An Adaptive Deep Learning Framework for Shipping Container Code Localization and Recognition

Ran Zhang, Zhila Bahrami, Teng Wang, *Student Member, IEEE*, and Zheng Liu, *Senior Member, IEEE*

***Abstract*— Shipping containers play an important role in global transportation. As container codes are the unique identifiers for shipping containers, recognizing these codes is an essential step to manage the containers and logistics. The conventional code localization methods can easily be interfered by varied noises and cannot identify the best regions for code recognition. In this article, we propose an adaptive deep learning framework for shipping container code localization and recognition. In the framework, the noisy text regions will be removed by an adaptive score aggregation (ASA) algorithm. The code region boundaries are identified by the average-to-maximum suppression range (AMSR) algorithm. Thus, the predicted locations can be adjusted within this range to fit the code recognition model to achieve higher accuracy. The experimental results on the comparative study with the state-of-the-art models, including EAST, PSENet, GCRNN, and MaskTextSpotter, demonstrated that the proposed framework achieved better localization performance and obtained 93.33% recognition accuracy. The processing speed reaches 1.13 frames/s, which is sufficient to meet the operational requirements. Thus, the proposed solution will facilitate the digital transformation of shipping container management and logistics at ports.**

***Index Terms*— Adaptive container code localization and recognition, convolutional neural network (CNN), deep learning, end-to-end text recognition, recurrent neural network (RNN), text detection.**

I. Introduction

SHIPPING containers are one of the most important assets in global transportation. A great number of containers are carried by ships, trains, and trucks. They are an indispensable part of international trading due to the large transportation capacities at relatively low freight rates. When shipping containers transit through ports, the port operators need to track and manage them. Intermodal containers shift among trucks, trains, and ocean transportation at terminals. Vehicle localization [1], [2] and object tracking [3] can detect and track trucks, whereas trajectory recognition [4] and estimation [5] provide better trajectory guidance for trucks. Nevertheless, recognizing shipping container code in a timely and precise manner is still necessary, as containers are moved from place to place and not restricted in a certain truck or ship. According to ISO 6346 [6], standard shipping container code is an 11-digit alphanumeric character sequence. The meaning of the code is shown in Fig. 1. The first four characters must be capital English letters. The next part is the six-digit numeric sequence and the last one is the check digit number.

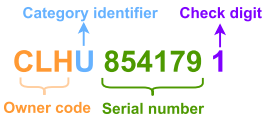


Fig. 1. Meaning of the shipping container code. The first three digits are the owner code that must be upper case letter. The fourth digit indicates which kind of equipment it is. The category identifier should also be upper case letter. The next six digits are serial numbers to identify the container in the owner. The last one is the check digit. It is used to validate the previous digits.

The port terminals used to manually check and record the container codes one by one. However, the costs of labor are high, and the checkers often make mistakes due to repetitive work that causes fatigue. Therefore, the automatic container code recognition (ACCR) is needed. Machine vision-based ACCR is widely used because the equipment can be easily deployed by putting cameras at the gates of ports. The machine vision-based methods integrate a variety of technologies and computational methods to provide imaging-based information [7]. A lot of studies have been conducted to recognize the container code from images automatically.

ACCR frameworks usually consist of two individual steps, code localization and recognition. For the code localization, traditional methods use features of strokes, contours, gray levels [8], edges, and histogram information from grayscale images. Filters [9] and masks [10] are also used to extract the text area features. Code localization is followed by the recognition. Traditional code recognition methods are divided into two steps, character isolation and single character recognition. A commonly used technique in character isolation is connected components analysis [11], which can find separated characters. When it comes to the single alphanumeric character recognition, image features, including histograms, gradients, and edge densities generated by the Sobel filter, are taken as inputs of the classifiers. Previously used classifiers include

Manuscript received February 23, 2020; accepted July 29, 2020. Date of publication August 13, 2020; date of current version November 20, 2020. This work was supported by Mitacs under Grant IT12857. The Associate Editor coordinating the review process was Dr. Dong Wang. *(Corresponding author: Zheng Liu.)*

The authors are with the School of Engineering, Faculty of Applied Science, The University of British Columbia Okanagan Campus, Kelowna, BC V1V 1V7, Canada (e-mail: ran.zhang@alumni.ubc.ca; zhilaai@mail.ubc.ca; teng.wang@ubc.ca; zheng.liu@ubc.ca).

Digital Object Identifier 10.1109/TIM.2020.3016108

support vector machine (SVM) and adaptive resonance theory network [12]. Single character recognition can also be implemented by template matching [13], [14], which can find the similar patterns with the templates. He *et al.* [13] used the median filter and adaptive linear filter to select the code lines and adopted template matching for code recognition. Kim *et al.* [12] used Sobel masking to detect edges and localize code and then proposed eight-directional contour tracking mask to extract features for code recognition. As character U is the category identifier for the freight containers, Kumano *et al.* [9] searched for the location of U to localize the container code. Wu *et al.* [15] designed a horizontal high-pass filter and scanline analysis strategy for code localization, implemented vertical projection on histogram for character isolation, and then used SVM for single-character recognition.

Recently, different deep learning methods have been designed for scene text detection and recognition. EAST [16], which is based on fully convolutional neural network (CNN), is capable of detecting text with different angles. PSENet [17] leverages feature pyramid networks [18] to extract features for scene text. With progressive scale expansion, PSENet is able to detect text with arbitrary shapes. For scene text recognition, convolutional recurrent neural networks (CRNNs) [19] can recognize text sequences in arbitrary length. A gated recurrent convolution neural network (GRCNN) [20] adds a gate to the recurrent convolution layer to recognize text in natural images. MaskTextSpotter [21], an end-to-end trainable neural network for text localization and recognition, can handle irregular text, including oriented text and curved text. Since the container code is one kind of scene text, a lot of deep learning models are introduced to container code localization and recognition. Mei *et al.* [22] combined template matching and CNNs for container code character recognition. Verma *et al.* [23] leveraged spatial transformer networks to recognize the single characters. Roeksukrungrueang *et al.* [24] used LeNet for code recognition. Liu *et al.* [25] combined a connectionist text proposal network and maximally stable extremal regions for code localization and used both character segmentation and CRNN for code recognition.

These previously code localization and recognition methods perform effectively in the condition with abundant illuminations and clear backgrounds. However, there are still a few limitations to be overcome. First, the existing localization models cannot distinguish container codes from noisy text in complex backgrounds. In the real industry environments, the distances of the camera to the shipping containers, as well as the illuminations, change greatly. Both of the noise text and codes are characters and have similar features. For instance, license plates of trucks and capacity information text on the backdoor of the shipping containers are noisy text. It is difficult for the existing methods to distinguish the container codes from noisy text in these situations. Second, the conventional shipping container code localization and recognition models work separately. The code localization model outputs the fixed areas of shipping container codes, which cannot be adjusted. These areas are cropped and directly used as the inputs of the code recognition model. There is no feedback from the recognition part to the localization part. The traditional open-loop frameworks lead to relatively low accuracy in the code recognition procedure.

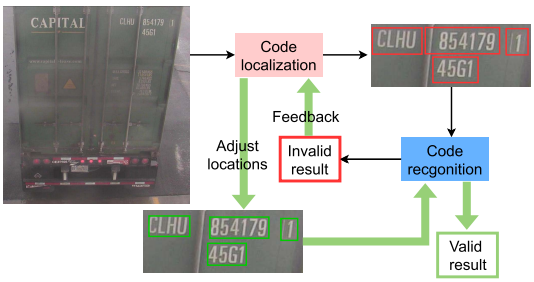


Fig. 2. Overall structure of our proposed adaptive framework for code localization and recognition.

Our contributions are summarized as follows.

1) In the adaptive deep learning framework, we propose an adaptive score aggregation (ASA) algorithm to clean the low confident and noisy localization proposals. ASA aggregates high confidences in the backdoor code areas that are clear and complete. Thus, the localization model is able to focus on these regions.
2) In this framework, there is a feedback mechanism from the recognition model to the localization model, as shown in Fig. 2. Different from the traditional ACCR framework, the localization outputs in our adaptive framework can be adjusted in a range given by our designed average to maximum suppression range (AMSR) algorithm. Therefore, the recognition model can achieve higher accuracy by choosing more appropriate input images.
3) A better code localization performance can be achieved by constructing localization deep neural networks and properly defining the loss in the training process.

II. Adaptive Framework for Shipping Container Code Localization and Recognition

This article proposes an adaptive deep learning framework for shipping container code localization and recognition. Different from traditional ACCR methods, the predicted locations of container codes in our adaptive framework can be adjusted to the best positions for the recognition model. We construct our localization model based on Resnet [26] and U-net [27] and recognition model based on CRNN [19]. The inputs of the localization model are images containing the shipping container backdoor, and the outputs are ranges of possible code regions. Then, CRNN is used to recognize codes in cropped code areas. After the code recognition step, we check whether the recognition results meet the ISO 6346 international shipping container standard. If the results meet the requirement, the process of recognizing codes in this image ends with valid container codes. Otherwise, the predicted locations of shipping container codes should be adjusted according to our designed strategy.

A. Shipping Container Code Localization

The localization model includes the feature extraction process, upscaling process, and postprocessing steps. The

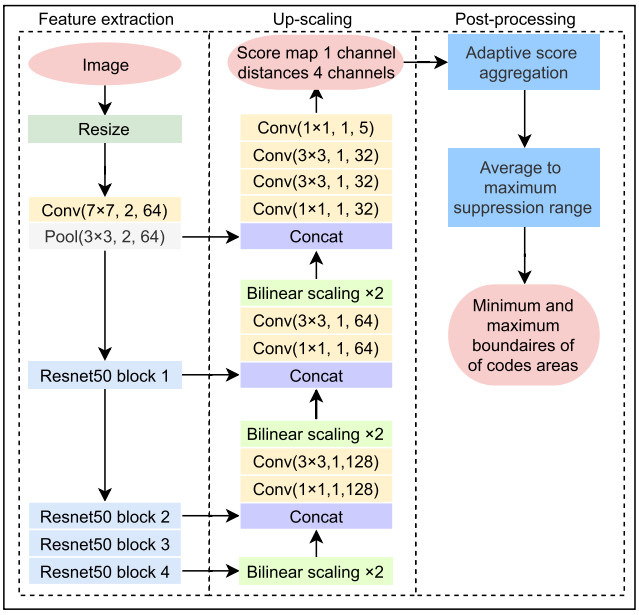


Fig. 3. Structure of proposed localization model. Feature extraction and upscaling is based on U-net and Resnet. We define the loss function for better localization performance. In postprocessing, we design the ASA and AMSR algorithms for adaptive localization.

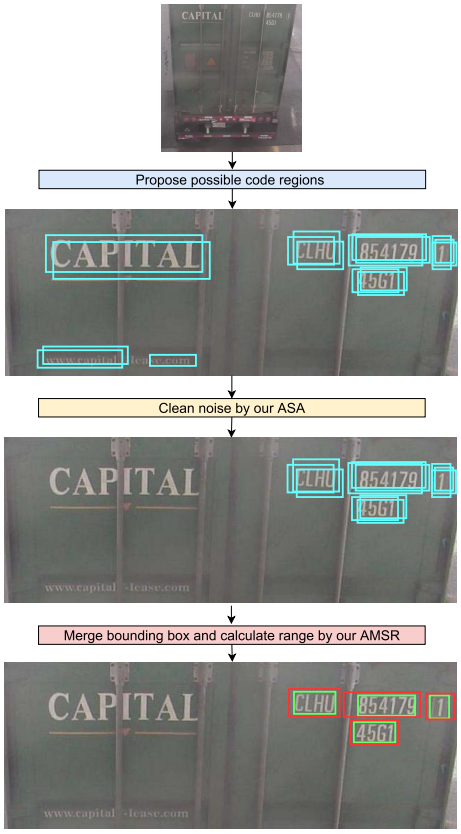


Fig. 4. Example of code localization. The feature extraction and upscaling process proposes possible code regions. However, there are noisy text regions that we do not need. Our ASA cleans these noisy text regions. After ASA, code areas have overlapped bounding boxes. Therefore, our AMSR merges these overlapped proposals and outputs the minimum and maximum code boundaries.

ASA and AMSR algorithms are in postprocessing steps. The localization model structure is shown in Fig. 3. One example of the localization process can be seen from Fig. 4. We adopt Resnet [26] as the backbone in the feature extraction process. In this process, the size of feature maps gets smaller and smaller to learn higher level features. The next process is the upscaling process when feature maps get larger and larger. Outputs of a few feature extraction layers are used as a part of the inputs of the concatenation layers in the upscaling process. The upscaling process finally outputs the one score map channel and four distance map channels. The score map indicates the confidences, or possibilities of every pixel location being in the shipping container code areas, whereas the distance maps measure distances of every pixel to the boundaries of code areas.

After the upscaling process, the postprocessing is leveraged to get the boundaries of shipping container codes. We propose the ASA algorithm for score map cleaning to remove low confidence detection, as well as wrong predictions on the noisy text. Thus, our model can focus only on the most confident part, which is the backdoor shipping container code area. After that, AMSR merges the rest overlapped bounding boxes that have relatively high confidences. The outputs of the localization model are the minimum and maximum boundaries of possible locations of the container codes.

1) Feature Extraction and Upscaling: The first step of feature extraction is to resize the input image according to the requirements of the following layers. The height and width of the input images should be multiple of 32 because there are one pooling layer and four convolutional layers with the stride of 2 during the feature extraction process. The parameters for these layers are given in Fig. 3. For instance, conv(7 × 7, 2, 64) means that the kernel size is 7 × 7 with stride 2, and the output feature map dimension is 64. The detailed layers of Resnet blocks are listed in Table I. We set the stride with 2 at the last layer of Blocks 1–3. The strides in other layers are set with 1. The concatenation layers merge two sets of feature maps into one set.

TABLE I
BLOCKS DETAILS IN THE ADOPTED RESNET

Blocks	Residual Units Number	Convolutional Layers in Each Unit
Block1	3	Conv(1×1,1,64) Conv(3×3,1,64) Conv(1×1,stride,256)
Block2	4	Conv(1×1,1,128) Conv(3×3,1,128) Conv(1×1,stride,512)
Block3	6	Conv(1×1,1,256) Conv(3×3,1,256) Conv(1×1,stride,1024)
Block4	3	Conv(1×1,1,512) Conv(3×3,1,512) Conv(1×1,1,2048)

Bilinear scaling is adopted in the upscaling process. Four pixels in an adjacent square area are considered as one unit. The scaling ratio is s. After the image scales up, the left-top pixel coordination is $(0, 0)$, and the right-bottom

pixel coordination is (s, s). The position of a new pixel can be denoted as (i, j), where i and j are the horizontal and vertical distance to the left-top position $(0, 0)$, respectively. $M_{(i,j)}$ is the value at position (i, j), which is calculated according to the following equation by bilinear scaling:

$$M_{(i,j)} = \frac{1}{s^2}(M_{(0,0)}(s-i)(s-j) + M_{(s,0)}i(s-j) + M_{(0,s}(s-i)j + M_{(s,s)}ij). \quad (1)$$

The outputs of the upscaling process are four distance channels and one score map channel. In the training process, the score map values are set to be 1 inside the bounding box of the labeled text area, and the other locations are set with 0. In the predicting process, the score map values should be less than or equal to 1. Values in the score map indicate the probabilities or confidences of the pixels being in text areas. With the score map and distance channels, the predicted text areas and their confidences can be calculated.

In the training process, the loss function L of the localization network is defined by

$$L = \lambda_s L_s + L_d \quad (2)$$

where L includes score map loss L_s and distances loss L_d. λ_s is the weight factor for L_s. In our model, we set this value to be 0.01 and use the dice coefficient loss function for L_s

$$L_s = \frac{1 - (2|P \odot T|)}{(|T| + |P| + 10^{-5})} \quad (3)$$

where P is the predicted score map, T is the target score map, and $\odot$ is the elementwise product. A distance loss function L_d consists of two parts

$$L_d = -\ln \mathrm{IoU}(A, A^*) - \ln \mathrm{proj}(A, A^*) \quad (4)$$

where A is the predicted code area and A^* is the ground-truth code area. On the right side of this equation, the first part is the loss for the intersection over union (IoU), and the second part is the loss for the projection of predicted area to the ground-truth area in the horizontal direction. The loss for IoU makes the model's prediction outputs can match the target area as much as possible, whereas the loss for the projection in the horizontal direction contributes to the output's horizontal coverage. Even though our designs in loss function try to make the predicted horizontal coverage being a little bit wider than the actual text area, this kind of prediction is more helpful in the following text recognition part since it can include the codes completely. The IoU and horizontal projection can be calculated by the following equations:

$$\mathrm{IoU}(A, A^*) = \frac{A \cap A^*}{A \cup A^*} = \frac{(\min(d_2, d_2^*) + \min(d_4, d_4^*))(\min(d_1, d_1^*) + \min(d_3, d_3^*))}{(\max(d_2, d_2^*) + \max(d_4, d_4^*))(\max(d_1, d_1^*) + \max(d_3, d_3^*))} \quad (5)$$

$$\mathrm{proj}(A, A^*) = \frac{\min(d_2, d_2^*) + \min(d_4, d_4^*)}{d_2^* + d_4^*} \quad (6)$$

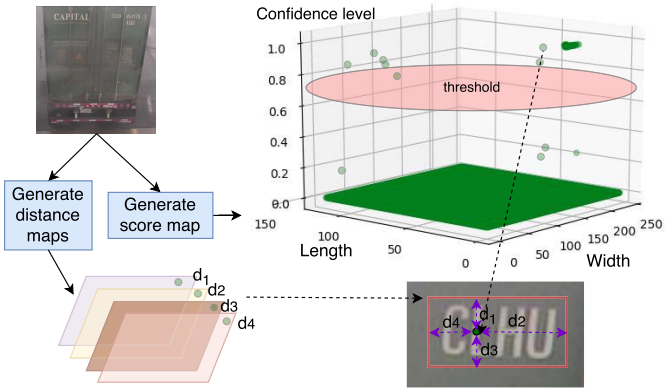


Fig. 5. Illustration of the text region proposal process. The localization model generates four distance maps and one score map. Regions with higher confidence levels above the threshold are proposed. The bounding box location is constructed from distances.

where d_1, d_2, d_3, and d_4 represent the values of four distance channels in prediction results, that is, distance to the top, right, bottom, and left boundaries of the predicted text area. Also, d_1^*, d_2^*, d_3^*, and d_4^* are the distances in ground truth. If one pixel (x, y) is inside a text area $[x_1, y_1, x_2, y_2]$, these four distances can be calculated as follows:

$$d1 = |y - y1| \quad (7)$$

$$d2 = |x2 - x| \quad (8)$$

$$d3 = |y2 - y| \quad (9)$$

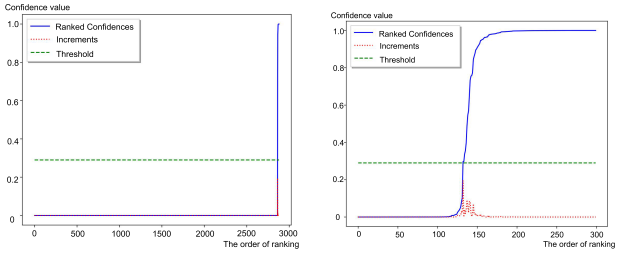
$$d4 = |x - x1| \quad (10)$$

where (x_0, y_0) is the left-top coordinate of the bounding box and (x_1, y_1) is the right-bottom coordinate.

After the feature extraction and upscaling, the network proposes possible shipping container code regions with score map values, i.e., confidences. The text region proposal process is shown in Fig. 5.

2) Postprocessing I, ASA on Score Map: In the postprocessing for shipping container code localization, we propose ASA to remove the low confidence regions as well as noisy areas. ASA calculates an adaptive threshold and aggregates confidences in adjacent regions. Different from threshold methods in image segmentation [28] which filter the pixels in images, our adaptive threshold in ASA focuses on the confidence levels of possible text areas in the score map. Thus, the adaptive threshold is used to remove the low confidence text proposals. Next, bounding boxes that draw the boundaries of text in adjacent areas are aggregated. Meanwhile, the scores of bounding boxes are accumulated in this area. At last, the set of bounding boxes that are close to each other and have the highest aggregation scores is selected as the outputs of ASA.

If the threshold for score map cleaning is set with high value, some areas with codes will be cut off. However, if the threshold is low, many false-positive positions will be added into the final localization outputs. Most score values in the score map are at low levels, as shown in Fig. 6(a), and only a small portion of them are relatively high. Therefore, the threshold in our proposed ASA is set adaptively according to the distribution of the score map values.



(a) (b)

Fig. 6. Example of ranked confidences on score map from low to high. The x-axis is the ranked order and the y-axis indicates the confidence values. Blue line is the ranked confidence value and red line is the increase of current value compared with the previous value. The green line is the adaptive threshold for the score map. In this example, (a) whole score map has around 3000 pixels and their ranked values. (b) Highest 300 confidences.

ASA consists of two steps, adaptive threshold calculation and scores aggregation. In order to calculate adaptive threshold, scores, or confidences, are ranked from low to high. The score map is a 2-D matrix corresponding to the confidences of the text area. We reshape this matrix to a 1-D vector and rank them from low to high. These ranked scores are denoted as $B(b_1, b_2, b_3, \ldots, b_n)$, and n is the length of this vector, which is equal to the total number of pixels in this score map. The value changes in the ranked scores can be defined as $R(r_1, r_2, r_3, \ldots, r_n)$, where $r_1 = b_1$, and $r_k = b_k - b_{k-1}$ if $2 \leq k \leq n$.

Then, the index I of inflection point can be found by

$$I = \arg_k \max(r_k), \quad 1 \leq k \leq n. \tag{11}$$

Thresholds are set between 0.2 and 0.8 to avoid extremely high or low values. Therefore, the adaptive threshold h can be calculated by

$$h = \begin{cases} 0.8, & \text{if } b_I \geq 0.8. \\ b_I, & \text{if } 0.2 < b_I < 0.8. \\ 0.2, & \text{if } b_I \leq 0.2 \end{cases} \tag{12}$$

where b_I is the score value at the inflection point. Then, the scores that are larger than or equal to the threshold h are considered as relatively high confidence. They are kept for processing in the next step. Scores that are less than h are removed before the next process. Fig. 6 shows an example of finding the adaptive threshold. The red line indicates the value changes compared with the previous score value. The maximum value of r_k is the maximum value of the red line in the y-axis, and the index I of inflection point is the x-axis value at that point. Finally, we can get the threshold value from the blue line at index I. Scores of recent areas are aggregated after the threshold. The set of bounding boxes with the highest aggregation score will be selected. The score aggregation process is shown in the below process.

1) Sort the bounding boxes by their left-top coordinates from high to low.
2) Select the first bounding box.
3) Put the selected bounding box bb_j into a new area set AS_i. The range of this new area set R_i is the same with this bounding box area r_j.

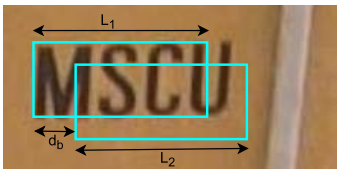


Fig. 7. Minimum border distance d_b is used to measure whether two bounding boxes are close or not. In this figure, d_b is the minimum distance of vertical edges. If d_b is less than $\min(L_1, L_2)$, we consider that these two bounding boxes are close horizontally.

4) Select the next bounding box. If all bounding boxes are calculated, go to step 6. Otherwise, go to step 5.
5) If the selected bounding box bb_m is close to one of the existing area set range R_n, put it into this area set AS_n and expand the area range R_n, which can cover the newly included bounding boxes. Otherwise, go to step 3.
6) Aggregate the scores of all bounding boxes in every area set. Output the area set AS_h with the highest aggregation score.

The minimum border distance d_b is used to measure whether two areas or bounding boxes are close or not. In the horizontal direction, d_b is defined by

$$d_b = \min(d_{11}, d_{12}, d_{21}, d_{22}) \tag{13}$$

where d_{ij} is the horizontal distance of vertical edge i in one bounding box to vertical edge j in another bounding box. If d_b is less than the width of the smaller area as shown in Fig. 7, we consider that they are close in the horizontal direction. In the same way, we can judge whether the two bounding boxes are close in the vertical direction. Only if they are close in both horizontal and vertical directions, they are considered close in our ASA process.

The ASA process finally outputs bounding boxes in code regions with high confidences.

3) Postprocessing II—Merging Overlapped Detected Areas by Using AMSR : After the low confident and noisy bounding boxes are removed by ASA, there are still some overlapped predictions or bounding boxes that should be merged in the code regions. Nonmaximum suppression (NMS) is a traditional and commonly used postprocessing method to handle overlapped bounding boxes. However, NMS only chooses the most confident detection result and discards other overlapped bounding boxes with lower confidence. The final bounding boxes of the detected area after NMS might be smaller than the ground truth. Therefore, we propose the AMSR algorithm to compute the minimum and maximum boundaries, which gives the range of possible text areas. Compared with standard NMS, AMSR can provide better text localization areas for the next text recognition step. The recognition model can search for suitable regions from AMSR then get better performances. Our proposed AMSR in one possible text area includes two bounding boxes and their confidence $[B^a, B^m, conf]$, where $B^a = (x_1^a, y_1^a, x_2^a, y_2^a)$ and $B^m = (x_1^m, y_1^m, x_2^m, y_2^m)$. B^a indicates the minimum boundaries of the AMSR. We use weighted merge [16] instead of the minimum merge to decide the minimum boundaries. B^m is the maximum boundaries.

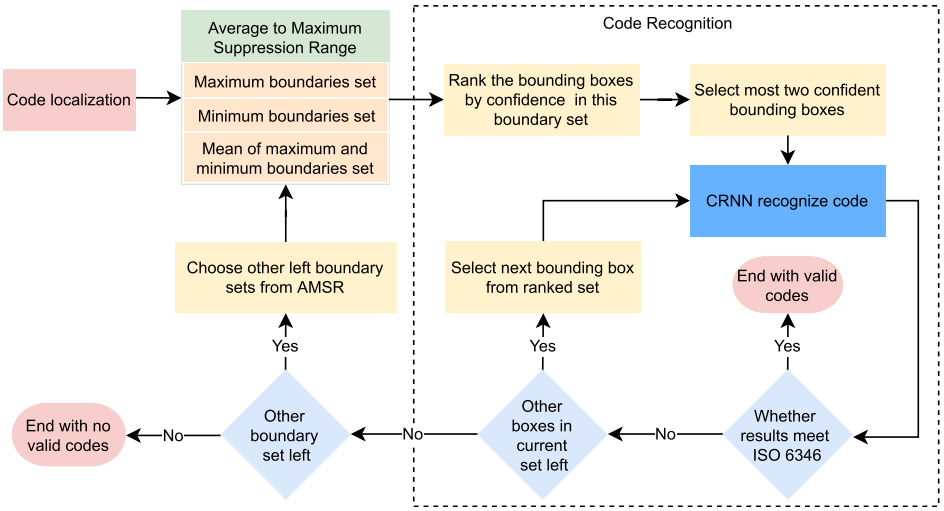


Fig. 8. Strategy for adaptive localization.

Algorithm 1: AMSR

Function AMSR (B, C);
Input : A list of bounding boxes B after score map cleaning, and their confidences C
Output: minimum and maximum boundaries B^a, B^m, and their confidences C^{out}
Sort B by confidence from high to low
while $B \neq \varnothing$ **do**
  $b_0^m \leftarrow$ pop out the highest confident bounding box from B
  $conf_0 \leftarrow$ pop out the highest confidence from C
  $b_0^a \leftarrow b_0^m$
  $l_b \leftarrow length(B)$
  for $j \leftarrow 0$ *to* l_b **do**
    if $IoU(b_0^m, b_j) > 0.3$ **then**
      $conf_0 \leftarrow max(conf_0, conf_j)$
      $b_0^a \leftarrow conf_0 * b_0^a + conf_j * b_j$
      $b_0^m \leftarrow maxmerge(b_0^m, b_j)$
      remove b_j from B
    end
  end
  append $b_0^a, b_0^m, conf_0$ into B^a, B^m, C^{out}
end

(x_1, y_1) is the top-left coordinate of the bounding box and (x_2, y_2) is the right-bottom coordinate. The proposed AMSR is shown in Algorithm 1.

After computing the AMSR in the postprocessing part, the localization model gives the minimum and maximum boundaries of container codes with their confidences.

4) Strategy for Adaptive Localization: An adaptive localization strategy is designed in our framework. This strategy can take advantage of the maximum and minimum boundaries from AMSR to provide multiple choices for the recognition model.

In our adaptive strategy, we give three possible boundary sets for AMSR, that is, maximum boundary set, minimum boundary set, and mean boundary set, which is the average of maximum and minimum boundaries. More boundary sets can be designed for the code recognition part. For instance, the mean of maximum boundary set and mean boundary set. Our adaptive framework is expected to get higher recognition accuracy provided with more boundary sets. If one of these boundary sets cannot get the required shipping container codes, our adaptive framework will automatically choose the next boundary for the container code recognition. In our experiments, we only give three boundary sets to avoid the possible endless loop. The adjustment strategy of localization can be seen from Fig. 8.

There are a few bounding boxes in one boundary set. These bounding boxes outline different parts of the container code. For example, in the maximum boundary set, the bounding boxes outline the maximum regions of container codes. Then, the code recognition model is used to recognize the text inside these bounding boxes. The results are finally merged into a code sequence.

B. Shipping Container Code Recognition

1) Code Recognition and Sequence Combination: Our adaptive deep learning framework aims to detect and recognize the 11-digit shipping container code. The outputs of our code localization consist of three bounding boxers if the last check digit number is far away from the previous numbers as Fig. 9(a). However, when the check digit is too close to the number on their left side, the seven-digit numbers will be put into one bounding box in the localization outputs, as shown in Fig. 9(b).

Therefore, the recognition results in cropped areas might be four digits, six digits, seven digits, or one digit according to the localization predictions. These results are combined into an 11-digit sequence. In the results combination process, the recognition results in cropped regions are ranked by their localization confidence from high to low. Then, the recognition results with the highest and second-highest confidence are combined into a sequence. If this sequence does not meet the ISO 6346 standard, the next confident bounding box

(a) (b)

Fig. 9. Different combinations of localization outputs. (a) 11-digit shipping container codes in localization procedure can be divided into three parts when the last check digit is far away from other numbers or (b) two parts when they are close.

area is taken into consideration for the combination until the combined sequence meets the standard. Otherwise, if the recognition results of all bounding boxes in a certain set of boundaries, for instance, maximum boundaries, cannot get the results that meet the ISO 6346 standard, the boundary set would be replaced by the next boundary set. In this way, our adaptive framework builds an interactive process between the code localization and recognition, as shown in Fig. 8.

2) CRNN: CRNN [19] is used to recognize a sequence of text without splitting the input images into single character images. Moreover, when we recognize codes by CRNN, there is no need to remove the rectangle box surrounding the check digit.

CRNN has advantages over other models. GRCNN [20] has gated recurrent convolutional layers where the gates control the context modulation. Its receptive fields are reduced compared with CRNN [19]. With reduced receptive fields, GRCNN is able to concentrate on the current single character and eliminate the effects of other pixels during the recognition process. The performance of GRCNN can be improved if the context is not useful for text recognition. However, when it comes to container code recognition, the context provides plenty of information for the container code recognition, as shown in Fig. 1. Therefore, CRNN with larger receptive fields performs better. MaskTextSpotter [21] is an end-to-end trainable neural network utilizing semantic segmentation. However, it still needs to separate characters one by one before character recognition. When the characters are too close to each other, it is hard for character separation. especially when container codes are visually connected to each other caused by the illumination changing and snow covering.

CRNN model consists of three parts, sequence feature extraction by CNNs, sequence scoring by recurrent neural networks (RNNs), and sequence translation by connectionist temporal classification (CTC) [29]. The structure of CRNN is shown in Fig. 10.

In CRNN, deep CNNs are used to learn the characteristics of the inputs and generate the feature sequence. The RNN part consists of two stacked bidirectional long short-term memory (BiLSTM), and every BiLSTM has two layers of LSTM [30]. One of them is forward and the other is backward. Both layers have 256 LSTM units separately. Finally, the concatenation

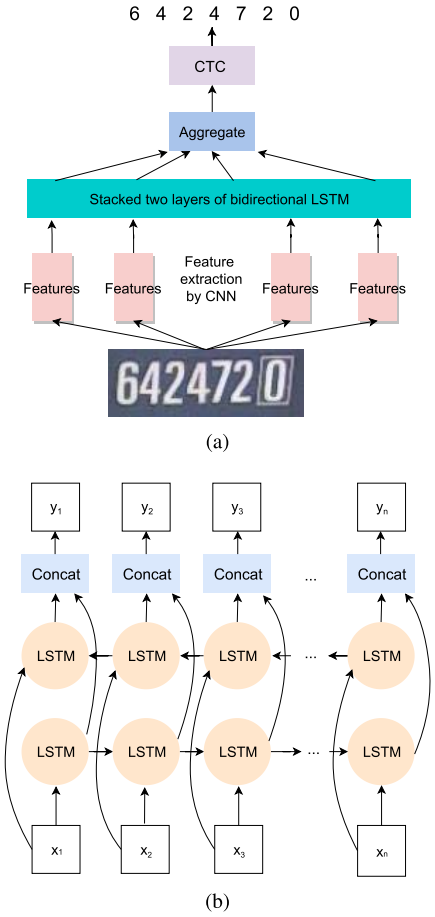


Fig. 10. Used CRNN structure in our framework. (a) Overall structure of CRNN. It takes the cropped code regions as inputs and recognizes text inside them. (b) One bidirectional LSTM layer. There are two layers of bidirectional LSTM stacked in CRNN.

layer is used to combine the RNN outputs and CTC is utilized to find the most likely alphanumeric character sequence, i.e., the predicted results.

The training loss function L of CRNN is defined by

$$L = -\sum_{y^* \in Y^*, y \in Y} \ln p(y^*|y) \tag{14}$$

where Y^* is the ground-truth label sequences of input images and Y is the corresponding predicted results.

III. Experimental Results

A. Data Collection

Experimental images are collected from the gates where trucks with shipping containers must pass through in the ports. The cameras are put at the back-top direction of shipping containers. In our training and testing data, the 11-digit shipping container code areas are completely included in the images. The data set used for code localization contains 3000 images, and the training data ratio is 0.8. After cropping the code regions, we have 23 418 cropped images for the code recognition data set. The target shipping container should not be too far away from the camera; otherwise, the code would be too small to read. There might be some containers that are

far away from the camera but are still in the image. They are considered as background. In order to validate our model in various environments around the shipping container, we take images in different illuminations and weather conditions into consideration when preparing the data set. For illuminations, images are collected in both the daytime with sunlight and the night when extra lighting is added. As for weather conditions, the ports need to keep the recognition system running all year round. Our data set needs to cover all 12 months, so we randomly choose 250 images of shipping containers from every month. All the data are RGB images with the height of 1080 and the width of 1920. Another factor taken into consideration in our data set is the font size. The trucks with containers were moving at different speeds when they passed the gate with cameras. Also, the images in our data set are randomly selected. Therefore, the distances of shipping containers to the camera are different in our data set. This leads to various font sizes of container codes.

We labeled the shipping container code areas with bounding boxes and the actual text in these areas. The 11-digit shipping container number might be included in two or three bounding boxes. The first four digits are upper case characters, which are usually in one bounding box. The rest seven numbers are separated into two parts if the check digit is far away from the other six digits. Otherwise, these seven digits are put into the same bounding box. For one bounding box, we give the label of the left-top, right-bottom coordinates, as well as the text inside this bounding box. The four-digit container type code is usually below the 11-digit shipping container code. We also label this by one bounding box because the text in this area is in a similar size and color with the unique 11-digit shipping container number. This kind of labeling style conforms to the localization outputs shown in Fig. 9.

B. Network Parameters for Training and Evaluation Metrics for Testing

The code localization deep neural networks and CRNN are trained separately, and then, they are put together in an interactive process of code localization and recognition in the testing or predicting procedure. Both the 11-digit shipping container code and four-digit type code are put into training data.

In the training process of our localization model, Adam optimizer [31] and decayed learning rate l_d are used

$$l_d = l_0 \times 0.94^{s/10000} \tag{15}$$

where l_0 is the initial learning rate, which we set it as 0.0001. s is the current training step. In the training process of CRNN, all the text inside one bounding box is taken as a whole target. There is no need to separate every single character and number. The text areas are cropped according to the bounding boxes and used as the input training images for CRNN. We also take the Adam optimizer but keep the learning rate steady to be 0.0001. The input image to CRNN should be resized to fit the model. We set the input height as 32 and keep the aspect ratio when resizing. Different IoU thresholds are used when we are evaluating the localization performance. Higher IoU threshold requires the localization model to fit the ground truth better. The hyperparameters of our framework are summarized in Table II.

TABLE II
HYPERPARAMETERS IN OUR FRAMEWORK

Hyperparameter	value/content
Upper limit of adaptive threshold on score map	0.8
Lower limit of adaptive threshold on score map	0.2
IoU threshold for merging bounding boxes	0.3
Training data ratio	0.8
Initial Learning rate for localization model	0.00001
Learning rate for recognition model	Steady at 0.00001
The height of resized inputs of CRNN	32
Layers of bidirectional LSTM in CRNN	2
IoU threshold used to evaluate performance	0.5, 0.55, 0.56, 0.60, 0.70, 0.75

To evaluate the performance in the testing process, we set a few metrics for localization and recognition. From the top-rear direction of the shipping container, we might see the codes at the backdoor and roof simultaneously. We do not consider the localization predictions of codes on the roof of the shipping container since the roof of shipping containers are easier to be corroded or sheltered by snow. Codes at that area are not as clear and compact as those on the backdoor. If the codes on the roof area are considered, the code recognition model will be affected by these bad quality data. Therefore, our code localization model is trained to ignore predictions on the roof of the shipping container.

We choose precision, recall, f1-score, and average precision (AP) to evaluate the performance of code localization. In the field of object detection, AP is almost equivalent to Area Under the ROC Curve (AUC). We adopt 11-point interpolated AP defined in PASCAL visual object classes challenge [32]. Also, accuracy is used for code recognition. Only when the whole predicted 11-digit shipping container code sequence is the same with the ground truth, we consider it to be correct. In industry, the port managers also require that the whole sequence should be correct; otherwise, they cannot get the correct unique identification for the shipping container even when only one digit is wrong. The precision, recall, f1-score, AP, and recognition accuracy is calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1 \ldots, 1\}} p_{\text{interp}}(r) \tag{19}$$

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{20}$$

where TP is true positive, TN is true negative, and FN is false negative. N_{correct} is the number of correctly predicted sequences and N_{total} is the number of total testing images.

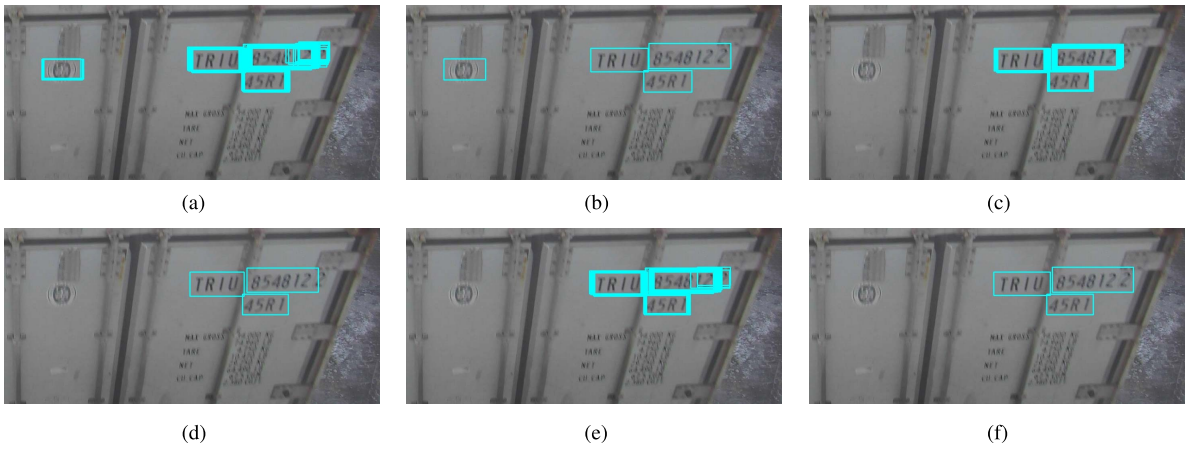


Fig. 11. Localization outputs by using different threshold methods on the score map. (a), (c), and (e) Proposed positions of shipping container codes. (b), (d), and (f) Maximum detection boundaries after the AMSR process. In (a) and (b), the threshold for the score map is set to 0.01. Some nontext areas are included. In (c) and (d), the threshold is set to be 0.95 and the last check digit is not fully covered by the bounding boxes. In (e) and (f), the score map is cleaned by our ASA. The noisy background bounding box is removed. (a) Bounding boxes before merging when the threshold is set fixed with 0.01. (b) Predicted maximum boundaries after merging when the threshold is set fixed with 0.01. (c) Bounding boxes before merging when the threshold is set fixed with 0.99. (d) Predicted maximum boundaries after merging when the threshold is set fixed with 0.99. (e) Predicted bounding boxes before merging when the proposed ASA is used. (f) Predicted maximum boundaries after merging when the proposed ASA is used.

The interpolated precision $p_{\text{interp}}(r)$ is calculated by

$$p_{\text{interp}}(r) = \max_{\hat{r} \geq r}(p(\hat{r})) \tag{21}$$

where $p(\hat{r})$ is the measured precision at recall $\hat{r}$.

C. Model Optimization

1) Optimization of Threshold Methods for Score Map Cleaning: Score map gives the confidences, or possibilities, of the code presence in the corresponding positions in the map. Some false positive predictions have high confidences because they have similar features with the ground-truth shipping container code. Although using threshold can remove the low confidence predictions, they cannot discern high confidence distractions. As shown in Fig. 11, if we clean the score map with a fixed threshold, the low threshold method would give some false positive predictions on the background, whereas the high threshold method might not include all the code areas.

In order to validate the performance of our proposed ASA method on the score map, fixed thresholds in different values are set in comparative experiments, as shown in Table III. In the fixed threshold method, scores, or confidences, which are above the fixed threshold are selected and others are removed from the score map. Our proposed ASA method only selects the set of scores in a specific area with the highest aggregation score after the adaptive threshold. Our ASA is followed by AMSR, which outputs the minimum-to-maximum area range of codes. The performance of threshold methods should be evaluated after merging overlapped bounding boxes since threshold methods influence the location predictions. Therefore, the merged bounding boxes are compared with the ground-truth bounding boxes by calculating the precision, recall, and f1-score. If the IoU of the predicted area and the ground truth is not less than the IoU threshold, we consider it to be true positive. As shown in Table III, our proposed ASA method achieves the highest precision, f1-score, and AP.

TABLE III
COMPARISON OF DIFFERENT THRESHOLD METHODS ON SCORE MAP

Threshod Method	Precision	Recall	F1-score	AP
0.1	0.9417	0.9397	0.9406	0.8919
0.2	0.9433	0.9401	0.9417	0.8923
0.3	0.9467	**0.9414**	0.9440	0.8929
0.4	0.9462	0.9405	0.9433	0.8923
0.5	0.9453	0.9369	0.9425	0.8915
0.6	0.9457	0.9369	0.9427	0.8915
0.7	0.9461	0.9369	0.9429	0.8915
0.8	0.9474	0.9397	0.9435	0.8915
0.9	**0.9482**	0.9401	**0.9442**	0.8920
Our ASA	**0.9482**	0.9401	**0.9442**	**0.8952**

Although the recall of our method is lower, the performance is very close.

As the visualization of score map in Fig. 5 and ranked confidence values in Fig. 6, the confidence levels are mostly in two ranges, i.e., a relatively high range, which is above 0.8, and a relatively low range, which is below 0.2. As shown in Table III, the highest precision is obtained by setting a threshold with 0.9, whereas the best recall is achieved with threshold 0.3. Empirically, most regions with confidence below 0.2 are noise or background, and regions with confidence above 0.8 are the text. We set the threshold adaptive between 0.2 and 0.8 for every case. Therefore, different inputs might get different thresholds. Our adaptive threshold maintains a part of the text proposals with confidence between 0.2 and 0.8. After merging the overlapped proposals, the code regions get higher confidence in our method, which contributes to the higher AP and benefits the code recognition process.

2) Optimization of Strategies for Merging Overlapped Bounding Boxes: The way how the overlapped bounding boxes are merged would affect the recognition model. If the merging method includes a larger area, there might be some

TABLE IV

RECOGNITION PERFORMANCE COMPARISON OF DIFFERENT MERGING METHOD FOR OVERLAPPED BOUNDING BOXES

Merging Methods	Overall Recognition Accuracy
Standard NMS	0.8567
Weighted merge	0.8950
Maximum merge	0.9067
Mean of weighted and maximum merge	0.9133
Our AMSR	**0.9333**

Fig. 12. Two examples of AMSR outputs. The red bounding boxes are the maximum boundaries and the green bounding boxes indicate the minimum ones. (a) Maximum boundaries set can give a better localization because minimum boundaries cannot cover the whole number area, while (b) minimum boundaries set is more suitable for recognition since maximum boundaries include more noisy background. The rod in (b) might be recognized as number 1 if maximum boundaries are used.

noisy text or background. If the merging method only gives a very limited area, some parts of shipping container codes might not be covered. Therefore, we use final recognition accuracy to compare our proposed AMSR method to the other three merging methods, as shown in Table IV. Maximum merge is to find the union of the overlapped bounding boxes. These merging methods are applied in the postprocessing steps of the code localization part, followed by the same trained CRNN recognition model.

The difference of our AMSR method and previous methods lies in that previous localization methods only give fixed bounding boxes, while the AMSR gives the maximum and minimum boundaries of the text area, as shown in Fig. 12. Therefore, the predicted bounding boxes of codes in our adaptive framework can be adjusted inside this range. CRNN can traverse the possible bounding boxes until it gets the expected recognition results. Experimental results show that our proposed AMSR with adjusting strategy achieves higher final recognition accuracy. AMSR can provide better localization outputs to the CRNN-based recognition model in our adaptive framework.

D. Comparison With the State-of-the-Art Solutions

For comparison, PSENet [17] and EAST [16] are trained separately by the same training data. CRNN and GRCNN [20] are used in the code recognition in cropped areas. MaskTextSpotter [21], an end-to-end trained text localization and recognition model, is also trained on our data set for comparison. Experimental results in Fig. 13 and Tables V and VI show that our proposed adaptive deep learning framework achieves the best localization performance and highest recognition accuracy.

Fig. 13 compares the localization performance when the IoU threshold is set with different values in the evaluation

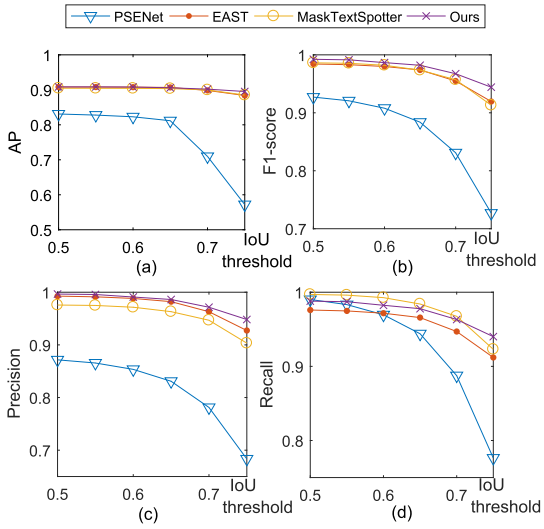


Fig. 13. Comparison of localization performance when IoU threshold is set with different values. AP stands for average precision. (a)–(d) Performance of AP, f1-score, precision, and recall.

TABLE V

LOCALIZATION PERFORMANCE WHEN IOU THRESHOLD IS SET TO 0.75

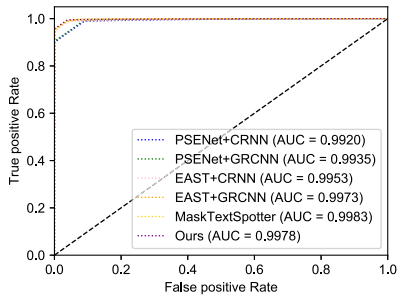
Methods	Localization Performance			
	Precision	Recall	F1-score	AP
PSENet	0.6833	0.7763	0.7269	0.5720
EAST	0.9274	0.9119	0.9196	0.8838
MaskTextSpotter	0.9033	0.9230	0.9131	0.8850
Ours	**0.9482**	**0.9401**	**0.9442**	**0.8952**

TABLE VI

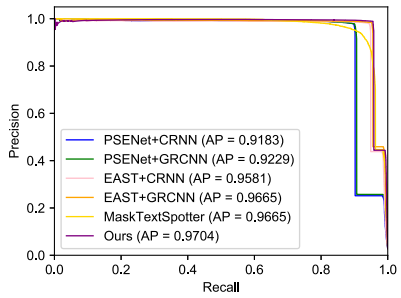
RECOGNITION PERFORMANCE AND PROCESSING SPEED

Methods	Recognition Accuracy	FPS
PSENet + GRCNN	0.7233	2.16
PSENet + CRNN	0.7283	2.21
EAST + GRCNN	0.8033	2.78
EAST + CRNN	0.7683	**3.01**
MaskTextSpotter	0.9017	2.40
Ours	**0.9333**	1.13

process. When the IoU threshold is set to 0.5, more predictions can be considered as true positives. When this threshold increases, the evaluation metrics have higher requirements for the localization. However, when the IoU threshold is higher than 0.75, it would be too strict for the localization. Therefore, we evaluated the performance with IoU threshold setting with values between 0.5 and 0.75. When the IoU threshold is set to 0.5, EAST, MaskTextSpotter, and our framework have a similar performance in AP, f1-score, precision, and recall. Performance of PSENet decreases more than other models, which means that it tends to have larger, smaller, or slightly deviated predictions. When the IoU threshold increases, our framework's performance degrades less than other models since our predictions have more overlapping with the ground truth. This also benefits the code recognition process. Having predicted better code location, our framework archives the higher recognition accuracy.



(a)



(b)

Fig. 14. (a) ROC curve and (b) precision–recall curve in individual character classification using different methods. AUC and AP values are calculated.

The AUC and AP for individual character classification of different methods have been compared in Fig. 14. Our method has the best AP and comparable AUC compared with other methods. The AUC values for individual character classification using different methods are all above 0.99. The high AUC in multiclass classification is led by the low false-positive rate, which is the result of high true negatives during the calculation of AUC. In our character classification process, the number of total classes is 36, including ten numbers and 26 uppercase letters. For an individual character, the expected outputs, i.e., the true condition, are one condition positive and rest 35 conditions negative. The only one condition positive corresponds to the real character label. The final false positive rate of multiclass classification is the average of false-positive rates of all classes. In the prediction process, the classifiers with high discriminability assign one class with high possibility and rest 35 with low confidences, leading to low false positive rate and high AUC. The performance with AUC is not differentiable in this scenario. Therefore, the AP is calculated to eliminate the effects of low false-positive rate. In our experiments, the performance is compared in both metrics. Noticeably, the differences of AUC and AP values for single character classification are not as large as the differences of code recognition accuracy using different methods. The reason is that ports consider that the code recognition is successful only if all characters in the whole code sequence of a container are correctly recognized.

E. Processing Speed of Code Recognition at Ports

The processing speed must be considered when the method is deployed in ports. In deep learning models, the computational complexity of a fully connected layer is $O(n_{i-1}n_i)$, where n_i is the number of nodes in the ith layer. The computational complexity of a convolutional layer is $O(twhf)$, where t is the total number of images, w, h are the width and height of the input image, respectively, and f is the number of feature maps in this layer. When the deep learning model is built, the number of feature maps f in every convolutional layer and the neuron nodes in fully connected layers, as well as the number of layers, are fixed. Different layers are stacked to form the deep learning architecture and the data are processed layer by layer. Therefore, the computational complexity of the deep CNNs in the feedforward process is $O(twh)$. All the methods in Table VI are with the complexity of $O(twh)$. Consequently, the processing speed, FPS, is more generally used to measure the computational complexity for deep learning methods, as shown in Table VI.

The processing speeds on testing data have been compared in Table VI. All the experiments were conducted on a single computer with a single GeForce GTX 1070 8-GB GPU and an Intel Core i7-8750H CPU at 2.20 GHz.

When the ports' regulations about trucks are considered, our method satisfies the requirements of processing speed. The size of the semitrailer truck that loads the shipping container should have the minimum length of 9.25 m [33]. The recommended safe distance between trucks is 6 m [34]. The speed limit is 10 km/h when the truck passes the gates in ports [35]. Hence, the minimum time of one truck passing the gate can be estimated as 5.49 s. On the other hand, the processing speed of our framework is 1.13 FPS, i.e., 0.88 s/frame. When the truck arrives at the gate, the camera on the gate will be triggered to take photos or record videos. If five frames are sampled during the truck's passing, our processing time would be 4.42 s, which is less than the time that an individual truck spends on passing the gate. Our method gains the practical value by improving the recognition accuracy under the condition of satisfying the need for processing speed in the industry.

IV. Conclusion

This article proposes an adaptive deep learning framework for shipping container code localization and recognition. The deep neural network for localization is constructed based on ResNet and U-net. The code recognition model is based on CRNN. In the code localization process, the proposed adaptive score aggregation algorithm selects aggregated areas with high confidences. The noisy text, such as license plates on the truck, and weight information on the container panel, are effectively removed. Moreover, the designed AMSR algorithm provides the maximum and minimum boundaries for code areas. In the proposed framework, the predicted code locations can be adjusted within these boundaries to find the most suitable positions for the recognition operation, gaining higher chances to perfectly match the code recognition model. Experimental results show that the proposed adaptive framework achieves better accuracy than the state-of-the-art solutions for code localization and recognition. The computational speed of our framework does meet the operational requirement at ports, where there is not a critical need for real-time processing. Thus, the proposed framework will empower the digitalization

of the logistics for shipping containers and contribute to global transportation and trading.

REFERENCES

[1] M. Wegener and E. Schnieder, "A measurement standard for vehicle localization and its ISO-compliant measurement uncertainty evaluation," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 11, pp. 3003–3013, Nov. 2012.

[2] Q. Xu, X. Li, and C.-Y. Chan, "Enhancing localization accuracy of MEMS-INS/GPS/In-Vehicle sensors integration during GPS outages," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 8, pp. 1966–1978, Aug. 2018.

[3] H. Liu, S. Li, and L. Fang, "Robust object tracking based on principal component analysis and local sparse representation," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 11, pp. 2863–2875, Nov. 2015.

[4] G. A. Borges, A. M. N. Lima, and G. S. Deep, "Characterization of a trajectory recognition optical sensor for an automated guided vehicle," *IEEE Trans. Instrum. Meas.*, vol. 49, no. 4, pp. 813–819, Aug. 2000.

[5] C. Barrios and Y. Motai, "Improving estimation of vehicle's trajectory using the latest global positioning system with Kalman filtering," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 12, pp. 3747–3755, Dec. 2011.

[6] *Freight Containers–Coding, Identification and Marking*. Document ISO 6346:1995, Accessed: Feb. 20, 2020. [Online]. Available: https://www.iso.org/standard/20453.html

[7] Z. Liu, H. Ukida, K. Niel, and P. Ramuhalli, Eds., *Integrated Imaging and Vision Techniques for Industrial Inspection*. London, U.K.: Springer, 2015.

[8] M. Goccia, M. Bruzzo, C. Scagliola, and S. Dellepiane, "Recognition of container code characters through gray-level feature extraction and gradient-based classifier optimization," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, Edinburgh, U.K., Aug. 2003, pp. 973–977.

[9] S. Kumano, K. Miyamoto, M. Tamagawa, H. Ikeda, and K. Kan, "Development of a container identification mark recognition system," *Electron. Commun. Jpn. (Part II, Electron.)*, vol. 87, no. 12, pp. 38–50, 2004.

[10] W. Al-Khawand, S. Kadry, R. Bozzo, and S. Khaled, "8-neighborhood variant for a better container code extraction and recognition," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 4, pp. 182–186, Apr. 2016.

[11] T. Szabo and G. Horvath, "Finite word length computational effects of the principal component analysis networks," *IEEE Trans. Instrum. Meas.*, vol. 47, no. 5, pp. 1218–1222, Oct. 1998.

[12] K. Kim, Y. W. Woo, and H. Yang, "An intelligent system for container image recognition using ART2-based self-organizing supervised learning algorithm," in *Proc. 6th Asia–Pacific Conf. Simulated Evol. Learn.*, Hefei, China, Oct. 2006, pp. 897–904.

[13] Z. He, J. Liu, H. Ma, and P. Li, "A new automatic extraction method of container identity codes," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 1, pp. 72–78, Mar. 2005.

[14] X. Zhou *et al.*, "Automated visual inspection of glass bottle bottom with saliency detection and template matching," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 11, pp. 4253–4267, Nov. 2019.

[15] W. Wu, Z. Liu, M. Chen, X. Yang, and X. He, "An automated vision system for container-code recognition," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2842–2855, Feb. 2012.

[16] X. Zhou *et al.*, "EAST: An efficient and accurate scene text detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2642–2651.

[17] W. Wang *et al.*, "Shape robust text detection with progressive scale expansion network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 9336–9345.

[18] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 936–944.

[19] B. Shi, 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 Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017.

[20] J. Wang and X. Hu, "Gated recurrent convolution neural network for OCR," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 335–344.

[21] 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.*, vol. 11218. Munich, Germany, Sep. 2018, pp. 71–88.

[22] L. Mei, J. Guo, Q. Liu, and P. Lu, "A novel framework for container code-character recognition based on deep learning and template matching," in *Proc. Int. Conf. Ind. Informat.-Comput. Technol., Intell. Technol., Ind. Inf. Integr. (ICIICII)*, Wuhan, Hubei, China, Dec. 2016, pp. 78–82.

[23] A. Verma, M. Sharma, R. Hebbalaguppe, E. Hassan, and L. Vig, "Automatic container code recognition via spatial transformer networks and connected component region proposals," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Anaheim, CA, USA, Dec. 2016, pp. 728–733.

[24] C. Roeksukrungrueang, T. Kusonthammrat, N. Kunapronsujarit, T. N. Aruwong, and S. Chivapreecha, "An implementation of automatic container number recognition system," in *Proc. Int. Workshop Adv. Image Technol. (IWAIT)*, Chiang Mai, Thailand, Jan. 2018, pp. 1–4.

[25] Y. Liu, T. Li, L. Jiang, and X. Liang, "Container-code recognition system based on computer vision and deep neural networks," in *Proc. 2nd Int. Conf. Adv. Mater., Mach., Electron.*, Xi'an, China, Jan. 20-21, 2018.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Munich, Germany, Oct. 2015, pp. 234–241.

[28] H. Gao, W. Xu, J. Sun, and Y. Tang, "Multilevel thresholding for image segmentation through an improved quantum-behaved particle swarm algorithm," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 4, pp. 934–946, Apr. 2010.

[29] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, Jun. 2006, pp. 369–376.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, San Diego, CA, USA, May 2015, pp. 1–41.

[32] M. Everingham, L. Van Gool, C. K. I. Williams, J. W8nn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[33] Transportation Association of Canada and Interjurisdictional Committee on Vehicle Weights and Dimension, *Heavy Truck Weight Dimension Regulations for Interprovincial Operations Canada: Resulting From Federal-Provincial-Territorial Memorandum Understand. Interprovincial Weights Dimensions*. Transportation Association of Canada, Ottawa, ON, Canada, 1993, p. 13.

[34] International Maritime Organization and Maritime Safety Committee, *Revised Recommendations on the Safe Transport of Dangerous Cargoes and Related Activities in Port Areas*, 3rd ed. London, U.K.: IMO, 2007, p. 44.

[35] Y. Port. *Yantian International Container Terminals–Port Facilities–Infrastructure–Port Traffic*. Accessed: Jun. 19, 2020. [Online]. Available: https://www.yict.com.cn/page/port-traffic.html?locale=en_US

Ran Zhang received the bachelor's degree in information engineering from Xidian University, Xi'an, China, in 2015, and the master's degree in computer software and theory from Capital Normal University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree in electrical engineering with the School of Engineering, The University of British Columbia Okanagan Campus, Kelowna, BC, Canada.

His research interests include computer vision, pattern recognition, and intelligent fault diagnosis.

Zhila Bahrami received the bachelor's degree in information technology and the master's degree in artificial intelligence from Kurdistan University, Kurdistan, Iran, in 2012 and 2015, respectively. She is currently pursuing the Ph.D. degree in electrical engineering with the School of Engineering, The University of British Columbia Okanagan Campus, Kelowna, BC, Canada.

Her research interests include computer vision, pattern recognition, and image processing.

Teng Wang (Student Member, IEEE) received the bachelor's degree in mechanical engineering from the Hefei University of Technology, Hefei, China, in 2016, and the master's degree in mechanical engineering from Shandong University, Jinan, China, in 2019. He is currently pursuing the Ph.D. degree in electrical engineering with the School of Engineering, The University of British Columbia Okanagan Campus, Kelowna, BC, Canada.

His research interests include machinery condition monitoring, intelligent fault diagnostics, remaining useful life prediction of machinery, and pattern recognition.

Zheng Liu (Senior Member, IEEE) received the Ph.D. degree in engineering from Kyoto University, Kyoto, Japan, in 2000, and the Ph.D. degree from the University of Ottawa, Ottawa, ON, Canada, in 2007.

From 2000 to 2001, he was a Research Fellow with Nanyang Technological University, Singapore. He then joined the Institute for Aerospace Research (IAR), National Research Council Canada, Ottawa, as a Governmental Laboratory Visiting Fellow nominated by the Natural Sciences and Engineering Research Council. After being with IAR for five years, he transferred to the National Research Council Institute for Research in Construction, where he was a Research Officer. From 2012 to 2015, he was a Full Professor with the Toyota Technological Institute, Nagoya, Japan. He is currently with the School of Engineering, The University of British Columbia Okanagan Campus, Kelowna, BC, Canada. His research interests include image/data fusion, computer vision, pattern recognition, senor/sensor networks, condition-based maintenance, and nondestructive inspection and evaluation.

Dr. Liu is a Senior Member of SPIE. He is the Co-Chair of the IEEE Instrumentation and Measurement Society Technical Committee on Nondestructive Evaluation (TC-1). He holds a Professional Engineer License in British Columbia and Ontario. He serves on the Editorial Board of peer-reviewed journals, including IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, IEEE TRANSACTIONS ON MECHATRONICS, IEEE JOURNAL OF RADIO FREQUENCY IDENTIFICATION, *Information Fusion, Machine Vision and Applications*, and *Intelligent Industrial Systems*.