A Computer Vision System for Automated Container Code Recognition

Hsin-Chen Chen, Chih-Kai Chen, Fu-Yu Hsu, Yu-San Lin, Yu-Te Wu, Yung-Nien Sun*

***Abstract*—Container code examination is an essential step in the container flow management. To date, this step is mostly achieved by human visual inspections, which are however time-consuming and error-prone. We hence propose a new computer vision system for automated container code recognition. The proposed system consists of model construction and code recognition stages. In the model construction stage, we first incorporate a locally thresholding method with prior knowledge of code character geometry to segment the code characters, including English characters A-Z and numeric characters 0-9, from a training set of container images. With the segmentation results of each character, we subsequently construct its Eigen-feature model using the principal component analysis (PCA). In the recognition stage, the code characters are firstly segmented from the given container image. Each segmented character is then recognized by finding the best matched Eigen-feature model that maintains the minimal PCA reconstruction error of the character appearance. Experiments showed that the proposed method achieved the code recognition with a high recognition rate and little recognition time for each image automatically. Overall, our proposed system has great potential for improving the efficiency of container terminals as well as enhancing the container management.**

***Index Terms*—code recognition, locally thresholding, character geometry, Eigen-feature model, principal component analysis**

I. INTRODUCTION

CONTAINER code examination is a critical step in the procedure of container security and flow management. Nowadays, human visual inspection, which is laborious and error-prone, remains the most common approach in port gates for checking the information of containers. Rapidly and reliably examining the container code, which is helpful for increasing the economic efficiency of container terminals, hence becomes important. For this purpose, our research aims to develop a computer vision system for automated container code recognition.

The proposed computer vision system contains two technical parts, which are code segmentation and recognition. Up to present, there have been several kinds of methods developed for image segmentation, and a popular one is the thresholding-based method [1]–[5]. Otsu's thresholding method [1] is a typical one of global thresholding. It divides the image into foreground and background regions based on the optimal computation with intensity histogram of the entire image. Basically, the Otsu's method is easy to implement and efficient in computation. However, it may not be suited for the images whose intensity histogram is not bimodal.

On the other hand, locally thresholding methods estimate a threshold value for every pixel in the image based on the intensity distribution of its neighborhood. Therefore, they are less constrained by the global intensity distribution of the image. Pradhan et al. [3] proposed a locally thresholding method with adaptive window to segment objects from the images under uneven lighting condition. Moreover, the locally thresholding techniques were employed in the applications of medical image analysis for segmenting tissues of interest, such as the works of Zhang et al. [4] and Peng et al. [5]. Overall, the thresholding-based methods segment objects mainly based on the intensity value. Hence, irrelevant regions with intensity values similar to the target objects' tend to be included in the thresholded results. To resolve this problem, features other than intensity value, should be taken into account in the segmentation protocols. For example, Sankaran et al. [6] incorporated the geometry information of cells, including area and shape, into the segmentation process and obtained satisfactory segmentation results.

As to the character recognition part, the correlation coefficient is commonly used to evaluate the similarity of grey-level distribution between two segmented character regions [7]. However, the correlation coefficient is sensitive to the pose variation between the characters. To overcome this problem, Wakahara et al. [8] proposed a character recognition algorithm based on the non-rigid registration. Unfortunately, their method suffers a lot from computational cost. The computation problem and recognition rate can be significantly improved by carefully selecting the image space of character recognition. Moreover, the eigen space is a

Manuscript received December 28, 2010. This work was supported under contract CSIST-0839-V201 (99) from the Chung-Shan Institute of Science and Technology, Taiwan, R.O.C. and utilized the shared facilities supported by the Program of Top 100 Universities Advancement, Ministry of Education, Taiwan, R.O.C..

H.C. Chen is with Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (e-mail: wale1212@gmail.com).

C.K. Chen is with Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (e-mail: heartthrob.kai@gmail.com).

F.Y. Hsu is with Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (e-mail: cash.barca@gmail.com).

Y.S. Lin is with Airborne Electronic Section, Electronic System Research Division, Chung-Shan Institute of Science and Technology, Taoyuan, Taiwan, R.O.C. (e-mail: dostoevosky@yahoo.com.tw).

Y.T. Wu is with Department of Biomedical Imaging and Radiological Sciences, National Yang-Ming University, Taipei, Taiwan, R.O.C. (e-mail: ytwu@ym.edu.tw).

Y.N. Sun is with Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (corresponding author to provide phone: +886-6-2757575 ext. 62526; fax: +886-6-2747076; e-mail: ynsun@mail.ncku.edu.tw).

widely adopted feature space for character recognition. Park et al. [9] and Manjunath et al. [10] recognized characters based on the Eigen-feature space and achieved good results. These literatures mainly focused on character recognition, and however, less discussed the details of segmentation.

In this paper, character segmentation and recognition are both addressed for automated container code recognition. The features of proposed method are described below. First, we incorporate a locally thresholding method with prior knowledge of character geometry to segment the characters from container images automatically. Second, the Eigen-feature space, which can better accommodate the noises and deformation in character appearances, is thus adopted to recognize the segmented code characters. Third, we employ the generation rule of code characters in the recognition protocol to reduce the mis-recognition rate. Experimental results showed that our proposed system could achieve promising results with a high recognition rate and low computational time.

II. Overview of The Proposed Method

The proposed method consists of model construction and code recognition stages. In the model construction stage, we firstly segment the code characters from a training set of container images by incorporating a locally thresholding method with prior knowledge of code character geometry. We subsequently construct an Eigen-feature model for each code character based on its segmentation results on the training images. Given a container image in the code recognition stage, we segment the code characters using the above-mentioned segmentation protocol. Each segmented character is then recognized by finding the best matched Eigen-feature model that maintains the minimal reconstruction error of character appearance. The details of the proposed method are described in the following sections.

III. Model Construction

A. Locally Thresholding Segmentation

Three intensity properties of container image are used to segment the container code characters. First, the code characters are with similar intensities to each other. Second, the intensity contrast of characters with respect to their surroundings is large. Third, there is possible uneven sunlight illumination in the image. Based on these observations, we hence select the locally thresholding method [11] to segment the container code characters.

The locally thresholding method is designed based on the intensity distribution of the neighborhood of every individual pixel in the image. At first, we transform the input image from RGB color to grayscale. Then, we slide a fixed-sized window over the image. For each pixel at the center of the window, we calculate the average intensity from the neighborhood pixels covered by the window. The examined pixel is classified as the foreground (with zero intensity in Fig. 1(c) and Fig. 1(d)) if its intensity is lower than c percent of the average intensity, and is assigned as the background otherwise. After this step, we can obtain the segmented

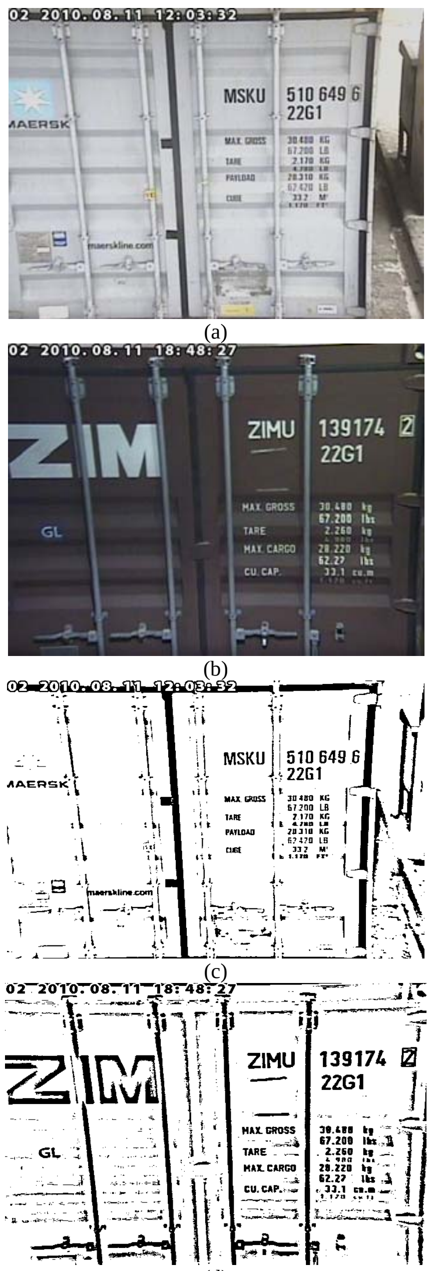

Fig. 1. Locally thresholding segmentation: (a) and (b) the original container images; (c) and (d) the segmentation results of (a) and (b) respectively.

foreground denoted as **FG**. We also employed the integral image technique to speed up the threshold computation. The length of the side of the sliding window was set to 1/30 of the image width, and c was assigned as 80. The aforementioned thresholding criterion is only suitable for the container image, in which the code character regions are darker than the surrounding regions. In some cases, the intensities of code characters, however, may be higher than those of surrounding regions. An erroneous segmentation may thus occur such that the code characters are misclassified into the background. Therefore, a confirmation strategy is designed to compensate this limitation. At first, we apply the Otsu's method [1] to the original grayscale image and obtain two clusters of pixels denoted as **C1** and **C2**. Since the container occupies most parts of the image, it is supposed to be included in the larger sized cluster (assumed to be **C1**). On the other hand, as the code characters and markers are darker or brighter than the surroundings in intensity (i.e. higher in intensity contrast) and smaller in size in the container image, the code characters are contained in the other cluster **C2**. We subsequently calculate the average intensity on the original grayscale image for **C1** and **C2** respectively. If the difference between average intensities of **FG** and **C1** is smaller than the one between **FG** and **C2**, an error of local segmentation is detected. We then change the criterion of the local thresholding and process the image again. A pixel is classified into the foreground if its intensity is higher than the average intensity of its neighborhood, and into the background otherwise. Based on the confirmation process, we efficiently obtain acceptable segmentation results of the code characters in either black or white, as shown in Fig 1.

B. Character Extraction Based on Geometry Features

The thresholding result however contains a number of non-code character regions. As the code characters are supposed to be with certain geometry properties, e.g., with the size in a certain range, we then estimate for each region several geometry parameters, including area, circumference, width, and height. If the values of parameters of a region are out of the empirically determined ranges, it is filtered out, and otherwise is preserved. After that, those irrelevant regions such as the levers on the container door can be easily removed, as shown in Fig. 2(a). Next, it is further considered that the code characters are located close together. We thus design a distance-based grouping algorithm to pick out the code character regions:

```
Step 1.  Label all the regions as FALSE and set their
         grouping states to UNGROUPED.
Step 2.  Label an arbitrary FALSE region as TRUE.
Step 3.  If there is any FALSE region satisfying the
         following two rules:
         (1) The distance between its center and the centers
         of any TRUE & UNGROUPRED regions is
         smaller than one-hundred pixels,
         (2) The difference of x- or y-axis coordinates
         between its center and the center of any TRUE &
         UNGROUPRED regions is smaller than eight
         pixels,
         then the examined region is labeled as TRUE.
Step 4.  Go to step 3 until there are no more TRUE regions
         found.
Step 5.  Group these TRUE & UNGROUPRED regions
         together, and set their grouping state to
         GROUPED.
Step 6.  Go to step 2 until there are no more groups found.
```

After the grouping process we can obtain several groups as shown in Fig. 2(b). The code character regions can be identified by the largest sized group indicated by the red rectangle, and each individual character can also be readily extracted (see Fig. 2(c)).

C. Eigen-feature Models

Using the proposed segmentation method, we can automatically extract the character regions from the training

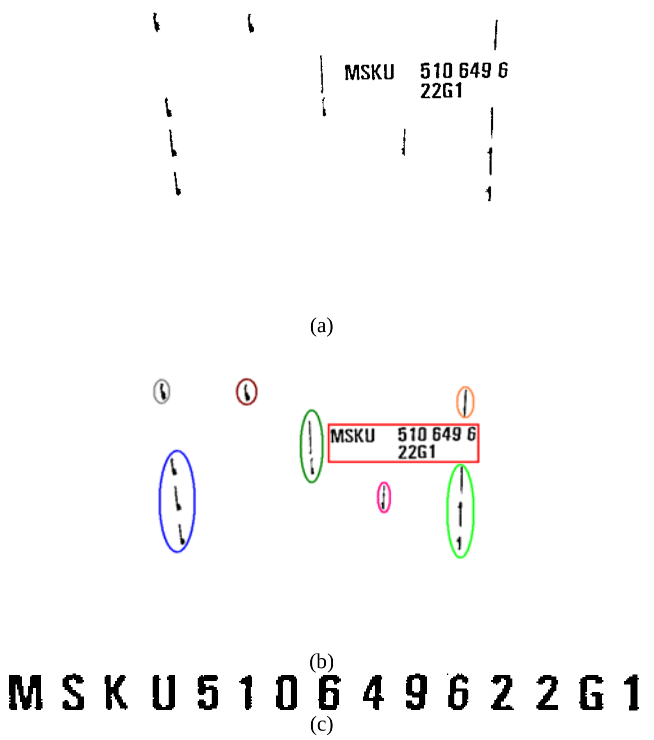

Fig. 2. Character extraction from Fig. 1(c): (a) the filtering result by geometry features; (b) the grouping result by character distance; (c) the character extraction result.

images. For each English and numeric character, we construct an Eigen-feature model based on the PCA to characterize its appearance variation among different training images. In the model construction process, its binary segmentation results are first aligned to each other to eliminate the pose differences. The alignment results then serve as the training samples, denoted as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_{m-1}, \mathbf{x}_m)$, where m is the number of training samples. Each training sample $\mathbf{x}_i$ is represented by a one-dimensional vector $(x_1, x_2, \cdots, x_j, \cdots, x_{n-1}, x_n)^T$, where x_j is the intensity value of the j-th pixel in the training sample and n is the number of pixels. Next, we average the m samples to obtain the mean vector $\bar{\mathbf{x}}$, and then calculate the covariance matrix $\mathbf{C}$ implying the variance between each training sample and the mean vector. And $\mathbf{C}$ is given by $\frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$. We subsequently solve the eigenvalues $(v_1, v_2, \cdots, v_i, \cdots, v_{n-1}, v_n)$ and their corresponding eigenvectors $(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_i, \cdots, \mathbf{u}_{n-1}, \mathbf{u}_n)$ from the covariance matrix. Given the eigenvalues in descending order, the dimension of the eigen features is reduced by preserving the first t components, subject to $(\sum_{i=1}^{t} v_i \Big/ \sum_{i=1}^{n} v_i) > 0.95$. At last, the resulting Eigen-feature model, characterized with the mean vector and eigenvectors, can be utilized to identify the character appearance. By applying this construction process to all the characters including 0-9 and A-Z, we can consequently obtain a set of Eigen-feature models and apply them to the subsequent recognition stage.

IV. Code Recognition

In the recognition stage, we first segment the code characters from a given image. Each unknown character is then compared to all Eigen-feature models, and is recognized by finding the best matched model which maintains the minimal PCA reconstruction error of the character appearance. The details of character recognition algorithm are delineated below.

Step 1. Input an unknown character $\mathbf{C}_{\text{unknown}}$ (i.e., a one-dimensional intensity vector) and the set of Eigen-feature models.

Step 2. Select an Eigen-feature model and project the character vector $\mathbf{C}_{\text{unknown}}$ from the image space onto the model space, $\mathbf{y} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_t \end{bmatrix} [\mathbf{C}_{unknown} - \bar{\mathbf{x}}]$.

Step 3. Reconstruct the character appearance through a back projection process, $\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{y} + \bar{\mathbf{x}}$.

Step 4. Calculate the reconstruction error, $\left\| \mathbf{C}_{unknown} - \hat{\mathbf{x}} \right\|$.

Step 5. Record the value of the error, and then go to step 2 until all the Eigen-feature models are examined.

Step 6. Find the character with the minimal reconstruction error as the recognition result of $\mathbf{C}_{\text{unknown}}$.

An example of the character recognition is demonstrated in Fig. 3. The unknown character is shown in Fig. 3(a), and the reconstructed appearances using the Eigen-feature models of ‘2’, ‘3’, and ‘7’ are displayed in Figs. 3(b)-(d), respectively. It is observed that using the corresponding Eigen-feature model to reconstruct the character appearance can lead to the smallest distortion. After all the characters are recognized, we can obtain the recognition result of the entire container code.

On the other hand, to increase the recognition rate of the proposed system, we further incorporate the information of container code generation rule into the recognition process based on the standard of ISO 6346 [12]. The generation rule provides the information that a character is supposed to be matched to either numeric or English characters, as shown in Table I. For example, the first three characters of the container code represent the owner id in English. Hence, only English characters are taken as feasible solutions for their recognition. Consequently, the mis-recognition between the numeric and English characters with similar appearance, e.g., ‘1’ and ‘I’, can be easily avoided.

V. Results and Discussion

The following experiments consisted of three parts including segmentation accuracy, recognition accuracy and computational performance evaluations. The validation work included 94 container images and was performed on a

Fig. 3. Example of the character recognition: (a) the unknown character; (b)-(d) the reconstructed appearances using three different Eigen-feature models.

TABLE I
Code Generation Rule for The Proposed Recognition Algorithm

1st Row		
Code characters	Specific meaning	Character type
The first three characters	Owner id	English character
The fourth character	Category identifier	English character
The last one character	Check digit	Numeric character
The other characters	Serial number	Numeric character
2nd Row		
Code characters	Specific meaning	Character type
The first two characters	Size code	Numeric character
The second character	Type code	English character
The last character	Type code	Numeric character

desktop PC with 3.0GHz Intel Core 2 Duo E8400 processor, windows XP Professional, and 2GB memory.

A. Segmentation Accuracy Evaluation

To evaluate the segmentation accuracy, the proposed method was applied to the 94 validation images. A successful segmentation in an image has to satisfy that every character (totally 15 characters in a container code) is included in the segmentation result. The success rate in these segmentations was 88.3 % (83/94), indicating a satisfactory accuracy of the proposed segmentation method.

B. Recognition Accuracy Evaluation

In this experiment, the recognition accuracy with respect to individual character and entire code was validated. At first, we counted the number of total occurrences of each character in the 83 successfully segmented images. For each individual character, its recognition success rate (RSR) was given by the proportion of the number of correct recognitions to the total occurrences, as listed in Table II. The average RSR of all the characters was 98.22 %. Moreover, it was observed that the RSRs of two characters, ‘B’ and ‘N’, were much lower than the others’. This may be because their sample sizes are very small. If more testing images are included, the RSRs are supposed to increase.

Moreover, we also validated the recognition accuracy with respect to the entire container code. We calculated the number of images, in which the code with 15 characters was correctly recognized. 73 successful recognitions were obtained among these 83 images, that is, 87.95 % recognition rate. If more data can be included in the future experiments, the recognition rate is expected to be further improved.

C. Computational Performance Evaluation

In this experiment the computational performance of the proposed system was also evaluated. For each of the 83 images, we measured the computational time in code segmentation and recognition, and obtained an average computational time less than 1 second. Overall, the proposed system can achieve the segmentation and recognition of container code efficiently.

VI. Conclusion

In this paper we have proposed a computer vision system for automated container code recognition. We employed the locally thresholding method and character geometry features to automatically segment the container code from the given image. In the segmentation process we further designed an intensity-based strategy to compensate the error of locally thresholding segmentation. In addition, the Eigen-feature models were utilized in the code recognition process for accommodating image noises and deformation in character appearances. And, the generation rule of container code was taken into account to avoid mis-recognitions between English and numeric characters with similar appearance. The experimental results showed that our proposed method can automatically achieve code recognition with a high average RSR. In the future research, more container images will be included to fine-tune the models in this system for improving the recognition rate. Moreover, the recognition of markers in container image (e.g., triangular or rectangular signs) will also be conducted for constructing a more complete examination system.

TABLE II
Recognition Success Rate (RSR) of Individual Character

Character	Number of occurrences	Number of correct recognitions	RSR
0	71	69	97.18 %
1	164	162	98.78 %
2	176	176	100 %
3	68	68	100 %
4	114	113	99.12 %
5	89	89	100 %
6	56	56	100 %
7	63	63	100 %
8	58	56	96.55 %
9	59	59	100 %
A	10	10	100 %
B	4	3	75 %
C	24	24	100 %
D	2	2	100 %
E	2	2	100 %
F	9	9	100 %
G	103	102	99.3 %
H	12	12	100 %
I	11	11	100 %
J	0	0	None
K	10	10	100 %
L	35	35	100 %
M	27	26	96.3 %
N	5	4	80 %
O	39	39	100 %
P	4	4	100 %
Q	0	0	None
R	9	9	100 %
S	23	23	100 %
T	15	15	100 %
U	94	93	98.94 %
V	0	0	None
W	2	2	100 %
X	3	3	100 %
Y	9	9	100 %
Z	7	7	100 %
Average RSR			98.22 %

References

[1] Otsu, N., “A threshold selection method from gray-level histograms,” *IEEE Trans. Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[2] Gonzalez, R.C., and Woods, R.E., *Digital Image Processing*. 3rd ed, Pearson Ed. New Jersey, USA, 2010, pp. 760–785.

[3] Pradhan, S.S., and Nanda, P.K., “Adaptive thresholding based image segmentation with uneven 11 lighting condition,” *IEEE Region 10 Colloquium and the third ICIIS 2008*, pp. 1–6.

[4] Zhang, J., Yan, C.H., Chui, C.K., and Ong, S.H., “Fast segmentation of bone in CT images using 3D adaptive thresholding,” *Computers in Biology and Medicine*, 40, 2010, pp. 231–236.

[5] Peng, J.Y., and Hsu, C.N., “Adaptive local thresholding for fluorescence cell micrographs,” Technical Rep. No. TR-IIS-09-008, Nov. 11, 2009.

[6] Sankaran, P., & Asari, V.K., “Adaptive thresholding based cell segmentation for cell-destruction activity verification,” in 2006 *IEEE Conf. AIPR*, pp. 14.

[7] Ozbay, S., and Ercelebi, E., “Automatic vehicle identification by plate recognition,” *Proceedings of World Academy of Science, Engineering and Technology*, 2005, pp. 222–225.

[8] Wakahara, T., and Odaka, K., “Adaptive normalization of handwritten characters using global/local affine transformation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1332-1341, Dec. 1998.

[9] Park HS, Kim SY, Lee SW, Gray-scale handwritten character recognition based on principal features. SPIE vol. 3027, pp. 40–49, 2000.

[10] Manjunath AVN, Hemantha KG, and Noushath S., “Two-dimensional matrix principal component analysis useful for character recognition,” ICIA, pp. 390–393, 2006.

[11] Bradley, D., and Roth, G., “Adaptive thresholding using the integral image,” *Journal of Graphics, GPU, & Game Tools*, vol. 12, no. 2, pp. 13–21, 2007.

[12] *Container Handbook: Cargo loss prevention information from German marine insurers*, ch. 3.4. [Online]. Available: http://www.containerhandbuch.de/chb_e/stra/index.html?/chb_e/stra/stra_03_04_00.html