

Digital transformation for shipping container terminals using automated container code recognition

Hoang-Sy Nguyen¹, Cong-Danh Huynh², Nhat-Quan Bui³

¹Becamex Business School, Eastern International University, Binh Duong Province, Vietnam

²Faculty of Economics, Thu Dau Mot University, Thu Dau Mot City, Binh Duong Province, Vietnam

³Centre for Artificial Intelligence Research and Optimisation (AIRO), Torrens University Australia, Adelaide, SA 5000, Australia

Article Info

Article history:

Received Jun 07, 2022

Revised Nov 16, 2022

Accepted Dec 28, 2022

Keywords:

Character isolation

Character recognition

Container codes recognition

Histogram of oriented gradients

Support vector machine

ABSTRACT

Due to the sweeping waves of global industry development, the number of containers passing through terminal ports increases every day. Therefore, it is essential to automate the identification process for the container codes to replace the manual identification for more efficient logistics and safer workplace. This paper aims to design and evaluate the performance of such a system. Specifically, automated container codes recognition (ACCR) has been implemented. This is a novel container tracking model based on image processing algorithms and machine learning (ML) algorithms to be applied in ports. There are three steps in this system: character detection, character isolation, and character recognition. The first step is to identify an area with 10 digits and 26 capitals. After detecting the text area, the second step is to separate the characters. Each character is recognized in the last step by the classification method. In particular, features are extracted with the histogram of oriented gradients (HOG) algorithm and support vector machines (SVMs) for training and prediction. The trained ML model is then used to classify characters and digits according to what it has learned. In general, the digital technologies in logistics and container management in ports will benefit from the proposed algorithms.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Cong-Danh Huynh

Faculty of Economics, Thu Dau Mot University

Thu Dau Mot City, Binh Duong Province, Vietnam

Email: danhhc@tdmu.edu.vn

1. INTRODUCTION

In recent years, the global supply chain has suffered from the impact of coronavirus disease of 2019 (COVID-19) due to strict lockdowns in many countries around the world, leading to the closure of many companies and factories. Most companies must adapt to this pandemic situation by equipping themselves with transformative technologies that could help them maintain their production lines [1], [2]. Specifically, tracking and tracing (T&T) systems have been implemented, i.e., the usage of barcodes and radio frequency identification (RFID) tags [3]. In the supply chains of vital medical products, RFID tags have been utilized to track and authenticate plasma, test kits, vaccines, and personal protective equipment (PPE) in [4], it was proven that the ability to record and exchange information in real time is of the utmost importance for supply chains in terms of collaboration and the ability to cope with and recover from disruptions. In the international logistic system, containers have undeniably become one of the most important assets for freight transport. The tracking and management of containers, thus, are also essential since the containers are shipped globally and frequently switched between different shipping vehicles. Port terminals are accustomed to making manual records on container codes. In fact, this requires high labor costs and leads to human errors due to fatigue from repetitive

work. In view of this, automatic container code recognition (ACCR) systems are deployed to automate the recording process at ports. All ACCR systems can be implemented relatively easily, by installing managing software and camera(s) at the port terminals. The system differs from another on the basis of the computational methods that are used to obtain information based on images [5]. A typical ACCR system is programmed with two steps, i.e., code localization followed by recognition. Traditional localization methods extract information from text areas by analyzing stroke features, gray level, edges, contours, and histogram information obtained after processing grayscale images [6]. Additionally, other features such as filters [7] and masks [8] can be utilized for the same purpose. After the codes are localized, the recognition part follows. To fully and accurately recognize container codes, characters are first isolated so that individual characters can be recognized. From previous studies of automatic license plate recognition (ALPR) can be seen that even if the recognition algorithm can handle different character orientations, sizes, and fonts, the results are still incorrect if the segmentation is set improperly [9], [10]. In addition, the segmentation process is affected by blurriness, noise, uneven illumination, and shadows. A typical ALPR uses optical character recognition (OCR) techniques in which characters are segmented from the license plate. On another hand, OCR is employed to recognize characters and digits in a situation where segmentation is difficult, i.e., multi-character reading. From another perspective, convolutional neural networks (CNNs) show their strength in dealing with multi-character contexts without the need to employ segmentation, especially in unconstrained images [11]–[14]. Algorithms used for ALPR are the templating matching method or machine learning (ML) [15], [16]. The ML methods are more robust, as they use more stable features, e.g., image density [17] or direction features [18]. Among the most common techniques used in research and practice are support vector machine (SVM) [19] and probabilistic neural network (PNN) [20]. In addition, in the context of scene text detection, it is worth mentioning that deep learning (DL) methods, e.g., convolutional repetitive neural networks (CRNN) have been exclusively developed [21], [22].

The role of digitalization in the container shipping supply chain (CSSC) was intensively discussed in Song [23]. In view of this, state-of-the-art technology such as DL has the potentials to improve all the major segments in CSSC, i.e., vessel/freight/container logistics. For the application of DL in container logistics, Li and He [24] has combined the DL with computational logistics to a so-called container terminal-oriented neural-physical fusion computation (CTO-NPFC), which is used to analyze the performance of the container terminal handling system (CTHS) at ports. In addition, Zhang *et al.* [25] introduced a highly accurate approach to localize and recognize the codes. Specifically, the authors deployed adaptive score aggregation (ASA) algorithm to remove the text regions with noises. The boundaries of the code regions were then identified using average-to-maximum suppression range (AMSR) algorithm. The proposed approach can proceed at 1.13 FPS and has the accuracy of 93.33%. Liu *et al.* [26] further improved the accuracy of the localization by a real-time ML system to predict the texts and their boundaries, consequently fuse the results to improve the segmentation accuracy. Their system achieved a F-measure result of 96.5% at 70 FPS while performed on the on-field datasets.

In short, the repeated manual identification process at terminal ports has hindered us from moving forward to a more efficient logistics system. This is due to the fact that manual check requires extra labour cost and is time-consuming. Besides, conducting the same job everyday would likely bring up human errors, and direct contact with containers from all over the world could put the port workers to health risks, e.g., from the current COVID-19 pandemic. Hence, automating this process is undeniably a necessity. Based on the reviewed literature and the above discussion, we present herein a container code recognition system, which includes three models, i.e., segmentation, isolation, and character prediction. The recognition model and the insights of the techniques are introduced and explained in detail. The case study is conducted in a port in Ho Chi Minh City, Vietnam, and the result is highly satisfying. In addition to the introduction, section 2 discusses the method that we employed. Section 3 shows the results of the proposed model compared to other similar models. The study is concluded in section 4.

2. METHOD

The container code recognizing system herein this study is equipped with three primary modules, namely segmentation, isolation, and character prediction. Specifically, at first, areas with texts from given grayscale images are detected (segmentation). The lines of texts are then separated to forward to the next step (isolation). Finally, SVMs are deployed to determine isolated characters. It should be noted that, for each step, appropriate algorithms must be chosen so that the overall performance can be optimized. The flowchart of the recognition system is shown in Figure 1. The all the modules are discussed in detail in Figure 1.

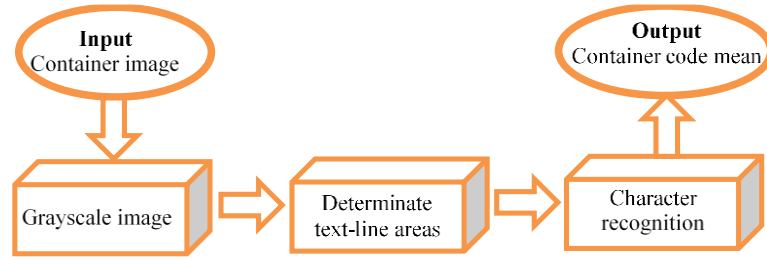


Figure 1. The recognition system's flowchart

2.1. Text area determination

For the first module, the text region location method is deployed aiming at separating the text line areas from the background. The grayscale images as inputs are usually poorly lit, contain reflection, and defects, which would downgrade the accuracy and efficiency of the segmenting process significantly. Thus, the input images are pre-processed with seven steps, presented in detail from section 2.1.1 to section 2.1.6. From Figure 2 to Figure 8, the image resulted after each step can be observed. From the original image in Figure 2, Figure 8 is produced as a final image that is ready for the separation and recognition of the characters.

2.1.1. Grayscale

First of all, a red, green and blue (RGB) image shows the back of the container being captured and shrunk. The center area of the image is set to the container area and grayscaled using the (1).

$$L = 0.299R + 0.587G + 0.114B \quad (1)$$

Where R, G, B respectively denotes the red, green, and blue color channels of the original image.

2.1.2. Gaussian blur

The grayscale image contains, yet, noises stemming from different illuminating directions, lighting conditions, the equipment that is used to capture the image itself. These defects are blurred to reduce noise and details with the Gaussian function with standard distribution in the 2D space (x, y coordinates) [27] (2).

$$G_{0(x,y)} = a \exp\left(\frac{-(x-b_x)^2}{2c_x^2} + \frac{-(y-b_y)^2}{2c_y^2}\right) \quad (2)$$

Where a is height of the curve's peak, $b_{x,y}$ is the mean (the peak), and $c_{x,y}^2$ is the variance of x and y variables.

2.1.3. Morphological gradient

Morphological gradient shows how the dilation and the erosion of an image are different. After going through a morphological gradient filter, the blurred image is sharpened. Herein, the morphological gradient was deployed with the help of the kernel (structuring elements) [28] (3).

$$G_1(x) = \begin{cases} 0, & \text{if } |x| \leq 1 \\ -\infty, & \text{otherwise} \end{cases} \quad (3)$$

Accordingly, the morphological gradient function for the grayscale image, $G(f)$, is given.

$$G(f) = f \oplus b - f \ominus b \quad (4)$$

Where \oplus and \ominus are respectively the dilation and the erosion. The symmetric short support is denoted as b .

2.1.4. Otsu's binarization

After calculating the morphological gradient and the gray-level distribution of the input image, we can obtain a gray-leveled graph with two vertices. The first vertex represents the regions with text, and the second represents the background. Otsu's method is then used for image binarization [29]. According to [30], the best adaptive threshold value k^* for the method can be calculated at value d^2 , which is (5).

$$d^2 = \omega_1(m_1 - m_t)^2 + \omega_2(m_2 - m_t)^2 \quad (5)$$

Where m_1 and m_2 stand for the mean value of segmentation 1 and 2 respectively. Coefficients a_1 and a_2 are corresponding frequencies of ω_1 and ω_2 . Given that $m_t = \omega_1 m_1 + \omega_2 m_2$, and $\omega_1 + \omega_2 = 1$, the (5) can be rewritten as (6).

$$d^2 = \omega_1 \omega_2 (m_1 - m_2)^2 \quad (6)$$

The ratio ω_j segmentation $j \in \{1,2\}$ has the total probability of (7).

$$\omega_j = \sum_{i \in c_j} P_i, j = 1,2 \quad (7)$$

Where P_i denotes the quotient for occurrence numbers of gray level I ($I = 256$) for text image.

Therefore, we can calculate the occurrence numbers in total for all the gray levels as:

$$\sum_{i=0}^{I-1} P_i = 1 \quad (8)$$

Where the performs all of points ω_j , with $j \in \{1,2\}$ level of average gray on j segmentation, less than or equal to k threshold. We pre-define a k threshold value for the gray level. If the gray level is higher than k , the pixel under consideration becomes black (value 1). On the other hand, if the gray level is equal to or lower than k , the pixel is turned white (value 0). Remarkably, the formula for calculating average of m_j is (9).

$$m_j = \sum_{i \in \omega_j} i - p_i / a_j \quad (9)$$

Consequently, the best k^* threshold value can be calculated from the vertex of d^2 .

2.1.5. Morphological close

After using Otsu's algorithm, the morphology is applied again in this morphological close step. The morphological close method will fill the small gaps, remove noise and smooth the contours of objects in the image. After this process, the broken pixels in the image can be repaired for better image quality. It should be noted that for real photo inputs, more than one pre-processing technique must be used to achieve efficient quality for the image-processing task.

2.1.6. Finding bounding boxes

We can specify the bounding boxes for the features we are interested in. Nevertheless, as shown in Figure 6, boxes containing no container code can also be created. These irrelevant boxes can be filtered by counting the nonzero pixels in the boxes, assuming that if the boxes isolate text areas, they must be filled with 50% of texts as a minimum.



Figure 2. Original container images



Figure 3. Gaussian blur



Figure 4. Morphological gradient



Figure 5. Otsu's binarization



Figure 6. Morphological close



Figure 7. Finding bounding boxes



Figure 8. Final results

2.2. Separation of characters in bounding boxes with texts

To ensure maximum accuracy in character detection, we continuously assess the positioning of each character. By using projection in both directions (vertical and horizontal), we can make sure that each individual character is captured with maximum precision. This helps to increase the accuracy of the character recognition process and makes it easier to identify the characters quickly.

2.2.1. Character division in text-line areas

Owing to the fact that the characters have rigid edges and are not subjected to poor lighting conditions, we can divide them into a certain number of regions. Consequently, the binary edge image can be generated with the most optimal adaptive threshold value specifically calculated for each image. By evaluating the vertical projection histogram, we can obtain the width and height of the text area. Based on this, we can eliminate the non-character edges if the edges do not meet the standard measurement of the font characters.

2.2.2. Projection method

In this study, the projection method according to [31] is utilized after the segmentation and character splitting have been performed. Using the histogram method obtained from the vertical projection, we can obtain the horizontal limits of a particular character. If there exist two adjacent zones that are close to one another, the two will be merged. In a similar manner, the bottom and top limits of a character can be determined with the help of a horizontal projection. As a result, all individual characters can be detected.

2.3. Character recognition

2.3.1. Feature extraction

The histogram of oriented gradients (HOG) is utilized to extract the desired features from a given image. To build a HOG vector, we have to consider four consecutive steps, that are: (i) gradient calculation; (ii) feature vector calculation on individual cells; (iii) block normalization; and (iv) HOG vector calculation.

Applying the convolution for the vertical and horizontal directions, we can calculate the image gradients. The derivative in the Ox and Oy directions is determined.

$$D_x = [-1 \ 0 \ 1], \text{ and } D_y = [-1 \ 0 \ 1]^T \quad (10)$$

The input image is assumed to be I , and its derivatives in two directions, I_x and I_y , can be calculated.

$$I_x = I \times D_x, \text{ and } I_y = I \times D_y \quad (11)$$

Eventually, the magnitude G and direction θ of the gradient can be computed.

$$G = \sqrt{I_x^2 + I_y^2} \text{ and } \theta = \arctan \frac{I_x}{I_y} \quad (12)$$

Secondly, after the gradient calculation, we can extract the feature vector from the individual cells. The image under consideration is divided into blocks, each contains a predetermined number of cells, and each cell is a collection of pixels. If we are given a 128×128-pixel image, assuming that each cell is sized 4×4 pixels, the image is now composed of 32×32 cells. Given another assumption that each block is sized of 4×4 cells, then the image is composed of 8×8 blocks. In another word, 1 block contains 4 cells or 16 pixels.

Figure 9 illustrates the block and how to process the feature vectors extracted from it. In particular, in each cell (red-outlined square), there are 16 pixels (green-outlined square). From each pixel, a vector with magnitude G and direction θ is calculated. We then classify the vectors calculated from all the pixels into a 9-bin histogram (HOG), as illustrated by the bar chart in Figure 9. After the classification, corresponding to 9 bins, we obtain 9 vectors representing the vectors in each cell. This classification process is continued for the rest of the cells in the block that is under consideration. After processing 16 cells in the block, we can combine 16 cells and multiply it by 9 representative vectors per cell, we can obtain a feature vector sized 144×1 per block.

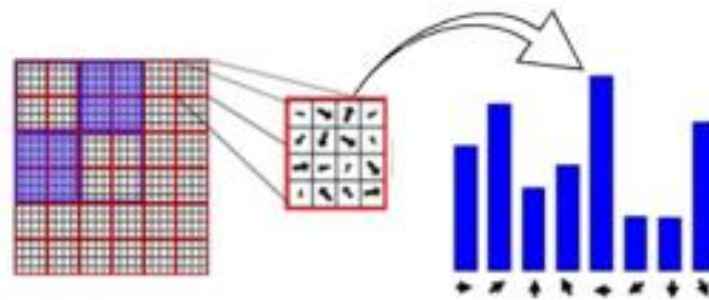


Figure 9. Example of 1 block containing 4×4 (16) cells, which are classified using HOG

As the third step, normalization is carried out on the blocks for better recognition performance. Specifically, based on the local histograms of the blocks, the threshold values for the intensity can be computed and used for cell normalization in the block. This results in a characteristic vector that is invariant to the lighting condition. Normalization can be obtained using the (13).

$$L2 - norm : f = \frac{v}{\sqrt{\|v_2\|^2 + c^2}}, \text{ and } L1 - norm : f = \frac{v}{\|v_1\|^1 + c^2} \tag{13}$$

Where v represents the vector that contains the block diagram after normalization, $\|v\|$ is the normalized value of v , and c constant.

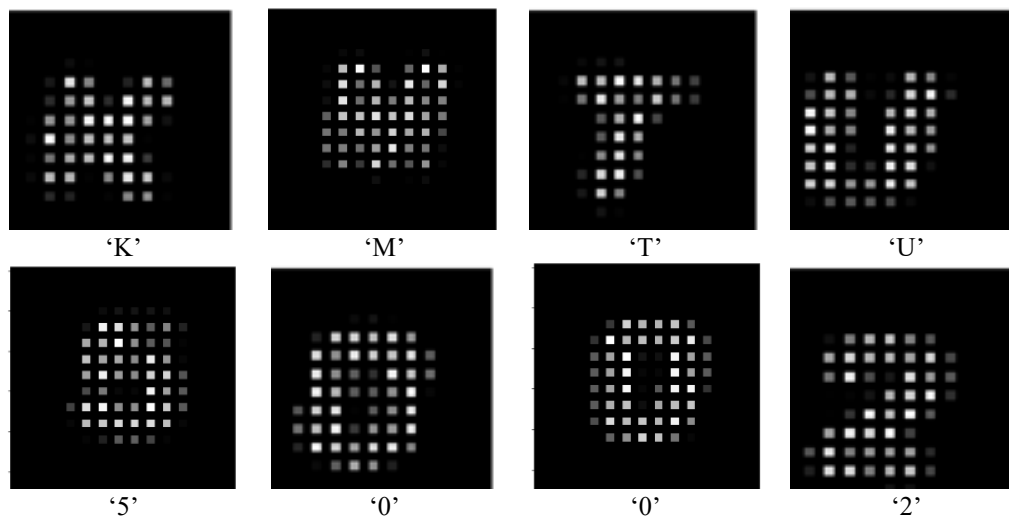


Figure 10. Different characters and digits after HOG application

In this study, HOG is applied to process 28×28-pixel binary images of characters and digits with different fonts. Because the images are small, we assume that a block has the size of a cell for easier processing. Thus, considering that each cell is 4×4 pixels in size, each block is also 4×4 pixels. Thus, we have in each image 7×7 blocks. After voting in the HOG diagram, we obtained one feature vector for each block. The feature vector of each block is normalized with the L2-normalizing method. After this normalization, we will obtain in each block one new feature vector. By combining the 7×7=49 feature vectors for 49 blocks, we receive 1 feature vector (column vector) for the entire image with the size of 49×9=441×1. Each character or digit has 1 distinct feature vector. For each character or digit with different formats, we can extract 1 distinct feature vector. The collection of feature vectors extracted from the characters and digits is used as the input for the SVM algorithm in the next step. Examples of the characters and digits after applying HOG can be seen in Figure 10.

2.3.2. Training and classifying

Datasets are either almost linearly separable or nonlinearly separable. To overcome this problem, kernel methods would seek an appropriate transformation to convert the non-linearly separable data to a new space in which the data becomes linearly separable. This is beneficial for soft margin SVM classification. Assuming that $\phi()$ is the new function we obtain after the transformation to this new space, x data in the old space will become $\phi(x)$. Indeed, $\phi(x)$ must not be calculated explicitly for each and every data point. Instead, we just need to calculate $\phi(x)^T \phi(z)$ for two x and z data points using the kernel trick.

$$k(x, z) = x^T z \quad (14)$$

It should be taken into account that we will create 36 classes in total for training and classification. For the application described herein, linear SVM is the most compatible. In general, SVMs are algorithms used to work with one or a number of hyperplanes in a high or endless dimensional space, making them highly applicable for classifying or, regressing tasks. A SVM classifier is a binary classifier that operates in a supervised manner [32]. The Char74k dataset was used here for model training because it offers a variety of characters formatted in different ways, which would diversify the learning of the features of the characters.

3. RESULTS AND DISCUSSION

3.1. Shipping container code and its meaning

In accordance to the ISO 6346 standard, a standardized container code consists of 11 digits, of which the first three must be in capital letters, indicating the container's owner. The following fourth digit classifies the equipment type and must also be in capital of the latter. The next six digits are the serial number of the container, and they can be validated using the last check digit. Thus, the text-line area must be cropped into four areas containing, respectively, 4, 6, 7, or 1 digit(s) with regard to the localization prediction. The obtained results are eventually combined into a sequence of 11 digits. The predicted sequences are then ranked according to the localization confidence for further evaluation.

3.2. Dataset

The data set herein is generated using various character fonts sizing 28×28 pixels that are binarized to facilitate easier training, testing, and predicting tasks. We employ 1024 images per character and modify them with image augmentation techniques (rotation, shear, reflection, scaling, and blurring) to generate numerous new images to diversify the training data set. The augmentation technique ensures that the characteristics of the characters are not changed and, at the same time, mitigates the overfitting problem. The modification of the characters (bold, italic, and bold, and italic) can be realized by changing its format, as shown in Figure 11.

To assess the system performance, we use more than 400 pictures of containers, sizing up to 800×600 pixels taken from real containers at port for the study. The images were captured under daylight conditions. Moreover, the images are with different contrast and brightness, the containers have different sizes and colors, and thus are located differently on the recording screen.



Figure 11. The modification of the characters

We need to design some criteria for localizing and recognizing tasks. Indeed, container codes can be captured on the back and the top of the containers. However, it should be noted that the codes on the top are poorly lit and not that compact compared to the ones on the back. Thus, we train the code localization model to neglect the codes on the top of the shipping container. In particular, we investigate the precision, recall, f1-score, and average precision (AP) to assess how the code localization model performs. Theoretically, for object detection, AP can be deployed as area under curve (AUC). Accordingly, we can calculate the interpolated precision $p_{interp}(r)$ (15).

$$p_{interp}(r) = \max_{\hat{r} \geq r} (p(\hat{r})) \quad (15)$$

Where $p(\hat{r})$ denotes the precision obtained from the measurement at the recall \hat{r} .

Then, the 11-point interpolated AP according to [29] is deployed. The system is considered to perform correctly if and only if the 11-digit code sequence it predicts coincides with the real codes on the field. A misprediction of even 1 digit is unacceptable, since the unique identification of a shipping container can be obtained using the exact 11 digits.

In addition, $Recall = TP/(TP + FN)$, $Precision = TP/(TP + FP)$, $AP = \frac{1}{11} \sum_{r \in \{0,0.1,\dots,1\}} p_{interp}(r)$, $f1\text{-score} = 2(Precision \times Recall)/(Precision + Recall)$, and $recognition\ accuracy(\%) = N_{correct}/N_{total}$, in which TP , TN , and FN abbreviate respectively true positive, true negative, and false negative. $N_{correct}$ denotes the number of correct predictions over the total prediction, N_{total} . The accuracy of the three consecutive modules of the recognizing model herein is listed in Table 1.

It can be observed in Table 1 that the accuracy of three modules in the proposed system is relatively high. Furthermore, to evaluate the model performance, we created a score map using the fixed threshold method presented in Table 2. In the score map, the confidences (scores) above some predetermined (fixed) threshold are kept, and the rest is removed. It should be noted that the threshold methods would affect the location prediction; thus, we need to first merge the bounding boxes which are overlapping. Then, the evaluation can be carried out. Because of this, we need to evaluate the precision, recall, and f1-score to facilitate the comparison of the merged bounding box with the ground truth bounding boxes.

As shown in Table 2, the threshold values for precision and recall are respectively 0.9 and 0.3. The regions with a confidence level above 0.8 are text while the regions below 0.2 are background or noise. Subsequently, the adaptive threshold value is set between 0.2 and 0.8. The proposed adaptive threshold can maintain the text proposals with a confidence level in the range of 0.2 to 0.8. As we merge the overlapped proposals, we can obtain a higher confidence, thus, a higher AP value, which is highly beneficial for the code recognition model.

Table 1. Execution assessment of the proposed strategy

Module	Accuracy (%)
Determine text-line areas	89.58
Isolation	90.24
Characters recognition	90.45
Overall	89.35

Table 2. Comparison of the threshold methods

Threshold method	Recall	Precision	AP	F1-score
0.1	0.9351	0.9421	0.8929	0.9405
0.2	0.9379	0.9443	0.8933	0.9407
0.3	0.9415	0.9476	0.8929	0.9420
0.4	0.9305	0.9466	0.8933	0.9413
0.5	0.9469	0.9435	0.8905	0.9420
0.6	0.9296	0.9447	0.8915	0.9417
0.7	0.9396	0.9416	0.8925	0.9419
0.8	0.9359	0.9447	0.8913	0.9437
0.9	0.9301	0.9481	0.8925	0.9442

4. CONCLUSION

In the context that logistic systems have suffered severe disruption due to the COVID-19 pandemic, the application of an automatic container code recognition system is highly beneficial. First, the proposed framework can deal with a variety of container types and colors, as well as different types of image defects caused by different lighting conditions at the port. Due to this, human contacts at ports subject to COVID-19 contagion, labor costs, and human-made errors can be reduced. Secondly, the model learns from the training data with a varied character font, so it can deliver accurate results (up to 90%) when predicting the 11-digit sequences in real-time.

The results of this study have connected research and practice whose approach is relatively simple, however, with highly accurate results. By working with the on-field datasets from ports, it shows the potentials to be widely applied to many other ports that would like to automate their code identification process. From the researcher's point of view, we can improve the accuracy of the model by using state-of-the-art DL techniques in the recent and upcoming year to the three modules of the proposed model. Future research on this topic can focus on capturing the images of containers while they are moving, under different lighting conditions (rainy days, nighttime), from different cameras and with varied angles. The size of the datasets can also be increased so that the model has more data to learn from to improve its predicting accuracy. By and large, automation of code recognition in the port is just a small part of the smart logistics, whose back-bone technologies can be applied to the development of smart cities in the near future.




REFERENCES

- [1] G. Büyükköçkan and F. Göçer, "Digital Supply Chain: Literature review and a proposed framework for future research," *Computers in Industry*, vol. 97, pp. 157-177, 2018, doi: 10.1016/j.compind.2018.02.010.
- [2] M. Anwar, L. Henesey, and E. Casalicchio, "Digitalization in Container Terminal Logistics : A Literature Review," in *Proc. of 27th Annual Conference of International Association of Maritime Economists (IAME)*, 2019, vol. 141, pp. 1-27. [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:1337559/FULLTEXT01.pdf>
- [3] D. Ivanov, A. Dolgui, and B. Sokolov, "The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics," *International Journal of Production Research*, vol. 57, no. 3, pp. 829-846, 2019, doi: 10.1080/00207543.2018.1488086.
- [4] Y. Sheffi and J. B. Rice Jr, "A supply chain view of the resilient enterprise," *MIT Sloan management review*, vol. 47, no. 1, pp. 41-48, 2005. [Online]. Available: <https://sloanreview.mit.edu/article/a-supply-chain-view-of-the-resilient-enterprise/>
- [5] Z. Liu, H. Ukida, P. Ramuhalli, and K. Niel, *Integrated Imaging and Vision Techniques for Industrial Inspection*. London: Springer, 2015, doi: 10.1007/978-1-4471-6741-9.
- [6] M. Goccia, M. Bruzzo, C. Scagliola and S. Dellepiane, "Recognition of container code characters through gray-level feature extraction and gradient-based classifier optimization," *Seventh International Conference on Document Analysis and Recognition, 2003 Proceedings*, 2003, pp. 973-977, doi: 10.1109/ICDAR.2003.1227804.
- [7] S. Kumano, K. Miyamoto, M. Tamagawa, H. Ikeda, and K. Kan, "Development of a container identification mark recognition system," *Electronics and Communications in Japan (Part II: Electronics)*, vol. 87, no. 12, pp. 38-50, 2004, doi: 10.1002/ecjb.20134.
- [8] W. Al-Khawand, S. Kadry, R. Bozzo, and K. Smaili, "8-neighborhoodvariant for a better container code extraction and recognition," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, pp. 182-186, 2016. [Online]. Available: https://www.researchgate.net/profile/Seifedine-Kadry/publication/305658135_8-neighborhood_variant_for_a_better_Container_Code_Extraction_and_Recognition/links/579889d908acc89db7bb4b9b/8-neighborhood-variant-for-a-better-Container-Code-Extraction-and-Recognition.pdf
- [9] T. K. Cheang, Y. S. Chong, and Y. H. Tay, "Segmentation-free vehicle license plate recognition using ConvNet-RNN," *arXiv preprint*, 2017, doi: 10.48550/arXiv.1701.06439.
- [10] L. Zheng, X. He, B. Samali, and L. T. Yang, "Accuracy enhancement for license plate recognition," in *2010 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 511-516, doi: 10.1109/CIT.2010.111.
- [11] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnaud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint*, 2013, doi: 10.48550/arXiv.1312.6082.
- [12] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, pp. 1-20, 2016, doi: 10.1007/s11263-015-0823-z.
- [13] V. Jain, Z. Sasindran, A. Rajagopal, S. Biswas, H. S. Bharadwaj, and K. R. Ramakrishnan, "Deep automatic license plate recognition system," in *Proc. of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2016, pp. 1-8, doi: 10.1145/3009977.3010052.
- [14] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and LSTMs," *arXiv preprint*, 2016, doi: 10.48550/arXiv.1601.05610.
- [15] G. -S. Hsu, J. -C. Chen, and Y. -Z. Chung, "Application-oriented license plate recognition," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 2, pp. 552-561, 2013, doi: 10.1109/TVT.2012.2226218.
- [16] S. Rasheed, A. Naeem, and O. Ishaq, "Automated numberplate recognition using Hough lines and template matching," in *Proc. of the World Congress on Engineering and Computer Science*, 2012, vol. 1, [Online]. Available: https://www.iaeng.org/publication/WCECS2012/WCECS2012_pp199-203.pdf
- [17] I. Giannoukos, C. -N. Anagnostopoulos, V. Loumos, and E. Kayafas, "Operator context scanning to support high segmentation rates for real time license plate recognition," *Pattern Recognition*, vol. 43, no. 11, pp. 3866-3878, 2010, doi: 10.1016/j.patcog.2010.06.008.
- [18] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. V. Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 830-845, 2011, doi: 10.1109/TITS.2011.2114346.
- [19] Z. He, J. Liu, H. Ma, and P. Li, "A new automatic extraction method of container identity codes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 72-78, 2005, doi: 10.1109/TITS.2004.838509.
- [20] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Transactions on Intelligent transportation systems*, vol. 7, no. 3, pp. 377-392, 2006, doi: 10.1109/TITS.2006.880641.
- [21] 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 Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298-2304, 2017, doi: 10.1109/TPAMI.2016.2646371.
- [22] Y. Liu, T. Li, L. Jiang, and X. Liang, "Container-code recognition systembased on computer vision and deep neural networks," in *AIP Conference Proceedings*, 2018, doi: 10.1063/1.5033782.
- [23] D. Song, "A Literature Review, Container Shipping Supply Chain: Planning Problems and Research Opportunities," *Logistics*, vol. 5, no. 2, 2021, doi: 10.3390/logistics5020041.
- [24] B. Li and Y. He, "Computational Logistics for Container Terminal Handling Systems with Deep Learning," *Computational Intelligence and Neuroscience*, 2021, doi: 10.1155/2021/5529914.
- [25] R. Zhang, Z. Bahrami, T. Wang, and Z. Liu, "An Adaptive Deep Learning Framework for Shipping Container Code Localization and




- Recognition,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-13, 2021, doi: 10.1109/TIM.2020.3016108.
- [26] K. Liu, C. Sun and H. Chi, “Boundary-based Real-time Text Detection on Container Code,” in *Proc. of 2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, 2021, pp. 78-81, doi: 10.1109/ISCSIC54682.2021.00025.
- [27] R. Szeliski, *Computer Vision: Algorithms and Applications*, Cham, Switzerland AG: Springer Nature, 2022, doi: 10.1007/978-3-030-34372-9.
- [28] E. R. Dougherty, *An Introduction to Morphological Image Processing*, Universitas Michigan: SPIE Optical Engineering Press, 1992. [Online]. Available: <https://spie.org/Publications/Book/48126?SSO=1>
- [29] N. Otsu, “A Threshold Selection Method from Gray-Level Histogram,” *IEEE Transactions Systems, Man, and Cybernetics*, 1979. [Online]. Available: https://cw.fel.cvut.cz/b201/_media/courses/a6m33bio/otsu.pdf
- [30] Y. X. Dong, “Review of Otsu Segmentation Algorithm,” *Advanced Materials Research*, vol. 989-994, pp. 1959–1961, 2014, doi: 10.4028/www.scientific.net/AMR.989-994.1959.
- [31] W. Zhu, Q. Chen, C. Wei, and Z. Li, “A segmentation algorithm based on image projection for complex text layout,” in *AIP Conference Proceedings*, 2017, doi: 10.1063/1.5005199.
- [32] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010, doi: 10.1007/s11263-009-0275-4.

BIOGRAPHIES OF AUTHORS






Hoang-Sy Nguyen    was born in Binh Duong province, Vietnam. He received the B.S. and MS.c degree from the Department of Computer Science from Ho Chi Minh City University of Information Technology (UIT-HCMC), Vietnam in 2007, 2013, respectively. He received his Ph.D. degree in computer science, communication technology and applied mathematics from the VSB-Technical University of Ostrava-Czech Republic, in 2019. He is a senior researcher at Becamex Business School, Eastern International University, Vietnam. His research interests include Energy efficient, wireless communications, Applied Mathematical, Artificial Intelligence, Big data, and Data analyst. He can be contacted at email: sy.nguyen@eiu.edu.vn.



Cong-Danh Huynh    is a Lecturer at the Faculty of Economics, Thu Dau Mot University, Binh Duong Province, Vietnam. He received the B.S. and MBA degree from the VNU HCMC-University of Science, and the University of Economics Ho Chi Minh City, Vietnam, respectively. His research interests are in Artificial Intelligence, Big data, and Data analysis. He can be contacted at email: danhhc@tdmu.edu.vn.



Nhat-Quan Bui    is a student in MSc program at the Centre for Artificial Intelligence Research and Optimisation (AIRO), Torrens University Australia, 88 Wakefield Street, Adelaide, SA 5000, Australia. He received a B.S. degree in Electrical Engineering Major from VietNam National University HCMC – International University, Vietnam in 2020. His research interests are in fields of Machine learning applications and Image processing. He can be contacted at email: nhat.bui@student.torrens.edu.au.