

Vehicle License Detection and Recognition

Xinyu Zhang
xz1753@rit.edu

Xinchi Huang
hx4949@rit.edu

Zichuan Wei
zw1109@rit.edu

Datong Guo
dg2715@rit.edu

Abstract— Our topic area focuses on vehicle license detection and recognition, which involves image processing and computer vision and may combine with big data. Through recognizing the license of each entered vehicle, parking lots or other charging places such as highway entrances can inquire information about the vehicle, like account balance, whether it is eligible for passing, and so on. What we want to do is to learn approaches from multiple papers and find the right approach for this task, then implement two to three algorithms to compare the accuracy of detection and recognition. The utilized algorithms will be based on the work of [1] [2] [3].

1. Introduction

With the increasing ownership of vehicles, high pressure is imposed towards vehicle-related fields; for example, vehicle license scanning technology has been implemented in China. Electronic Toll Collection (ETC), works similarly to E-Z passes but will scan the plate of the vehicle when a vehicle reaches the highway entrance and show the license and the toll of the vehicle. What's more, many parking lots use such plate scanning technology to gain information of the vehicle in order to calculate the parking charge. Such technology has the ability to avoid traffic jams and gain an environmentally-friendly outcome for parking lots. The three papers analyzed in this work mainly introduce how to extract the precise information of the vehicle (i.e.Vehicle License), captured by sensors.

2. Related Work

2.1 License Plate Recognition using SVM(support vector machine)

This task can be divided into 3 processes, vehicle license extraction, characters segmentation and characters recognition.

2.1.1 License extraction

How to extract the part of license in an image can be seen as how to enhance the pixels of the license and weaken the pixels of other parts, which can be called as pre-process in our task.

Noise is a common element in images, and Gaussian filter is used to remove this effect by replace a pixel by weighted calculation based on its neighbors.

$$I_{\sigma} = I * G_{\sigma}$$

$$G_{\sigma} = \frac{1}{2\pi\sigma} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

I : raw image

G_{σ} : gaussian filter kernal

I_{σ} : new image

σ : standard deviation

It can be considered as using a square to scan the image and recalculate the central pixel, which will be the average of its neighbors.

Gray transformation will transfer the colorful image into gray image.

$$new_value = 0.30 * R + 0.59 * G + 0.11 * B$$

Histogram equalization will help us get a better image to extract the part we need, because it enhances the contrast of the image. Assume there are x_n different pixel values (in order) in the image.

$$f(x) = 255 * \sum_0^{x_i} \frac{h(x_i)}{w * h}$$

x : cuurent pixel

w : weight, h : height

In other words, this will make pixel values distributed evenly.

Edge detection is an important step in extraction process. If considered the gray image as a valley, edges are peaks and troughs, so that extract information to calculate the total gradient based on vertical and horizontal directions. In image processing, Sobel operators can help:

Use Sobel operator to detect the edges.

$$Sx: \begin{matrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{matrix}$$

$$Sy: \begin{matrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

Use these two operators to calculate the gradient of a pixel.

$$\begin{aligned} Gx &= Sx * I \\ Gy &= Sy * I \\ G &= \sqrt{Gx^2 + Gy^2} \end{aligned}$$

If G is larger than a threshold, the current pixel is a part of edge. In an informal way, it replaces the current pixel by calculating the variety in x-direction and y-direction.

Then process image binarization by choosing a threshold to determine whether the new value of a pixel should be 0 or 255 to enhance the information which will be demanded and remove the noise.

$$p = \begin{cases} 255 & p.value > threshold \\ 0 & p.value \leq threshold \end{cases}$$

Dilation and erosion are two important operations in image processing. Dilation fills tiny voids, connect adjacent objects while erosion remove tiny objects and disconnect objects. Through dilate and erode the image in order, which is called close operation.

Contours detection is similar with edge detection, which is find the contour of car license and extract it from raw image.

2.1.2 Characters segmentation

Segmentation is much simpler than extraction. This project the pixel value of car license image after image binarization vertically, and segment the image based on the troughs.

2.1.3 Characters recognition

The support vector machine is used to do recognition. Assume we need to recognize a number, 10 SVMs are needed, because one category in turn is grouped into one category when training. When recognizing, the number image is put into those 10 SVMs and get 10 results, choose the greatest value as a result.

2.2 Experiment

Here are some figures of an experiment.



Figure 1: The raw image taken by sensor with observable license plate.

First of all, get the image like figure 1, and then use gray transformation to such image to figure 2 in order to further recognition for SVM.



Figure 2: After gray transformation, histogram equalization, edge detection, median filter and image binarization.

Then, having figure 2 to close operation and erosion and get figure 3 as follows:



Figure 3: The image after close operation and erosion.

Therefore, there have the contours of the image where the plate's located.



Figure 4: Image we draw contours.

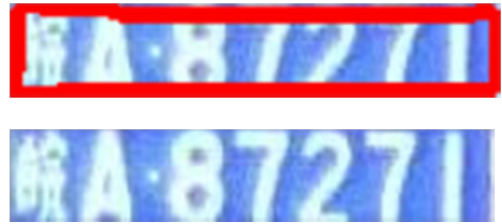


Figure 5: Vehicle license.

After implement the gray image of figure 5 and dilation it in order to get more reliable data, which shown below:



Figure 6.1: Gray image of the license



Figure 6.2: Dilation of the image

Now, below figure is the vertical projection after dilation the license.

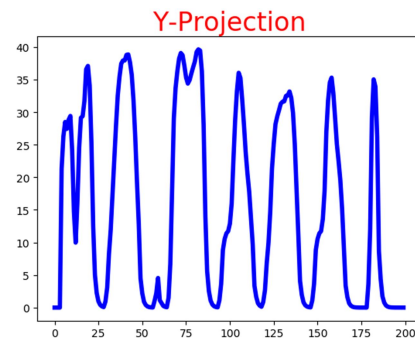


Figure 7: Vertical projection of Figure 6.2

Eventually, getting the characters of the license where shown in figure 8.

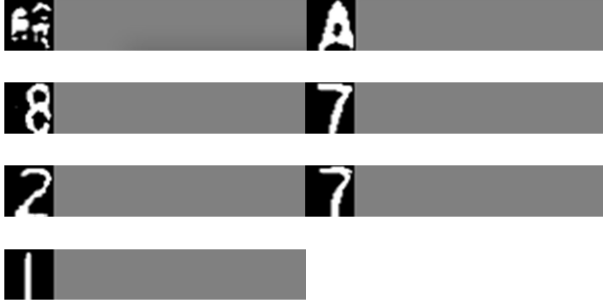


Figure 8: the characters of the car license

Thus, the result from image to get the final recognition by using SVM is figure 9 as follow:

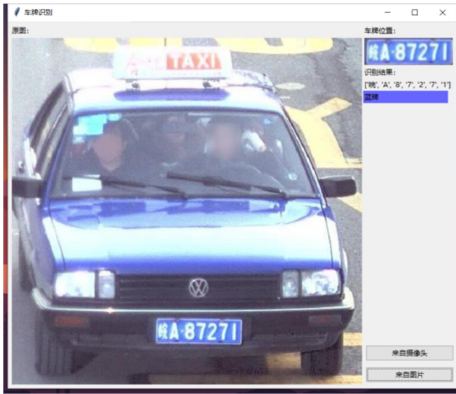


Figure 9: Result from SVM.

This SVM is an open source on

<https://github.com/wzh191920/License-Plate-Recognition>.

2.2 License Plate Recognition using OpenCV and OCR

The thought of this project could be separated to three parts, License Plate Detection, Character Segmentation, and Character Recognition. The first step use contour option in OpenCV to detect for rectangular objects to find the number plate, second step will crop number plate and turn it into a new image, third step would use OCR get the text in this specific image.

2.2.1 Process

For this part, we will introduce algorithms in License Plate Recognition using OpenCV and OCR.

2.2.1.1 BilateralFilter algorithm

The first algorithm is BilateralFilter algorithm and the definition as follows:

BilateralFilter algorithm keep the blur method in Gaussian blur, at the same time it can save the edge information in original picture, in order to get the number plate.

$$GB[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

↓
Space weight

↓
Range weight

In this equation, $GB[I]_p$ is result at pixel p , I_q is the intensity at pixel q , S is the adjacency of p , $\frac{1}{W_p}$ is normalization factor.

In the vertical area of the image, the expected value changes little, and the corresponding predetermined range domain weight is close to 1. At this time, the space weight plays a major role, which is equivalent to performing Gaussian blur; in the edge area of the image, the value changes greatly. Range weights become larger, thus maintaining edge information

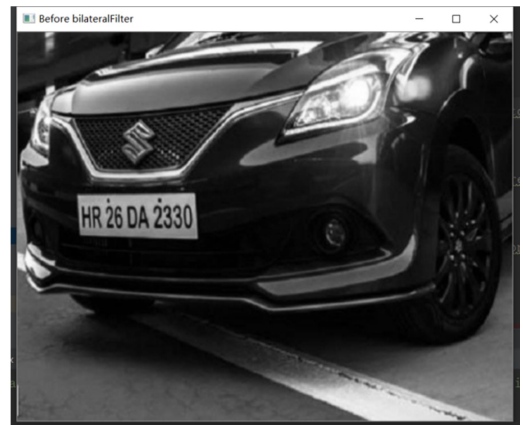


Figure 10: Image before bilateralFilter

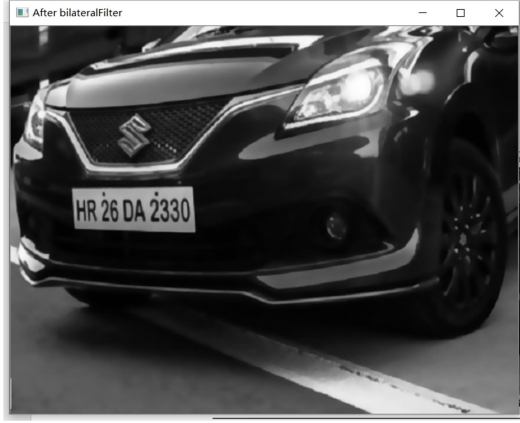


Figure 11: Image after bilateralFilter

2.2.1.2 Canny Edge Detection algorithm

The second algorithm is Canny Edge Detection algorithm. This algorithm is for capturing as many edges as possible in the image. First algorithm calculate the edge gradient of every pixel in image, and select the pixel which meets specific requirements.

The definition of edge and angle are as follows:

$$\text{Edge_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle } (\Theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

A point is considered to be an edge point if it is in vertical direction of gradient direction and it is a local maximum.

After all edges were selected, algorithm uses two threshold values, called minVal, maxVal. Any edges with intensity gradient more than maxVal are sure to be real edges and those below minVal would be discarded. Intensity gradient which are between minVal and maxVal would be selected to be real edges only if they are connected to “sure edge” pixels.

The following figure represents this processing:

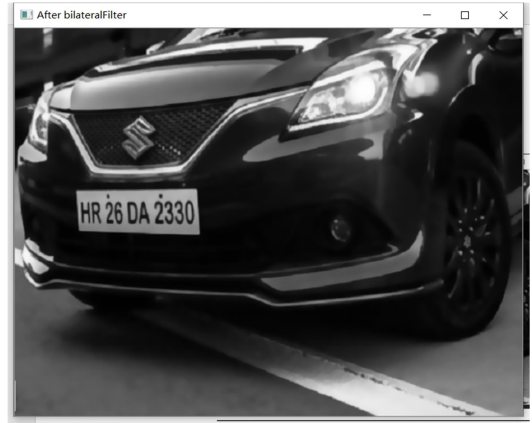


Figure 12: Image before Canny Edge Detection

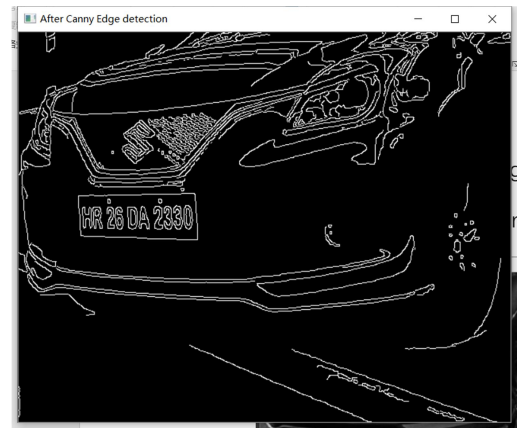


Figure 13: Image after Canny Edge Detection

2.2.1.3 Ramer-Douglas-Peucker algorithm

The main thinking is turning continuous smooth curve into polyline as follow figure shows:

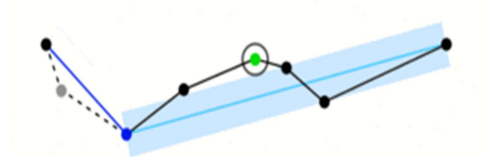


Figure 14: Turing continuous smooth curve into polyline

For getting the contour of number plate, Ramer-Douglas-Peucker algorithm is used here, Frist algorithm set a threshold value, it recursively keep pixel P in a given image, whose distance to the given line is

greater than the threshold value, and the result would be a collection of pixel. Threshold value is called “epsilon”. When “epsilon” becomes smaller, shape of the final poplin would be more like a smooth curve.

Here, if the result contains four pixels, which means that the contour is a rectangular, then it can be recognized as the contour of license plate. Note that this calculation method of threshold value is tested manually, a better way is using machine learning to train based on car images and then use a more suitable value.

The contour in license picture is like this:

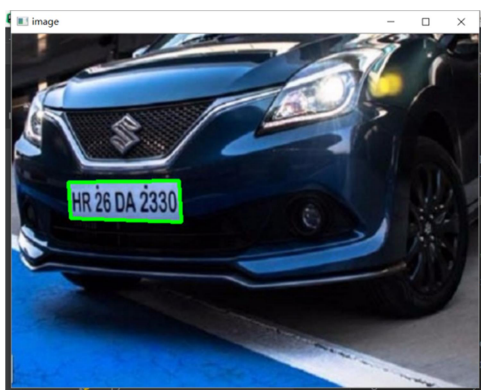


Figure 15: Contour picture in license

2.2.1.4 Tesseract OCR technology:

Tesseract OCR technology is an open-source OCR engine to detect inverse text, and it is the first engine able to handle white-on-black text so trivially. It could be separated to two processes; the first one is to recognize each word in turn, second path is running over the page in which words that were not recognized well enough.

In this application, the first step is embodied as word finding, fixed pitch detection and chopping technology. Word finding was done by organizing text into blobs. Tesseract chops the words into characters using pitch and disables the chopper and associate on these words for next step.

A picture should be like this:



Figure 16: Using OCR to recognize the characters

The next step is word recognition, after getting every character, it still need to further segmentation. The main idea is segement of polygonal approximation, which can separate a characters into different polygon called features and store them in a list for every characters. Features is determined by x, y position and angle. At the same time, a class pruner would create a list of characters classes that feature might match. Each unknown feature would look up to the lists of prototypes of training data, and find the most similar one. Actual similarity is calculated by distance calculation, which saves a record of the total similarity evidence of each feature in each prototype. Tesseract technology improves distance calculation by two numbers: first one is called “confidence”, is minus the normalized distance from the prototype; second one is called rating, multiply “confidence” with total outline length in the unknown character. In this way, features in a word would be summed more accurately since the number of polygon in a word is the same.

The whole process can be represented as follows:

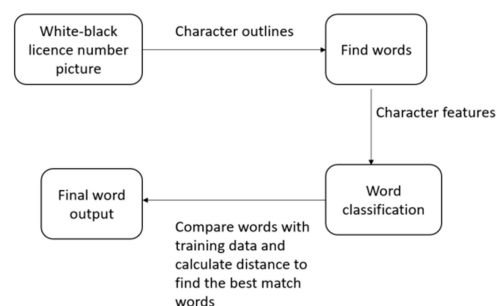


Figure 17: The process of finding the words

In the actual process should be embedding like this:

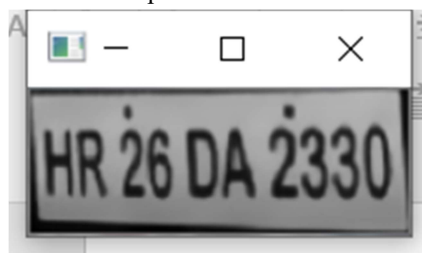


Figure 18: Input license.

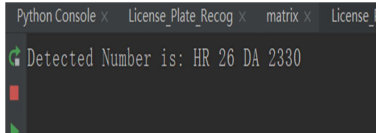


Figure 19: The output of the license.

2.2.2 Experiments

Selected data sets are grouped in two directions. One should be like the camera location which the car is pointing towards. Shown as figure 20 and 21



Figure 20: Example 1



Figure 21: Example 2

Another direction is whether the picture includes the entire car. Shown as figure 22 and 23



Figure 22: Example 3



Figure 23: Example 4

One worst scenario is that the contour fails to detect the license the license correctly; this is because different images have different contour structures, but the Ramer-Douglas-Peucker algorithm uses a certain specifying parameters to get four pixels of the license plate number. Also it is possible that other contour is a rectangle, so it needs to rely on machine learning to train the specific number. Here, 0.018 should be trained if a better result set needed.

Another disadvantage is that if the vehicle doesn't face the camera directly, it would fail to get right word sometimes. Either better direction of image or improving configuration of Tesseract OCR engine.

The reason is that in the actual situation, it cannot be promised that the camera can catch a whole car in a right direction all the time. By comparing different situations, the advantages and disadvantages of every method can be embodied better.

This code is an open source on:

<https://circuitdigest.com/microcontroller-projects/license-plate-recognition-using-raspberry-pi-and-opencv>

2.3 License Plate Recognition using CNN (Convolutional Neural Network)

2.3.1 Pre-processing

Load the image contain the license plate and transfer the RGB image into a Grayscale image using `cv2.COLOR_BGR2GRAY` [22] from Python OpenCV, using gray image can reduce the amount of data in the image and accelerate the edge detect process.

Then, smoothing the image [23] by using blur to reduce some small noise which will affect the identity.

Sobel operator [24] can be used to detect the edge of plate for our further operation.

For the plate always have same certain background color, now we transfer the original RGB image into HSV [25] to check the color and multiply it with the image after Sobel process, we can only keep the parts with the color close to license plates.

Otsu's Binarization [26] can use Grayscale image to generate a Binary image, a binary image can be clear to detect the contour and has less amount of data

To make sure the edge detected of plate is whole, this process need to do closing [27] for the binary image to connect the edges of license plate.



Figure 24: The image to recognize



Figure 25: The image after pre-processing

2.3.2 Position the plate

After the image processing, the program will roughly find the license plate, but there's still too much interference part, therefore, it is vital to eliminate them and keep the plate only.

Get the contour of each part in the image after pre-processing, and calculate the circumscribed rectangle of them, then we can eliminate some part by the length, width and the aspect ratio of circumscribed rectangle for license plate has some certain aspect ratio.

Now we use Flood Fill [28] to combine color check and shape check.

Mask [29] can be used to block the word characters on the license plate to check the background color. By generating some seed points in the rectangular area, the range of those seed points is the length and width of its circumscribed rectangle, and the color of those seed points need to be the color of real license plate. Then calculate the circumscribed rectangle after doing mask operates to do shape check by aspect ratio.

Even after this, there still can be some part like license plate, therefore, the next step is to filter them.



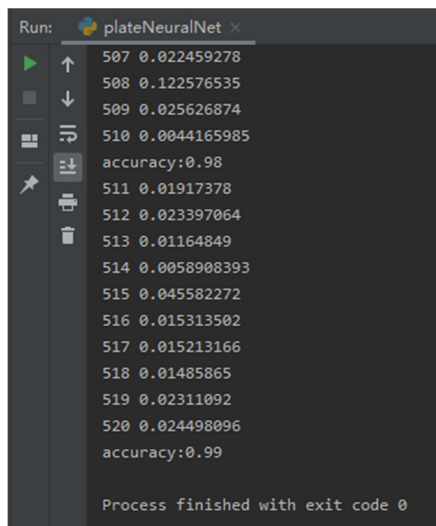
Figure 26: Identify the license's position by the shape

2.3.3 License plate recognition

The majority of this step is to determine if a part with a shape looks like a plate and has a background color close to license plate.

Convolutional Neural Network (CNN) is good at image processing [30]. For a image can contain large amount of pixels, which means a huge amount of data, CNN can simplify complex issues by reduce the large number of parameters in to a small number, like human's eyes, human can easily identity whether a dog or a cat is in the image even reduce the image from 1000 pix to 200 pix, so dose CNN.

We have a dataset to train CNN, it contains some image of real license plate and some looks like a real one. In plateNeuralNet.py it can be trained to learn the feature of real license plate from the dataset and create a meta file for further identify.



```
Run: plateNeuralNet x
507 0.022459278
508 0.122576535
509 0.025626874
510 0.0044165985
accuracy:0.98
511 0.01917378
512 0.023397064
513 0.01164849
514 0.0058908393
515 0.045582272
516 0.015313502
517 0.015213166
518 0.01485865
519 0.02311092
520 0.024498096
accuracy:0.99
Process finished with exit code 0
```

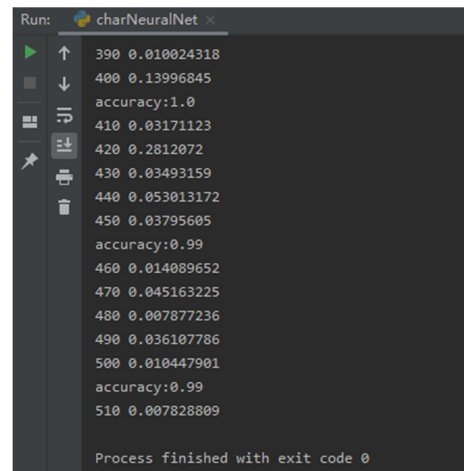
Figure 27: The result of CNN training in license plate recognition after 520 learning.

2.3.4 Character segmentation

Horizontal projection for the binary image, we can get a longest continuous projection as the character area. Because characters are always separated by a gap, they can be used as the rules for vertical projection to segment characters. And the width of each segmented part must reach the average width to be counted as a character to remove the symbol and small noise.

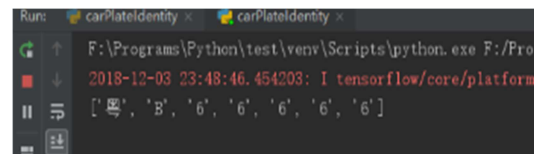
2.3.5 Character recognition

We still use Convolutional Neural Network (CNN) to identify characters from the segmented parts above. There is a data-set of many kinds of characters for CNN to learn, also generate a meta file. While identifying from a segmented part, take the most possible as the output.



```
Run: charNeuralNet x
390 0.010024318
400 0.13996845
accuracy:1.0
410 0.03171123
420 0.2812072
430 0.03493159
440 0.053013172
450 0.03795605
accuracy:0.99
460 0.014089652
470 0.045163225
480 0.007877236
490 0.036107786
500 0.010447901
accuracy:0.99
510 0.007828809
Process finished with exit code 0
```

Figure 28: The result of CNN training in character recognition after 510 learning



```
Run: carPlateIdentity x
F:\Programs\Python\test\venv\Scripts\python.exe F:/Pro
2018-12-03 23:48:46.454203: I tensorflow/core/platform
[' ', 'B', '6', '6', '6', '6', '6']
```

Figure 29: The output

Limitations

For the License Plate Recognition using SVM (support vector machine), the disadvantages are as follows:

The SVM construction method we use is called one against all, in which the training set is biased because of our division method. For example, when training the SVM which recognizes number 1, we divide the training set into [1] and [0,2,3,4,5,6,7,8,9]; the latter set is much larger than the previous one. One against all construction method might be unreliable sometimes. <The uneven result of the divided set means that a smaller percentage of the available training data is able to be used for any given number. (?)>

For the License Plate Recognition using OpenCV

and OCR, the disadvantages are as follows:

One worst scenario is that the contour fails to detect the license correctly; this is because different images have different contour structures, but the Ramer-Douglas-Peucker algorithm uses a certain specifying parameters to get four pixels of the license plate number. Also it is possible that other contour is a rectangle, so it needs to rely on machine learning to train the specific number. Here, 0.018 should be trained if a better result set is needed.

Another disadvantage is that if the vehicle doesn't face the camera directly, it would fail to get the right word sometimes. Either better direction of image or improving configuration of Tesseract OCR engine would be needed.

For the License Plate Recognition using CNN (Convolutional Neural Network), the disadvantages are as follows:

When the plate in the image is severely tilted, the efficiency cannot be promised. The accuracy of CNN for license plate and character recognition needs to be improved because it relies heavily on the size of the dataset. With increased dataset size, the outcome will be promising. A second disadvantage is that there is background color check in plate recognition; some outdoor scenes can affect the recognition of license plate with blue and yellow background colors.

Conclusion

In this essay, we explore different papers on different aspects of the detection of vehicle licenses in different usage situations. These post significant outcomes towards the datasets and algorithm in the essay for us to fully understand the details of such technologies in the papers and how to utilize them in real life and what are the pros and cons of them. Of all the papers we fully studied, those authors utilize their creativity and knowledge to solve their problems together with both classical methods and AI methods in order to create a more efficient and effective as well as precise way to detect the vehicle plate.

Every method could be separated into three processes: pre-processing, character segmentation, and

character recognition. The differences are the actual algorithms that are used in the three processes.

OpenCV with OCR is a relatively traditional solution: the first two processes are based on the attribute of the pixel of the picture, finding the specific image using geometry knowledge. The third step is a relatively new technology, which can turn graphics into words by using a supervised learning model.

SVM works relatively well if there are clear margins of separation between classes and it is proved that it works well on number recognition. Meanwhile, SVM is effective when characters are in high dimensional spaces like taking all pixels as characters in plate recognition. SVM is also effective when number of dimensions is greater than the number of samples, which means a higher quality model can be created with relatively fewer training data.

CNN simplifies complex issues by reducing a large number of parameters into a small number; in this way it can improve efficiency and accuracy largely.

It is vital to search for more efficient and effective solutions towards vehicle license recognition algorithms. Researchers are interested in how to improve the accuracy of training data-set in order to learn more efficiently, but the most important thing is that researchers spend time constructing a system to enable them to capture license plate photos at different shooting angles and different image blurs.

References

- [1]. Z. Selmi, M. Ben Halima and A. M. Alimi, "Deep Learning System for Automatic License Plate Detection and Recognition," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, 2017, pp. 1132-1138.
- [2]. Rahul R. Palekar, Sushant U. Parab, Dhruvil P. Parikh, Vijaya N. Kamble, "Real time license plate detection using openCV and tesseract," 2017 International Conference on Communication and Signal Processing (ICCSP), India, 2017
- [3]. S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape

- contexts," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, April 2002.
- [4]. Y. Qiu, M. Sun and W. Zhou, "License Plate Extraction Based on Vertical Edge Detection and Mathematical Morphology," 2009 International Conference on Computational Intelligence and Software Engineering, Wuhan, 2009, pp. 1-5.
- [5]. Shi X., Zhao W., Shen Y. (2005) Automatic License Plate Recognition System Based on Color Image Processing. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2005. ICCSA 2005. Lecture Notes in Computer Science, vol 3483. Springer, Berlin, Heidelberg
- [6]. H. Turki, M. Ben Halima and A. M. Alimi, "Text detection in natural scene images using two masks filtering," 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, 2016, pp. 1-6.
- [7]. N. Wang, X. Zhu and J. Zhang, "License Plate Segmentation and Recognition of Chinese Vehicle Based on BPNN," 2016 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, 2016, pp. 403-406.
- [8]. T. D. Duan, T. L. Hong Du, T. V. Phuoc, and N. V. Hoang, "Building an automatic vehicle license plate recognition system," in Proc. Int. Conf. Comput. Sci. RIVF, 2005, pp. 59–63
- [9]. Kim KB., Jang SW., Kim CK. (2003) Recognition of Car License Plate by Using Dynamical Thresholding Method and Enhanced Neural Networks. In: Petkov N., Westenberg M.A. (eds) Computer Analysis of Images and Patterns. CAIP 2003. Lecture Notes in Computer Science, vol 2756. Springer, Berlin, Heidelberg
- [10]. S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, and T. Shiose, "A novel adaptive morphological approach for degraded character image segmentation," Pattern Recognit., vol. 38, no. 11, Nov. 2005, pp. 1961–1975,
- [11]. "One Against One" or "One Against All" : Which One is Better for Handwriting Recognition with SVMs? Jonathan Milgram, Mohamed Cheriet, Robert Sabourin
- [12]. SVM Based License Plate Recognition System, Kumar Parasuraman, Member IEEE and Subin P.S.
- [13]. Image Classification Using SVMs: One-against-One Vs One-against-All, Gidudu Anthony, Hulley Gregg, Marwala Tshilidzi
- [14]. <https://blog.csdn.net/piaoxuezhong/article/details/78302920>. Retrieved April 1, 2020.
- [15]. <https://www.geeksforgeeks.org/python-bilateral-filtering/>. Retrieved April 1, 2020.
- [16]. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html. Retrieved April 1, 2020.
- [17]. https://blog.csdn.net/qq_27278957/article/details/84749993. Retrieved April 1, 2020.
- [18]. <https://blog.csdn.net/qingyafan/article/details/53157609>. Retrieved April 1, 2020.
- [19]. <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf>. Retrieved April 1, 2020.
- [20]. <https://nanonets.com/blog/ocr-with-tesseract/#technologyhowitworks>. Retrieved April 1, 2020.
- [21]. <https://github.com/wzh191920/License-Plate-Recognition>. Retrieved April 1, 2020.
- [22]. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html Retrieved April 2, 2020.
- [23]. https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html Retrieved April 2, 2020.
- [24]. https://en.wikipedia.org/wiki/Sobel_operator Retrieved April 2, 2020.
- [25]. https://blog.csdn.net/taily_duan/article/details/51506776 Retrieved April 2, 2020.
- [26]. https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html Retrieved April 2, 2020.
- [27]. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/close.htm> Retrieved April 2, 2020.
- [28]. https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html#highlight

[=floodfill](#) Retrieved April 2, 2020.

[29]. https://docs.opencv.org/3.4/d7/d37/tutorial_mat_mask_operations.html Retrieved April 2, 2020.

[30]. <https://missinglink.ai/guides/computer-vision/opencv-deep-learning/> Retrieved April 2, 2020.

[31]. https://blog.csdn.net/GK_2014/article/details/84779166?depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1&utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1 Retrieved April 2, 2020.