Traffic Sign Detector

1. Requirement:

To detect and localize traffic signs with yellow + red colors in 25 images. The approximate number of signs in the images is 31.

2. Methodology:

This problem can be solved either by traditional computer vision tools, such as edge detection and image processing, or it can also be solved using deep learning models such as Faster R-CNNs.

Using deep learning for this problem is an overkill and will lead to unneeded complications. Also, to get acceptable results, a big enough dataset should be available to train the model. If we can find such a dataset, we will need to filter out the images with yellow + red signs only, which will most probably shrink the size of the dataset dramatically. We can use the full dataset, but we will have to do image processing at the end to get only signs with yellow + red colors. Not to mention the time needed to train the model in the first place.

That is why I decided to solve the problem with traditional computer vision and image processing techniques.

3. Proposed solution:

Traffic signs have standard shapes, such as triangles, circles, squares, etc. The signs provided in the 25 images are mainly triangles and circles. Thus, we have an important feature to search for in the image, which is the shape. However, shape is not sufficient on its own since we can find triangles and circles in the images that are not signs. Yet, we already know the colors of the signs we are searching for. Hence, my method will be based on color and shape.

3.1 Color

The first step is to filter out the colors that are not needed in the image and keep red and yellow objects only. Since it is hard to differentiate between colors in the RGB color space, I will convert the image to the HSV color space (Hue, Saturation, Value). HSV is designed to imitate how humans perceive colors and it makes it easier to deal with colors. As shown in Figure 1, Hue specifies the angle of the color on the RGB color circle. Saturation controls the amount of color used. A color with 100% saturation will be the purest color possible, while 0% saturation yields grayscale. Value controls the brightness of the color.

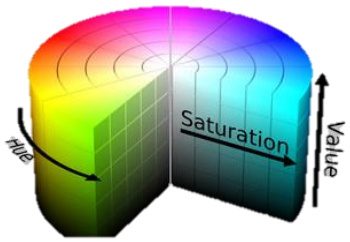

Figure 1- HSV Space

In order to get the yellow and red colors in the image, several experiments have been done to reach a range that provides good results (Table 1).

Color	Hue	Saturation	Value
Red1	$0 < H < 10$	$120 < S < 255$	$50 < V < 255$
Red2	$170 < H < 180$	$70 < S < 255$	$50 < V < 255$
Yellow	$10 < H < 30$	$100 < S < 255$	$90 < V < 255$

Table 1

These values are used to generate a binary mask that will be used to focus on the region where the target sign exist. To generate this mask, I search in the HSV image for the provided ranges in the table. This will give me Red1 mask, Red2 mask, and Yellow mask. Red1 and Red2 are ORed together to get Red mask. Then, I do morphological processing, such as corrosion and expansion and filling for the binary masks to get them ready to use.

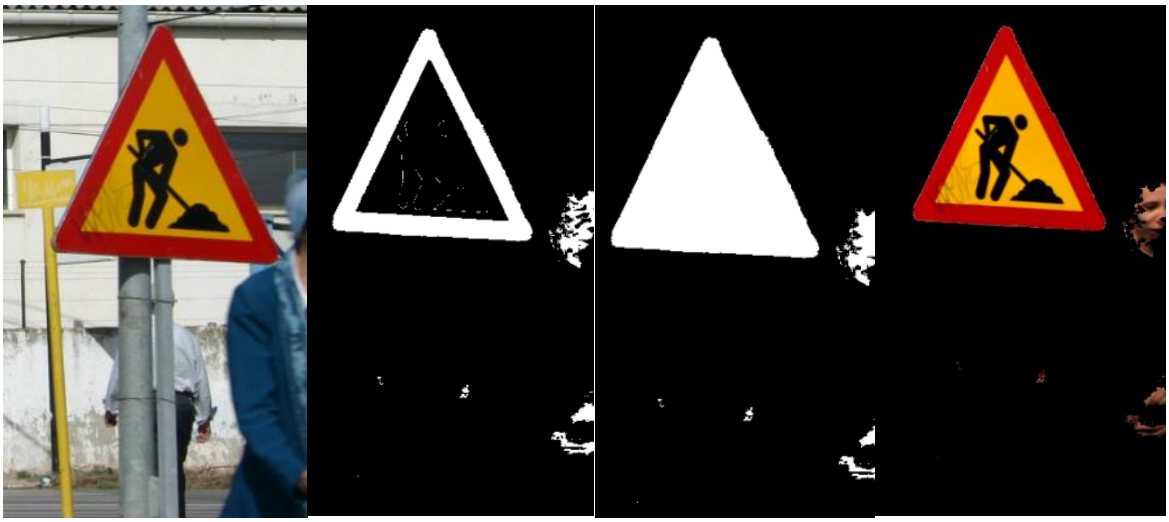

Image *Red mask* *Red mask filled* *Result*

The shown above is the processing of the red mask. In order to get the full mask that includes red and yellow, I perform ANDing between Red mask and Yellow mask. The resulted mask provides only the yellow color that coexisted with a red color. This will help avoid getting other red or yellow objects.

Yellow part of the sign

The next step after generating that mask is to find the contours (outlines) of the objects in the mask. I will then draw a bounding box surrounding everyone of these contours. However, if I just draw the bounding box, it will be smaller than the sign since we are dealing with the yellow part only. To overcome this, I added to the height and width a ratio of themselves, which represents the red part of the sign. The only problem with this method is if the inner yellow is so small like the image below.

The result applying the filter to this sign will be only the yellow dash inside, which is relatively small and will not be able to cover the whole sign.

3.2 Shape

After finding the contours of all the objects in the mask, we will need to filter them out to get rid of objects that are not signs. To do so, I use a similar idea to template matching but without any convolving. I simply cropped two signs from the images, a circular sign and a triangular sign, removed their backgrounds and used them as templates. I loop over the bounding boxes of all the generated contours, crop the part of the original image corresponding to that box, and I reshape the two templates to be of the same size as the cropped part. Then, I calculate the MSE and SSIM (structural similarity percentage) between each of the templates and the cropped part. Now I have two MSE values and 2 SSIM values; I take the minimum value of the MSE and the maximum value of the SSIM and compare them to a threshold to decide if the cropped part is similar enough to any of the two templates to draw a bounding box around.

These are the two templates after removing their background.

Shown above are the two templates I am comparing with (two on the left) and the cropped part of the image (on the right) after rescaling and masking.

```
MSE C 40.070748688148484
MSE T 36.226217977724325
SSIM C 0.5534666518255961
SSIM T 0.6796689750213356
```

Comparison results

Shown above is the result of the MSE and SSIM after comparison. We can notice that the MSE of the triangular template is less than the MSE of the circular template while the SSIM is larger. That makes sense because the cropped part is a triangle.

After checking similarity, I draw the bounding box if needed and this will give the result shown below. Also, a table summarizing the results can be shown below.

4. Results

Sample result

Image ID	Number of signs	Detected signs	Missed signs	Extra false detections	SSIM	MSE
1	1	1	0	0	0.72	36.14
2	1	1	0	0	0.63	52.54
3	3	1	2	0	0.53	43.5
4	2	2	0	0	0.68	40
5	1	1	0	0	0.54	53.8
6	1	1	0	0	0.56	53.5
7	2	1	1	0	0.6	42
8	2	1	1	0	0.61	42
9	2	1	1	0	0.7	39.3
10	1	0	1	0	-	-
11	1	1	0	0	0.65	46
12	1	1	0	0	0.64	48
13	1	1	0	0	0.54	34
14	1	1	0	0	0.63	40
15	2	1	1	0	0.63	35
16	2	2	0	0	0.51	37
17	2	1	1	0	0.6	53
18	1	1	0	0	0.49	43
19	2	1	1	0	0.67	38
20	1	0	1	0	-	-
21	1	1	0	0	0.62	45
22	4	2	2	0	0.42	53.8
23	1	1	0	0	0.52	52
24	1	1	0	0	0.66	36.4
25	1	1	0	0	0.57	37.5
Total	**38**	**26**	**12**	**0**		

Table 2

NOTE: For images with more than one sign, I reported the minimum SSIM and the maximum MSE to help understand the thresholding.