Computer Vision

Final Project

Automatic Licence Plate Reading

Bressan Giulia
I.D. 1206752
A.Y. 2018/2019

In the following report I present the main steps and tasks elaborated for the development of the project, whose objective was the automatic license plate recognition and reading in some provided car images.
The report is structured as follows: in the first section I am going to present the main steps of the procedure used for the recognition and extraction of the plate, in the second section I will explain the preprocessing and selection procedure used to extract the individual characters of the plate and, finally, in the third section, I will present the algorithm used to classify and read each character.

1 Plate Extraction

The plate extraction is performed in the following steps:

- Image preprocessing
- Contours extraction
- Searching for characters
- Plate extraction from the image

1.1 Image preprocessing

The first step for the plate extraction is the preprocessing of the original image. This is done in order to obtain a better result in the following steps of the pipeline.
At the beginning the image is converted to grey scale, then, to enhance the contrast, a morphology transformation is applied. Top hat and black hat images are computed, and then they are respectively added and subtracted to the grey scale image found before.
After that, a Gaussian filter is applied to the resulting image, in order to smooth edges of the image. The parameters were set manually, searching for the best result.
Finally, a threshold is applied to the image. I used an adaptive threshold, where its value is calculated individually for each pixel.
The output of the process is a binary (inverted, from the settings of the adaptive threshold) image. An example is shown in Figure 1, on the right side. We can see how both the plate and the characters inside of it are well recognizable.

1.2 Contours extraction

The following step is to extract the contours from the preprocessed image. This is done by using the function "findContours", that finds contours in a

Figure 1: *Original image (left) and processed image (right)*

binary image, as the one we obtained from the preprocessing part. These contours are then used to extract the possible characters from image, which are contained in the plate.
An example of the contours extraction is show in Figure 2.

Figure 2: *Contours image*

1.3 Searching for characters

In order to localize the plate position in the image I searched for the possible contours of a character in the contours image. This is done by constructing a bounding rectangle that contains each contour in the image and filtering them by their area, width, height and ratio between width and height.

The values of these parameters were hand tuned, in order to leave the contours corresponding to characters and discard the others.
The result of this first filtering is shown in Figure 3.

Figure 3: *First filtering of the contours to extract the characters, with the bounding rectangles.*

The next step is performed in order to refine the selection of the contours corresponding to the plate characters. To do so I selected only the contours from the previous filtering that are at a certain distance and angle between each other, and they have similar area, width and height. In order to obtain the expected result, the character were sorted by its position in the image, before performing the refinement.
From Figure 4 we can see that only actual characters needed to properly locate the plate are left in the image.

1.4 Plate extraction from the image

The final step of this section is the extraction of the plate from the original image.
The location of the plate is computed by looking at the position of the first and last contours of the characters previously found. As the images of the plates are relatively simple and the plates are not too much inclined I decided not to take into account the inclination of the bounding rectangles of the characters, even though this step should be implemented in order to obtain a better result in worse conditions.
Once the position of the plate is found in the image, the extraction of the sub-image of the plate is performed. The output image is a resized - in

Figure 4: *Second filtering of the contours to leave only actual characters, with the bounding rectangles.*

order to have all the plates of the same dimension for each image - and interpolated (with Lanczos method) version of the extracted plate.
Figure 5 shows all the extracted plates.

Figure 5: *Extracted plates.*

2 Characters Extraction

In the second part of the algorithm the accurate extraction of each single character in the plate is performed.

The procedure is similar to the one performed for the plate extraction, but more precise and accurate.
First, a preprocessing of the plate is performed. The pipeline I followed is made of the following steps:

- Conversion to grey scale
- Denoising of the image, with Non-Local Means algorithm
- Thresholding with adaptive threshold
- Erosion with a rectangular structuring element

Figure 6: *Preprocessed plate.*

The output is shown in Figure 6.
After the preprocessing the contours are found, as it was done for the whole plate extraction, and the characters are selected.
Character selection is done by filtering the bounding rectangles of the contours by their area, weight, size and ratio between weight and height, with the value of the parameters selected by hand looking at the size of the rectangles.
An example of the result is shown in Figure 7. We can see that all the characters are found and can then be extracted. This is true for the whole set of images.
The final step is the sorting of the extracted characters and their extraction from the binary image, so that they can be used in the classifier.

3 Classification

The last task the algorithm performs is the reading of the characters in the plate. This is done with the K-Nearest Neighbours algorithm and a dataset created by merging different plate characters from other datasets.

Figure 7: *Contours of the characters in the plate, with the bounding rectangles.*

In order to obtain the training samples in a layout that is supported by the "kNearest" algorithm in OpenCV, the images are reshaped as flattened and saved in a single matrix, with a number of rows equal to the training samples. The labelling of the samples is done following the Unicode encoding.
The same procedure is done with the images of the characters to be classified, after a resize of each image, in order to obtain the same size as the training images.
The results of the plate reading for the whole set of images is shown below.

4 Conclusion

In general, the plate and character extraction part of the algorithm works fine for the whole set of images.
Despite that, we can see from the reported results that the classification fails with some characters. This may happen due to the fact that the training set is not very huge, but also to the quality of the characters extracted from the plate.
For example, Image 1 shows the worst classification result and we can see that the quality of the plate in the whole image is way worse than in the other images.
In Image 2 the final E is recognized as an F, in Image 3 the O is classified as a D, and in Image 4 the G is read as a C. All these errors are due to a poor performance of the classification algorithm.
For the other images, all the characters are correctly classified.

Figure 8: *Image 0.*

Figure 9: *Image 1.*

Figure 10: *Image 2.*

Figure 11: *Image 3.*

Figure 12: *Image 4.*

Figure 13: *Image 5.*

Figure 14: *Image 6.*