



An automated vision system for container-code recognition

Wei Wu^a, Zheng Liu^{b,*}, Mo Chen^a, Xiaomin Yang^a, Xiaohai He^a

^a School of Electronics and Information Engineering, Sichuan University, Chengdu 610064, China

^b School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada K1A 0R6

ARTICLE INFO

Keywords:

Computer vision
Text-line location
Character isolation
Character segmentation
Character recognition
Support vector machine

ABSTRACT

Automatic container-code recognition is of great importance to the modern container management system. Similar techniques have been proposed for vehicle license plate recognition in past decades. Compared with license plate recognition, automatic container-code recognition faces more challenges due to the severity of nonuniform illumination and invalidation of color information. In this paper, a computer vision based container-code recognition technique is proposed. The system consists of three function modules, namely location, isolation, and character recognition. In location module, we propose a text-line region location algorithm, which takes into account the characteristics of single character as well as the spatial relationship between successive characters. This module locates the text-line regions by using a horizontal high-pass filter and scanline analysis. To resolve nonuniform illumination, a two-step procedure is applied to segment container-code characters, and a projection process is adopted to isolate characters in the isolation module. In character recognition module, the character recognition is achieved by classifying the extracted features, which represent the character image, with trained support vector machines (SVMs). The experimental results demonstrate the efficiency and effectiveness of the proposed technique for practical usage.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, ship transportation industry becomes more and more important with the development of internationalization. Meanwhile the number of containers transported has increased with this trend. Recognizing container identity code (also known as container code) becomes an essential for container management. The traditional manual method to recognize container code has lots of shortcomings, including slow speed, high error rate, etc. So an automatic technique to identify the container is desired. Comparing with other techniques, such as wireless sensor network and radio frequency identification (RFID), a computer vision based automatic container-code recognition (ACCR) is a vital technique to manage containers efficiently. The ACCR system described in this paper is to recognize ISO-6346 code on containers (ISO-6346, 2010). According to ISO standards, the ISO-6346 code consists of three parts: four capital letters, six digits, and one check digit. There may be extra characters beside eleven ISO characters on the container; however, these eleven ISO characters defined by ISO standard are considered as a unique code to identify different containers.

The ACCR system consists of three main modules: (1) locating text-line regions (location), (2) isolating container-code characters

(isolation), and (3) recognizing container-code characters (character recognition). In the location module, text-line regions are identified based on the features of container code. In the isolation module, container-code characters are segmented from the backgrounds. In the third module, the extracted character images are recognized and transferred into actual characters. The flowchart of the ACCR procedure is given in Fig. 1.

The ISO standard only defines the code types on the container. Colors of the characters and backgrounds, font types and sizes, and container-code positions vary from container to container. This introduces challenges to the ACCR application. Examples of typical container images are given in Fig. 2. The characteristics of the container summarized below:

1. Container code may occur in different positions, and may have different colors, sizes, and font types.
2. The contrast between container-code characters and backgrounds is sharp, which leads to strong vertical edges for each character.
3. The alignment modes of container-code characters are varied. For instance, container-code characters may be aligned horizontally in one row to multi-row or vertically in one column.
4. For horizontal alignment mode, spaces between two successive characters are small. And there are always several characters in each text-line region. An example of text-line regions is shown in Fig. 3.

* Corresponding author.

E-mail address: zheng.liu@ieee.org (Z. Liu).

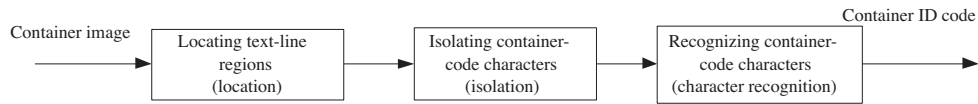


Fig. 1. Flowchart of automatic container-code recognition.

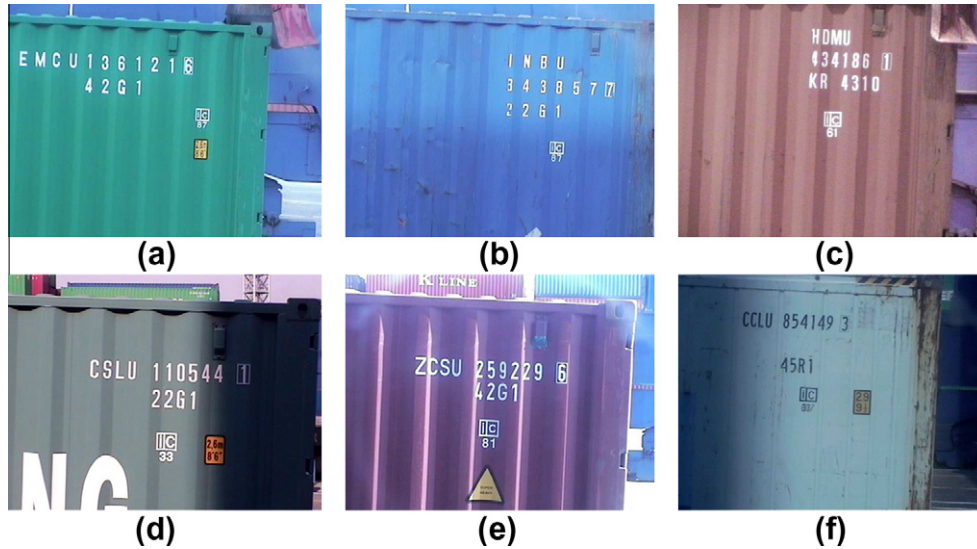


Fig. 2. Examples of container-code images.

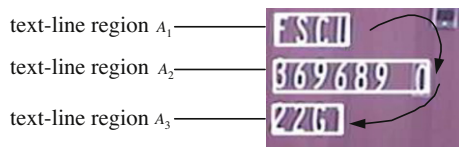


Fig. 3. The text-line regions and its process order.

Since the horizontal alignment mode is a dominating mode, this paper mainly describes the method applied to the horizontal alignment mode. The process can be easily adapted to vertical alignment mode by rotating the container image 180°. Only few parameters need to be adjusted.

Although container-code recognition is similar to vehicle license plate recognition (for example, both include three same function modules: location, isolation, and character recognition), container-code recognition is facing more challenges compared with license plate recognition. Color (Deb, Kang, & Jo, 2009) and plate edges (Duan, Du, Phuoc, & Hoang, 2005), which are very useful for locating license plate region, cannot provide useful information for container-code location. Moreover, container-code characters are more easily affected by outdoor illumination conditions when compared with license plate characters. Especially, surface curvature introduces severe illumination change for container-code characters. Due to such differences, methods for license plate recognition may be not applicable to the container-code recognition.

In this paper, a computer vision based container-code recognition technique is proposed. This technique consists of three function modules: location, isolation, and character recognition. A scanline-based text-line region location algorithm is implemented in the location module, which identifies the text-line regions by using a horizontal high-pass filter and scanline analysis. In isolation module, a two-step segmenting operation is first applied to text-line regions. Then, a projection process is implemented to isolate characters from these regions. In the character recognition

module, SVMs are trained to classify the isolated characters with extracted features. Experiments are carried out with 1214 container images.

The rest of this paper is organized as follows. Related works is overviewed in Section 2. An algorithm to locate text-line region is described in Section 3. Sections 4 and 5 are devoted to the isolation and character recognition respectively. Experimental results are presented in Section 6. This paper is summarized in Section 7.

2. Related works

Locating text-line regions on a container is a challenge because their positions, colors of characters and backgrounds, and font types, sizes may vary from one to another. Kim, Kim, and Woo (2007) proposed using an adaptive resonance theory (ART-2) based quantization method to identify text-line regions with features of characters such as color, size, and ratio of height to width. However, this method may fail when the container image suffers from nonuniform illumination. Moreover, the ART-2 based quantization is computation intensive. Similar to the methods in (Abolghasemi & Ahmadyfard, 2009; Huang, Chang, Chen, & Sandnes, 2008; Huang, Chen, Chang, & Sandnes, 2009), He, Liu, Ma, and Li (2005) proposed a method, where image edges were extracted and projected horizontally with a Gaussian smoothing filter. The position of the local maximums and local minimums of the smoothed histogram were found. From each local maximum, the top and bottom position of each text-line region can be obtained. However, this method is a still lack of robustness for container images with noises and character likenesses.

The purpose of character isolation is to isolate all eleven ISO characters from backgrounds. Character isolation methods can be roughly classified into two classes, i.e. global optimization-based method and segmentation-based method. In global optimization-based methods, the goal is to obtain a combined result of character spatial arrangement and single character recognition result rather

than only to obtain good recognition results for each character. In Franc and Hlavac (2005) and Fan and Fan (2009), a hidden Markov chain was used to formulate the dynamic segmentation of characters, and the segmentation problem was expressed as the maximum a posteriori estimation from a set of admissible segmentations. However, the global optimization-based method is impractical for real-time implementation. In segmentation-based methods, text-line or plate region is segmented first. Then, vertical and horizontal projections, connected component analysis (Li, Zeng, & Lin, 2006; Mahini, Kasaei, Dorri, & Dorri, 2006; Martin, Garcia, & Alba, 2002), or contour analysis (Anagnostopoulos, Anagnostopoulos, Psoroulas, Loumos, & Kayafas, 2008) are applied to obtain the position of each character and other binary object measurements such as height, width, and area may be used to eliminate noises. Since segmentation with one global threshold cannot always achieve good results with nonuniform illumination, local segmentation method becomes a better choice in this case. The segmentation methods proposed in Niblack (1986), Sauvola and Pietikinen (2000), Nakagawa and Rosenfeld (1979) are adopted in Coetzee, Botha, and Weber (1998), Anagnostopoulos, Anagnostopoulos, Loumos, and Kayafas (2006), Chang, Chen, Chung, and Chen (2004) respectively. Although these segmentation methods are well applied to vehicle license plate images, they may fail for container images encountering nonuniform illumination. In Naito, Tsukada, Yamada, Kozuka, and Yamamoto (2000), an image is first divided into $m \times n$ fixed-size blocks, and then a threshold is chosen for each block. However, the fixed-size block is not an optimal choice for segmentation. Meanwhile, it is hard to determine the size of blocks.

To recognize segmented character images, numerous methods from template matching and neural network to SVM have been investigated. The template matching technique is suitable to recognize characters with non-deformable and fixed-size fonts. However, container-code characters do not meet such requirements. Self-organized neural network (Chang et al., 2004), probabilistic neural networks (Anagnostopoulos et al., 2006), and back-propagation neural network (Jiao, Ye, & Huang, 2009), HMM (hidden Markov model) (Duan et al., 2005), and SVM (Dong, Krzyzak, & Suen, 2005; Shanthi & Duraiswamy, 2010) are well adopted methods in recognition. It has demonstrated that SVM outperforms other recognition methods in previous study (Tapia & Rojas, 2005).

3. Scanline-based text-line region location

Text-line region location is the first module in the whole recognition system. Its purpose is to distinguish text-line regions from the backgrounds. Poor image quality, disturbances from other characters, and the reflection from the container surface make it a challenge to locate text-line regions accurately and efficiently.

As the color information is not useful for this application, a color container image is first converted to a grayscale image with the following equation:

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

where the R/G/B represent the red/green/blue component of the color image.

According to container-code characteristic (2) and (4), luminance values change sharply at a line crossing a text-line region; furthermore, such changes in the text-line region are much more frequent and significant than in other non-text-line regions. Fig. 4 gives an example of luminance changes in a container image. Fig. 4(b) shows the cross-section line at position 1 while Fig. 4(c) illustrates the luminance change at position 2. Based on above observation, a text-line region can be identified by these sharp

changes and the change frequency. The proposed location procedure is shown in Fig. 5.

3.1. Generating edge image

Since container-code characters have strong vertical textures, a region with vertical textures will be identified and extracted as a character region. A horizontal high-pass filter is applied to detect the vertical edges. Such operation can be expressed as:

$$I_{vg} = I * H \quad (2)$$

where I and I_{vg} are the original and vertical edge image respectively. Operation $*$ stands for convolution and H is a horizontal high-pass filter, which is expressed as:

$$\frac{1}{n} [-1, \dots, -1, k, -1, \dots, -1]_n \quad (3)$$

where n is the length of H , and $k = n - 1$. We chose $n = 13$ in our system. Fig. 6(b) shows the results of filtering two original container images, which are shown in Fig. 6(a).

After filtering operation, every pixel in I_{vg} above a threshold T is considered as a salient edge pixel. As the threshold is higher in brighter regions, an adaptive thresholding operation based on the mean luminance value in the filter window is implemented. The pixel at (x, y) of the binary edge image I_{vb} is set to either 0 (non-edge point) or 1 (edge point) according to:

$$I_{vb}(x, y) = \begin{cases} 0, & I_{vg}(x, y) < \beta M_V(x, y) \\ 1, & I_{vg}(x, y) > \beta M_V(x, y) \end{cases} \quad (4)$$

where M_V is the mean image of I obtained by an average filter. $M_V(x, y)$ refers to a pixel value of M_V , whose coordinate is (x, y) . And β is a coefficient, which is set as 0.1 by trial and error. The binary edge images are shown in Fig. 6(c).

3.2. Removing the non-character edges

Besides the character edges, there are many non-character edges in I_{vb} , such as random noise edges, reflection edges, and background edges. These non-character edges may interfere with the container-code location.

In order to suppress these non-character edges, we obtain each edge's height and position by applying connected component analysis, firstly. Then, we remove two types of suppressed non-character edges, which are defined as follows:

- Any edge shorter than the minimum height of container-code character C_{min}^h ;
- Any edge longer than the maximum height of container-code character C_{max}^h on condition that the distance between the edge and any character edge is larger than the maximum distance between two successive characters C_{dis} .

Suppressed edges of the first type are mainly caused by stains and random noises while the second type is usually background edges or reflection edges, which are generated by the rugged container surface or by the nonuniform illumination. The condition in the second type is to prevent the character edges, which adhere to the reflection edges, to be considered as non-character edges. Fig. 7 shows two-type suppressed edges to be removed. After removing these edges, most of the non-character edges are eliminated while character edges are preserved. The binary edge images after removing non-character edges are shown in Fig. 6(d).

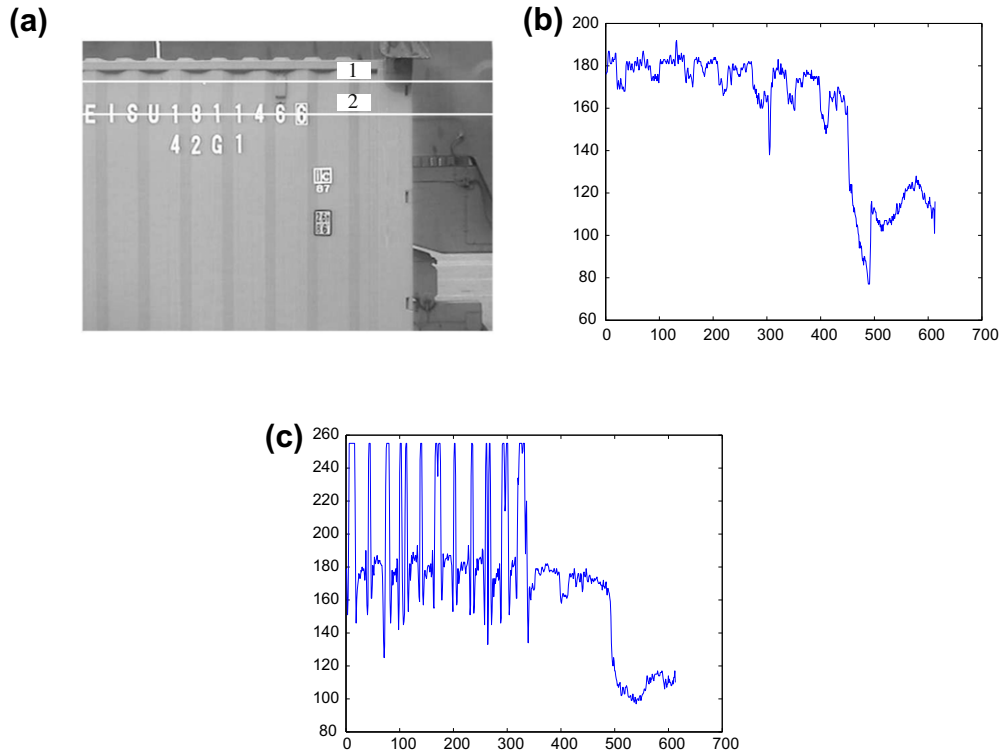


Fig. 4. Example of luminance changes (a) the luminance image; (b) cross-section line at position 1; and (c) cross-section line at position 2.

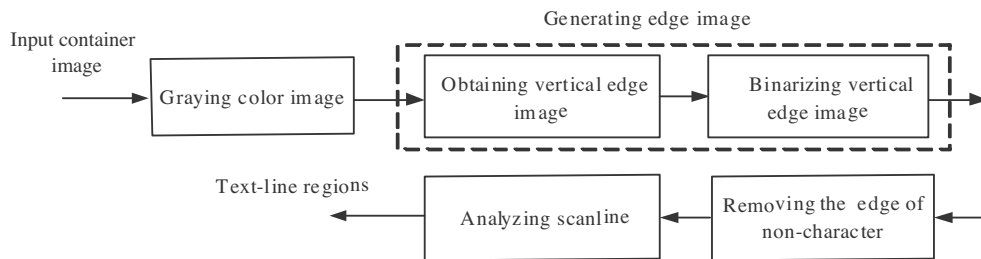


Fig. 5. Flowchart of the text-line location.

3.3. Analyzing scanline

The procedure to analyze scanline for locating text-line regions is illustrated in Fig. 8. Suppose that a “jump” is characterized by a change of the luminance value from “1” to “0” or from “0” to “1”.

Herein, we define a new term “scanline” for locating text-line regions. A scanline is defined as a line at a row of I_{vb} , which starts with a jump and ends with another jump. A group of neighboring scanlines can comprise of a text-line region. Fig. 9 shows a scanline and its jumps. The jumps in one scanline must satisfy the following conditions:

- The distances of any two successive jumps in a scanline are smaller than C_{dis} .
- There are at least $N_{pl} \times 2$ continual jumps in a scanline. N_{pl} denotes the minimum number of characters in a text-line region. As there are at least three characters in a text-line region, we set $N_{pl} = 3$ in our system.

To locate text-line regions, we first scan image I_{vb} from left to right for each row to find scanlines. Then, we connect the neighboring scanlines together to comprise of text-line regions. The relationship of neighboring scanlines can be expressed as:

$$Con(S_i^m, S_{j=i\pm 1}^n) = \begin{cases} 1, & S_i^m(end) > S_j^n(start) \text{ and } S_i^m(start) < S_j^n(end) \\ 0, & otherwise \end{cases} \quad (5)$$

where S_i^m denotes the m th scanline in i th row; $S_i^m(start)$ and $S_i^m(end)$ refer to horizontal start and end position of S_i^m respectively. If S_i^m and S_j^n are connected to each other, $Con(S_i^m, S_j^n)$ returns one; otherwise it returns zero.

Once the relationship of scanlines is identified, “scanline-group” regions can be located by connected component analysis. However, not all of these “scanline-group” regions are text-line regions. Since a text-line region is determined by its characters’ edges, the height of text-line region is approximately equal to the height of its corresponding characters. Therefore, any region, whose height is shorter than C_{min}^h or longer than C_{max}^h is removed. The final location results are shown in Fig. 6(e).

4. Container-code character isolation

The isolation module will isolate all the eleven ISO container-code characters from the text-line regions. The isolation procedure is illustrated in Fig. 10. Since container code may be aligned in several text-line regions, we need to identify these text-line regions

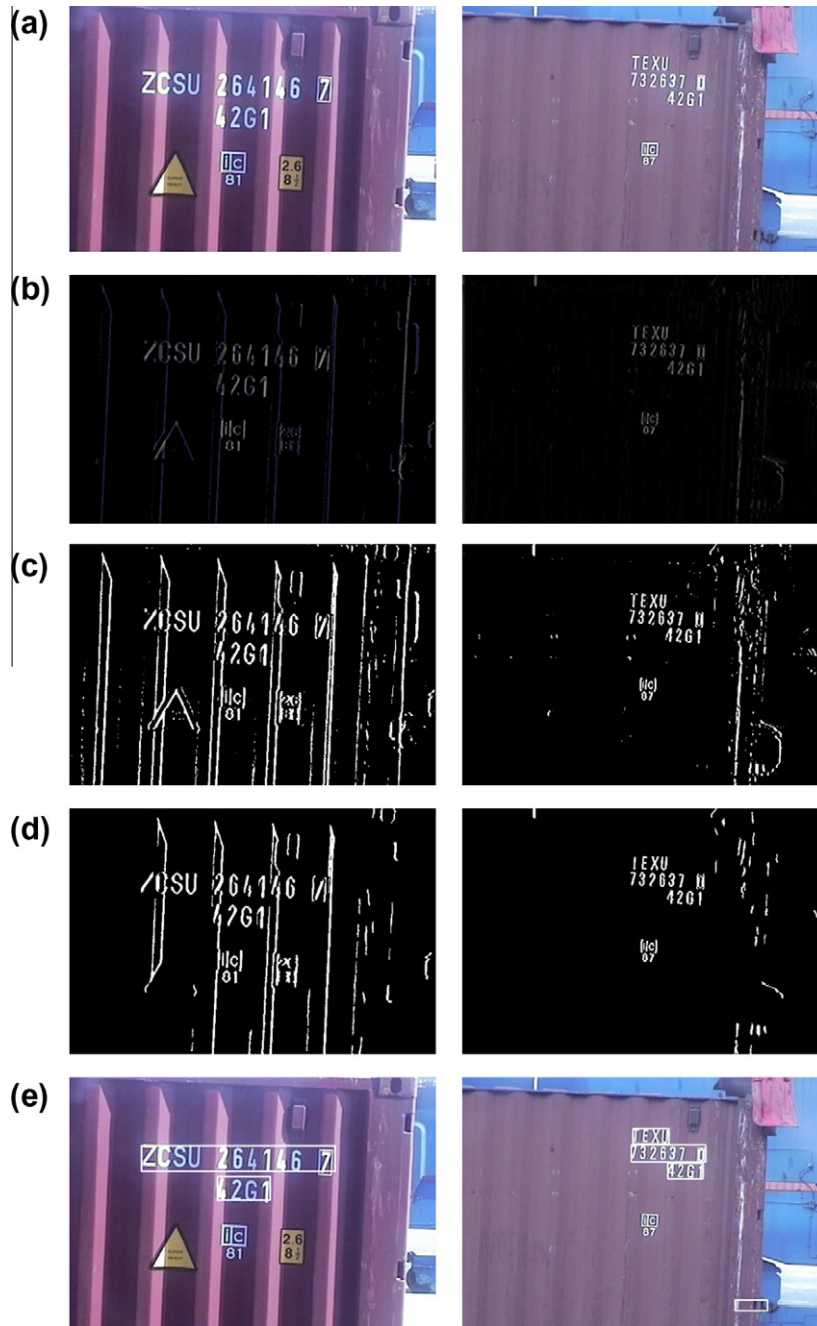


Fig. 6. Some example results for location procedure (a) original container images; (b) edge images I_{vg} ; (c) binary edge images I_{vb} ; (d) binary edge images after removing non-character edges; and (e) final results of text-line region location.

based on the characteristics of the alignment modes. As the ISO characters are printed from left to right and from top to bottom, we process the text-line regions in the same order. Fig. 3 illustrates such an order.

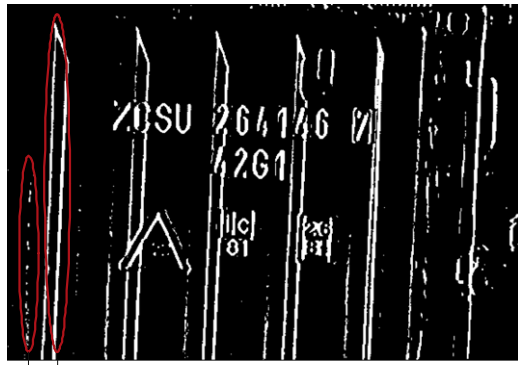
Suppose current text-line region A_c is fed into this module and each character in this region is isolated. If all the eleven ISO characters are not isolated, we move to the text-line region $A_{neighbor}$ adjacent to A_c until all the ISO characters are isolated. Specifically, the relationship between $A_{neighbor}$ and A_c should satisfy either of the two conditions:

$$l_n - r_c < D_h \quad \text{and} \quad \begin{cases} |t_c - t_n| < \varepsilon A_c^{height} \\ |b_c - b_n| < \varepsilon A_c^{height} \end{cases} \quad (6)$$

or

$$t_n - b_c < D_v \quad \text{and} \quad \begin{cases} |l_c - l_n| < \varepsilon A_c^{height} \\ |A_c^{height} - A_{neighbor}^{height}| < \varepsilon A_c^{height} \end{cases} \quad (7)$$

where t_c, b_c, l_c, r_c and t_n, b_n, l_n, r_n refer to the top, bottom, and left and right position of A_c and $A_{neighbor}$ respectively. D_h and D_v are the maximum distance between two text-line regions in the horizontal and vertical direction. A_c^{height} and $A_{neighbor}^{height}$ are the height of A_c and $A_{neighbor}$ respectively. We set $D_h = D_v = A_c^{height} / 2$ according to the priori knowledge of alignment mode. ε is a small plus constant, which is tolerance for noise. And we set $\varepsilon = 0.1$ in our implementation. Eq. (6) represents the horizontal relationship of two text-line regions



Type one suppressed non-character edges Type two suppressed non-character edges

Fig. 7. Type one and two suppressed edges to be removed.

while Eq. (7) is based on the vertical relationship. Eqs. (6) and (7) are illustrated in Fig. 11(a) and (b) respectively.

As character sizes, and the distances between successive character are roughly same in one container image, the standard variances of them are used to valid these isolated container-code characters. If the summation of the standard variances is smaller than a threshold, these isolated characters are validated. Otherwise, we continue the isolation process.

Usually, the check digit of container code is within its bounding rectangle (see Fig. 12(a)). To recognize the check digit, this rectangle needs to be removed. Detailed descriptions on the check digit rectangle elimination and the character isolation from the text-line regions are provided in the following subsections.

4.1. Isolating characters from a text-line region

To isolate the characters, the text-line regions are firstly segmented. Then, vertical and horizontal projection is applied to get the position of each character. This operations illustrated in Fig. 13.

4.1.1. Segmentation

Since container images are captured in the outdoor environment, they are subject to nonuniform illumination, reflections, and shadows. This makes it difficult to separate characters from backgrounds. Many segmentation approaches, which have been successfully applied to license plate recognition, are not suitable to container-code characters. Examples of a nonuniform illumination are given in Fig. 2(d)–(f).

To deal with these problems, we propose a two-step approach to segment characters from the backgrounds. Although the whole text-line region is subject to nonuniform illumination, local regions can still be considered in an ideal condition of uniform illumination. In the first step, the text-line region can be roughly divided into three area types, namely non-character area, character area, and reflection area. The three area types in an original grayscale image are illustrated in Fig. 14(a). In the second step, a combined segmentation strategy is applied to these areas.

Dividing. Because character edges are stable and not sensitive to nonuniform illumination, the text-line region is divided into

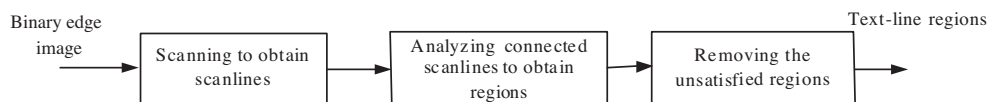


Fig. 8. The procedure for scanline analysis.

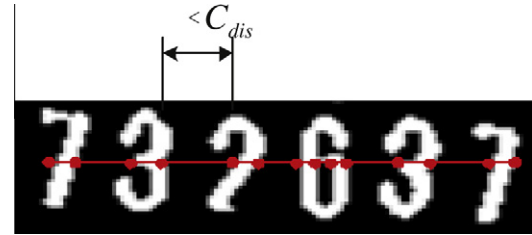


Fig. 9. A scanline and its jumps.

different areas with the edges. To reduce the computational load, the vertical version of filter H is applied to $I_{vg}(A)$ i.e. text-line region A in I_{vg} to generate the edge image $I_{vhg}(A)$. $I_{vhg}(A)$ consisting of both vertical edges and horizontal edges can be expressed as:

$$I_{vhg}(A) = I_{vg}(A) * H^T = I(A) * H * H^T \tag{8}$$

where H^T denotes the vertical version of filter H . After obtaining $I_{vhg}(A)$, the same adaptive threshold used in location module is applied to generate the binary edge image $I_{vnb}(A)$. Fig. 15(a) and (b) show the original image $I(A)$ and its binary edge image $I_{vnb}(A)$ respectively. Similar to the location module, we remove the non-character edges according to edges' height and width (any edge, whose height is shorter than C_{min}^h ; or any edge, whose height is longer than C_{max}^h and width is shorter than $A_c^{height}/2$, is removed). The binary edge image after removing non-character edges is shown in Fig. 15(c). Subsequently, the vertical projection histogram of $I_{vnb}(A)$ is calculated as:

$$h_A(i) = \sum_{j=1}^{A^{height}} I_{vnb}(A)(i, j), \quad i = 1, 2, \dots, A^{width} \tag{9}$$

where A^{height} and A^{width} represent the height and width of the text-line region A respectively. Fig. 15(d) shows the vertically projected results. Since usually reflections appear vertically and cross text-line region (see Fig. 2(e)), the $h_A(i)$ in these areas are almost equal to A^{height} . Therefore, the areas with $h_A(i)$ larger than the threshold $T_1 = (1 - \epsilon)A^{height}$ are identified as reflection areas. Since non-character areas have few edges, $h_A(i)$ of non-character area are below a threshold T_2 , where $T_2 < T_1$ and $T_2 = \epsilon A^{height}$. A character area is defined as an area with its $h_A(i)$ falling in $[T_2, T_1]$. The illustration of a text-line region with different areas and their corresponding thresholds is shown in Fig. 14(b). And Fig. 15(e) shows the different areas after dividing.

Segmenting. After dividing the text-line region into different areas, a combined segment strategy is applied. We only need to deal with the areas containing characters, i.e. character areas and reflection areas. For character areas, we assume that the character and the background have two normal distributions with similar variances. Therefore, Otsu's segmentation method is applied to each character areas (Otsu, 1979). For reflection areas, we use a threshold defined as $T_3 = (1 - \epsilon)v_{max}$ to segment every reflection area. v_{max} is the mean value of the largest k luminance values in the reflection area. Here, we set k equal to 5% of pixels of the reflection area. Fig. 15(f) shows the final segmentation results.

The comparison of different segmentation methods is shown in Fig. 16. We can observe that our approach is better than other state-of-art methods. Each character is clearly separated from the

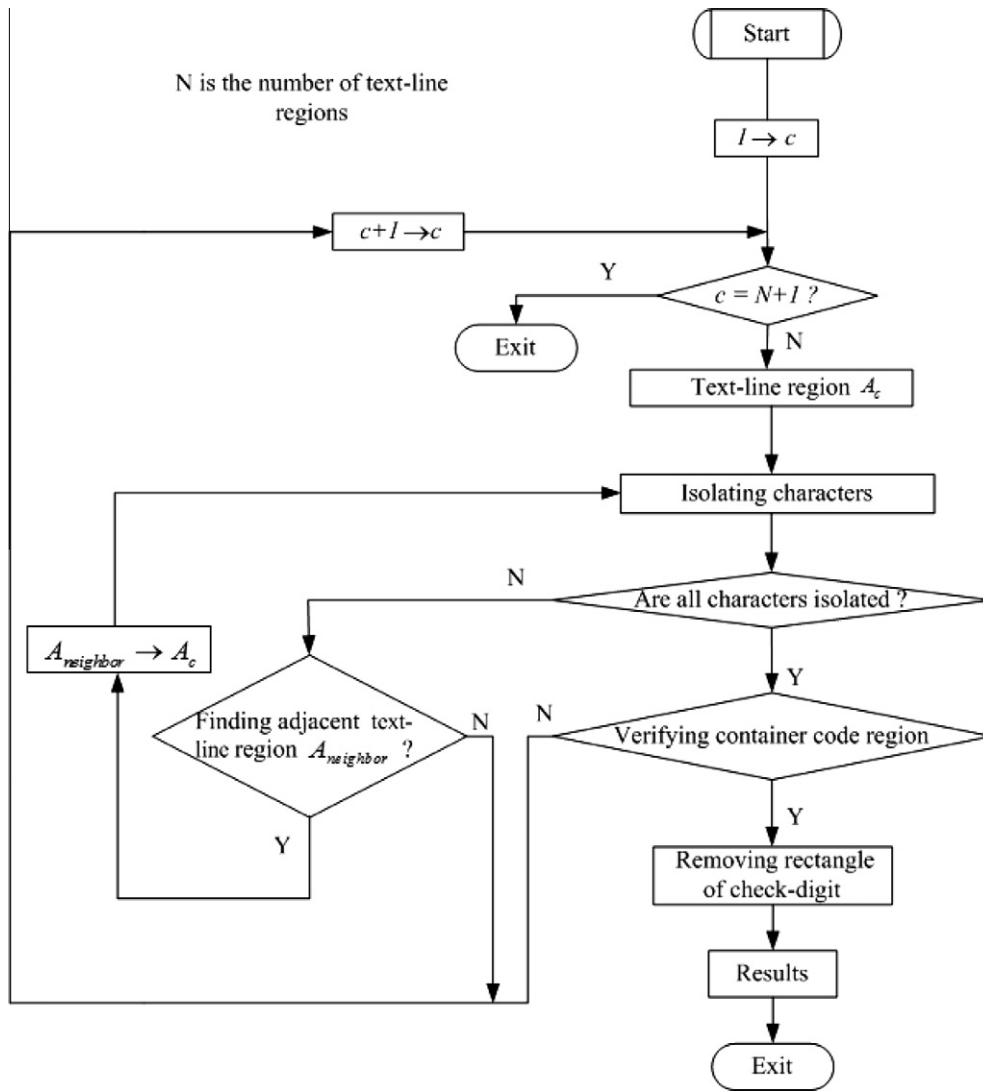


Fig. 10. Isolation framework.

backgrounds, and there are few noises in our results when compared with others.

4.1.2. Projection

After segmentation, projection method (Shi, Zhao, & Shen, 2005; Wang, Ni, Li, & Chen, 2004) or connected component analysis (Li et al., 2006; Mahini et al., 2006; Martin et al., 2002) is adopted to isolate the characters from the text-line region. In our implementation, a projection method is used for computational simplicity. To identify the horizontal confine of a character, vertical projecting is carried out to obtain a histogram, whose minimum values allow us to divide the text-line region into confined character areas. If two adjacent areas are close enough, they will be merged into one. Similarly, the top and bottom confine of each character can be located by using horizontal projection. With the above process, each character can be accurately located.

4.2. Removing the rectangle of the check digit

To remove the rectangle from the check-digit image, we use a similar method as described in Kumano et al. (2004). Take the left edge of the rectangle as an example. The horizontal position of first change from “1” to “0” can be obtained by scanning check-digit

image from left to right at each row as shown in Fig. 12(b). Then, vertical projection histogram of the change-point map I_{ch}^p is calculated, which can be expressed as:

$$h_{A_{check}}(i) = \sum_{j=1}^{A_{check}^{height}} I_{ch}^p(i, j) \quad i = 1, 2, \dots, A_{check}^{width} \quad (10)$$

where A_{check}^{height} and A_{check}^{width} are the width and height of the check-digit image. A vertical projection example is shown in Fig. 12(c). Since the method in Kumano et al. (2004) does not work well in all situations, we choose the first position k in $h_{A_{check}}(i)$ bigger than T_4 , where there is $T_4 = \epsilon A_{check}^{height}$. After obtaining k , the left edge of the image is made to be zero from 0 to k . Performing a similar operation on the right, upper, and bottom sides of the check-digit image, we can obtain a check digit image without the bounding rectangle (see Fig. 12(d)).

5. Character recognition

The character recognition module is to convert isolated character images into characters. We first extract the features which represent the character image. Edge densities generated by Sobel operator in character image patches are used as the features in this

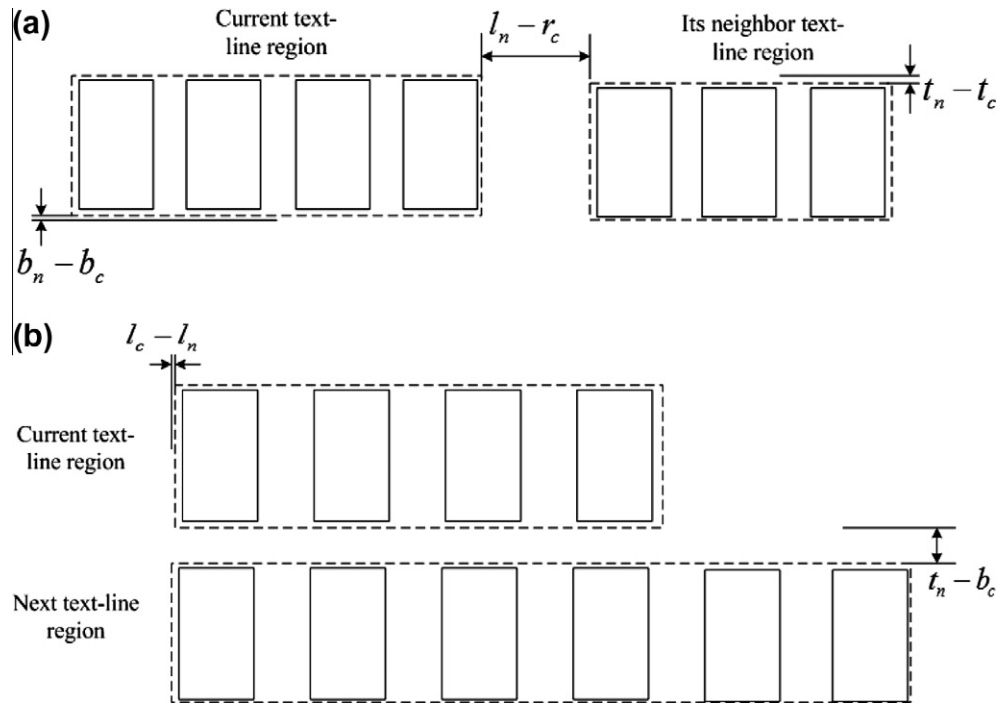


Fig. 11. The characteristics of the alignment modes: (a) the horizontal relationship; and (b) the vertical relationship of two text-line regions.

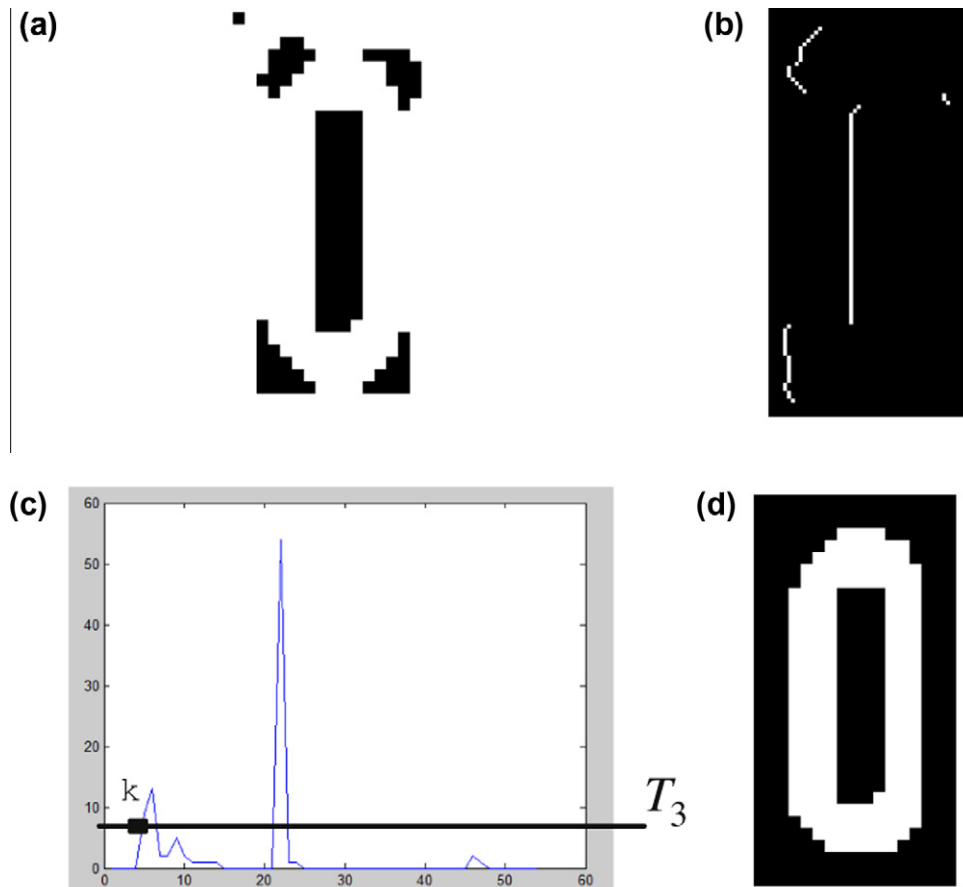


Fig. 12. Example of removing a rectangle from check-digit image: (a) original check-digit image; (b) the change point map (change points are shown in white color); (c) the histogram of changing points; and (d) the final result.

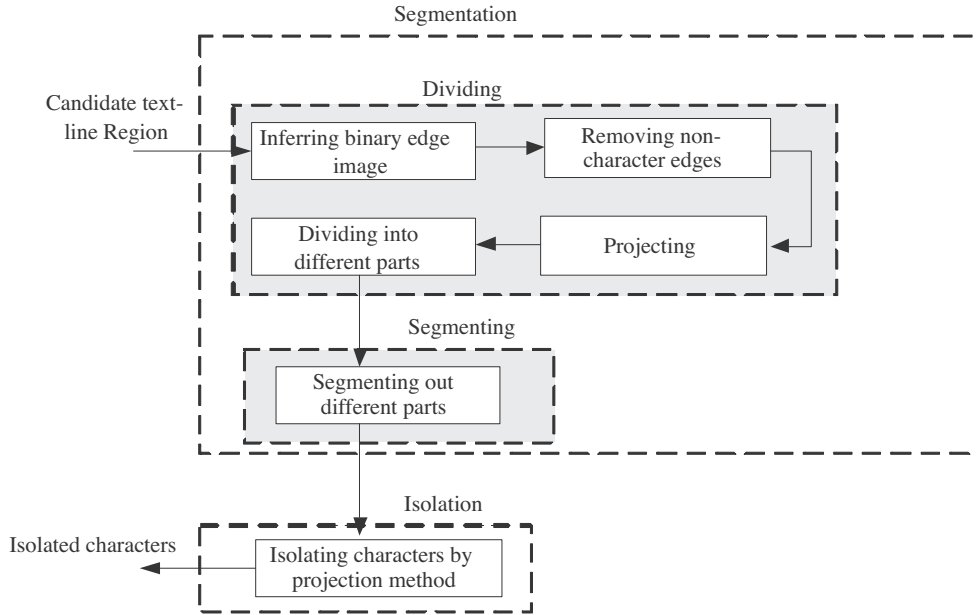


Fig. 13. The text-line segmentation process.

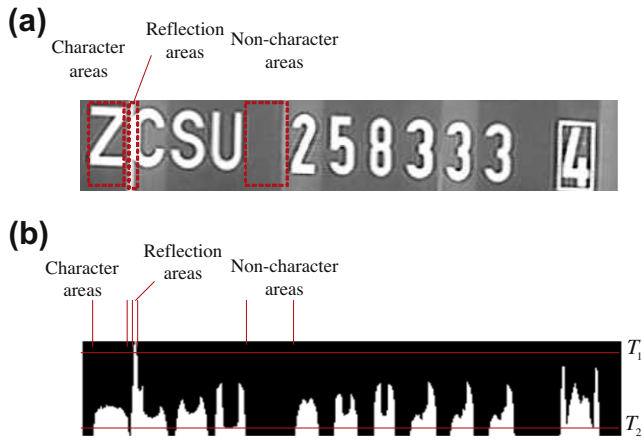


Fig. 14. Illustration of text-line region with non-character areas, character areas, reflection areas, and corresponding thresholds: (a) original grayscale image of a text-line region and (b) histogram of the vertical projection of the binary edge image derived from (a).

paper. Then, these features are fed into SVMs for classification. Since there are two types of container-code characters, i.e. capital letter and digit, we build two SVM based recognition models for character recognition, one for capital letters and the other for digits. The flowchart of the proposed character recognition is shown in Fig. 17.

5.1. Feature extraction

Sobel operator is first applied to extract vertical edge map C_x and horizontal edge map C_y from a character image I_c . C_y and C_x can be obtained as follows:

$$C_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I_c \tag{11}$$

$$C_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I_c \tag{12}$$

Fig. 18 gives an example of extracted edge maps. To emphasize the characteristics of each character, twenty-five local patches are defined as the shadow regions in Fig. 19. In Fig. 19(a), there are $4 \times 4 = 16$ patches, which evenly divide the character image. In Fig. 19(b), there are $3 \times 3 = 9$ patches, which overlap those patches defined in Fig. 19(a). We can divide C_x and C_y into 25 patches respectively. For each patch, the edge density is calculated as a feature. Suppose that F_x^l and F_y^l are the edge densities of C_x and C_y in zone l and can be expressed as:

$$F_x^l = \frac{\sum_{(i,j) \in Z(l)} C_x(i,j)}{Z_w \times Z_h} \tag{13}$$

$$F_y^l = \frac{\sum_{(i,j) \in Z(l)} C_y(i,j)}{Z_w \times Z_h} \tag{14}$$

where $(i,j) \in Z(l)$ refers to the position of i, j belonging to zone l ; Z_w, Z_h is the width and height of the zone respectively. Thus, each character image has 50 features fed into the SVM classifier.

5.2. Classification

A SVM classifier is generally a supervised binary classifier based on the statistical theory of Cortes and Vapnik (1995) and Burges (1998), where experimental data and structural behavior are taken into account for better generalization capability based on the principle of structural risk minimization (SRM). Given a training data set $(x_i, y_i); i = 1, \dots, l$, where $x_i \in R^n$ is samples and $y_i \in \{1, -1\}$ are labels; l refers to the number of samples. The SVM classifies an input z through the function:

$$f(z) = \sum_{i=1}^l \alpha_i y_i K(z, x_i) - b \tag{15}$$

$$b = y_r - \sum_{i=1}^l \alpha_i y_i K(x_r, x_i) \quad r = 1, 2, \dots, l \tag{16}$$



Fig. 15. Two segmentation examples: (a) original container-code images; (b) binary edge images; (c) denoised binary edge images; (d) vertically projected results; (e) different areas; and (f) final segmentation results.



Fig. 16. Comparison of segmentation: (a) original images; (b) results of Otsu's method (Otsu, 1979); (c) results of Niblack's method (Niblack, 1986); (d) results of logical level technique (LLT) (Kamel & Zhao, 1993); (e) results of Sauvola's method (Sauvola & Pietikinen, 2000); and (f) our results.

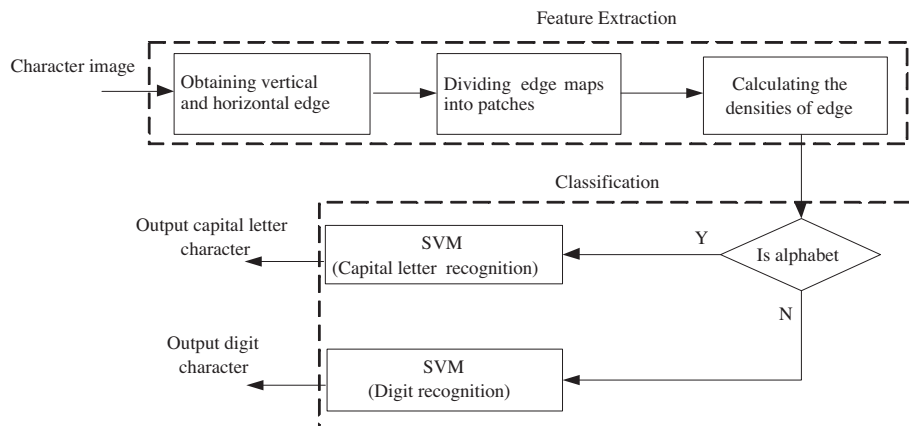


Fig. 17. Flowchart of character recognition.

$$C_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I_c$$

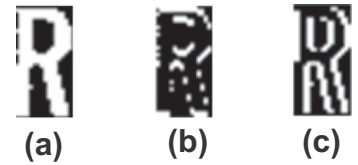
$$C_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I_c$$


Fig. 18. An example of edge maps (a) original binary character image; (b) horizontal edge map; and (c) vertical edge map.

where coefficient α_i are non-zero only for the subset of the input data called support vectors; $K(x,y)$ is a kernel function, which satisfies the Mercer conditions. The most commonly used kernels, including radial basis, polynomial, and sigmoid. In this study, the radial basis function (RBF) kernel is used, which can be express as:

$$K(x,y) = \exp\left(\frac{-\|x-y\|^2}{2\sigma^2}\right) \tag{17}$$

When training the models, samples were collected by hand for each character. Over 300 samples were marked for each character to ensure the recognition performance.

6. Experimental results

The success of the container-code recognition is defined as a process that is able to extract the text-line regions, isolate all the eleven ISO characters, and recognize all these characters correctly. The input to the recognition system is an image with container code, and the output is a series of characters derived from the image. This section presents the experimental results of the proposed

Table 1
Performance evaluation of the proposed technique.

Module	Daytime (%)	Night (%)	Accuracy (%)
Location	98.33	97.07	97.94
Isolation	96.36	94.25	95.71
Character recognition	98.11	97.09	97.80
Overall performance	92.96	88.84	91.68

container-code recognition system, including location, isolation, recognition, and overall recognition results.

In order to evaluate our method, 1214 container images (640 × 480 pixels) obtained from real port environments were used in the experiment, among which 376 and 838 container images were captured during night and daytime respectively. Both the brightness and contrast of the images change rapidly. And the container codes in the images are of varied colors and sizes, and are aligned and located differently.

The recognition results are given in Table 1, which includes location, isolation, recognition, and the overall accuracy rate. The accuracy rate for each module is defined as:

$$Accuracy\ rate = \frac{Number\ of\ correctly\ processed\ samples}{Number\ of\ all\ test\ samples} \times 100\% \tag{18}$$

The overall accuracy rate is the product of accuracy rates of the three modules.

The proposed algorithms can accurately locate the text-line regions for varied positions, colors, illumination conditions, alignment modes, and character sizes. Text-line region location accuracy rate is as high as 97.94%. Examples for location during daytime and night are given in Figs. 20 and 21 respectively. Multiple missing characters and severe noise contamination are the major reasons for incorrectly locating container code. Some examples of failure in location module are shown in Fig. 22. Since faded and seriously nonuniform illumination introduces difficulties to the isolation module, 30 images at daytime and 21 images at night were not successfully isolated. Thus, the isolation module achieved an accuracy rate of 95.71%. Examples of isolation results are given in Fig. 23. And some examples of the failure in isolation module are shown in Fig. 24. Due to character deformity in some cases, 25

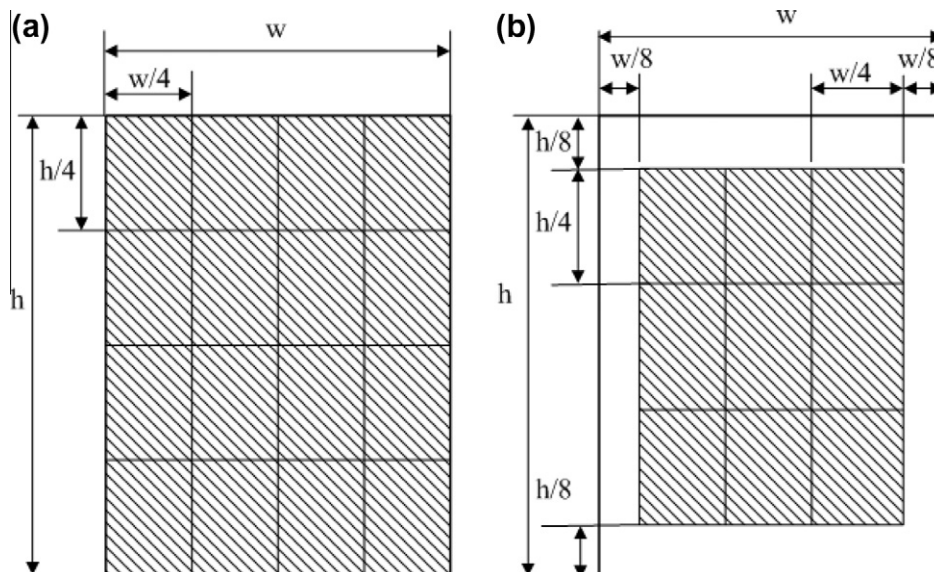


Fig. 19. Local patches for feature extraction.

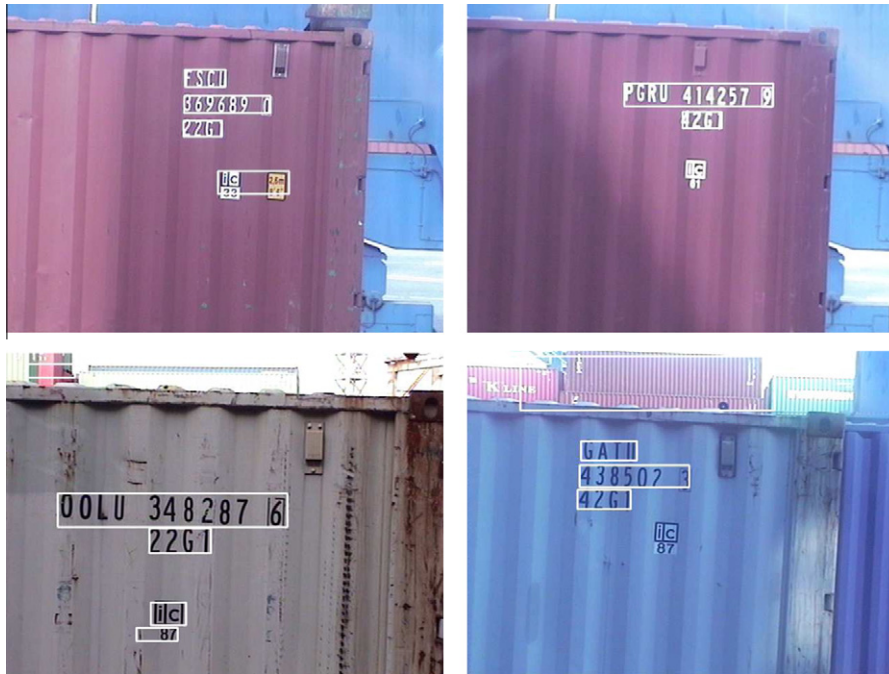


Fig. 20. Location examples of variety of containers at daytime.



Fig. 21. Location examples of variety of containers at night.

character images were not recognized correctly, which led to a character recognition rate of 97.80%. The overall recognition rate is 91.68%, for the all 1214 images.

A comparison of He's method in He et al. (2005) with ours for location and isolation is given in Table 2. Since the details of character recognition were not described in He's paper, the comparison of character recognition and overall performance is impossible. We adopted four different segmentation algorithms (Kamel & Zhao,

1993; Niblack, 1986; Otsu, 1979; Sauvola & Pietikinen, 2000) in the implementation of He's isolation method. Our proposed method outperforms He's location and isolation algorithms, which relies only on vertical edges of characters. In contrast, our location algorithm not only makes use of vertical edges of characters but also exploits the spatial relationships between successive characters. In addition, the proposed segmentation approach can deal with the nonuniform illumination and reflection on the containers very well.



Fig. 22. Some failure examples of location.



Fig. 23. Examples of isolation results.



Fig. 24. Some examples of failure in isolation.

Table 2
Performance comparisons.

Method	Location (%)	Isolation (%)
He's method (He et al., 2005)	94.91	88.16 (Ostu's method)
		90.12 (Niblack's method)
		92.75 logical level technique (LLT)
		92.97 (Sauvola's method)
Proposed method	97.94	95.71

7. Summary

The time complexity of the proposed technique is linear with the number of pixels of processed image. In the location module, image graying, edge extraction, non-character edge elimination, and scanline analysis, all have a time complexity of $O(s)$, where $s = W \times H$, and W and H define the width and height of the container image. For container-code character isolation, processing unit is text-line region, which is only a small part of the container image. Furthermore, all the operations of generating edge image, eliminating noises, and projecting in a text-line region have a time

complexity $O(a)$, where $a \ll s$ is the number of pixels of the text-line regions. For character recognition, the feature extraction procedure has no normalization, which results in that this procedure performs very fast. In the implementation of SVM, the time complexity of training algorithms is much higher than that of testing phase of the SVM. However, the SVM training process can be performed in advance. The SVM testing phase only requires $O(MN_s)$ operations, where N_s refers to the number of support vectors. And N_s is usually quite smaller than S , which refers to the number of samples while M is the dimension of the feature vector. The average time for processing a container image is less than 110 ms on a personal computer (Pentium-IV 2.4G with 1G RAM).

In this paper, we present a container-code recognition technique based on computer vision. There are two major contributions from this technique. The first one is the scanline-based algorithm to extract text-line regions, which combines the vertical character edges and the spatial relationship between successive characters. The proposed algorithm is suitable to locate container codes of different colors, sizes, and alignment modes. The second contribution is a two-step segmentation approach in the isolation module to tackle the severity of nonuniform illumination. The experiments with container images captured from the real port environment demonstrate the effectiveness of the proposed technique.

Future work will focus on recognizing container code from multiple images captured from the same container with one or multiple cameras during its moving. The container images are captured at the different positions or different time. Thus, the characters on the container may be clear in one image but not in another one. The authors believe that integrating or fusing these images for the same container can improve the performance of automatic

container-code recognition. This remains a topic for our future development.

References

- Abolghasemi, V., & Ahmadyfard, A. (2009). An edge-based color-aided method for license plate detection. *Image and Vision Computing*, 27(8), 1134–1142.
- Anagnostopoulos, C., Anagnostopoulos, I., Loumos, V., & Kayafas, E. (2006). A license plate-recognition algorithm for intelligent transportation system application. *IEEE Transaction on Intelligent Transportation Systems*, 7(3), 377–392.
- Anagnostopoulos, C.-N. E., Anagnostopoulos, I. E., Psoroulas, I. D., Loumos, V., & Kayafas, E. (2008). License plate recognition from still images and video sequences: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 9(3), 377–391.
- Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 121–167.
- Chang, S., Chen, L., Chung, Y., & Chen, S. (2004). Automatic license plate recognition. *IEEE Transaction on Intelligent Transportation Systems*, 5(1), 42–53.
- Coetzee, C., Botha, C., & Weber, D. (1998). Pc based number plate recognition system. In *IEEE international symposium on industrial electronics* (pp. 605–610).
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297.
- Deb, K., Kang, S.-J., & Jo, K.-H. (2009). Statistical characteristics in HSI color model and position histogram based vehicle license plate detection. *Intelligent Service Robotics*, 2(3), 173–186.
- Dong, J.-X., Krzyzak, A., & Suen, C. Y. (2005). An improved handwritten chinese character recognition system using support vector machine. *Pattern Recognition Letters*, 26, 1849–1856.
- Duan, T. D., Du, T. L. H., Phuoc, T. V., & Hoang, N. V. (2005). Building an automatic vehicle license plate recognition system. In *Proceedings of international conference on computer science* (pp. 59–63).
- Fan, X., & Fan, G. L. (2009). Graphical models for joint segmentation and recognition of license plate characters. *IEEE Signal Processing Letters*, 16(1), 10–13.
- Franc, V., & Hlavac, V. (2005). License plate character segmentation using hidden markov chains. In *Lecture notes in computer science, Vienna* (Vol. 3663, pp. 385–392).
- He, Z., Liu, J., Ma, H., & Li, P. (2005). A new automatic extraction method of container identity codes. *IEEE Transaction on Intelligent Transportation Systems*, 6(1), 72–78.
- Huang, Y.-P., Chang, T.-W., Chen, Y.-R., & Sandnes, F. E. (2008). A back propagation based real-time license plate recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(2), 233–251.
- Huang, Y.-P., Chen, C.-H., Chang, Y.-T., & Sandnes, F. E. (2009). An intelligent strategy for checking the annual inspection status of motorcycles based on license plate recognition. *Expert Systems with Applications*, 36(Issue 5), 9260–9267.
- ISO-6346, retrieved in 2010. URL <http://en.wikipedia.org/wiki/ISO_6346>.
- Jiao, J., Ye, Q., & Huang, Q. (2009). A configurable method for multi-style license plate recognition. *Pattern Recognition*, 42(3), 358–369.
- Kamel, M., & Zhao, A. (1993). Extraction of binary character/graphics images from grayscale document images. *Graphical Models and Image Processing*, 55(3), 203–217.
- Kim, K. B., Kim, M., & Woo, Y. W. (2007). Recognition of shipping container identifiers using art2-based quantization and a refined rbf network. In *Lecture notes in computer science* (Vol. 4432, pp. 572–581).
- Kumano, S., Miyamoto, K., et al. (2004). Development of a container identification mark recognition system. *Electronics and Communications in Japan-part 2*, 87(12), 38–50.
- Li, G., Zeng, R., & Lin, L. (2006). Research on vehicle license plate location based on neural networks. In *Proceedings of the first international conference on innovative computing, information and control, Beijing, China* (Vol. 3, pp. 174–177).
- Mahini, H., Kasaei, S., Dorri, F., & Dorri, F. (2006). An efficient features based license plate localization method. In *International conference on pattern recognition, Hong Kong* (Vol. 2, pp. 841–844).
- Martin, F., Garcia, M., & Alba, J. L. (2002). New methods for automatic reading of vlps (vehicle license plates). In *The IASTED international conference on signal processing, pattern recognition, and applications*.
- Naito, T., Tsukada, T., Yamada, K., Kozuka, K., & Yamamoto, S. (2000). Robust license-plate recognition method for passing vehicles under outside environment. *IEEE Transaction on Vehicular Technology*, 49(6), 2309–2319.
- Nakagawa, Y., & Rosenfeld, A. (1979). Some experiments on variable thresholding. *Pattern Recognition*, 11(3), 191–204.
- Niblack, W. (1986). *An introduction to digital image processing*. Englewood Cliffs, NJ: Prentice-Hall.
- Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1), 62–66.
- Sauvola, J., & Pietikinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33(2), 225–236.
- Shanthi, N., & Duraiswamy, K. (2010). A novel svm-based handwritten tamil character recognition system. *Pattern Analysis & Applications*, 13(2), 173–180.
- Shi, X., Zhao, W., & Shen, Y. (2005). Automatic license plate recognition system based on color image processing. In O. Gervasi, et al. (Ed.), *Lecture notes on computer science, Springer* (Vol. 3483, pp. 1159–1168).
- Tapia, E., & Rojas, R. (2005). Recognition of on-line handwritten mathematical expressions in the e-chalk system – An extension. In *Proceedings of the eighth international conference on document analysis and recognition* (pp. 1206–1210). Washington, DC, USA: IEEE Computer Society.
- Wang, T. H., Ni, F. C., Li, K. T., & Chen, Y. P. (2004). Robust license plate recognition based on dynamic projection warping. In *IEEE international conference on networking, sensing and control* (pp. 784–788).