CHALMERS

Study Real World Traffic Environment Using Street Views

A Virtual Expedition

Master's thesis in Systems Control and Mechatronics

KONSTANTINOS MITRITSAKIS

Department of Signals and Systems
Automation and Mechatronics Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015
Master's thesis EX072/2015

Master thesis EX072/2015

Study Real World Traffic Environment Using Street Views

A Virtual Expedition

KONSTANTINOS MITRITSAKIS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Automation and Mechatronics Engineering
Chalmers University of Technology
Gothenburg, Sweden 2015

Study Real World Traffic Environment Using Street Views
A Virtual Expedition
KONSTANTINOS MITRITSAKIS

Supervisor: Feng Liu, CAE Active Safety, Volvo Cars Corporation
Examiner : Prof. Irene Yu-Hua Gu, Dept of Signals and Systems Chalmers University of Technology

Master thesis EX072/2015
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Busy road, real world traffic environment, Heraklion, Greece. Image by Manos Tsirantonakis.

Typeset in LaTeX
Printed by Chalmers Bibliotek, Reproservice
Gothenburg, Sweden 2015

Study Real World Traffic Environment Using Street Views
A Virtual Expedition
KONSTANTINOS MITRITSAKIS
Department of Signals and Systems
Chalmers University of Technology

Abstract

This thesis was conducted at Volvo Cars Corporation located in Torslanda, Gothenburg. The aim of this thesis is to develop a method able to obtain images of road scenery and recognize traffic signs from these images. An almost unlimited source of traffic sign images can be found in Google Street View, thus the method is able to download images from Google Street View for any given route and look for traffic signs. By a sliding window approach the image is extensively scanned for traffic sign locations hypothesis. For every hypothesis within the initial image, features based on Histogram of Oriented Gradients (HOG) are being extracted and used as an input to the detector. The proposed approach uses Support Vector Machines (SVM) in order to detect and extract the traffic signs from the images. During the training procedure of all the SVMs, artificially created traffic signs were used and they have been processed in such a way to look pragmatical. Each of the extracted signs is then classified further according to the information containing via the one against one approach. The classification module also uses an Optical Character Recognition (OCR) function that is able to recognize characters within the traffic signs. The final result of the current traffic sign recognition module stems from combining both the SVM classifier and the OCR function. Application of this master thesis is to find a large amount of traffic signs capable of training neural networks for Volvo's own purposes or to test other traffic sign recognition functions. Basic tools for the implementation of the algorithms were Python and Matlab.

Keywords: Traffic Sign Recognition, Detection, Classification, Support Vector Machines, Histogram of Oriented Gradients, Google Street View.

Acknowledgements

I would like to thank my family and my friends for their continuous and unbounded support. Also I would like to thank my colleagues at Volvo Cars Corporation and more specifically Georgios Minos's group for their help and for their efforts to make me feel welcomed. Furthermore I should thank professor Irene Yu-Hua Gu for her advice and her contribution of the current report.
Last but not least I would like to thank my supervisor from Volvo Cars Corporation, Feng Liu for his everyday unlimited efforts for productive guidance.

Konstantinos Mitritsakis, Gothenburg, 2015 12

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

In a complex traffic environment traffic signs play a pivotal role in the regulation of any kind of traffic anomalies and aim to dynamically warn, prohibit and enforce the driver for specific actions. In order to do that traffic signs adopt standardized properties such shape, colour, pattern and text to visualize important information to all road users. These standardized properties are being used at the most from a Road Sign Detection System in order to assist a driver. Aspects for such a system will be introduced and the whole procedure from training until testing will be analyzed in the next sections. This chapter concentrates in analyzing what kind of research has been done in the area of detection and recognition of traffic signs. Also it gives an overview of the different traffic signs and how they look like while it mentions which of them are within the scope of the current thesis. Finally the potential applications are being introduced, while also the challenges of building a system like that are being explained.
In chapter 2 all the basic theory related to this thesis is being explained while chapter 3 contains a thorough overview of how this theory was implemented. In chapter 4 all the related results are being presented followed by the conclusion.

1.1.1 Previous Work

Traffic sign detection and recognition is an important part of Advanced Driver Assistance Systems (ADAS) and is a complex subject that can be partitioned in many critical sub-problems.
It appears in the literature that a widely used way to divide the problem of traffic sign recognition is into two major steps: The first step is to find a way to search for traffic signs and then to classify them. This classification could involve two cases e.x. traffic sign or not, or more cases e.x. distinguish between more traffic signs. A traffic sign detection and recognition project [1] was recently conducted on behalf of Volvo Cars Corporation for specific areas in China that was able to detect and classify a large number of different traffic signs. In the following sections apart from the stages of detection and recognition important sub steps will be covered in order to clarify what is the previous relevant work in each step.

1.1.1.1 The Search Problem

Since traffic signs are designed to be easily noticeable and distinguishable from background, they are brightly colored in red, blue and yellow and at the same time are being grouped into categories via specific shape. So by simply grouping pixels in images according to colours like those, traffic signs can be detected fast. This first assort is performed by using different colour information in the majority of the literature. It is widely known that algorithms that make use of colour information are sensitive to illumination changes and can easily be affected by weather conditions too. Due to various colour properties different colour spaces are being used in a variance of methods. For instance in [2] a colour segmentation algorithm based on Hue, Saturation and Intensity (HSI) colour space has been used with an extra red enhancement and adaptive thresholding taking into account the illumination of all the pixels within the image. Furthermore in [4] after having tested several different colourspaces over the same images, it has been concluded that HSI performs better, due to the fact that it is more robust in illumination changes, while according to [10] HSI is computationally expensive. [12] explains the invariance properties of another color space that takes the logarithm of ratios of the well know Red Green Blue (RGB) colour space and creates a two dimensional colour space that is capable of being an efficient solution for road sign segmentation. Finally in [5] a method in normalized RGB space is being evaluated in faded and vandalized traffic signs. Not all the approaches segment the image thus it is not always applied. Usually when a matter of computational speed occurs a way to reduce the search space within an image must be found. Segmentation is one way.
After colour filtering has determined potential regions of interest (ROI) a further shape analysis usually takes place. Multiple shape detection algorithms have been created and the majority of them use basic image analysis techniques, like edge detection, in order to reveal known geometrical shapes within an image. These techniques are not only used due to the fact that they are invariant to illumination changes but they are also used for overtaking the difficulties of recognizing signs in different countries. For instance in [14] the benefits of adding shape analysis are being investigated and two systems were developed. A shape detection system based on Hough-transformation for circular signs is able to recognize signs from the European Union and a rectangle-detection system based on edge detection is able to detect signs in the USA. In this way a system could be designed to detect different geometric shapes of signs worldwide. Algorithms that make use of shape information can suffer since the signs can appear in the images rotated and/or distorted due to the lens of the camera that captures the images. In [6] the problem of rotation was solved by applying geometrical criteria in the ROIs while in [7] and [9] the problem of distortions was solved by applying affine transformations in order to rectify the shape of the captured sign. It is also stated that deformed and rotated traffic signs can negatively affect the recognition.

There are some cases of applications that do not apply directly the previous two methods, instead the whole image is scanned with a sliding window approach. In this method there is usually a pre-set of the size(s) and the step(s) of the sliding window and then the whole image is scanned extensively. The content of every

window is evaluated by the classification algorithm determining if exists traffic sign or not. Also this method can be combined with the methods mentioned before. E.g. in [8] no prior assumption on the location of the sign within the image is made and instead a sliding window approach is being used in order to check for signs in all possible locations. Sliding window approach is used extensively in another major ADAS that has to do with pedestrian detection. This due to the reason that pedestrians do not have an exact geometrical shape or exact colour thus methods that look for colour or shape can not be applied.

1.1.1.2 Classification

The classification methods that are being used for predisposing the substance of the traffic sign are devived into two major methods: In learning methods and in nearest neighbor methods.
In the learning methods an optimal separation boundary between the different categories is trying to be achieved. These methods includes Support Vector Machines (SVM), Neural Networks (NN) and Adaboost. In nearest neighbor methods are correlating the current testing sample with the most relevant training sample. In this category techniques like template matching and tree classifiers are being used.
A lot of available work exists when it comes to the classification task. In [10] a thorough comparison of different classification techniques take place. A K-d tree classifier, a Random Forests and an SVM classifier are being compared and it is concluded that the choice of the suitable classifier depends on the cardinality of the training data set. In [11] multiple SVM classifiers are being used only to classify the shape of the traffic sign, while in [3] cascaded SVM classifiers are trained in order to recognize the sign. On the other hand in [2] traffic signs are being classified by SVM, K-d trees and Random Forests and a synopsis of the different classification results is being offered.
There is also a special case of classification that aims to directly recognize the content of the traffic signs (mainly speed limits,information signs, etc.) by applying Optical Character Recognition (OCR) [1]. In this technique the alpharithmetical contents of the traffic sign are being extracted by classical image analysis algorithms (like edge detection) and used as an input to an OCR algorithm. Then the output of this software determines were the sign is being classified. In [1] text from traffic signs is being isolated, filtered and finally processed by an external Matlab toolbox called 'VLFeat'. In [14], [15], [16] and [18] a publicly available external software named 'Tesseract' is being used. It is concluded that OCR has a high potential in traffic sign classification but its accuracy can be easily hindered by factors such as blurred, low quality, degraded and partially covered signs.

1.1.1.3 Important sub steps

Two important sub steps of the whole process is how to find traffic signs (samples) to train or test a traffic sign recognition algorithm and depending the learning method used in what kind of features these samples must be transformed.

It seems that according to the literature they are three ways of obtaining samples for the TSR function. Depending the kind of traffic signs the TSR function is used for usually, at least in an initial stage the most applications use traffic signs that they can be found either publicly available or within the inner circles of private institutions. For instance large universities or companies that are working with computer vision applications related with traffic signs have their own data base of traffic signs. A well known and widely used data base for German traffic signs is the German Traffic Sign Detection Benchmark commonly referred as GTSDB [19]. In [10], [13], [1], GTSDB has been used at some parts either to train or test a TSR function. A similar data base which is used in [17] is called NICTA Road Scene DataBase (NICTA RSDB) [20]. On the other hand during a TSR function development often a very simple and less accurate system is being used to avoid any manual work of collecting samples. Usually a system is being built based on the methods described before in the Sections 2.2.1.3 and 1.1.1.2 in images obtained usually by a video stream or by an application similar to Google Street View (GSV) [1], [11], [8]. Finally another possible way to find samples for a TSR function is to create them. In this case if the available samples are not enough, the available ones can be modified by changing their geometrical and illumination properties in order to produce more samples. In [3] states that it is time consuming to collect samples for training and sometimes is impossible to gather a sufficient amount of training data especially if the sign that needs to be recognized is not common to find it. Thus it is proposed to use easily available graphical data and synthetically generate training data for the classifiers.

Another step that needs to be investigated is the necessity to choose the correct features to represent the samples in order to use them as an input to a classifier. It seems that there is a variety of possible features (descriptors) that an image can be transformed. The majority of the literature uses the SVM approach with some kind of differentiation of Histogram of Oriented Gradients (HOG) features [2], [3], [8], [10], [11], [13], [1], [17]. In [10] an SVM detector is used and a comparison of HOG features in RGB and greyscale images is performed and is concluded that HOG in grayscale images results in higher precision rates. In [13] different ways to extract HOG features are being investigated and compared using SVM with different kernels. In [17] it is mentioned that HOG features gave excellent results with SVM on a challenging problem like pedestrian detection but the whole process was quite slow. Different kinds of HOG are being evaluated also in speed terms. It is concluded that with more powerful descriptors the potential to imprint patterns training samples more efficiently.

1.2 Potential Challenges

1.2.1 Countries and Differences

Initially the desired algorithm should be able to detect and classify traffic signs from different countries, thus it is necessary to present some of the important aspects. The main problem is that traffic signs differ among countries. Despite the fact that

Figure 1.1: Speed limits Europe

they look similar in human vision and more or less could represent the same thing with different colours, content and shape for computer vision is a real problem.

1.2.2 Different weather and illumination conditions

Since the images are obtained by Google Street View the possibility of having an image in dark scenery is minimized. Generally Google Street View operates with good weather conditions mostly sunny or cloudy but never during rain or snow. There is no point of having images of a road that actually the vision is not clear. There are some cases of images that the rain drops where in the camera's lens but this is not the rule. The obtained images can be with different illumination conditions due to the time of the day and the location in relation with the sun. That can be a potential challenge.

1.2.3 Traffic Signs that are artificially blurred by Google Street View

In many cases the traffic signs (especially the speed limits) appeared blurred and is difficult to determine what actually the sign illustrates. This could be due to the algorithm that Google uses to blur the registration plates of each car in order not to be obvious in public.

1.2.4 Resolution of Google Street View images

The relatively low resolution of Google Street View images makes the classification part difficult especially when the signs appear far in the distance.

1.2.5 Time to train the system to recognize signs

In order to make the system able to recognize traffic signs, the signs must be separated into categories and train the system to recognize each category. The amount of different categories makes the training procedure a time consuming process. Furthermore if samples in each category need to be collected this is also a task that takes time.

1.3 Traffic signs within the scope of the thesis

Since many different countries could be involved in the current project and traffic signs differ substantially among different countries the scope of this thesis is limited to Swedish speed limits since the principals to expand the current project into other signs are going to be the same. Sweden has an extensive and uniform traffic sign system. All signs have predefined standard shapes and colours, according to specifications and use easy-to-understand international pictograms and symbols. Officially, Swedish traffic signs divided into four major categories warning signs, prohibitory signs, mandatory signs and signs giving information. Speed limits belong to the prohibitory signs that can be seen in the following figure.

Figure 1.2: Prohibitory signs

1.4 Application of the current Traffic Sign Recognition system

Application of this master thesis is to find a large amount of traffic signs capable of training a neural network for Volvo's own purposes or to test other traffic sign recognition functions.

2
Theory

In this chapter the basic parts of the related theory will be reviewed. The challenging problem of recognition (detection and classification) of traffic signs in images that have been collected from different real traffic environments is being partitioned and explained in order for the basic concepts used in the current thesis to be grasped. Despite the fact that recognition of signs is a problem that a lot of research has been done since the early days of computer vision, lately by introducing the problem in machine learning framework the development of generic recognition algorithms can still be challenging.

2.1 Object Recognition

Object recognition is a complex sub process of visual perception. Due to the fact that many biological vision systems use this complex process as part of their nature and is being applied instantly, this does not imply that the same occurs also for artificial systems. For instance human brain is able to instantly recognize visual incentives without needing explicit rules even if these incentives are from different point of view or under different illumination. On the other hand an artificial system needs well-pre-defended rules in order to recognize an object. A trivial example would be that an artificial vision system faces huge difficulties to differentiate between a bike a tree and a car, while for a human this is matter of a spontaneous, almost automatic, categorization.
In order to clearly define what is the exact meaning of object recognition an example will be used. The purpose of this thesis is object recognition. Objects to be recognized are traffic signs. The first task is to distinguish a traffic sign from other things that are present in an image but are not objects to be recognized. This first target is called detection. Also it is not satisfiable enough to detect a traffic sign but also it is desired to further categorize this sign. This is referred as the classification. Is it a speed limit sign? If yes what are the numbers on the traffic sign that define the speed limit.

2.2 Object Detection

As mentioned previously, object detection is the first step while classification comes afterwards. In fact, object detection is a special type of classification. Basically in the detection step only two classes are being used. These classes are object(a traffic sign) and no-object(background). Object detection's goal is to return any instances

of the class object in an unknown image. As a consequence detection problem is further subdivided into search and classification problem. A search mechanism provides the classifier with regions of image that are object hypothesis and classifier decides if these hypothesis are objects or not.

2.2.1 The Search Problem

During the detection phase the search problem is defined as follows: What kind of search strategy will be used in order to detect and classify all possible regions within the image. Common methods here are to look within an image either for specific shapes or colours. Also some hybrid methods can be applied.

2.2.1.1 Obtaining Shape Information

During this phase an algorithm is searching for particular shapes within the image. A common approach to do that is to look within an image for edges. Edge detection includes mathematical methods which aim at identifying points within an image in which the image intensity has discontinuities. Also connected component algorithms could be applied in order to see which of the pixels are part of what shape. Once a shape has been extracted from an image a template matching technique, that uses known shapes as templates, can be used to identify the extracted shape.

2.2.1.2 Obtaining Colour Information

Since traffic signs have specific colours then this could be used to identify areas of the image that contain colours of interest. In order to describe colours different colour spaces can be used due to the fact that they are governed by different properties. Colour space is how different colors are being organized while colour model is a mathematical model expressing the way that colors can be represented as triples or quadruples of numbers [25]. A colour model always must be accompanied by a mapping function which indicates how this model is associated with a globally known colour space.
The RGB colour space is one of the most well known colour spaces. Its primary colours Red, Green and Blue can be mixed in order to produce any other desired colour. Thus the value of any colour can be expressed as a vector of RGB elements. Another well known colour space is YIQ. Components I,Q represent colour while component Y represents intensity. Similarly to RGB, it can be described by a vector of one luminance value and two chroming values. This can be seen as a the amount of blue and red in the colour. In a similar manner, YUV colour space can be described.
Another widely applied in computer vision colour space is HSV. HSV stands for Hue, Saturation and Value or it can be seen as HSB for brightness. The main advantage of HSV is that it decouples any intensity information from colour while hue and saturation are colour attributes which can stimulate human perception. A broader explanation of colour spaces and models can be found in [26]. The important fact in order to obtain colour information accurately is to achieve somehow robustness in illumination changes. A robust method that can achieve high detection rates will

be described in the next paragraph.
The log-chromaticity colour space (LCCS) is a two-dimensional color space [12]. It can be described by the logarithm of ratios of colour channels. The illumination invariance of the current colour space can be seen by expressing the intensity of each pixel of an image for a specific colour channel as:

$$\mathrm{I}_c = \sigma \int E(\lambda)S(\lambda)\mathrm{Q}_c(\lambda)\, d\lambda, c \in \{R, G, B\} \tag{2.1}$$

With σ expressing the Lambertian shading, E is the illumination power spectral distribution, S is the surface reflectance function, while Q is the sensor's (camera) sensitivity function. The limits of the integral is the visible spectrum Ω. If illumination is restricted to be Planckian equation 2.1 for a narrow band camera can be transformed to:

$$\mathrm{I}_c = \sigma I k_1 \lambda_c^{-5} e^{\frac{-k_2}{T\lambda_c}} S(\lambda_c)\mathrm{q}_c \tag{2.2}$$

T is the colour temperature and k_1,k_2 constants. Two-band-ratio chromatic components can be formulated by taking the ratios of two bands in respect to another. To remove the nonlinear component the logarithm of the ratios is being used thus from 2.2 the following equation is being obtained,

$$\begin{gathered}\mathrm{P}_c = log(\tfrac{I_c}{I_G}) = log(\tfrac{s_c}{s_G}) + (e_c - e_G)/T, \text{ with} \\ c \in \{R, B\},\ \mathrm{s}_c = k_1 \lambda_c^{-5} S(\lambda_c) q_c \text{ and } e_c = -k_2 \lambda_c\end{gathered} \tag{2.3}$$

while the intensity and shading information are being removed from the equation. As temperature T changes, the 2-vectors will form a straight line in the LCCS, while different surface characteristics will produce different parallel lines. From these lines red color can be obtained by setting a small range threshold and a pixel of the image then can be classified as red if this threshold is fulfilled. Experiments in [12] indicate that this simple rule makes red colour detection robust in different illumination conditions and it can be used as the first step in a detection algorithm. At this point it should be reminded that red colour is the target since speed limits contain an outer ring of red making them easily to be detected.

2.2.1.3 Hybrid Search Methods

Different hybrid methods have been developed combining different information in order to provide accurate locations of traffic signs within an image. In the current thesis the search problem will be solved with a direct brute force approach. An image will be scanned using rectangular windows of different sizes and every window will be filtered for red colour. This will ensure that all the possible locations within the image will be filtered and sent to the classifier to decide if traffic signs exists in this image.

2.2.2 The Classification Problem

Since now the search strategy has been solved all the possible traffic signs must be confirmed. In order to do that they must be encoded in features and then passed in the classification scheme in order to be decided whether they are traffic signs or not. Both these steps are being explained in the next section as well as in section 3.3.7.

2.3 Features

If the target is to make an artificial vision system then everything is observed by this system must be represented by a computer efficient way. Thus measurable properties of an object must be created in order to make the most sufficient and at the same time generic representation of an object. As features lines, edges, colours and other important details that describe an object can be used. The process of defining an object by specific features is called feature extraction. The result of the feature extraction is a vector of real numbers, thus the feature extraction could be easily described as a function that its input is an image and its output is a vector. This could be noted as: $f : Im \mapsto \mathcal{R}$. The more details of an object that are being captured by the features the more powerful the features are, but at the same time capturing more details can be computationally expensive [22]. Hence creating powerful and computationally plausible features has become an indispensable part of object recognition and more in general in pattern recognition. When a pattern is tried to be recognized the key factor for successful results is the features to remain unaffected(invariant) to image transformations like translation, rotation and scaling. Different ways to extract features exists but in the current thesis histogram of oriented gradients method will be used in order to represent an object.

2.3.1 Histogram Of Oriented Gradients (HOG)

The original HOG features were presented for first time by Dalal and Trigs in 2005. HOG descriptor has been used in order to perform person detection. The underlying idea behind HOG is that object shape and appearance can be represented by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions [21]. In order to compute the HOG features of a grayscale image the image is being divided into cells. Cells are small connected regions that their pixels will be used to compute a histogram of gradient directions. These histograms together are characterizing the descriptor. In order for the performance to be improved these cells are combined into overlapping blocks and the histograms obtained from those cells are being normalized by taking into account the intensity of the block. This normalization is necessary in order to make the descriptor invariant to lighting conditions. A step by step implementation of HOG algorithm will follow.

2.3.1.1 Computing the Gradient

Within each cell the magnitude and the orientation of the gradients are being computed. In order to do that the x direction and y direction derivatives must be computed by applying the convolution between the following filter kernels F and the image Im resulting in the gradient vector G.

$$F_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, Im_x = Im * F_x$$

$$\mathrm{F}_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, Im_y = Im * F_y$$

$$\mathrm{G}=\begin{bmatrix} Im_x \\ Im_y \end{bmatrix}$$

The magnitude of the gradient vector is given by: $|G| = \sqrt{Im_x^2 + Im_y^2}$, while the orientation of the gradient is: $\theta = \arctan \frac{Im_y}{Im_x}$.

2.3.1.2 Cell Histograms

Now for each cell a histogram is being created. By creating a histogram all gradients of that cell are being quantized in eight or nine values depending the number of the bins of the histogram. Each pixel that is part of this cell is placing a weighted vote for one bin of the orientation based histogram. This weight of each vote is depended by the magnitude found earlier. The bins of the histogram are evenly cover the orientations from 0° to 180° (unsigned gradient) or 0° to 360° (signed gradient). In the original work [21] found that 9-bin-histogram performs better.

2.3.1.3 Creating Overlapping Blocks

In order to make features invariant to illumination changes the histograms must be normalized. Instead of performing normalization to each histogram separately the neighboring cells are grouped into bigger overlapping blocks. This overlap makes each cell to contribute more than once in the final descriptor. The vector of HOG descriptor is then composed by concatenating the normalized histograms of all the blocks. The blocks can either be of rectangular (R-HOG) or circular (C-HOG) shape.

2.3.1.4 Block Normalization

According to [21] different methods to perform block normalization exist. Let v be the non-normalized vector that contains the histograms of one block, $||v||_k$ be its k-norm ($k = 1, 2$) and $e = constant$ (value of e does not affect the result). Then the normalization can be:

- $L1 - Norm = \frac{v}{||v_k||_1 + e}$

- $L2 - Norm = \frac{v}{\sqrt{||v_k||_2^2 + e^2}}$
- $L1 - Sqrt = \sqrt{\frac{v}{||v_k||_1 + e}}$

2.4 Classification

Once an object of interest is retrieved and its features are extracted it has to be classified by choosing a model that is highly related to its pattern. In machine learning framework classification is the problem of determining in which set of categories, a new instance (observation) belongs to. Classification could be also approached as a way of supervised learning, meaning that during the learning (training) procedure the category (class) of the samples used is known and divided in advance. On the other hand, in the unsupervised procedure the label of the samples used is not known in advance and involves categorizing the data based on other techniques such as similarity or distance. Different ways to classify objects currently exists. The most straight forward supervised approach is a linear classifier.

2.4.1 Support Vector Machine

Known as SVM, support vector machine algorithm originally developed by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. Later on in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik presented a new way that creates nonlinear classifiers by the usage of a kernel trick in order to maximize the hyperplanes [23]. Given a set of training examples that are pre-labeled in two classes during the training phase the algorithm is responsible to build a model that is able to classify a new unseen example into one of the two categories. During that phase SVM represents the training examples as points in a plane and the ultimate target of the training algorithm is to separate the plane linearly in such a way that the two classes are placed in the plane within different locations that have the longest possible distance. Later on, during the classification phase new unseen instances are then mapped into the same plane and according to their position into the plane their class is being determined.

In the current section a brief review of SVMs will be given. Starting by introducing hyperplane classifiers that were the precursor of SVMs and then it will be discussed how the SVMs have been evolved in order to be able to solve problems that are not only linearly separable. Also matters like different kernels and model selection will be examined.

2.4.1.1 Hyperplane Classifiers

Classifiers need training samples in order to create statistical models able to perform pattern recognition. Training data T have the following form.

$$\mathrm{T}=[(\mathrm{x}_1, y_1), \ldots, (x_i, y_i)]$$

$$\text{with: } \mathrm{x}_i \in \mathbb{R}^n, y_i \in \{-1, 1\}$$

where y_i could be either {-1} or {1} (it could be referred also as {0} or {1}) , labeling the class to which the point x_i belongs and x_i is a n-dimensional real vector. For instance a label with value of +1 indicates that the vector is classified to class +1. At this point a function needed of the form: $f(x) = y : \mathbb{R}^n \to \{-1, 1\}$ able to apart from classifying patterns which belong in the training set it should also be able to classify new unseen patterns correctly. Being able to classify both known and unknown patterns is called generalization.

In other words, its aim is to find a hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points x satisfying:

$$w \cdot x + b = 0$$
$$\text{with: } w \in \mathbb{R}^n, b \in \mathbb{R}$$

Also w is n-dimensional real vector that defines a perpendicular direction to the hyperplane and by changing the values of b the hyperplane is able to be moved parallel to its self.

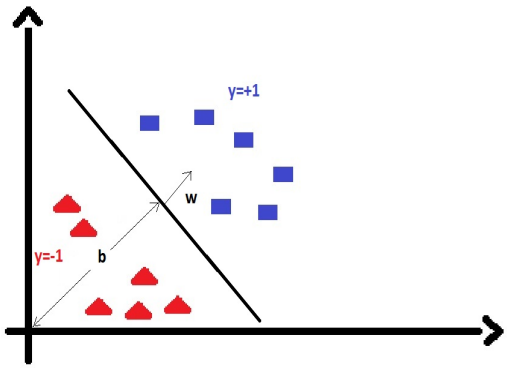

Figure 2.1: Separating hyperplane

In Figure 2.1 training data of two classes separated by one line can be seen. Six 2-dimensional vectors of class -1 are being separated linearly from six 2-dimensional vectors of class +1 satisfying $w \cdot x + b = 0$.

Finding Optimal Hyperplane

Right after the inspection of Figure 2.1 a question that emerges is why this hyperplane has been used to separate the data. In fact there are more than one hyperplanes that could be used to classify the data but a better solution could be implemented. It is desired to divide the hyperplane in such a way that the classes must be as far as possible from each other in order to obtain the best generalization performance. This hyperplane is unique and known as the optimal or the maximal margin hyperplane.
As it can be seen from Figure 2.2 two hyperplanes can be selected in such a way that the data can be separated and there are no points between them, and at the

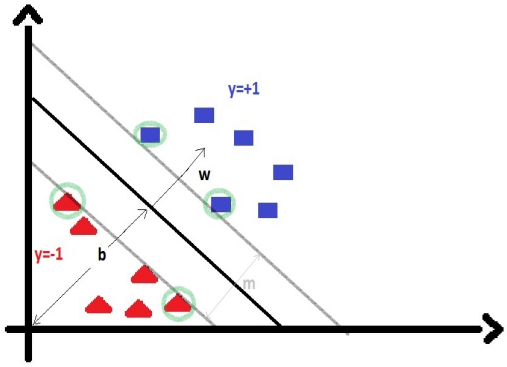

Figure 2.2: Optimal separating hyperplane

same time try to maximize distance between them. Hyperplanes can be described by following equalities:

$$\mathrm{w}{\cdot}x + b = 1,\ w \cdot x + b = -1. \tag{2.4}$$

The distance in between the two hyperplanes is called margin $m = \frac{2}{\|w\|}$ and since the distance must be maximized, then the term $\|w\|$ must be minimized. Also as earlier mentioned no points must be within the margin thus the following constraint must be added.

$$\begin{aligned} w \cdot x_i + b &\geq 1 \qquad \text{for } y_i = 1 \\ w \cdot x_i + b &\leq -1 \qquad \text{for } y_i = -1 \end{aligned} \tag{2.5}$$

This can be combined as:

$$\mathrm{y}_i(w \cdot x_i + b) \geq 1 \quad \text{for all } 1 \leq i \leq n \tag{2.6}$$

Thus the following optimization problem can be formulated.

Minimize:

$$\|w\| \tag{2.7}$$

subject to:

$$\mathrm{y}_i(w \cdot x_i + b) \geq 1 \quad \text{for all } 1 \leq i \leq n \tag{2.8}$$

The solution for a two dimensional problem can be seen in Figure 2.2. For those training points that the inequality becomes equality are called support vectors. Support vectors can be seen in figure 2.2 as the light-green-encircled points that are located exactly on the margin. If for any reason the support vectors are decided to be removed the solution of the problem will change and it will be a different pair of hyperplanes. Also the value $|w \cdot x_i + b|$ can be used as an indication of how far a sample lays from the separating hyperplane.
In order to solve this constraint minimization problem the use of Lagrangian multipliers is essential to simplify the solution of the problem. The Lagrange multipliers are:

a_i, with a_i>0 and $i = 1, \ldots, m$

Also it is difficult to solve the problem since it is highly depended on $\|w\|$, the norm of w, which involves a square root. This can be modified by substituting (2.7) with $\frac{1}{2}\|w\|^2$. The factor $\frac{1}{2}$ will be used for mathematical convenience without any changes in the final solution. The constraints then are multiplied by the Lagrange multipliers and subtracted by the modified objective function that is about to be minimized formulating the Lagrangian (L):

$$\mathrm{L}= \tfrac{1}{2}\|w\|^2 - \textstyle\sum_{i=1}^{m} \alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^{m} \alpha_i \tag{2.9}$$

This problem can be solved by standard quadratic programming techniques, thus the solution of the problem can be obtained by determining the saddle points of the equation (2.9).

$$\frac{\partial L}{\partial b} = \textstyle\sum_{i=1}^{m} \alpha_i \mathrm{y}_i = 0 \tag{2.10}$$

$$\frac{\partial L}{\partial b} = \mathrm{w} \cdot \textstyle\sum_{i=1}^{m} \alpha_i \mathrm{y}_i x_i = 0 \tag{2.11}$$

$$w = \mathrm{w} \cdot \textstyle\sum_{i=1}^{m} \alpha_i \mathrm{y}_i x_i \tag{2.12}$$

Using the fact that $\|w\|^2 = w^T \cdot w$ and by substituting (2.10) and (2.12) into the Lagrangian (2.9) the dual formulation of it can be obtained.

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.13}$$

An important property of (2.13) is that if (2.13) is constraint by $\alpha \geq 0$ the maximum value of (2.13) can be found by some values of the w, α, b and at the same time if (2.13) is constraint by $y_i(w \cdot x_i + b) - 1 \geq 0$ the minimum value of (2.13) can be found by same values of the w, α, b as before.
Thus the dual problem can be now reformulated to:
Maximize:

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.14}$$

subject to:

$$\textstyle\sum_{i=1}^{m} \alpha_i \mathrm{y}_i = 0 \tag{2.15}$$

Given the parameter w, b can be calculated by b =y_i- w·x and the solution of the problem of classifying a new unseen data point would be the computation of :

$$sign(\sum_{i=1}^{m} \alpha_i y_i x_i \cdot x + b) \tag{2.16}$$

where x is the new unseen data point and x_i is a support vector.

Soft Margin Classifier

Until now, the idea of linearly separating data was demonstrated. Real world problems as it is known are not always linear, meaning that errors and noise can be included in the data. In this section the case of having data that are difficult to be classified by the previous approach will be discussed.

The idea of having a modified maximum margin that allows mislabeled data was proposed by Corinna Cortes and Vladimir N. Vapnik in 1995 [23]. The Soft Margin method is covering the case of; if there exists no hyperplane that can divide perfectly the data, Cortes' and Vapnik's method is able find a hyperplane that divides the data as good as possible. This will be done by introducing ξ_i, new non-negative slack variables that are able to measure the degree of the misclassified data x_i. The concept of the Soft Margin can be seen in Figure 2.3. Some points are light-red-encircled and they appear to be on the margin or on the wrong side of the margin. This is due to the fact that the Soft Margin algorithm was not able to divide the data perfectly.

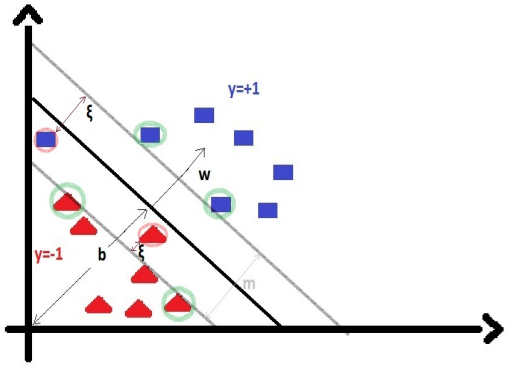

Figure 2.3: The soft margin

The degree of misclassified data is being measured by their distance from the margin. For instance a non zero value of ξ_i allows x_i to meet the margin specifications by a cost. This cost is proportional of ξ_i. Thus the constraint problem referred with (2.7) and (2.8) can be relaxed by adding the slack variable ξ_i and can be rewritten as follows:
Minimize:

$$\tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i = 0 \tag{2.17}$$

subject to:

$$\begin{gathered} \mathrm{y}_i(w \cdot x_i + b) \geq 1 - \xi_i \\ \text{for all } 1 \leq i \leq n \text{ and } \xi_i \geq 0 \end{gathered} \tag{2.18}$$

That is the Generalized Optimal Separable Hyperplane problem. The term C is a regularization parameter with the purpose to restrain the amount of intervene between maximizing the margin and minimizing the training error. By setting large values to C the algorithm may over-fit the training data, while by setting a a small C the margin becomes 'harder' thus more difficult to accept misclassified training

data. Similarly as before the dual form of the problem can be obtained by the use of Lagrangian multipliers reformulating as follows:
Maximize:

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{2.19}$$

subject to:

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \tag{2.20}$$

The advantageous reformulation of the linear penalty function is that the slack variables ξ_i are not present in the the dual problem, and the constant C appearing only as an additional upper bound constraint on the Lagrange multipliers.

The Kernel Trick

There are some cases of training data that are not linearly separable. Even by applying the Soft Margin method discussed previously the results will be poor(almost random). An example of such a data set can be seen in Figure 2.4 and is obvious that can not be separated linearly.

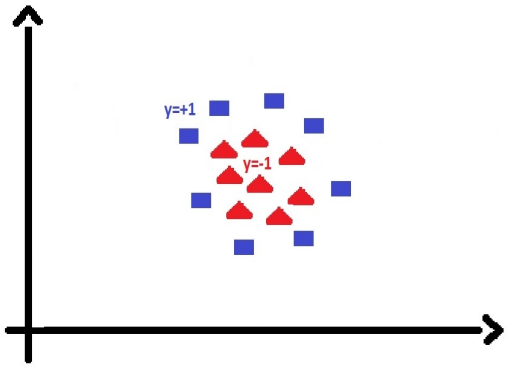

Figure 2.4: Example of non linearly separable data

In most of the real word applications the data are not linearly separable. Thus a way to classify these kind of data must be found. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik [23] introduced a new method to create classifiers for nonlinear data by applying a kernel trick to maximum-margin hyperplanes. So the basic idea is to create a classifier to unequivocally discover decision boundaries (hyperplanes) with arbitrary shapes, not just linear. This has an immediate impact in the optimization problem and as a consequence higher number of iteration must be used to produce such a solution, thus more time needed.
The kernel idea was born from the fact that classifiers for N-dimensional training data find an N-1-dimensional separating boundary. Thus by trying to project the N-dimensional training data to another space (called feature space), usually higher-than-N dimensional space, could lead to linearly separable data. This projection can be seen in Figure 2.5.

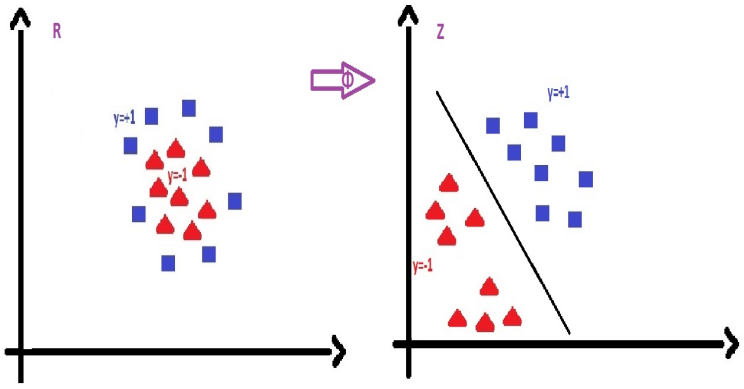

Figure 2.5: Projection to feature space

If having an input data $x \in \mathbb{R}^n$, let $\phi(x)$ representing the corresponding feature data with Φ being a mapping function from $\mathbb{R}^n$ to the feature space $\mathbb{Z}$, noted as $\Phi : \mathbb{R}^n \to \mathbb{Z}$. Since the classifier uses the training data into form of dot products $(x_i \cdot x_j)$ in $\mathbb{R}^n$ this will be mapped as well as $\phi(x_i) \cdot \phi(x_j)$ in $\mathbb{Z}$. The problem is that this projection is computationally expensive.
By applying a simple kernel function, $K(x_i, x_j)$ in $\mathbb{R}^n$ space it results directly in the dot product of the $\mathbb{Z}$ space, while still remaining in $\mathbb{R}^n$, thus avoiding the mapping $\Phi(x)$ which can be sometimes impossible to compute especially if $\mathbb{Z}$ space is multi-dimensional. Using a kernel function the possibility to implicitly transform the data set into a higher dimensional space is given with a minimal computational effort, depending the complexity of the kernel. Finally using the kernel trick the feasible option of separating nonlinear data is given within the existing support vector machine framework.

Kernel Selection

Different kernel functions exist. Among others the most popular are the polynomial, the radial basis and the sigmoid kernel. Also there is always the option ,if none of this kernels works well enough with the data set, to create a customize kernel. There is no particular method to choose the appropriate kernel thus the brute force method could be used and different kernels can be tested.

The Polynomial Kernel: It results in a polynomial classifier of user defined degree p. The kernel has the form of : $K(x_i, x_j) = (1 + x_i \cdot x_j)^p$.

Gaussian Radial Basis Function: Known as RBF kernel. $K(x_i, x_j) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2$.

The Sigmoid Kernel: The kernel has the form of : $K(x_i, x_j) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + \theta)$, for some $\kappa > 0$ and $\theta < 0$. Both κ and θ are user defined values.

Model Selection

Support vector machines have only two parameters to be set by the user. The first parameter is the C that was explained before and the choice of the kernel function. The majority of the kernel functions have a set of parameters that need to be set by the user as well. For instance if the RBF kernel will be chosen as the appropriate kernel the user has to set the parameter C and σ. Setting the values on these parameters is not known before hand and different values stem in different results. The goal in this case is to find such values of the parameters that the classifier is able to accurately predict the class of the new unseen data. An approach on this would be to just divide all the available data into a training set and into a validation set and then train and validate the performance of the classifier appropriately by continuously changing values of C and σ. The performance of the classifier will be reflected by the values C and σ on the validation set, which in fact will be unseen data.
A more improved version for estimating how the results of a statistical analysis will generalize to an independent data set called k-fold-cross-validation (CV). In this method the training set is divided into k subsets of equal (or almost equal) size. In succession on subset is tested using the classifier that is previously trained in the rest $k-1$ subsets. The cross-validation is then repeated k times (folds). Then the k results from the iterations are being averaged in order to produce one single estimation of how the classifier performs in new unseen data. The advantage of this method is that all available samples are being used for both training and validation. A commonly used number for the parameter k is the 10-fold cross-validation, but generally parameter k can be a free parameter.
In order to make a good model selection for the classifier the grid-parameter-search of C and σ combined with the k-fold-cross-validation method can be used [24]. Values of C and σ are being set to a 2-dimensional grid and for each possible pair of values the classifier is validated with cross validation. If possible a deeper investigation of the values of the grid could be also used for more accurate results.

2.4.1.2 Evaluating the performance of a classifier

The performance of a classifier can be measured with different ways. In order to evaluate the performance the validation set is being used. Validation set is a subset of the original training set containing samples from all the classes with their classes known. By already knowing their classes a comparison is being made with the prediction that the classifier results. The outcome of this comparison can be translated in a matrix with the correctly and incorrectly classified samples of all classes, named confusion matrix. Also it can be found in the literature as contingency table or error matrix.
The confusion matrix for a binary classification problem can be seen in Figure 2.6.

The values of Figure 2.6 can be defined as:

- True positive (TP): The classifier correctly predicted the positive class.
- True Negative (TN): The classifier correctly predicted the negative class.

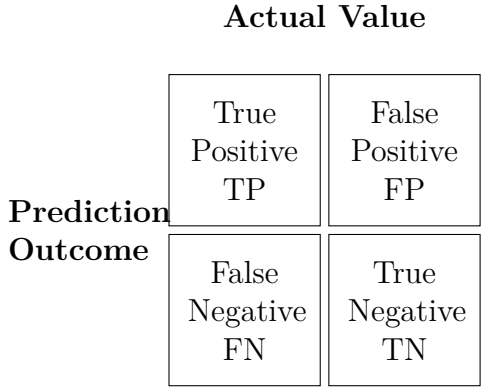

Figure 2.6: Confusion matrix for a binary classification problem

- False Positive (FP): The classifier predicted the class positive and the class was negative, also known as Type I error.
- False Negative (FN): The classifier predicted the class negative and the class was positive, also known as Type II error.

And the total number of samples: TS=TP+TN+FP+FN. A simple way to evaluate the performance of the classifier by the confusion matrix is that it is desired to have the highest possible number in the main diagonal of the confusion matrix. In the main diagonal of the confusion matrix is where the true positive samples and the true negative samples lay. From the combination of those values different performance indicators can be produced such as.

- Accuracy, AC=(TP+TN)/TS: How often the prediction is correct.
- Misclassification Rate(Error Rate), MR=(FP+FN)/TS= 1-AC: How often the prediction is wrong.
- True Positive Rate(Sensitivity, Recall), TPR= TP/Actual Positive: When the class is positive, how often the classifier predicts it as positive.
- False Positive Rate, FPR=FP/Actual Negative: When the class is negative, how often the classifier predicts it as positive.
- Specificity, S=TN/Actual Negative=1-FPR: When the class is negative, how often the classifier predicts it as negative.
- Precision, P= TP/predicted Positive: When the classifier predicts it as positive, how often this prediction is correct.

A product of those indicators that can be used to evaluate the performance of the system is the Receiver Operating Characteristic (ROC) curve. The curve is created by plotting the TPR against the FPR at various threshold settings.

2.4.1.3 Multi-Class Classification

The classification problem discussed in the previous sections was involving an SVM classifier that was able to distinguish between two classes (binary classification). In reality, depending the application it could be more than two classes that samples must be classified.

A multiclass SVM is able to handle more than two classes. In order to have multiclass SVMs the single multiclass problem must be reduced into multiple binary

classification problems. Common methods to do that reduction are the one-versus-one and the one-versus-all approach. During prediction, both methods can suffer from ambiguities and a new sample is possible to be classified more than once, thus the distance between the hyperplane and the sample must be taken into account for each classifier.
In the one-vs-one reduction, if the multi-class problem has K different classes, $\frac{K(K-1)}{2}$ binary classifiers must be trained. Each of the classifiers is trained with a pair of two classes and is responsible to classify samples between this two classes. During the prediction time (when the classifier will have to classify a new unseen sample) a sample will be evaluated by all the binary classifiers and all of them will return their results and the results will be accumulated(voting). The class with the most results is the final class of the sample.
On the other hand, in one-vs-all strategy one classifier is being trained for each class, eventually having K classifiers. Each classifier is being trained with positive samples from one specific class and as negative samples the rest of the classes' samples are being used. During the prediction time one at least classifier will classify the sample as positive thus its class is the final class of the sample.

3
Methods

In this chapter the overview of the system will be given. All the following steps will be explained by applying the theory covered in the chapter 2 and explanatory figures will be used as well as describing processes and algorithms. The structure of this chapter will be divided in tree parts in order to cover the process step by step. The process is summarized on the right of the Figure 3.14.
In the first part the image acquisition procedure will be explained, meaning how the system obtains the appropriate images containing road scenery in order to feed the detection part. In the second part the detection procedure will be discussed, meaning how the system was trained in order to detect traffic signs. In the final part the classification procedure will be examined, explaining how the system was trained in order to fully identify traffic signs.

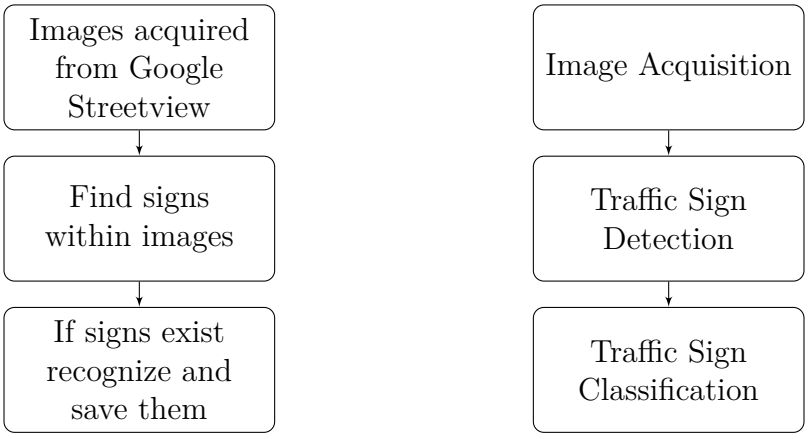

Figure 3.1: A basic overview of the TSR function

3.1 Image acquisition

This section will cover the details of how the image acquisition algorithm was carried out. More specifically the procedure of how from a pair of geographical coordinates the system obtains an image contains road scenery and feeds that image into the detector. The basic architecture can be seen in Figure 3.2. The system takes input coordinates from Line Router does some processing by a Matlab script and then passes all the coordinates to a Python function in order for the images to be downloaded.

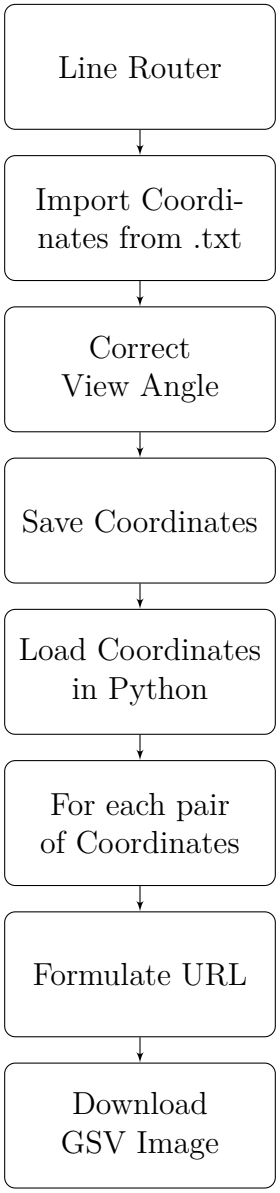

Figure 3.2: The image acquisition architecture

3.1.1 Line Router

Line Router is a web browser application that uses the interface and the capabilities of Google Maps in order to let the user set a route worldwide.The user just by setting the starting point and the final destination dynamically in a Google-Maps-style interface is able to obtain the desired route. By choosing so, the application is able to download all the geographical coordinates that assemble this specific route and save them in the computer's hard disk drive in a text file format.

3.1.2 Matlab Processing Script

This script looks for the coordinates saved from Line Router in a text file. It imports them in Matlab as two matrices. One for the latitude and one for the longitude.

At this point it should be mentioned that before the development of Line Router the coordinates received were all the possible coordinates of the route with really high precision. The problem with that was that for a number (that number was not stable) of sequential coordinates the same image was appearing in Google Street View (GSV). That problem was overcome by developing a script that calculates the geographical distance between coordinates and rejects the ones that have small distance between them. That resulted in having one unique image for each pair of coordinates.

Since GSV apart from coordinates needs also an angle of view to be set as an input this needed to be taken care. A script was developed that actually calculates the difference in the angle between two sequential coordinates for points in the earth, expressed as longitude and latitude, and uses this to determine the angle of view of each download request.

Since the necessary corrections have been done the script saves a .mat file containing the longitude, latitude and the angle of view for each image that is about to be downloaded. Then it initializes the Python script in order to start downloading the images.

3.1.3 Python Processing Script

This script is written in the programming language Python and it is responsible to download all the images for the given coordinates. First it loads the .mat file that was obtained from the subsection 3.1.2. For every coordinate the script is creating a unique Uniform Resource Locator (URL) which is consisted from one longitude one latitude and the heading angle. Also it contains the size of the image in pixels, the field of view angle, pitch angle and a personal key in order for Google to identify who is downloading the images. By using this URL the script requests and downloads each image. The script can easily be expanded to download images of one location with 360 degrees field of view. Due to that key there is a daily limit of the images that the script is able to download. Thus if the script downloads 25.000 images within the same day it pauses the execution of the code and it waits for 24 hours before to resume the execution. The script saves every image in the hard disc drive of the computer for further process.

3.2 Traffic Sign Detection

In this section the technicalities about the detection part will be embodied. Matters like how exactly the training samples were obtained, in what features were encoded to and how the training procedure was executed in order for the detector to be created and make the target of this thesis feasible to detect traffic signs. Furthermore it will be discussed how the detector discovers the signs within each image obtained by the procedure described in the section 3.1.3.

3.2.1 Training Samples

Training samples are images provided to a learning method (detector) with ulterior target this learning method to learn how to categorize objects. Thus training samples for each and every category are needed for the process of training. In the case of the current thesis the detector uses only two classes. Traffic signs is the positive class (Positive training samples) and all the other things that someone can see in a street are constituting the negative class (Negative training samples).

3.2.1.1 Positive Training Samples

As discussed in the section 1.1.1.3 obtaining training samples can be a time consuming process. In order to avoid that an initial thought about obtaining samples was to use samples from GTSB make a detector and then use GSV images obtained in Sweden to collect Swedish traffic signs. That was possible since for speed limits the signs in Germany and Sweden have negligible differences for designing an initial detector. That detector was used in the early stages of the thesis but it was proven insufficient due to the high number of miss-detections.

Then the idea of using computer generated training samples was born. Taking into account another project in Volvo Cars that were using computer generated signs, to test some functions for their own other purposes, with some extra process, training samples for the purpose of this thesis could be obtained. Initially traffic signs are created in one rendered image with a complete black background. They are located in the left and the right side of the image and they start from big and obvious signs that all the details can be seen and they end up in small and distant traffic signs hard to discriminate details.

So since these images have no background (it is black) it must be added to make them look more pragmatical. Images obtained by GSV were checked in order not to have any other real traffic signs within the image and were used to become the background of the rendered image. The rendered image was superimposed on the image obtained by GSV. Since the exact location of the sign within the rendered image is priory known the only thing left to be done is to crop the signs from the new merged image. At this point it should be mentioned that by adding few pixels of background around the sign was assumed to improve detection. Thus it was decided that around 12% of the total surface of the sign would be enough background to be added. The result of this process can be seen in Figures 3.3 - 3.5. In Figure 3.3 an image obtained by Google Street View outside the conference site Gothia Towers, before entering in the Korsvägen square. As it can be seen no obvious traffic signs are in the image. In Figure 3.4 is exactly the same image but now a computer created traffic sign is located on the street in the right side. In fact it happens to be on the top of a bike street. In Figure 3.5 the traffic sign is cropped from the rest of the image and is ready to be used as a training sample.

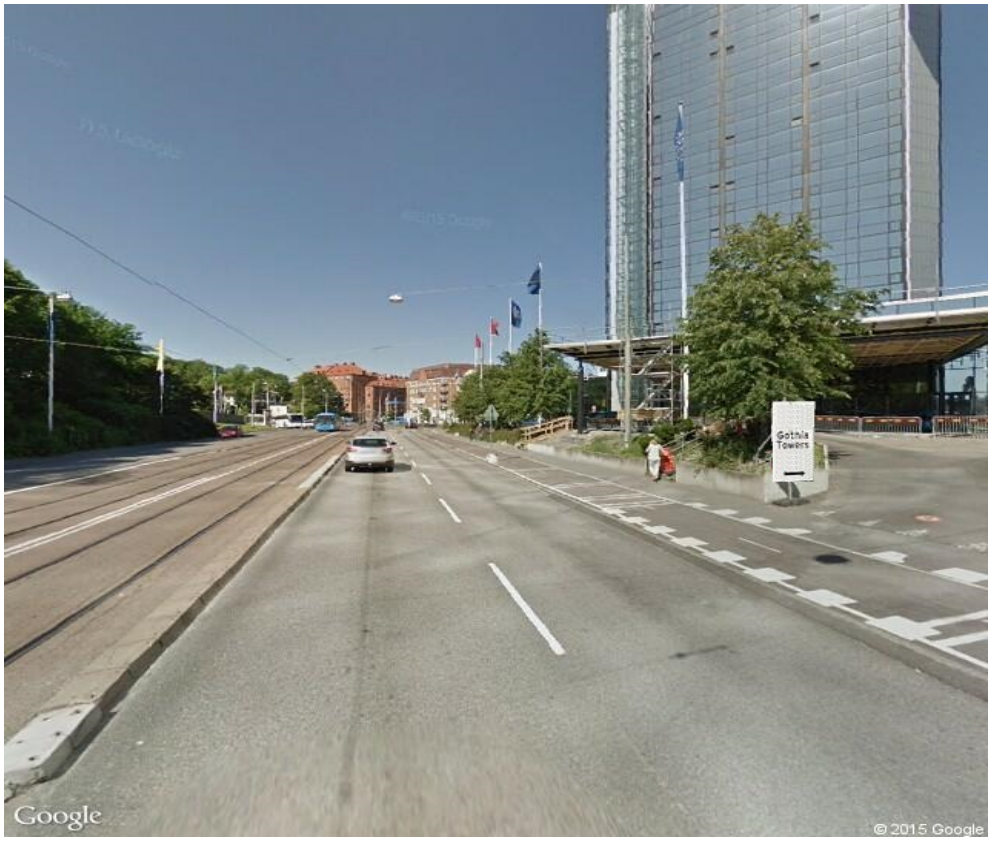

Figure 3.3: Original image by GSV

Figure 3.4: Merged image

Figure 3.5: Computer generated training sample

A script in Matlab was created in order to automate the whole process. The result of this scripted process was a sufficient amount of a variety of computer generated samples with different sizes that they are ready to be used as positive training samples. In Figure 3.6 a variety of positive training samples that used to train the detector can be seen.

Figure 3.6: Examples of positive training samples

3.2.1.2 Negative Training Samples

Those 'other things' referred in the section 3.2.1.2 will be used as negative training samples. By that, it is meant objects that someone can see in a GSV image excluding traffic signs. These can be fractions of road, surrounding buildings, cars ,trees, sky etc. In order to obtain negative training samples a script was made that it partitions a GSV image that has no traffic signs in smaller images of randomized size that could contain anything. Also all the miss-detections from the initial detector trained with

GTSB referred in the section 3.2.1.2 were used as well. Some examples of negative training data can be seen in Figure 3.7.

Figure 3.7: Examples of negative training samples

3.2.2 Features

Since the problem of collecting training samples was solved the problem of feature encoding emerges. This actually means, how each sample can be encoded in order to be at the same time generic and keep most of its details. Since Histogram of Oriented Gradients (HOG) found to be among the most powerful features it was decided that all the related images should be encoded to these features. In order to find the appropriate parameters a subset of all samples was picked to be used as training sample, a detector was trained and validated. The parameters which have resulted in the highest classification accuracy were kept as the HOG parameters.

3.2.3 Training Procedure

Now that the samples were taken care of, they must be transformed into HOG features. All the available samples were transformed into HOG feature vectors in order to be used by the SVM. The SVM detector now needs to be trained. This is one of the most time consuming processes. This is due to the reason that the appropriate SVM values of S and C must be found. It was decided that a direct approach would be used. A grid of parameters would be searched in order to find the pair of parameters in which the smallest classification error would occur.
At the same time for every pair of parameters that are about to be evaluated in order to make the training procedure more accurate the five-fold-cross-validation approach was also used. This means that, for each pair of values and for five times, the training set was randomly fragmented into five subsets of approximately same size. Each time the $\frac{4}{5}$ of the initial training set was actually used for training while the rest $\frac{1}{5}$ was used to test the detector (validation).

Table 3.1: The training set of detector

Training set with rendered positive samples and real negative samples											
	Positive Class									Negative Class	
Class	30	40	50	60	70	80	90	100	110	Negative	Total
Samples	313	313	313	313	313	313	313	313	313	20701	2817

Since the above training procedure fragments the initial training set that apart from negative samples it also contains positive computer generated training and validation samples, a secondary set of validation data needed to be created. In order to do that the negative validation data for that set were obtained with the same process described in section 3.2.1.2. The positive validation data set now was assembled by a large number of real traffic signs found in GSV images in the area of Stockholm and Goteborg. This change in the secondary validation set was decided since the purpose of the detector is to detect signs from GSV images thus in order to validate its performance real traffic signs must be used. The performance of the detector with real traffic signs will be discussed in the section 4.1.1. In order to automate this procedure a script was created that for a given pair of SVM parameters calculates the sensitivity, the specificity and the error of the detector with the five fold cross-validation method and then it calculates the same values with the secondary validation set. This is done for every values within the grid. After all grid parameters have been evaluated the best parameters must be picked. The criterion to choose the best C and S values comes out of the fact that the detector must have the highest sensitivity and specificity and the lowest error in the five fold cross validation method and the same applies for the secondary validation set.

3.2.4 Detection Of Traffic Signs

Since the detector is now trained to distinguish between traffic signs and background a way to constantly feed the detector with traffic sign hypothesis must be found.

In order to do that the sliding window approach will be used. An overview of the detection procedure can be seen in Figure 3.8.

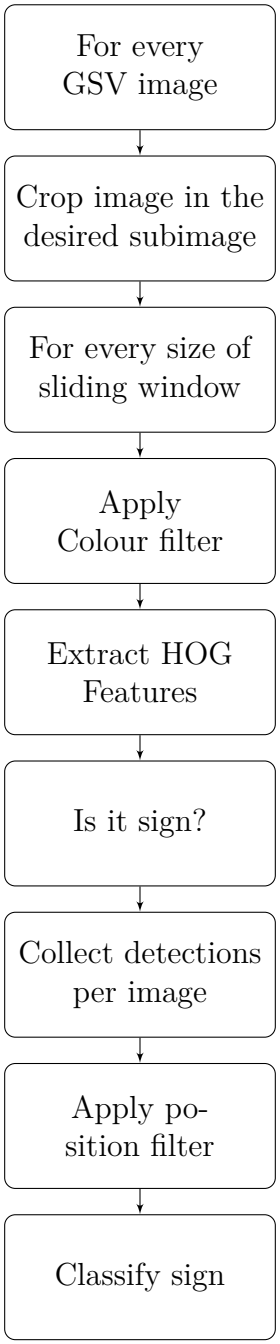

Figure 3.8: The detection process

Since this approach is known to be computationally expensive it was decided to exclude some regions of the image since a traffic sign is not expected to be found there. The excluded regions are regions on the top and the bottom of the image. The rest of the image is being scanned extensively from top to the bottom and from left to right with sliding windows of different sizes. This is due to the fact that the traffic signs are expected to appear in the image on different sizes. They can

be really small, meaning that they are far in the horizon and bigger meaning that they can be close to the car. In Figure 3.9 the whole GSV image can be seen with the cropping boundaries while, in Figure 3.10 the actual area which is going to be scanned by the multiple sized sliding windows is illustrated.

Figure 3.9: Example of the processing area

Figure 3.10: Sliding windows of different size

Each and every position of the sliding window within the image can be a traffic sign location hypothesis. In order to verify that each sliding window is being filtered

and if red colour is found then the HOG features are being extracted and fed to the SVM detector. The SVM detector is trained to recognize the pattern of a traffic sign thus is able to distinguish the traffic signs from the background. There are few cases that the detector miss-detects some hypothesis to traffic signs but actually they are not. This is due to the fact that the pattern in those images could be the same like a traffic sign. In order to overcome this issue the priory mentioned colour filter is being applied in order to detect red colour within that hypothesis thus the number of miss detections is being reduced. If the filter finds red colour then the image is eligible to proceed for further processing. Also it was observed that multiple detections can occur around a traffic sign. This is due to the fact that while the window slides over a sign the whole sign can be appeared in more than one windows but not in all of them the sign appears in the center of the image. In order to suppress the rest of the detections and keep the one that the sign appears to the center, a position filter was applied.

3.2.4.1 Colour Filter

In order to reduce the processing time and the number of miss-detections every image passes a preliminary check in order to be determined if there is any red colour in that image. In order to do that each image is being transformed in the LCCS colour space mentioned in section 2.2.1.2. A threshold was set by testing with images contain signs and with images with no signs.

3.2.4.2 Position Filter

All the detections within the same image that are able to pass the colour filter and the detector have to be checked again in order to avoid having multiple detections of the same sign within the same image. To achieve that, all the detections are being grouped according to their position within the image in order to eventually have groups of detections that are close to each other. The algorithm is able to handle cases of having more than one traffic signs within the same GSV image. Then the image from the group with the highest detection confidence is being kept while the rest are being discarded. At this point it should be mentioned that the detector has higher confidence to images similar to the ones is being trained with, meaning that the sign must be preferably placed to the center of the sliding window.

3.3 Traffic Sign Classification

In this section the details of the classification algorithm will be discussed. This means that right after the signs have been detected it should be decided what exactly traffic sign has been obtained. The same order will be followed as before starting from obtaining samples for training the classifier until the explanation of the whole classification procedure. Furthermore since the Optical Character Recognition is part of the classification algorithm its further implementation will be discussed.

3.3.1 Training Samples

Training samples were obtained with the same procedure described in the section 3.2.1.2. The only difference is that for each obtained sign a number of effects were applied in order to increase the training set and also create samples that possibly could meet expectations of real word. In Figure 3.11 on the top left corner is the rendered sign while the rest of the signs are variations of it. It should be noted that now the negative signs have been changed and they are not background images like in the detector. They are again traffic signs. A further explanation is covered in the section 3.3.3.

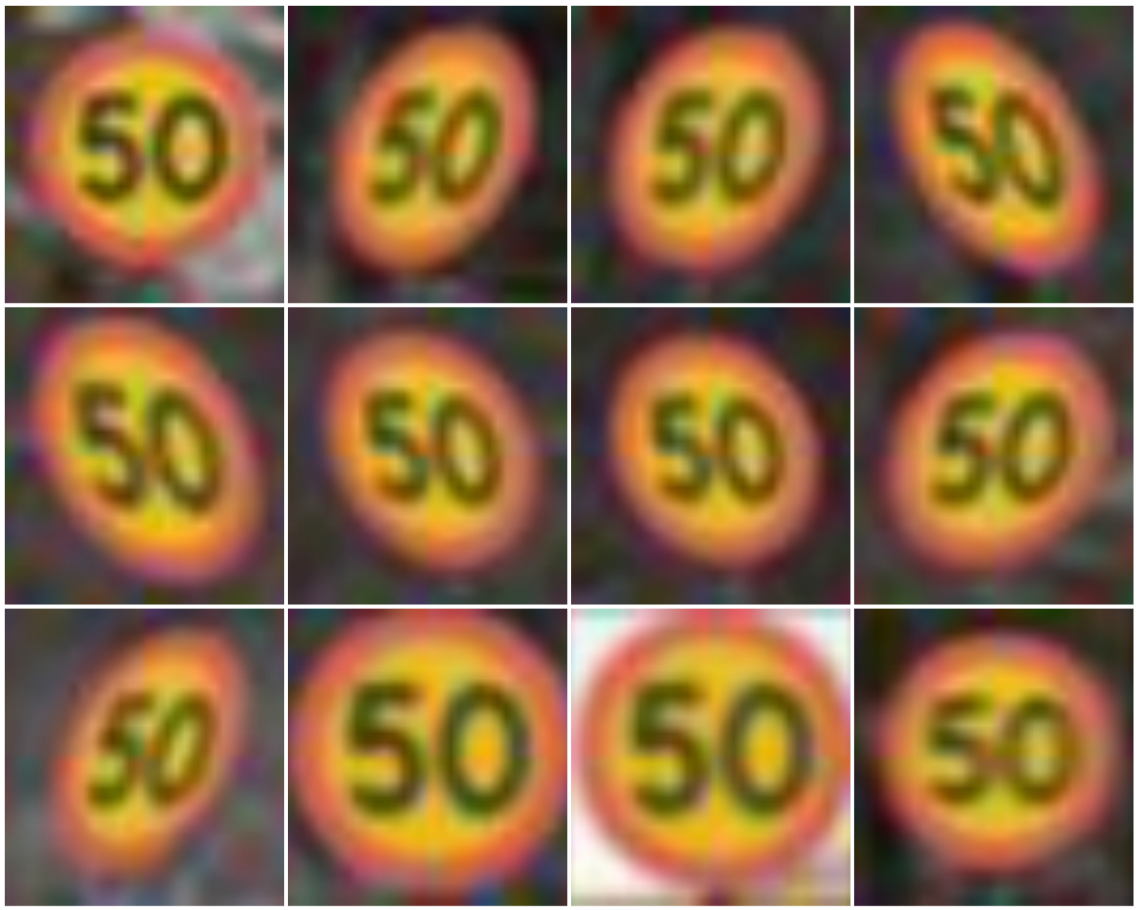

Figure 3.11: Effects of signs

3.3.2 Features

Since exactly the same learning method has been decided to be used, the features were extracted with exactly the same technique used before in section 3.2.2

3.3.3 Training Procedure

Since it was decided the one-against-one SVM classification approach to be used thirty six pairs of classifiers were trained exactly like the training procedure used for

the detector, mentioned before in the section 3.2.3, with the only difference that the procedure was repeated thirty six times. The large number of the classifiers comes due to the fact that they are nine different categories and by applying the formula in section 2.4.1.3 found that thirty six different classifiers must be used. Figure 3.2 shows all the different categories and the classifiers. Despite the large number of the classifiers the process is faster than before since the training samples that define the positive and the negative categories are fewer than the detector's case. Furthermore the classes of the classifier can be later expanded if needed.

Table 3.2: Pairs of classifiers One-against-One approach for 9 classes

The 36 pairs of classifiers									
Class	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**
30	0	1	2	3	4	5	6	7	8
40		0	9	10	11	12	13	14	15
50			0	16	17	18	19	20	21
60				0	22	23	24	25	26
70					0	27	28	29	30
80						0	31	32	33
90							0	34	35
100								0	36
110									0

For instance the first classifier uses as positive training samples signs of speed limit 30 and as negative training samples signs of speed limit 40. It is not important which of them will be positive or negative samples but it should be kept noticed. The training samples are computer generated samples and the validation samples are part of the secondary set of real signs. Likewise applies to the rest of the classifiers until all of them are trained and validated separately.

Table 3.3: The training set per class of rendered classifier

Training set with rendered samples with transformations									
Class	30	40	50	60	70	80	90	100	110
Rendered Samples	3452	3451	3454	3452	3452	3453	3449	3452	3455
Total number of training samples:31070									

3.3.4 How Classifier Works

The result of the classifier is based in the fact that all the classifiers have to make a decision and 'vote' in order to determine the result. Each traffic sign will pass

from all the thirty six classifiers and each classifier will classify it in one of the two categories that is trained with. Also each classifier will return how confident is for the current decision, meaning that how far the classifier classified the sample from its decision border. When all the thirty six classifiers have vote, the class with the highest number of votes is the result of the whole classification module. In case there is a draw between the classes the class with the highest confidence wins the draw.

3.3.5 Optical Character Recognition

After the detection part the traffic sign passes also from the Optical Character Recognition (OCR) function. This part is responsible to extract the text from the traffic sign. In the current thesis it will be used as a secondary classification module that a speed limit sign will be inserted and the OCR function will recognize the number in the speed limit. The reason of this secondary classification is that the main classifier will classify every input in one of the classes that is trained with. For instance if a speed limit of 120 will be used as an input the classifier will classify it in something among 30 to 110 but never 120. Technically speaking this could be solved by creating training samples of the speed limit 120 in the computer and then add one more class in the classification procedure. But sometimes unpredictable signs can be seen in the street and there must be a way to instantly classify them. One of this unpredictable signs can be seen in Figure 3.12.

Figure 3.12: A real traffic sign found in Stockholm

This traffic sign was accidentally found in a GSV image in Stockholm. That was due to the fact that they were reconstructing the road and the driver should maintain a low speed. Signs like that could be classified by the OCR function. Before feeding the sign to the OCR a pre process must occur and after the OCR gives the result this result must also be processed in order to achieve the desired goal of recognizing speed limits.

3.3.5.1 Pre-process for Optical Character Recognition

In order to achieve the highest possible results by using the OCR the text that exists in the traffic signs must be sequestered. The best way to do that is to create a black and white image that the text will be white and the rest of the image will be black. The developed method initially tries to isolate the text from the rest of the image by applying the colour filter discussed in section 3.2.4.1. By applying the filter only the red colour elliptic ring is visible and nothing else. So by excluding the ring and what ever is outside only the part encircled by the ring is being kept and by making the image black and white the text is being isolated by the rest of the image and

is ready to become the input to the OCR function. An overview of the pre-process can be seen in Figure 3.13.

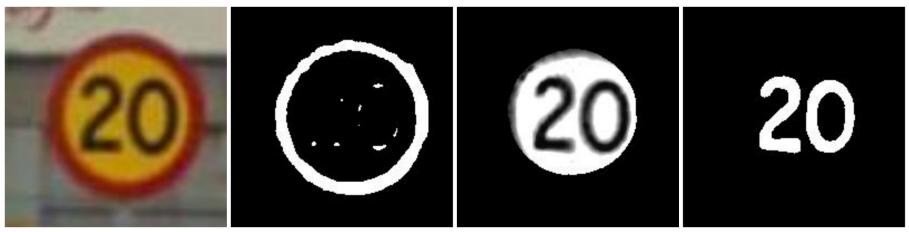

Figure 3.13: Pre-process for OCR

3.3.5.2 Post-process for Optical Character Recognition

Since the scope of the thesis does not contain information panels some signs can be excluded from the OCR classification procedure and as a consequence not to be interested in alphabetical characters but only in numbers. This gives a small opportunity to improve the OCR accuracy by excluding letters. Having also letters to the OCR can give confusing results since there a is high possibility of confusing some numbers with letters. A typical example is confusing '0' with 'O' or '2' with 'Z' and so on. In order to overcome possible confusions occur by the OCR all real signs were passed from the OCR in order to figure out possible repeatable confusions and then by creating a post processing script acting like a look up table and correcting the confusions the problem was solved.

3.3.6 Merging Results of Classifier and OCR to determine the final result

In order to have one final result taking into consideration both SVM classifier and OCR, both results need to be investigated further. In order to determine the final result from images with speed limits the digits were extracted with the same method described in section 3.3.5.1. The digits from the speed limits were separated in order to form smaller images of digits between zero and nine. Few samples from each digit were collected to formulate the templates of each digit. When the results between the SVM classifier and the OCR module are different then a template correlation method is being applied. For each of the two results two different images are being reconstructed from the template digits mentioned before. For instance if for any reason the result of the SVM classifier is 100 and the result of the OCR module is 110 then these two results are being reconstructed and correlated with the original image obtained by the detector. The image which is highly correlated with the original one then this is the final result of the classification. The main reason that the results are being reconstructed by single digits is due to the fact that from the OCR module is expected to have any possible combination between zero and nine as a result.

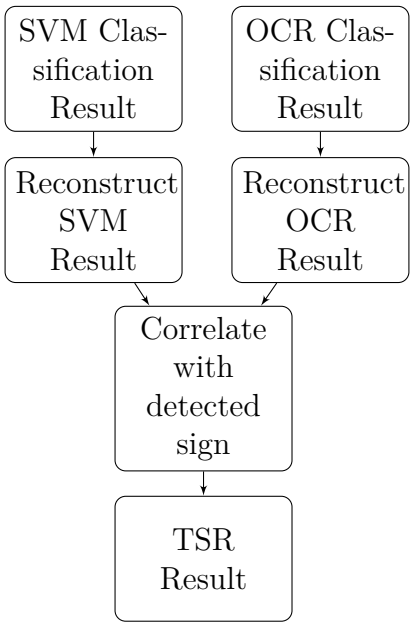

Figure 3.14: In order to obtain the final TSR result both SVM and OCR results are being reconstructed and correlated with the detected sign.

3.3.7 Training Of A Hybrid Classifier

It was further decided that a new hybrid classifier is necessary to be trained. Its hybrid properties stem from the fact that both real samples collected from expeditions and synthetically generated signs with transformations were used as the training set. The training set of the hybrid classifier can be seen in Table 3.4.

Table 3.4: The training set per class of hybrid classifier

Training set with real samples and rendered samples with transformations									
Class	30	40	50	60	70	80	90	100	110
Rendered Samples	3452	3451	3454	3452	3452	3453	3449	3452	3455
Real Samples	266	44	293	44	250	92	129	90	83
Total	3718	3495	3747	3496	3702	3545	3578	3542	3538
Rendered samples:31070 Real samples:1291									
Total number of training samples: 32361									

Since all the collected real signs were used for training the performance of the hybrid classifier at that moment was evaluated only in rendered signs. The purpose of this hybrid classifier is to compare its performance with the classifier mentioned in section 3.3.3. This will be feasible when a new sufficiently large validation set will be gathered and the performances of the two classifiers can be compared on that validation set. In this way it can be decided if it is better to use a hybrid training set or a training set by only rendered signs.

4

Results

The current traffic sign recognition function is able to search for traffic signs in a given root described by a set of coordinates. The function downloads all the images of the designated root and then asks the user to give a name to that specific virtual expedition. Apart from the images all the necessary information are being stored under that name. These information include the .txt file from Line Router, the .mat file of the coordinates and a useful image that has the desired expedition on the map. Furthermore statistical results can be derived for the extracted signs, while there is also the option of creating a video of the whole root. These additional files could be used in case the user would like to check additional details of the expedition. When the processing of each expedition is done then all the signs are being accumulated and sorted in a folder that contains all the signs that have been collected by all expeditions and a file that indicates what expeditions have participated in the accumulated results. The current system is able to classify speed limits but it can be easily expanded in more traffic signs. For each expedition the processing time for each image varies from few milliseconds to few minutes depending on how many traffic sign hypothesis could possible be detected. The whole TSR function was tested within small expeditions in the area of Goteborg and Stockholm in order to increase the number of traffic signs in the validation set. The primary validation set can be summarized in the Table 4.1

4.1 Preliminary Results

4.1.1 Detection Results

The detector was trained and cross validated with the procedure described in section 3.2.3. It was validated with rendered signs in this procedure but the detector's performance is needed to be evaluated with real traffic signs. Thus it was validated with 1337 real speed-limit signs and 77556 negative samples collected gradually during the whole time of the current project. The main objective of the detector is to maximize the amount of true positive samples, even if this could result in few false positive samples. The validation set can be seen in the following table.

Table 4.1: The validation set of detector

Validation set with real positive samples and real negative samples													
	Positive Class												Negative Class
Class	15	20	30	40	50	60	70	80	90	100	110	120	Negative
Samples	6	12	266	44	293	44	250	92	129	90	83	28	77556
Total signs:1337 Total negative samples:77556													
Total number of validation samples:78893													

The confusion matrix of the detector can be summarized in the Table 4.2.

Table 4.2: Confusion matrix of detector

	Actual 1	Actual 0
Predicted 1	1332	93
Predicted 0	5	77463

In the columns of the confusion matrix are the actual values of the samples while in the rows the classification result can be seen. The detector successfully detected 1332 out of 1337 total speed limits while, failed to detect 5 signs. By the confusion matrix the performance indicators on Table 4.3 can be extracted.

Table 4.3: Performance indicators of detector

Overall accuracy: 99.8 %

Class	Precision	Recall
Sign	0.934	0.996
No Sign	0.999	0.998

4.1.2 Classification Results

4.1.2.1 SVM classifier trained with rendered samples

The one versus one classifier was trained and validated with rendered signs as it is mentioned in section 3.3.3. As it is also applied with the detector the true performance must be evaluated with real signs. From the validation set of 1337 signs, the 1291 belong to the classes that the classifier is trained to recognize. The validation samples can be seen in Table 4.4

The classifier alone, has achieved $94,1\%$ accuracy and the confusion matrix of the classifier can be seen in the Table 4.5. The number of total samples for each class

Table 4.4: Validation set per class of rendered classifier

Validation set with real samples									
Class	30	40	50	60	70	80	90	100	110
Real Samples	266	44	293	44	250	92	129	90	83
Total number of validation samples:1291									

could be also related to how often a speed limit of this class can be met in the real world. For instance speed limits of 30, 50 and 70 is more likely to be found in the real world compare to the rest categories thus the validation samples between the categories are not equally distributed.

Table 4.5: Confusion matrix of rendered classifier

	Validation set of 1291 real speed limits								
Actual Class →	30	40	50	60	70	80	90	100	110
Predicted 30	261	0	6	0	0	9	3	7	2
Class ↓ 40	0	44	2	0	0	0	0	0	0
50	1	0	281	0	0	1	2	1	0
60	0	0	0	44	3	1	0	0	0
70	3	0	0	0	234	0	0	0	4
80	0	0	0	0	1	79	2	1	1
90	0	0	1	0	2	0	120	0	0
100	1	0	3	0	0	2	1	80	0
110	0	0	0	0	10	0	1	1	76

From the Table 4.5 the following performance indicators can be derived :

Table 4.6: Performance indicators of rendered classifier

Overall accuracy: 94.5 %		
Class	Precision	Recall
30	0.906	0.982
40	0.956	1.000
50	0.982	0.959
60	0.916	1.000
70	0.971	0.936
80	0.940	0.859
90	0.976	0.930
100	0.919	0.888
110	0.863	0.915

4.1.2.2 SVM classifier trained with hybrid samples

The validation set of Table 4.4 was used as part of the training set of the hybrid classifier thus more traffic signs needed to be collected in order to evaluate the performance of the current classifier. This will be discussed in section 4.2.1.2.

4.1.2.3 OCR classifier

For the optical character recognition the publicly available software called Tessaract has been used as a black box approach. The OCR classification module was validated with 1291 real speed limits. The validation samples that used for the OCR are the same that are used to validate the SVM classifier trained with rendered samples and can be seen in Table 4.4. For comparison reasons the accuracy of the OCR classifier found to be 83.4 % for that validation set.

4.1.2.4 TSR Results

In order to obtain the final TSR result the system was tested with the validation set of Table 4.4. The final result of the current traffic sign recognition function takes into account both SVM classifier trained with rendered samples and OCR. The system was tested with 1291 real speed limits and reaches the overall accuracy of $95,1\%$. This small elevation of the final result compared to the classifier's result stems from the combination of the result of the support vector machine classifier with the optical character recognition result. The way that the two results are being combined as it is described in section 3.3.6 actually manage to slightly increase the final result.

4.2 Results

It was decided to further test the current traffic sign recognition function in a virtual expedition from Goteborg to Jokkmokk. It was decided to use this specific route since a winter test truck of Volvo is in Jokkmokk and many test drivers follow this route many times per year. The route is 1.379km and around sixteen hours of driving. This results in about 8GB data thus 156.562 images to be scanned for signs. The total number if signs found during this expedition is 895 speed limits and can be seen in Table 4.7.

Table 4.7: Detected signs in expedition Goteborg to Jokkmokk

Class	30	40	50	60	70	80	90	100	110
Real Samples	13	5	53	40	165	134	119	243	123
Total number of speed limits: 895									

4.2.1 Classification Results

The Table 4.7 will be used as ground truth data to evaluate and compare the performance of the two SVM classifiers, the optical character recognition module and the overall system performance.

4.2.1.1 SVM classifier trained with rendered samples

The confusion matrix derived from the SVM classifier trained with rendered samples over signs from Table 4.7 can be seen in Table 4.8

Table 4.8: Confusion matrix of rendered classifier Goteborg to Jokkmokk

	Validation set of 895 real speed limits								
Actual Class →	30	40	50	60	70	80	90	100	110
Predicted 30	13	0	1	1	8	15	4	8	3
Class ↓ 40	0	5	1	2	3	2	0	0	0
50	0	0	51	1	0	4	7	2	0
60	0	0	0	36	0	1	0	0	1
70	0	0	0	0	133	0	1	0	16
80	0	0	0	0	6	108	3	5	0
90	0	0	0	0	1	0	101	0	1
100	0	0	0	0	3	4	2	226	3
110	0	0	0	0	11	0	1	2	99

From the Table 4.8 the following performance indicators can be derived:

Table 4.9: Performance indicators of rendered classifier

Overall accuracy: 86.2%		
Class	Precision	Recall
30	0.245	1.000
40	0.384	1.000
50	0.784	0.963
60	0.947	0.900
70	0.886	0.806
80	0.885	0.806
90	0.980	0.848
100	0.949	0.930
110	0.876	0.804

4.2.1.2 SVM classifier trained with hybrid samples

The confusion matrix derived from the SVM classifier trained with hybrid samples over signs from Table 4.7 can be seen in Table 4.10

Table 4.10: Confusion matrix of hybrid classifier Goteborg to Jokkmokk

	Validation set of 895 real speed limits								
Actual Class →	30	40	50	60	70	80	90	100	110
Predicted 30	5	0	0	0	1	8	0	1	0
Class ↓ 40	0	3	0	3	1	0	0	0	0
50	4	2	53	0	0	1	0	4	0
60	0	0	0	33	0	1	0	1	0
70	0	0	0	1	158	0	1	0	4
80	4	0	0	0	0	122	0	1	0
90	0	0	0	0	0	1	117	0	0
100	0	0	0	1	0	1	0	230	0
110	0	0	0	2	5	0	1	6	119

From the Table 4.10 the following performance indicators can be derived:

Table 4.11: Performance indicators of hybrid classifier

Overall accuracy: 93.8%		
Class	Precision	Recall
30	0.333	0.384
40	0.428	0.600
50	0.828	1.000
60	0.942	0.825
70	0.963	0.957
80	0.961	0.910
90	0.991	0.983
100	0.991	0.946
110	0.894	0.967

4.2.1.3 OCR classifier

The OCR classification module has achieved 50% accuracy for the given set of Table 4.7.

4.2.1.4 TSR Results

The final results of the TSR module is the combination of SVM classifier and the OCR module. OCR result doesnt depend on the SVM classifier. The TSR classification results for the set of Table 4.7 are being summarized in the Table 4.12.

Table 4.12: Overall traffic sign recognition accuracies

Classifier	SVM Accuracy	OCR Accuracy	TSR Accuracy
Rendered	0.862	0.500	0.866
Hybrid	0.938	0.500	0.890

5

Conclusion and future work

In the current thesis a complete method based on support vector machines was developed able to recognize Swedish speed limit road signs from Google Street View images. In order to download the images from Google Street View a script was implemented in Python programming language. The different SVMs were trained with synthetically generated signs that have been processed in such a way to look pragmatical. For the SVM implementation the Matlab functions 'svmtrain' and 'svmclassify' have been used without any further usage of ready-made toolboxes, while HOG features were calculated with the external matlab function 'hogcalculator'. The TSR function was complemented with an OCR module. For the Optical Character Recognition the publicly available software 'Tesseract' has been used. The TSR function was evaluated with a sufficient amount of real data in order to assess its performance. It was further validated with a long virtual expedition from Goteborg to Jokkmokk.

The TSR results of the expedition are being summarized in the Table 4.12. For the detector it can not be concluded if was able to detect all the signs of the expedition since all the images must be inspected one by one for their signs and then verify if the detector found them all. But according to the preliminary results of Table 4.2 since the detector has such a high accuracy it is expected to be able to detect a large amount of signs within the images. All the miss-detections from the expeditions can be used to complement the negative training set so the detector can be retrained and avoid those miss-detections next time.

For the SVM classification part, It should be mentioned that the amount of speed limits found may be insufficient to obtain an accurate classification result since some of the classes have relatively low number of signs. Despite that, if the performances of rendered classifier and hybrid classifier can be compared, according to the Table 4.12 it can be concluded that the hybrid classifier is more accurate than the rendered classifier. If there are no available real signs to train a classifier then rendered signs can be used as training samples. That classifier can be used as a basic tool to obtain real signs with some miss-classifications expected. The real signs can be used to train either a hybrid classifier or a classifier that has only real training samples. It would be useful to train three classifiers' one rendered, one hybrid and one only with real samples and compare their performances over the same validation set. Since the current classifier is able to recognize speed limits between thirty to hundred-ten then it must also be expanded to recognize more speed limits. Furthermore since the pattern in speed limits is similar to some of the rest of the prohibitory signs

some miss-classifications can occur. This can also be solved by expanding the categories of the current classifier. Since the classifier is trained with the one against one approach it is easy to expand the categories without discarding the current one.

The current accuracy of the hybrid classifier should be investigated concerning if it is possible to be increased. This could be done by either re-investigating the number of necessary real training signs or the number of the HOG features on the samples. It should also be considered to apply some dimensionality reduction technique. In [10] it is mentioned that with a a well chosen smaller feature vector the accuracy of an SVM could be increased. If a dimensionality reduction were capable to increase the classification accuracy, then it should be applied also in the detector as well.
This has also a major impact on the speed of the algorithm. Having training samples of high dimensionality leads to much higher training times and considering the fact that the algorithm is written in Matlab, which is widely known that is slightly slower than C programming language leads to relatively high processing time.

The OCR module is the least accurate as expected. Despite the fact that 'Tesseract' is a powerful tool in this project it is much depended on image processing and apparently is not always robust. It should be always used as a complementary classification module. Furthermore the way that the SVM classification result and the OCR result are being combined in order to derive the final TSR result should be investigated. The purpose of this combination should be to use OCR result in order to elevate the SVM result, if possible. During the preliminary tests correlating the SVM result, the OCR result with the detected sign seemed a promising way since the correlation result was never lower than the svm classification result. According to Table 4.12 TSR accuracy is the same or even lower than the SVM accuracy and it is something worth investigating further.

Another important factor that downgrades the accuracy of the TSR function is the relatively low resolution of the GSV images. Signs that appear small in the images are a major source of miss-classifications. Technically this is not a big issue since the traffic sign that is currently far will come closer to a later frame and it will be easy to classify it.

Finally in order to speed up the time of searching for signs within an expedition the concept of tracking should be considered. Since the frames are sequential it would be easier to know where to look for a sign within the next image and that will save a lot of time.

Bibliography

[1] Keren Fu, Irene Y. H. Gu *"Recognition of Chinese Traffic Signs from Street Views", Chalmers Technical Report No. R004/2015, 42 pages, Dept. of Signals and Systems, Chalmers University of Technology, Sweden, Jan 2015.*

[2] Fatin Zaklouta, Bogdan Stanciulescu. *Real-time traffic sign recognition using spatially weighted HOG trees.*

[3] Greenhalgh J. , Mirmehdi M. *Real-Time Detection and Recognition of Road Traffic Signs.*

[4] S. Lafuente-Arroyo, P. García-Díaz, F.J. Acevedo-Rodríguez, P. Gil-Jiménez, S. Maldonado-Bascón. *Traffic sign classification invariant to rotation using support vector machines.*

[5] Floros G, Kyritsis K, Potamianos G. *Database and baseline system for detecting degraded traffic signs in urban environments.*

[6] Jesmin F. Khan, Sharif M. A. Bhuiyan, and Reza R. Adhami. *Image Segmentation and Shape Analysis for Road-Sign Detection.*

[7] Yixin Chen, Yi Xie, Yulin Wang. *Detection and Recognition of Traffic Signs Based on HSV Vision Model and Shape features.*

[8] Vahid Balali, Elizabeth Depwe, Mani Golparvar-Fard. *Multi-class Traffic Sign Detection and Classification Using Google Street View Images.*

[9] Xian-Zhong Han, Chen Chen, Ke-Jian Wang, Yingchun Yuan, and Yulong Song. *Study on Detection and Localization Algorithm of Traffic Signs from Natural Scenes.*

[10] Fatin Zaklouta, Bogdan Stanciulescu. *Real-time traffic sign recognition in three stages.*

[11] Haojie Li, Fuming Sun, Lijuan Liu, Ling Wang. *A novel traffic sign detection method via color segmentation and robust shape matching.*

[12] Bishesh Khanal, Sharib Ali and Desire Sidibe. *Robust road signs segmentation in color images.*

[13] Ye Sun. *Research on Traffic Sign Classification Algorithm Based on SVM*

[14] Fabien Moutarde, Alexandre Bargeton, Anne Herbin, and Lowik Chanussot *Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system.*

[15] A. Gonzalez, L.M. Bergasa, J. Javier Yebes and J. Almazan *Text Recognition on Traffic Panels from Street-level Imagery*

[16] Minetto R., Thome N., Cord M., Stolfi J., Precioso F., Guyomard J. Leite N.J. *Text detection and recognition in urban scenes.*

[17] Gary Overett, Lars Petersson *Large Scale Sign Detection using HOG Feature Variants.*

[18] Mammeri A.,Khiari E.-H., Boukerche A. *Road-Sign Text Recognition Architecture for Intelligent Transportation Systems*
[19] The German Traffic Sign Detection Benchmark,
`http://benchmark.ini.rub.de/?section=homesubsection=about`
[20] NICTA RoadScene DataBase,
`http://www.nicta.com.au/`
[21] Navneet Dalal, Bill Triggs *Histograms of Oriented Gradients for Human Detection*
[22] Escalera S., Baró X., Pujol O., Vitrià J., Radeva P. *Traffic-Sign Recognition Systems*
[23] Support vector machine, Wikipedia the free encyclopedia
`https://en.wikipedia.org/wiki/Support`$_v$*ector*$_m$*achine/*
[24] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin *A Practical Guide to Support Vector Classification*
[25] Color space, Wikipedia the free encyclopedia
`https://en.wikipedia.org/wiki/Color`$_s$*pace/*
[26] Milan Sonka, Vaclav Hlavac, Roger Boyle *Image Processing Analysis and Machine Vision*