Digital Image Processing Report
Project 4: License Plate Reading

2019233181 Jiale Xu
xujl1@shanghaitech.edu.cn
2019233167 Shiwei Lv
lvsw@shanghaitech.edu.cn

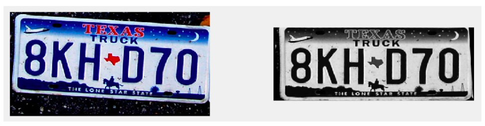
December 28, 2019

1 License Plate Detection

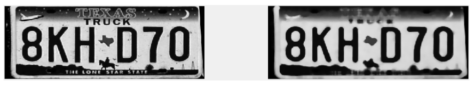
2 Character Segmentation and Recognition

2.1 Method

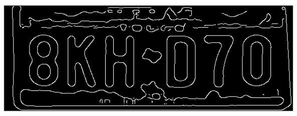
Given an extracted license plate image from the first section, we are now trying to segment and recognize the characters on it. Fisrt of all, we need to transform the image so that the characters on it is horizontal:



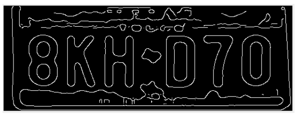
Then, we use median filter to reduce noise, which can be implemented by function *medfilt2*:



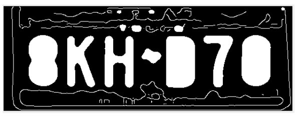
Then, we use a canny detector to detect the edges, which can be implemented by function *edge*:



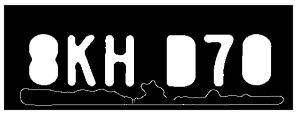
Then, we perform closing operation with a srtuture element to smooth contours and fill gaps in the contour. The function *imclose* can do this job:



After that, we fill the holes on the edge image to obtain regions, this can be implemented by function *imfill*:

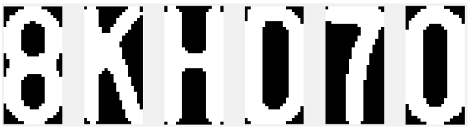


As we can see, the characters and some other small regions were filled. Therefore, we need to remove the small regions. This can be impleted by function *bwareaopen*:



Now the small regions have been removed, but there are still some 'line regions'. We perfoem an erision operation to remove them:

Now, for each region, we can obtain its bounding box and cut the corresponding area from the denoised image. Then we binarize the cutted image, and this is a segmented character. We repeat this operation for all regions and obtian the characters' segmentation result:



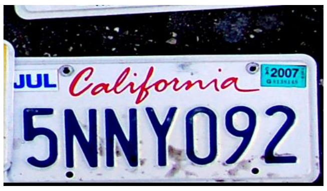
And the last step is to recognize the characters by matching it with templates. The matching process is accomplished with the normalized correlation function, which can be implemented by function *corr2*:

$$\operatorname{corr}2=\frac{\sum_m\sum_n\left(T_{mn}-\bar{T}\right)\left(O_{mn}-\bar{O}\right)}{\sqrt{\left(\sum_m\sum_n\left(T_{mn}-\bar{T}\right)\right)^2\left(\sum_m\sum_n\left(O_{mn}-\bar{O}\right)\right)^2}}$$

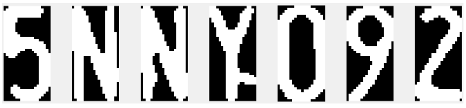
Where T is the template image, O is the image of the object, m and n are the image sizes that must be the same, $\bar{T}$ is the mean of T, and $\bar{O}$ is the mean of O.

2.2 Result

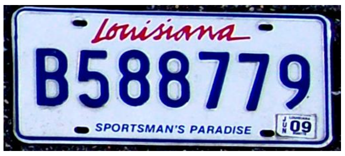
The recognizing result of the example above is '8KHD70', which is correct. More results will be listed below.
Input:



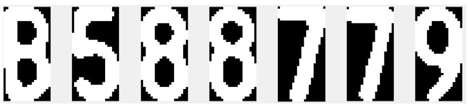
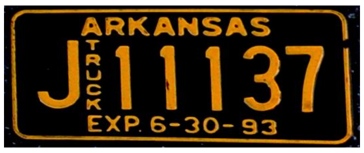
Output: '5NNY092'

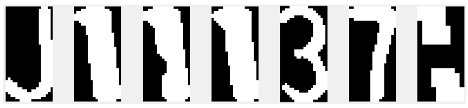


Input:

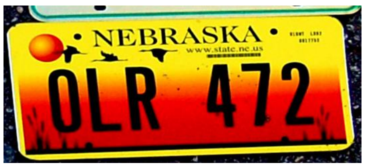


Output: 'B588779'

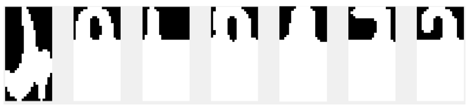





Input:
Output: 'J111371'
Input:



Output: '1442A32'



3 Optional

3.1 Method

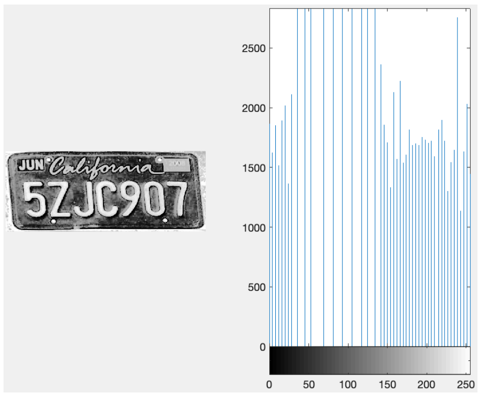
From the results of the last section, we can see that there are many failed cases produced by the described method above. Therefore, we will try to improve the method so as to obtain better segmentation and recognition results.
Fist of all, we make the image horizontal:

Then, we make sure that the license plate image has dark (low intensity) background and bright foreground (high intensity) which is the characters' area we are interested in. We can do this statistically with the mean value of the whole image:

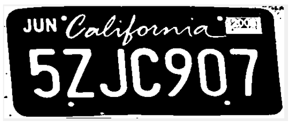
```
image_mean = mean(image, 'all');
if image_mean > 100
   image = 255 - image;
end
```

The result is:

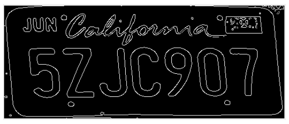
Then, we perform histogram equalization on the image:



Then, we binarize the image immediately with a threshold 0.75, which is observed that can distinguish the character foreground from the background exactly.

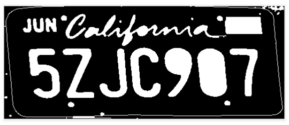


Then, we use Canny detector to detect the edges:

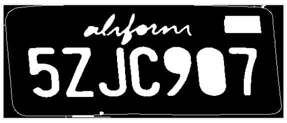


Then, we perform closing operation with a srtuture element to smooth contours and fill gaps in the contour:

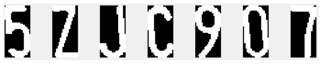
After that, we fill the holes on the edge image to obtain regions, this can be implemented by function *imfill*:



As we can see, the characters and some other small regions were filled. Therefore, we need to remove the small regions. This can be impleted by function *bwareaopen*:



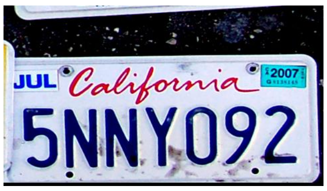
Now, we have a seris of regions. To filter the non-character regions, we can use the ratios of regions to achieve this. Suppose a region's height and width are h and w, respectively. And the height and width of the whole image is H and W, respectively. If $0.3 \leq h/H \leq 0.8, w/W \leq 0.2, 1.0 \leq h/w \leq 5.0$, we will regard this region as a character. We cut these regions from the binarized image, the result is:



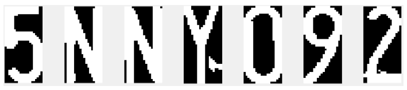
The final step is to recognize the characters, the method is the same as described in the last section.

3.2 Result

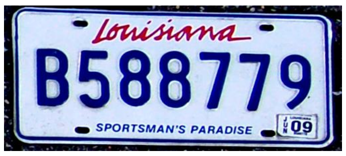
The recognizing result of the example above is '5ZJC907', which is correct. More results will be listed below.
Input:



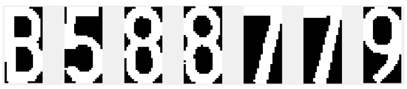
Output: '5NNY092'



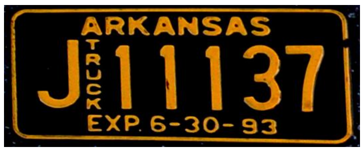
Input:



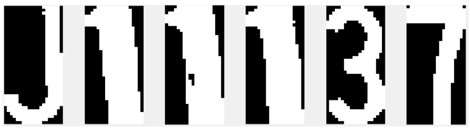
Output: 'B588779'



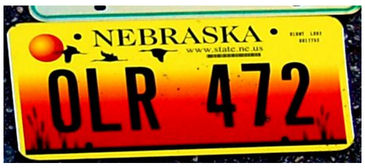
Input:



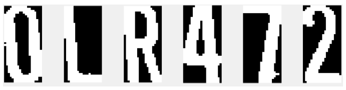
Output: 'J11137'



Input:



Output: '3LR472'



From the results we can see that this improved method can segment those images which were mistakely segmented by the former method correctly.