Deep License Plate Recognition

Haotian Shi, Wenrui Li, Yu Yang, Yang Zheng, Xiaodi Zhang

Research School of Computer Science,
ANU College of Engineering and Computer Science

{u6158063, u6361099, u6412985, u6287751, u6368740}@anu.edu.au

Abstract. License Plate Recognition (LPR) system is widely used in our daily life. Researchers have developed several methods to implement LPR system by using morphology method, machine learning, or neural network. Our group project aims to implement LPR system. Our approach combines morphology method in plate detection and character segmentation, and neural network in character recognition. The overall accuracy reaches to 87%.

Keywords: LPR, CNN, license plate, image segmentation, character recognition

1 Introduction

Vehicle License Plate Recognition (LPR) system normally has three steps: car plate detection, character segmentation, and character recognition. Car plate detection is to locate the area of the plate from the origin image, and then extract the car license plate image for next character segmentation and recognition. We use the method which is based on the image morphology to detect the area of car license plate. Character segmentation is to segment six characters from the car license plate to prepare the dataset for the recognition task. Finally, we use CNN to conduct the character recognition.

CNN is one of the classical deep learning network frameworks that widely used in image classification domains [1]. The performance of CNN on image classification has been proofed in 2012 on the ImageNet [2]. In our system, we use a CNN with two convolutional layers to classify the characters cropped from the template. The cropped characters include digits and letters, which align with the font of the ACT standard template. We train the network using a character dataset collected online and test the images of automobiles on the street, all of images are collected by the team members.

2 Related Work

There are two main methods to recognize license plate. The first one is using morphology [3]. An image with license plate is preprocessed: denoise and binarization. Then, the plate is extracted by recognizing rectangle regions in the image, filtered by the region area, the ratio of height and width, and the color information. The third step is to

segment characters along vertical and horizontal direction with some denoise processing. Finally, minimum Euclidean distance based license plate matching is used to recognize the characters. However, the total accuracy of this method is not very high due to the limitation of morphology.

The second method is combining morphology, neural network and machine learning [4]. The first step is to detect cars in an image by simply using color based approach. Then, a convolutional neural network is built to analyze and filter color and texture properties, so that the car license can be detected and extracted. Finally, this method chooses support vector machine to recognize characters on the plate. Based on the experiments, the accuracy can reach to 97.2%. Hakan and his colleagues [5] improve this method so that the license plate recognition system can be used in the video. They use Gabor filter and connected component labelling algorithm to detect license plate region, and then use self-organizing map neural network to identify the characters. This method is faster and efficient than previous one, and the accuracy is very high.

3 Methodology

Due to the influence of weather change, light intensity, the vehicle cleanliness and other outdoor conditions, the images of the car in the actual scene are very complicated, so the images need to be preprocessed before the car license plates locations. In our system, the pretreatment includes image resize, image denoising filtering and image equalization. After that, obtain the edge image using the binarization and edge detection algorithm. Process the edge image with the dilation, erosion opening and closing algorithm to obtain the protein blocks from the image. Then, screen the blocks with features of ACT standard car license plates to find the target block which contains the area of car plate. Finally, using the foreground extraction model to cut the car plate image.

According to the above conceptual design, the process flow of the car license plate detection is like this:

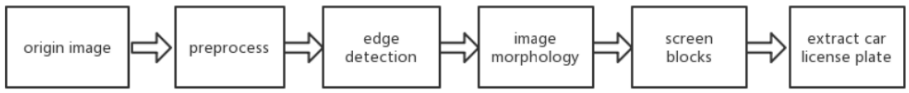

Fig 1. Process flow of the car license plate detection.

3.1 Localize License Plate in Images

3.1.1 Preprocessing

The preprocess part includes image resize, image grayscale conversion, image de-noising filtering, and image equalization. Since the size of the collected original images is large. The image resize is to transform the original image to a small size, improving the speed of image processing. When taking the images with camera, some noise may be brought in. So we should denoise the images. Neighborhood average method, median filter and Gaussian filter can all be used to denoise. In our system, the Gaussian filter with 5 size has a better outcome, so we can use it to filter the image. In addition, some

images are not well lit, and there are some reflections on the part of the car, so some grayscale images are poor in contrast. It is necessary to increase the contrast between the license plate area and the background area. We use the image equalization algorithm to improve contrast.

3.1.2 Edge Detection

The edge detection is to obtain the boundaries of objects within the images and it is a gradient sharpening idea. We find that using image morphology on the edge images we can acquire better outputs, especially for white color cars. Because for the white cars, when using morphology operations on these binary images, basically the car and the plate are fused as one big block. It is impossible to extract the car plate. So the precise information of edge is the important feature of car license plate detection.

There are many common algorithms to conduct the edge detection, such as Sobel, Roberts, Prewitt, Laplace and Gaussian and Canny. In our image dataset, the Canny algorithm has a better performance than other detection algorithms. The Canny was developed by John F. Canny in 1986, using a multi-step algorithm to detect edges [6].

3.1.3 Image Morphology

Image morphology common operations include dilation, erosion, opening and closing. After obtaining the edge binary image, we combine image morphology operations to process the image to fuse the area of car license plate as a whole white block. We find that the suitable combination of open and closed operations can acquire satisfactory results on our image dataset. Firstly, using closing operation to connect the broken parts inside the image and the kernel size is 10-by-30. Secondly, operating the opening to eliminate small objects and separate objects at cutting points with the same kernel. Lastly, using the opening operation with 20-by-40 kernel to smooth the boundaries of objects and eliminate small objects again.

3.1.4 Screen Blocks

After image morphology processing, we get several white blocks in the binary image. We utilize the contours detection algorithm to find the contours of all the blocks. The OpenCV provides *FindContours()* function to detect contours in the image. Basically, every contours are irregular polygons. In our system, we compute the coordinates of top right corner and left bottom corner of one block to form a rectangle to represent the block. After obtaining the rectangle of every block, we begin to screen the blocks.

Firstly, using the length-to-width ratio to remove some unsatisfactory blocks. In our system, the ratio is set as 1.2 for default and the width and height should be more than 10 pixels. Secondly, employing the features of blue words and white background of standard ACT license plates to check every left block. Count the proportions of blue pixels and white pixels within every block. Then compute the sum of proportions. For avoiding these situations that all pixels are white or blue within the block, we set weights for the two proportions which are 0.7 for blue proportion and 0.3 for white proportion. Finally, we return the block with maximum sum of proportions as the target block.

3.1.5 Extract Car Plate

When we get the initial target block, the block usually includes some extra parts which are the parts of the car. So we need to utilize the initial target block to acquire the precise car license plate area. In our system, we use the Grabcut Algorithm to extract the better car license plates. We input the initial target block to the Grabcut, then it will treat pixels outside the block as background. It uses the texture information (color) and boundary information (contrast) of the image to identify the pixels inside the block as foreground or background [7]. Finally, we can obtain the better foreground which is usually a more precise car license plate area without extra parts. After we get the better target block, we need to transform the target block to the original image, conducting a scale transformation on the target block. Since the original image has better image quality, we can cut clearer car license plate images.

3.2 Segment Characters from License Plate

After obtaining the license plate cropped from the original image, we need to segment six characters to prepare the dataset for the recognition task. Briefly, the process of segmentation can be described in six steps, border cropping, binarization, noise elimination, contours finding and filtering, character image resizing and padding.

3.2.1 Border Cropping

Some car owners used license plate frames for decoration or protecting the license plates from bending or distortion. In our experiments, we found the top and bottom sides of frames may have negative effect when finding contours for the characters, so the borders of the license plates were removed. Generally, we cropped 24% in total from top and bottom of an image, and 2% in total from left and right sides of an image. Some license plates were not perfectly localized that some unwanted areas were included in widths. In such cases, the ratio between width and height was greater than 3.5, so the perception cropped from the height would reduce to 10%. An example of original image and its cropped image is shown in Fig. 2.

Fig. 2. Original and cropped images

3.2.2 Binarization

The second step is converting the image to binary representation. We adopted the green channel (the red channel is also acceptable) for binarization, in which the threshold is the average of the maximum greyscale value and the minimum greyscale value. In the result, the background is white and the foreground is black. A binarized image is shown in Fig. 3

Fig. 3. Some noises in a license plate

3.2.3 Denoise

Except the noises introduced by dirt, there are some other kinds of noises that affects the segmentation like some decorative shapes in the middle of the license plates (framed in red in Fig. 4 (a)) and some words or sentences near the target characters (framed in green in Fig. 4 (a)). We eliminated the noises by Gaussian blur and median blur, after which the eroding and dilating operations are performed. These morphology functions could remove the small noises, make those decorative shapes invisible, and blur the small words and sentences such as "ACT", "CANBERRA – THE NATIONS CAPITAL", "CELEBRATION OF A CENTURY 2013". The result after removing the noise is illustrated in Figure 4 (b).

(a) Some noises in a license plate (b) After removing the noise

Fig. 4. An example of noise elimination

3.2.4 Counters Finding and Filtering

The contours related steps are most important to segmentation. We used the *findContours()* function of OpenCV to identify the contours of the shapes in the binary images. Besides a binary image, we passed two parameters to this function, *cv2.RETR_CCOMP* and *cv2.CHAIN_APPROX_SIMPLE*.

The first parameter is a mode of the contour retrieval algorithm that determines which contours to be retrieved and whether to establish their hierarchical relationships. cv2.RETR_CCOMP means that we retrieve all of the contours and organizes them into a two-level hierarchy. The top level is a contour that contains the whole plate. The second level are the contours of the characters to be segmented, in which there are six contours for the characters. Note that this algorithm may include some unwanted contours which should be excluded. For example, a red contour of the dot located in the center of the plate is shown in Fig. 5 (a). Any contours inside a character's contour are assigned to the top level. The second parameter is a contour approximation algorithm, which compresses vertical, horizontal, and diagonal segments and only leaves the ending points. With the value of *cv2.CHAIN_APPROX_SIMPLE*, it uses an up-right rectangle encoded with 4 points to define a contour for any shapes in a plate. For more details, see [8] [9].

To find the contours of the characters precisely, we performed a filtering for the contours. Firstly, the contour of the plate whose x and y coordinates start at 0 would be excluded. Secondly, any contours of which the height is less than 55% of the height of

the license plate would be excluded. Thirdly, only the contours in the second hierarchy are acceptable. The result of filtering is shown in Fig. 5 (b).

(a) A contour of the dot in the center (marked in red)

(b) Contours after filtering

Fig. 5. Contours finding and filtering

3.2.5 Other Processing

With all the above, we can segment six characters from a license plate by the 4 ending points of each contour. To make them ready for recognition, and each character image is resized and padded to size 28x28, and the black and white colors are also swapped. If the six characters are segmented successfully, they will be saved to specific directories with BMP format.

3.3 Recognize Segmented Characters

Our network model design aligns with the classical CNN layer, the basic structure is one Convolutional Layer followed by an activation function. At the end of the layer, a pooling layer is used for down-sampling.

The convolving process can be denoted as

$$h(x) = \int_{-\infty}^{\infty} f(x-u)g(u)du \tag{1}$$

where f represent the input of the convolutional layer, g is the kernel with size u, h is the result convolved by g. Because the input is an 2D matrix which represent a grey-scale digital image, the above equation can be represented as

$$h(x-a, y-b) = \sum_{y=0}^{cols}(\sum_{x=0}^{rows} f(x-a, y-b)g(a,b)) \tag{2}$$

where g is a 2D kernel with the size of (a,b).

ReLU is used to be the activation function of CNN, which located after convolutional layer to maintain the non-linearity, which is shown below.

$$ReLU(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases} \tag{3}$$

In our solution, max-pooling is applied to sample the feature map, the mathematical model is denoted as below.

$$f(a,b) = max\big(g(a,b)\big) \tag{4}$$

Where $g(a,b)$ is the sampling window with width and height as a and b, f is the max sample in the sampling window.

For back-propagation process, we use the cross-entropy loss combined with the Soft-Max function to get the loss value, according to the deduction of the previous study, the loss function represents as the following equation. x is the output and the j means jth output.

$$Loss(x) = -log\left(\frac{e^x}{\sum_j e^x}\right) \tag{5}$$

The overall network structure is shown in Fig. 6, the input is 28x28 grey scale image and the output is a vector that has 33 elements. One fully connected layer is used to combine the feature maps convolved by the previous convolutional layers.

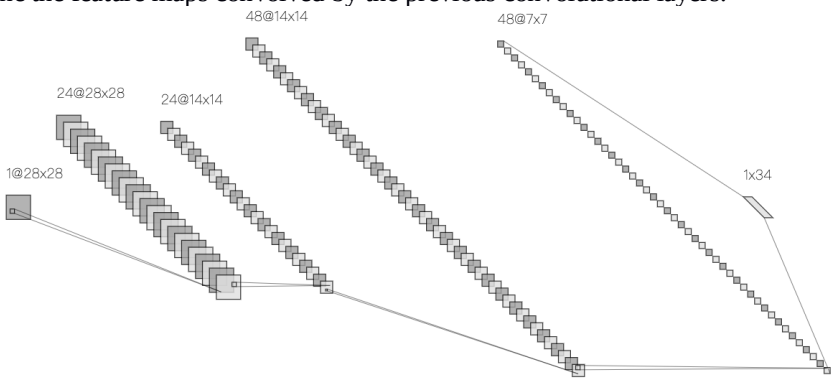

Fig. 6. The structure of the CNN template character classifier

Table 1 list the parameters of each layers, some experiments are made to select the combination of the all the layers and the currant solution are best fit for our training data and made a balance between the accuracy and the calculation speed.

Table 1. The structure of the CNN template character classifier

Layer	Input Size	Output Size	Kernel Size	Stride	Channels	Padding
Conv 1	**28x28**	28x28	5x5	1	24	2
Max-Pool 1	28x28	14x14	2x2	1	24	1
Conv 2	14x14	14x14	5x5	1	48	2
Max-Pool 2	14x14	7x7	2x2	1	48	1
Linear	7*7*48	**34**	-	-	-	-

We established a training and testing dataset combining the online resource and self-collected samples. The dataset has two splits for training and validating. The total class number is 34, which contains 10 digits and 23 letters. All the letters are capitalised and a merge are applied to O & 0 and I & 1 class because of the similarity of the edge and geometric features. The overall training set includes 756 samples and the testing set have 360 characters from detected result of the template samples.

4 Result and Discussion

4.1 Car Plate Recognition

We use phone cameras to take 120 car images in the real world. The car license plates of all the images are standard ACT car license plates. Experiments are based on this image dataset.

The Fig 7 shows the result of preprocessing. The first one is the original image. The second one is the result of resize and Gaussian filter. We set the width of images as 500 pixels, and the height is equal scale transformation. The size of Gaussian filter kernel is 5. The third image is the grayscale image. The last one is the result after equalizing the image using the inbuilt function of OpenCV.

Fig 7. The results of preprocessing

The figure 8 shows the result of edge detection using the Canny algorithm. We can see that the edges of this car license plate is detected well. In addition, the edges of car plate are separated with other parts of the car. This is very important for next image morphology operation.

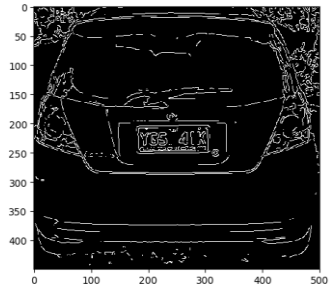

Fig 8. Edge image

The Fig 9 shows the result of image morphology. The left image is the result of operating the closing with 10-by30 kernel. Compared with the figure 3, we can clearly see that the area of car plate basically is connected as a whole block. The middle image is the result of opening operation with 10-by-30 kernel. We can see that the small blocks are removed and boundaries of every block become larger and clearer. The right image is the result of operating opening with 20-by-40 kernel. It is clear that the blocks become very satisfying.

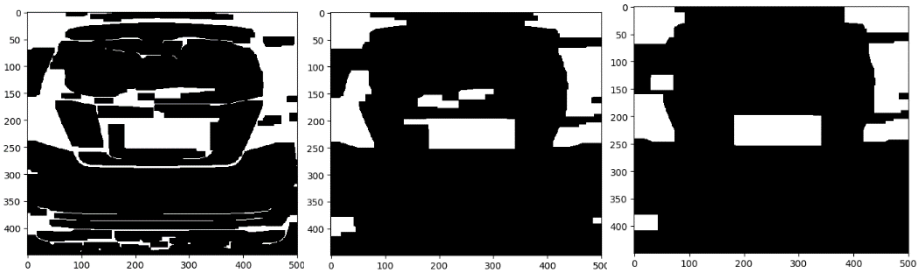

Fig 9. Results of image morphology

The Fig 10 shows the results of extracting car license plate. The left one is the initial target block. We can see that it includes unnecessary part. The middle one is the target block after Grabcut operation. The image only contains the area of car license plate. The right one is the image which is cut from the original image. It is clear that the quality of the image is better than the middle one.

Fig 10. Results of extract car license plate

We test the car plate location on 120 images, the algorithm can correctly detect 76 images, which is about 63% accuracy. The important reason that the plates cannot be effectively detected is that we cannot obtain the block which contains the area of car plate after the image morphology stage. In fact, when we adjust the parameters of the closing and opening, it will work for some images which are not detected on pervious parameters. However, the new parameters will not work for some pervious successful images. Basically, it is hard to find the same hyper-parameters to satisfy all the images. Although we find such hyper-parameters which satisfy our 120 images, it is obvious that the hyper-parameters cannot satisfy all other new images. We think that this is a large limitation for the car detection method based on the image morphology.

4.2 Character Segmentation

Fig 11 shows the result of character segmentation. We segment each character from the plate, transfer it into white colour with black background, and resize them to 28 × 28.

We test 178 cropped car plates, and successfully segmented characters take up to 94.4%. This experiment result suggests that our method for character segmentation is outstanding.

Fig. 11. The segmented characters of "YLB02U"

4.3 Character Recognition

The result shows that the classifier accuracy reached 96% of the of the total training and validation characters. The testing input contains 6 characters images which is already cropped and resized by the segmentation process. The standard of the overall accuracy testing is that one wrong character classification failed this template. Therefore, the overall accuracy is lower than the single character testing dataset. From the observation, most of the failed samples contains 8, B O and Q, which have a similar edge feature of each other, like Fig. 12. The last letter is B, but the curve on the left is not obviously. The reason is that when resizing the images to 28x28, it is hard to keep the original shape of the character edge, which leads to the network prediction cannot distinguish the difference between the letters and have higher confidence rate the wrong class.

Fig. 12. Binarized plate license

4.4 Overall Accuracy

We totally test 66 images for LPR system, and 58 plates can be correctly recognized. The overall accuracy is 87.8%.

5 Conclusion and Future Work

Our group uses morphology to detect car plate and segment the characters. The accuracy for this car plate detection and character segment is about 63.3% and 94.4% respectively. Then, we use CNN to recognize the characters, and the accuracy of CNN is 96%. Finally, we test the overall accuracy of our LPR system, which reaches to 87.8%. This suggests that our LPR system is efficient and accurate.

Due the limitation of car plate detection method based on image morphology (discussed in 4.1 part), we will study the object detection methods using deep learning, such as the YOLO which is the state-of-art object detection system [10].

For characters' recognition, the next step can be resolved the wrong classification between 8 and B. The potential solution that avoid this problem is using high quality image for input but may leads to an increase of the network scale. On the other hand, the plate licenses in this paper is standard plate licenses, but in real world, there are many different ACT plate licenses. Trying to classify more kinds of plate licenses is another good topic.

6 Summary of Learning Outcome

We learnt lots of knowledge about car license plate recognition, and obtained the experience of applying the vison technologies learned in the CV class to solve the practical problem. Through conducting the car license plate detection and characters segmentation, we learnt to use the low-level and mid-level vision technologies, such as the image denoising, image filtering, image morphology and image edge and contour extraction, etc. At the character recognition stage, we learnt to use the high-level vision which is deep learning for vision to settle this problem. In addition, we also learnt and practiced the teamwork ability, including how to assign tasks appropriately, how to write a co-authored paper and so on.

References

[1] Jaswal, D., Vishvanathan, S. and Kp, S., 2014. Image classification using convolutional neural networks. *International Journal of Scientific and Engineering Research*, *5*(6), pp.1661-1668.

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[3] Shi, X., Zhao, W. and Shen, Y., 2005, May. Automatic license plate recognition system based on color image processing. In *International Conference on Computational Science and Its Applications* (pp. 1159-1168). Springer, Berlin, Heidelberg.

[4] Kim, K.K., Kim, K.I., Kim, J.B. and Kim, H.J., 2000. Learning-based approach for license plate recognition. In *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No. 00TH8501)* (Vol. 2, pp. 614-623). IEEE.

[5] Park, S.H., Kim, K.I., Jung, K. & Kim, H.J. 1999, "Locating car license plates using neural networks", *Electronics Letters*, vol. 35, no. 17, pp. 1475-1477.

[6] Canny, J. 1986, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698.

[7] Rother, C., Kolmogorov, V. & Blake, A. 2004, ""GrabCut": interactive foreground extraction using iterated graph cuts", *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309-314.

[8] OpenCV Team. Structural Analysis and Shape Descriptors.: Contour Approximation Modes (2019). Retrieved May 30, 2019, from https://docs.opencv.org/4.1.0/d3/dc0/group__imgproc__shape.html#ga4303f45752694956374734a03c54d5ff

[9] OpenCV Team. Structural Analysis and Shape Descriptors.: RetrievalModes. (2019). Retrieved May 30, 2019, from https://docs.opencv.org/4.1.0/d3/dc0/group__imgproc__shape.html#ga819779b9857cc2f8601e6526a3a5bc71

[10] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).