Project 6 (ENPM673)

Author Name
Aalap Rana (116421393)
Koyal Bhartia (116350990)
Harsh Bharat Kakashaniya (116311236)

Implementation of Traffic Sign Detection and Classification using MSER and SVM Model .

Date : 27/2/2019

Contents

List of Figures

1 Introduction

As we know that the number of accidents are increasing exponential the research on self driving cars and autonomous vehicles is on a boom. The goal of achieving autonomous driving without collision can not be implemented without the proper recognition of traffic signs. Thus,this problem could be solved by application of computer vision techniques.

In computer vision the main focus is on understanding the surrounding. This task of understanding the environment can be broadly classified into two sub-tasks i. e. detection and classification.

As after the standardization of traffic sign it became easy to detect the traffic sign. Now we can consider the shape and color of the traffic sign to differentiate required sign from the pool of traffic signs. The project focuses on detection of 8 signs in the real scenario. Using the properties like sign color and shape we can differentiate one sign from the other. Here we have 2 sets of sign based on color (blue and red) and 4 sets based on shape (circle,triangle,rectangle,hexagon). The implementation of project is done using MSER method and SVM Model.

The process of Traffic Sign Recognition is divided into 2 parts namely: Traffic Sign Detection and Classification as described below.

2 Traffic Sign Detection

The different traffic signs that we detect and classify as a part of the project include:

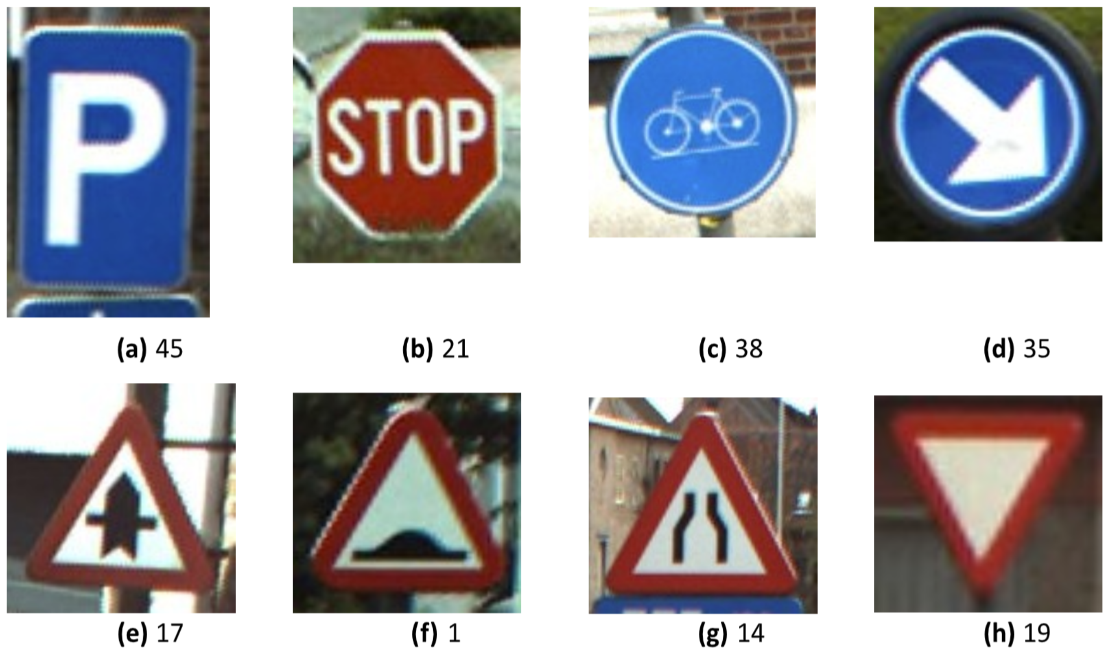

Figure 1: Traffic Signs with labels

2.1 Image Preprocessing

Following is a sample of the input image on which the process of traffic sign detection has been carried on. The different traffic signs that we detect and classify as a part of the project include:

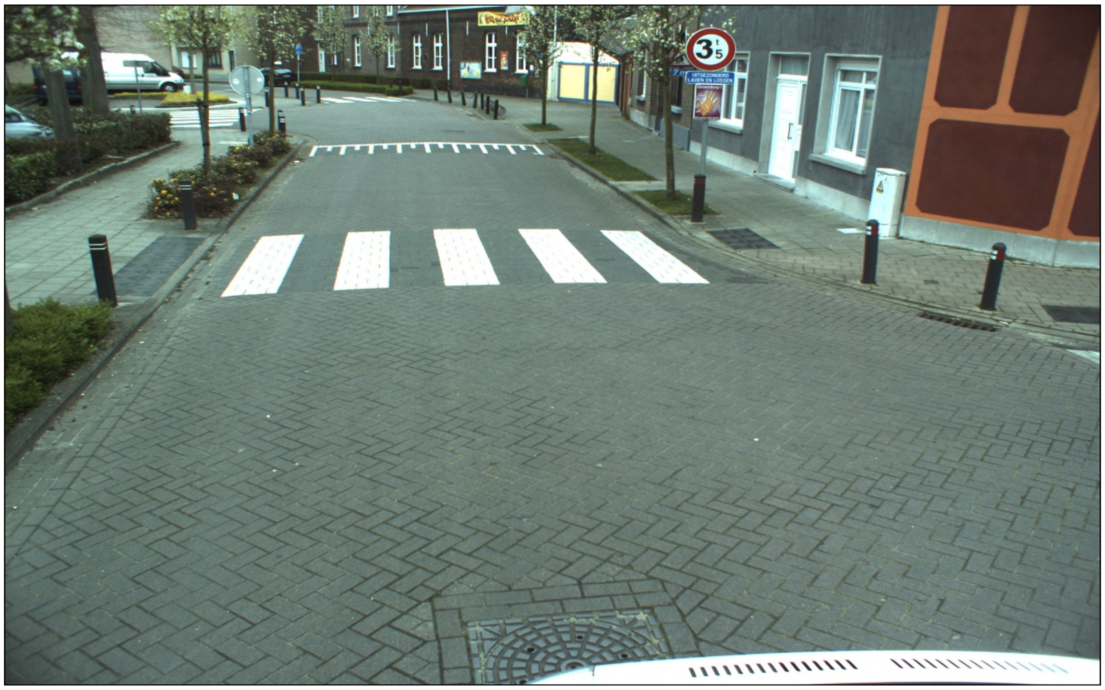
Figure 2: Sample input road image

2.1.1 Signal Recognition/Detection

The whole process of traffic signs detection is achieved by following the below mentioned the steps: 1)Defining the Region Of Interest 2)Contrast Improvement 3)Noise Reduction 4)Color Space Selection Threshold 5)Finding traffic signal

2.1.2 Region Of Interest

On inspection of the given data set we observed that there is no traffic signal occurrences in the bottom part of the image. Thus to improve the process and reduce the computation we defined a region of interest(ROI) which is the upper 60% of the image and making the lower 40% all zeros. By appending zeros in the bottom part we made the image of the original shape. Thus, maintaining the dimensions.

2.1.3 Contrast Improvement

As the data provided is obtained by camera mounted on a moving car there is a lot of regions of varying contrast. Thus,this varying brightness makes the detection difficult. Therefore, we converted our image to LAB color space and separated them into respective channels and applied histogram equalization to the L channel and then recombine the three channels to form the contrast equalized image which can be used for further process.

2.2 Thresholding in HSV Color Space

- **Noise Reduction:** The image is contrast equalized but has noise in it thus this noise is removed by using the in build function cv2.fastNlMeansDenoisingColored(contrast_eq_image,None,10,10,7,21)

- **Color Space Selection Threshold:** By following the above steps we obtained a contrast equalized and noise free image. The next task in the pipeline is to separate the traffic signal from the background. The image is converted to the HSV color space from BGR color space. This was done because in HSV color space is BGR is highly sensitive to illumination and thus the color of the same object changes dramatically with a slight change in illumination. The image obtained is a HSV image and on this we apply upper and lower threshold to get the signal. In case of blue traffic signal the threshold values are [100,50,50]for lower threshold and [110,255,255] for upper threshold. In case of red traffic signal the lower bound is [0,40,30] and the upper bound is [10,255,255]. When this masks are applied to the image we get two binary image which have white pixels corresponding to red and blue color. the result obtained is illustrated in the underlying figures.

2.3 Detection of the Traffic Sign

2.3.1 Finding the sign

The image obtained is a binary image. As we know that the detection of small object is very difficult because they have very few pixels allocated to them in the whole image. Consider the case of traffic sign it can be assumed to be of size 20*20 pixel in the image of 2000*2000 pixel. Thus, we increase the size of image so the process of detection becomes easy. After the image is increased in size we convert it to a gray scale image and provide it to MSER function to detect the regions of same color intensity. The list of all the contours obtained needs to be filtered to get the contour representing traffic sign. This is done by applying two conditions for removing contours first is the area (if less than the set threshold remove that contour) and second the aspect ratio(if aspect ratio is in the range . 9 to 1 it is as sign). The result obtained is shown below.

2.3.2 Maximally Stable Extremal Regions - MSER algorithm

Maximally Stable Extermal Region is a blob detection method employed in computer vision.For implementation of the method the image is converted to gray and passed to the function.The MSER function finds gray level region where the intensity remains constant when subjected to varying thresholds. The algorithm poses the following advantages: It is invariant to affine transformation. Multi-scale detection without any smoothing involved,which can detect both fine and large structures. The list of all MSER regions is provided considering the worst case scenario's.

3 Traffic Sign Classification

The SVM has been trained on the Training images folder that have been provided and tested on the Testing images.

Following is the procedure followed to get the trained SVM model:

3.1 HOG Descriptors

The HOG descriptors needs to be extracted to train The SVM. To get a uniform size descriptor for all the images, we first resize the image to a 64*64 size. The Opencv function used to get the HOG descriptors is: **hog=cv2.HOGDescriptor()** and **hog.compute(image)**. The several hyperparameters which goes into this function needs to be tweaked. After several trails and errors, the parameters that we finally fixed are:

- winSize: (64,64)
- blockSize = (16,16)
- blockStride = (8,8)
- cellSize = (8,8)
- nbins = 9
- derivAperture = 1
- winSigma = -1.
- histogramNormType = 0
- L2HysThreshold = 0.2
- gammaCorrection = 1
- nlevels = 64
- signedGradients = True

The HOG descriptors are stacked up for all the training as well as testing images which is later passed as an input to the SVM method.

3.2 Training the SVM Model

Along with creating the stack of the training images, we also stack up the Labels corresponding to each image descriptor.

This labels matrix is also used to train the SVM. The labels matrix for the testing images is used to calculate the E_{out} or the out of sample error.

Avoiding False Negative: To avoid the case of false negative, we have taken an additional folder where we have other extra images from the input images i.e crops of parts of cars, houses etc. We have used even these images as a part of the training images and label these as 0. The other images are labelled with the one among the 8 signs that we have to detect i.e [1,14,17,19,21,35,38,45]

The training of the SVM model needs the training and testing descriptor set along with the traning image labels as obtained above.

The value of the parameters ν and Gamma have been taken as 0.5 and 2 respectively after several trials. The opencv function used to create the SVM is svm=cv2.ml.SVM_create(). The type and Kernel of the SVM is set to cv2.ml.SVM_NU_SVC and cv2.ml.SVM_RBF respectively.

Once the SVM model is created, svm.trainAuto() is used to get the trained SVM. This trained SVM is used to make the prediction on the testdata.

3.3 Testing of the SVM model

The test image descriptors stored in the first step here along with the Test image descriptors is used for the prediction purpose using **svm.predict()**.

The predicted answer obtained here is compared with the test descriptor labels which is used to get the E_out error for the sake of completeness and checking the correctness of the model.

The E_out we get for our model is 97.38%. This shows that 97% of the testing images were classified correctly.

4 Evaluation of Model and Sign Prediction

The SVM model trained above is saved in an xml format as **svm_model.xml** which can be reused to get real time classification of images. Following is the process that we have followed to get the real time predictions and use the same to attach a copy of the sign detected beside the sign.

4.1 Crop

The detected traffic sign from the first part of the project is cropped out of the input image. This cropped image is resized to 64*64. We then find the HOG descriptors of the resized image.

4.2 Predict

The saved svm_model.xml is loaded. The above descriptor of the cropped image is given to the SVM predictor.

Depending on the prediction value of the SVM, we paste a sample of the sign beside the original image for the purpose of checking.

Following is a sample of the final output that we get:

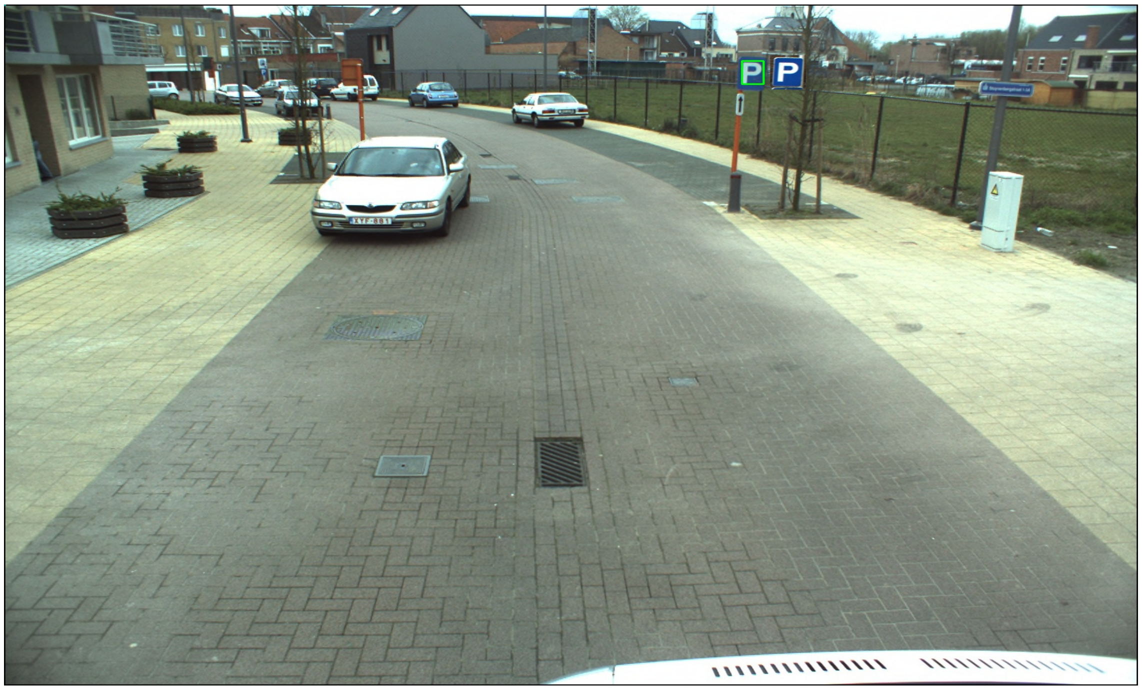
Figure 3: Sample Output

Link with video:
https://drive.google.com/open?id=1O8y0RN8_g2uS3laisUn9x9jFTM96MabK