I.J. Image, Graphics and Signal Processing, **2013, 7, 54-63**
Published Online June 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2013.07.08

A Novel Container ISO Code Localization Using an Object Clustering Method with Opencv and Visual Studio Application

Ronesh Sharma
Department of Electronics Engineering, Mokpo National University, Mokpo, South Korea
sharmaronesh@yahoo.com

Seong Ro Lee
Department of Information & Electronics, Mokpo National University, Mokpo, South Korea
srlee@mokpo.ac.kr

Abstract— An automatic container code recognition system is of a great importance to the logistic supply chain management. Techniques have been proposed and implemented for the ISO container code region identification and recognition, however those systems have limitations on the type of container images with illumination factor and marks present on the container due to handling in the mass environmental condition. Moreover the research is not limited for differentiating between different formats of code and color of code characters. In this paper firstly an object clustering method is proposed to localize each line of the container code region. Secondly, the localizing algorithm is implemented with opencv and visual studio to perform localization and then recognition. Thus for real time application, the implemented system has added advantage of being easily integrated with other web application to increase the efficiency of the supply chain management. The experimental results and the application demonstrate the effectiveness of the proposed system for practical use.

Index Terms— Object clustering, Opencv, Container code localization, Image segmentation, Container recognition

I. Introduction

In recent years, the amount of containers being transported has been increased due to the shipment demands with the increasing size of ships and also with the low transportation cost and charges [1]. To this end thousands of containers need to be registered while it arrives and when discharged from the terminal every day. However till now, manual registration of container numbers has been in practice, where the terminal clerks note the numbers. Unfortunately, errors can be encountered in noting and this paradigm tends out to be very slow for handling thousands of containers each day. Thus an efficient automated system needs to be implemented for the container terminals to recognize container numbers and to expedite the process with fewer errors to improve the accuracy as well as to improve the productivity of the supply chain management. Moreover a high level of security is needed for the supply chain management.

To some extend many research works have been carried out to automate as well as to simplify the recognition process at the container terminals, however those are subjected to certain constraints. Due to the outdoor environmental factors of the containers, marks present on the container with some missing characters, damaged containers and the illumination condition turns out to affect the recognition process. The researchers have previously developed methods of character extraction with the use of histogram threshold methods [2], using a top-hat morphology operation [3, 4] with horizontal pulses for the edges of the characters in the horizontal direction [5]. Using fuzzy logic and applying a sober mask to noise-removed images [6] are both alternatives to edge detection and histogram threshold [7]. Liu [8] proposed the method of scan lines and object filter to extract an object region, while Koo [4] used top hat, bot-hat and a spatial window to detect each format but lacks differentiating between different formats of code and the color of codes character. Furthermore many methods have been proposed for detecting each format and each color of code separately, also the accuracy is low when the methods are combined as a whole for real time application. Moreover the performance of the system decreases when blurry and damaged container images appear.

In this paper, firstly we propose a container code localization method based on object clustering. We use the information of object size, object dimension, anobject location to determine the presences of ISO container code in the image. Using the above information we are able to identify which format of the code is present in the image, and even we are able to identify the color of code characters. Finally we feed the selected container code region to an OCR engine for recognition. Secondly we present the overview of

implementing real time container code recognition system with opencv and visual studio. When compared with other research works, our method shows the robustness in detecting container code in the presence of marks on the container, with missing characters and even when blurry images are involved. Moreover our method is able to differentiate between different formats of code and identify which color of code character present. Using opencv as vision processing software for developing ISO container code detection and recognition algorithms with visual studio, it easily gives access to other applications and hardware for being integrated.

II. OVERVIEW OF THE SYSTEM

A. Visual Studio and Opencv

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services [9]. OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez [10]. It is free for use under the open source BSD license. The library is cross-platform; it focuses mainly on real-time image processing. We integrate opencv libraries with visual studio to process Container serial number localization. With Visual studio on its own we can create other web application to enhance the system.

B. Types and Format of Code Printed on Container

The code is always printed in black or white color and the code format is either horizontal or vertical form with vertical being 2 line code and horizontal of 2 or 3 line code. Fig.1 shows several code types and formats.

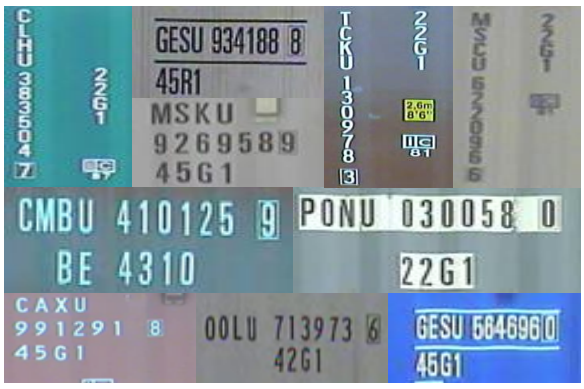


Figure.1 Format and types of code

C. System Design

A code region localization technique is being proposed in this paper; the algorithm detects the container code region location from the captured image and is able to differentiate between different formats of code, and even the color of the code character. The algorithms are developed with Visual studio together with opencv for implementation of real time container terminal application.

III. PROPOSED ALGORITHM

The proposed architecture for localizing is shown in Fig.2 It takes the grayscale image as input and returns the output image as selected clusters of the ISO code area region and code region bounding location information. The system architecture shown is achieved by the method of applying binary threshold, object clustering with analyzing objects area, width, height and length and finally setting a code location ratio condition between each line of code forming the container code region. The ISO container code has 4 general formats of vertically format white color, horizontally format white color, vertical format black color, and horizontal format black color code. In our architecture we propose to threshold and cluster the objects, then to filter the objects in the image and search for vertically format white color code first, then search for the other 3 formats respectively. Once the code is detected, the other stages of searching the other formats are skipped.

A. Searching for vertical format white color code

The program thresholds the input grayscale image to a binary image to obtain white color objects from the image, each object detected in the image is threshold with the area, width, and height of the object with the threshold values. The threshold object is then connected to each other using Morphological closing operation with the structuring element constructed to form vertically connected image. Fig.3 shows a grayscale image(a), the white color objects detected images(b), counter area threshold image(c), object width/height threshold image(d),and vertically object connected image(e) respectively.

Those connected objects are then again threshold for its width and height with the cross ponding threshold values for a vertical format code. If still there remains some objects in the image, further to verify that vertical code exists, using bounding rectangle, all objects height and width are calculated, since for the code to exist in the image there should be at least one long connected object with threshold height. This threshold height is the average value for the longest height of the vertical code. Therefore if one of the objects height is detected above the threshold value, then all the objects in the images are arranged according to decreasing height of the objects in vector form. According to the heights, the top 3 largest height object is chosen and their locations are checked with the condition as shown in Fig.4.

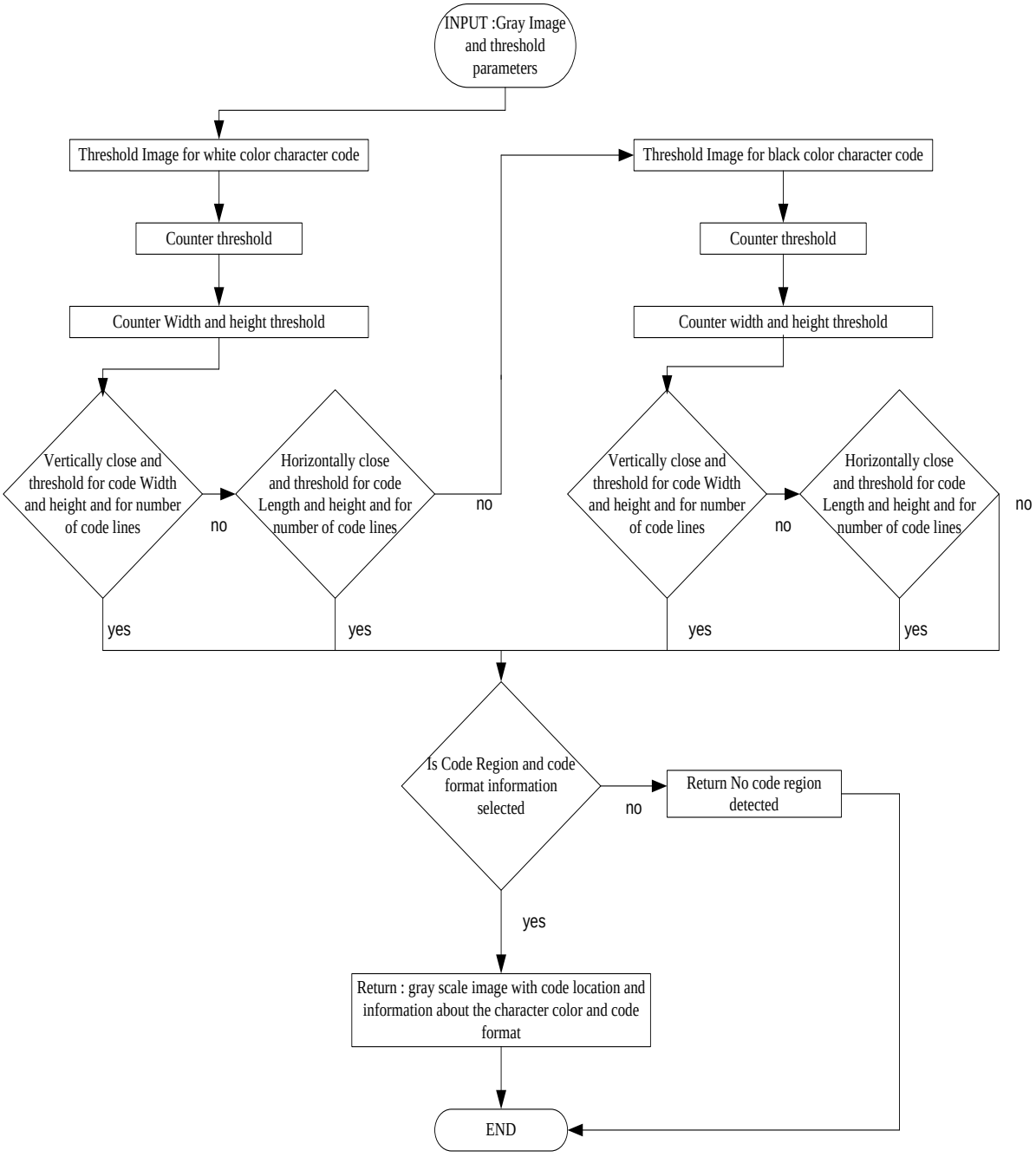


Figure.2 Code region localization method

Once the condition is checked, and if the number of objects in the image is equal to either 2 or 3 then only it is verified that vertical code is detected with white color code characters. Finally if code is detected, bounding box is drawn at the location of the selected area of code and the system returns the images with location information, code format and color of code character. Fig.5 shows the threshold image for vertical code(a) and the region selected on the gray image(b) if detected respectively. The next section will proceed with steps when code is not identified as vertically format white coder code.

B. Searching for horizontal format white color code

If the code is not detected in the last section, the next step is to search and identify if it is horizontal format white color character code. From the last section, using the output image of objects area, width and height threshold, each object in the image is connected to each other using Morphological closing operation with the structuring element constructed to form a horizontally connected image. Fig.6 shows a grayscale image(a), the white color objects detected image(b), counter threshold image(c), object width and height threshold image(d), and horizontally object connected image(e) respectively.

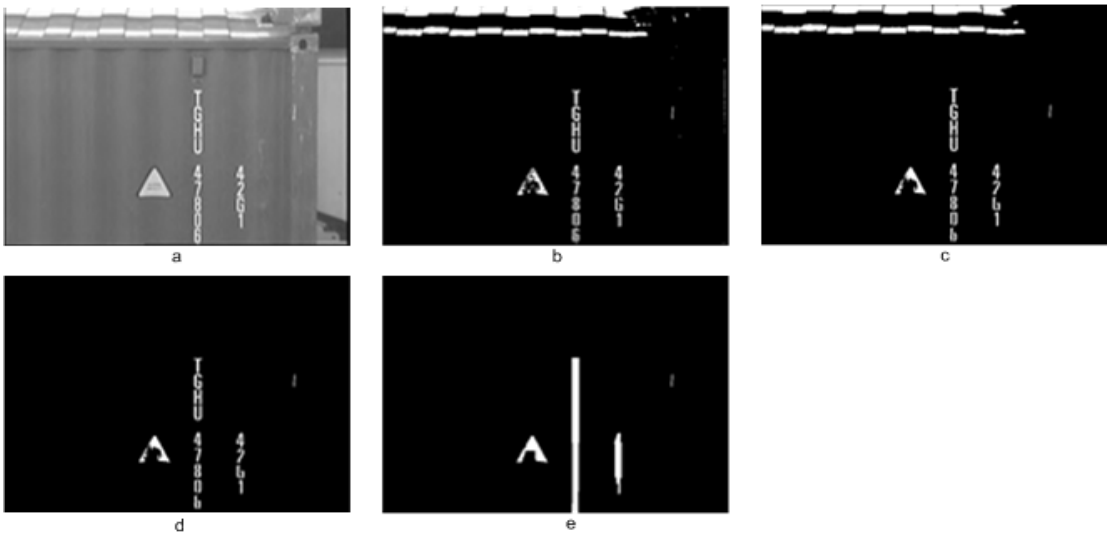
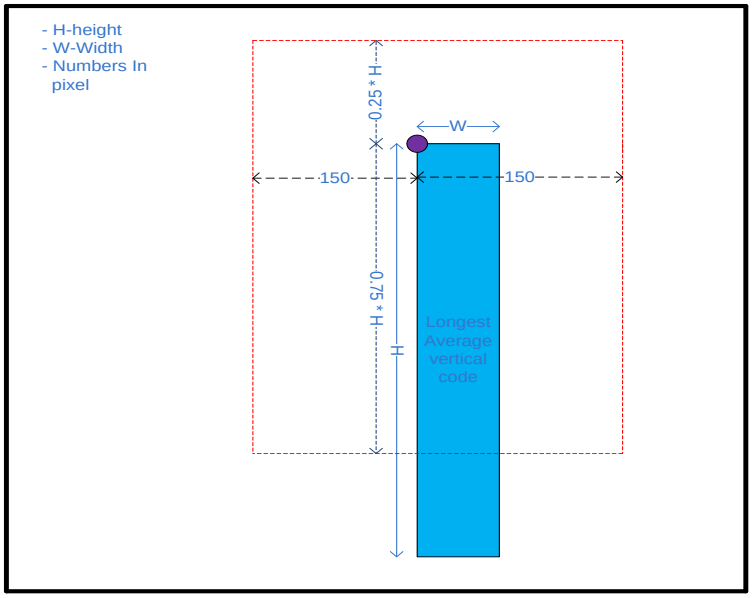


Figure.3 Searching for vertical format white color code



- Image with black border line
- Blue rectangle shows average longest vertical code
- Purple mark on rectangle represents the top left corner of the vertical code
- Longest height code always exists on left side of the image
- Red border line rectangle shows the inside position of the image where the second vertical codes top left corner can exist
- The second code is choose where by its top left y axis value is smaller then that of other code lines selected in the region

Figure.4 Condition for vertical format code location

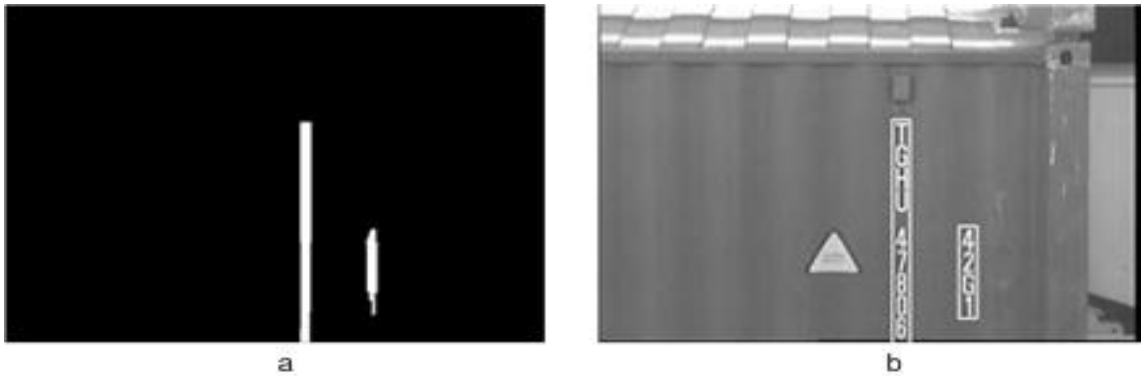


Figure.5 Threshold image for vertical format code with region located

Those connected objects are then again threshold for its length and height with the cross ponding threshold values for the horizontal code. If still there remains some objects in the images, further to verify that horizontal code exists, using bounding rectangle, all objects height and length are calculated, since for the code to exist in the image there should be at least one object with threshold length. This threshold length is the average

value for the longest length of the horizontal code. Therefore if one of the object's length is detected above the threshold value, then all the objects in the images are arranged according to decreasing length of the objects. According to the lengths, the top 5 objects are chosen and their locations are checked with the condition as shown in Fig.7.

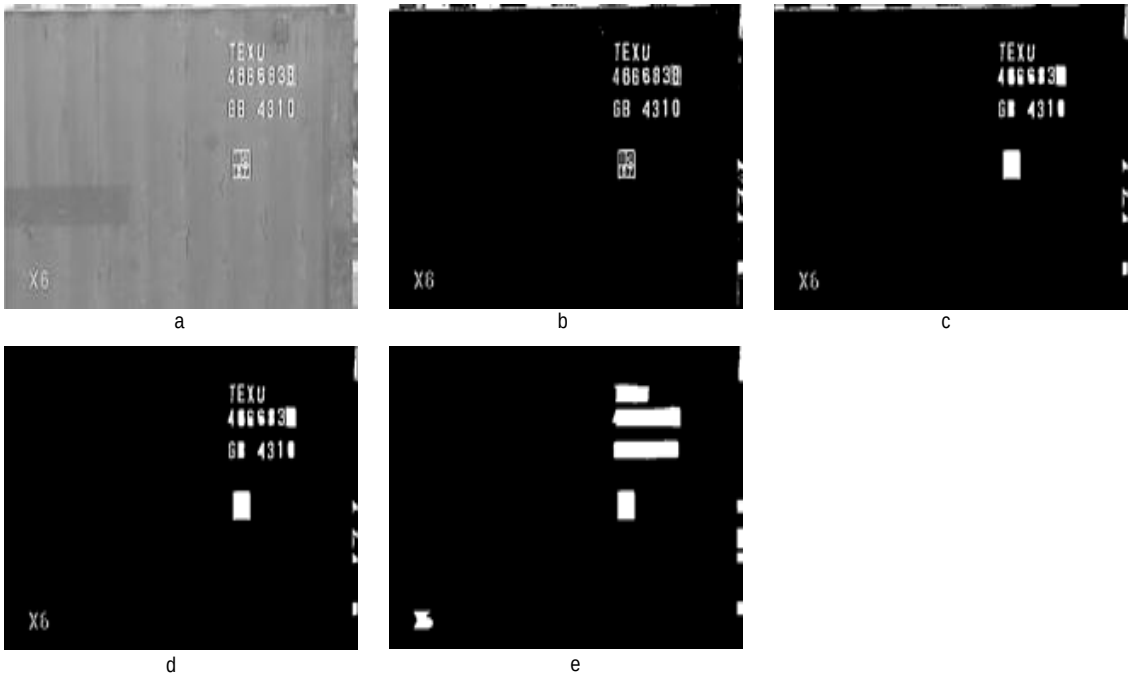
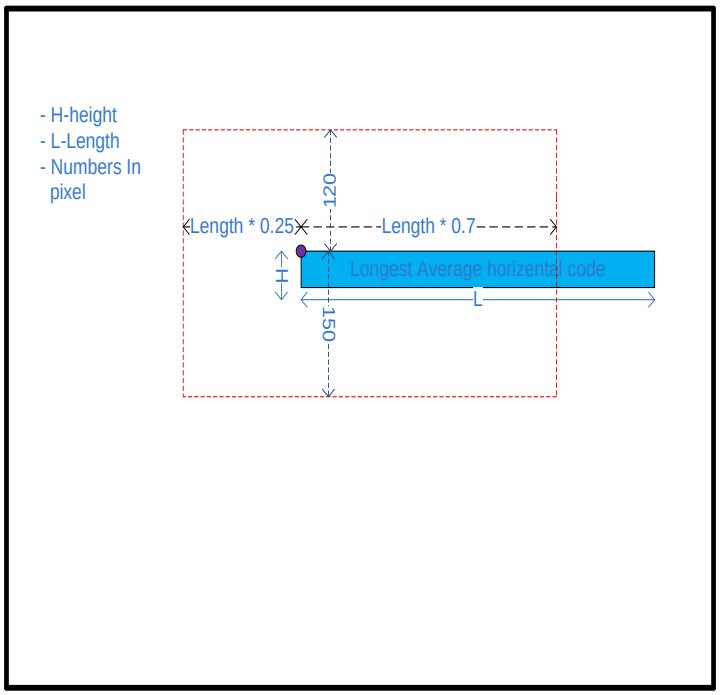


Figure.6 Searching for horizontal format white color code



- Image with black border line
- Blue rectangle shows Average longest horizontal code
- Purple mark on rectangle represents the top left corner of the horizontal code
- All code line must have similar height respect to longest line code with uncertainty of 0.75*height(longest)
- Red border line rectangle shows the inside position of the image where the other vertical codes top left corner can exist
- A threshold distance of 2.5*height(longest) is set for vertical distance between each line of code

Figure.7 Condition for horizontal format code location

Once the condition is checked, and if the number of objects in the image is either more than 1 or less than 5 then only it is verified that horizontal code is detected with white color character. Finally if code is detected, bounding box is drawn on the location of the selected area of code and the system returns the images with location information, code format and color of code character. Fig.8 shows the threshold image for horizontal code(a) and the region selected on the gray image(b) if detected. The next section will proceed with steps when code is not identified as vertical or horizontal format white color code respectively.

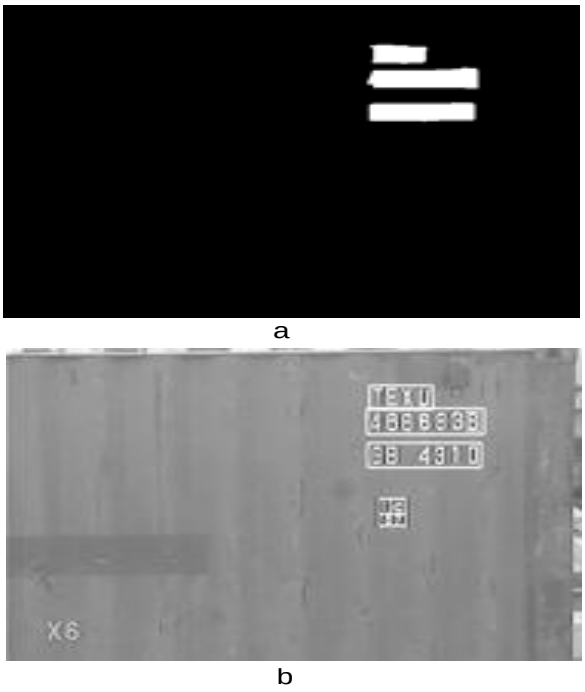


Figure.8 Threshold image for horizontal code with region

C. Searching for vertical format black color code

If the code is not detected in the last two sections, the next step is to threshold the input image to obtain black color objects from the image. After threshold, the algorithm for identifying vertical format code is repeated while eliminating the process of object width and height to identify if vertical format black character code exists. If the code is detected, the function returns the outputs otherwise the program proceeds to the next section to check if horizontal format black color character code is present.

D. Searching for horizontal format black color code

If the code is not detected in the previous three sections, the output image of counter area from the last section is taken, height and width threshold is performed and the algorithm for identifying horizontal format code is repeated to identify if horizontal format black character code exists, if the code is detected the function returns the output otherwise the program proceeds to ask the end user to enter the code.

E. Methods to detect objects in an image

For detecting white color objects, the algorithm uses the threshold binary function to threshold the grayscale image to a binary image by selecting all pixel values between 210 to 255 as white objects and below 210 as all black objects. The resulting output turns to be all white color objects detected. Direct threshold method for detecting black color objects turns to be worst when blurry images are involved. Thus in the algorithm, morphological operation bot-hat is being used to detect black color object. We use the size of structuring elements as 15 pixels.

This transformation is formed by performing closing operation on the image and then subtracting the original images from it. This transformation is best suited even in the case of images being very blurry.

IV. Experiment Simulation and Result Analysis

388 images were taken to test the code region detection algorithm for real time implementation, 54 of them were averaged good quality images with characters properly printed with good background and 334 of them were averaged medium quality images with some blurry images, some character missing, damaged container, extra marks on the container, and with uneven position of container image capturing. Fig.9 and Fig.10 shows examples of average good quality images and examples of average medium quality images respectively.

54 good quality images were sent for code region location with the localization algorithm, for 53 container images the ISO code was 100 percent correctly localized while one did not. This one failure resulted from touching characters of vertical white format code.

334 medium quality images were sent for code region location, 312 were 100% correctly localized while 22 did not. From those 22, 8 were recognized except for one character with scratches, 4 were recognized except for missing character in the starting of the code or at the beginning of the code. The 10 failures resulted from extra marks on the container; very bury images, uneven position of the camera capturing the image, extra missing characters, and extra and uneven symbol position on the container. Even in some case the full region of code is not detected but the main 11 characters and numbers for the container to be recognized are detected. Thus the experiment results shows for good quality images, the localizing algorithm has 100% accuracy, while for container images falling in medium class and in state of recognizable condition with little marks and missing characters, accuracy of 97%. Fig.11 shows some example of the code region being detected from real time images.

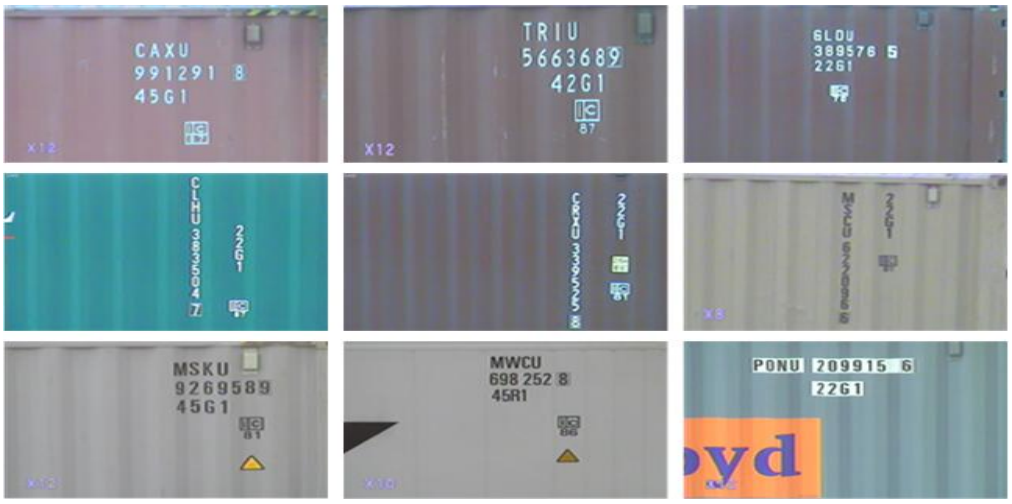


Figure.9 Average good quality images

Figure.10 Average medium quality images

Figure.11 Localized code region location

For images falling in very low quality with character missing, damaged container, and with extra marks present, the code region location accuracy decrease to 91.5%. On average scale the code localization algorithm takes 200ms to detect the code present in the image using a dual core 4GHz processor with 3 GB Ram.

Symbols are mostly present on the face side of the container, the proposed algorithm is able detect the code

region with the presents of symbols. In Fig.12(a&c) the code region is detected successfully excluding the symbols, while in Fig.12(b) the symbol is selected with code region. The location of the symbols has a great impact on the selection of code region from the image, in case of Fig.12(b&d) the preprocessing algorithm can deduce the code and exclude the symbol, but if the symbol is placed on top side of the code or either on the right or left side and is selected with the code region then preprocessing algorithm would not be able to correctly extract the code characters.

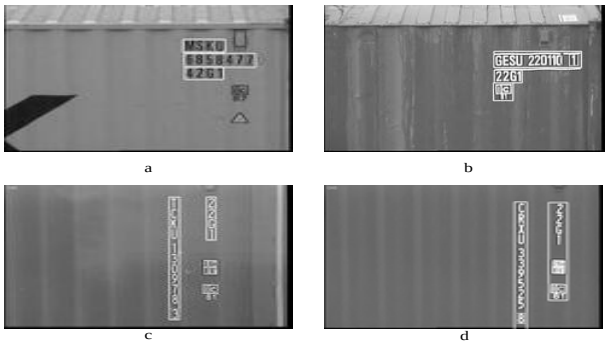


Figure.12 Selected code region with other objects present

When the distance between the camera and the container increases or either the code character size printed on the container is smaller than the threshold, the code region is not selected or some other region is selected as shown in Fig.13. In worst condition when some regions behaves in a similar structure as the code it is then selected as a code region as shown in Fig.13(a&d). The proposed algorithm detects the code according to its threshold values and the condition as described in section I.

In our current work, we worked out on selecting one code region from the image which matches with the stated condition and threshold to the maximum. In our future work will restructure our algorithm to detect all code present in the image first, then to select the code which is needed for recognition when the system is implemented in a real time application in the container terminal. A decision can be made for choosing a container code from the image with considering the container trucks approaching the real time scenario with realizing the center of the camera view and choosing the appropriate code.

Certain times due to unbalance of container truck or other factors the code presented in the image is tilted at an angle. The tilting mechanism increases the height of the code and decrease the probability of the of code selection since the selection of the code region highly depends on the code height. Fig.14 shows the tilted code increases the height of the code, though the code region is selected. The preprocessing algorithm will correct the inclination angle, however to increase the probability of selecting code region in our future work will design a robotic pan & tilt mechanism to rotate and tilt the camera to capture the container image so that the code inclination angle with the x axis is 0 degrees.

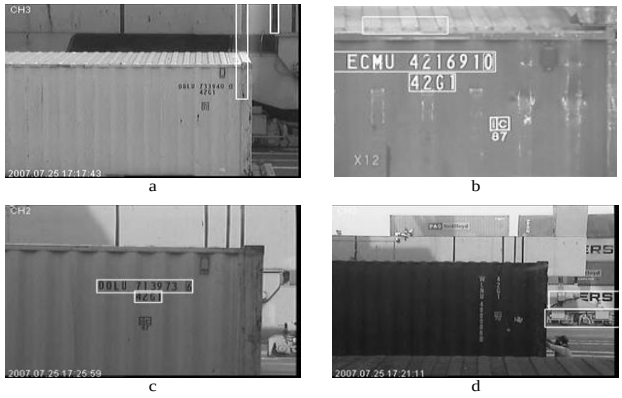


Figure.13 Selected code region with other envirnment objects present

Figure.14 Conatiner code ilted at an angle

The handling of containers in the mass environmental condition leads to containers being damaged, code characters missing, with scratches and marks present on the container. Fig.15(a) shows the algorithm locates the code region with rusted marks present on the image. The missing characters lead to code not being selected fully, if the missing code characters fall in starting or end of code line then the code is selected as partly as shown in Fig.15 (f). If the missing characters fall in the middle of the code line then, the full code region is selected as a result shown in Fig.15(e&d) and further the recognition and preprocessing algorithm needs to post process and recognize the missing characters.

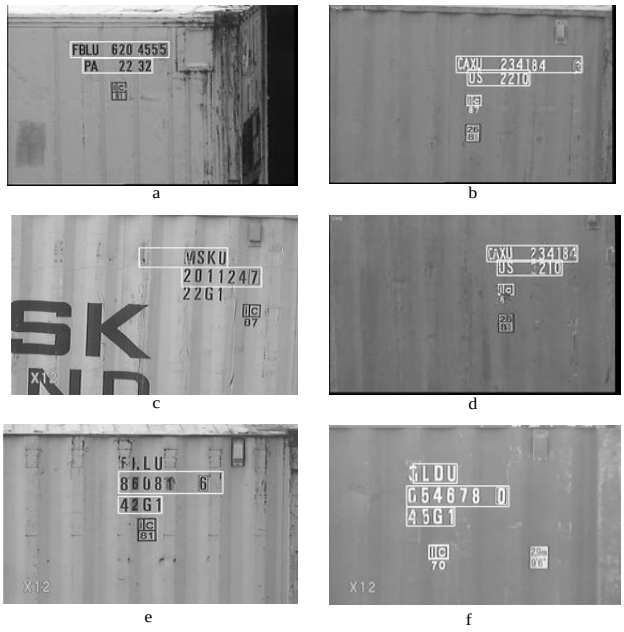


Figure.15 Conatiner code with missing code characters

V. Conclusion

We present a method for localizing the container code region location with the application of real time images using an object clustering method. It is an efficient method for detecting container ISO serial code in the presence of uneven environmental condition and even when dealing with noisy container images. Moreover our method identifies the format of the code and the color of code characters. In this paper a reasonable application of container identification system was developed using opencv, computer vision software with visual studio for real time application. This system has added advantage of being easily integrated with other application. With this robustness and performances, this application should be more convenient compared to other similar applications. With such application of opencv and visual when integrated for container code identification and recognition systems, the accuracy can be further increased by directly controlling the acquisition system and also other web application can be implemented with the system to increase the productivity of the supply chain management.

Acknowledgment

The work was supported by Priority Research Centers program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2009-0093828) and MKEC (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) supported program supervised by the NIPA, National IT Industry Promotion Agency (NIPA-2012-H0301-12-2005).

References

[1] S. Kumano and K. Miyamoto, "Development of a Container Identification Mark Recognition System*", Electronics and Communications in Japan*, 87(12), 2004.

[2] R. 0hlander, K. Price, and D. R. Reedy, "Picture Segmentation Using A Recursive Region Splitting Method", *Computer Graphics and Image Processing*, 8: p. 313-333,1978.

[3] I. S. Igual, A. P. Jimenez, and G. A.Garica, *Preprocessing and Recognition of Characters in Container Codes*, 16th International Conference on Pattern Recognition, August 11-15, 2002.

[4] K. Koo and E. Cha, "A Novel Container ISO-Code Recognition Method using Texture Clustering with a Spatial Structure Window", *International Journal of Advanced Science and Technology*, 41, 2012.

[5] J. X. Wwan and Z. Zhou, *The Research and Realization of Vehicle License Plate Character Segmentation and Recognition Technology*, International Conference on Wavelet Analysis and Pattern Recognition, Portugal, July 10-14, 2010.

[6] K. K. Baek, W. Y. Woon, and Y. H. Kyu, *An Intelligent System for Container Image Recognition Using ART2-Based Self-organizing Supervised Learning Algorithm*, in Simulated Evolution and Learning, Springer Berlin Heidelberg, p. 897-904, 2006.

[7] S. H. Ong, N. C. Yeo, and K. H. Lee, *Segmentation of color images using a two-stage self-organizing network*, Image and Vision Computing, 20(4): p. 279-289, 2002.

[8] W. Wei, L. Zheng, and C. Mo, "An automated vision system for container-code recognition" *Expert Systems with Applications*, 39(3): p. 2842-2855, 2012.

[9] Wikipedia. Microsoft Visual Studio, Decmber 15; Available from: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio.

[10] Wikipedia. opencv,Decmber 20; Available from: http://en.wikipedia.org/wiki/opencv.

[11] C. John and M. Lee, *Automatic character recognition for moving and stationary vehicles and containers in real-life images*, International Joint Conference on Neural Networks, Washington, DC, July 16, 2010.

[12] N. Murthy and K. swamy, "A Novel Method for Efficient Text Extraction from Real Time Images with Diversified Background using Haar Discrete Wavelet Transform and K-Means Clustering", *International Journal of Computer Science*, 8 ,2011.

[13] G. Bradski and A. Kaehler, M. Loukides, *Computer Vision with Opencv Library*, O'Reilly Media, Tokyo, 2008.

[14] M,Mori., ed. *Character Recognition*, Sciyo, Croatia ,2010.
[15] Wikipedia. ISO 6346. ,Decmber 18; Available from: http://en.wikipedia.org/wiki/ISO_6346.

AUTHORS

R.Sharma Sep. 2011 ~ present MS Student, Dept. of Electronics Engineering, Mokpo National University, South Korea. Nov. 2007 Received Bachelor of Engineering Technology from University of the South Pacific, Fiji <Research Area> Image Processing, Pattern Recognition, Wireless Communication, M2M and Robotics application.

S.R.Lee Sep. 1997 ~ present Professor at Mokpo National University, Department of Information & Electronics Engineering. Aug. 1996 ~ Received Ph.D. form Korea Advanced Institute of Science and Technology. Feb. 1990 ~ Received Master of Engineering from Korea Advanced Institute of Science and Technology. <Research Area> USN / Telematics applications, Embedded systems, Digital communication systems, Mobile and Satellite communication systems.