Freie Universität Berlin

Bachelor Thesis
at the Department of Mathematics and Computer Science

Traffic Sign Detection for Model Cars using the Histogram of Oriented Gradients and Support Vector Machines

Rima Badran
student ID: 4873833
rima.badran@fu-berlin.de

advisor:	Prof. Dr. Daniel Göhring
first examiner:	Prof. Dr. Daniel Göhring
second examiner:	Prof. Dr. Dr.(h.c.) habil. Raúl Rojas

February 20, 2020

Abstract

Autonomous driving has become very popular over the years. To successfully develop a car that drives by itself and more importantly follows the traffic laws, many necessary factors need to be considered. One important factor is the detection of traffic signs in order to be able to react properly. Thus, one component of self-driving cars are traffic sign detection systems.
Such a detection system has been developed for the model cars of the research group autonomous cars using support vector machines and the histogram of oriented gradients. For training and evaluating the system, a new dataset consisting of images, made in the robotic laboratory, has been created. In addition to the dataset, recorded rosbag files of a driving model car have been used for the evaluation.
The developed system is able to classify six traffic signs and reaches a high precision. The recall, however, needs to be improved. In order to achieve an improvement, a few adaptations have been suggested at the end.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 20.02.2020

Rima Badran

Contents

List of Figures **iv**

List of Tables **v**

1 Introduction **1**

2 State of the Art **2**

3 Background **4**
3.1 Traffic signs 4
3.2 Dataset 5
3.3 Support Vector Machine 5
3.3.1 Linear separable data 6
3.3.2 Non-linear separable data 7

4 Proposed System **10**
4.1 Overview 10
4.2 Image preprocessing 11
4.2.1 RGB to HSV 12
4.2.2 Color segmentation 13
4.2.3 Filtering regions of interest 15
4.3 Feature extraction 16
4.3.1 Histogram of Oriented Gradients 16
4.4 Classification 18

5 Experiments **20**
5.1 Experiment 1: Dataset 20
5.2 Experiment 2: Driving model car 22

6 Conclusion and Outlook **24**

Bibliography **26**

List of Figures

3.1 Overview of the selected traffic signs 4
3.2 Optimal hyperplane separating the data 6
3.3 Separation of data with two outliers 8
3.4 Non-linear separable data 8

4.1 Overview of the proposed system 11
4.2 Representation of the RGB and HSV color models [1] 12
4.3 Color segmentation of a stop sign image 15
4.4 Background removal 15
4.5 HOG of a speed limit sign 17

5.1 Detected traffic signs while driving 23

List of Tables

5.1 Exp. 1: precision and recall 21
5.2 Confusion matrix 21
5.3 Average processing time 22
5.4 Overview of the detected traffic signs in rosbag files 22
5.5 Exp. 2: precision and recall 22

1 Introduction

Traffic signs are an essential part of traffic. They are used to set rules but also to warn and inform the drivers. Seeing, recognizing and interpreting those signs is crucial for a controlled and efficient traffic. Normally, this is part of the driver's job. But in case of autonomous driving, the car must replace the driver and deal with all the tasks the driver originally performed. Only then, the participation in traffic is possible. Therefore, one of the tasks these cars must be able to perform by themselves is detecting traffic signs. To do so, traffic sign detection systems are needed. Those systems have to work reliably, detecting and classifying signs without missing or misclassifying them. Otherwise, it may result in an accident.
This thesis presents a traffic sign detection system for the model cars of the research group autonomous cars of the Freie Universität Berlin. The system uses vision-based methods to localize the traffic signs and extract their features. Therefore, two color models are being compared in order to find the best suited one for the color segmentation. The histogram of oriented gradients and support vector machines are being used for feature extraction and classification. To evaluate the system, two experiments have been performed. At the end, the results are being interpreted and a brief overview of possible improvements is presented.

2 State of the Art

Many scientific papers of implementing a fast and accurate traffic sign detection system have been published through the years [2]. Each of them aimed for higher precision and recall and a faster computation time. To reach that, different approaches have been developed. Overall, the main structure is almost the same in the approaches. The systems are divided into three steps: localization, feature extraction and classification [2][3]. In order to locate traffic signs, regions of interest are being searched in an input image and extracted. Since traffic signs have a significant color and shape, this can be done either by applying color-based or shape-based methods [1][2]. Mainly, color segmentation is used because it is less complex and time-consuming than shape-based methods. Unfortunately, this type of method is prone to factors like weather and changing lighting [2]. To minimize the impact of these factors, the authors in [1] apply color segmentation on a previous to HSV converted image to get the desired locations. A few authors use the shape to localize a region of interest due to the disadvantages of color-based methods. Hough-transformation is a popular method for this purpose. Moutarde et al. use it in [4] to detect the round shape of European speed limit signs.
After detecting those regions, significant features are extracted which will serve as input for the classifiers. If the shape has not been already determined in the localization step, it will mostly be done afterwards to retrieve crucial features. Mainly, the edges and corners are being detected in the regions of interest, which will give a clue about the geometrical shape of the object. This information forms the input vectors for the classifiers. Despite the Hough-transformation, a popular method to detect circles is the fast radial symmetry detection method, short FRS. It is being used in [5] by Paulo and Correia. Also, they are using the Harris corner detection algorithm in combination with checking, in which area the detected corners are, to determine the shapes. Chen et al. chose the Speeded Up Robust Features, short SURF, to extract relevant features [6]. This method compares the found features with the ones of a template [6]. Maldonado-Bascón et al. use a different kind of method in [7] to determine the shape. They measure the distance from each side of the object to the borders of the bounding box and form a feature vector with those values which will be the input for a linear support vector machine. Since there exist many feature extraction method, Ellahyani et al. compare different features and classifiers to find the most accurate system in [8]. The features included are the HOG, Gabor, LBP and LSS features. They conclude that using a combination of the HOG and LSS features in combination with a random forest as a classifier gives the best results with an accuracy of over 96% [8].
A classifier that is gaining more popularity is the neural network. Such a classifier, more precisely a multilayer perceptron neural network, has been used by Moutarde et al. reach-

ing over 90% accuracy in detecting speed limit signs [4]. Yolo is another neural network presented by Redmon et al. in [9]. This classifier is one of the first ones to be able to process images in real-time which is a desired factor in traffic sign detection. The idea behind that system is not to divide the architecture into two or three steps but to do everything in one step. Tabernik and Skočaj present a system able to detect a much larger number of traffic signs than most other proposed approaches are capable of [10]. They use the convolutional neural network Mask R-CNN consisting of a region proposal network and the region based network Fast R-CNN to achieve this [10].

Nevertheless, other supervised classifiers are still being considered for such a detection system since neural networks require high computational power and a large dataset. Greenhalgh and Mirmehdiuse use linear support vector machines and the histogram of oriented gradients in [11]. They reach over 80% in recall and precision with their real-time approach. Also, Maldonado-Bascón et al. chose support vector machines as classifiers [7]. Firstly, they classify the shape with a linear support vector machine. Then, they use a different support vector machine with a Gaussian kernel for each color and shape combination. In [12] a multi-class support vector machine classifies traffic signs based on their pictogram. The system reaches over 98% in the GTSRB competition.

3 Background

This chapter provides background information about German traffic signs in general, the signs that have been chosen and about the acquisition and preparation of the dataset used in this thesis. Also, the support vector machine will be addressed to get a basic understanding regarding the classification part.

3.1 Traffic signs

Many countries rely on traffic signs when it comes to regulating the traffic. Thus, distinguishing them easily from other objects is essential. For this purpose, the German traffic signs exist in a range of geometrical shapes from circles through triangles and squares and up to octagons [1]. Also, the signs appear in only striking colors, like red, blue and yellow, to distinguish them from their surroundings [1]. In addition to the color and shape, pictograms, mainly in black and white, are used to catch the attention and to contribute to the meaning of a sign [13]. The traffic signs are divided into six groups corresponding to their meaning: danger signs, regulatory signs, aiming circle, traffic requirements, other signs and supplementary road signs [14].

To minimize the complexity that the development of a detection system for all German traffic signs would imply, the presented system will only detect six of the over 500 official German traffic signs. The signs have been chosen based on the importance of their meaning. Some of the signs, like the sign for slip hazard, are less relevant for an indoor environment than other signs. Referring to the importance, the stop sign, three speed limit signs, the priority road sign and the yield sign have been chosen. They can be viewed in Fig. 3.1. All but the priority road sign, which belongs to the aiming circle, are part of the regulatory signs. These signs feature all mentioned shapes and the color combinations red and white, red, white and black and yellow and white.

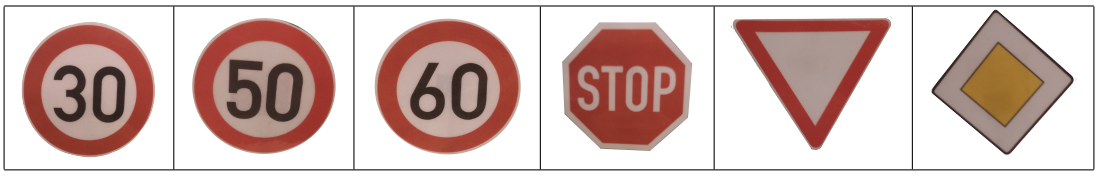


Figure 3.1: Overview of the selected traffic signs

3.2 Dataset

Data acquisition and preparation is key when developing a detection system. A few traffic sign datasets exist online that have been used in the most approaches, like the German traffic sign datasets from the INI Benchmark site [2]. These datasets contain mainly images taken in traffic. Since the detection system is being developed for model cars, the factors that need to be considered differ partially from the ones outside. There is no weather changing the color or the shape of the signs, no extreme lighting conditions [2]. Furthermore, the signs outside are sometimes not completely visible due to the leaves of a tree, other vehicles or because traffic signs are overlapping each other [15]. These factors should have been considered while developing the system when using one of the online datasets. Thus, a new dataset has been generated in the robotic laboratory to adapt to the environment. In addition to a smartphone's camera with 24MP, rosbag has been used to record images while moving the signs in front of a model car's camera. This made it easier to collect various images of each sign. After that, the images of the rosbag files have been extracted and saved as JPEG in order to have a uniform format. The dataset includes only colored images showing the traffic signs in different lighting conditions, in a variety of angles and with varying distance to the camera. After collecting enough images, unusable images have been removed. This includes images showing exactly the same and images where the signs are too blurry and therefore not recognizable. The remaining images of the smartphone have been cropped to 3968x2976 to meet the aspect ratio of $1.\overline{3} : 1$, which is the aspect ratio of images taken with the model car's camera. That way, these images will not become distorted when resizing them later. Depending on whether they are taken with the camera of a model car or with the smartphone, the sizes vary between 640x480 and 3968x2976. Furthermore, the data used for training have been cropped to a size where only the traffic sign is visible.

To be able to easily label the traffic signs later in the classification phase, the images of each sign were moved to separate folders. The folders have been named by the traffic sign they contain. Thus, labeling the images individually is not necessary. The resulting dataset contains approximately 3530 images with 450 to 650 images per traffic sign and 160 images without any traffic sign.

3.3 Support Vector Machine

The support vector machine, a supervised classification method, has firstly been introduced by Vapnik and Cortes [16]. In contrary to unsupervised classification methods, supervised methods need labeled data to train the classifiers. The training data are examples of the classes the classifier is supposed to learn to differentiate. That is why, the images of the dataset have been moved to different folders (see chapter 3.2).

The support vector machine has originally been developed for binary classification of linear separable data [16]. Linear separability is not always given which is why a version for non-linear separable data exists [17].
At first the support vector machine for linear separable data will be explained and after

that the adaptations, made for non-linear separable data.

3.3.1 Linear separable data

No matter what kind of data is given, the main goal remains the same. The goal is to separate the training data by finding a hyperplane where the space between the hyperplane and the nearest data points on each side of the hyperplane is maximized [13]. This space is referred to as the margin. Fig. 3.2 shows such an optimal hyperplane separating two classes with an optimal margin. The data points in the squares are the support vectors forming the border lines which are parallel to the hyperplane. Their meaning will later be further explained.

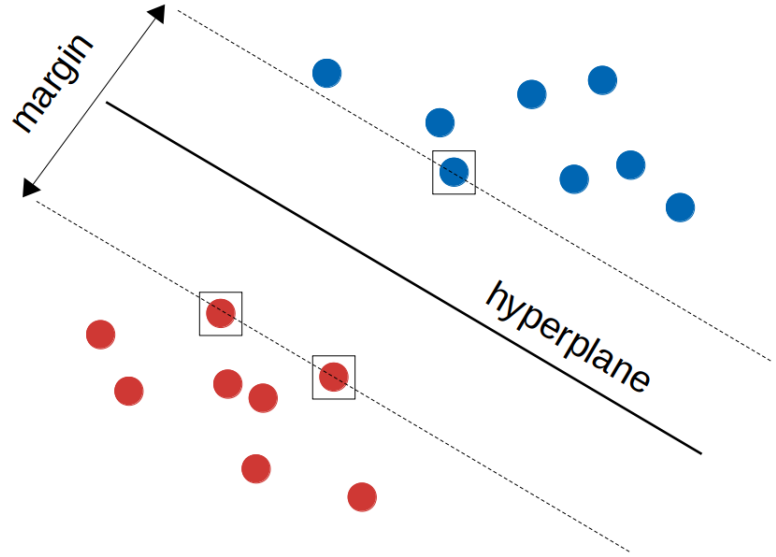


Figure 3.2: Optimal hyperplane separating the data

The two classes of the training data are labeled in the following way

$$(x_1, y_1), ..., (x_n, y_n) \qquad y_i \in \{-1, 1\},\ x_i \in \mathbf{R}^d. \tag{3.1}$$

In case of linear separable data, the equation

$$w_0 \cdot x + b_0 = 0 \tag{3.2}$$

with w being a vector and b a scalar, needs to be satisfied in order to find the optimal hyperplane [17]. On each side of the hyperplane are the data points of one of the classes. The data on the left has the label -1 and the data on the right side is labeled with 1 [17]. This can mathematically be written as

$$\begin{aligned} w \cdot x_i + b &\geq 1 \quad if \quad y_i = 1 \\ w \cdot x_i + b &\leq -1 \quad if \quad y_i = -1 \end{aligned} \tag{3.3}$$

with $i = 1, ..., n$. Putting these two equations together results in

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, ..., n. \tag{3.4}$$

Only the data that fulfill the following equation are marginal and are the previous mentioned support vectors [17].

$$y_i(w_0 \cdot x_i + b) = 1 \tag{3.5}$$

These support vectors are the points that lie nearest to the hyperplane [17]. To get the largest possible distance between them and the hyperplane, the expression $w \cdot w$ needs to be minimized under the inequality constraint of (3.4) [17]. This constrained optimization problem has been solved by applying the optimization technique of Lagrange [17]. With the Lagrange multiplier method and the existing conditions, the equation

$$\mathcal{L}(w, b, \mathcal{A}) = \frac{1}{2} w \cdot w - \sum_{i=1}^{n} \alpha_i [y_i(x_i \cdot w + b) - 1] \tag{3.6}$$

with $\mathcal{A}^T = (\alpha_1, ..., \alpha_n)$ and α_i being the Lagrange multiplier is formed [17]. Using the equation in (3.6), the equation to compute w_0 can be formed as the linear combination of the support vectors

$$w_0 = \sum_{i=1}^{n} \alpha_i^0 y_i x_i \tag{3.7}$$

with $\alpha_i^0 \geq 0$. This sort of margin is called the hard margin as there are no data points allowed to cross the border lines formed by the support vectors [17]. The next section will show how this method has been adapted to allow a few outliers.

3.3.2 Non-linear separable data

The data is in many cases not linear separable either because of some outliers or because linear separating the whole data is not possible in that dimension [12]. The soft margin exists for the first case. This form of the support vector machine allows a small number of outliers in order to be able to separate the remaining training data [17]. Therefore, the constraint (3.4) has been extended by the slack variable ξ resulting in (3.8).

$$\begin{aligned} &y_i(w \cdot x_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \quad for \quad i = 1, ..., n \end{aligned} \tag{3.8}$$

The slack represents the distance between a data point and the hyperplane [18]. If a data point is on the correct side of the hyperplane but in the area of the margin, the slack will be $0 < \xi_i \leq 1$ [18]. If it is on the correct side and outside of the margin, it is $\xi_i = 0$ and if it is on the wrong side of the hyperplane, it is $\xi_i > 1$ [18]. The separation of two classes with outliers is demonstrated in Fig. 3.3. Another new variable that has been defined for this case is the regularization parameter C [17]. This parameter defines the permissible amount of error [17]. It can be specified by the user. Depending on the value of C, the size of the margin and the number of outliers will change [17]. If C is larger, the margin will be smaller and the classifier less tolerant to outliers [17]. In the other way round, the margin will become larger with a smaller value of C and the tolerance will also grow resulting in more accepted outliers [17]. As a consequence of adding these two new variables, the optimization problem of the support vector machine must be adapted [17]. The goal is still to reach a maximized margin but also to minimize the error represented

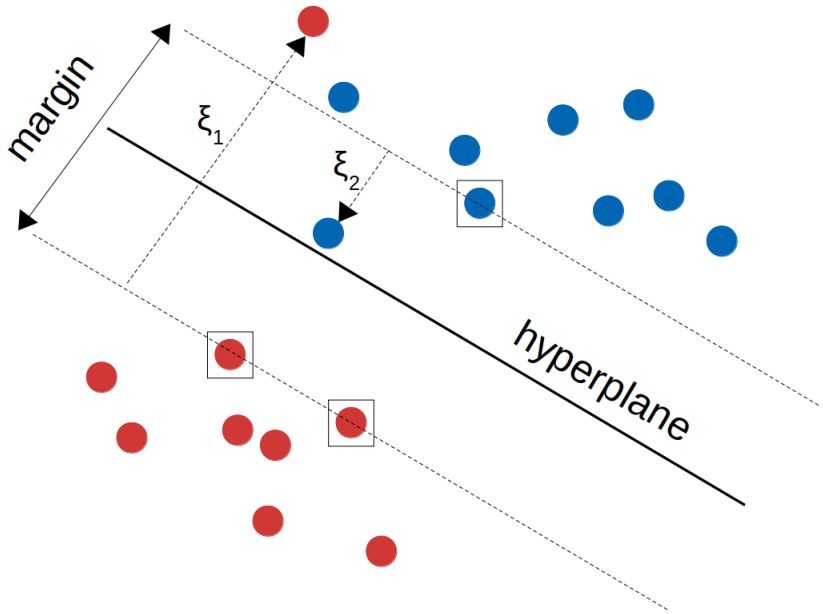


Figure 3.3: Separation of data with two outliers

by the amount of slack subject to the constraint (3.8) [17]. The expression that needs to be minimized changes from $\frac{1}{2}w^2$ to

$$\frac{1}{2}w^2 + C(\sum_{i=1}^{n} \xi_i). \tag{3.9}$$

This adaptation does not suffice for the second case where the whole data is not linear separable like the data in Fig. 3.4.

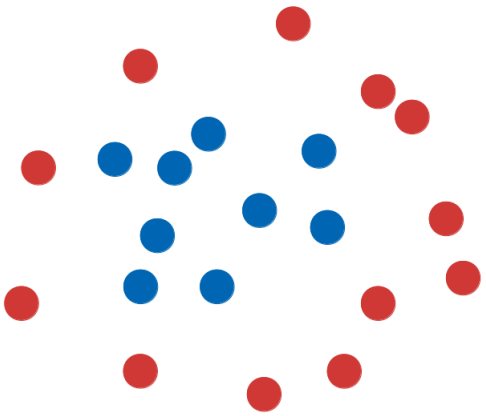


Figure 3.4: Non-linear separable data

To be able to separate data of that type, the data needs to be mapped into a higher

dimension [18]:

$$\Phi : \mathrm{R}^d \mapsto \mathrm{R}^N. \tag{3.10}$$

To avoid the necessity of finding out the correct dimension R^N and mapping the whole data into that dimension, the "kernel trick" is used [18]. With this trick, the data does not need to be mapped into a higher dimension. Instead, a kernel function is used for the dot product $x_i \cdot x_j$ resulting in

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{3.11}$$

with x_i being a support vector and x_j the input data [18]. That way, the costly computation time of mapping the data could be avoided [18]. In case of w, the kernel function is also used to adapt the equation of w to

$$w = \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i). \tag{3.12}$$

Using (3.12), the following function to determine the class of input data is formed:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) \cdot \Phi(x) + b \\ &= \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b. \end{aligned} \tag{3.13}$$

There exist a few popular kernel functions, like the radial basis function kernel (RBF) , that have been defined. The kernel function should always be chosen depending on the data to achieve good results. The one used in this approach will be described in chapter 4.4.

4 Proposed System

4.1 Overview

The proposed system is divided into three steps: image preprocessing, feature extraction and classification. The whole process of detecting traffic signs is displayed in Fig. 4.1. Firstly, images are being preprocessed to localize the traffic signs. Therefore, the images will be converted into the HSV color space and color segmentation will be applied to detect red and yellow colored areas. The resulting images contain the regions of interest. For these regions, the histogram of oriented gradients, short HOG, will be calculated, which will function as input for a previous trained support vector machine. Then, the classifier will return the class of the predicted traffic sign. If the predicted class belongs to a speed limit sign, a new histogram of oriented gradients feature vector will be computed and passed to another support vector machine. This one will predict the exact speed limit sign. After classification, a bounding box is drawn around the found traffic signs, showing where and which signs have been found. The implementation of the system can be found on GitHub [19].

The following sections will explain in detail the steps and the functionality of the chosen methods.

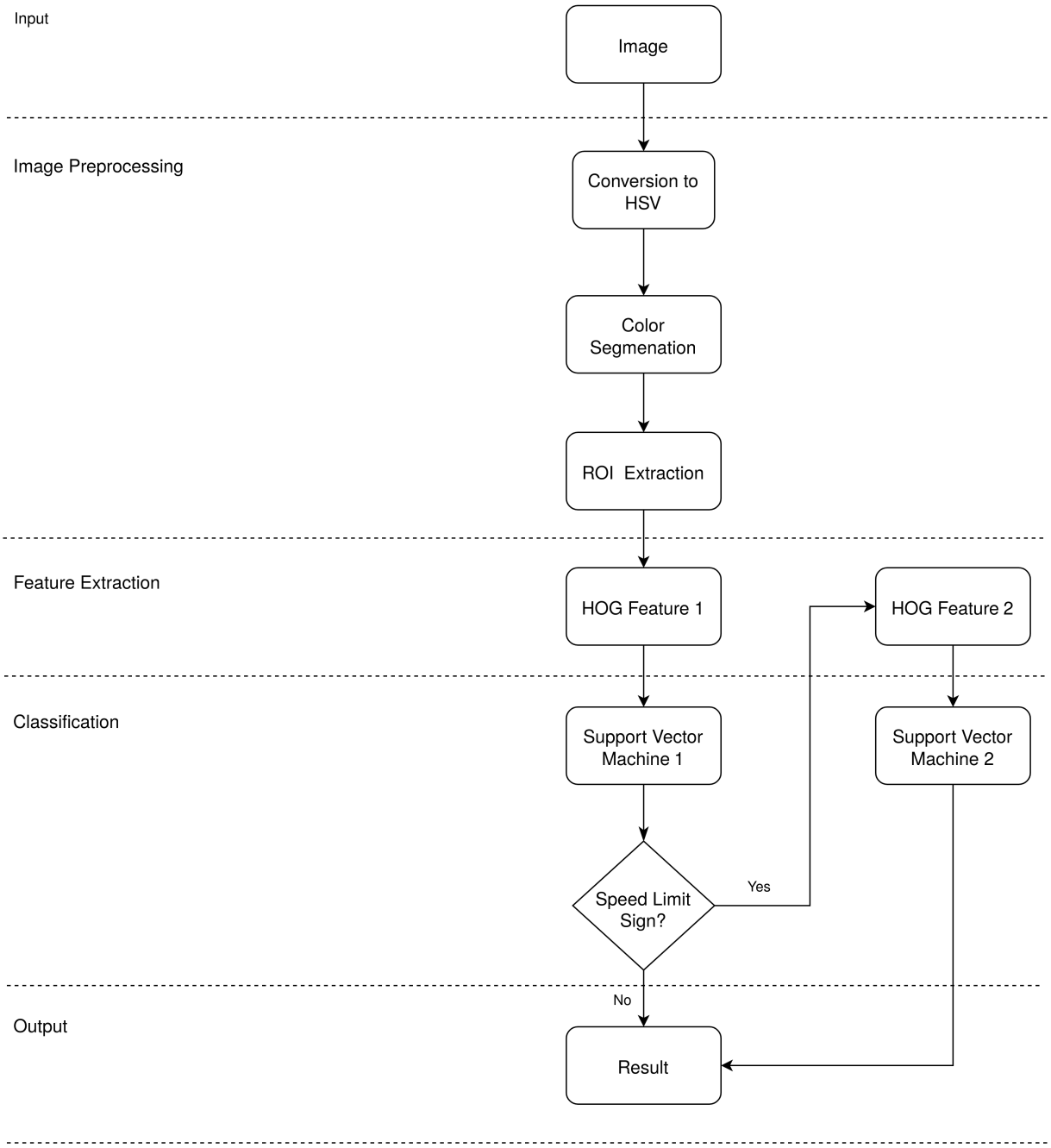


Figure 4.1: Overview of the proposed system

4.2 Image preprocessing

Localizing traffic signs in the preprocessing phase is significant for the following steps [20]. That is why finding a suitable color model and performing an accurate color segmentation is a fundamental step [2]. The following section describes the procedure of choosing a color model, finding regions of interest and preparing these regions for the next step.

4.2.1 RGB to HSV

Depending on the factors, that need to be considered, different color models lead to different efficiencies of the color segmentation [20]. Since this approach is not developed for the traffic outside of the laboratory, most of the factors, that are relevant outdoors, do not need to be considered as stated in chapter 3.2. A factor that needs to be considered even in an environment like the robotic laboratory is the change of lighting, which is caused by the windows and the ceiling lights. Also, reflection and blur are critical factors indoors. But these two factors are not being considered while choosing a suitable color model because image manipulation methods are needed to balance reflection and blur. Hence, a color model, that is at least not prone to changing light, is needed [20][3]. Many approaches use images in RGB because this model is not as complex as other color models and converting into other models requires computation time [21]. A color represented in RGB consists of the three colors red, green and blue. Because of this composition, the brightness and saturation are not independent values but part of the three values [22]. Therefore, all three RGB values of a color will change if the lighting conditions have changed [22]. Using this color model would have an impact on the color segmentation part, making it less reliable and accurate [6][1][12].

There are other color models where this problem does not exist, like HSI or L*a*b [6]. One of these models is the HSV model, which is also being used often [5]. HSV stands for hue, saturation and value and resembles the color perception of a human [1]. In contrast to the RGB model, which is represented as cube, the HSV model is formed like a cone as shown in Fig. 4.2 [23][1]. The hue is the angle from 0° to 360° where the color red

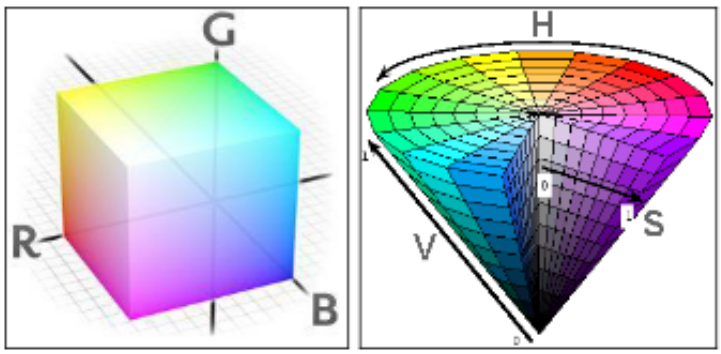


Figure 4.2: Representation of the RGB and HSV color models [1]

begins at 0°, green at 120° and blue at 240° [23]. Because the top of the cone is formed like a circle, red appears again at 360° [1] [23]. The saturation describes the intensity of the hue in a range from 0 with the most gray in the color to 1 with no gray at all [23]. This value is located from the center of the cone to the edge of it [22][23]. The value represents the brightness and ranges also from 0 resulting in a black color to 1 a very bright color [1]. In the cone it is represented as the height. Together these three values form a color in HSV [21]. The separation of the hue and the other two components is making this model less vulnerable to different lighting conditions and more suitable for the color segmentation [22] [21]. That is why, the HSV model has been selected.

Originally, the images that are being received from the model cars' camera have the format sensor_msgs/Image [24]. Using CvBridge provided by OpenCV, these images can only be converted to greyscale, BGR or RGB images [24]. Because converting to HSV in one step is not possible, the images are firstly being converted to RGB and then to HSV. Even though this costs a bit computation time, the time needed for the conversion does not affect the system's processing time that much to still consider the RGB model. The method *cvtColor*, provided by OpenCV, is being used for the conversion from RGB to HSV.

$$V = \max(R, G, B) \tag{4.1}$$

$$S = \begin{cases} V - \dfrac{\min(R,G,B)}{V} & \text{if V} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

$$H = \begin{cases} 60 \times \dfrac{(G-B)}{V - min(R,G,B)} & \text{if V} = \text{R} \\ 120 + 60 \times \dfrac{(B-R)}{V - \min(R,G,B)} & \text{if V} = \text{G} \\ 240 + 60 \times \dfrac{(R-G)}{V - \min(R,G,B)} & \text{if V} = \text{B} \end{cases} \tag{4.3}$$

$$If \quad H < 0 \quad then \quad H = H + 360 \tag{4.4}$$

First, the maximum value of the RGB values is identified and passed to the value (4.1) [25]. Then, the saturation and hue are calculated with the equations in (4.2) and (4.3) using the previous computed value [25]. If the hue is not a positive number, the equation (4.4) is used to adapt the hue [25]. As described earlier, the saturation and value lie between 0 and 1 and the hue lies between 0 and 360. Values higher than 255 would cause an overflow in an 8-bit image that is why the hue needs to be adapted [25]. Depending on the library, the range of each component varies. In OpenCV, the hue is divided by 2 to prevent the overflow [25]. This causes the range of the hue to only goes from 0 to 179 in OpenCV [25]. The saturation and value range from 0 to 255. To reach that range, they are each multiplied with 255 [25]. For image data types with more bits, different adaptations are applied since the ranges can be larger [25]. But using one of those data types would result in more memory usage for saving the images and a higher computation time while working with them.

4.2.2 Color segmentation

Instead of classifying each part of an image which would be time-consuming, regions, where a traffic sign could be, are being searched to classify them later on. Two distinct features of the signs can be used to find those regions of interest [1]. As described in

chapter 3.1, every sign has a geometrical shape and a striking color [1]. Therefore, various methods exist using either the color, the shape or both to locate and extract regions that could contain a sign [2]. The advantage of a color-based method is its simplicity and its fast processing time but there is also a disadvantage. A color can change as described in the previous section leading to no localization if the lighting is poor or if there is a reflection [2]. Nevertheless, color segmentation gives mostly good results in finding regions of interest [2]. In contrast, shape-based methods are not as prone to these factors but they require a much higher computation time [3]. That needs to be considered since the goal of a traffic sign detection system is not only high precision and recall but also low computation time [2][3]. Otherwise, the image processing could be lacking behind when the system is used while a model car is driving.

Before segmenting the converted images, they are being resized to 400x300. This improves the processing time as less pixels need to be manipulated [15]. To segment the images, thresholding using the *inRange* function of OpenCV is applied. Because the traffic signs considered for this approach are either yellow and white, red and white, or red, white and black, two color segmentations are applied. The first one will find the red areas and the second one the yellow areas. For the *inRange* function, ranges, forq the hue, saturation and value, are needed. The guide values for the red and yellow ranges have been defined in RGB as it is easier to define a color in RGB. Using the same function as in converting the whole image, the two colors are converted to HSV [26]. Then, the allowed ranges for the hue, saturation and value must be defined [26]. This has been done by trying out some ranges and adapting them until the best suited ones have been found. Here it is important, that the ranges may not be too large because then too many objects in the background would also be detected making the system less efficient. This could lead to discarding an area containing a traffic sign because too much of the background has also been extracted in that region of interest and the conditions that will be described in the next chapter will not be met. Even if those conditions will be fulfilled, a misclassification is likely to happen due to too much noise in such an area.
In case of the value, the average brightness is calculated to adapt the range to the lighting conditions in the image. If it is over 100, the minimum value will be set to 100. Otherwise, it will be set to 70. This technique has not been applied to the saturation because a few objects in the background have a red cast. Adapting the saturation to the average saturation of the image leads to intensifying the red color in these objects and extracting a lot of the background.

The defined ranges are used to check if a pixel value of an image lies within these ranges using (4.5) [27]. If the hue, saturation and value fit into the ranges, the pixel value will be changed to represent white, otherwise it will get the value of black [27].

$$\begin{aligned} dst(I) = & lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0 \ \wedge \ lowerb(I)_1 \leq src(I)_1 \\ & \leq upperb(I)_1 \ \wedge \ lowerb(I)_2 \leq src(I)_2 \leq upperb(I)_2 \end{aligned} \tag{4.5}$$

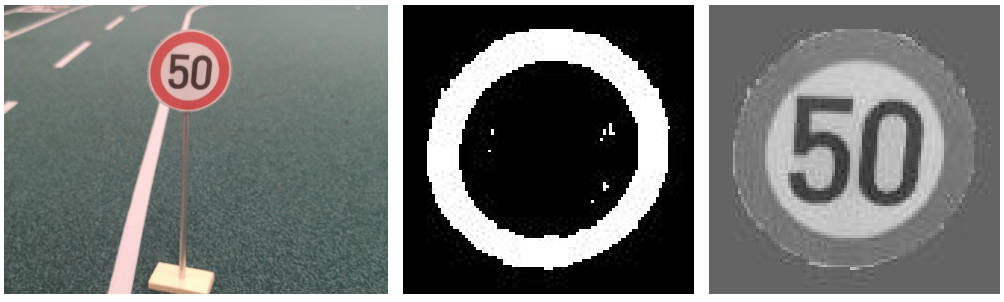
That way, two binary images will be produced for each image, one showing the areas of red spots and the other one the yellow spots. Fig. 4.3 shows such binary images for a stop sign image. The white areas are the regions of interest and represent the red and yellow areas.

(a) (b) (c)

Figure 4.3: Color segmentation of a stop sign image. (a) input image, (b) red segmentation, (c) yellow segmentation

4.2.3 Filtering regions of interest

Even though the ranges of yellow and red have been adapted in a way to not accept too much of the background, the resulting images normally still contain too many regions. To process each of them would cause the feature extraction and classification to be too time-consuming. That is why, criteria have been developed that need to be satisfied to keep an area as region of interest. First of all, the size of an area must be large enough. Traffic signs of a too small size should not be considered because they are too far away [2]. Because of the small size, the shape and numbers might not be clear enough to be properly recognized in the classification phase [7][5]. Such signs will be recognized later when the distance between the car and them gets smaller. Another criteria is the aspect ratio. Every traffic sign's height is nearly the same as its width. Therefore, areas where the proportion is greater than 1:1.4 or 1.4:1 will be discarded. An aspect ratio of 1:1 has not been chosen because such an aspect ratio is rarely reached due to background noise and overlapping objects. The remaining areas are retained for further processing.
To prepare these areas for the feature extraction, the background will be removed. Because support vector machines and the histogram of oriented gradients are quite prone to background noise, removing it will help to reach a higher precision. OpenCVs method *floodfill* is used for the background removal [28].



(a) (b) (c)

Figure 4.4: Background removal using floodfill. (a) original image, (b) extracted region of interest, (c) processed region of interest

The area enclosing the white colored spots, which possibly resemble the traffic sign, will be colored gray. Afterwards, a gray version of the original image will be created. This

grayscale image will be used to cut the area of to the region of interest out. Then, this grayscale region of interest will be mapped to the binary region of interest. That way, the traffic sign is in the foreground and the background has a uniform color as shown in Fig. 4.4.

4.3 Feature extraction

Extracting significant information from an image is essential because those information help to classify the objects in it. As mentioned in the chapter "State of the Art" , there are many methods that can be used to extract the distinctive features of an object.

4.3.1 Histogram of Oriented Gradients

A common feature extraction method is the histogram of oriented gradients. It computes a vector that resembles the change of color and structure in an image. In [29], the authors Dala and Triggs presented this method in connection with human detection. To compute the HOG feature, the gradient image must firstly be calculated [29]. It is obtained by calculating the dot product of a chosen mask and a vector of the same size containing pixel values [29]. The mask is sliding, shifted by one, over the image so that each pixel, besides the ones in the corner, contributes to three dot products [29]. For this, various kinds of masks have been proposed where the centered 1-D point derivative [1,0,-1] has been proven to be the most suited one [29]. Therefore, this mask has been chosen in most implemented versions of the histogram of oriented gradients. This applies also to scikit-image's implementation, which is being used here. Since the input images are grayscale, this step outputs only one image gradient. If a colored image would be given as input, one image gradient for each channel would have been computed and the one with the highest norm would be used [29].

$$\theta(x, y) = tan^{-1}(\frac{f/y}{f/x}) \tag{4.6}$$

$$m(x, y) = \sqrt{(\frac{f}{x})^2 + (\frac{f}{y})^2} \tag{4.7}$$

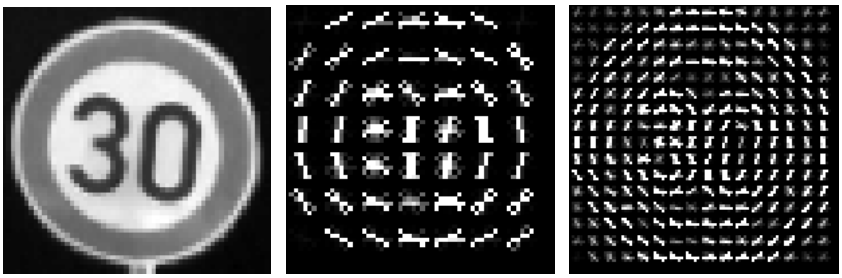
In the next step, the histogram of gradients is generated for each cell, which consists of a specific number of pixels [29]. Therefore, the vote and orientation are computed for each pixel in a cell [29]. The orientation, representing the direction of the gradient, is computed with the equation in (4.6) [29]. Different functions exist to calculate the vote, like the square of the magnitude or the square root of the magnitude [29]. Here, the vote is calculated using the function to compute the magnitude of a gradient (4.7). The calculated votes are being allocated to the orientation bins according to the orientation value [29]. The number of orientation bins tells which angle is associated with which bin. The orientation bins range from 0° to 360° if the gradient is signed [29]. If the gradient is unsigned, the bins range only to 180°, which is the normal case [29]. The gradients here are also unsigned. The votes in each bin are being added resulting in a histogram

of gradients for each cell [29].

$$b = \frac{b}{\sqrt{(\|(b)\|^2 + \epsilon^2)}} \qquad \text{with b = block of n} \times \text{m cells} \tag{4.8}$$

In the final step, normalization is performed [29]. In order to make the histogram of oriented gradients feature resistant to lighting changing and contrast, blocks of a previous defined size of cells are normalized using one of the four presented normalization types in [29]. In this approach, the L2-normalization is applied to blocks of 2x2 cells which are shifting by one to overlap each other [29]. L2-normalization is calculated using (4.8). Because scikit-image's implementation of the HOG feature is used, ϵ = 1e-5 [30]. The results of the block normalizations form the histogram of oriented gradients descriptor.

Dala and Triggs showed that adapting the variable parameters is an important step in order to find the ones with the best results when using the HOG feature [29]. Especially the size of the cells and blocks depend on the image size and how detailed the structure needs to be represented [29]. They found out that nine bins give the best results for their approach which also applies for this system [29]. The number of pixels has been set to 8x8 to get a rougher result when computing the HOG for all classes. In case of the speed limit signs, a more precise histogram is needed which is why the number of pixels for the second HOG feature descriptor has been defined as 6x6. As mentioned, the number of cells per block is 2x2 for both methods.



(a) (b) (c)

Figure 4.5: HOG of a speed limit sign. (a) input, (b) using the parameters for the first feature vector, (c) using the parameters for the second feature vector

A necessary criterion when using the feature descriptors as an input vector for the support vector machines is a uniform size. In case of the first HOG method, the regions of interest will be resized to 60x60 before the HOG feature descriptor is calculated. This size is large enough to obtain the shape. But for the second HOG feature descriptor, a larger size is needed to differentiate the numbers. Here it is important to not choose a too large size because otherwise it would cost too much computation time. Therefore, the images are resized to 100x100. An example of the difference between the two histogram of oriented gradients is shown in Fig. 4.5. It is visible that the first output is much rougher than the second one where the number is much clearer. Concluding, the resulting feature descriptor will be flattened before it is used as an input vector in the following classification part.

4.4 Classification

The last step of the detection system includes the classification where the located regions of interest are either classified as a specific traffic sign or rejected.
The support vector machine introduced in chapter 3.3 is only able to differentiate two classes. Given that this detection system is supposed to classify six traffic signs, the binary version of the support vector machine is not suitable.
Instead, multi-class support vector machines are needed. Two common versions exist: *one-vs-one* and *one-vs-rest*. Because the machine learning module of OpenCV is used, the multi-class approach *one-vs-rest* is applied for both classifiers [31]. This approach creates M support vector machines for M classes [32]. Each of these classifiers will be trained to detect a specific class I [32]. The samples of the other M-1 classes will belong to the second class [32]. Predicting an object using this technique involves using all trained classifiers to get M predictions and choosing the class with the largest margin as result [32]. Unfortunately, this method requires a lot memory while training the classifiers [32]. In addition to that, the ratio between the classes is always unbalanced [32]. An unbalanced number of samples can result in a badly trained classifier and a bad prediction accuracy. Since more samples of one class are used while training the classifier, it will predict more often the class with the many samples. To prevent this problem, weights can be added to even this out [17].
In contrast to *one-vs-rest* where the number of classifiers equals the number of classes, the number in one-vs-one can get much larger because it equals the binomial coefficient of M and two [32]. But the balance between the two classes is much better [32].

The classifier's type is C_SVC. This type provides the regularization parameter C and as described in chapter 3.3.2 allows a small number of outliers in order to separate the data linearly [17]. Also, the kernel trick has been applied. Even though the most common kernel is the RBF one, this system uses the histogram intersection kernel. Swain and Ballard introduced the technique of histogram intersection in 1991 [33]. Back then, the goal was to develop a new method for color indexing. The first time it has been used in form of a kernel was in [34] for image classification. The histogram intersection kernel returns a value representing the similarity between two given histograms [34]. The authors state that histograms are a simple and efficient way to represent color information making this kernel so efficient [34]. In order to apply this kernel, both histograms must have the same size so that the number of bins is equal [34]. Only then, their pixels can be compared using the equation

$$\mathrm{K}_{int}(A, B) = \sum_{i=1}^{n} \min(a_i, b_i) \tag{4.9}$$

with n representing the number of pixels and therefore the number of bins [34]. It has been stated that this type of kernel is faster and gives a more precise prediction than other kernels [34].

Both support vector machines have been trained using images of the created dataset. The training set consists of 20-26 images for each class. A larger number would lead to a higher computation time with similar results. Contrary to the first support vector

machine, the second one has been trained using only the images of the speed limit signs since it is supposed to classify only these traffic signs. In addition to the six and three classes for the traffic signs, a class has been added for discarding an input and labeling it as "no sign". Otherwise, the support vector machines would not have the option to label an input as no traffic sign. The regularization parameter C has been set to 10 in both classifiers after comparing the system's performance using different C values. The traffic signs have been labeled with 1 to 6 for the speed limits signs 30, 50 and 60, the stop, priority road and yield signs. The additional "no sign" class has the label 0. The speed limit signs have not been labeled with the same number but keep the label 1,2 and 3 in the first support vector machine in order to prevent an unbalanced number of training data.

5 Experiments

Evaluating a new system is an important part of the whole development process. It shows how well the system works and where its weaknesses are.
The presented system is being evaluated utilizing the results of two experiments. In the first experiment the system's performance is being analyzed using the generated dataset (chapter 3.2). The second one uses recorded rosbag files of a driving model car. Three values are being analyzed: precision, recall and the time needed to process an image.
To calculate the precision, the equation

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{5.1}$$

has been used. This value resembles how often an object has been correctly recognized and how many objects have been wrongly assumed to be that object. Another important factor is the recall. It reflects how often an object has been misclassified. The recall is computed using

$$recall = \frac{true\ positive}{true\ positive + false\ negative}. \tag{5.2}$$

The experiments have been performed on a computer with a 2.0 GHz Intel Core i7 CPU and 8G RAM.

5.1 Experiment 1: Dataset

After removing the training data from the created dataset, 3344 images remain that are being used for this experiment. There are between 420 and 620 images for each sign. They include different lighting, angles, distances to the camera and a few blurry images.

Table 5.1 shows the calculated precision and recall values. The precision reached over 90% for each sign, in case of the yield sign even 100%. The lowest precision belongs to the speed limit sign 30 with 93.3%. However, the recall value is not that high since it is approximately 77%.

traffic sign	precision	recall
30	93.3%	76.2%
50	97.6%	77.9%
60	96.4%	78.6%
stop	99.8%	76.0%
priority	98.0%	72.5%
yield	100%	81.4%
macro average	97.5%	77.1%

Table 5.1: Exp. 1: precision and recall

To get a more detailed overview on the distribution of the classification, a confusion matrix is provided by Table 5.2. There, the concrete numbers of correct and wrong classifications and the signs which have been mistakenly classified as other signs can be viewed. It is visible that the number of correct classifications resembles the precision values. The speed limit signs have been mixed a few times because of their high degree of similarity. Especially, when a speed limit sign is blurry, the chance of a mix-up is high. Furthermore, the priority road sign has been mistakenly detected when there was no sign. Both, the base of that sign and some of the boxes in the robotic laboratory are yellow leading to such a misclassification. Noticeable is that the false negatives mainly consist of missed traffic signs. Accordingly, it can be assumed that if a traffic sign can be located, there will be a high chance of correct classification.

		predicted class						
		no sign	30	50	60	stop	priority	yield
true class	no sign	133	2	0	0	0	7	0
	30	89	349	11	9	0	0	0
	50	118	9	483	9	0	1	0
	60	114	14	1	477	1	0	0
	stop	135	0	0	0	428	0	0
	priority	148	0	0	0	0	390	0
	yield	79	0	0	0	0	0	346

Table 5.2: Confusion matrix

Another crucial factor is the computation time. The average time needed to detect one traffic sign in an image is represented in Table 5.3. The processing time for the speed limit signs is approximately one third higher than the time needed for the other signs. This is caused by the application of the second classifier in case of a speed limit sign. In total, the processing time lays under 100ms.

traffic sign	time per image
30	96ms
50	85ms
60	89ms
stop	50ms
priority	62ms
yield	52ms

Table 5.3: Average processing time

5.2 Experiment 2: Driving model car

The second experiment deals with detecting traffic signs while a model car is driving. For this task a model car with a velocity of 130 rpm was driving on the outer lane of the oval route in the robotic laboratory. The traffic signs have been placed in varying positions next to that lane. Table 5.4 represents the number of missed and correct and wrong classified signs. The rosbag files five and six, where most of the signs have been positioned near a curve, show a higher number of missed traffic signs than the other files where most of the signs have been placed on a straight lane.

	traffic signs	correct classified	misclassified	missed
rosbag file 1	3	2	0	1
rosbag file 2	5	3	1	1
rosbag file 3	16	14	0	2
rosbag file 4	15	11	0	4
rosbag file 5	19	7	1	11
rosbag file 6	6	4	0	2

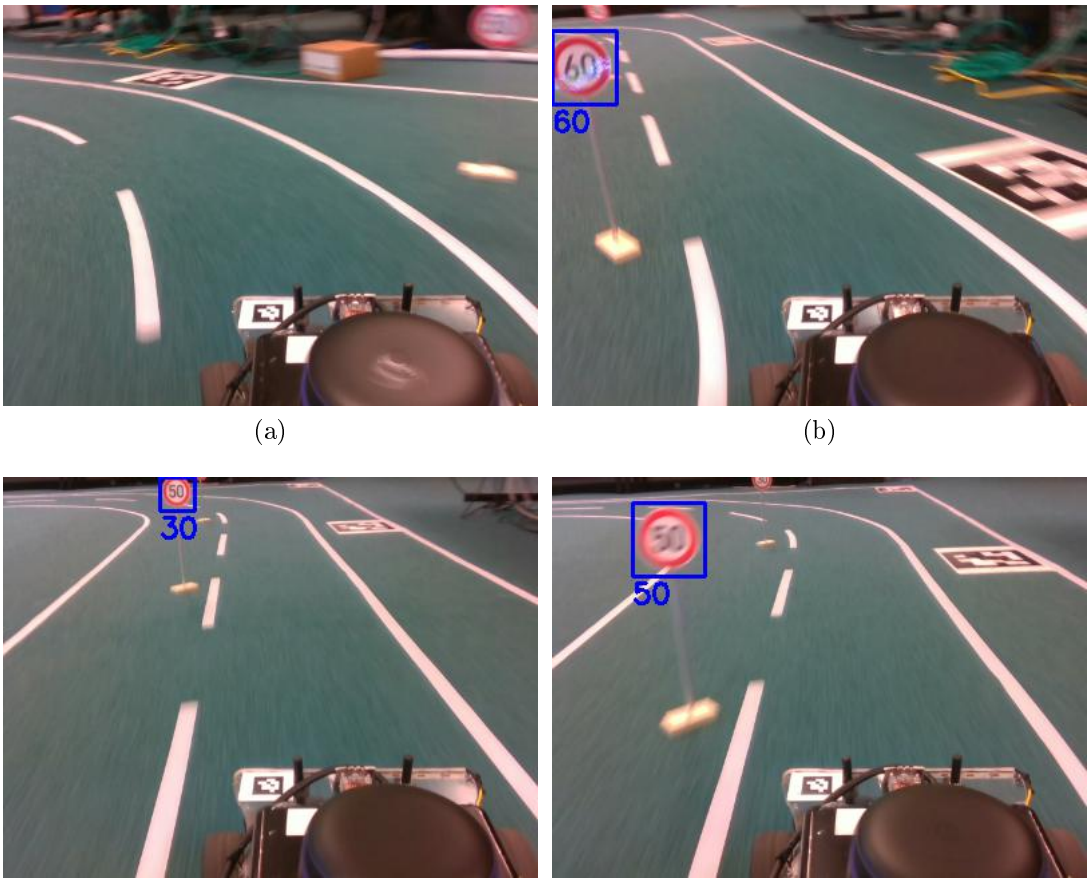
Table 5.4: Overview of the detected traffic signs in rosbag files

The recall values in Table 5.5 are reflecting the high missing rate, too. Especially the fifth rosbag file has the worst recall value due to the many signs placed near and in a curve. Examples of the different circumstances are shown in Fig. 5.1. However, no matter where the signs have been placed, the number of misclassifications is quite low resulting in precision values over 70%, mostly even 100%.

	file 1	file 2	file 3	file 4	file 5	file 6	macro average
precision	100%	75.0%	100%	100%	87.5%	100%	93.75%
recall	66.7%	60.0%	87.5%	73.3%	36.8%	66.7%	65.2%

Table 5.5: Exp. 2: precision and recall

It is important to remark that those numbers can vary using other datasets. It is not possible to catch all possible positions and angles in the experiments. Therefore, it is hard to make a clear statement about the precision ad recall values of a vision-based system. However, the experiments revealed a lot about the factors that matter in order



(a) (b)

(c) (d)

Figure 5.1: Detected traffic signs while a car was driving (a) in a curve, (b) with reflection on a sign, (c) with a large distance, (d) with a blurry sign.

to detect traffic signs properly. It is noticeable that some positions are not optimal for the system to detect the signs properly. Traffic signs that are too far on the left or right side have not been able to be seen by the camera because they have not been in its view field. Another aspect is the placement in or near a curve. Because the model car needs to steer to follow the lane in a curve, the camera is not able to take clear images in those situations. These images are so unclear that the traffic signs in them are not even recognizable by the human eye.

6 Conclusion and Outlook

This thesis presented a traffic sign detection system for the model cars of the research group autonomous cars. The system has been developed using support vector machines and the histogram of oriented gradients to calculate feature vectors as input for these classifiers. This combination has been proven to be a good choice, especially when using the histogram intersection kernel with the histogram of oriented gradients. Since the system has been trained with data created in the environment of the robotic laboratory of the Freie Universität Berlin, the approach will most likely not work well in an outdoor environment. Since it has been purposely adapted to the surroundings it will be used in, many objects outdoors would irritate the system and compromise the performance.

It has been shown that the system is very precise, but it lacks slightly in recall as it reaches only 77% using the dataset and 65% using the rosbag files. Such a recall value is critical for a traffic sign detection system, especially if it would be used outdoors. Missing traffic signs could cause a lot of damage on the streets. The cause of these low recall values are mainly two factors. It turned out that the placement of traffic signs plays a decisive role while detecting them. Signs near a curve will most likely be missed by the system but signs on a straight line will be detected and in most of the cases correctly classified. Furthermore, the limited view field of the model cars' camera needs to be considered when placing the traffic signs. They need to be close enough to the lane the car is driving on, in order to be able to be recorded. Also, the angle at which a sign has been placed is a very essential criterium. The other factor leading to these recall values is the method used to remove the background in a region of interest and the ranges that have been defined for the color segmentation. Removing the background is necessary due to the sensitivity of the support vector machines to background noise but the used method needs to be improved.
Nevertheless, the resulted system provides a basis for future work, though it needs to be adapted to give more reliable results. A few possible adaptations will be described below.

In order to improve the recall, the image preprocessing part needs to be adapted.
Improving the existing method or finding a different method to remove the background may be a crucial point. The current method, *floodfill*, relies on a perfectly complete white area, the border or the traffic sign, which encloses the inner part. Only then, it is possible to color the background in gray without taking parts of a traffic sign and coloring them, too. Furthermore, the background will sometimes not be removed at all, leading to too much noise in a calculated histogram of oriented gradients. These factors result in misclassification and primarily rejection. With a more reliable method, the number of

missed traffic signs would decrease.
The color segmentation is another point which needs improvement. Many traffic signs have not been detected because their color values did not lay within the defined ranges. But expanding the ranges results in extracting too much background noise as stated in chapter 4.3. If a method that separates the traffic signs from the background is added, the ranges can be expanded and the noise can be removed before the regions of interest are being further processed. In case of the priority road sign, another color segmentation to detect white areas would help to make sure that a white area is enclosing a yellow area. That way, other yellow areas can be rejected reducing the number of misclassification.

To achieve a higher precision, combining the histogram of oriented gradients with other feature extraction methods could be tested since Ellahyani et al. stated that such a combination can achieve better results [8]. Moreover, trying out different classification methods, like neural networks, can result in an overall better performance. In some papers the usage of neural network has resulted in detection systems that are capable of real-time processing without compromising the precision and recall [9][4][10]. Of course, it needs to be taken into account that the hardware used in these papers is more powerful than the one of the model cars. But with some adaption, neural networks might be another good classifier.

Bibliography

[1] K. Horak, P. Cip, and D. Davidek, "Automatic traffic sign detection and recognition using colour segmentation and shape identification," *MATEC Web Conf.*, vol. 68, pp. 17002–1, Aug. 2016, Accessed on: Jan. 5, 2020. [Online]. Available: https://doi.org/10.1051/matecconf/20166817002

[2] S. B. Wali, M. A. Abdullah, M. A. Hannan, A. Hussain, S. A. Samad, P. J. Ker, and M. B. Mansor, "Vision-based traffic sign detection and recognition systems: Current trends and challenges," *Sensors*, vol. 19, no. 9, May 2019, Accessed on: Jan. 20, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/19/9/2093

[3] Y. Saadna and A. Behloul, "An overview of traffic sign detection and classification methods," *International Journal of Multimedia Information Retrieval*, vol. 6, no. 3, pp. 193–210, Sep. 2017, Accessed on: Feb. 7, 2020. [Online]. Available: https://doi.org/10.1007/s13735-017-0129-8

[4] F. Moutarde, A. Bargeton, A. Herbin, and L. Chanussot, "Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system," in *2007 IEEE Intelligent Vehicles Symposium*, June 2007, pp. 1122–1126, Accessed on: Feb. 1, 2020. [Online]. Available: https://doi.org/10.1109/IVS.2007.4290268

[5] C. F. Paulo and P. L. Correia, "Automatic detection and classification of traffic signs," in *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '07)*, June 2007, pp. 11–11, Accessed on: Jan. 20, 2020. [Online]. Available: https://doi.org/10.1109/WIAMIS.2007.24

[6] L. Chen, Q. Li, M. Li, and Q. Mao, "Traffic sign detection and recognition for intelligent vehicle," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 908–913, Accessed on: Feb. 5, 2020. [Online]. Available: https://doi.org/10.1109/IVS.2011.5940543

[7] 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*, vol. 8, no. 2, pp. 264–278, June 2007, Accessed on: Feb. 1, 2020. [Online]. Available: https://doi.org/10.1109/TITS.2007.895311

[8] A. Ellahyani, M. El Ansari, I. El jaafari, and S. Charfi, "Traffic sign detection and recognition using features combination and random forests," *International Journal*

of Advanced Computer Science and Applications, vol. 7, Jan. 2016, Accessed on: Jan. 6, 2020. [Online]. Available: https://doi.org/10.14569/IJACSA.2016.070193

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788, Accessed on: Jan. 10, 2020. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[10] D. Tabernik and D. Skočaj, "Deep learning for large-scale traffic-sign detection and recognition," pp. 1–14, May 2019, Accessed on: Jan. 7, 2020. [Online]. Available: https://doi.org/10.1109/TITS.2019.2913588

[11] J. Greenhalgh and M. Mirmehdi, "Real-time detection and recognition of road traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1498–1506, Dec. 2012, Accessed on: Jan. 4, 2020. [Online]. Available: https://doi.org/10.1109/TITS.2012.2208909

[12] A. Madani and R. Yusof, "Traffic sign recognition based on color, shape, and pictogram classification using support vector machines," *Neural Computing and Applications*, vol. 30, pp. 2807–2817, Feb. 2017, Accessed on: Jan. 15, 2020. [Online]. Available: https://doi.org/10.1007/s00521-017-2887-x

[13] K. Chaiyakhan, A. Hirunyawanakul, R. Chanklan, K. Kerdprasop, and N. Kerdprasop, "Traffic sign classification using support vector machine and image segmentation," Jan. 2015, pp. 52–58, Accessed on: Jan. 26, 2020. [Online]. Available: https://doi.org/10.12792/iciae2015.013

[14] B. der Justiz und für Verbraucherschutz. (2017, May) Allgemeine verwaltungsvorschrift zur Änderung der allgemeinen verwaltungsvorschrift zur straßenverkehrs-ordnung (vwv-stvo) teil 1 in der bekanntmachung vom 22. mai 2017 (ebanz at 29.05.2017 b8). Accessed on: Feb. 1, 2020. [Online]. Available: https://www.bundesanzeiger.de/ebanzwww/contentloader/BAnz_AT_29_05_2017_B800.pdf?state.action=genericsearch_loadbundolpdf&state.filename=BAnz_AT_29_05_2017_B800.pdf&state.pubcode=14901027&&state.orig_filename=170411001079M001.pdf

[15] T. T. Le, S. T. Tran, S. Mita, and T. D. Nguyen, "Real time traffic sign detection using color and shape-based features," in *Intelligent Information and Database Systems*, N. T. Nguyen, M. T. Le, and J. Świątek, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Mar. 2010, pp. 268–278, Accessed on: Feb. 5, 2020. [Online]. Available: https://doi.org/10.1007/978-3-642-12101-2_28

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, p. 273–297, Sep. 1995, Accessed on: Jan. 20, 2020. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[17] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, May 2011, Accessed on: Jan. 21, 2020. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[18] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, Jan. 1998, Accessed on: Jan. 21, 2020. [Online]. Available: https://doi.org/10.1023/A:1009715923555

[19] source code of the implementation. (2020, Feb.). [Online]. Available: https://github.com/RimBaa/traffic_sign_detection.git

[20] D. J. Bora, A. K. Gupta, and F. A. Khan, "Comparing the performance of l*a*b* and hsv color spaces with respect to color image segmentation," June 2015, Accessed on: Jan. 10, 2020. [Online]. Available: http://arxiv.org/abs/1506.01472

[21] X. Wang, R. Hänsch, L. Ma, and O. Hellwich, "Comparison of different color spaces for image segmentation using graph-cut," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 1, Jan. 2014, pp. 301–308, Accessed on: Jan. 10, 2020. [Online]. Available: https://www.cv.tu-berlin.de/fileadmin/fg140/VISAPP_2014_127_CR.pdf

[22] J.-D. Chang, S.-S. Yu, H.-H. Chen, and C.-S. Tsai, "Hsv-based color texture image classification using wavelet transform and motif patterns," vol. 20, Jan. 2010, pp. 63–69, Accessed on: Jan. 10, 2020.

[23] S. Sural, Gang Qian, and S. Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," in *Proceedings. International Conference on Image Processing*, vol. 2, Sep. 2002, pp. II–II, Accessed on: Jan. 11, 2020. [Online]. Available: https://doi.org/10.1109/ICIP.2002.1040019

[24] G. Solak. (2019, Mar.) cv_bridge/tutorials/convertingbetweenrosimagesandopencvimagespython - ros wiki. Accessed on: Dec. 1, 2019. [Online]. Available: http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

[25] G. Bradski. (2020, Jan.) Opencv: Color conversions. Accessed on: Jan. 10,2020. [Online]. Available: https://docs.opencv.org/trunk/de/d25/imgproc_color_conversions.html

[26] ——. (2019, Dec.) Opencv: Changing colorspaces. Accessed on: Dec. 20, 2019. [Online]. Available: https://docs.opencv.org/trunk/df/d9d/tutorial_py_colorspaces.html

[27] ——. (2019, Dec.) Opencv: Operations on arrays. Accessed on: Dec. 20, 2019. [Online]. Available: https://docs.opencv.org/trunk/d2/de8/group__core__array.html#ga48af0ab51e36436c5d04340e036ce981

[28] ——. (2019, Jan.) Opencv: Miscellaneous image transformations. Accessed on: Jan. 9, 2020. [Online]. Available: https://docs.opencv.org/trunk/d7/d1b/group__imgproc__misc.html#gaf1f55a048f8a45bc3383586e80b1f0d0

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1, Accessed on: Jan. 17, 2020. [Online]. Available: https://doi.org/10.1109/CVPR.2005.177

[30] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 6 2014, Accessed on: Jan. 20, 2020. [Online]. Available: https://doi.org/10.7717/peerj.453

[31] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000, Accessed on: Jan. 14, 2020.

[32] M. Pal, "Multiclass approaches for support vector machine based land cover classification," Mar. 2008, Accessed on: Jan. 20, 2020. [Online]. Available: https://arxiv.org/abs/0802.2411v1

[33] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, p. 11–32, Nov. 1991, Accessed on: Jan. 23, 2020. [Online]. Available: https://doi.org/10.1007/BF00130487

[34] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," vol. 3, Oct. 2003, pp. III – 513, Accessed on: Jan. 22, 2020. [Online]. Available: https://doi.org/10.1109/ICIP.2003.1247294