Blob Extraction based Character Segmentation Method for Automatic License Plate Recognition System

Youngwoo Yoon, Kyu-Dae Ban, Hosub Yoon, and Jaehong Kim
Robot/Cognition System Research Department
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, Korea

***Abstract*— A character segmentation algorithm for automatic license plate recognition is presented in this paper. Character regions are selected through binarization, connected component analysis, and character recognition. A blob analysis operation excludes noisy blobs, merges fragmented blobs, and splits clumped blobs. A character segmentation module achieved an accuracy rate of 97.2%. The recognition accuracy of the complete system with license plate localization was 90.9%. In depth analysis of failure cases is also provided for better understanding of the algorithm and a future development direction.**

ANPR; license plate; character segmentation; character recognition

I. INTRODUCTION

Identification of cars is a demanding function for surveillance and control systems. People can recognize automobiles through license plates which consist of alphabets and numbers. We can use the uniqueness of a combination of characters in license plates for many purposes. For example, an arrest of a suspect's vehicle, imposing parking violation fines, and entrance authentication are possible. However, it is a labor intensive job to identify all passing or parked vehicles' license plates. There was the demand of unmanned license plate recognition (LPR) system, and the development of digital cameras and machine vision algorithms made possible automatic LPR. It is obvious that the automatic LPR system should have few errors. The system is related to authentications, making charges, and law enforcements which are not tolerant to errors. Some LPR systems [1], [2] achieved satisfactory accuracies in restricted conditions, and recent studies are trying to increase the recognition accuracies and to reduce the restrictions [3], [4].

Most automatic LPR systems consist of license plate localization (LPL), character segmentation (CS), and character recognition (CR) modules as shown in Fig. 1. This paper focuses on the CS module which finds character regions from license plate images. Many CS methods are introduced before. Anagnostopoulos et al. [5] reviewed the related studies of vertical and horizontal projections, mathematical morphology, contour tracking, local/adaptive thresholding, and utilizing spatial and temporal information for the CS. Two most popular methods are using vertical projection [6-8] and connected component analysis (CCA) [9], [10]. These studies are reporting quite good segmentation accuracy, but there is a lack of algorithm description, evaluation, and result analysis. In addition, the segmentation algorithms in the previous studies take few steps that are not enough to handle low quality images. In this study, we made a more robust CS module by adopting the previous techniques of CCA, vertical projection and local binarization methods. The CR module is also utilized for better segmentation results. The next section briefly describes the license plate localization module. The following sections contain the details of the CS algorithm and experimental results with in depth analysis of failure cases.

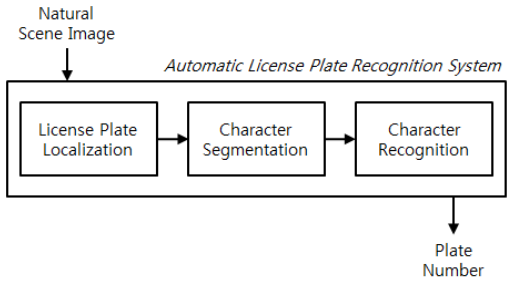

Figure. 1. General framework of the automatic license plate recognition system.

II. LICENSE PLATE LOCALIZATION

A development of the LPL module is not in the scope of the paper so we briefly describe the module. The LPL module finds license plates from a natural scene images. The module uses an appearance based character detector. The detector is trained with many character images appeared in license plate images. The detector finds characters on the license plate by inspecting every regions of the image as sliding the inspection window. In this scheme, the census transform feature is used to reduce an effect of illumination variations, and the Adaboost algorithm is used for the classification of the feature vector. The detailed description of the algorithm can be found in the face detection study [11]. After finding regions of continuously positioned characters, the module expands the regions to a license plate by the flood fill method which finds pixels of

This research was supported by the Converging Research Center Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2010K001126)

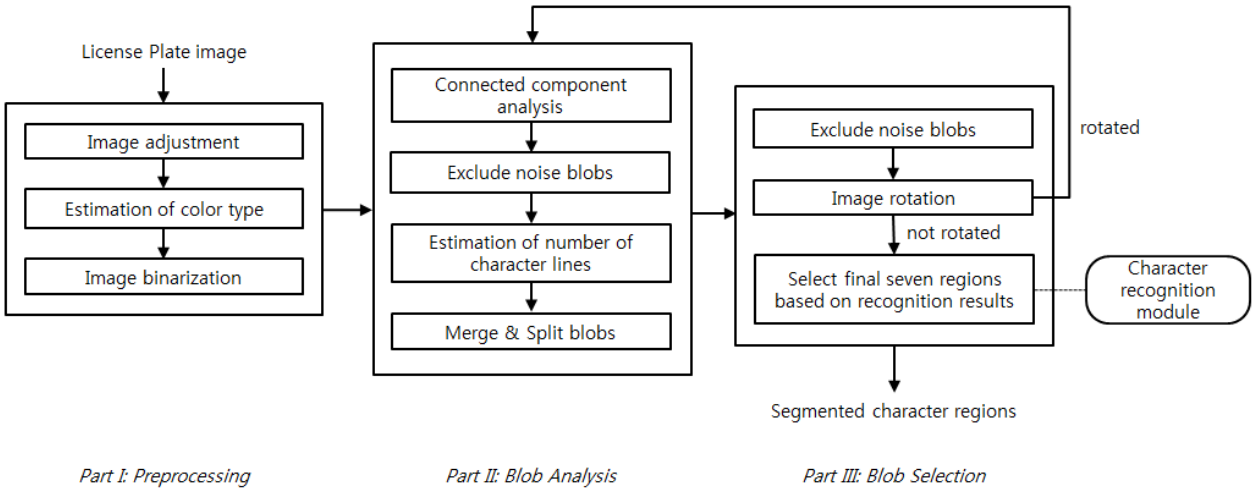

Figure. 2. The flow chart of the character segmentation module. The input of the module is a license plate image and the output is seven segmented character regions. The procedure consists of three main parts and the parts runs in consecutive order.

similar color values. Lastly, the module finds four boundaries of the license plate through the Hough transform, and corrects the rotation and skew of the image. In a few cases, it is failed to find four boundaries correctly and the module may output slightly rotated or skewed images.

III. CHARACTER SEGMENTATION

Fig. 2 shows the overall procedure of the CS. The procedure consists of three sub parts. The first part adjusts the input image and generates a binary image. The second part analyzes blobs which is a connected set of pixels in the binary image, and removes noisy blobs. Merging and splitting the blobs is also done in this part. The third part corrects rotated plate images and selects final seven character blobs.

A. Part I: Preprocessing

The module obtains a grey-level image of a license plate from the LPL module. First of all, an image adjustment is applied to increase the contrast of the image. In this function, we normalized the intensity values so that the value range is 0 to 255, and we also made 1% of pixels are saturated at lowest or highest intensity values. This saturation is effective to reduce noises when we use a local binarization method which is sensitive to intensity changes even at low and high intensities.

Next step is an image binarization step. Most popular image binarization technique is Otsu's method. Otsu's method is a kind of global thresholding method, so it uses a single threshold value to segment foreground regions. This global thresholding method is easy to make errors when it processes images having shadows or heavy illumination changes. We also confirmed that the Otsu's method is not working successfully for some plate images as shown in Fig. 3. The alternative solution is Niblack's and Sauvola's algorithms [12] which are popular local thresholding methods. Niblack's algorithm calculates binarization thresholds for every pixel.

Figure. 3. Comparison of the three binarization methods.

The pixelwise threshold is computed by using the mean and standard deviation of the pixels in the surrounding regions. Sauvola's method is a modified version of Niblack's which introduced one more control parameter to give improved results. In our preliminary experiment, we compared three binarization methods by using the license plate images as shown in Fig. 3. The two local thresholding methods have similar performance, but we selected Niblack's method as the binarization algorithm in the system because it showed better results for some cases. Sauvola's method tends to outputs two or more blobs for a character compared to Niblack's when the character has weak edges. The human can recognize the characters easily even the blobs are fragmented, but the system can be confused. He et al. has also reported that Niblack's showed better performance than Sauvola's method [12]. The adjustable parameters in the binarization methods are tuned for the experiments.

Most European license plates have white background and black characters, but some exceptional plates have blue or red background and white characters. The binary images for the exceptional cases should be inverted since the binarization method assumes that the darker pixels make foreground regions. For detection of the exception cases, we used a simple idea that background area of the plates is larger than the foreground area, so the binary image is inverted when the number of white pixels as considered as background region is

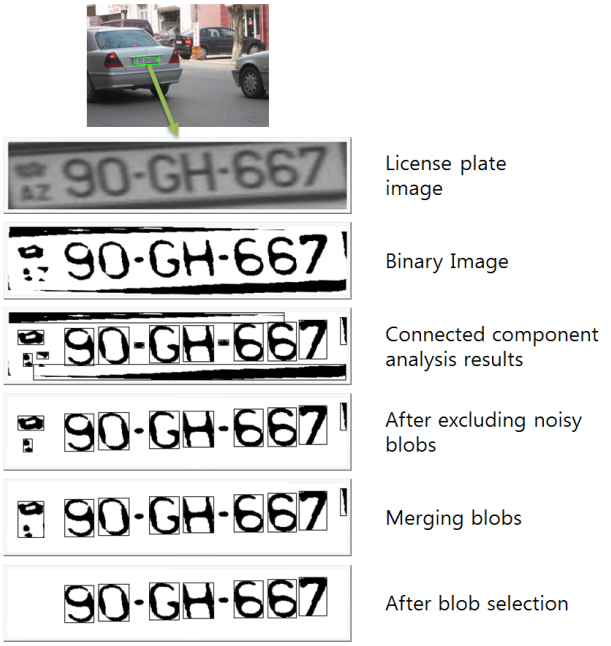

Figure. 4. Results of the character segmentation steps. The process goes from top to bottom and rectangular boxes in the image represent labeled blobs.

less than the blacks. Here, only center part of the binary image is targeted because the other part may include car or road scene which misleads estimation of the plate color type.

B. *Part II: Blob Analysis*

Part II starts with the CCA which makes labels for each isolated blobs. The module inspects all labeled blobs to exclude non-character blobs. Plate boundaries, small dirties, irrelevant marks or bars are excluded. We used the following conditions to remove the blobs. Here, width (W), height (H), area (A), center of the bounding box (X, Y), and center of mass (X_m, Y_m) of the blobs are used. I_w and I_h means width and height of the input license plate image.

1) Too large or small blobs; $W < 0.0625 \times I_w$, $W > 0.3 \times I_w$, $H < 0.125 \times I_h$
2) Small area compared to the size of the bounding box; $A < 0.15 \times W \times H$
3) Blobs positioned at too high or low part; $Y < 0.25 \times I_h$, $Y > 0.75 \times I_h$
4) Blobs at the corners of the image. It excludes the blobs of the license plate boundaries;
   $(X_m < 0.015 \times I_w \,\&\, Y_m < 0.015 \times I_h)$,
   $(X_m < 0.015 \times I_w \,\&\, Y_m > 0.985 \times I_h)$,
   $(X_m > 0.985 \times I_w \,\&\, Y_m < 0.015 \times I_h)$,
   $(X_m > 0.985 \times I_w \,\&\, Y_m > 0.985 \times I_h)$

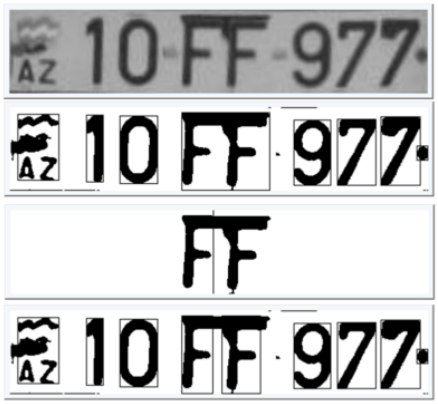

Figure. 5. A sample case of the splitting blobs. The center blob is split into two blobs containing only one character. The algorithm found a splitting line as shown in the third image.

Before proceed to the merging and splitting blobs, the CS module estimates a number of character lines. Most recent license plates are single-lined as shown in Fig. 4, but some plates have two lines of characters. First line of the double-lined plates has one or two characters and the second line has five or six characters. Therefore, the arrangement of the blobs gives a cue to estimate that the plate is double-lined or not. The module uses two conditions for finding of double-lined plates. First one is that the median of blobs' heights should be less than $I_h \times 0.5$. Second condition says one or two blobs positioned at upper half part of the image and the other blobs are at lower half part. If the plate is double-lined, the CS module resized the binary image into the image of doubled height for the next steps.

The next one is a merge step. The merge step is to merge the fragmented blobs for a character. It merges two or more blobs if they overlap each other in a vertical direction. For example, an alphabet 'E' may have two blobs of upper part and lower part by the incorrect binarization and horizontal motion blurs which may shade the edge of vertical lines. In this case, the module merges two blobs of 'E' since the blobs are positioned almost identical horizontal point. At the merge step, we also compared the height. The merging process is applied only when the height of the merged blobs is similar to the other normal blobs. The module did not merged blobs in a horizontal way since it is difficult to distinguish the fragmented blobs from the normal blobs which are arranged horizontally. The merging step is also skipped for the double-lined plate images.

The next step is for splitting a blob which contains two or three characters. We used a vertical projection histogram to separate the characters. The vertical projection histogram is computed by counting the black pixels in each vertical scan. The connecting region between the characters has relatively small number of foreground pixels compared to the character regions, so the minimum value of the vertical projection histogram is used to find out the splitting line. At finding the minimum value, left and right ending sides of the blob is not

considered because the ending parts may have the minimum value even it is not a point splitting characters. Based on the splitting line, the module found precise position and width of the split blobs. The width set to be the median of the width of the other blobs. The horizontal position is determined that the blob has maximum summation value of the vertical projection histogram. If the blob is wide enough to have three characters, the module finds two splitting lines and separates the blob into three character regions. Fig. 5 shows an example of the split step.

C. Part III: Blob Selection

As you see in Fig. 4, the CS module almost completely identified the character regions after merging and splitting blobs. However, some non-character blobs may exist. At this point, the outlier blobs are excluded by comparing its height. When a variance of height of the blobs is more than a predefined threshold, the most distant blob from the others in terms of a height value is abandoned. Excluding outlier process iterates until the variance condition do not meet or only seven blobs are remaining.

Before selection of the final seven blobs, the CS module corrects rotation of the image. The recognition module requires upright character images for the better recognition, so it is necessary to correct rotated license plate images. The degree of rotation can be estimated from the arrangement of the character blobs. We found a fitting line for the center points of the blobs. If the plate is double-lined, only the blobs on lower part are used. The CS module rotates the whole image by the degree estimated from the fitting line. If the rotation is corrected, the process backs to the start of Part II in Fig. 2 since the noise filtering and split process could went wrong with the rotated images. The Part II and III can be done iteratively with the rotation, but we assumed the CS process failed when the number of iteration exceed three times.

After the rotation test, we select final seven blobs that are expected to be character blobs. At the previous stages, we used geometric information to filter out non-character blobs, so the remaining non-character regions are hard to be excluded from the geometric information. Therefore we used the character recognition module which recognizes numbers or alphabets from a character image. The recognition results include best matched character and corresponding matching scores. It is evident that the blobs with a low matching score tend to be non-character blobs. Among the blobs, the CS module selects seven blobs with higher matching scores. The character recognition module used 8-direction gradient features and the artificial neural network as a classifier [13].

IV. Experiments

A database of 1944 license plate images is prepared for the evaluation of the LPR system. The images were captured by a handheld digital camera and the image resolution was 1280 x 960 pixels. License plates occupy about 235 x 55 pixels on average, and the minimum size was 107 x 21 pixels. All plates follow the European license plate format and capturing site was Azerbaijan. Most common plate is with white color of background, black color of characters, and seven characters at a line. There also exist license plates having yellow, red, and blue color of background, white color of characters, and two lines of character.

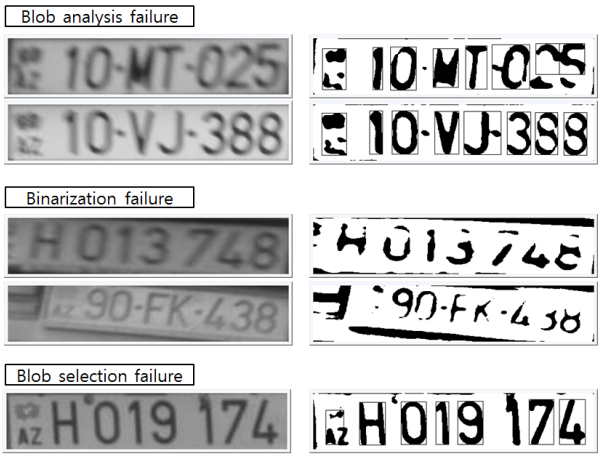

Figure. 6. Failure cases of the character segmentation.

We implemented a complete LPR system which consists of the LPL, CS, and CR modules. The system is evaluated with the database. When all seven characters in the license plate are correct, the system counts the plate is correctly recognized. In this experiment, the recognition accuracy was 90.9%. The failure cases are reviewed by the authors and decided which module is failed. We judged that the LPL module is failed when any character is not readable or the image includes too much background regions not the license plate. The CR module is considered as failed when the recognition result is not correct even the CS module found seven regions of character blobs correctly. Among 177 error cases, the LPL module failed for 69 images. 54 and 41 cases are failed by the CS and CR module. The 54 failure cases of the CS module are analyzed in detail. We categorized the failure cases into five groups. Following describes each group with their frequencies and Fig. 6 shows representative failure cases of the CS module.

1) Blob analysis failure (21 cases): This case includes failures of merging and splitting bobs, so there are fragmented characters or over-sized blobs. A fragmented character means a character divided into two or more blobs. Low resolution images or motion blurred images has weak edges on characters. These weak edges lead some part of characters considered as background region in the binarization step, and the CCA makes two or more blobs for a character. The failure of the split step makes over-sized blobs which has two or more characters or non-character parts. Short horizontal bars or plate boundaries tend to be connected to the character regions. First two examples in Fig. 6 show the blob analysis failures. First one is that the merging blobs did not work properly. Labeling of last two characters went wrong. A

blob of alphabet 'J' in the second case includes the short horizontal bar. It led wrong character recognition results.

2) Binarization failure (15): We considered the binarization is failed when seven characters on the binary image is hard to be recognized even by the human. Damaged plates, severely dirty plates, or motion blurred images makes this bad binary images. The binary images of the third and fourth cases in Fig. 6 are hard to be recognized.
3) Blob Selection failure (13): Noisy blobs can be selected instead of character blobs. When the non-character blobs have similar size with the character blobs, the blob selection problem occurs. Additionally, some noisy blobs which has similar appearance of number '1' or alphabet 'L' are selected as character blobs even after the last step of the CS module that incorporates with the CR module. The last case in Fig. 6 shows the failure of the blob selection. Number '1' in the image is not selected since its height was bigger than the other characters.
4) Extras (5): The color estimation task in the part I is failed for four cases. The detection of double-lined plates is failed for once.

The system takes 0.32 sec to process one image in the database. When we run the CS module only, it takes about 0.15 sec. A computer with 3.3GHz CPU is used for this experiment.

V. CONCLUSION

This paper presented the CS module for the automatic LPR system. We introduced the robust CS algorithm by integrating the local binarization method, merge and split blobs, and incorporation with the CR module. The CS module achieved 97.2% of success rate and the complete LPR system with the LPL and CR module had 90.9%. The performance is not directly compared with the previous systems but we analyzed the failure cases closely. The most important step in the CS module was the binarization. Not only 15 cases of the binarization failure but also blob analysis failures are related to the binarization step. The merge and split steps in the blob analysis part are designed to supplement the incomplete binary image, and naturally the failures of the blob analysis can be removed through more accurate binarization methods. The future studies will focus on this binarization problem. In the present study, we found a single parameter for Niblack's binarization method. The parameter is good for the general images, but it is not the best one for each image. The scheme that the parameters changes according to the image quality or results of the CCA would be effective to increase the quality of the binary image. In addition, the use of the prior knowledge of the character position will be helpful. In the blob selection part, a noise blob was competitive with a character blob for some samples since the noise blob has similar height and appearance. The prior knowledge of the character arrangement may be another cue to determine which blob is making more probable arrangement of characters.

REFERENCES

[1] S.L. Chang, L.S. Chen, Y.C. Chung, and S.W. Chen, "Automatic license plate recognition," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, no. 1, pp. 42-53, 2004.

[2] H.A. Hegt, R.J. De la Haye, and N.A. Khan, "A high performance license plate recognition system," in *Proc. IEEE Int. Conf. System, Man, and Cybernetics*, vol. 5, 1998, pp. 4357–4362.

[3] B.R. Lee, K. Park, H. Kang, H. Kim, and C. Kim, "Adaptive Local Binarization Method for Recognition of Vehicle License Plates," R. Klette, J. Uniæ (Eds.), *Lecture Notes on Computer Science*, vol. 3322, Springer, New York, 2004, pp. 646–655.

[4] S.Z. Wang and H.J. Lee, "A cascade framework for a real-time statistical plate recognition system," *Information Forensics and Security, IEEE Transactions on.* vol. 2, no. 2, pp. 267–282, Jun. 2007.

[5] C. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas, "License Plate Recognition From Still Images and Video Sequences: A Survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 3, pp. 377-391, 2008.

[6] H.T. Lue, M.G. Wen, H.Y. Cheng, K.C. Fan, C.W. Lin, and C.C. Yu, "A Novel Character Segmentation Method for Text Images Captured by Cameras," *ETRI Journal*, vol. 32, no. 5, pp. 729-739, Oct. 2010.

[7] X. He, L. Zheng, Q. Wu, W. Jia, B. Samali, and M. Palaniswami, "Segmentation of Characters on Car License Plates," in *Proc. IEEE Workshop on Multimedia Signal Processing*, 2008, pp. 399-402.

[8] Y. Zhang and C. Zhang, "A new algorithm for character segmentation of license plate," in *Proc. IEEE Intelligent Vehicle Symposium*, 2003, pp. 106–109.

[9] F. Martin, M. Garcia, and J.L. Alba, "New methods for automatic reading of VLP's (Vehicle License Plates)," in *Proc. IASTED Int. Conf. SPPRA*, 2002.

[10] S. Wang and H. Lee, "Detection and recognition of license plate characters with different appearences," in *Proc. Conf. Intell. Transp. Syst.*, 2003, vol. 2, pp. 979–984.

[11] P. Viola and M. Jones, "Robust Real-time Object Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, 2002.

[12] J. He, Q.D.M. Do, A.C. Downton, and J.H. Kim, "A Comparison of Binarization Methods for Historical Archive Documents", in *Proc. ICDAR*, 2005, pp. 538-542.

[13] C.L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, pp. 2271-2285, 2003.