Detection and recognition of traffic signs

Luca Rinelli
Politecnico di Torino
Turin, Italy
luca.rinelli@studenti.polito.it

Maicol Dolci
Politecnico di Milano
Milan, Italy
up201803710@fe.up.pt

Tiago Abreu
FEUP
Porto, Portugal
up201404197@fe.up.pt

***Abstract*—The present work servers as a set of algorithms for traffic sign detection, classification and recognition given a reasonable image. In these tasks color segmentation is used together with edge detection to construct binary images that are then filtered and analyzed based on their properties. Lines and circles detection with Hough transform is used to confirm some results. Detection can be calibrated to reach precision over 90% with a recall over 80% on the test cases provided. The classifier reaches 100% precision and recall. Good results are obtained also with the recognizer.**

I. Methods

A. Detection

The script for the detection of the signals starts by color thresholding the image in 4 binary masks

- just_red
- just_blue
- just_whitish
- just_yellow

from the hsv color space. Edges for a gray version of the images are computed. With the goal of obtaining many separated regions in the colored masks, and to remove ambiguous regions

- just_red and just_blue are subtracted from just_whitish
- and what was in just_whitish is subtracted from just_red and just_blue
- then the edges are subtracted from just_blue, just_red and just_whitish

At this point to improve performances we remove very small regions with a hit-or-miss filter that uses multiple structuring element together to avoid destruction of relevant information in the masks. The just_yellow mask is filtered easily with an 'open' using a diamond structuring element. The next step is the labeling of each region in each binary image and the extraction of region properties, at this stage we discard all the regions that do not meet specific constraints on the bounding box size

- regions that are too big or too small are discarded
- regions that are too wide or too tall are discarded

At this stage the script starts collecting 'hypothesis', that are basically the bounding boxes of regions satisfying the previous criteria. Special attention goes to

- just_whitish and just_yellow bounding boxes, that are enlarged uniformly in all the directions before becoming hypothesis

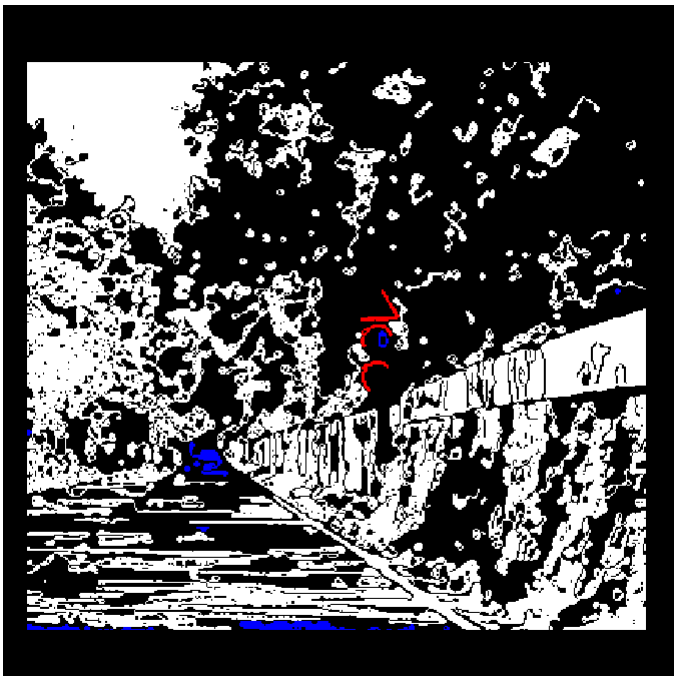

Fig. 1. Detail of masks for region analysis

- just_red bounding boxes with something close to a 2:1 width-height ratio generate also another overlapping hypothesis extending for double the height over or below depending on the centroid of the region

These last two operation are needed to later combine different regions of the same sign in case they are splitted, in our case it happens quite often with 'forbidden' signs. The hypothesis collected in the previous part are then fed in order of reliability (for example a 'red' hypothesis is more reliable than a 'white' one that enlarge the boundary box catching more noise) of their source binary image to the 'reduced classifier'. This function gets a window over the binary images as they were before filtering and performs basically the same operations described in the next subsection with some differences. The output of the reduced classifier is a 'verdict' and eventually a more precise region of interest. The regions of interest obtained are eventually joined if overlapping and returned as output in total_output.

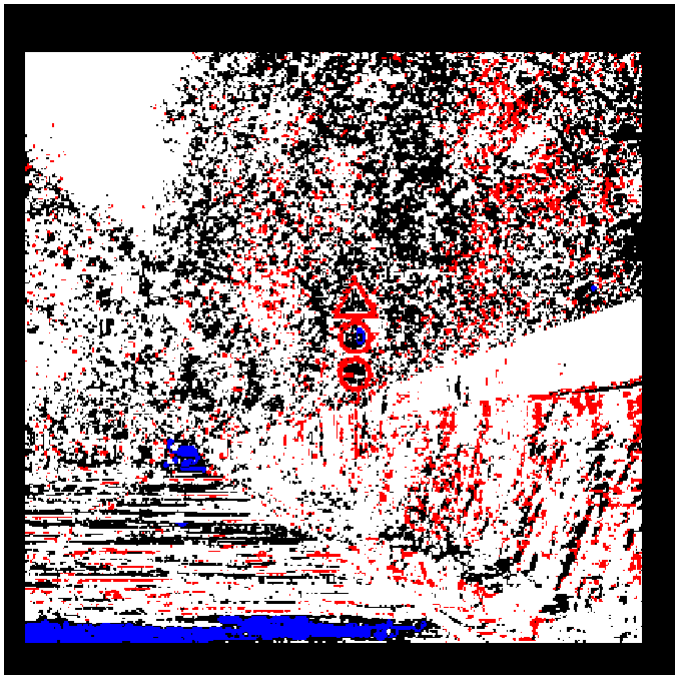

Fig. 2. Detail of masks for reduced classifier windows

B. Classification

We start by adjusting the contrast of the original image and then also, in this case, the script thresholds the image in 4 binary masks

- redMask
- blueMask
- yellowMask
- whitishMask

from the hsv color space. Edges from a gray version of the contrasted image are then computed. At this point the script looks for circles in the edges, redMask and blueMask and then lines in the edges and in redMask. Then masks with the shapes found in road signals are generated and dynamically adjusted using image size and features from the previous step to match the signal. The script then computes a list of scores multiplying some of the color masks with some of the shape masks, then based on those scores and the number of lines and/or circles found, it selects a class for the road signal.

C. Recognition

The last part of the Project deals with the recognition of a road signal. The input image of the program is defined by the figure of the road signal. From task 2 we get previously the type of signal: ('danger', prohibitory, 'mandatory'). The approach consists in feeding previously our software by giving it some known signals. To do that we split the different types of signals into different folders. For example, the "danger" folder is divided in 8 different folders: -trafficlight -bottleneck -leftcurve -rightcurve -crossroad -scurve -snow -exclamationpoint We have to do the same also for the prohibitory and mandatory signals. Then, by feeding these

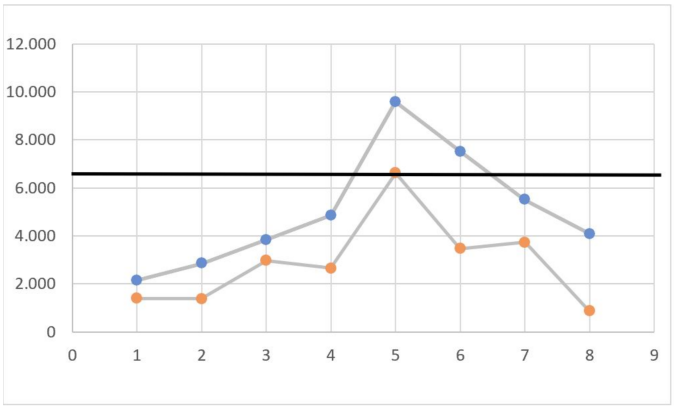

Fig. 3. first step to recognize crossroad signal

folders to the Training function in matlab, it returns two arrays of structs that contains for each type of danger signal (for example) some of the main features to be able to recognize it. We get 2 arrays because one comes from the normal mask with divided regions in the mask and one comes from the convexHull mask. It means that all the regions in the mask are contained in only one region. The main features were found thanks to the use of 'regionprops' matlab function. These features are:

- Area
- MajorAxisLength
- MinorAxisLength
- Eccentricity
- Centroid
- Perimeter

Then, for every one of these features, for the amount of training signals that we have, we define the max and min value for every one of them in order to have some ranges. Thanks to all this data, after getting a new image in the main program, we are able to compare the features of the image with the features that we get from the training function. If the data from two different type of signal are not well defined, we have the support of the convex Hull features and try to find in that array some elements that can distinguish two signals.

Here below, I can make a simple example related on how we worked on this task. We want to recognize if the image get from the main program is a Crossroad signal or not.

The upper line defines the Max area of each type of signal from danger folder. The Lowest line defines the min area of each type of signal again from danger folder. The Numbers are related on:

1) traffic light
2) bottleneck
3) left curve
4) right curve
5) crossroad
6) s curve
7) snow
8) exclamation point

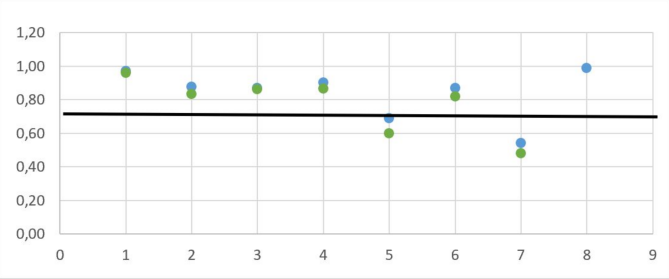

Fig. 4. second step to recognize crossroad signal

Now, taking again the chart, we want to find a region that contain only the crossroad signal or at maximum with only few of the other signals.

To do that, we draw a line that pass through the minimal Area founded from the training data signals. As we can see, it's possible that also an 's curve' signal is contained in that region. We have to analyse another feature of our input signal and try to distinguish it between 'crossroad' signal and 's curve' signal. Of course, we can say that if we find a signal that has an area from the mask higher than 8000, is a 'crossroad' signal. But this is not enough to define the type of signal. We have again the range of area between [min_crossroad_area; 8000]. Because of that, we have to look also for the Eccentricity of the ConvexHull area of the mask:

As we can see we have a perfect division between the Eccentricity given by the crossroad signal and the one given by the s curve signal. Now, we can recognize the crossroad signal just taking into account the Image that, after passing the first test, they have an eccentricity value less the max value of eccentricity of the crossroad signal plus some tolerance.

II. Discussion and analysis of the results

All the code here mentioned is available starting 17/12/2018 on github. [1]

A. Detection

On the set of images provided to test this task, the script is currently able to score PRECISION 9.193548e-01 and RECALL 8.142857e-01. Keeping into account the variety of the set and the pretty bad quality of some images the result can be considered a good one. With more calibration higher precision and/or higher recall can be reached with this approach, but taking into account the limited time available to work on this assignment was difficult to go much further. As of now the main difficulties are related to very poorly lighted road signals, very small or very skew ones.

B. Classification

On the set of images provided to test this task, the script is currently able to score PRECISION 1 and RECALL 1. The result is perfect but it is so very likely just because we are working on a limited set of images. Would be interesting to test it on more images.

C. Recognition

The idea of training the software with known signal in task three works fine. The limitation that we could have is that a bad Pre-processing method of the images gives a bad mask of the signal. A bad mask of the signal means that the 'regionprops' function applied on it doesn't work very well. For example, if the three round regions of the traffic light warning signal are not well defined, we will not find three objects in our variable that defines the number of regions in the mask. This leads to an incorrect recognition of the signal. For example, traffic light can be recognized as an exclamation point. Then, we have to say that, for lack of time, we didn't define the recognition for the prohibitory and mandatory signals but in our program we defined some useful functions that, after getting the Pre-processing image of these categories, are able to get the features also of those ones. After that, we just have to find some ways to divide the signals depending on their features. At the end of the recognition phase we got 43/45 danger signals detected that means a percentage of the 95.5%

III. Conclusions

We are satisfied with the results obtained. Of course, the performance of our project can be improved. The lack of time pushed us to find a compromise between results and details on the project.

References

[1] lucarinelli/assignment-sistemas-baseados-em-visao. https://github.com/lucarinelli/Assignment-Sistemas-Baseados-em-Visao. Accessed December, 2018.