

# ENPM673

## Traffic Light Recognition

Ashwin Varghese Kuruttukulam(115906518)  
Rachith Prakash (116141468)

OUTPUT



## Method

### Traffic sign Detection

Approach:

We considered both RGB channels and HSV channels for thresholding the image to get blue and red areas. For the RGB thresholding, we used the suggestion in section 3A [1] to get red and blue areas. We then used contour detection to get contours and applied constraints on them to get possible candidates. Then passed this to the classifier. Here we considered only top 700 pixels [row-wise]

for detection as no sign would be below that. Even if it is, it will be very far and won't be useful.

We tried MSER feature detection, but found that a lot of bounding boxes were being detected compared to our previous method. Hence, we stuck to our previous method.

### Denoising

Applied *localNlMeanDenoisingColored* for denoising the image. This particular technique is a good method to make the image smooth. We denoised the image across all channels. We also denoised the image in HSV channels too. In the following images we see the effect of smoothing.





### HSV thresholding

We converted the image to HSV, denoised it and thresholded to obtain red and blue segments. We used the following thresholds for blue:

Lower\_threshold = [90, 90, 0]

Upper\_threshold = [200, 200, 255]

For red, we used two kinds of thresholding, since, the red channel is across the boundaries of Hue channel i.e. 10-350 (0-180 in opencv) as it varies from 0 -360. We used the following thresholds for red channel:

First\_lower\_threshold = [0, 100, 0]

First\_upper\_threshold = [20, 255, 255]

Second\_lower\_threshold = [160, 100, 0]

Second\_upper\_threshold = [180, 255, 255]

Obtained the following results after thresholding using the above channels.

For **red** thresholding



For **blue** thresholding



Output of using section 3A[1] i.e. we obtained the red channel and blue channel separately and performed contrast stretching as they mentioned . Once the contrasted image is obtained , we are thresholding it to get a binary image. This is done to eliminate unwanted detections. The outputs of this step are as follows:

For **red** thresholding of the same image shown above.



For **blue** thresholding of the same image as above.



After combining the above two thresholds, we have the following images which are better than the individual images.



Bounding boxes



Once we have the best candidates w.r.t color segmentation, we draw bounding rectangles around these and then do processing to remove unwanted bounding boxes. The criteria considered for removing unwanted bounding boxes are as follows:

- Area threshold for blue - [500,5000] for red - [400,5000]
- Aspect ratio of the width and height of the bounding rectangle
  - For red - [0.4, 2.5]
  - For blue - [0.3, 3.3]



## Traffic sign Classification

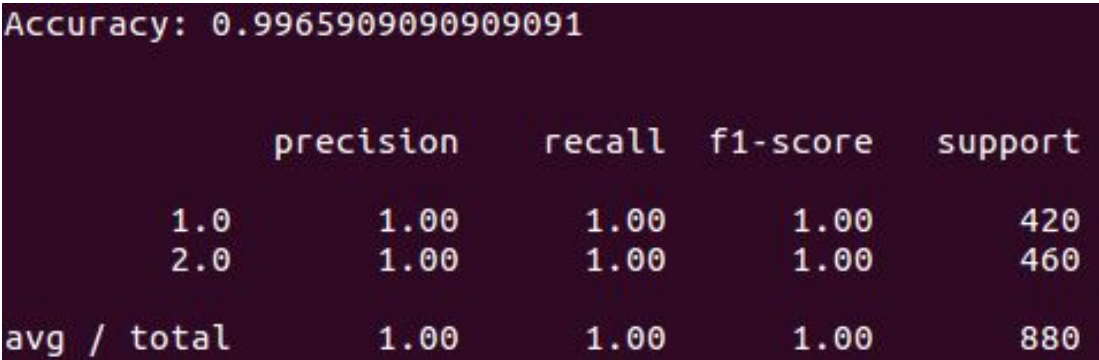
Once we get detections i.e. possible candidates for a sign, it is necessary to classify what sign it is. The bounding boxes obtained from the detection pipeline will not always be of perfect size. Also it is not necessary that the detection is positively a sign. Hence, we also need to take care of the negative cases i.e. detection which are not signs.

We are using svm classifier to further check whether the received cropped window is part of any of the specified classes or whether it is not a traffic sign at all. In order to do this we need negative images. In order to do this we generated over 7000 negative images. The random images were generated by cropping out random images of size 64x64 from random images in the input sequence and

then checking whether this cropped image was a traffic sign or not. In total we extracted about 9k negative images. This is a very crucial step as we do not want any random detection to be classified as a sign.

## 2 Class Classifier

Our first strategy was to use all the given data and the negative image data to classify whether the received image is a traffic sign or not. Although this classifier has a very high accuracy, the number of false positive from the detection were quite high. Therefore, about 1 in 200 would fail so this did not work well for us.



```
Accuracy: 0.9965909090909091
```

	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	420
2.0	1.00	1.00	1.00	460
avg / total	1.00	1.00	1.00	880

## 9 Class Classifier

In order to improve from this, we used only the images from the required labels and a class for negative images. Since this would give more variation for the classifier to leave out negative classes, we found this to work better than the 2 class classifier.

Accuracy: 0.997229916897507				
	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	28
14.0	1.00	1.00	1.00	13
17.0	1.00	1.00	1.00	51
19.0	1.00	0.97	0.99	79
21.0	1.00	1.00	1.00	14
35.0	1.00	1.00	1.00	39
38.0	1.00	1.00	1.00	93
45.0	1.00	1.00	1.00	30
62.0	0.99	1.00	1.00	375
avg / total	1.00	1.00	1.00	722

## 64 Class Classifier

But after this we realized that it would be even better to include the traffic signs that we do not classify. This would allow the classifier to learn from even more training images and would enable it to better classify between the sign and not a sign.

## Result

Video included in the folder.