Traffic Sign Detection and Classification

Diogo Samuel Fernandes
up201806250@up.pt
Paulo Ribeiro
up201806505@up.pt
Pedro Ferreira
up201806506@up.pt

Faculdade de Engenharia da
Universidade do Porto,
Porto, Portugal

Abstract

This work focuses on the detection of colour and shapes regarding traffic signs under many different circumstances and combinations of illumination, angle, and contrast. After the foundation was laid, we began to handle these changes ourselves, which presented numerous obstacles. To evaluate the results, a dataset collected from a Kaggle competition containing many different images from many traffic signs was used.

The results were satisfactory when distinguishing between the different types of traffic signs. However, in the future, more work might be done to help enhance the detection of some traffic signs in more intricate scenarios where the current technique fails.

1 Colour Detection

For colour detection, we need to find, in the provided images, blue and red traffic signs. For that, colour masks are applied to the selected image with a dynamic colour range, based on the properties of the chosen image.

1.1 Light and Saturation

Because not all photos have the same colour intensity and brightness might differ from one image to the next, a simple colour threshold would only operate in particular circumstances.

This happens due to some light issues that might influence the colour of the signs. Furthermore, since the majority of them are outside and exposed to the elements of nature, some images may depict fading traffic signs that have had too much exposure to the sun. As a result, some red signs may take on a reddish or orange tone, while blue ones may take on lighter shades of blue.

So we could cope with this situation better, we considered saturation when gathering the red and blue parts of the images. So, before thresholding the images to their colours, the original image is converted from RGB to HSV, and the saturation mean is calculated using the S parameter from the same. This operation splits photos into bright and dark and then creates thresholds depending on the resulting value.

This ensures better results when calculating the red and blue values from the image, as areas with undesirable colours are removed.

To better illustrate this, both traffic signs in the figure 1 and figure 2 are identified, despite the fact that one has a high saturation level and the other has a low saturation level.

1.2 Background

Another problem faced when trying to gather the desired colours is the background. Some images either have reddish buildings behind or simply because the sky is the background and consequently all blue shaded (e.g., images 3 and 4).

A morphologic treatment is conducted to the mask to try to distinguish traffic signs from the background and remove any difficult-to-separate continuing boundaries. In this scenario, the most efficient procedure is the morphologic open, which involves a dilation after an erosion. As a result, and because many signs have a very little white border, which can be incomplete if the image quality isn't high enough, the continuity between items is broken.

1.3 Masking and Final Solution

Once these colours are defined, the procedure is relatively straightforward: create different masks and apply them to the original image.

Since the red colour actually has two different zones of occurrence in the HSV spectre, it is necessary to apply two separate thresholds to the original image before combining the results.

It's worth noting that the entire procedure occurs twice: once for the red regions of the image and again for the blue (e.g., images 5 and 6). By doing this, the shape detection step is facilitated in the sense that red objects will not interfere in the detection of blue shapes and vice-versa.

2 Shape Detection

The previous step results in two images, one containing the blue parts of the original image and the other containing the red parts. The next step is to detect the shapes contained in each of these images.

Our first idea consisted of basing our main strategy on the *findContours* function, which, as the name implies, finds the contours (curves joining all the continuous points along the boundary having the same colour or intensity) in an image. The method is executed for each of the colour images (red and blue) and starts by converting the image to grayscale and eliminating some noise using a **Gaussian** filter with a kernel size of 7. Then, we find the contours present in the image and go through each one of them. If the contour has an area smaller than 2000 pixels, we ignore it and proceed to the processing of the next.

Now we have to classify the contour shape, which will be explained in the next sections concerning existing shapes. In case the contour is not declared as unidentified, which happens when it does not fit into any of the existing categories, we store this contour in an array that will be returned later as the set of contours of the traffic signs that were found. Now we will specifically explain each of the shapes that can be detected.

2.1 Existing Shapes

For each contour, we calculate its perimeter using the *arcLength* function and perform an approximation of the shape of the contour using the function *approxPolyDP*, with an epsilon (maximum distance between the approximation of a shape contour of the input polygon and the original input polygon) equal to the perimeter of the contour multiplied by 0.01 for the red contours and by 0.02 for the blue contours. After that, we just need to check the number of sides of the polygon.

2.1.1 STOP Signs

For classifying a contour as a STOP sign, we only needed a simple check verifying that the number of sides of the approximate polygon is 8 (STOP signs are octagonal) and that the contour colour is red. The image 7 shows the detection of a STOP sign.

2.1.2 Rectangles/Squares

As for the blue signs, we first obtained the dimensions of the quadrilateral surrounding the contour and calculated the aspect ratio by dividing the width by the height. Next, we consider that it is a square if the aspect ratio has a value between 0.90 and 1.10, or that it is a rectangle otherwise. Image 8 and 9 show a blue square and a blue rectangle being detected, respectively.

2.1.3 Circles

Our first approach to detecting circles was to simply consider as a circle all polygons that have a high number of sides (e.g., greater than 10). However, this will lead to erroneous detection of signals in the images, as very irregular polygons, far from circles, were detected. That said, we chose

to perform the detection of circles using the *HoughCircles* method, which uses the **Hough Transform** in `HOUGH_GRADIENT_ALT` mode with an inverse ratio (*dp*) equal to 1, a minimum distance of 30 pixels to avoid overlapping circles, and with *param1* equal at 250 and *param2* at 0.75, the latter representing the measure of perfection of the detected circles.

After that, we still need to do a little processing, by going through all the circles and eliminating the irrelevant ones, which were internal to other circles. This happened, for example, in the signs of the "red circles" class, whose red border led to the detection of two distinct circles: those delimited by the inner or outer part of the edge (e.g., no vehicles sign). This verification consisted of checking whether the centres were the same and, if so, keeping only the outer circle, which contains the sign in its entirety.

The images 10 and 11 illustrate the previous process.

2.1.4 Triangles

One of the extra improvements we made was triangle detection. Although it seems something simple because intuition says that it would only be considered as triangles all polygons with three sides, the fact that the triangular signs present curved corners resulted in a greater number of sides. This could not be resolved by increasing the polygon approximation factor as this would affect the detection of other types of traffic signs.

Therefore, our solution to this problem consisted of first checking if the detected polygon had six or seven sides (most triangles resulted in these values), and then checking if, of the detected sides, three of them had dimensions much larger than another three sides (we set a sufficient factor ratio of 1/4). The final result ends with the detection of a triangle (e.g., 12).

3 Image Matching

To improve the results further, we also decided to use feature matching for the images where some traffic signs are not found. Since matching all the possible traffic signs was almost infeasible and very costly, we decided to use this strategy for the three most common ones.

This way, if, in the final image, no STOP, zebra-crossing, or bump sign is identified, a new matching technique is employed to ensure that these signs are not in the picture.

3.1 Feature detection

Before applying the algorithm, we used a **Gaussian** smoothing technique both in the example sign image and the test image.

Because the bulk of the images contains certain distortions and diverse view perspectives, we chose **SIFT** feature detection for our matching approach. **SIFT** also allows us to match the signs regardless of the size and transformations it suffered.

3.2 Matching algorithm

Jointly with **SIFT**, we opted to use the **FLANN** algorithm to do the actual matching. Because images typically only have one sign, this strategy was preferred over the brute force approach since more than one matching operation will be performed. This operation works very well, even when the background and luminosity influence the image, for example, when the sun is behind the sign (13).

The zebra-crossing sign, on the other hand, may vary in direction, i.e., the man in the picture may be walking to the right or left. Two independent matching operations must take place to solve this problem, one with the man walking left and the other with him heading right. The match with the most found feature points is then used (e.g., images 14 and 15).

3.3 Eliminate false matches

We employ two verification strategies to cope with inaccurate key point matching in the image. **Lowe's ratio** algorithm is the first, and the **RANSAC** algorithm is the second. Doing so enhances our matching results and allows us to use the matching strategy more efficiently.

Finally, instead of the green contour utilised for our main strategy, a blue square emerges around the sign to distinguish the technique used to identify it. Image 16 shows a sign detected by our main strategy and another one by matching.

4 Post Processing

After obtaining the shapes present in the image, we still need to process and solve some problems related to their identification.

The more serious issue arises when shapes are detected inside other objects, such as a circle detected in the "O" letter of a STOP sign. Since it is a red circle, this detection is valid, but is also unwanted.

To tackle these issues, a set of requirements was devised in which not all of the detected shapes would make the final cut, but only those that met the following criteria:

- If the form is a circle, it will be valid only if none of the pre-processed circles with the same center are two-thirds larger;
- If the shape is red and inside a stop sign, it will not be taken into account because no red shapes should be identified inside a STOP sign;
- If the form has already been processed, but a new STOP sign shape is being processed that involves the prior one, the first one should be rejected if it is red.

The images 17 and 18 show the before and after states of the filters being applied during the post processing phase.

5 Efficacy

During the development of the project, we faced different problems, of which we highlight the most relevant in the following sub-sections.

5.1 Light and saturation problems

As previously said, several photos had objects with different tones of the same colour, particularly the most relevant colours to our work, red and blue. To try and attenuate this problem, thresholds were created when dealing with this set of colours. These thresholds took into account the photos' average brightness and saturation, allowing for a clearer separation from non-relevant objects. With these dynamic thresholds the results improved significantly, allowing a better detection in the following phases. As previously demonstrated, the images 1 and 2 illustrate two photos, one with low and other with high saturation.

5.2 Objects detected inside others

As already mentioned in previous sections, objects were sometimes detected inside traffic signs (e.g., a red circle in the letter "O" of the STOP sign). To solve this, we loop through all detected contours and filtered those that were internal to others already processed. This greatly reduces the amount of objects detected, showing only those meaningful to us - traffic signs. As shown before, this optimization process can be be seen in the figures 17 and 18.

5.3 Different sign perspectives

Like formerly stated, some photos might have traffic signs that are harder to detect due to having different perspectives and distortions in the image. We approached this problem by using a matching algorithm. To accomplish so, we chose the **SIFT** algorithm for feature detection because it is good at dealing with distortions and perspective problems, and we paired it with the **FLANN** method, using the **RANSAC** and **Lowe's ratio** algorithms for false positive detection. Our performance increased substantially as a result of using this matching strategy, and we were able to identify more objects, for example the images 19 and 20, that were not detected using the previous method.

6 Status and Degree of fulfillment

In short, we were able to successfully complete the main objectives of the project by detecting traffic signs and classifying them into three base classes: STOP sign, red circles and blue squares/rectangles. In addition, we also implemented the suggested improvements, namely:

- The program can easily handle more than one traffic sign per image, classifying each of them correctly (e.g., **9**, **12**, **21**, and **26**). A more complicated case would consist of two signals being superimposed, something that our main strategy does not detect (since only one contour would be assigned to the two signals). However, the use of matching corrects this problem, as can be seen in figure **16**.
- The program also detects other classes of signals, namely triangular signals and blue circles, in addition to distinguishing between squares and rectangles. These additional classes can be seen in the figures **21** and **22**.
- The program also handles poorly illuminated signs, due to the dynamic thresholds we apply based on the average saturation and hue of the image. However, it doesn't work perfectly for very extreme cases, since, to work in these cases, it would be necessary to change some values, which would affect the performance in most images. For example, traffic signs are detected in the low quality figures **13** and **23**.
- The program can also handle slightly skewed signals detected by matching (e.g., **13**, **24** and **25**). However, extreme cases of tilt are not detected.

In conclusion, we are satisfied with the results of this first project, which allows the correct detection and classification of most signals in the various images. We are also curious to discover the advantages that Deep Learning strategies will bring during the second project.

7 Performance

To evaluate the performance of our program, we decided to use the various images from the Kaggle Competition Road Sign Detection [1]. We chose to use a subset of this repository, composed of the first 158 images (road52.png → road209.png), which were the ones we considered most relevant for this evaluation.

We went through each one of them, manually counting the number of signs that existed for each class (total), the well-classified ones (true positives) and the incorrect classifications (false positives).

7.1 STOP signs

We can recognise practically every STOP sign using image matching and colour/shape detection, leaving only one or two behind.

True Positives	44
False Positives	4
Total Number	50
Accuracy (%)	88

The detection accuracy of these signs is high, and the number of false positives is small, which corresponds, for example, to cases of red walls whose resulting polygon also had eight sides.

7.2 Red and Blue circles

The results for matching circle signs are excellent, with a proportion of 88.1% matches properly detected as we can see by the following table:

True Positives	52
False Positives	32
Total Number	59
Accuracy (%)	88

Despite this, a large number of circles are incorrectly recognized. This is due to some minor detections in some photos, which are caused by little spots with red or blue colours. It's worth noting that both red and blue circles were considered in these calculations.

7.3 Blue squares/rectangles

This class of signs also obtained great accuracy, in most cases, due to the matching strategy, where we use the two zebra-crossing templates, with the man in the picture walking to the right or left.

True Positives	54
False Positives	6
Total Number	61
Accuracy (%)	89

In addition, the number of false positives is small, so we are satisfied with this performance.

7.4 Triangles

The accuracy figures were extremely pleasing in terms of performance. However, as compared to the other traffic signs, we found that spotting triangles was a little more challenging. The following are the results we obtained:

True Positives	2
False Positives	4
Total Number	6
Accuracy (%)	33

Only 20% of the actual triangles are spotted in the test photographs, as shown above. The difficulty of the used method in spotting the corners of the triangles, which are rounded, accounts for the low percentage. As a result, it recognises them as a new edge, making their shape too inconstant and hard to detect.

References

[1] Kaggle Road Sign Detection Competition. Last Accessed April 2022. https://www.kaggle.com/datasets/andrewmvd/road-sign-detection.

A Results

Figure 1: Low Saturation Image

Figure 2: High Saturation Image

Figure 3: STOP sign with a red wall as background

Figure 4: Blue rectangle with the blue sky as background

Figure 5: Red mask of an image with a circular red sign

Figure 6: Blue mask of an image with a zebra-crossing sign

Figure 7: Detection of a STOP sign

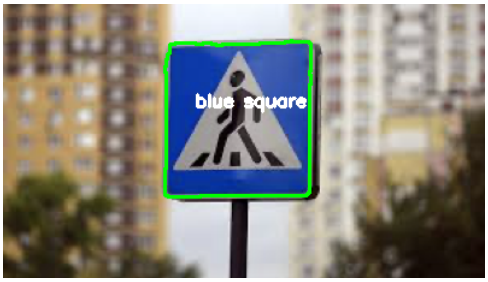

Figure 8: Image containing a blue square sign

Figure 9: Image containing a blue rectangle sign

Figure 10: Image with a red circular sign

Figure 11: Another image with a red circular sign

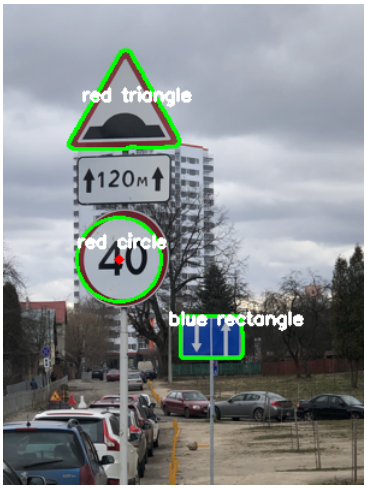

Figure 12: Image with a red triangular sign

Figure 13: Image with a STOP sign detected by matching

Figure 14: Image with a zebra-crossing sign, detected by matching, where the pedestrian is walking to the left

Figure 15: Image with a zebra-crossing sign, detected by matching, where the pedestrian is walking to the right

Figure 16: Image with a sign detected by colour and shape detection superimposed by another sign detected by matching

Figure 17: Program results before the filters applied in the post processing phase

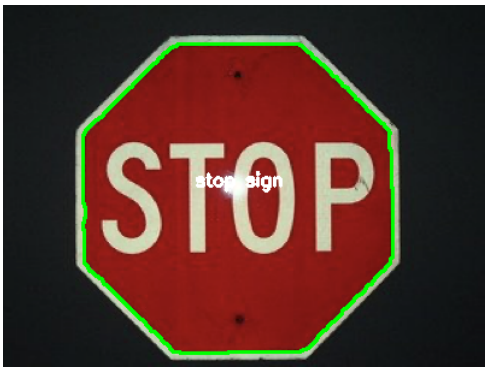

Figure 18: Program results after the filters applied in the post processing phase

Figure 19: Slanted and poorly illuminated STOP sign detected by matching

Figure 20: Slanted blue squared sign detected by matching

Figure 21: Image with a triangular red sign

Figure 22: Image with a blue circular sign

Figure 23: Poorly illuminated image where a STOP sign is detected

Figure 24: Image with a skewed STOP sign

Figure 25: Image with a skewed blue squared sign

Figure 26: Image with multiple traffic signs present