Hidden-Markov-Model-Based Segmentation Confidence Applied to Container Code Character Extraction

Mo Chen, Wei Wu, Xiaomin Yang, and Xiaohai He

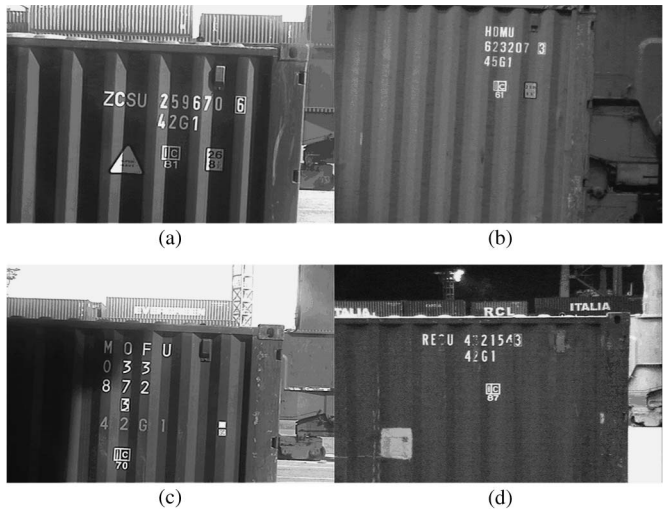
***Abstract*—Automatic container code recognition (ACCR) has become an indispensable aspect of current intelligent container management systems. In real applications, an ACCR module sometimes faces the problem of missing characters, i.e., not all the 11 container code characters (CCCs) appear in the input image. However, a few of the present methods can process container code images with missing characters. Therefore, a method is proposed to extract the CCCs for both the situation wherein all the 11 CCCs appear in an image and the situation wherein some CCCs are missing. In this method, hidden Markov model (HMM)-based segmentation confidence is proposed to describe the probability of the segmented characters belonging to the container code. Based on the segmentation confidence, the segmented characters are determined whether they belong to the container code or not, and if there are some characters missing, the positions of these characters can be estimated. Various container code images have been used to test the proposed method. The results of the tests show that the method is effective.**

***Index Terms*—Character extraction, container code, hidden Markov models (HMMs), image processing, segmentation confidence.**

I. Introduction

With the accelerating pace of globalization, fast and efficient transportation of goods between countries and regions has become increasingly more important. Intelligent identification of container codes is one of the most important tasks in transportation. Although some novel techniques, such as radio frequency identification [1] and wireless sensor networks, have been proposed for container code identification, automatic container code recognition (ACCR) from visual images is still an indispensable technique in current container intelligent management systems for its convenience and low cost.

Generally speaking, the ACCR process can be divided into three steps, i.e., code region localization, container code character (CCC) segmentation, and CCC recognition. The first step, i.e., code region localization, roughly separates the code region and the noncode region apart. The second step, i.e., CCC segmentation, extracts code character blocks from the located code region. The last step, i.e., CCC recognition, transfers the segmented character blocks into actual characters. It is important that the CCCs must be first properly extracted before the accurate recognition result being obtained.



(a) (b) (c) (d)

Fig. 1. Some container code images. (a)–(c) Images with different arrangements of CCCs. (d) Image with missing CCCs.

To effectively extract the CCCs, the characters' features are first reviewed.

Fig. 1 shows some container code images. The CCCs have several features [2], which can be briefly listed here.

1) A container code consists of 11 characters. The 11 characters can be divided into three parts, i.e., four English letters, six digital numbers, and one checkout digital number.
2) The arrangement of the three parts has many forms: They may horizontally or vertically align in only one line or multiple lines.
3) For different images, the distances between characters largely vary. However, for a given image, the distances between successive characters are almost equal.
4) There is a sharp contrast between the CCCs and the background.

However, there may be some "accidents" that have occured on the obtained container images, in which some characters are missing, due to being worn, eroded, or out of camera scope. (Fig. 1(d) displays an image with missing CCCs.)

At present, various methods have been proposed to fulfill the character extraction task, but most of these methods assume that all the characters appear in the input image. When there are

Manuscript received June 2, 2009; revised January 29, 2010, May 19, 2010, and January 9, 2011; accepted April 6, 2011. Date of publication May 12, 2011; date of current version December 5, 2011. The Associate Editor for this paper was N. Papanikolopoulos.

The authors are with the College of Electronics and Information Engineering, Sichuan University, Chengdu 610064, China (e-mail: wuwei@scu.edu.cn).

Digital Object Identifier 10.1109/TITS.2011.2145417

some characters missing, these methods would fail to extract the remaining characters. However, these characters in such cases are not totally useless because they can be used as a clue to search in a database containing all the existing container codes to find candidate container codes. Thus, a more compatible solution should be proposed to treat those images with missing characters. In this paper, a concept of the segmentation confidence is proposed, which is defined as the probability of the extracted characters, e.g., objects, belonging to the container code. Based on the segmentation confidence, even if some characters are missing, the remaining characters can still be extracted.

The rest of this paper is organized as follows: Literature review is given in Section II. The localization method is introduced in Section III. Our proposed segmentation confidence and character segmentation process are presented in Section IV. Experimental results are provided and discussed in Section V. The conclusions are given in Section VI.

II. Literature Review

Extraction of CCCs includes the first two steps of ACCR. The first step is to locate the candidate regions that may contain CCCs. The second step is to segment the CCCs from the proper regions. The steps are similar to the steps of vehicle license plate recognition (VLPR); thus, the techniques used in VLPR can be referred to.

For license plate localization, according to [3], the related methods can be categorized into four main categories, i.e., binary image processing [4]–[7], gray-level processing [8], [9], color processing [10]–[12], and classifiers [13], [14]. However, considering the lower computational complexity and robustness, binary image processing methods are most commonly used. These methods are based on the fact that the change in brightness in the plate region is more remarkable and more frequent than elsewhere. In these methods, edge detection and mathematical morphology [4] (or convolutions [5]) are often combined. Generally speaking, these methods are of low computation complexity and good effect for license plate localization. However, as the CCCs are arranged sparser than the characters of license plate, morphology methods may not work so well for container code images. As the discrete wavelet transform (DWT) can provide both the vertical and horizontal gradients of the image, Wang *et al.* [6] proposed to use the one-level 5/3 DWT to locate the car plate. This method can deal with some complex environments such as low contrast images and dynamic-range problems. However, the result was not stable under uneven light conditions. Al-Hmouz and Challa [7] also proposed to use various detection thresholds to detect the car plate. These detection results were then fused within Bayesian framework to give a final result. However, this method assumed that the plate region should form a connected area under some threshold value, but in the container code images' cases, this assumption does not hold.

As far as the container code region localization is concerned, it is difficult to directly locate the CCCs' region, as the CCCs' arrangement has various forms, and there are also some other characters that exist that are hard to eliminate in the localization process [e.g., there are four other characters -"42G1" in Fig. 1(a)]. Therefore, a fast code region preselection process, which can quickly find some code region candidates, is more attractive. Yin *et al.* [15] proposed to use the morphological methods to obtain the approximate container code areas and then use connected region analysis and projection operation to split out the fine code region. However, this method was mainly proposed for container back-view images, whereas for container side-view images, which are more popularly used for container code recognition, it does not work so well, as the CCCs are apart from each other in a larger distance. For container side-view images, in [2], the authors proposed to calculate a differential image in the horizontal orientation and to use the horizontal projection result to detect the position of text lines whose vertical textures are abundant. Since this method will not further locate the left and right bounaries of the container code region, more noise would be introduced.

After the character region candidates have been located, the next step is to segment the characters. The methods used in the license plate character segmentation can be classified into three classes [3], i.e., algorithms based on binary image processing, gray-level processing, and classifiers. In binary image processing algorithms, the method, which exploits vertical and horizontal projections of pixels, is the most common and simplest [9], [10], [16], [17]. Connected component analysis (CCA) is also frequently involved in binary image processing algorithms and often works with other binary object measurements such as height, width, and area to eliminate noises [18]–[20]. In gray-level processing algorithms, researchers are more inclined to choose dynamic threshold techniques to segment characters. Draghici [21] proposed to combine various histogram-thresholding techniques to determine the threshold. Other local binarization methods, such as calculating threshold for each image block [22] or even for every pixel [23], are employed to improve the segmentation performance. In classifier-based methods, the character segmentation is modeled as a global optimization problem [24].

After the bounding box of each character being obtained, the arrangement of the characters can be known in the character segmentation step for license plate. If the arrangement matches with the plate format, the segmentation results are defined as valid plate license characters. However, as far as the segmentation of CCCs is considered, the situation is more complex, because there are various align modes for CCCs. He *et al.* [2] proposed a template-matching method, which first brings forward a series of standard templates according to the standard align modes of the CCCs, then calculates the character align mode in candidate regions, and then uses the template-matching method to determine whether the extracted characters are CCCs.

Based on the preceding literature review, although many character extraction methods have been developed, these methods are mainly developed under the assumption that all the characters should appear in the input images. If some characters are missing, these methods would fail to extract the remaining characters.

Aiming at this problem, we propose a method to extract the CCCs, whether all the characters appear in the image or not.

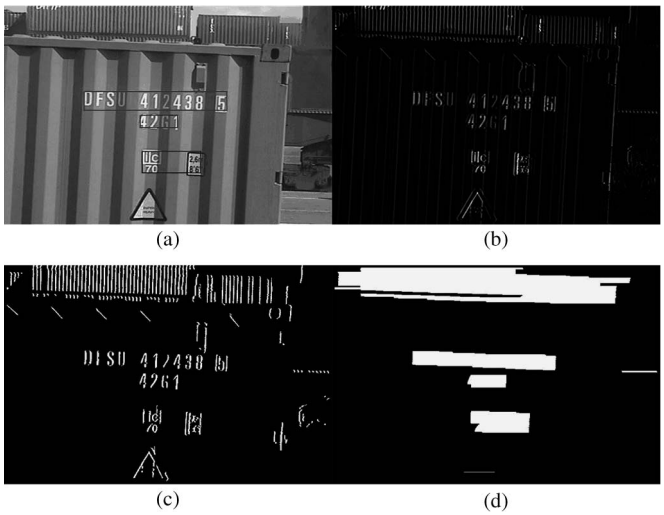


Fig. 2. Localization procedure. (a) Input image (with localization result). (b) Vertical textures. (c) Binary result (after noise removal). (d) Line connecting result.

III. Localization Algorithm

The image preprocessing involves transforming a color image into a gray one and applying a median filter to reduce the noise. For horizontally aligned container codes, we assume that a region with "dense" vertical textures should be extracted as a character region candidate. In our localization algorithm, a simple and effective method is used to detect vertical textures. Then, CCA is employed to eliminate the noise textures. Finally, each region of "dense vertical textures" is defined as a candidate region.

To obtain the vertical textures, a horizontal difference image is calculated, i.e.,

$$g = f - f \otimes H \tag{1}$$

where g is the horizontal difference image, f is the original image, "$\otimes$" stands for convolution, and H is the horizontal low-pass filter taking the form of

$$\frac{1}{n}\mathbf{1}_n^{\mathbf{T}} \tag{2}$$

where $\mathbf{1}_n$ is an $n \times 1$ vector of 1's.

In our work, we choose 11 for n, which is about twice the maximum character stroke width. The result is shown in Fig. 2(b). (For better visibility, a linear grayscale extension is performed.)

The binarization result of g can be obtained by an adaptive threshold method. For each pixel P, the threshold $T(P)$ is determined by the following:

$$T(P) = (f \otimes H)(P) * \alpha. \tag{3}$$

Here, $f \otimes H$ is the low-pass filtered image, and α is the adaptive parameter. In practice, an upper limit $T_{\max}$ and a lower limit $T_{\min}$ are used to limit the value of $T(P)$. In our work, we choose 0.1 for α, 20 for $T_{\max}$, and 8 for $T_{\min}$, respectively. If the pixel P's value is greater than $T(P)$, P is defined as an object pixel. After the binary image being obtained, CCA is used to eliminate the noise textures, which are too long or too small. The result is shown in Fig. 2(c).

To obtain the candidate character regions (which are of "dense" vertical textures), we first scan each row of the binary image from left to right; if two neighboring object pixels' distance is less than a threshold T_1, the two pixels are considered to be in a "cluster," and if there are more than threshold T_2 object pixels in a "cluster," we use a line to connect the object pixels together, as shown in Fig. 2(d). After all the rows in the image being processed, CCA is used to get the candidate regions, and the regions that are too large or too small will be removed. The final localization result is shown in Fig. 2(a). There are also some noise regions (the two regions below) that will be eliminated in the following character segmentation phase.

IV. Character Segmentation

In these located regions, it is inevitable to have some false regions that contain other characters or some character-similar objects. Thus, the main purpose of character segmentation is to segment out the CCCs and eliminate other characters or character-similar objects. Although various methods for character segmentation were proposed, the CCC segmentation is more challenging, as the align modes of CCCs are various, and there are cases that some CCCs are missing. As a result, for CCC segmentation, we propose the "segmentation confidence" concept, which is the probability of the segmented characters (e.g., objects) belonging to the container code. Based on the segmentation confidence, it is easy to determine whether the segmented characters are the CCCs, even if some characters are missing.

A. Binarization Method

Before using the "segmentation confidence" to determine whether the segmented characters (e.g., objects) belong to the container code, the binarization process is first applied to obtain the characters in each candidate region. As the obtained images may be affected by uneven illumination, a local binarization method is more suitable. Here, we adopt the "Logical Level Technique (LLT)" method [25]. In this method, a pixel is determined to be in the foreground or not through logical analysis between the pixel and its neighbor pixels. The key parameters include the character stroke width SW and the threshold T. For the parameter SW, $SW = H_{\mathrm{ROI}} \times 0.12$ is used to give an estimation, where H_{ROI} stands for the input region's height. For the threshold T, a fixed value 8 is adopted.

The binarization result is shown in Fig. 3(c); Fig. 3(b) also gives the binarization result of the global thresholding method of Otsu for reference. In the binary image, the CCA technique is used to segment the characters (e.g., character-similar objects) and eliminate other noise objects whose height or width exceeds the desired limits. Each segmented character is identified by its bounding box, as shown in Fig. 3(d).

Fig. 3. Binarization methods comparison. (a) Gray image. (b) Binary result by Otsu's method. (c) Binary result by LLT. (d) Characters marked with their bounding boxes.

Fig. 4. Segmentation result for a region group. (a) Region group. (b) Character segmentation result.

Fig. 5. Eleven-state HMM.

B. HMM-Based Segmentation Confidence

After the candidate regions have been obtained, the adjacent regions are first grouped. As shown in Fig. 4(a), the three candidate regions are grouped, and we mark them as region 1, region 2, and region 3 from top to bottom. When the objects in each region are extracted, their position in the original image can also be obtained, as shown in Fig. 4(b). Then, the next task is to find the 11 characters whose arrangement conforms to some known CCCs' align mode. For example, in Fig. 4, region 1 and region 2 have 11 characters, whereas regions 2 and 3 also have 11 characters. However, only the characters in regions 1 and 2 are of the arrangement conforming to one of the known align modes. As a result, the characters in regions 1 and 2 are to be extracted, whereas the characters in region 3 shall be eliminated as noises. To solve this problem, a model that decribes the CCCs' align mode should first be formed.

Based on the facts that the 11 CCCs are aligned from left to right and from top to bottom and, in most cases, each character's position has a close relationship to its neighbors, we propose to use a hidden Markov model (HMM) chain to model the align mode of the CCCs. In this HMM, there are 11 hidden states, and each state stands for a character's ordinal number, i.e., the hidden state 1 stands for the first character, the hidden state 2 stands for the second character, etc. These hidden states can only transit from 1 to 11 in order, as shown in Fig. 5, where the transition from state 1 to state 2 is approved, whereas the transition from state 1 to state 3 or from state 2 to state 1 is not allowed. Adopting the 11 hidden states, the single-transition model has two reasons, i.e., first, we can easily define different statistical properties for characters at different positions (as each character corresponds to a distinct hidden state) and, second, the computational complexity can be largely reduced (as there is only one transition probability for each state).

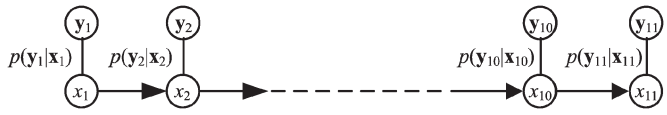


Fig. 6. HMM chain for CCCs.

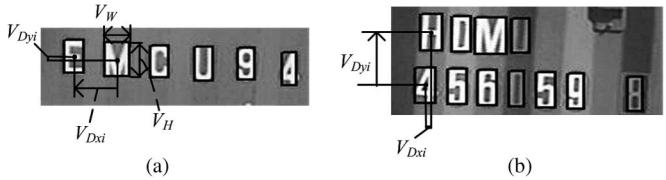


Fig. 7. Observation representation. (a) Four features. (b) Features for the object at the start of a new line.

The HMM chain for the 11 CCCs is shown in Fig. 6, where $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{11})$ stands for the observation sequence, i.e., the characters' spatial positions, and $\mathbf{X} = (x_1, x_2, \ldots, x_{11})$ stands for the hidden states.

We first consider the situation in which all the 11 CCCs are segmented out. Under this situation, the HMM output probability is represented as follows:

$$P(\mathbf{Y}) = \sum_{\mathbf{X}} P(\mathbf{Y}|\mathbf{X})P(\mathbf{X})$$
$$\mathbf{X} = (x_1, x_2, \ldots, x_{11}) \tag{4}$$

which can be further written as

$$P(\mathbf{Y}) = \sum_{\mathbf{X}} \prod_{i=1}^{11} p(\mathbf{y}_i|x_i) \left[p(x_1) \prod_{j=1}^{10} p(x_{j+1}|x_j) \right] \tag{5}$$

where $p(\mathbf{y}_i|x_i)$ is the emitting probability of the hidden state x_i, and $p(x_{j+1}|x_j)$ is the transition probability from hidden state x_j to hidden state x_{j+1}.

As the model is constructed based on the distribution characteristics of CCCs, it is natural to define the output probability $P(\mathbf{Y})$ as the segmentation confidence, i.e., the probability of the segmented characters belonging to the container code. To calculate the output probability, we still have to choose a more specific mathematical representation for the model.

C. Representation for the HMM

To get the specific mathematical representation of the aforementioned HMM, proper representation for the observations should first be provided. As an object found in the candidate region is identified by its bounding box, we use four features to describe each object's spatial position, i.e., the object's width V_W, height V_H, the horizontal center point distance to the previous object V_{Dx}, and the vertical center point distance to the previous object V_{Dy}, as shown in Fig. 7(a). [However, if the object is at the start of a new line, the V_{Dx} should be the horizontal center point distance to the previous line's first object, and the V_{Dy} should be the vertical center point distance

to the previous line's first object, as shown in Fig. 7(b).] As a result, the observation $\mathbf{y}_i$ can be described as $\mathbf{y}_i = (V_{Wi}, V_{Hi}, V_{Dxi}, V_{Dyi})$.

Still, we have to select the forms of the transition probability $p(x_{i+1}|x_i)$ and the emitting probability $p(\mathbf{y}_i|x_i)$. In the proposed HMM (as shown in Fig. 5), the state transition is fixed to the only path that is from the current state to its next state in order. Thus, for simplicity, we define

$$p(x_1) = \begin{cases} 1, & \text{if } x_1 = 1 \\ 0, & \text{otherwise} \end{cases}$$
$$p(x_{i+1}|x_i) = \begin{cases} 1, & \text{if } x_{i+1} = x_i + 1 \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

As a result, the HMM output probability [see (5)] can be rewritten as

$$P(\mathbf{Y}) = \prod_{i=1}^{11} p(\mathbf{y}_i|x_i), x_1 = 1, x_2 = 2, \ldots, x_{11} = 11. \quad (7)$$

In addition, for emitting probability $p(\mathbf{y}_i|x_i)$, we use four independent Gaussian distributions to describe the probability, i.e.,

$$\begin{aligned} p(\mathbf{y}_i|x_i) &= \prod_{s=1}^{4} N\left(\mathbf{y}_i^{(s)}; \mu_{x_i}^{(s)}, \sigma_{x_i}^{(s)}\right) \\ &= \exp\left(-|V_{Wi} - \mu_W|^2/2\sigma_{Wi}^2\right) \\ &\quad * \exp\left(-|V_{Hi} - \mu_H|^2/2\sigma_{Hi}^2\right) \\ &\quad * \exp\left(-|V_{Dxi} - \mu_{Dx}|^2/2\sigma_{Dxi}^2\right) \\ &\quad * \exp\left(-|V_{Dyi} - \mu_{Dy}|^2/2\sigma_{Dyi}^2\right) \end{aligned} \quad (8)$$

where $N(\mathbf{y}_i^{(s)}; \mu, \sigma)$ stands for Gaussian distribution with the mean value of μ and the covariance of σ; $\mathbf{y}_i^{(s)}$ stands for one of the four features, i.e., $(V_{Wi}, V_{Hi}, V_{Dxi}, V_{Dyi})$, of the ith character; and μ_W, μ_H, μ_{Dx}, and μ_{Dy} are the average character width, the average character height, the average horizontal distance, and the average vertical distance of current segmented characters, respectively. Each σ_{si} in (8) represents the standard deviation of feature s under the condition of hidden state x_i. Each σ_{si} is calculated by the corresponding feature's average value μ_s multiplied by a coefficient ζ_{si}. The coefficient ζ_{si} represents the normalized standard deviation and is learned from the training images, i.e.,

$$\zeta_{si}^2 = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\sigma_{si}^{(j)}}{\mu_s^{(j)}}\right)^2 = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{V_{si}^{(j)}}{\mu_s^{(j)}} - 1\right)^2. \quad (9)$$

In (9), j stands for the jth training image, n is the number of the training images, and "s" can stands for W, H, Dx or Dy. For example, ζ_{Wi} is the standard deviation of the ith character's width, assuming the average character width to be 1, and σ_{Wi} can be written as $\sigma_{Wi} = \zeta_{Wi} \times \mu_W$.

However, there are still some special cases. For the object at the start of a new line, as shown in Fig. 7(b), the μ_{Dx} used in (8) is set to 0, the μ_{Dx} used in (9) is set to μ_W, and the μ_{Dy} used in (8) and (9) are both set to the average vertical center point distance between the corresponding characters in the two lines.

After all these parameters being obtained, (7) can be calculated by

$$\begin{aligned} P(\mathbf{Y}) = \prod_{i=1}^{11} \Big\{ & \exp\left(-|V_{Wi} - \mu_W|^2/2\sigma_{Wi}^2\right) \\ & * \exp\left(-|V_{Hi} - \mu_H|^2/2\sigma_{Hi}^2\right) \\ & * \exp\left(-|V_{Dxi} - \mu_{Dx}|^2/2\sigma_{Dxi}^2\right) \\ & * \exp\left(-|V_{Dyi} - \mu_{Dy}|^2/2\sigma_{Dyi}^2\right) \Big\}. \end{aligned} \quad (10)$$

However, the probability in (10) would have a very small value; therefore, we use the normalized form as the segmentation confidence, i.e.,

$$\hat{P}(\mathbf{Y}) = P(\mathbf{Y})^{1/44}. \quad (11)$$

With the help of the logarithm, the computational complexity can be reduced. After the $P(\mathbf{Y})$ being calculated, a threshold T is used to determine whether the segmented characters belong to the container code region or not.

As previously mentioned, the CCCs' distributions have many forms, which will inevitably cause different parameters for different forms.

D. Multiple Character Align Modes

The CCCs in container images can be vertically or horizontally aligned and can be arranged in one line or multiple lines. In addition, the characters of equal distances, which are also called successive characters, can be grouped into a segment. Based on the number of segments and the number of characters in each segment, we can define some align modes [2], such as 1COL_11, 1ROW_11, 1ROW_4-7, 2ROWS_4_7, 2ROWS_4_6-1, 4ROWS_4_3_3_1, etc. The first part of each mode tells the align orientation and the number of lines, e.g., "1COL" stands for one vertical text line, and "3ROWS" stands for three horizontal text lines. The following parts are the numbers of successive characters with equal spaces and the separators "_" / "–," where "_" stands for different text lines, and "–" represents different text segments in the same text line. For example, "2ROWS_4_6-1" means that there are three parts of the container code aligned in two horizontal text lines with four characters in the first text line and six successive characters (of similar distances) plus one character apart in the second text line. Each mode represents a kind of character distribution of the standard container codes. Each mode should have its own set of HMM parameters, which are defined in (9). Thus, we classify our training images into classes according to these modes and train the parameters for each mode.

For a given segmentation result, estimating its segmentation confidence is a problem to maximize the output probability, i.e., $\max_{AM_i} P(\mathbf{Y}|AM_i)$, where AM_i stands for each possible CCCs' align mode, and $P(\mathbf{Y}|AM_i)$ is (10) calculated using AM_i's set of parameters.

In practice, the characters' most probable align mode is directly estimated. First, the characters of equal distances are grouped into a segment. Whether the characters $\{c_{i-1}, c_i, c_{i+1}\}$ are of equal distances is determined by

$$|V_{Dxi} - V_{Dx(i+1)}| < \delta * \frac{\left(V_{Dxi} + V_{Dx(i+1)}\right)}{2} \tag{12}$$

where V_{Dxi} is defined in Fig. 7(a), and δ is an adaptive parameter. Second, based on the number of segments and the number of characters in each segment, the most probable align mode AM_p can be known. In addition, the segmentation confidence $P(\mathbf{Y}|AM_p)$ can then be calculated using the align mode AM_p's set of parameters.

E. Missing Characters' Situation

If there are some characters missing, i.e., not all 11 CCCs appear in the input image, the remaining characters should still be extracted, and the missing characters' positions (e.g., ordinal number) should be estimated. (Then, in further character recognition phase, these missing characters can be replaced with a wildcard, such as "*".) Based on the segmentation confidence previously described, it is natural to propose that the missing characters should be placed to maximize the segmentation confidence. To achieve this, the spaces where the missing characters can be placed should first be detected.

Given two neighboring characters, how many characters can be inserted into their space is calculated based on the distance between them, i.e.,

$$n_i = V_{Dx_i}/\mu_{Dx} - 1. \tag{13}$$

If $n_i > 0$, the space is marked as a possible insertion position with the attribute n_i, which indicates how many characters can be inserted into the space. In addition, at the beginning or at the end of a text line, an arbitrary number of characters can be inserted.

The next step is to insert the missing characters into these possible spaces. Obviously, there are various insertion possibilities. For each possibility, its align mode can be calculated, and if the align mode is not a known align mode for the CCCs, the insertion possibility should be eliminated. After this step, we have several estimations of the missing characters' positions. For each insertion possibility, the missing characters' observation $\mathbf{y}'_i$ should be chosen to maximize the segmentation confidence. After calculating the segmentation confidence for each insertion possibility, the one with the maximum segmentation confidence is selected as the final estimation.

Take Fig. 8 for example, we only find nine characters, i.e., two characters are missing. The next step is to estimate where the two missing characters are. Of course, the head and tail of the candidate line can always possibly place the two missing objects. In addition, from the distance analysis, there is a space that can accommodate one object between the fifth object ("3") and the sixth object ("9"). Based on this information, an HMM chain is created, as shown in Fig. 9(a), where the white nodes are for observations, the black nodes are for hidden states, and the gray nodes with '?' indicate the possible positions at which

Fig. 8. Container code with missing characters.

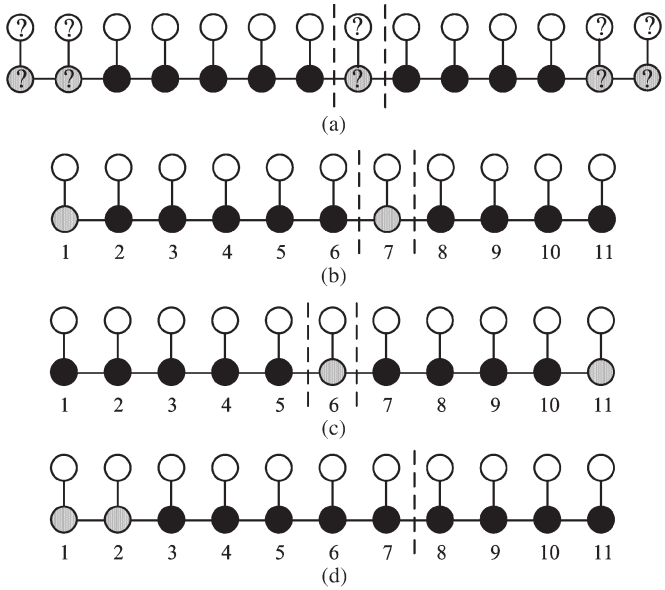


Fig. 9. Insertion probabilities for Fig. 8. (a) Insertion probabilitiy analysis. (b)–(d) Three insertion probabilities.

to insert the missing nodes and observations. (Note that the vertical dashed lines in Fig. 9 are just marks for the position between the fifth object ("3") and the sixth object ("9") in the container image of Fig. 8.) As a result, there are many possibilities to place the two missing objects, and Fig. 9(b)–(d) are three possibilities, where the gray nodes stand for the inserted nodes, and the 11 hidden states have been assigned to the corresponding nodes in order. In addition, after placing the two missing objects, the character align mode should conform to one of those CCC align modes. Thus, we can eliminate some "noises," such as Fig. 9(d), which compose an align mode of 1Row_7-4, which is a non-CCC align mode. After this process, there are only two possibilities left, which are shown in Fig. 9(b) and (c). After the segmentation confidence under each possibility being calculated, the maximum one is chosen as the final estimation. In this experiment, the character arrangement shown in Fig. 9(b) produces a segmentation confidence of 0.94, whereas the character arrangement shown in Fig. 9(c) produces a segmentation confidence of 0.87. As a result, the first one will be chosen as the final estimation.

If the final result is larger than a threshold T, the existing characters will be identified as CCCs.

F. More Discussion About the Missing Characters' Situation

The container code includes 11 characters, in which the first four characters are letters, and the later seven characters are digits. Based on the ordinal number of each existing character, we can choose to feed each existing character to the letter recognition module or number recognition module, and we can use a wildcard character (such as "*") to represent each missing

Fig. 10. Container code images with CCC segmentation results.

character. The recognition result can serve as a clue to search in a container code database, which records a list of all existing container codes, to obtain the proper container code.

In practice, there are sometimes several different insertions that can produce similar segmentation confidence. This is particularly the case when the missing character occurs at the head or tail of the container code, as placing the insertion character at the head or tail is of little difference from the perspective of character arrangement. In these cases, we can provide both possible insertion results. In addition, in the further recognition phase, the recognition result with the fourth character to be "U" can be selected as the final result, as the fourth character in the container code is fixed to "U."

V. Experiment Results

Although the proposed method is mainly about the horizontally aligned container codes, it can be easily adapted to vertically aligned cases. The slight changes include the localization process being vertically performed and the align modes being accordingly modified.

To evaluate our proposed method, 1020 container images obtained in a real port environment are tested. Fig. 10 shows some image examples used in this experiment. (The black boxes are the character segmentation results.) These images are of 640 (width) $*$ 480 (height) pixels. Both of the images obtained in day and night are included. Some of the images are affected by uneven illumination. In different images, the CCCs' sizes largely vary. The minimum character is about 20 pixels in height, whereas the maximum character is about 50 pixels in height. All images are obtained by the front cameras. However, the images may have some rotation angles. The container code in the images is of various align modes and may have some characters missing.

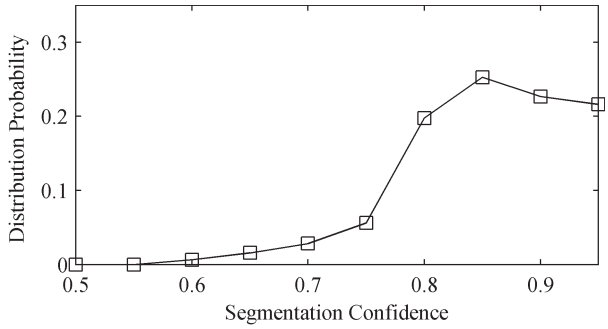


Fig. 11. Distribution of the CCCs' segmentation confidence.

A. Training Phase

The purpose of the training phase is to obtain the deviation parameters described in (9). For each align mode, we select at least 20 images, in which all of the 11 CCCs appear, to train the parameters. However, to make the value of the total probability more in line with our intuitive, in our experiment, we multiply the value of $\zeta_{x_i}^2$ [which is calculated by (9)] by 2.5. The parameters are set to allow each component probability $\exp(-|V_{si} - \mu_s|^2/2\sigma_{si}^2)$ to be roughly 0.8 at its mean deviation. Fig. 11 shows the distribution of the CCCs' segmentation confidence. This is the statistics of the real CCCs. In Fig. 11, each square symbol (x, y) represents the proportion y of those images, whose segmentation confidence values are in the range of $[x, x + 0.05)$, to the all images. As the objects segmented in the noise regions are seldom of the align mode similar to the CCCs', the segmentation confidence values for those non-CCC objects are very small. As a result, we choose 0.6, which is almost the minimum segmentation confidence value for real CCCs, as the threshold to determine whether the segmented characters belong to the container code.

B. Test With Real Container Code Images

After all the parameters being obtained, our proposed method is performed on the 1020 real container images. In the container code region localization phase, 1003 container code images' code regions have been successfully located. (The located regions may also contain other noise regions.) We also select the DWT-based method [6] as a reference, which successfully located the code regions in 884 images.

Figs. 12 and 13 give an intuitive comparison of the two methods. (In Figs. 12(b) and 13(b), for better visibility, we perform a simple linear grayscale extension on the original image.) From Figs. 12(c) and 13(c), we can see that our method can be more robust under uneven illumination condition.

In the character segmentation phase, 972 container code images' CCCs are successfully segmented out. The segmentation

(a) (b) (c)

Fig. 12. Localization result by the DWT-based method. (a) Localization result. (b) HL subband of one-level DWT. (c) Binary result (after noise removal).

(a) (b) (c)

Fig. 13. Localization result by our method. (a) Localization result. (b) Vertical texture. (c) Binary result (after noise removal).

TABLE I
EXPERIMENT RESULTS

Missing characters	0	1	2	3	>3
Total Images	976	22	12	8	2
Our Method	939	19	9	5	0
He's Method [2]	927	0	0	0	0

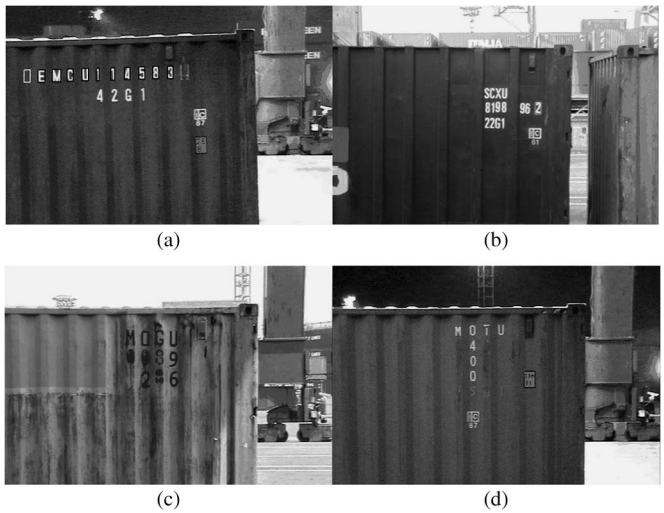
rate is 95.3%. We also select He's method [2] as a reference, which extracts the characters of the container code for 927 container images and provides a segmentation rate of 90.9%. The detailed results are given in Table I.

The differences between our method and He's method [2] are given as follows: 1) Our method can extract CCCs, even if there are some characters missing. 2) He's method uses character boundary distance to describe the character distribution; however, when there's character "1" in the image, the distance is sometimes not very stable; in our method, character center point distance is used, and we also considered the characters' width and the characters' height, which makes our method more robust. 3) In He's method, the code region localization result is based on horizontal projection, and no further operation is employed to define the code region's left and right boundaries, which will introduce more noise.

It should be noted that, if there are too many characters missing, our method will not try to extract the remaining characters either. As in this case, the process will be more time consuming and will introduce more false segmentation results. In our experiment, we only process the container codes with at most three characters missing, which can cover the most cases in real container images.

We present some segmentation examples with missing characters estimation in Fig. 14. The black boxes are the segmented characters of the container code, and the white boxes are the estimated missing characters. In Fig. 14, we can robustly see the proposed method works for the CCCs of various align modes, including characters aligning in one line, two lines, and multiple lines. The CCCs in these images are successfully segmented out, and the missing characters' position is properly estimated. The results indicate that our proposed method can

Fig. 14. Successful examples with missing character estimation.



(a) (b) (c) (d)

Fig. 15. Unsuccessful examples. (a) Missing character estimation error. (b) Abnormal arrangement of CCCs. (c) Image with too much noise. (d) Too many character missing.

effectively tackle the missing characters' situation, which is rarely considered in present literatures.

However, Fig. 15 also shows some unsuccessful examples. In Fig. 15(a), although the ten CCCs in the image are successfully segmented out, the missing character's position is not properly estimated. This problem can be corrected according to the discussion in Section IV-F. Fig. 15(b) shows a container code image with abnormal arrangement of CCCs. As the align mode only appears twice in our tested images and is not defined in our algorithm, the segmentation result is rejected. Fig. 15(c) shows an image with too much noise on it, which makes our algorithm fail to extract the characters out. In Fig. 15(d), there are too many characters missing, and our algorithm will omit the remaining characters.

The average processing time of the method is lower than 300 ms, on a personal computer (Celeron 2.13G, 256M Ram). In most cases, the container code recognition systems are used to recognize a stationary or slow-moving container; as a result, this method's performance can meet real applications' time demand.

VI. Conclusion

The HMM-based segmentation confidence has been proposed to describe the probability of the segmentation results belonging to the container code. The proposed method can work not only under the situation that all the 11 CCCs are segmented out but also under the situation that some characters are missing, which is seldom considered in other literatures. With the existing CCCs being recognized and each missing CCC being represented by a wildcard character (such as "*"), the result can serve as a clue to search in a container code database to obtain candidate container codes.

Considering the computational efficiency of the algorithm and the trustiness of the estimated result, we mainly focused on tackling the situations wherein no more than three characters are missing. For those images with too many characters missing, we can also try to extract the CCCs from the other side of the container, such as top-view image or back-view image, which may contain more complete information. It is also one of our future directions that provides more robust ACCR result from container code images of multiple-views. In this phase, the segmentation confidence can also give a clue of the characters' segmentation quality.

Another thing to be mentioned is that this work can be compatible with newly emerging CCCs' align modes, through introducing these align modes' corresponding sets of parameters into the framework. This work can also be applied to other applications, such as license plate recognition by changing the number of HMM states and redefining the characters' align modes.

References

[1] C. Oberli, M. Torres-Torriti, and D. Landau, "Performance evaluation of UHF RFID technologies for real-time passenger recognition in intelligent public transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 748–753, Sep. 2010.

[2] 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.

[3] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 377–391, Sep. 2008.

[4] B. Hongliang and L. Changping, "A hybrid license plate extraction method based on edge statistics and morphology," in *Proc. ICPR*, 2004, pp. 831–834.

[5] D. Zheng, Y. Zhao, and J. Wang, "An efficient method of license plate location," *Pattern Recognit. Lett.*, vol. 26, no. 15, pp. 2431–2438, Nov. 2005.

[6] Y. R. Wang, W. H. Lin, and S.-J. Horng, *Fast License Plate Localization Using Discrete Wavelet Transform*. Berlin, Germany: Springer-Verlag, 2009, ser. Lecture Notes in Computer Science, pp. 408–415.

[7] R. Al-Hmouz and S. Challa, "License plate localization based on a probabilistic model," *J. Mach. Vision. App.*, vol. 21, no. 3, pp. 319–330, Apr. 2010.

[8] C.-N. I. Anagnostopoulos, I. E. Anagnostopoulos, E. Kayafas, and V. Loumos, "A license plate recognition system for intelligent transportation system applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 377–392, Sep. 2006.

[9] T. D. Duan, T. L. H. Du, T. V. Phuoc, and N. V. Hoang, "Building an automatic vehicle license plate recognition system," in *Proc. Int. Conf. Comput. Sci. (RIVF)*, 2005, pp. 59–63.

[10] X. Shi, W. Zhao, and Y. Shen, *Automatic License Plate Recognition System Based on Color Image Processing*. Berlin, Germany: Springer-Verlag, 2005, ser. Lecture Notes in Computer Science, pp. 1159–1168.

[11] R. O'Malley, E. Jones, and M. Glavin, "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 453–462, Jun. 2010.

[12] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, "Automatic license plate recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, Mar. 2004.

[13] L. Dlagnekov (Mar. 2004), License Plate Detection Using AdaBoost, San Diego, CA. [Online]. Available: http://www.cse.ucsd.edu/classes/fa04/cse252c/projects/louka.pdf

[14] J. Xiong, S. Du, D. Gao, and Q. Shen, "Locating car license plate under various illumination conditions using genetic algorithm," in *Proc. ICSP*, 2004, pp. 2502–2505.

[15] J. Yin, A. Cao, and J. Li, "A new method for container code location," in *Proc. IEEE ICAL*, 2007, pp. 2842–2845.

[16] T.-H. Wang, F.-C. Ni, K.-T. Li, and Y.-P. Chen, "Robust license plate recognition based on dynamic projection warping," in *Proc. IEEE Int. Conf. Netw. Sense Control*, 2004, pp. 784–788.

[17] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "A novel adaptive morphological approach for degraded character image segmentation," *Pattern Recognit.*, vol. 38, no. 11, pp. 1961–1975, Nov. 2005.

[18] G. Li, R. Zeng, and L. Lin, "Research on vehicle license plate location based on neural networks," in *Proc. ICICIC*, 2006, pp. 174–177.

[19] H. Mahini, S. Kasaei, and F. Dorri, "An efficient features-based license plate localization method," in *Proc. ICPR*, 2006, pp. 841–844.

[20] V. Shapiro and G. Gluhchev, "Multinational license plate recognition system: Segmentation and classification," in *Proc. ICPR*, 2004, pp. 352–355.

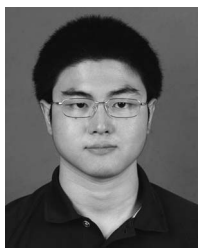
[21] S. Draghici, "A neural network based artificial vision system for license plate recognition," *Int. J. Neural Syst.*, vol. 8, no. 1, pp. 113–126, Feb. 1997.

[22] T. Naito, T. Tsukada, K. Yamada, K. Kozuka, and S. Yamamoto, "Robust license-plate recognition method for passing vehicles under outside environment," *IEEE Trans. Veh. Technol.*, vol. 49, no. 6, pp. 2309–2319, Nov. 2000.

[23] D. Llorens, A. Marzal, V. Palazon, and J. M. Vilar, *Car License Plates Extraction and Recognition Based on Connected Components Analysis and HMM Decoding*. Berlin, Germany: Springer-Verlag, 2005, ser. Lecture Notes in Computer Science, pp. 571–578.

[24] V. Franc and V. Hlavac, *License Plate Character Segmentation Using Hidden Markov Chains*. Berlin, Germany: Springer-Verlag, 2005, ser. Lecture Notes in Computer Science, pp. 385–392.

[25] M. Kamel and A. Zhao, "Extraction of binary character/graphics images from grayscale document images," *Graph. Model. Image Process.*, vol. 55, no. 3, pp. 203–217, Mar. 1993.

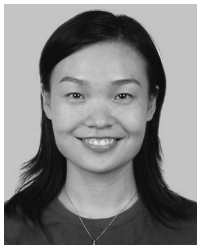


Mo Chen received the B.S. and M.S. degrees in pattern recognition and intelligent systems, in 2004 and 2007, respectively, from Sichuan University, Chengdu, China, where he is currently working toward the Ph.D. degree in communication and information systems with the College of Electronics and Information Engineering.

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

Wei Wu was born in 1975. He received the B.S. degree from Tianjin University, Tianjin, China, in 1998 and the M.S. and Ph.D. degrees in communication and information systems from Sichuan University, Chengdu, China, in 2003 and 2008, respectively.

He is currently a faculty member with the College of Electronics and Information Engineering, Sichuan University. His current research interests include image processing and video communications, wireless communications, and super resolution.



Xiaomin Yang was born in 1980. She received the B.S. and the Ph.D. degrees in communication and information systems from Sichuan University, Chengdu, China, in 2002 and 2007, respectively.

She is currently a faculty member with the College of Electronics and Information Engineering, Sichuan University. Her current research interests include computer vision and pattern recognition.

Xiaohai He received the B.S. and M.S. degrees in electronics and electrical engineering and the Ph.D. degree in biomedical engineering from Sichuan University, Chengdu, China, in 1985, 1991, and 2002, respectively.

He is currently a Professor and Ph.D. supervisor with the College of Electronics and Information Engineering, Sichuan University. He is an Editor for the *Journal of Information and Electronic Engineering* and was an Associate Editor for the *Journal of Data Acquisition & Processing*. His research interests include image processing, pattern recognition, computer vision, image communication, and software engineering.

Dr. He is a Senior Member of the Chinese Institute of Electronics.