Course of Computer Vision, Final Project: Autonomous Driving: Road Sign Recognition

Andrea Ziggiotto 1140706

July 9, 2018

1 Goal of the project

In the field of the autonomous driving many computer vision systems are used and one of these is the recognition system for the road sign in order to allow the autonomous navigation system to understand them.
The goal of this project is to detect and recognize some road sign in a series of image, in particular the application must detect the warning sign, detected and recognize the velocity limit in the specific sign and find the no parking sign.

2 Description of the code

In order to achieve the goal of the project , input images are preprocessed in order to find circular and triangular forms. In the preprocessing part the input image is converted from a color image to a grayscale image and two gaussian filter with size 3 and 2 are executed subsequently with the aim to partially remove the noise.
After that, with the goal of retrieving the edge of the image a Canny algorithm is applied with an lower and higher threshold depending to the median value (*median*) of the intensity values in the image. In particular the lower and the higher threshold are computed, using a *sigma* value of 0.33, in this way:

$$T_L = max\{(1 - sigma) * median, 0\} \quad (1) \qquad T_L = min\{(1 + sigma) * median, 255\} \quad (2)$$

Once is computed the output of the Canny algorithm the *findContours* function of the *OpenCV* library is used to take contours from the figure. All the contours are divide by the number of points that they have, in particular if a contour has from 10 to 20 points is considered a circle, if instead a contour has exactly 3 points is considered a triangle and finally if contour has from 4 to 5 points is considered a spurious polygon that could be approximate to a triangle.
The presence of spurious polygon is due to the noise in the image that could modify contours. When all interesting contours are placed in the correct list , for every of these lists a different procedure is execute.

2.1 Circle Sign

For every item in the candidate circle list, the *minEnclosingCircle* is computed, this function takes in input a contour and gives in output the center and the radius of the minimum circle that wraps a specific contours. Every computed circle is checked if there is another circle with the same center and a larger radius or if there is another circle that wraps the considered one, if the circle pass both of this control the circle is added to the list of valid circles.
Once the list of circles is computed, for every circle a region of interest of the input image that wraps the circle is taken and inserted to a list of region of interest, this operation is done by checking if the region of interest goes outside of the picture.
For every of this region an threshold with the aim to take only the black region is used to build a mask that includes digits of the speed limit, using this mask the function that gives in output the

contours is used to extract the contours that represent all digits. All of these contours are processed with *boundingRect* function that gives in output an rectangle that could contained only a digit, to avoid the presence of noise, rectangles near the border are deleted and rectangles contained in another one are not considered.
At the end of this step there is a series of rectangles for every region and every rectangle is pass to the OCR with the aim to discovered the digit that appears in the rectangle, when all the rectangles of a region is computed the resulting speed limit is build and a check for the resulting number is make using a dataset of possible speed limit.
OCR (Optical character reader) is a conversion of the image in the typed that are draw in the image, in this particular project the only interesting character are the digit (0-9) that compose the speed limit. The classifier used in this project is the *OCR Beam Search CNN model* given in the *OpenCV* library.
In the output image, speed limit signs are highlighted with a blue circle that wraps the sign and writing on the sign the retrieved speed limit.

2.2 Triangle Sign

For every item in the candidate triangle list, the *minEnclosingTriangle* is computed, this function takes in input a contour and gives in output the three vertex of the triangle that wraps a specific contours. Every computed triangle before being inserted in the list of valid triangle is checked to avoid the presence of another triangle with all the vertex near to the considered one, to make this a threshold for the distance is used.
Due to the noise present in the image, most triangle sign have more than 3 point and for these a special procedure is executed. For every item in the list of the spurious polygon, points that below to the contour are checked and if two points are near, only one is keep. Subsequently a check for the angular coefficient of every segment in the triangle is made to avoid the presence of false triangle that collapses to a segment. Once all these controls are made the candidate triangle is added to the list of triangle after the check of the presence of another triangle in the list with vertexes near to the considered one.
Executing the algorithm, the output figure shows other triangles that are not triangle signs, for this problem a control of the length of triangle segments is made, with this check all the triangle that is not similar to an equilateral triangle is discard, because the triangle sign is a equilateral triangle.
In the output image retrieved triangle signs are highlighted with an green triangle that wraps the sign.

2.3 No parking sign

To retrieve no parking sign after trying to use an SIFT feature descriptor with a brute force matcher that giving not successfully results, it has been trained a Haar classifier but the resulting are poorly because there are many false positives.
To train the classifier, two routine of *openCV* are used: the first is *create_sample* that permits to build a number of positive samples that are distorted version of images given in input, the second is *opencv_traincascade* that is used to train the classifier.
In the training of this classifier 25 images are used to create 2500 positive samples and 1600 are used as negative samples, the number of round of training is 20.
Images used in the training are taken from the dataset *(http://www.dis.uniroma1.it/ bloisi/ds/dits.html)*, the 25 positive image are cropped image of the no parking sign instead the 1600 negative samples are composed by images of the other sign in the dataset. As mentioned above results with this classifier are not good maybe because there are few negative samples not related to sign image. To obtain better results the dataset of negatives samples had to grow up and understand other object such as tree or part of road.
For using the trained classifier is used the method *detectMultiScale* that performs the detection and gives in output a vector of rectangle, subsequently all these rectangles are draw in the output image.

3 Conclusions and Results

In this algorithm are used some space threshold used to make some checks, these threshold are roach because images in the dataset have different sizes. Another threshold is used to take black regions

for the digit segmentation, for this threshold there is a value that works well in every image except one, this value is [180, 255, 90] in the HSV color space.
In the attached folder with the resulted image it is possible to see all the triangle signs are correctly detected except one because is partially covered by another sign [Fig.1e].
In the case of the speed limit sign all the signs are discovered except ones in the highway picture due to the problem with the color threshold, changing this threshold it is possible to retrieve those speed limit.

(a) (b) (c) (d) (e) (f)

Figure 1: Example of software output