# EECS 442 Final Project - License Plate Recognition

Jiongsheng Cai
Jingyao Hu
Yucheng Yin

December 7, 2017

[We have sent private Github repo invitation link to Siyuan.]

## 1 Introduction

Automatic License Plate Recognition (ALPR) has become a popular technology since it was invented in 1976 at the Police Scientific Development Branch in the UK [1]. It can be used in different fields like checking whether a vehicle is registered or licensed by police forces, or used for pay-per-use roads like electronic toll collection and parking lot. Our group tries to solve the license plate recognition problem by following a classical series of steps. That is, plate localization and extraction, segmentation, and finally character recognition by CNN.

## 2 Related Work

Since the technology has been invented for over 40 years, there are quite a lot of well-implemented tools and libraries available. Our goal is not to directly use these well-developed tools but to implement our own version of LPR and most importantly, deepen our understanding of the typical algorithm. We searched online for typical algorithms and find generally two sets.

### 2.1 Classic Method

As mentioned in the relatively earlier papers [2], a typical process of license recognition could include:

- **Plate localization and extraction.**
- **Character segmentation.**
- **Recognition.**

### 2.2 Methods utilizing Neural Networks

As the development of Neural Networks these years, there have been a lot of NN-related methods to solve License-plate recognition problems [3]. Compared with the classic methods, the NN-related methods have a faster processing speed and higher accuracy in recognition.

## 3 Methods

Our method is based on the classic model of LPR and utilize the basic techniques we have learned in EECS442. Some of the procedures have been simplified due to the time limit and all the key components of the algorithm are implemented by ourselves.

## 3.1 Dataset

There are a few dataset for license plates online but some of them are significantly variant in terms of illumination, viewing angle and plate structure, which will cause great difficulty to our project. Therefore, we choose a relatively well-built dataset. Note that the dataset doesn't come with labels which cause us difficult to manually count the accuracy of recognition due to the relatively large size of the dataset.

## 3.2 License Plate Localization and Extraction

- **Edge Detection** Before we start to find the plate location, we need to remove unnecessary information from the image. Although MATLAB has built-in functions for achieving such goals, we implemented our own version of edge detection which utilizes the sobel filter. In short, we apply two matrix

$$S_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

  on the original image to calculate the strength at each pixel and after normalization, we convert the gray-scale image to a black-and-white image with edges detected.

- **Histogram variation** After the edge detection, we calculate the horizontal and vertical histogram. For the horizontal histogram, it's calculated by traversing each column of the image. It starts traversing with the second pixel form the top of each column and the difference between second and first pixel is calculated. In the meantime, we maintain a sum value for each column. If the difference is greater than a threshold, it is added to the total sum of differences. In the end, we get an array for column wise sum. The vertical histogram is calculated exactly same as horizontal histogram.

- **Image cropping** After we get the two histograms, we start to find the horizontal and vertical location of the license plate. Take horizontal location as an example. We maintain a starting point and an end point while traversing the horizontal histogram array. If certain value is greater than pre-defined threshold, we mark it as starting point and continue until certain value is smaller than threshold, then we mark it as end point. Therefore we will get many pairs of staring and ending points. We choose the pair with largest difference (length). The process for finding vertical location is the same story.

## 3.3 Character Segmentation

- After we get the extracted plate, we need to segment each character of the plate for the recognition. We take the advantage of the large color difference (black and white) between the character and the space in the middle of two characters. The following is the specific description of the algorithm.

```
1: procedure SEGMENT
2:     $start\_position = 0$
3:     for i:column(image) do
4:         Record the position when we fully pass a character
5:         Record the position when we meet the second character
6:         Reset the start position and flags
```

- After we get the raw segmentation, we further remove the noise by utilizing the factor that the ratio of white weight to black and the ratio of column to row are in some specific range for the characters we want. Therefore, we manage to filter out some noisy segmented results.

## 3.4 Character Recognition

- After we get the exact segmentation for each character in the license plate. We decided to implement a character-level character recognition. Our work is based on a convolutional neural network (CNN) trained on EMNIST dataset [4]. EMNIST dataset is an extension of MNIST dataset, which shares the same image structure and parameters as the original MNIST task, involving handwritten letters (uppercase and lowercase) and digits.

- We chose to train our network on balanced classes, which contains 131,600 characters and 47 balanced class. Considering computational efficiency, we implemented a common and simple network structure for MNIST training. For image pre-processing, we resize every image (of character segmentaion) into 28*28. The network structure follows as two 3*3 convolutional layers, one max-pooling layer, and one fully-connected layer. We use softmax funcion for multiple classes classification.
- Our network shows an accuracy of 87.33% on EMNIST test data.

## 4 Results

As mentioned above, our dataset doesn't come with labels. Due to the large size, we don't count manually to calculate the accuracy for these plate numbers. By our observation, we find that the plate extraction and character segmentation is quite good, while the character recognition is relatively poor, which causes the recognition accuracy relatively low. We will present one set of good example and one set of bad example. The reason for good or bad recognition will be discussed in the following section.

- One good example.

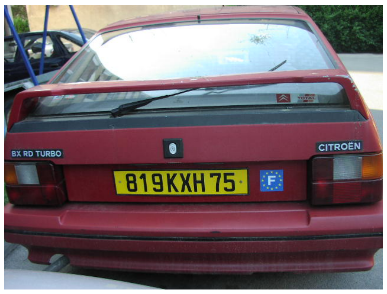
Figure 1: Original Image

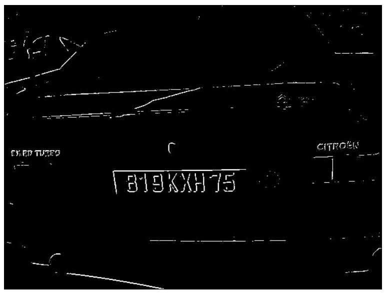
Figure 2: After edge detection

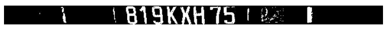
Figure 3: After segmentation

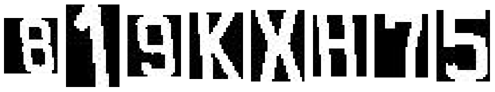
Figure 4: Cropped characters.

**Recognition Result: 819KXH75**

- One bad example.

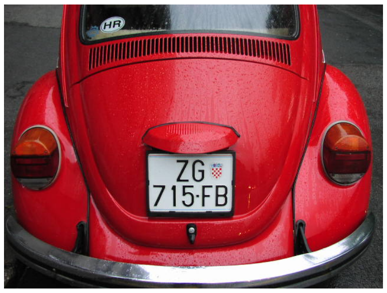
Figure 5: Original Image

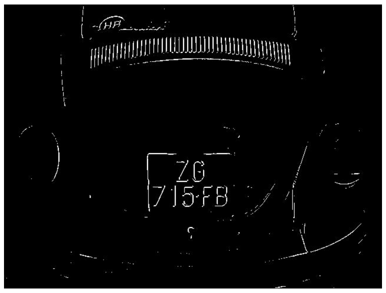
Figure 6: After edge detection

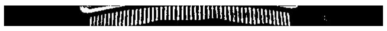
Figure 7: After segmentation

**Recognition Result: Failed. No character segmentation.**

## 5 Discussion

### 5.1 Result Anaylsis

In general, our method does a great job in plate extraction and segmentation and didn't perform very well in the character recognition using CNN. The possible reason can be:

- Although the dataset is chosen wisely, there are still quite a few images with relatively poor illumination and strange viewing angles. Our current version doesn't include a rotation-invariant characteristic since the rotation is not in 2-D. Instead it's a projective transformation which requires some techniques in parameters settings. For the illumination, we try to remove shades from the region of interest and it also removes some key information while removing the noises.
- The relatively poor performance of CNN could attribute to the input images. Since the input images are cropped and segmented by ourselves, it has some differences with the train images EMNIST used by the CNN. For example,

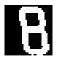

Figure 8: Confusing digit.

The actual digit is '8' while it is quite similar to character 'B'. In such cases, CNN will have a poor performance even with enough training examples.

### 5.2 Improvements

- Improve the plate extraction and segmentation. We could use some techniques to achieve rotation and illumination invariance which will lead to great improvements in removing noise and extraction. Also, by using some more complex segmentation techniques, we could achieve better segmentation results which will help improve the accuracy.
- Using R-CNN or other CNN models. Currently we only use CNN at the last stage - recognition. If we could use R-CNN to achieve segmentation and recognition at the same time, the results could be improved. Also, with more accurate and dynamic CNN models (parameters), the recognition could be more robust to confusing characters, which will also help improve the accuracy.

## References

[1] "Automatic number-plate recognition," Dec 2017. [Online]. Available: https://en.wikipedia.org/wiki/Automatic_number-plate_recognition

[2] S. Toda, "License plate recognition," Apr. 21 2005, uS Patent App. 10/966,622. [Online]. Available: https://www.google.com/patents/US20050084134

[3] Y. Wu and J. Li, *License Plate Recognition Using Deep FCN.* Singapore: Springer Singapore, 2017, pp. 225–234. [Online]. Available: https://doi.org/10.1007/978-981-10-5230-9_25

[4] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *CoRR*, vol. abs/1702.05373, 2017. [Online]. Available: http://arxiv.org/abs/1702.05373