[ENPM673] Project 6

Anton Mitrokhin, Kanishka Ganguly, Cornelia Fermüller

Due Date: May 18, 2019

1 Traffic Sign Recognition - 100Pts

In this project we aim to do Traffic Sign Recognition. You will perform the two steps of detection and recognition. You can use existing OpenCV code (HOG feature detector, MSER feature detector, SVM routines) to create the complete pipeline. The challenge will be in tuning the system to detect well. The approach described here will probably be insufficient to get very good results and full score - you are free to experiment and add your own improvements!

2 Details

Traffic Sign Recognition can be staged into two sections: *Traffic Sign Detection* and *Traffic Sign Classification*. In the Detection stage we aim to extract possible candidates (or regions) which contain a traffic sign (in this part, we do not care what the sign might be). In the Classification stage, we go over each Region of Interest (RoI) extracted previously and try to identify a traffic sign (which might not be in that RoI at all).

2.1 Data

You are given images from a driving car, training and testing images for a set of signs. Your output should be a video submission - (click here to download).

2.1.1 Grading Criteria

For the final pipeline you need to highlight the bounding box of the traffic sign and paste the appropriate sign (from one of the sample training images) beside it. The Traffic Signs to be detected are 45, 21, 38, 35, 17, 1, 14, 19. The grading will consider false positive detections (the areas that are not signs, but the ones you label as signs), false negatives (if you miss the actual sign - it is a very dangerous failure for a car) and misclassifications (when the sign is detected but classified incorrectly). Please see this video (for example): (link).

2.2 Traffic Sign Detection - 50 Pts

Needless to say, there are a lot of ways this can be done. Let's see some of the common pipelines. Traffic Signs can be classified into two categories *RED* (Danger and Prohibitory) and *BLUE* (Mandatory) based on color (this is just one way as the true classes are Danger, Prohibitory and Mandatory - based on shape, color and content of signs).

In this section the objective is to extract the Region of Interest (i.e., around a traffic sign), you can try to do this with simple thresholding in an appropriate color space. Below are two ideas, one is using simple thresholding and the other is using a more robust method to identify regions of *similar* intensity.

YOU CAN USE ANY OTHER APPROACH as you feel comfortable, which might not be given here. In any case, PLEASE SUBMIT A REPORT explaining your approach with relevant outputs after each step.

2.2.1 Thresholding in HSV Color Space

Here the idea is, that any traffic sign will be of a typical color composition; let's say red. The HSV Color Space serves better to identify appropriate bands for H, S, V channels to model the color composition of a traffic sign. Also it isolates intensity/brightness (unlike RGB) which helps with robustness to illumination.

1. Denoise the image - (Noise Removal).
2. Model/Threshold in HSV color space (Ref. [1], [2]) to extract possible blobs for traffic signs.
3. Analyze the properties of each blob (e.g., size, aspect-ratio) to determine if it corre- sponds to a traffic sign.
4. Extract the bounding box. Make sure the bounding box covers the entire sign, if it doesn‘t then build an algorithm to stretch/shrink depending on blob/traffic-sign properties. The cues used for modifying bounding boxes SHOULD NOT BE hardcoded for any particular sign.

2.2.2 Using Maximally Stable Extremal Regions - MSER algorithm

A trivial intuition is that MSER gives regions of similar intensity given a grayscale image (Ref. [4]), and we know that a traffic sign is mostly a uniform intensity region, be it red or blue.

1. Denoise the image - (Noise Removal).
2. You might want to do contrast normalization over each channel (as mentioned in 3A of [4]) - (one approach is described here)
3. Create an appropriate grayscale image that best highlights the sign that you want to detect. Say, for a red sign, if you consider the RGB color space, you can simply use the Red channel as the grayscale image or use a weighted combination of R,G,B channels to create the grayscale image. The same goes for blue signs. **Experiment with other color spaces as well**. The objective is that for example for the red sign the generated grayscale image should be brighter (towards white) in the red region and darker elsewhere.
4. Normalize intensity of the image (Ref. [4])

$$C = \frac{C}{R+G+B} \tag{1}$$

$$C = max(0, \frac{min(R-B, R-G)}{R+G+B}) \tag{2}$$

$$C = max(0, \frac{min(B-R)}{R+G+B}) \tag{3}$$

   Here, 1 is a generic normalization, 2 is particularly useful for red signs and 3 is useful for blue signs.
5. Extract MSER regions from the image. This is the key part where you have to experiment with various parameters of the MSER region extractor. Your aim is to extract all the true sign regions while minimizing false positive regions.
6. Fit a bounding box to the sign, you might have to tweak the bounding box i.e., in case if MSER identifies an inner portion of a particular sign then you might want to scale up the bounding box so that it covers the whole sign. Build an algorithm to SCALE ONLY IF NECESSARY - probably using blob properties (e.g., size, aspect ratio).

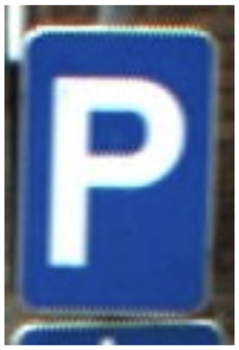
2.3 Traffic Sign Classification - 50 Pts

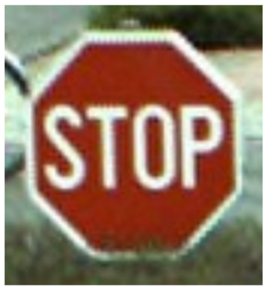
You are given sample images for different signs, you can resize the images to a standard size say (64 × 64) and extract HOG features. Once you get the HOG features, train a multi-class SVM for various signs. You can test the classier performance using test data. This is independent of the Traffic Sign Detection section.

To achieve a complete Traffic Sign Recognition pipeline, you can do one of the following

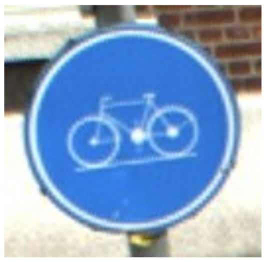
- Find a closest square to the bounding box extracted from the Sign Detection stage, resize it to a standard size (used in Trac Sign Classication say 64 × 64), extract HOG features and predict the sign using the trained SVM model.
- Fix an appropriate square size (depending on the size of bounding box), sweep the square over the bounding box and for each position of the square over bounding box do the previous step. For clear details look up 'PyImageSearch Blog' in 2.6.1

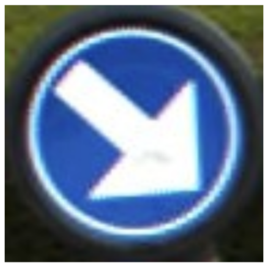
The HOG feature is usually of very high dimension. Though you can use them directly for SVM classification (Ref. [4]), you can also do dimensionality reduction over these feature vectors e.g., using PCA, LDA (Ref. [3]).

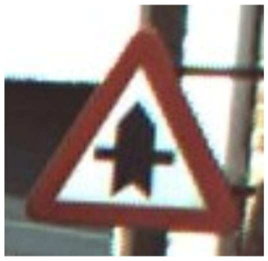

(a) 45

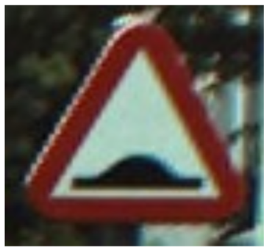


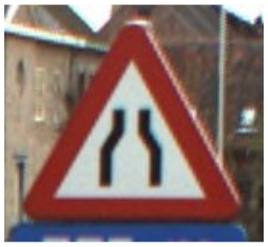
(b) 21


(c) 38


(d) 35


(e) 17


(f) 1


(g) 14

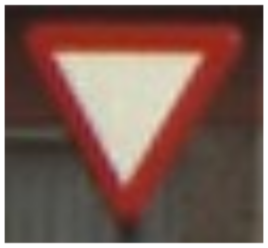

(h) 19

Figure 1: *Different Traffic Signs.*

2.4 Few Other Tricks

In Traffic Sign Detection, the size of the bounding box of the detected sign can reveal information about the possible location of the sign i.e., if the bounding box is quite small then it could be that the sign is really far away and it should be closer to the horizon (or top) in the image. Else if it is bigger, then it should be closer to the center line of the image. You can also observe that the traffic sign might not be appearing in, say, the lower one-third of the image. There could be many such cues from the images that you can exploit to remove false positives in Traffic Sign Detection.

Traffic Sign Classification is relatively straight forward, and you might not need any tricks apart from experimenting with the parameters. In the case that the HOG features are not robust to illumination in the given dataset, you can try the normalization approach from [4].

2.5 Submission Guidelines

Your submission SHOULD be a ZIP folder (no other file extensions) with the naming convention YourDirectoryID proj6.zip on to ELMS/Canvas. Additionally follow the guidelines given below:

1. Have a parent directory P6_Submission.
2. Under P6 Submission you should have two sub-folders: "code" and "output".
3. You should also submit a report (Report.pdf) in the P6 Submission folder.
4. Process **all** dataset frames, and submit them as a video.

2.6 Useful Resources

Traffic Sign Recognition has become widely popular mainly because of the German Traffic Sign Detection Challenge. Also there are many example scripts from Matlab/Python.

2.6.1 MSER Tutorials

- OpenCV code sample for MSER Feature detection - link
- VLFeat Tutorial - link
- MSER Theory Slides - link

2.6.2 HOG - SVM

- OpenCV tutorial for Digit Classification - link
- VLFeat Tutorial - link
- PyImageSearch blog - link

3 Acknowledgement

The Traffic Sign Dataset used in this project is by courtesy of Radu Timofte from ETH-Zurich Vision Lab (Ref. [3]).

4 Collaboration Policy

You are allowed to work in teams of up to 3 and discuss the ideas with any students, but you need give credits in the report, if they are not from your team. If you **DO USE** (try not to, it is not permitted) external code - do cite the source. For other honor code refer to the University of Maryland Honor Pledge. You are responsible for finding your teammates, and the projects can be done alone. Nevertheless, you may contact the instructor or the TAs and we might be able to help you find a teammate.

You should take the effort to search online for any help regarding library function usage, however, any concept related queries can be discussed during Offce Hours.

References

[1] K. Athrey, B. M. Kambalur, and K. K. Kumar. Traffic sign recognition using blob analysis and template matching. In *ICCCT '15*, 2015.

[2] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras. Road-sign detection and recognition based on support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):264–278, June 2007.

[3] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool. Traffic sign recognition — how far are we from the solution? In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Aug 2013.

[4] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. Di Stefano. A traffic sign detection pipeline based on interest region extraction. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Aug 2013.