustrated at Fig. 5.

Fig. 5. Text detection with preprocessing (left) and without preprocessing (right)

3.4 Process 4: Text Area Recognition Postprocessing

We perform postprocessing for the resulting text areas from Process 3. It includes two steps as follows.

1. *Long block splitting*: Splitting is an image processing technique used to segment an image. The image is successively split into quadrants based on a homogeneity criterion. Here we split long blocks whose length is greater than 7 characters, in accordance with the ISO6346 discussed above. Cutting text region on scaled image which is the input image for CRAFT model in previous process has been scaled to a certain width. The image with long block will have more than 7 characters. Measuring a specific width, we can define a threshold. If block's width is greater than threshold, we consider that is long block.
2. *Empty block removing*: In our model, we need to detect some texts which have blanks. For example, "PONU 010659 6" is a text which has a blank area. Unfortunately, ASTER model is not trained for images whose contents include blank areas. As a consequence, we remove that blank area in this postprocessing step.

The result of this process is illustrated in Fig. 6.

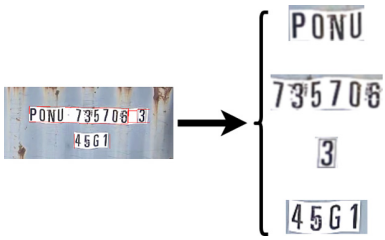

Fig. 6. Illustration of text area recognition postprocessing step

3.5 Process 5: Character Recognition

At this stage, we undertake character recognition using the ASTER model. However, once executing the ASTER model, result we got may be the opposite of our expectation, e.g. the result may be "321" whereas expected as "123". Thus, we also carry on the *text inversion job* (once needed), as illustrated in Fig. 7.

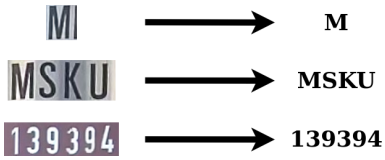

Fig. 7. Illustration the character recognition step, when the code is broken down into sub-parts and recognized accordingly

3.6 Process 6: Character Recognition Postprocessing

Last but not least, postprocessing is carried out with the input which is the results from Process 5. It comprises two steps:

1. Block result connection: In this step, we connect those relevant segmented image blocks to get an intact and clear foreign target image.
2. ID format correction: In this step, we utilize heuristics and experiments to correct mistakes caused by the recognition process. Since the format of the container is well-defined as discussed, we use heuristics to correct the ID when detecting a mismatching.

The result of the final step is illustrated in Fig. 8.

Fig. 8. Illustration of the postprocessing step

3.7 Summary

In the summary, an image is passed into our pipeline will go through all above step and be returned as in Fig. 9:

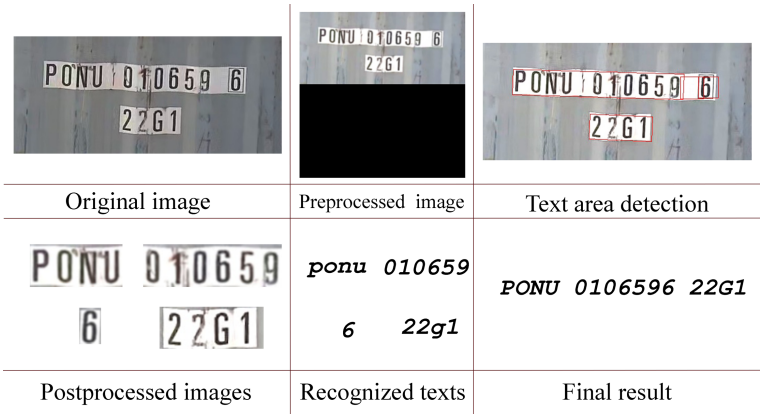

Fig. 9. An overview of returned results after each step.

4 Experiment

In this experiment, we put into use a dataset with a total of 700 images capturing real-life container images, provided by the CyberLogitec company. The whole dataset consists of 7 divided ones (100 images per set) as described below:

- Dataset 1: Images that are clear, easy to recognize, with black letters and bright background.
- Dataset 2: Images that are clear, easy to recognize, with white letters and dark background.
- Dataset 3: Images with inclined letters (Fig. 2c).

- Dataset 4: Images captured with diverse camera angles (Fig. 2d).
- Dataset 5: Images that have different brightness (Fig. 2e).
- Dataset 6: Blurry, unclear images due to the far distance of camera when taking picture (Fig. 2f).
- Dataset 7: Images with deformed or discolored letters (Fig. 2g).

For comparison being fair and most suitable, we divide the whole process into 2 steps: detection and recognition. In detection step, we use F1-score computed using precision and recall to compare between methods, and in recognition step, it is Character Error Rate (CER).

The experiment results show that the deep learning models suffer from a great deal of difficulty when dealing with real images captured from surveillance cameras. We use CER (character error rate) to evaluate the accuracy of the models.

In our experiments, even though the most powerful models and approaches are deployed, including ones popularly applied in industry, the results are still poor without proper pre-and post-processing stages. The Tesseract-OCR model obtain imperfect results because the training of this model does not match the test images. Meanwhile, with EAST-Tesseract, the result of text detection is clear, but the model often misses a number of characters. Despite that, unfortunately, the text recognition phase is not effective. For the EAST-StarNet model, the result of text detection is close to EAST-Tesseract model, but the StarNet model is better than Tesseract. When we combined CRAFT with other character recognition models, the image needs scaling to the appropriate size. Notwithstanding, due to its low resolution, the image became pixelated and the experimental results suffer poor performance. When we enhanced them with post-processing process, the detection of the letters shows positive results, however, the recognition does not meet our expectation.

Our MultiDeep pipleline, once fully deployed, enjoys the most accurate results, which are sufficient to be applied in the real application. The result accuracy is not high, but it is up to our expectation. It is also noted that, among all literature, MultiDeep is the sole model that can handle the container code recognition in an end-to-end manner.

We can see that, our failed cases can interfere human to read. In 6th dataset, failed cases have some character's color mixed with background's, and in 7th dataset, they are very blur and skew, so that pretrained CRAFT model can not detect them. Due to the above errors, Aster model and our postprocessing methods gives wrong outputs. These result is showed in Fig. 10 (Tables 1, 2 and 3).

Table 1. Text detection evaluation with F-measure.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7
East	0.9205	0.7713	0.9499	0.5055	0.7458	0.7963	0.9373
Pre + East	0.8971	0.5961	0.9283	0.3492	0.6087	0.8107	0.9408
Craft	0.4021	0.7079	0.3715	0.8548	0.6079	0.4019	0.3474
Pre + Craft	0.9879	0.9794	0.9819	0.9817	0.9447	0.7808	0.9683

Table 2. Text regconition evaluation using character error rate.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7
Tesseract	99.6%	100%	99.9332%	99.7898%	99.5506%	99.6916%	99.8667%
StarNet	13.2%	17.8667%	17.4465%	18.5004%	16.7041%	22.2051%	12.4%
Aster	5.6667%	11%	24.4652%	27.3301%	12.6592%	26.2143%	12.7333%

Table 3. End-to-end evaluation.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7
Multi-deep	65%	61%	35%	65%	40%	15%	21%

(a) 6th dataset

(b) 7th dataset

Fig. 10. The results from 6th and 7th dataset

5 Conclusion

Container code recognition is a realistic problem appealing to huge attention due to its practicality. One of emerging trends to address this puzzle is employing deep learning models trained for text recognition. Nonetheless, these models are sensitive to instability of real images' quality captured at the sites. By presenting MultiDeep pipeline, we are capable of recognizing the container identifica-

tion and obtaining its right format. We have implemented MultiDeep with real dataset of an industry company and obtained promising initial results. With the current results, we need to optimize the model to seek better recognition so as to better adapt to practical problems.

In our experiments, we can take into account real-life images which are not always helpful for the machine learning models to yield high results. There are multiple obstacles when performing character recognition in real life. Besides, there are approaches which can help to boost the accuracy, some of which are listed below.

- Providing re-trained deep models of CRAFT and ASTER with larger dataset.
- Trying other Computer Vision algorithms in preprocessing and postprocessing steps.
- Improving the heuristics algorithms in postprocessing after recognition.

Those are also the directions we intend to pursue in future researches.

Acknowledgement. We are grateful to CyberLogitec company for funding and providing real dataset for this research.

References

1. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9365–9374 (2019)
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
3. Zhou, X., et al.: East: an efficient and accurate scene text detector. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5551–5560 (2017)
4. Liu, W., Chen, C., Wong, K.Y.K., Su, Z., Han, J.,: Star-net: a spatial attention residue network for scene text recognition. In: BMVC, vol. 2, p. 7 (2016)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
6. Shi, B., Yang, M., Wang, X., Lyu, P., Yao, C., Bai, X.: Aster: an attentional scene text recognizer with flexible rectification. IEEE Trans. Pattern Anal. Mach. Intell. **41**(9), 2035–2048 (2018)
7. Smith, R.: An overview of the tesseract OCR engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), vol. 2, pp. 629–633. IEEE (2007)
8. Pan, W., Wang, Y., Yang, H.: Robust container code recognition system. In: Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788), vol. 5, pp. 4061–4065. IEEE (2004)
9. Wu, W., Liu, Z., Chen, M., Liu, Z., Wu, X., He, X.: A new framework for container code recognition by using segmentation-based and hmm-based approaches. Int. J. Pattern Recognit. Artif. Intell. **29**(01), 1550004 (2015)

10. Wu, W., Liu, Z., Chen, M., Yang, X., He, X.: An automated vision system for container-code recognition. Exp. Syst. Appl. **39**(3), 2842–2855 (2012)
11. Verma, A., Sharma, M., Hebbalaguppe, R., Hassan, E., Vig, L.: Automatic container code recognition via spatial transformer networks and connected component region proposals. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 728–733. IEEE (2016)