Instruction for license plate reader

Shanyi Jiang (sj3358) & Sheng Tong (st4048) & Haotian Wu (hw2716)

Contents

Before using the code

Introduction:

The application of artificial intelligence and computer vision has greatly improved every aspect of our lives nowadays. And as an important branch, autopiloting has become more popular than any other period of time in our social development. Specifically, a reliable tool that allows machines to detect the car license from an image will be our main focus in this report. So, the goal of this project is to locate and identify car license plates from an image based on some computer vision techniques. And in this report, we will introduce methods, performances and future improvement of our program.

We have developed the code to try to automatically locate car license plates and recognize letters on plates. And there are two version of our code, normal python program version and jupyter notebook version. You can choose either one to run and test our code.

Using this code, you can automatically capture car license plates, and read the license number of them. You can also adjust the parameters on your own to get better performance and achieve better accuracy.

Package used:

Before using our code, you need to install following packages:

cv2
numpy
imutils
matplotlib
math
skimage
os
shutil

Data used:

We use the Indian Number Plate Detection on Kaggle to develop and test our code, some of the pictures are already included in the zip package along with the code. Here is the link for this dataset: Indian Number Plate Detection | Kaggle

Python version

How to run the code:

It's a normal python program with several .py files and some picture files. You can run the code like any other small python program.

There are two parts of our methods, one is letter cutting that locate the license and cut all letters out and save them as separate pictures; and the other one is recognizing those letters and print the matching results.

You can run the code in cmd, type

$$pyhthon\ cut.py$$

to perform the cutting process, during the process if you have set debug = True, you can watch the whole process of locating the plate, first primary letter cut, secondary refinement and final letter cuts step by step. Then you can adjust parameters in the code to get better cutting results based on what you have seen. An ideal result of secondary refinement is that every letter is in the upright position. This process is very fast, so you can try many times to find best parameters to get best results.

After cutting process, pieces of letter cuts are saved in "out" directory, then in the cmd, you can type

$$pyhthon\ match.py$$

to perform the matching and recognizing process, every letter piece will match with all our templates saved in "newt" directory, and the program will return the matching result with letters on the plate in order. This step takes a bit of long time, about 1-2 minute, there are less parameters adjustable here, but you can still try to find better parameter sets.

How to adjust parameters:

①In cutting process (when you use python cut.py), there are five parameters you can adjust:

1. img_name:

the name of car license plate image you are going to read, they are saved in “in” directory, you can pick any of them. If you have debug mode on, when you run the code, there shows an image with a red rectangle surrounding the car license plate, it means that plate locating succeeds.

2. mask_threshold:

the threshold for making binary image to perform letter cutting later. If you have debug mode on, when you run the code, there shows a binary image of the license plate. You should decrease it if there are noise points that you don't want to take into consideration, and increase it if parts of letters are lost.

3. second_refine_ratio:

ratio for secondary refinement, it controls the power of secondary refinement which rotate the plate image and try to make it upright. If you have debug mode on, when you run the code, there shows two images, one called “before”, one called “after”, we perform secondary refinement on “before” and get “after”. Decrease it if the picture rotates too much in the secondary refinement, increase it if the secondary refinement is not enough, you can even set a negative value for it to rotate the image in a reverse direction.

4. sideCut:

A Boolean parameter to decide whether perform side cut, it will cut the boundary of the plate to avoid influence from uncleaned plate boundary. You can set it to False if you lost parts of letters.

5. debug

If debug = True, debug mode is on, it will allow you to look at every step of the process so you can easily adjust the parameter

② And in matching process (when you use python match.py), there are four parameters you can adjust:

1. img_name:
   the name of car license plate you are going to match their letters, they should already be saved in “out” directory, so this parameter is set to be equal to the same name parameter in cut.py.

2. binary_letters_threshold:
   the threshold for making the letter pieces binary to perform latter template matching method. It is set to be equal to the mask_threshold in cut.py, usually you do not need to adjust it. But you can decrease and increase it to change the template matching results. Increase it will cause more parts of letter pieces to be valid in template matching, and decrease it will cause less parts to be valid.

3. down_sample_scale:
   The down sample rate we use in template matching process to accelerate the computing process, increase to get faster performance, decrease to get better result.

4. penalty:
   penalty in the template matching for one pixel if letter doesn't match the template, we have done many experiments to find out that 0.2 is a good value for this parameter to get an good average accuracy, but you can change it as you like. The value for this parameter should be positive.

How to add your own files:

You can add your own car license plate images in “im” directory.
And you can add your own template in “newt/_all” directory, the name for the template file should be “X_Y.pny” where X is the letter of the template, Y is the number of this template.

Jupyter notebook version

How to run the code:

In the version of jupyter notebook, you can run our code with high interactivity and get responses in real time, which helps you better adjust the parameter.

Open the code in jupyter notebook, run all the blocks that import packages and define functions. The last two blocks area what you need to run. The first block is nearly same as cut.py in python version, and the second block is nearly same as match.py in python version. But you do not need to type code in the cmd and change parameters in the code every time, you can repeatedly run same block several to find best parameters. By the way, in jupyter notebook version, we do not save letter pieces somewhere, we save them as a list in the program.

How to adjust parameters:

The parameters are exactly same as in python version, the only two differences is that first in debug mode we show most of our process in jupyter notebook rather than a window, second there is no another img_name for matching part.

Please refer to “How to adjust parameters” in python version for how to adjust each of these parameters.

How to add your own files:

You can add your own car license plate images in “pictures” directory.
And you can add your own template in “newt/_all” directory, the name for the template file should be “X_Y.pny” where X is the letter of the template, Y is the number of this template.