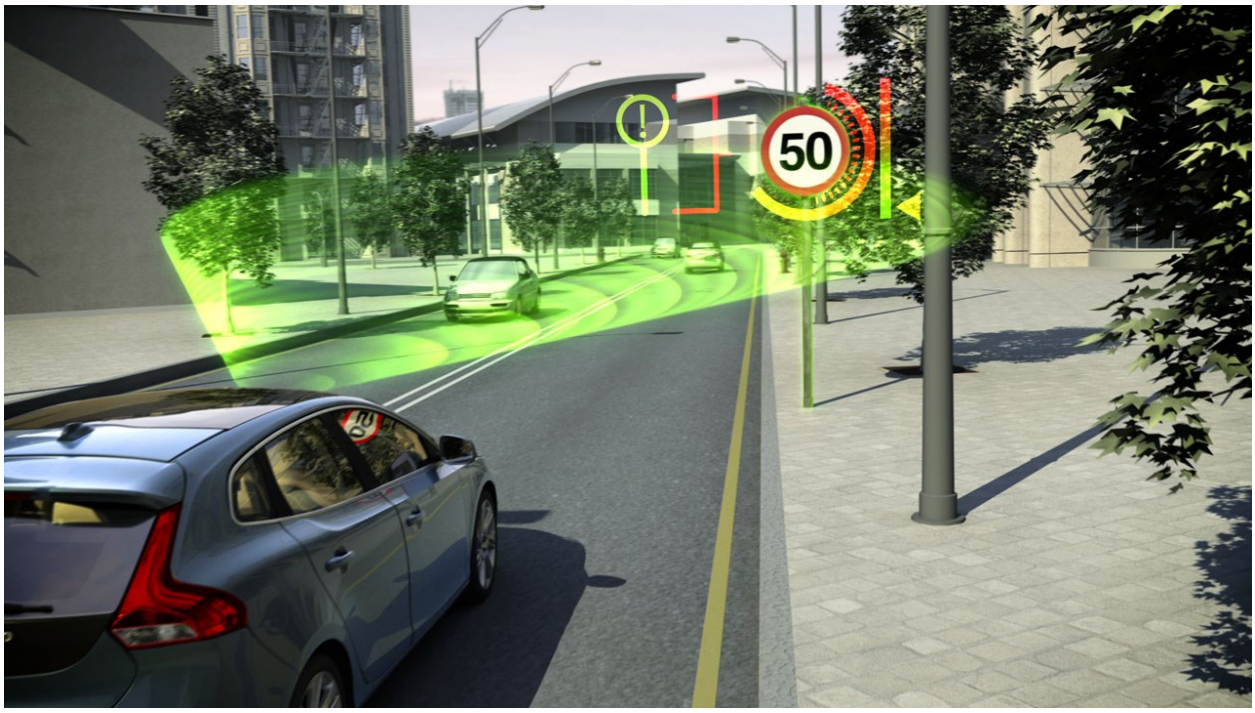
CHALMERS
UNIVERSITY OF TECHNOLOGY



Automatic Offline Annotation of Road Sign Information

Developing a validation tool for speed limit data

Master's thesis in Systems, Control and Mechatronics

NIKLAS EWALDSSON
ANTON PETTERSSON

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Automatic Offline Annotation of Road Sign Information

Developing a validation tool for speed limit data

NIKLAS EWALDSSON
ANTON PETTERSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
Chalmers University of Technology
Gothenburg, Sweden 2022

Automatic Offline Annotation of Road Sign Information
Developing a validation tool for speed limit data
NIKLAS EWALDSSON
ANTON PETTERSSON

Supervisor: Jibin Ou, Volvo Cars
Supervisor: Nima Hajiabdolrahim, Department of Electrical Engineering
Examiner: Erik Ström, Department of Electrical Engineering

Master's Thesis 2022
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image showing the Road Sign Information function in a Volvo V40

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Automatic Offline Annotation of Road Sign Information
Developing a validation tool for speed limit data
NIKLAS EWALDSSON
ANTON PETTERSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In modern cars, Intelligent Speed Assistance (ISA) is becoming increasingly common. The purpose of an ISA system is to advise the driver of the current speed limit on the road. These systems do not work flawlessly, and in 2022 the European Union will implement requirements regarding their correctness. Volvo Cars currently uses manually annotated speed limit data to verify the correctness of their ISA system. This work aims to investigate whether speed limit data from Volvo Cars can be automatically annotated through a combination of test drive data and offline data sources, thus removing the need of manual annotations. The offline model runs in a simulation environment using pre-collected data from test drives. Two data sources are used by the online ISA system in the test cars: speed sign detections from the camera system and speed limit data from the digital map supplier Google Maps.

In this work, two offline data sources are investigated. One based on speed limit data from the digital map supplier Here Technologies, and the other based on a neural network for detecting speed signs in videos collected by the test car. To improve the performance further, speed signs close to off-ramps are filtered out using an off-ramp detection system. The reason for filtering out speed signs close to off-ramps is because they might belong to adjacent roads and can still be detected by the camera system, leading to an incorrect speed limit prediction. To account for variations in data source quality between different countries, tuning parameters are introduced in the offline model.

This work shows that by fusing data from the test car with an offline digital map, a promising model for offline annotation of speed limits can be developed. The offline model was evaluated on log files that make up 2526 kilometers of test driving. Compared to the ISA system in the car which attained a correctness of 69.8%, the offline model attained a higher correctness of 84.2%. The correctness is measured as the distance with the correct annotated speed limit, divided by the total distance driven. The tuning parameter method makes the model adapt well to log data from different countries, outperforming the ISA system in five out of the six countries log data was collected for. The high correctness and the adaptability of the offline model makes it a suitable choice to propose speed limit annotations for test drives, compared to manually annotating data.

Keywords: Dempster-Shafer theory, computer vision, digital maps, sensor fusion, machine learning, advanced driver assistance systems

Acknowledgements

First and foremost, we would like to thank our supervisor Jibin Ou at Volvo Cars for being involved in the process from the beginning to the end, contributing with valuable suggestions for continuous improvements and for being an exceptional mentor during our time as Master's thesis students at Volvo Cars. We would also like to thank our supervisor at Chalmers, Nima Hajiabdolrahim, for giving useful insights on the language and structure of the report. We would like to thank our examiner Erik Ström for support and guidance during our thesis work. We would also like to show our gratitude to our families, especially Lisa Lundgren and Johanna Lindén for valuable feedback on the report. Last but not least we would like to thank Elizabeth Peetso for helping us announce our thesis presentation.

Niklas Ewaldsson, Gothenburg, June 2022
Anton Pettersson, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAS	Advanced Driver-Assistance Systems
ADASIS	Advanced Driver-Assistance Systems Interface Specifications
AHP	ADAS Horizon Provider
AHR	ADAS Horizon Reconstructor
CAN	Controller Area Network
DS	Dempster-Shafer
EH	Electronic Horizon
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
IMU	Internal Measurement Unit
ISA	Intelligent Speed Assist
IoU	Intersection over Union
LiDAR	Light Detection And Ranging
MPP	Most Probable Path
PRN	Pseudo-Random Noise
R-CNN	Region Based Convolutional Neural Networks
RSI	Road Sign Information
YOLO	You Only Look Once

Nomenclature

Below is the nomenclature of the sets, parameters, and variables that have been used throughout this thesis.

Sets

θ	Frame of discernment (all possible speed limits)
2^θ	Power set of θ (all possible combinations of the elements in θ)
$\emptyset$	Empty set
x, y, z	Propositions

Parameters

λ_{coord}	Coordinate loss weight parameter
λ_{noobj}	Object loss weight parameter
κ_t	Curvature threshold value.
α_{cam}	Camera reliability tuning parameter.
α_{Google}	Google map reliability tuning parameter.
α_{Here}	Here map reliability tuning parameter.
$\alpha_{off-ramp}$	Off-ramp detection confidence gain tuning parameter.

Variables

TP_D	True positive distance.
m_i	Probability mass value.
k	Conflict factor.
L	Loss function.
(x_i, y_i)	Center of predicted bounding box j in grid cell i.
$(\hat{x_i}, \hat{y_i})$	Center of ground truth bounding box j in grid cell i.

(w_i, h_i)	Width of predicted bounding box j in grid cell i.
$(\hat{w}_i, \hat{h}_i)$	Height of ground truth bounding box j in grid cell i.
C_i	Confidence score of predicted bounding box j in grid cell i.
$\hat{C}_i$	Confidence score of ground truth bounding box j in grid cell i.
$p_i(c)$	Predicted conditional class probability of object class c in grid cell i.
$\hat{p}_i(c)$	True conditional class probability of object class c in grid cell i.
$\mathbb{1}_{ij}^{obj}$	Binary variable in loss function when there is an object in grid cell i in bounding box j.
$\mathbb{1}_{ij}^{noobj}$	Binary variable in loss function when there is not an object in grid cell i in bounding box j.
κ	Curvature.
$\kappa_{right}, \kappa_{left}$	Curvature of left and right road edge clothoid.
$\gamma(t)$	Parametrization of clothoid by parameter t.
x_t	Maximal distance the clothoid model is valid.
$R_p(t)$	Pseudorange.
$t_r(T_2)$	Time when GNSS signal is received.
$t^s(T_1)$	Time when GNSS signal was transmitted.
m_{Google}	Probability mass values based on speed limit data from Google Maps.
m_{Here}	Probability mass values based on speed limit data from Here Map.
m_{cam}	Probability mass values based on speed limit detection from camera.
$m_{Google,cam}$	Fused probability mass value from Google Maps and camera.
$m_{Google,cam,Here}$	Fused probability mass value from Google Maps, camera and Here map.
s_{Google}	Speed limit reading from Google Maps.
s_{Here}	Speed limit reading from Here map.
$s_{cam,i}$	Speed limit reading i from speed sign detection by camera.
$c_{cam,i}$	Speed limit reading confidence i from speed sign detection by camera.
s_{be}	Best estimated true speed limit.

Contents

List of Figures

List of Tables

1
Introduction

Car manufacturers have traditionally tried to make their vehicles as safe as possible in case of an accident. In recent years, driving assistance technology has evolved to prevent accidents from happening in the first place. Consequently, the driver support system Intelligent Speed Assistance (ISA) is now becoming increasingly common in new cars. The role of an ISA system is to advise the driver of the speed limit on the road. A study conducted by the Norwegian Institute for Transport Economics (TØI) in 2014 compared driving support systems regarding driver safety and found ISA to be the most effective one [1].

The importance of improving road safety cannot be overstated. According to the World Health Organization, about 1.3 million people die each year as a result of road traffic injuries [2]. In the EU alone, it is estimated that 115 lives are lost every day due to road traffic injuries [3], causing an unimaginable tragedy. It has been shown that there is a strong connection between the behavior of the driver and the risk of accident [4], where disobeying the speed limit is an evident risk behavior. ISA can provide the driver with the current speed limit of the road, thus making the driver more likely to follow it.

In 2022 the European Union will make ISA a mandatory function on all new cars sold in the EU and set a number of requirements for these systems [5]. One of these requirements is that over a distance driven, the ISA system needs to show the correct speed limit for 90% of that distance. To measure this, the metric *true positive distance* (TP_D) is used, defined as the distance with correct annotated speed limit (d_{correct}) divided by the total distance (d_{total}) driven [5].

$$TP_D = \frac{d_{\text{correct}}}{d_{\text{total}}} \tag{1.1}$$

For Volvo Cars, these criteria need to be met by their ISA system, called the Road Sign Information (RSI) function. Currently, the true positive distance is measured by using manually annotated speed limit data collected during test drives. Manually annotating data is a labor-intensive and low efficient process that cannot be done efficiently at a large scale. If this process could be automated, it would save both time and resources. Therefore, this work aims to investigate whether speed limit data can be automatically annotated through a combination of test drive data and offline data sources.

1.1 Background

The RSI system uses cameras for speed sign recognition and the GPS to obtain speed limit data from a digital map supplier. These two data sources have their respective strengths and weaknesses. The reliability of the camera depends on good lighting conditions, the presence of speed signs and a well-functioning sign detection algorithm in order to make accurate predictions. The digital map speed limit data depends on the GPS signals not being blocked or masked to obtain the correct vehicle position [6]. Even when map data is obtained, it is not faultless. For example, during road work, the speed limit might be decreased temporarily, which is not always corrected by the digital map supplier. To counter these weaknesses, the two data sources are fused in order to make the best possible estimation of the correct speed limit. The RSI function, interchangeably referred to as the *onboard* model since it runs in the car, is limited by the computational power in the car.

To evaluate the performance of the RSI function, manually annotated speed limit data is currently used. By processing the data collected by the test car offline, a model for automatically annotating the speed limits could be built which essentially removes the constraint regarding computational power. Running the model offline also enables the use of additional data sources that are not available in the car. This model will be referred to as the *offline* model, since it does not run in the car. Data from all time steps can also be used, unlike in the RSI which must run in real-time. Road geometry and sign positions, which can be modelled using data from the forward-looking camera in the car, are two potentially useful additional data sources [7], [8]. This data can be used to generate estimates of the relevant speed sign positions and distinguish them from signs belonging to adjacent roads. In addition, more map data sources could be incorporated as a complement to the map data currently available in the car. In Figure 1.1, an overview of the input and output of the offline and onboard models is illustrated. To evaluate the performance of the offline model, the correctness measurement defined in (1.1) will be used.

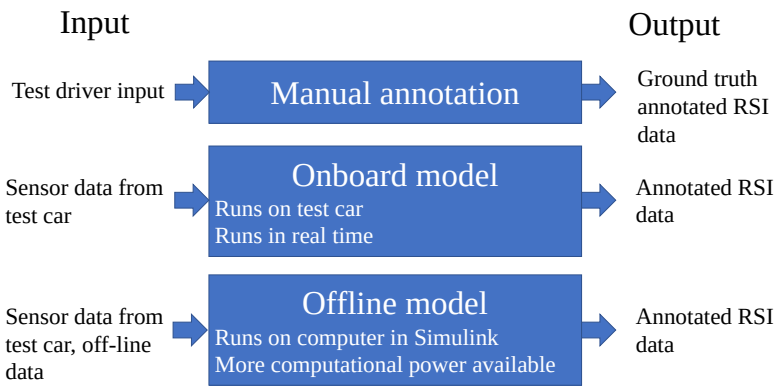


Figure 1.1: The inputs and outputs of the onboard model, the offline model and the manual annotation.

1.2 Purpose

The purpose of this work is to build an offline model which automatically annotates speed limit data based on data log files from Volvo Cars test vehicles, in order to replace the manually annotated data. The offline model does not run in the car, nor in real time. The aim is for the offline model to come as close to the manually annotated data as possible, regarded as the ground truth. The goal is for the speed limit data produced by the offline model to be able to validate the RSI function in the car.

1.3 Scope and delimitations

The scope of the work is to build a model in Simulink, a graphical programming environment for modeling and simulating systems, which automatically annotates the speed limit. Since the model runs offline, data from all time steps can be used to make estimations. Building an offline model allows more computational power to be used, thus allowing additional data sources to be incorporated. The work aims to investigate if additional data sources, for example the sign positions and the lane geometry as well as additional map data sources, can be incorporated in order to build an automatic speed limit annotation model. The goal is for the offline model to achieve a higher correctness, defined in (1.1), than the RSI function in the car.

The performance of the model is limited by the performance of the sensors and the data processing accuracy. For example, the signs captured by the cameras have been classified by a machine learning algorithm in the car, which is not necessarily strictly accurate [9]. However, evaluating the performance of the classification algorithm in the car is out of the scope of this study.

In this study, one of the potential additional data sources that will be investigated is an additional speed sign detector that uses the videos recorded by the test car. To limit the amount of data needed to train this model, only Swedish road signs are to be used. However, for the evaluation of the offline model, data from various countries are used. The reasons for this are that the amount of data the model can be evaluated on is significantly larger than if only Swedish log files are used, and to show that the model can adapt to different data source quality in different countries.

1.4 Ethical and sustainability aspects

The ethical challenges of this project are related to the field of autonomous driving. Two main challenges are described by Zhu *et al.* [10]. The first challenge addresses moral dilemmas for autonomous vehicles, for example whose safety should be prioritized. Should the car act egoistically and try to save itself and its owner, or should it try to save as many people as possible? Does age matter or should all people be viewed as equal? Questions like these have to be answered before self-driving cars can become a reality. The other problem stated is about where responsibility lies

when an accident happens. The three main agents are the driver, the car and the car manufacturer. A report by the German Ministry of Transport states that as cars get more autonomous, the responsibility for accidents move from the driver to the car manufacturer [11]. Even though advanced driver assistance systems (ADAS) are implemented in new cars, the driver still bears the utmost responsibility. Car manufacturers will likely want to keep it this way for as long as possible. Perhaps legislative changes will be needed to clearly define where responsibility lies as cars get more and autonomous.

Regarding sustainability, one study performed partly by Volvo states that there are large environmental benefits within autonomous cars [12]. This paper found that that when comparing cars using adaptive cruise control (ACC) versus cars without ACC, the *travel-weighted fuel consumption rate* was 5-7% lower when ACC was used [12]. The paper also states that the framework used for quantifying the difference that ACC makes could be used for other future changes that follows, suggesting that several other fuel consumption benefits can be drawn from a higher level of autonomy within cars.

2
Theoretical Background

This chapter introduces the theoretical concepts needed in order to build the offline model. Firstly, Dempster-Shafer theory is introduced, which enables fusion of multiple data sources. Secondly, stochastic optimization is described, which is applied to tune parameters in the offline model. Car data from the vision, digital map and Global Navigation Satellite Systems (GNSS) will be incorporated into the offline model. Therefore, theories regarding road sign detection, road edge detection, global navigation satellite systems, and digital maps are explained. Within the theories, the respective strengths and weaknesses of the different data sources are discussed. Based on this, the aim of this chapter is to provide a foundation for understanding the core concepts of the development of the offline function and a motivation to why fusion of the data sources can lead to a well-functioning offline model.

2.1 Sensor fusion

A common definition of sensor fusion is the process in which data from multiple sources is combined. By fusing the data, it is possible to reduce the uncertainty of the individual sensors. In this work, sensor fusion will be applied to speed limit sign detections from the camera as well as speed limit data from the digital map. Both data sources collect the same attribute, the speed limit at each time instance. Hence, *competitive fusion* is used with the aim of making the system more robust [13]. In the following section, Dempster-Shafer theory is introduced which is used for the fusion of vision- and digital map data.

2.2 Dempster–Shafer theory

The Dempster-Shafer theory is a commonly used sensor fusion algorithm for fusing vision- and map data in road sign recognition systems. Using Dempster-Shafer theory in this context would be a well-functioning method based on [14], [15]. The reasons for using Dempster's rule of combination to fuse data are that it provides a method of low computational complexity which can handle multiple data sources, and that the confidence from each sensor reading can be used as evidence in the Dempster-Shafer fusion.

The Dempster-Shafer (DS) theory is defined as a scheme for handling uncertainties in different propositions [16]. One proposition can for example be a single value,

x, or a set of values, $\{x, y, z\}$. The set of all mutually exclusive and exhaustive propositions is called the frame of discernment and is denoted by θ [14]. Mutually exclusive and exhaustive propositions means that the propositions are not subsets of each other and that only one proposition can be true at a time. The power set of θ is denoted by 2^θ. The difference between θ and 2^θ, given the mutually exclusive and exhaustive propositions x, y, z, is illustrated in (2.1).

$$\begin{aligned} \theta &= \{x, y, z\} \\ 2^\theta &= \{x, y, z, \{x, y\}, \{x, z\}, \{y, z\}, \theta\} \end{aligned} \tag{2.1}$$

Each proposition gets assigned a probability mass value based on the evidence for each proposition. The evidence can for example be a sensor reading that indicates a certain proposition. The definition of a mass value for proposition x is given in (2.2) where the empty set is denoted by $\emptyset$.

$$\begin{aligned} 0 \leq m(x) \leq 1 \\ \sum_{x \in 2^\theta} m(x) = 1 \\ m(\emptyset) = 0 \end{aligned} \tag{2.2}$$

In order to fuse data according to Dempster-Shafer theory, Dempster's rule of combination is used which combines mass values from two data sources [17]. The Dempster's rule of combination is defined by first introducing two massfunctions, $m_1(x)$ and $m_2(x)$ defined for $x \in 2^\theta$. The conflict factor, $k_{1,2}$, can then be computed as

$$k_{1,2} = \sum_{\substack{x_1, x_2 \in 2^\theta \\ x_1 \cap x_2 = \emptyset}} m_1(x_1) m_2(x_2) \tag{2.3}$$

The summation is over all pairs of propositions (x_1, x_2) such that $x_1 \cap x_2 = \emptyset$. The Dempster-Shafer combination of $m_1(x)$ and $m_2(x)$ for all $x \in 2^\theta$ can be computed when $k_{1,2} < 1$ as

$$m_{1,2}(x) = (m_1 \oplus m_2)(x) = \tag{2.4}$$

$$= \begin{cases} \frac{1}{1-k_{1,2}} \sum_{\substack{x_1, x_2 \in 2^\theta \\ x_1 \cap x_2 = x}} m_1(x_1) m_2(x_2), & x \neq \emptyset \\ 0, & x = \emptyset \end{cases} \tag{2.5}$$

$$\tag{2.6}$$

Dempster's rule of combination is a way of calculating the combined mass values of two data sources by summing the product of the two mass values when the data sources agree. This is mathematically defined as $x_1 \cap x_2 = x$. The conflict factor is defined as the summation of the mass values when the data sources disagree, denoted by $x_1 \cap x_2 = \emptyset$. The data sources are perfectly aligned when $k = 0$ and in total conflict when $k = 1$.

Dempster's rule of combination is commutative and associative which makes it possible to combine more than two data sources sequentially in any order [18]. An

example of this is shown in (2.7) where three mass values (m_1, m_2, m_3) are combined into the mass value m_4. Note that the symbol, $\oplus$, is used for Dempster's rule of combination between mass values. As shown, the order in which the mass values are combined does not affect the result.

$$\begin{aligned} m_4 &= (m_1 \oplus m_2 \oplus m_3)(x) \\ &= (m_1 \oplus (m_2 \oplus m_3))(x) \\ &= (m_2 \oplus (m_3 \oplus m_1))(x) \\ &= (m_3 \oplus (m_2 \oplus m_1))(x) \end{aligned} \tag{2.7}$$

2.3 Stochastic optimization

The following section aims to introduce evolutionary algorithms, which is a stochastic optimization method. Unlike classic optimization algorithms which most often require a differentiable objective function, stochastic optimization algorithms do not. Evolutionary algorithms are a subcategory of stochastic optimization methods which draw inspiration from biological mechanisms. These evolutionary algorithms use concepts such as reproduction, mutation and selection [19].

To make this more concrete, a simple example will be given where the aim is to find a pair of tuning parameters for a data fusion algorithm. The objective function which is to be minimized measures the quality of the data fusion algorithm by computing the error. This error measures the difference between the output of the data fusion algorithm and the ground truth data, giving an error percentage. The objective function here is non-differentiable, and therefore a stochastic optimization method would be a suitable choice. In genetic algorithms, the goal is to maximize a fitness function [19]. This is done by computing the fitness value, which in this case measures the similarity between the output of the data fusion algorithm and the ground truth data, giving a correct percentage. This means that minimizing the objective function is analogous to maximizing the fitness function. In Figure 2.1, the following three steps are illustrated. Each individual, illustrated as the red dots, is a pair of candidate parameter values. The following steps are taken in order to find the optimal parameter values.

1. Initialize a population randomly
2. Evaluate the fitness of all individuals. Crossover the individuals with the highest fitness values to form new, slightly mutated, individuals
3. Repeat step 2 until a satisfactory solution has been found

2.4 Computer vision

The following section aims to describe the theoretical framework regarding object detection, road sign classification, road edge modelling and road edge detection. All of these are computer vision methods which use images as their primary data source. A general overview of different object detection and road sign classification methods are given and the YOLO (You Only Look Once) algorithm is described. Finally, common error sources among computer vision methods are discussed.

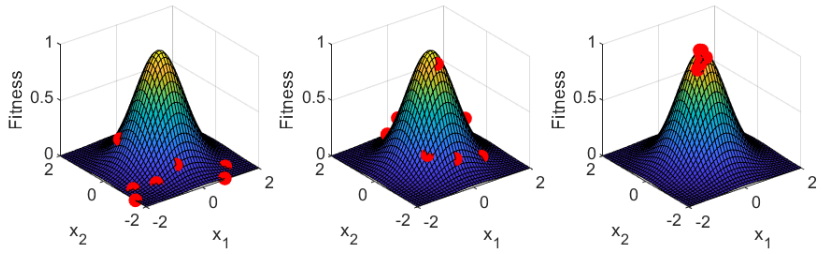


Figure 2.1: Three steps during the process in which an evolutionary algorithm maximizes a fitness function.

2.4.1 Object detection

Object detection is the task of both identifying the presence and location of objects in images. Object detection can be divided into two main approaches, neural network-based or non-neural network-based. Among the non-neural network-based approaches there are three main feature extraction methods, Haar- [20], SIFT- [21] and HOG features [22]. In this context, *features* refer to parts or patterns in an image which helps to identify objects. This can for example be the edge of the traffic sign, which is typically clearly different from its background.

Among the neural network-based methods, R-CNN (Region Based Convolutional Neural Networks) [23] is a common method, as well as YOLO (You Only Look Once) [24]. When non-neural network-based methods are used, an additional classification step is needed, where support vector machines (SVM) are commonly used [25]. The neural network-based methods do not need explicitly defined features for classification and can run significantly faster than non-neural network-based methods. In comparative studies, YOLO was found to run more than ten times faster than the SVM methods [24], [26]. YOLO has also been reported to be feasible to run in real time, which is not possible for the non-neural network-based methods [24].

When Fast R-CNN and YOLO was compared in a previous study, it was shown that Fast R-CNN ran at 0.5 frames per seconds and YOLO at 45 frames per second [24]. The mean average precision (mAP) of the models was found to be 63.4 for YOLO and 70.0 for Fast R-CNN, hence a slight advantage for Fast R-CNN. The average precision (AP) is a combined measurement of the precision and recall for a certain object. This is done using several different overlap thresholds for the predicted- and ground truth bounding boxes and the average value is computed, since there is a trade-off between precision and recall. To obtain the mAP, the mean of the AP for all classes is computed. This metric is commonly used to evaluate the performance of object detection algorithms.

2.4.2 Road sign classification

Road sign classification is the task of classifying what speed limit value is shown on the sign in an image. The state-of-the-art models for doing road sign classification can achieve human-level performance, meaning a correct recognition rate of

around 99% when tested on the German Traffic Sign Recognition Benchmark data set (GTSRB) [27]. These models are neural network based methods. An important distinction between the theoretical classification processes and sign classification in the car is that in the car, the traffic sign can be seen in multiple frames. If that sign then is classified equally across multiple frames, the cumulative classification accuracy is increased. The signs themselves can however be a source of error. For example, deterioration, vandalism or rotation can complicate the road sign classification process [25]. What most classifiers do is that they augment the road sign data during training to synthesize real-life variations of the road signs. This is done by, for example, random rotation and scaling, changing the brightness, saturation and contrast, or by adding noise to the images [9].

2.4.3 YOLO

YOLO is an algorithm which does object detection and classification. This is done by taking an image as input, and computing the output which is the predicted bounding boxes and class probabilities of objects in the image. These predicted bounding boxes and class probabilities are the output of the algorithm. The evaluation can be divided into two steps. The first step is to propose a region in the image in which there is a speed sign, and the second step is to classify the speed sign. Following is a brief description of how YOLOv2 does detection and classification [28]. The network architecture used has a total of 119168 parameters which are learned through training, which consists of weights and biases in each layer of the network.

First, the image is divided into 13x13 grid cells, which can be seen to the left in Figure 2.2. For each of these grid cells, a few *anchor boxes* are tested which can be seen to the right in Figure 2.2. The size, shape and number of anchor boxes are hyperparameters in the network which affect both the accuracy and the computational cost. The anchor boxes are proposals of the appropriate bounding boxes, which encode where in the image the speed sign appears. Anchor boxes should represent the general shape and size of the true bounding boxes in the training data. The bounding boxes are not limited within the grid cells that they belong to but can scale and move freely within the image. However, the objects will most likely belong to the grid cell in which they have their center. This also allows the network to associate objects with a certain part of the image. Then the *objectness* is computed, which is a measurement of how confident the network is of the presence of an object in a certain bounding box. For each of the bounding boxes, the speed sign class probabilities are also computed. The final step is to perform non-maximum suppression to obtain as many predictions as desired.

2.4.3.1 Loss function

Since the objective of the model is both to predict accurate bounding boxes as well as the correct classes, a multi-part loss function is needed. This loss function was proposed by Redmon *et al.* [24] and is here divided into three parts: localization-, confidence- and classification-loss, according to

$$L_{total} = L_{localization} + L_{confidence} + L_{classification} \tag{2.8}$$

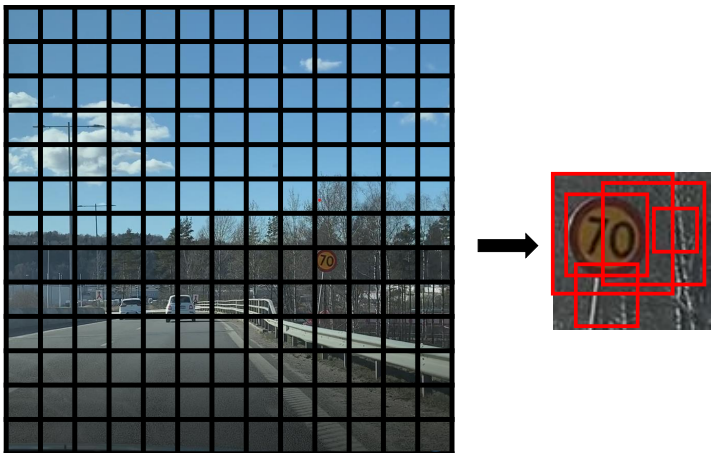


Figure 2.2: Left: the image divided into grid cells. Right: a number of bounding boxes tested for a grid cell to find the one that fits the true bounding box the best.

The localization loss is two-fold and can be seen in (2.9). The first term accounts for the distance between the center of the predicted bounding box to the center of the true bounding box. Here the variables (x_i, y_i) and $(\hat{x}_i, \hat{y}_i)$ denote the center of the predicted- and ground truth bounding box j in grid cell i. The variables (w_i, h_i) and $(\hat{w}_i, \hat{h}_i)$ denoted the width and height of the predicted- and ground truth bounding box j in grid cell i. The second term accounts for the error regarding the width and height of the bounding box. The binary variable $\mathbb{1}_{ij}^{obj} \in \{0, 1\}$ is used to ensure that only the best bounding box is penalized. The best bounding box is the one which has the highest intersection over union, a measurement of how well the predicted bounding box fit the true bounding box. This is discussed further in Section 2.4.3.2. The variable $\mathbb{1}_{ij}^{obj} = 1$ for the bounding box with the highest IoU, and $\mathbb{1}_{ij}^{obj} = 0$ for the other bounding boxes. This means that only one bounding box is responsible for the predicted object. A tuning parameter denoted $\lambda_{coord} > 0$ is also used in this loss function, which can change the weight of prediction errors for the bounding box. The summations indices S^2 and B stand for the total number of cells and bounding boxes respectively, both which are visualized in Figure 2.2.

$$\begin{aligned} L_{localization} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \tag{2.9}$$

The confidence loss is defined in (2.10). The two terms account for when there is, and when there is not, an object in bounding box j in grid cell i. To ensure that only the best bounding box is penalized, the binary variables $\mathbb{1}_{ij}^{obj} \in \{0, 1\}$ and $\mathbb{1}_{ij}^{noobj} \in \{0, 1\}$ are used. To decide which bounding box is the best, the intersection over union (IoU) for the different bounding boxes predicted by the algorithm is computed before the loss is evaluated. $\mathbb{1}_{ij}^{noobj} = 1$ if bounding box j in grid cell i does not contain an object, and $\mathbb{1}_{ij}^{noobj} = 0$ otherwise. A tuning parameter denoted $\lambda_{noobj} > 0$ is also used, which can change the weight of the error when no object is

detected in the predicted bounding box. The reason for doing this can be seen in Figure 2.2, where most of the grid cells does not contain speed signs, and therefore the loss from these cells might overpower the loss from cells with objects. The tuning parameter λ_{noobj} aims to neutralize this. The variable C_i and $\hat{C}_i$ denotes the confidence score of both the prediction and the ground truth, meaning the certainty of there being an object in bounding box j in grid cell i.

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{2.10}$$

The definition of classification loss is presented in (2.11). Here the predicted speed sign class is compared to the true speed sign class. Since only grid cells with objects should be considered for classification errors, the binary variable $\mathbb{1}_i^{obj} \in \{0, 1\}$ is used. The value of $\mathbb{1}_i^{obj}$ is 1 if an object is detected in the grid cell regardless of by what bounding box, and 0 if no object is detected. The variable p_i and $\hat{p}_i$ denote the estimated and true conditional class probabilities in each grid cell.

$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{2.11}$$

2.4.3.2 Anchor box estimation

The anchor boxes are a crucial hyperparameter in the network which affect the accuracy and the computational cost. Both the size, shape and number of anchor boxes must be selected. To estimate how well the anchor boxes fit the true bounding boxes, the *intersection over union* (IoU) can be computed [28]. An illustration of the IoU is provided in Figure 2.3. The IoU is scale invariant, meaning that the error is equal for both small and large anchor boxes. Many anchor boxes will likely increase the IoU, since more boxes increases the probability that one of them fits each of the true bounding boxes well. This does however increase computational cost and could potentially lead to overfitting. The suitable number of anchor boxes is therefore a trade-off between accuracy and computational speed.

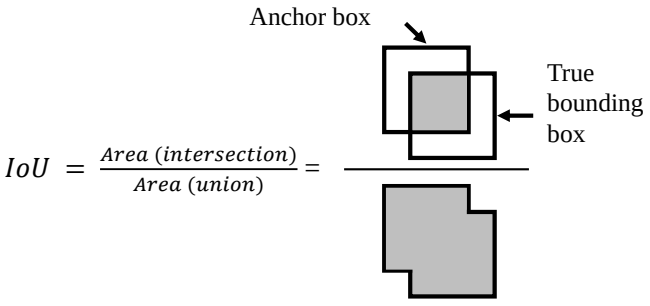


Figure 2.3: The IoU measurement, used to compute appropriate anchor boxes.

2.4.3.3 Transfer learning

To make YOLO work well, transfer learning is often used [28]. The idea behind transfer learning in image classification is that the first layers of the network are responsible for feature extraction, and the final layers are responsible for mapping combinations of features to specific objects. By replacing the final layers of the network, it is possible to remap these to objects in a new dataset with significantly less time and less training data required, opposed to what would be needed to train the model from scratch [29]. When YOLO was originally introduced [24], the model was trained on the ImageNet 1000-class competition dataset, which has over 590,000 annotated objects [30]. After this, the model was fine-tuned to the PASCAL VOC datasets from 2007 and 2012 [31], [32]. These datasets contain 12,608 and 27,450 annotated objects in their training- and validation sets. These datasets contain 20 classes and have an average of about 2350 objects per class.

2.4.4 Road edge modelling

Clothoids are curves whose curvature changes linearly with their length. Because of this property, they are commonly used in highway- and railroad curve design to prevent sudden changes in lateral acceleration [33]. To reduce computational complexity, the following approximation of the clothoids is used, which can be seen in (2.12) where $F(x)$ is a polynomial which represents the shape of the road edge. This is inspired by Sánchez-Reyes and Chacón, who used polynomials to approximate clothoids [34].

$$F(x) = a + bx + cx^2 + dx^3, \quad \text{if } 0 \leq x \leq x_t \tag{2.12}$$

The clothoids are in the car's coordinate system, with the origin in the center of the rear axis, see Figure 2.4. In (2.12), x_t denotes the maximal longitudinal distance for which the model is valid.

To find out to what degree the road turns and in which direction, the *signed curvature* can be computed [35]. The signed curvature is expressed in (2.13), x', y' and x'', y'' are the first and second derivatives with respect to time.

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{2.13}$$

However, using the formula expressed in (2.13) requires a parametrization of the clothoid defined in (2.12), since the functions need to be differentiable, and hence the following expression is used. Here $\gamma(t)$ denotes the parametrization of the polynomial defined in (2.12) expressed by the parameter t representing time, and t_t represents the upper threshold corresponding to the maximal longitudinal distance which the road edge is valid.

$$\gamma(t) = (x(t), y(t)) = (t, a + bt + ct^2 + dt^3), \quad \text{if } 0 \leq t \leq t_t \tag{2.14}$$

Positive curvature values mean the road edge is turning left with regards to the direction that the car is driving, and negative values means that the road edge is turning to the right [35].

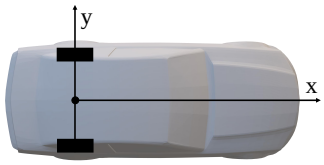


Figure 2.4: Car coordinate system aligned with the rear axis. The longitudinal- and lateral axis are denoted by x and y respectively.

2.4.5 Road edge detection

The detection of road edges is crucial in ADAS, since this information can be used to prevent the vehicle from driving off the road. Janda *et al.* [36] defines the road edge as the change from road surface to non-road area, as illustrated by the red lines in Figure 2.5. The difficulties faced when doing road edge detection and classification are like those faced when doing road sign detection. The main issues are adverse lighting conditions and shadows, as well as the possible absence of lane markings [36]. Some road edge detection systems do not only use the cameras, but also the radar sensors to detect road edges [36]. The latter is particularly useful when there are barriers on the side of the road, which is a common feature on highways for example. An example of this is illustrated in Figure 2.5, where there are concrete barriers to the left. In this scenario the radar sensors could be useful for detection of the concrete barriers.

Figure 2.5: Image from the forward-looking camera. The road edges are illustrated by the red lines.

2.4.6 Error sources for vision-based methods

The mentioned vision-based methods for object detection, road sign classification, road edge modelling and road edge detection have common sources of error. Among

the most common are adverse lighting and weather conditions, which can affect the camera performance. Examples of such conditions are low lighting, direct sunlight, shadows and rain, which all affect the camera performance negatively [25]. Snow can also make the lanes as well as road edges hard to distinguish. Besides, both cars and trees might occlude objects which will affect the performance of all aforementioned methods. Another issue is viewpoint variation, where objects look different depending on the point of view [37]. It is essential when merging vision-based data with data from other sources to understand the strengths and weaknesses of each data source.

2.5 Global navigation satellite systems

Global navigation satellite systems (GNSS) are based on satellite navigation where a global network of satellites is orbiting the earth and transmitting radio signals to a ground-based antenna in order to determine e.g. position and velocity of a ground object. There are in total five different GNSS constellations in operation where the two largest ones are the Global Positioning System (GPS) owned by the United States and GLONASS owned by the Russian Federation. The remaining three are QZSS from Japan, Beidou from China, and Galileo which is developed by the EU and the European Space Agency [38].

The frequency bands of the transmitted GNSS signals are different for each constellation and range from 1 to 2 GHz. Each GNSS constellation transmits signals in at least two different frequencies. The three main components of the signal are the carrier frequency, the ranging code and the navigation message. The carrier frequency is the frequency of the electromagnetic signal that gets transmitted from the satellite. The ranging code is the main component and consists of a sequence of binary numbers called Pseudo-Random Noise (PRN) [39]. The PRN sequence holds information about the identity of the satellite and a time stamp of when the signal was transmitted. The navigation message contains data about the health of the satellite, biases of the satellite clock, the satellite ephemeris (parameters for calculating the position and velocity of the satellite) and other complementary data.

The PRN code and the navigation message are combined using modulo 2 summation performed using an exclusive OR operation. The result is a new binary sequence including information from both the PRN sequence and the navigation message. The carrier signal is then modulated using the Binary Phase Shift Keying (BPSK) modulation technique. In this technique, the phase of the carrier signal is shifted positively 180° if the binary signal is changed from 0 to 1, and negatively by the same amount if the signal is changed from 1 to 0 [39].

Since the transmitted signal contains information about when the signal was transmitted, an approximate range between the satellite and the receiver can be computed. This range is referred to as the pseudorange and can be seen in (2.15) [40].

$$R_p(t) = c[t_r - t^s] + \epsilon \tag{2.15}$$

In (2.15), $R_p(t)$ denotes the pseudorange, c is the speed of light, t_r is the time when the signal is received measured by the receiver clock, t^s is the time when the signal was transmitted measured by the satellite clock and ϵ represents the error term. The error term will be discussed further in Section 2.5.1. The transmitted time, t^s, is decoded either by code phase processing using the PRN signal or by carrier phase processing using the carrier signal directly. Receivers that use carrier phase measurements are in general more accurate and of higher complexity, compared to the ones that use code phase measurement [41].

The main information that can be extracted from a single satellite is the time difference between when the signal is transmitted and received, which can be used to calculate the distance between the ground object and the satellite. This is however not enough to determine the position of the ground object. Positioning is done by combining distances to several satellites using a geometrical method called trilateration [42].

The concept of trilateration is illustrated in Figure 2.6 where three satellites (located at A, B and C) and one object (located at D) receiving satellite signals are shown. Each satellite has a circle with a radius equal to the measured distance from the object to the satellite. By intersecting the three circles, it is possible to find a unique intersection point corresponding to the 2D position of the object. This can be extended to 3D where spheres are used instead of circles, in this case four satellites are needed to determine the 3D-position of the object.

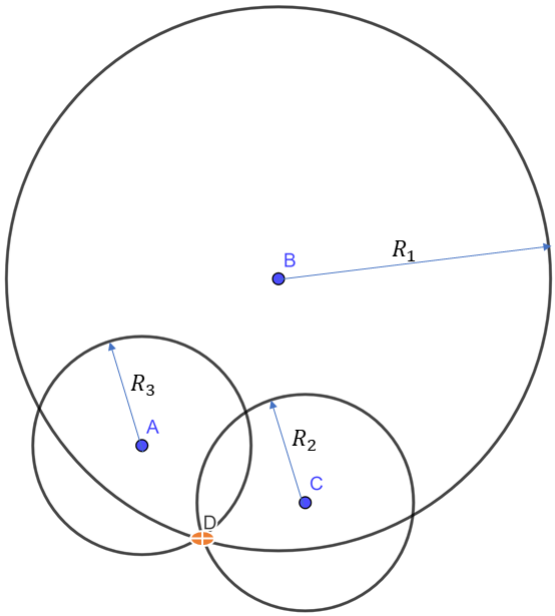


Figure 2.6: Trilateration to find position of object D using satellite A, B and C.

2.5.1 Limitations of GNSS

The limitations of GNSS are are important to acknowledge when designing a system using GNSS since the performance cannot be expected to be perfect in all cases and

depends on a number of different factors. The GNSS accuracy is affected by the satellite, the receiver and by the propagation medium. These factors are discussed in the following sections. The concepts that are introduced in the following sections apply to all GNSS systems, but the accuracy metrics are specific for GPS systems.

2.5.2 Satellite errors

The satellites contain very accurate atomic clocks that are used to obtain a precise timestamp of when the signal was transmitted (contained in the PNR). The satellite clocks are however not perfect and have an error of about 8.64 to 17.28 ns per day which corresponds to a distance error of 2 to 5 m [43]. The accuracy of the satellite clocks is verified by ground stations based on the biases transmitted from the satellites as part of the navigation message [44].

Another aspect that affects the GNSS accuracy is the ephemeris error, caused by an incorrect prediction of the satellite position. The satellite positions are predicted by overlapping observations from the ground stations together with a model of the forces acting on the satellite. This error is usually in the range of 2 to 5 m [44].

2.5.3 Receiver errors

The crystal clocks that are generally used in receivers are far less accurate than the atomic clocks used in satellites [45]. The errors from the inaccurate receiver clocks can however be minimized by differentiating between different satellites, or by treating the receiver clock error as unknown when making the prediction [44]. Differentiating between satellites essentially means that two or more satellites are used at the same time to cancel out the clock error. This is possible since two observations made at the same time have the same clock error which makes it possible to cancel this term.

The signal path from the satellite to the receiver is important for making an accurate prediction of the position. In some scenarios, the signal is reflected which causes the antenna to receive the signals from different paths, see Figure 2.7. This error is called multipath error and is caused by interference between the reflected- and direct signal [46].

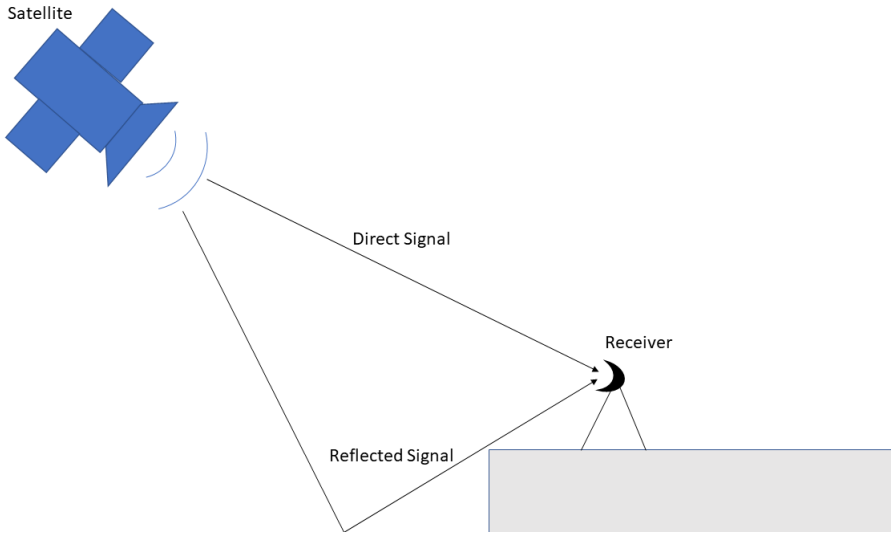


Figure 2.7: Illustration of multipath error.

The effect of multipath error can be reduced by using a receiver location with few reflective areas, introducing a choke ring antenna that attenuates the reflected signal, or by using an antenna with a similar polarization as the GNSS signal [44].

Hardware impairments at the receiver side are also a source of error. This adds noise to the received signal which decreases the accuracy of the estimated position [44].

2.5.4 Signal propagation errors

Signal propagation errors occur when the signal is delayed during propagation through the atmospheric layers. The gases in the ionospheric layer are ionized, causing the electron density to vary. High electron density causes the ionospheric layer to be a dispersive medium which means that the propagation speed of an electromagnetic wave is dependent on the frequency [47].

A GNSS signal consists of multiple components at different frequencies. Since the propagation rate in the ionospheric layer depends on the frequency of the signal, each component of the GNSS signal is delayed differently. The speed of the carrier signal is increased while the speed of the PRN- and navigation message is decreased in the ionospheric layer. The result is a longer measured distance if the PRN code is used and a shorter measured distance if the carrier signal is used [47].

The signals are also affected by propagation through the troposphere, called tropospheric delay. The troposphere is, in contrary to the ionospheric layer, a non-dispersive medium for frequencies below 15 GHz. This means that all frequencies in GNSS signals, which are electromagnetic waves, are delayed equally [48]. This causes the same delay for all GNSS components, and the measured distance will be longer than the actual distance regardless of which component that is used.

The tropospheric delay cannot be compensated by using information from the different components since all components are delayed by the same amount. Instead, mathematical models based on meteorological measurements are used to predict the amount of delay [44]. The delay can be divided into a dry and a wet component. The dry component causes approximately 90% of the delay and can be accurately predicted, while the wet component, causing 10% of the delay, is harder to predict [49].

2.6 Digital Map

A digital map is a data set that describes a geographical location virtually [50]. Depending on the available data, digital maps have varying levels of detail. Typical attributes of digital road maps are lane geometry, location of traffic lights, the current speed limit and the number of lanes [51].

Digital maps have long been used together with GNSS-coordinates for navigation and routing purposes in cars. In recent years, the scope of applications has increased and several ADAS functions demand digital map information [52]. A key motivation for using the digital map is that it allows the car to see further ahead compared to built in sensors like the LiDAR, cameras and radars, which have a limited range of up to a few hundred meters [51]. Another benefit of the digital maps is that they provide data past solid objects such as cars and buildings, something not possible for the previously mentioned sensors.

2.6.1 ADASIS

In 2001, the Advanced Driver Assistance Systems Interface Specifications (ADASIS) Forum was founded with the purpose of creating a standardized map data interface between stored map data and ADAS applications [53]. The members of the ADASIS Forum are high profile automotive manufacturers such as BMW and Ford. Since the forum was founded three versions of the ADASIS protocol have been released and the protocol is nowadays accepted as an industry standard [54].

2.6.2 ADASIS Horizon

The core idea of ADASIS is to provide a method for extracting necessary attributes-from the digital map and utilize it for ADAS applications. The first step of this process is to extract the vehicle's position, speed and heading using a combination of GNSS-positioning and dead reckoning. The GNSS-positioning is used in most cases but sometimes is the accuracy limited due to blocked signals, discussed in section 2.5.1. The solution is to use dead reckoning in these cases, which is a method for fusing information from the internal measurement unit (IMU) to calculate the heading, speed and position of the car. However, dead-reckoning drifts over time because of integration errors and can therefore only be used for short periods of time, for example when the car drives trough a tunnel [55].

The second step is to perform map-matching to connect the vehicle to the road network [56]. This process uses information such as the position, speed and heading of the vehicle to calculate the road segment that the vehicle is most probable to travel on, called the *Most Probable Path* (MPP). One approach for estimating the MPP is to assign weights to road segments nearby based on different criteria [56]. One weight could for example be how much the road segments are aligned with the heading of the vehicle. The map-matched position is generated by choosing the road segment with the highest sum of weights, and by matching the GPS vehicle position to the closest point on that segment. The third step is to extract all necessary attributes from the digital map of the MPP to create an *Electronic Horizon* (EH). The EH is a collection of road attributes such as the speed limits, sign information and other relevant information given by the map data. Figure 2.8 illustrates the MPP and the corresponding EH of a road network with four paths. The attributes of the EH are in this case two speed signs, one elevation indication and one stop sign.

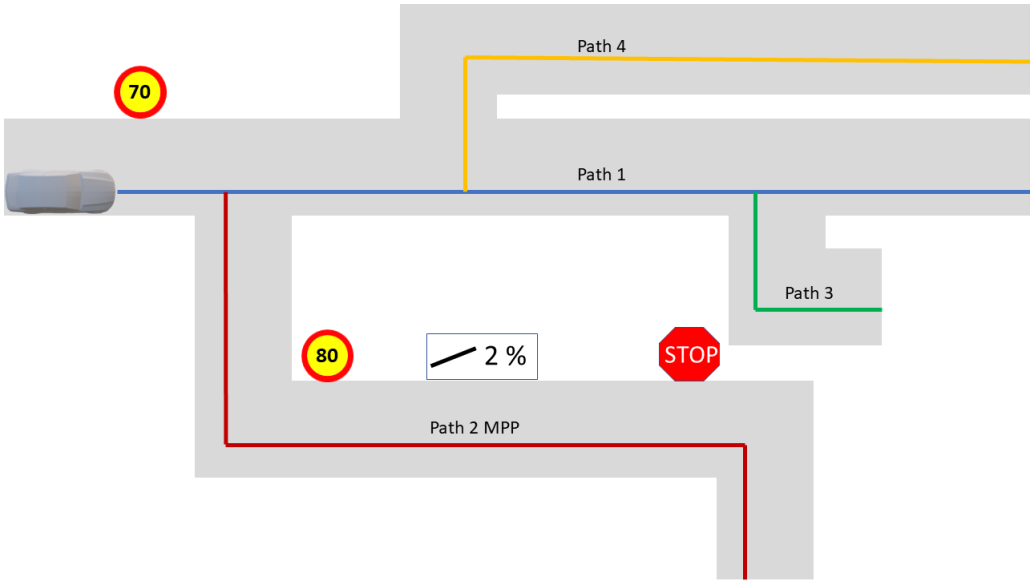


Figure 2.8: MPP and EH of a road network with four paths.

2.6.3 ADASIS System architecture

The core components of the ADASIS system are the ADAS Horizon Provider (AHP), the ADASIS protocol and the ADASIS Horizon Reconstructor (AHR) [54]. The AHP is responsible for extracting all relevant attributes from the digital map, the ADASIS protocol describes how the EH is decoded to enable Controller Area Network (CAN) communication and finally, the AHR is used to reconstruct the EH for each ADAS application. A simplified version of the system architecture is illustrated in Figure 2.9.

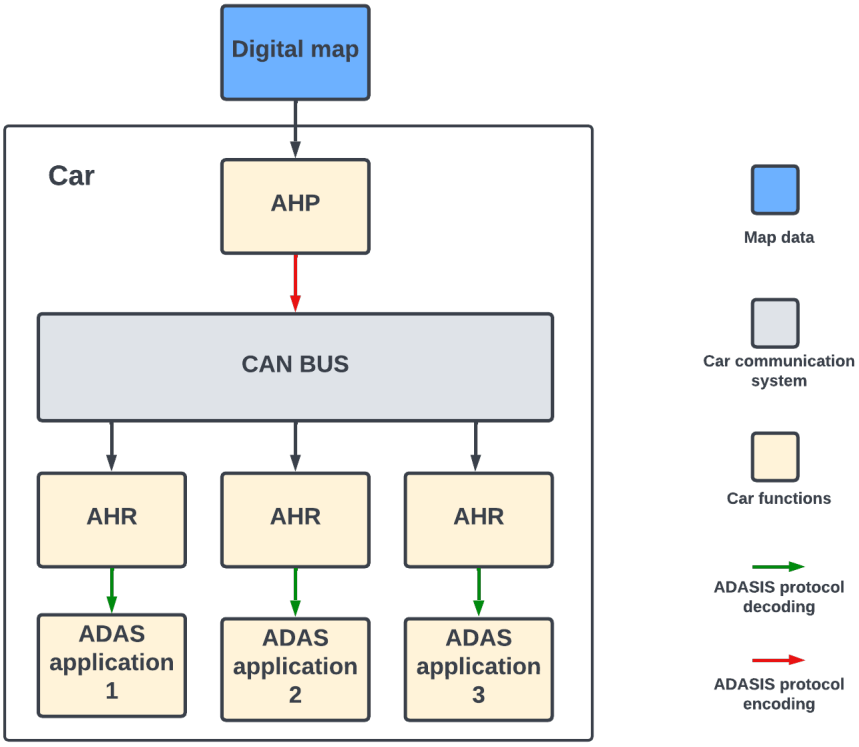


Figure 2.9: ADASIS system architecture.

2.6.4 ADASIS messages

The digital map attributes becomes translated to a number of different messages in the AHP [54]. These messages are listed below:

1. **Segment message**: extracting and merging the most important attributes of a road segment. The attributes can for example be the speed limit, road class, number of lanes and so on. A segment is defined as a fraction of a road where the attributes are constant. A full path is built by one or several segments.
2. **Profile message**: Includes the road curvature and geometry.
3. **Attachment Message**: Includes attributes such as signs on a path. The path is identified using an index, and the message also includes information about the distance to the signs.
4. **Stub Message**: Similar information as the Attachment message besides that the Stub message only contains information from the MPP which simplifies the reconstruction since less data is regarded.
5. **Position Message**: Information about the current position of the vehicle. The vehicle speed, timestamp from the positioning system and the current lane is also included.

The messages get decoded and encoded via the ADASIS protocol which enables transmission of the EH data to ADAS applications.

3
Methods

This chapter aims to provide a description of the different methods used for the development of the offline model. First, a proposed model is introduced, which gives an overview of the different subsystems and how they will be combined to produce a final model. Subsequently, a section about the data collection from the test vehicle is included. The section provides an insight into what data is extracted from the test vehicle and how manual annotations are used as ground truth. Each subsystem defined in the proposed model is then described and evaluated. The last section outlines the architecture of the final model based on the performance of each subsystem.

3.1 Proposed model structure

The first step of the development of the offline model was to make a design choice of the system architecture. This model is referred to as the proposed model. The core idea was to use processed sensor data provided by the sensor setup in the test car and fuse it with additional offline data sources to create a more reliable model. Figure 3.1 shows the proposed model structure.

The main input of the proposed model is the test-vehicle data log files containing all sensor data sampled at 40 Hz. The data is then filtered and only the necessary data is extracted. The vision- and Google data can be processed and used directly to produce mass values for the Dempster-Shafer fusion, see section 2.2. The offline blocks, illustrated by the gray color in Figure 3.1, are additional data sources used for speed limit estimation. The first block utilizes the position, speed, and timestamps as input and produces map-matched positions that can be combined with an additional digital map to produce speed limit estimations. The second block takes the front camera video stream as input and classifies the speed signs using a YOLO network.

One of the core issues, described in section 1.1, was that the vision system captures speed signs belonging to adjacent roads creating wrong speed limit estimations from the vision. To solve this issue, an off-ramp filtering system is used which decreases the probability of the vision reading when an off-ramp is present.

The output from the map and the vision systems for the off- and online data is first fused together separately using Dempster-Shafer fusion. Thereafter, the output is fused again to produce a final speed estimation. The speed estimation from the

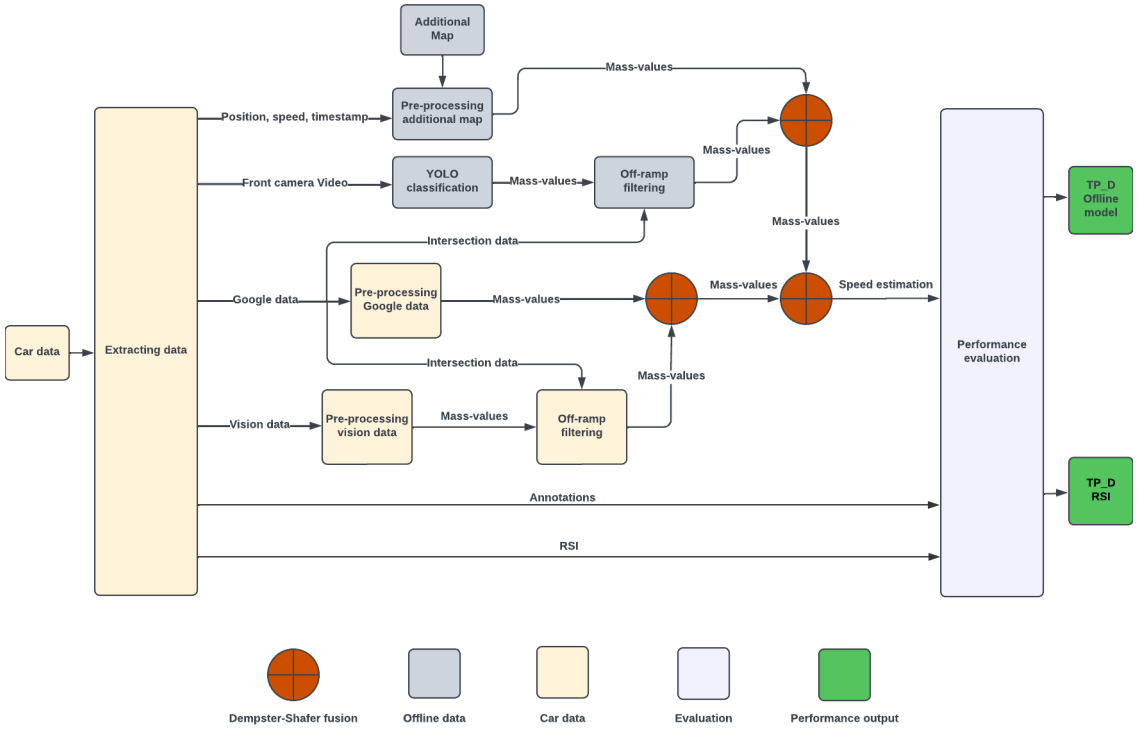


Figure 3.1: Proposed system architecture.

offline model is finally compared with the manual annotations in the performance evaluation block to calculate the true positive distance, TP_D, defined in Section 1. The RSI speed limit estimation is also extracted and compared to the manual annotations in order to obtain a performance reference for the offline model.

Each subsystem of the proposed model will be described, tested and evaluated separately in the following sections. Finally, the subsystems with the most promising performance will be combined to create a final system architecture of the offline model.

3.2 Data collection from test vehicle

The data used for developing the offline model was based on data log files from Volvo's test vehicles. The sensor data is sampled at 40 Hz and stored as .mat-files in a common data cluster. Relevant sensor signals can be retrieved from the files and used for simulation purposes. Sensor signals related to object detection, positioning, the Google map, RSI and manual annotations were extracted for development of the offline model, see proposed model, Figure 3.1.

Performance of the offline model is evaluated based on the ground truth data. The ground truth is defined as the true speed limit on road segments and consists of manual annotations. Each annotation gets a timestamp which makes it possible to sync it with the current position and thus a continuous signal for the true speed limit can be created.

3.3 Fusion between map- and camera data from test vehicle

Fusion between map- and camera data from the test vehicle is a core component of the offline model. Before fusing the data using Dempster's rule of combination, the data must be pre-processed. This section will first describe the map- and camera data pre-processing steps, respectively, and then how the data was fused. As the final step, performance metrics will be obtained from the fused data.

3.3.1 Pre-processing of Google data

The host vehicle retrieves attributes from the Google data through the AHP. One of those attributes is the speed limit on the current road. The map attributes are organized as different messages and profiles in the forward horizon, as described in Section 2.6.4. The different attributes are associated with a path and a longitudinal offset (relative to the host car). To retrieve the current status of map attributes, an AHR is used to align the vehicle's current position on its MPP to attributes on the associated path. If the AHR works correctly, a speed limit estimation can be given directly by the AHR.

However, the onboard AHR sometimes experiences software bugs resulting in an incorrect speed limit estimation. But since all map messages and profiles are logged, it is possible to reconstruct the map attributes using an offline implementation of the AHR. This bypasses the software issues and makes the map data usable even when the onboard function has errors.

The software bugs for the AHR and AHP are systematic for some countries in the EU which makes it possible to create a general offline reconstructor. The offline reconstructor uses the map messages and profiles with corresponding longitudinal positions and timestamps to match it to the position and MPP of the host car.

The speed limit from the Google map is static which means that new speed limit changes, due to roadwork or similar, are not captured by the digital map. To account for these errors, a tuning parameter, α_{Google}, was introduced. This tuning parameter was meant to decide how much the current reading was trusted. For example, if $\alpha_{Google} = 1$, then the Google speed limit is trusted to 100%, but if α_{Google} is a lower value, then the map is not trusted fully. This parameter was used to construct mass values used in the DS-fusion. The collection of all possible speed limits in a country is denoted by θ, and is defined as the *frame of discernment* by Nienhuser *et al.* [14]. In Sweden this set is $\theta = \{10, 20, ..., 120, 130\}$. The Google map speed limit reading is assigned a mass value of α_{Google} and the *frame of discernment*, θ, is assigned a mass value of $1 - \alpha_{Google}$. This is mathematically defined as:

$$\begin{aligned} m_{Google}(\emptyset) &= 0 \\ m_{Google}(\{s_{Google}\}) &= \alpha_{Google} \\ m_{Google}(\theta) &= 1 - \alpha_{Google} \end{aligned} \tag{3.1}$$

Where s_{Google} is the speed limit given by the Google reading and $m_{Google}(\emptyset) = 0$ means that no mass value is assigned if the Google reading does not contain any information (generating $\emptyset$). One example of assigning mass values from the Google data is given below where $s_{Google} = \{80\}$ and $\alpha_{Google} = 0.9$:

$$\begin{aligned} m_{Google}(\{80\}) &= 0.9 \\ m_{Google}(\theta) &= 0.1 \end{aligned} \tag{3.2}$$

These mass values mean there is a probability of 90% that the speed limit is 80 km/h and a 10% probability that the speed limit is any value in the set of θ.

3.3.2 Pre-processing of test vehicle vision data

The onboard vision system is sampled at 40 Hz and is limited to maximally eight road sign detections at each time step. The highest possible number of detections in one frame is a design choice by the onboard object detection supplier. Each detection has a classification confidence. Since the signs are only classified in a few camera frames, but the fusion algorithm runs continuously at each time step, the camera readings need to be kept for some time. In the following list, the processing steps of the camera signals is described. The inputs are the speed sign detections, the classification confidences, and the relative distance to the sign from the car. The input is then mapped to the probability mass values for different speed limits.

1. Only consider speed signs within a 10 m radius of the car.
2. If multiple signs of the same speed limit are detected in a single frame, for example two 50-signs, use the detection with the highest confidence.
3. If multiple signs with different speed limits are detected in a single frame, for example a 50-sign and a 70-sign, decrease the probability of both detections in proportion to their classification confidence.
4. Adjust the probabilities using the camera confidence tuning parameter α_{cam}.
5. Keep the last speed sign detection for up to five minutes, given that no new speed signs are detected.

In what follows $s_{cam,i}$ denotes the detected speed limits in a camera frame, and $c_{cam,i}$ represents the corresponding confidence. Since eight detections can be made in one camera frame, the index $1 \leq i \leq 8$ is used for labeling the detections.

Even if the confidence of the classification is close to perfect ($c_{cam,i} = 1$), there is still a possibility that the detected sign belongs to the wrong road. To capture this uncertainty a tuning parameter denoted by α_{cam} was introduced. This parameter was used together with the confidence $c_{cam,i}$ to determine the mass value for each speed class, defined in Section 3.3.1. The mathematical definition of mass values are shown in (3.3).

$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{s_{cam,i}\}) &= \alpha_{cam} \cdot \frac{1}{n} \cdot c_{cam,i} \\ m_{cam}(\theta) &= 1 - \alpha_{cam} \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} c_{cam,i} \end{aligned} \tag{3.3}$$

In (3.3), $s_{cam,i}$ and $c_{cam,i}$ are the speed limit and confidence respectively for the i-th sign detection. The total number of detections are denoted by n. The benefits of assigning mass values in this way are to be able to use the tuning parameter α_{cam} to decrease the confidence of the readings, to be able to handle multiple detections and finally to decrease the confidence if two or more different speed signs are captured at the same time since only one of the detections is reliable. Some examples of assigning camera mass values are shown below.

A toy example of a perfectly classified 70 km/h ($c_{cam,1} = 1$) is shown in (3.4). The tuning parameter α_{cam} is set to 0.9, meaning that even a perfect classification is not trusted undoubtedly. The probability of any speed class, θ, is 0.1 in this case.

$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{70\}) &= 0.9 \cdot \frac{1}{1} \cdot 1 = 0.9 \\ m_{cam}(\theta) &= 1 - 0.9 \cdot \frac{1}{1} \cdot \sum_{i=1}^{1} c_{cam,i} = 0.1 \end{aligned} \tag{3.4}$$

When the confidence of the classified sign is decreased to 0.8 ($c_{cam,1} = 0.8$) while α_{cam} is kept the same, the mass values of $m_{cam}(\{70\})$ will decrease and $m_{cam}(\theta)$ will increase. This is shown in (3.5).

$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{70\}) &= 0.9 \cdot \frac{1}{1} \cdot 0.8 = 0.72 \\ m_{cam}(\theta) &= 1 - 0.9 \cdot \frac{1}{1} \cdot \sum_{i=1}^{1} c_{cam,i} = 0.28 \end{aligned} \tag{3.5}$$

In (3.6), an example where two different signs are detected at the same time, both with probability 1 is considered. The tuning parameter is kept the same ($\alpha_{cam} = 0.9$). Both speed limits have the same mass values which is expected since they have the same confidence.

$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{70\}) &= 0.9 \cdot \frac{1}{2} \cdot 1 = 0.45 \\ m_{cam}(\{80\}) &= 0.9 \cdot \frac{1}{2} \cdot 1 = 0.45 \\ m_{cam}(\theta) &= 1 - 0.9 \cdot \frac{1}{2} \cdot \sum_{i=1}^{2} c_{cam,i} = 0.1 \end{aligned} \tag{3.6}$$

Another case is considered in (3.7), where the 80 km/h sign is classified with a confidence of 0.8 and the 70 km/h sign is classified with a confidence of 0.6. The tuning parameter is set to $\alpha_{cam} = 0.9$. In this case, any speed limit, θ, is believed with the highest probability. This is because of the two conflicting speed sign detections, as

well as the camera confidence tuning parameter α_{cam} being less than 1.

$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{70\}) &= 0.9 \cdot \frac{1}{2} \cdot 0.6 = 0.27 \\ m_{cam}(\{80\}) &= 0.9 \cdot \frac{1}{2} \cdot 0.8 = 0.36 \\ m_{cam}(\theta) &= 1 - 0.9 \cdot \frac{1}{2} \cdot \sum_{i=1}^{2} c_{cam,i} = 0.37 \end{aligned} \tag{3.7}$$

3.3.3 Dempster-Shafer implementation

By using Dempster-Shafer's rule of combination, the mass values from the camera and the Google map can be fused. This can be seen in (3.8), where the variables x, s_{Google} and $s_{cam,i}$ are single speed limit values.

$$\begin{aligned} m_{Google,cam}(\{x\}) &= \frac{1}{1-k} \sum_{s_{Google} \cap s_{cam,i} = x} m_{Google}(\{s_{Google}\}) \cdot m_{cam}(\{s_{cam,i}\}) \\ k &= \sum_{s_{Google} \cap s_{cam,i} = \emptyset} m_{Google}(\{s_{Google}\}) \cdot m_{cam}(\{s_{cam,i}\}) \end{aligned} \tag{3.8}$$

The best estimation from the fused data is then selected as the speed limit with the highest fused mass value $m_{Google,cam}$. This is mathematically defined as:

$$s_{be} = \arg\max_{\theta}(m_{Google,cam}(\{x\})) \tag{3.9}$$

with s_{be} representing the best speed estimation (with the highest confidence). Some examples of different scenarios are considered below.

In scenario 1, shown in Table 3.1, both the Google map and the camera shows 50 km/h with high confidence. In this case, both data sources are perfectly aligned resulting in a fused mass value of $m_{Google,cam}(\{50\}) = 0.995$ and a conflict factor of $k = 0$, resulting in a best estimation of $s_{be} = 50$ km/h. The fused mass value is higher than the mass values of the individual sensors which means that fusion strengthens the accuracy when the sensors are aligned.

Table 3.1: Camera and map data fusion scenario number 1.

	m_{Google}	m_{cam}	$m_{Google,cam}$
$\{50\}$	0.9	0.95	0.995
$\{\theta\}$	0.1	0.05	0.05
k	0	0	0
s_{be}	0	0	$\{50\}$

In scenario 2, shown in Table 3.2, the Google map receives a reading of 80 km/h and the vision system receives two readings of 70 km/h and 80 km/h. The two data sources are not perfectly aligned, since two different readings are retrieved from the vision, resulting in a conflict factor of $k = 0.2430$. The fused mass value with the highest probability is $m_{Google,cam}(\{80\}) = 0.9155$ and a best estimation of $s_{be} = 80$ km/h.

Table 3.2: Camera and map data fusion scenario number 2.

	m_{Google}	m_{cam}	$m_{Google,cam}$
$\{70\}$	0	0.27	0.0357
$\{80\}$	0.9	0.36	0.9155
$\{\theta\}$	0.1	0.37	0.0489
k	0	0	0.2430
s_{be}	0	0	$\{80\}$

In scenario 3, shown in Table 3.3, the two sensors are in total conflict with a reading of 90 km/h from the Google map and readings of 100 km/h and 80 km/h from the camera. The result is a high conflict factor of $k = 0.54$ and the highest mass value of $m_{Google,cam}(\{100\}) = 0.6087$ resulting in a best estimation of $s_{be} = 100$ km/h. The fused mass values are lower than the individual mass values for each sensor since the sensors are in conflict.

Table 3.3: Camera and map data fusion scenario number 3.

	m_{Google}	m_{cam}	$m_{Google,cam}$
$\{80\}$	0	0.2	0.1739
$\{90\}$	0.6	0	0.1304
$\{100\}$	0	0.7	0.6087
$\{\theta\}$	0.4	0.1	0.0870
k	0	0	0.5400
s_{be}	0	0	$\{100\}$

3.3.4 Performance evaluation

To evaluate the performance of the DS-fusion between the camera and the Google map, six simulations are executed for different countries. Each simulation consists of signal pre-processing for the Google- and vision data, DS-fusion and performance evaluation where the performance was evaluated in terms of true positive distance TP_D defined in (1.1). The tuning parameters are set to $\alpha_{Google} = 0.5$ and $\alpha_{cam} = 0.5$ which gives equal reliability for the Google map and the camera. The tuning parameters are not optimized in any way but set to equal values to give a first indication of the performance.

Table 3.4 shows the TP_D for the camera, Google map and the fused result. The conclusion is that DS-fusion improves the performance of the individual sensors for log files 1,2,3 and 6. The performance for log files 4 and 5 are however decreased which indicates that a different tuning setup is needed for these countries.

Table 3.4: Performance (TP_D) of the camera, Google map and DS-fusion between the Google map and the camera.

Log	Country	Driving distance [km]	Camera (%)	Google (%)	Google & Camera (%)
1	Germany	103.6	41.4	64.4	66.2
2	Sweden	181.4	48.4	94.0	96.8
3	Netherlands	110.3	64.9	95.5	95.6
4	France	316.2	33.5	87.6	86.6
5	Luxembourg	56.3	33.3	86.0	84.7
6	Belgium	42.3	33.3	80.6	80.7

3.4 Additional map data source

As a part of the proposed offline model, defined in Section 3.1, an external digital map data source is to be added. The requirement for the external digital map was to provide speed limits given map-matched position coordinates with coverage of all relevant road networks in Europe. One digital map supplier that offers this is Here Technologies [57]. The data provided by Here is available through an open API where road attributes can be extracted directly. To simplify the process further, a function called *Here Route Matching v8* was used to calculate the MPP and extracting the speed limit from the EH [58]. The input to *Here Route Matching v8* is a GNSS-trace, headings, time stamps and vehicle velocities which can be extracted from the log files collected by the test vehicles.

3.4.1 Dempster-Shafer implementation

The process of adding the Here-map consists of four steps, shown in Figure 3.2. The first step was to retrieve GNSS-traces, headings, timestamps and vehicle velocities from the test vehicle data for each time step. The second step was to process the data and create a CSV-file that can be used as input to the *Here Route Matching v8* function. The third step was to run *Here Route Matching v8* to get map matched positions and corresponding speed limits for each time step.
The output speed limit output from the Here map can then be used to construct mass values in a similar way as for the Google map but with a different tuning factor, α_{Here}. This is illustrated in (3.10).

$$\begin{aligned} m_{Here}(\emptyset) &= 0 \\ m_{Here}(\{s_{Here}\}) &= \alpha_{Here} \\ m_{Here}(\theta) &= 1 - \alpha_{Here} \end{aligned} \tag{3.10}$$

The Google-, camera- and Here-data can then be combined by applying Dempster's rule of combination in two steps:

$$m_{Google,cam,Here} = (m_{Here} \oplus (m_{Google} \oplus m_{cam}))(x) \tag{3.11}$$

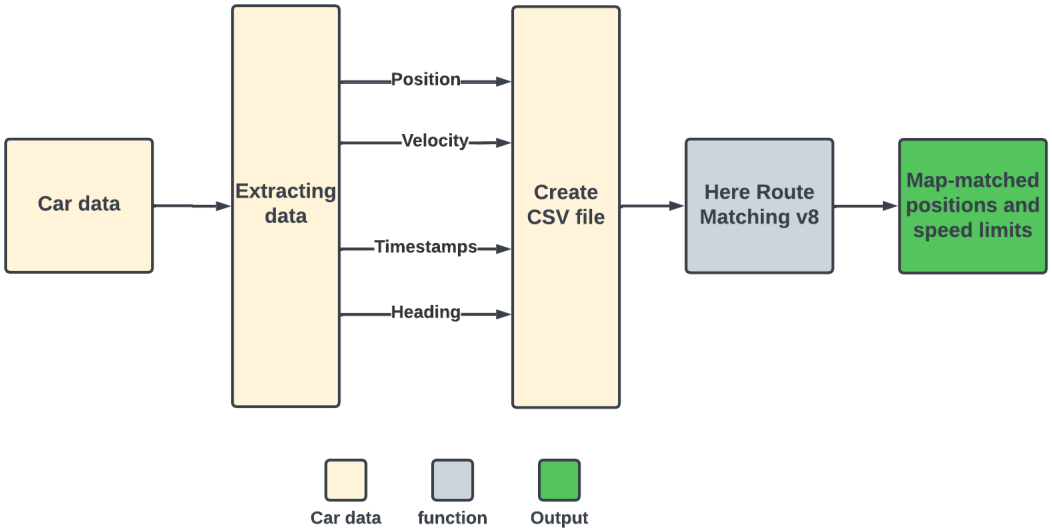


Figure 3.2: Process of extracting speed limits from the Here map.

One example of a multiple fusion between the three data sources is shown below in Table 3.5, where both the Google map and the Here map retrieves a reading of 40 km/h with a mass value of 0.8 and 0.65 respectively. The vision gets a reading of 30 km/h with a mass value of 0.9. The result of only fusing the Google- and vision-data is a speed limit of 40 km/h but with a probability of 0.2857 that it also could be 30 km/h. Fusing the result with the Here data gives a best estimation of 30 km/h. The result illustrates that having two sources with lower mass values (0.8 and 0.65) provides more evidence for a proposition compared to having one source with a high mass value (0.9) when using DS in this case.

Table 3.5: Mass value output by performing DS-fusion between the camera, the Google map and the Here map.

	m_{Google}	m_{cam}	m_{Here}	$m_{Google,cam}$	$m_{Google,cam,Here}$
$\{30\}$	0.8	0	0.65	0.2857	0.5706
$\{40\}$	0	0.9	0	0.6429	0.3865
θ	0.2	0.1	0.35	0.0714	0.0429
k	0	0	0	0.72	0.4179
s_{be}	0	0	0	$\{40\}$	$\{30\}$

3.4.2 Performance evaluation

The same six log files as shown in Table 3.4 were used to evaluate the performance after adding the Here map. The tuning parameter is set to $\alpha_{Here} = 0.5$ which is the same value used for the Google map and the camera. The results show an increased performance for all log files except the one collected in the Netherlands, log file 3. This is explained by the speed regulations on highways in the Netherlands where 100 km/h is valid during the day (06:00-19:00) and 120 km/h or 130 km/h is valid

during the night (19:00-06:00) [59]. The Here map does not take this into account, which results in a worse performance after adding it. As shown in Figure 3.3, the red graph of the upper plot illustrates the speed limit from the Here map, which gives incorrect outputs of 120 km/h and 130 km/h. This results in some disturbances in the fused result which can be seen by the blue line representing the offline speed limit in the lower plot. Some odd values can also be seen in the upper plot, where the value 0 km/h corresponds to no speed limit value from the camera, and 250 km/h corresponds to unlimited speed limit.

Table 3.6: Performance (TP_D) of the DS-fusion between the Google map, the vision and the Here map.

Log	Country	Driving distance [km]	Google & Camera (%)	Google & Camera & Here (%)
1	Germany	103.6	66.2	68.6
2	Sweden	181.4	96.8	97.4
3	Netherlands	110.3	95.6	92.5
4	France	316.2	86.6	89.1
5	Luxembourg	56.3	84.7	87.8
6	Belgium	42.3	80.7	83.5

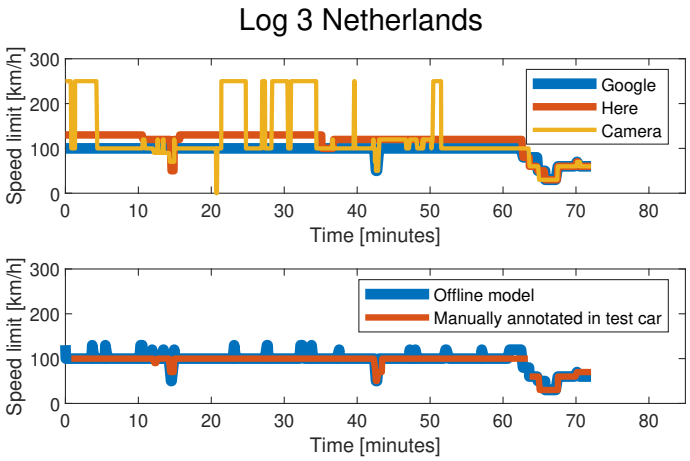


Figure 3.3: Speed limit estimations for log 3. Data collected in the Netherlands.

Adding the Here map can improve the performance of the speed limit estimations, however it can degrade the performance as well, since the map quality seems to vary for different countries. The solution is to tune α_{Here} differently for each country.

3.5 Road sign classification and detection implementation

In the test car, there is a forward-looking camera which records a video while driving. These videos can be used as input to an offline speed sign detection and classification algorithm, which provides the offline model with an additional vision data source. By implementing a detection and classification network and adding it to the offline model, the correctness of the model could be improved. Further, this provides an insight into how object detection and classification algorithms works in practice. The following section aims to show that YOLO as a network architecture is a suitable choice for the purpose of detecting and classifying speed signs. The network will be trained using a dataset which only contains Swedish speed signs. The reason why a Swedish dataset is used is the fact that lots of videos from driving in Sweden are available from the test cars, and therefore these are suitable to use when testing the detection and classification model.

3.5.1 Network architecture

In order to detect and classify speed signs a convolutional neural network architecture called Tiny YOLOv2 was used. As the name suggests, this network is a reduced version of the YOLOv2 architecture. A smaller network should reduce both the risk of overfitting as well as the training time, albeit at the cost of some potential performance. The model used is pre-trained on the Common Objects in Context (COCO) dataset, containing over 1.5 million object instances [60].

3.5.2 Training dataset

To train the model, a Swedish road sign dataset is used [61]. Since our network model aims to solely detect and classify speed signs, these signs were extracted from the dataset. Extracting only images with speed signs made the dataset significantly smaller, and therefore some additional images were manually annotated. The images used for this were sampled from the Mapillary Street-Level Sequences dataset (MSLS) [62]. This dataset consists of street-leveled images with geographical data that are crowd-sourced, making the dataset very large (over 1.6 million images in June 2020). Computer vision technologies have then been used in order to match these images to a digital map, which can be seen on Mapillary's webpage [63]. 285 images were manually annotated, creating a total data set of 1283 images.

Since the dataset is small compared to other datasets used to train YOLO models, the expected performance is limited. The mean number of images in each class is 117, significantly less than the PASCAL VOC datasets that were used to fine-tune the original YOLO model [24], which has about 2000 images per class [31], [32].

A random sample from both the Swedish dataset and the MSLS dataset can be seen in Figure 3.4, where it is possible to see that the images have similar characteristics. As illustrated in Figure 3.5, some sign types are over-represented. The cause of this

Figure 3.4: Left: sample from the MSLS dataset [62]. Image by Magol is licensed under CC-BY SA 2.0. Right: sample image from the Swedish dataset [61].

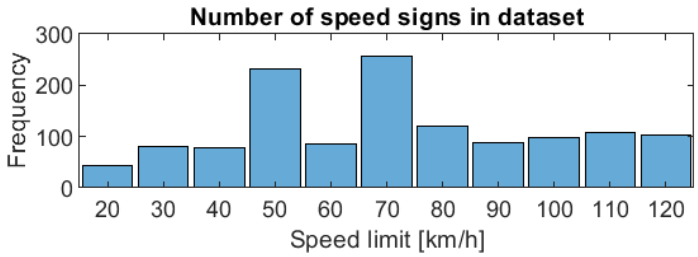


Figure 3.5: Histogram showing frequency of each sign type in the dataset.

is likely that some signs are more common, and that more images can be captured if the signs are passed slowly. This distribution might lead to overfitting to the most common sign types in the dataset.

3.5.3 Anchor box estimation

Since anchor boxes are crucial hyperparameters in the network which has to be estimated. By using the bounding boxes in the training data, the mean IoU can be computed to get an estimation of how well the suggested anchor boxes fit the bounding boxes in the training data. As can be seen in Figure 3.6, a higher number of anchor boxes gives a higher IoU. However, choosing a high number of anchor boxes increases the computational complexity of the network [28]. Therefore five anchor boxes seem like a reasonable trade-off.

3.5.4 Network training

To train the model, augmented training images were used. This is done in order to synthesize variations in speed signs and hence decrease the risk of overfitting. For training, 80% of the total dataset was used. The images were scaled up or down randomly as much as 10%, and the contrast, saturation and brightness were adjusted

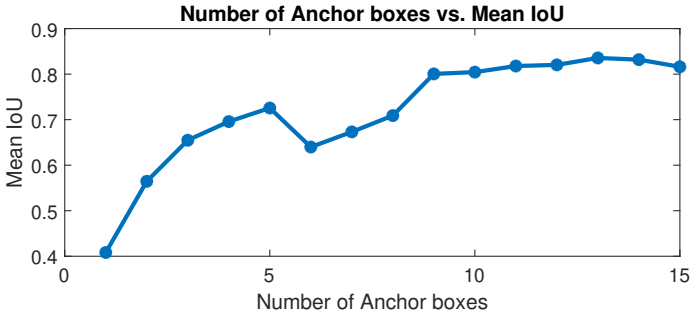


Figure 3.6: Intersection over union for different number of anchor boxes.

randomly as well. To be able to stop training before the model overfits, a validation dataset was used, making up to 15% of the total dataset. These images are not used in training and can be used to compute the validation loss, which gives an unbiased evaluation of the model's performance. Finally, a test set was also created for the final assessment of the model's performance, consisting of 5% of the total number of images. The model was trained for 200 epochs until the validation loss did not decrease anymore, a process which took about 20 hours using a Nvidia GeForce RTX 2060s graphics card.

3.5.5 Network evaluation

The model's performance was evaluated from three perspectives: detection accuracy, classification accuracy and inference time. The detection accuracy, meaning the model's ability to detect where in the image the speed sign is, was checked visually using the images in the test set. This also provides a sense of how well the predicted bounding boxes fit the speed signs. A montage of the predicted bounding boxes plotted over their corresponding speed signs can be seen in Figure 3.7. What is interesting to note from this montage is that some of the speed signs are of such low resolution that it is difficult to even manually distinguish the speed limit of the sign. This means that the dataset is perhaps not ideal to train a classifier on, since the digits on the signs need to be clearly visible for the classifier to learn the right features. Overall, the model predicts bounding boxes that fit the speed signs well. It is crucial for the classification part that the bounding boxes contain the entire sign, so that all digits on the sign are visible. An example of when this does not occur can be seen in the top left image in Figure 3.7, where the necessary features for an accurate classification are not present within the bounding box.

The classification accuracy is tested by comparing the predicted speed sign classes with the true speed signs, using the images in the test set. Since the model can predict multiple overlapping bounding boxes and labels at once, non-maximum suppression was used in the final step of the process. For each bounding box, the product between the objectness and the classification score is computed. This product gives an indication of the combined confidence of both the detection and the classification. This value can be used to threshold out predictions where the score was too low, effectively removing uncertain predictions.

Figure 3.7: Bounding boxes predicted by the YOLO network.

How the threshold value is set affects the performance of the model. A high threshold value means that only high-scoring predictions are used, which might filter out correct ones that have a low score. On the other hand, a low threshold leads to the use of low-scoring predictions which are more likely to contain incorrectly classified speed signs, certainly an unwanted behavior. To evaluate the classification accuracy, the precision and recall can be computed. The precision states how many of the predictions that are correct divided by the total number of predictions. This is important since false positive classifications are undesirable. The recall computes how many of the predictions that are correct divided by the total number speed signs. The test set consists of 92 speed signs distributed over 66 images. The model makes 58 correct and 12 incorrect predictions of the true speed limit in the images. Multiple signs can be detected in the same image. This gives the following metrics, seen in (3.12).

$$\text{Precision} = \frac{58}{58+12} \approx 0.83 \qquad \text{Recall} = \frac{58}{92} \approx 0.62 \tag{3.12}$$

From the precision value, it is possible to see that when the model classifies a speed sign, it is on average correct about 83% of the time. The model does however miss a fair number of signs. As can be seen in (3.12), only about 62% of the speed signs in the test set are detected and classified correctly. However, this might not be as big of an issue as it seems, since the camera on the front of the car captures multiple frames when a speed sign is passed. This means that the model gets multiple chances to make a prediction. Figure 3.8 shows an image sequence from when a 70-sign is passed. Here the model makes predictions in four sequential frames.

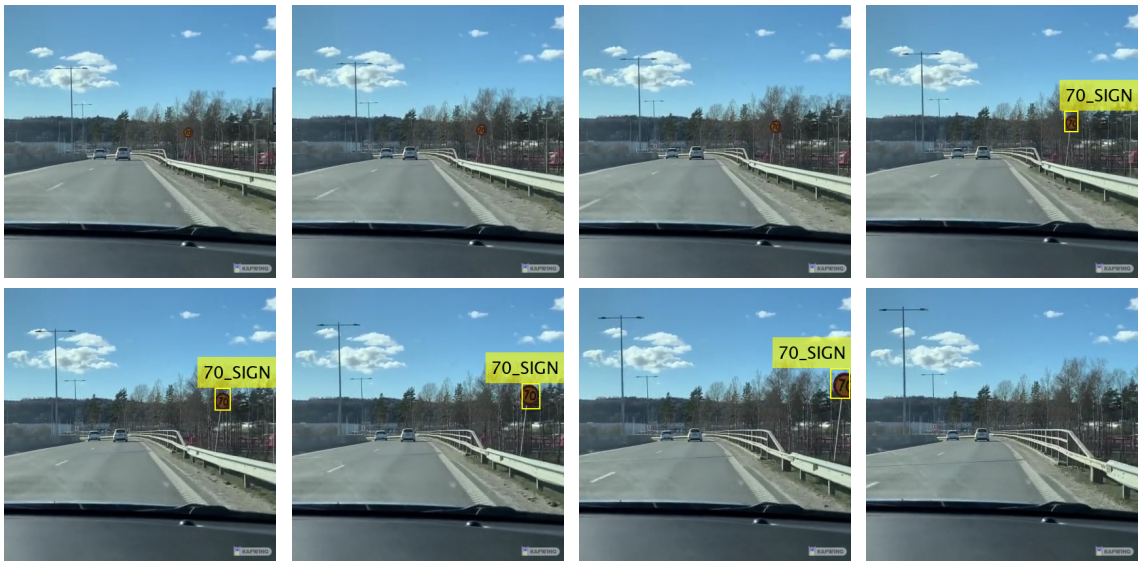


Figure 3.8: Video sequence where 70-sign has been detected and classified in four subsequent frames.

The inference time is the time the model takes to process an image and compute a prediction. To get a reference of how fast the YOLO network runs, a R-CNN network was also trained for the same purpose [23]. The YOLO network has 119168 parameters, and the R-CNN network has 119436 parameters, meaning both networks have roughly the same number of parameters. The mean inference time was computed when the networks run on a large set of images. The YOLO network has a mean inference speed of 94 frames per second using the Nvidia GeForce RTX 2060s graphics card. The RCNN network obtains just 5 frames per second on the same setup. This means that YOLO runs over 18 times faster than R-CNN. This difference makes YOLO feasible to run in real-time and to be used on the videos from the test vehicle.

To summarize the evaluation of the network performance, the following can be said about detection, classification and inference.

- When a detection is made with reasonable confidence, the speed sign is most often contained within the bounding box.
- A more diverse dataset is required to generalize the network.
- The low inference time makes YOLO feasible to run at real time.

This leads to the conclusion that using YOLO for speed sign detection is a promising method, but it will not be incorporated in the offline model. The main reason for this is because the network does not seem to generalize well to images outside of the test set. The precision, shown in (3.12), also indicates that the classification does not work flawlessly.

Previous work shows that YOLO networks trained on large and diverse datasets generalize well [24], meaning that this should be possible speed signs as well. However, it is beyond the scope of this work and would likely require a larger and more

diverse dataset than the one used. More computational resources would also be required since the training time would increase tremendously.

Finally, it was found that manually annotating images and adding to the dataset was a feasible method to increase the amount of data, and that the MSLS dataset was a suitable source of additional images [62]. The Swedish dataset is not ideal for training a classifier since some of the images are of such low resolution that the digit features become indistinguishable.

3.6 Off-ramp road sign filtering

One source of error in the RSI function is the case when speed signs belonging to adjacent roads are detected by the camera system. This can for example happen when an off-ramp is passed. By finding out when an off-ramp is passed, the speed sign detections in that region can be filtered out by decreasing the mass value from the camera reading. This was done by adding a tuning parameter, $\alpha_{off-ramp}$, which takes a value between 0 and 1. This tuning parameter is then multiplied with α_{cam} to decrease the mass value when an off-ramp is present. The corresponding mass value calculation is illustrated in (3.13).

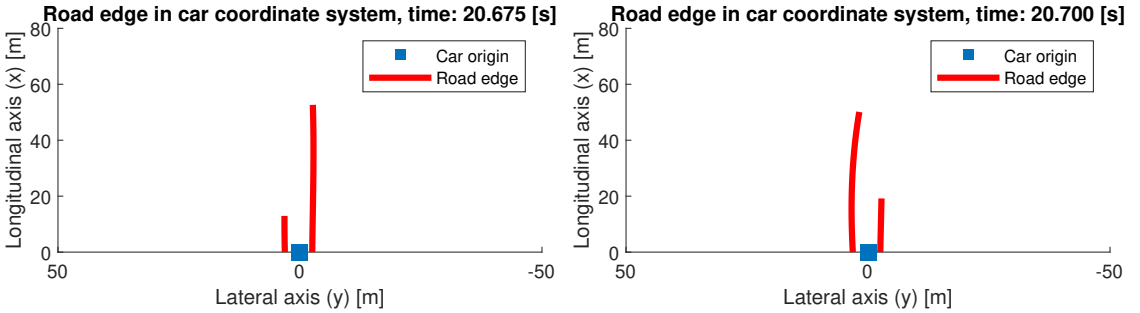
$$\begin{aligned} m_{cam}(\emptyset) &= 0 \\ m_{cam}(\{s_{cam,i}\}) &= (\alpha_{off-ramp} \cdot \alpha_{cam}) \cdot \frac{1}{n} \cdot c_{cam,i} \\ m_{cam}(\theta) &= 1 - (\alpha_{off-ramp} \cdot \alpha_{cam}) \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} c_{cam,i} \end{aligned} \tag{3.13}$$

This solution relies on first finding the off-ramps. To do this, two methods have been tested. One based on camera data, and one based on data from the GPS and digital map. In the camera-based method, the road edges are detected and then modeled using clothoids. This method is described in detail in Section 3.6.1. In the other method which can be seen in Section 3.6.2, the GPS position is combined with digital map data to compute the distance to the intersection. The performance of both methods is evaluated in Section 3.6.3.

3.6.1 Off-ramp detection using road edges

To filter out the speed sign detections close to off-ramps, the road edges that are detected by the camera system are modelled as clothoids. The theory behind modelling the road edges is covered in Section 2.4.4. The camera detections are used to reconstruct polynomials which describes the shape of both the left- and right road edge. The results of reconstructing the road edges in a couple continuous time steps can be seen in Figure 3.9, where the blue square denotes the origin at the car's rear axis.

To filter out off-ramps using the road edge detections, a case where an off-ramp with a different speed limit than the main road was studied, illustrated in Figure 3.10. Here the speed sign belong to the off-ramp (dark grey road segment) but is



(a) Road edges reconstructed in first time step

(b) Road edges reconstructed in second time step

Figure 3.9: Road edges reconstructed in two subsequent time steps.

positioned similarly as if it belonged to the main road (light grey road segment). A human driver could easily understand this, but the RSI system might struggle in this scenario. If the sign is classified with high enough confidence, the speed limit 70 km/h is incorrectly shown to the driver by the RSI function. This sequence of events can be seen in Figure 3.11, where the RSI function starts showing the incorrect speed limit in the last time step.

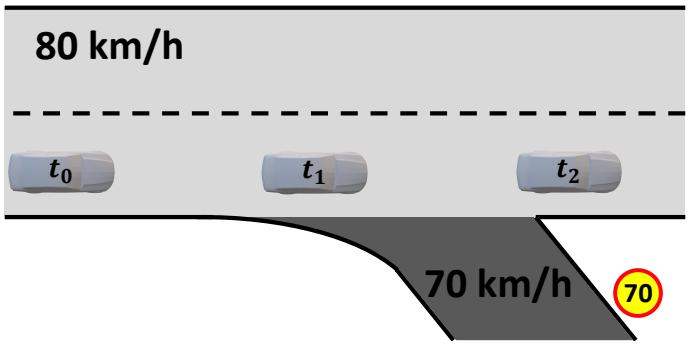
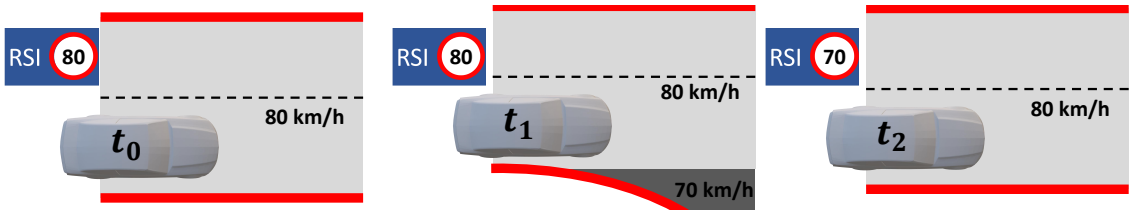


Figure 3.10: Car passing an off-ramp and sampling data in three different time steps. The light grey road segment has a speed limit of 80 km/h, and the dark grey a speed limit of 70 km/h. The speed sign is positioned such that it belongs to the off-ramp.

By illustrating the road edges seen by the car in different time steps when passing an off-ramp, conclusions can be drawn about how the shapes of these could be used. In Figure 3.11, it can be seen that before and after the off-ramp (t_0 and t_2), the road edges are straight, but just when passing the off-ramp (t_1), the right road edge clothoid turns to the right.

By looking at the right road edge in Figure 3.11b, it can be seen that it turns to the right in relation to the car. By computing the curvature for the right road edge, defined in (2.13), a negative value would be obtained. This could potentially be used to filter out off-ramps.



(a) RSI correct. **(b)** RSI correct. **(c)** RSI incorrect.

Figure 3.11: Road edges (red) detected by the car for the three different time steps shown in Figure 3.10.

To use the curvature of the road edges in order to detect possible off-ramps, a couple of conditions have to be formulated, where κ_t denotes a threshold value used to figure out if the road edge is turning enough. Positive curvature values mean the road edge turns to the left, and negative values mean that the road edge turns to the right. If both of the conditions shown in (3.14) are fulfilled, then the off-ramp case shown in Figure 3.11b should be possible to detect. The first condition shown in (3.14) tests if the right road edge turns to the right in relation to the direction that the car is driving. The second condition shown in (3.14) tests if the left road edge is relatively straight. This is done using two curvature values denoted κ_{right} and κ_{left}, which are the curvature of the left and right road edge respectively. A positive threshold value κ_t is used to

$$\kappa_{right} < -\kappa_t, \qquad -\kappa_t < \kappa_{left} < \kappa_t \tag{3.14}$$

As can be seen in Figure 3.9, reconstruction sometimes differs a lot between two subsequent time steps, in particular regarding how far ahead of the car the clothoid model is valid. By plotting this value for a sequence of ten seconds, which is shown in Figure 3.12, it can be seen that the values are rapidly varying. Janda *et al.* states that in their road edge detection method, individual image frames from the video are used to extract reference points [36]. This can cause the reference points to vary a lot between each frame. From these reference points the road edges are then modelled. It is reasonable to assume that something similar is done here, and therefore the maximal distance for which the clothoid model is valid depends on the reference points that the camera system is able to find in each frame. Since each frame is processed individually, it is not unreasonable that the result varies a lot between each image, as can be seen in Figure 3.12. Another notable aspect about Figure 3.12 is that sometimes the road edge cannot be modelled at all, which happens when the maximal distance is equal to zero. This is likely a result of the camera detection system not finding enough valid reference points. Both the rapidly varying character of the data, and the fact that occasionally the road edges cannot be modelled at all, makes off-ramp modelling using road edges challenging.

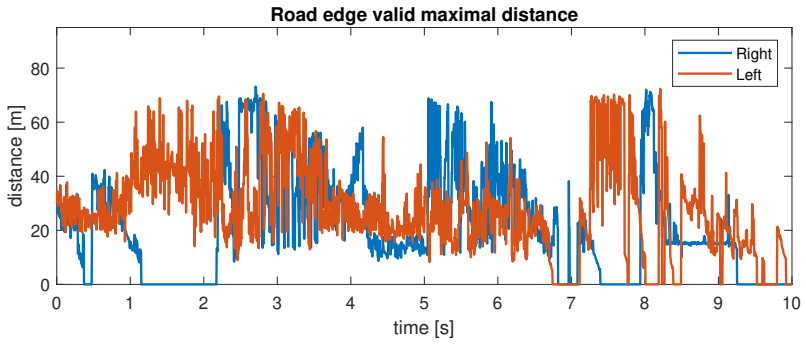


Figure 3.12: Maximal distance for which the road edge is valid for a driving sequence of 10 seconds. The variable shown here is used as x_t in (2.12).

3.6.2 Off-ramp detection using map data

By combining the vehicle position from the GPS with data from Google maps, the distance to the next intersection can be obtained. The intersection data is one of the attributes provided by the EH for the MPP by Google. By combining this data, it can be detected if the car is in the vicinity of an off-ramp.

The logic used was to first filter out off-ramp intersections and then use the distance to intersection as an indicator of when the vehicle is close to an off-ramp. To compensate for possible inaccuracies in the map data, a distance of 50 m around the intersection will be used to define an off-ramp area. This scenario is illustrated in Figure 3.13 where the car at time t_1 and t_2 is considered to be inside the off-ramp area. The main issue with using this approach is that the map matching and MPP prediction need to be correct. Incorrect MPP predictions will lead to off-ramp indications for wrong road segments compared to the actual trajectory of the car.

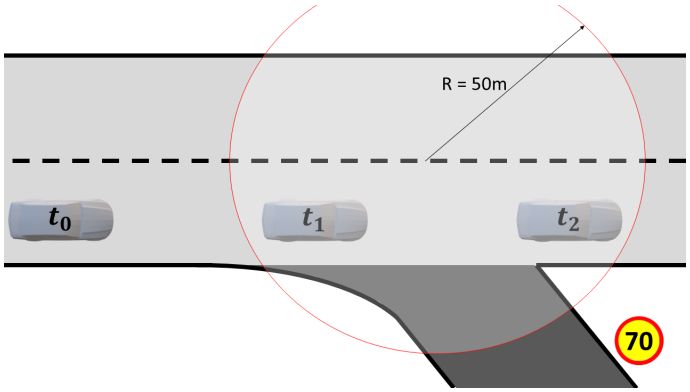


Figure 3.13: Illustration of the 50 m radius around an intersection indicating an off-ramp area.

3.6.3 Off-ramp road sign filtering evaluation

Two different log-files from the test vehicle were used to evaluate the performance of the off-ramp detection systems. The specification of each log is shown in Table

3.7. These were selected since they contained many off-ramps.

Table 3.7: Specifications of data log files used for off-ramp filtering testing.

	Driving distance	Country	Start location	End location	Number of off-ramps
Log 1	182.1 km	Sweden	Volvo Torslanda	Strömstad	38
Log 2	172.6 km	Sweden	Volvo Torslanda	Lidköping	39

By running both off-ramp detection systems on the log files, the systems makes predictions about whether there exists an off-ramp or not at each time steps. To evaluate if these predictions were correct, the output from the functions was inserted into Quantum Geographic Information System (QGIS) which is an open source platform for geographic information were the full trajectories could be visualized. One example is shown in Figure 3.14. Here, both the map- and the road edge system detects the off-ramp successfully. The red dots belong to the map system and mean that the car is within 50 m of the point that Google maps defines as the center of the intersection. The blue dot is where the road edge system has detected an off-ramp. The green dots means that neither system has detected an off-ramp. The evaluation

Figure 3.14: Output from the off-ramp-detection systems. Green dots mean no off-ramp detected, red dots indicate an off-ramp detected by the map system and blue dots indicate an off-ramp detected by the road edge system.

was done manually for the full trajectories using the QGIS visualization illustrated in Figure 3.14. A detection is considered to be true if it is within 50 m of the center of the off-ramp.

Two performance metrics are used for the off-ramp systems. The precision captures how often the retrieved off-ramp detections are correct, and the recall how many of all true off-ramps that are detected. The result is shown in Table 3.8. Since the evaluation is performed manually, the metrics might have some errors and can only give a rough estimation of the performance.

	Precision: Map (%)	Recall: Map (%)	Precision: Road edge (%)	Recall: Road edge (%)
Log 1	100	94.7	16.3	21.0
Log 2	94.6	89.7	8.5	10.3
Total	**97.4**	**92.5**	**12.5**	**15.4**

Table 3.8: Precision and recall for the map- and road edge based off-ramp detection systems. Combined values for both log files are shown in the last row of the table.

The conclusion from the evaluation is that the precision and recall is significantly lower for the road edge system compared to the map system for both log files. The low precision for the system can be explained by many false positive cases due to wrong road edge geometry estimations. One example of this is illustrated in Figure 3.15 where the camera detects a road edge (red lines) on a side road which gives a false indication of an off-ramp using the conditions in (3.14). The result from

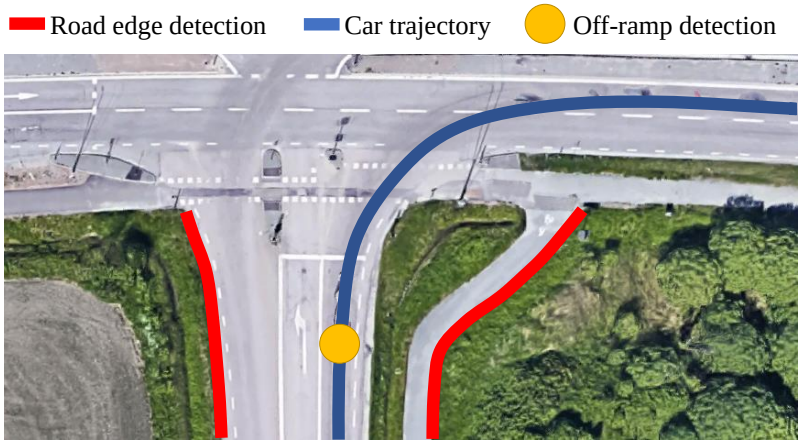


Figure 3.15: Detection of road edge on side road. Adapted from [64].

the evaluation gives a clear indication that the map-system should be used as an off-ramp detector when constructing the offline model. This means that no further improvements was done on the road edge system. The reason for this is because the input data for re-constructing the road edges is inconsistent and sometimes inaccurate, for example in the case when a road edge of a side road is detected.

3.7 System architecture

This section presents the final system architecture used for the offline model. The choice of which subsystems to be included in the offline model was based on the performance of each subsystem, evaluated in Section 3.3-3.6. The final system architecture is illustrated in Figure 3.16.

The final architecture is similar to the proposed architecture defined in Section 3.1 but with some changes. First of all, the YOLO network for speed sign classification showed limited performance. Therefore, it is not considered to be able to enhance

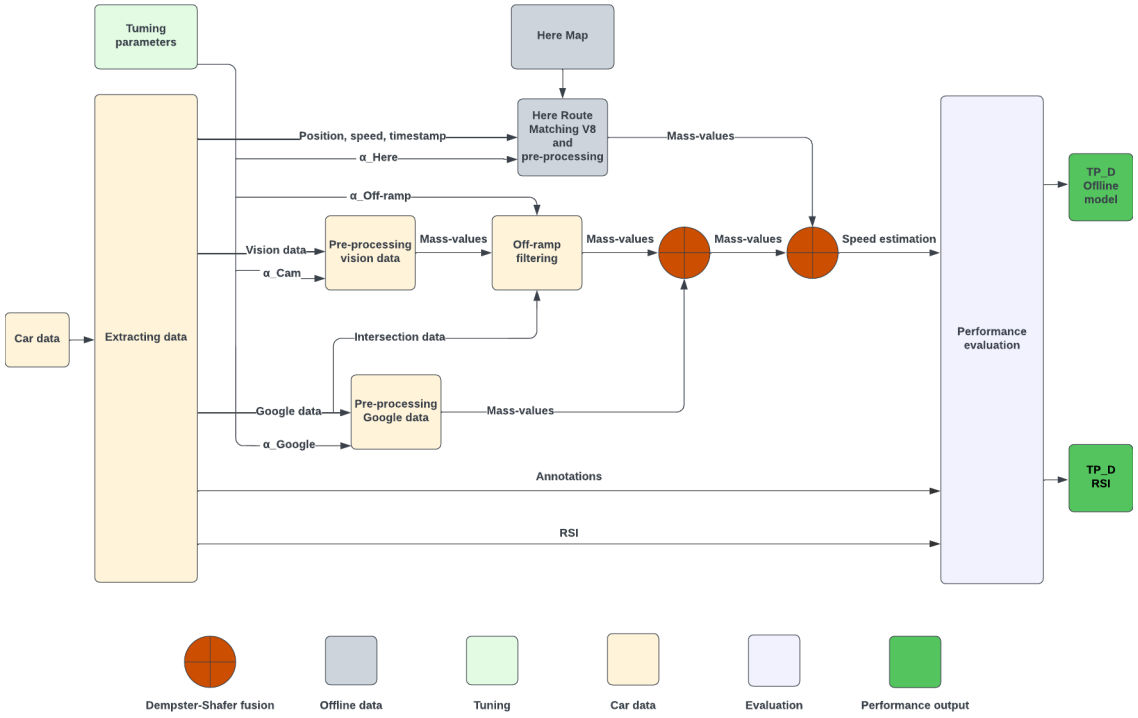


Figure 3.16: Final system architecture.

the performance of the offline model and is hence excluded. The second change is the addition of tuning parameters as input to the offline model, which gives the possibility to tune the reliability of each sensor differently for each country. The tuning is done using an evolutionary algorithm that updates the parameters in each iteration until a satisfactory result is reached. This process is illustrated in Figure 3.17.

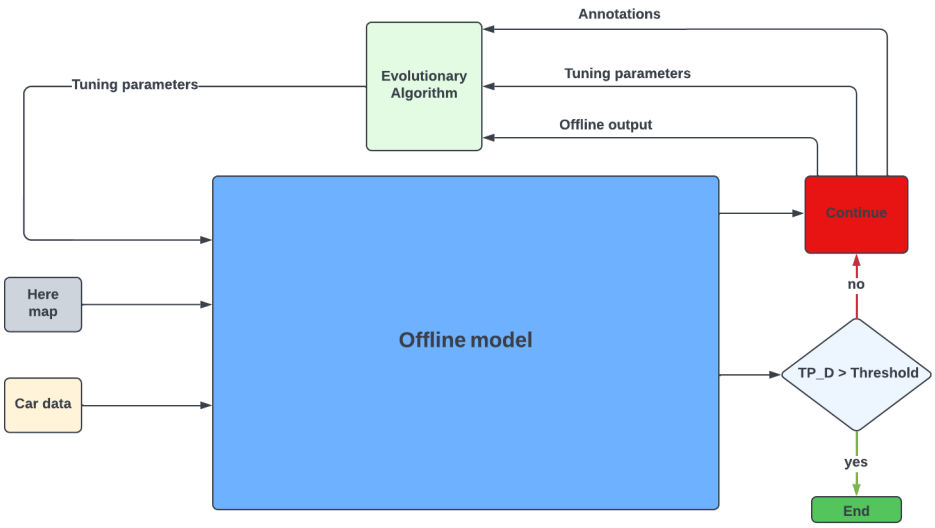


Figure 3.17: Optimization of tuning parameters.

3.7.1 Parameter tuning and performance evaluation

The offline model was evaluated using log files from several different countries. For each country, some of the log files are used to tune the parameters, and the rest of the log files are used to test the offline model. About 30% of the total driving distance was used for tuning, and log files are selected accordingly. Of course, more test data can be used for tuning, but this also means less data will be available for evaluation, which makes it harder to draw general conclusions from the result.

The tuning parameters affect to what degree each data source is trusted. Country-specific tuning is necessary since the quality of the data sources vary by country. For example, in some countries one digital map is more reliable than the other. The same applies for the vision-based data source, whose reliability also varies between different countries. This could be because the appearance and frequency of speed signs differs slightly between different countries, which affects the performance of the sign detection system. The values of the tuning parameters give an indication of how reliable each data source is in each country.

Because two maps and one camera are used, there are three tuning parameters which indicate the reliability of the different data sources: α_{Google}, α_{HERE} and α_{cam}. If any of these have a high value, it indicates that the data source should be used to a large extent in order to obtain good performance.

Off-ramps might entail that speed signs on adjacent roads, which do not apply to the road that the car is driving on, are detected. Therefore an off-ramp tuning parameter is used, denoted by $\alpha_{off-ramp}$. This parameter affects to what degree the mass value of the speed sign detections from the vision are decreased when an off-ramp is passed, as described in Section 3.6. A low $\alpha_{off-ramp}$ decreases the trust for the vision data while $\alpha_{off-ramp} = 1$ has no effect on the mass value.

4
Results

In the following chapter, the performance of the offline model is presented for several European countries. The performance of the model is measured using the metric true positive distance, TP_D, shown in (4.1). It is defined as the distance with correct annotated speed limit (d_{correct}) divided by the total distance (d_{total}) driven during a test drive [5]. To put the performance of the offline model in context, it is compared with the performance of the RSI function in the car. Since the performance of the RSI function in the car depends on the Google map, which is continuously updated, the results may differ depending on the time that the log files are collected.

$$TP_D = \frac{d_{\text{correct}}}{d_{\text{total}}} \tag{4.1}$$

The motivation behind evaluating the performance country by country is the changes in quality of the data sources. This shows that the model can adapt to the quality of various data sources by adjusting the reliability tuning parameters. Besides the performance, the values of the reliability tuning parameters are also shown, which gives an indication of how reliable each data source is in each country. Furthermore, specifications about the log files used for parameter tuning and testing are shown. The total driving distance for each country gives an indication of how well-founded the conclusions drawn for each country are. Countries with lots of data, such as Sweden, can provide a more thorough analysis. Finally, some log files are analyzed in depth. This aims to illustrate why the offline model works well for some scenarios but struggles for others.

4.1 Sweden

To tune the parameters in the model for Sweden, two log files have been used, which can be seen in Table 4.1. The log files used for tuning contain 28.7% of the total driving distance for Sweden. The rest of the Swedish log files are used for evaluation of the offline model which is compared with the RSI function in the car.

Table 4.1: Log files from Sweden for parameter tuning.

Log	Time [min]	Distance [km]	Country
1	137	172.0	Sweden
2	113	102.1	Sweden

In (4.2), the parameter values obtained when tuning the model for Swedish log files are shown.

$$\alpha_{cam} = 0.44, \qquad \alpha_{Google} = 0.44, \qquad \alpha_{Here} = 0.45, \qquad \alpha_{off-ramp} = 0.69 \tag{4.2}$$

All tuning parameters have approximately the same value which indicates that all data sources are equally reliant in Sweden. The off-ramp tuning parameter is used to temporarily decrease the reliability of the camera when an off-ramp is passed, by multiplying the camera reliability tuning parameter with the off ramp tuning parameter. For the parameter values presented in (4.2), both maps have a higher reliability than the camera during off-ramps. The performance is shown in Table 4.2. The total performance is 97.8% for the offline model which is an improvement compared 96.1% for the RSI. The offline model performs better for all log files except for log 6 and 8. The difference between the TP_D of the RSI function and the offline model is shown in the rightmost column in the table below, where positive values indicate that the offline model performs better than the RSI function.

Table 4.2: Performance evaluation for log files from Sweden.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	132	181.4	97.4	97.9	0.5
2	87	134.3	94.2	99.4	5.2
3	65	75.8	96.0	98.7	2.7
4	108	101.3	96.5	97.3	0.8
5	84	79.2	96.5	97.5	1.0
6	35	23.0	93.3	90.5	-2.8
7	77	48.0	93.9	97.7	3.8
8	30	39.4	99.7	96.7	-3.0
Total	**618**	**682.4**	**96.1**	**97.8**	**1.7**

Two log files are analyzed in depth. The offline model performs poorly for log file number 8, which is shown in Figure 4.1. Small errors have a large effect on the true positive distance since the total driving distance is relatively short for this log file. By looking at Figure 4.1, the incorrect output of the offline model can be explained. Since the reliability tuning parameters are similar for all three data sources, the model often choose the speed limit of the data sources that agree. Therefore, the main source of error is when two data sources agree on an incorrect speed limit. This happens at around minute 16 and 20, which can be seen in Figure 4.1. At minute 26-29, an off-ramp is detected which decreases the reliability of the camera. This is the reason why the offline model does not always trust the two data sources that agree during this time sequence.

For log file number 2, the offline model performs very well and obtains an accuracy of 99.4% which is shown in Figure 4.2. In this case, two data sources agree most of the time and therefore the model obtains a satisfying performance. Here, the model successfully filters out the incorrect speed limit predictions from the camera system at minute 40 to 45.

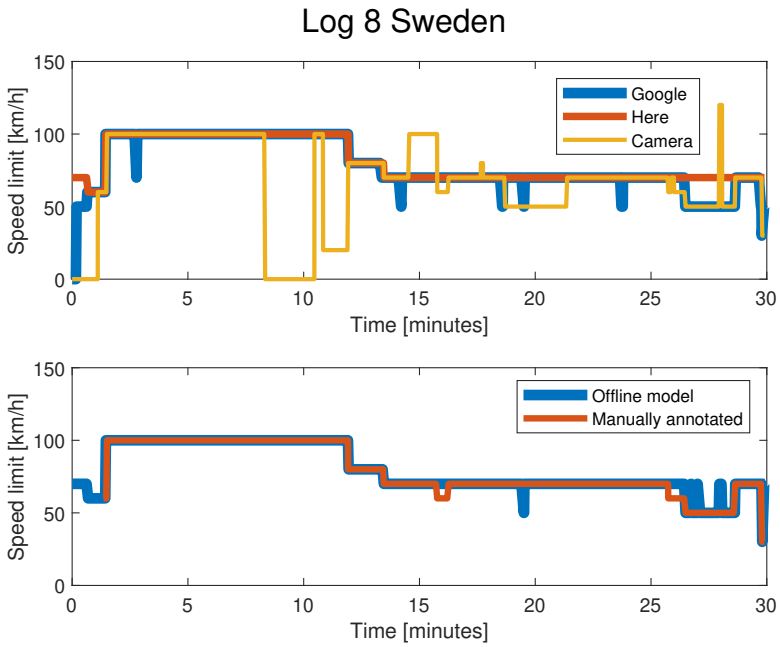


Figure 4.1: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth.
Data collected in Sweden.

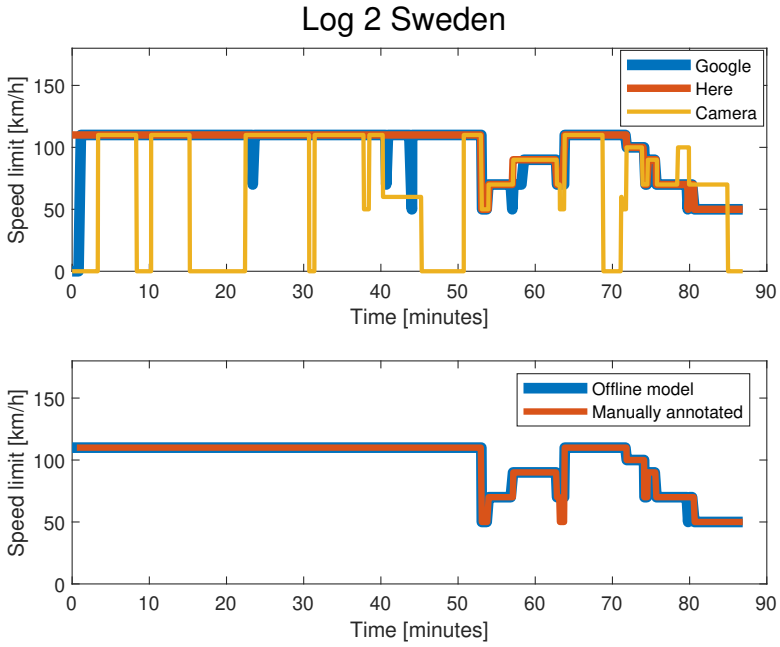


Figure 4.2: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth.
Data collected in Sweden.

4.2 The Netherlands

The specification of the tuning log for the Netherlands is shown in Table 4.3. The driving distance for the tuning log is 27.2% of the total driving distance.

Table 4.3: Log files from the Netherlands for parameter tuning.

Log	Time [min]	Distance [km]
1	82	100.4

The tuning paramaters for the Netherlands are shown in 4.3.

$$\alpha_{cam} = 0.41, \qquad \alpha_{Google} = 0.54, \qquad \alpha_{Here} = 0.18, \qquad \alpha_{off-ramp} = 0.97 \tag{4.3}$$

The Google map tuning parameter gets the highest reliability followed by the camera and lastly the Here map. This indicates that the Google Map is a reliable data source in the Netherlands and should be used to a large extent in the offline model to obtain good performance.

As can be seen in the total performance of Table 4.6, the offline model performs significantly better than the RSI function. The RSI function worked poorly at times during the test runs, due to software issues in the AHR. This caused an incorrect speed limit to be delivered to the RSI function. These errors were corrected by using the offline reconstructor of the AHR, discussed in Section 3.3.1. The overall

Table 4.4: Performance evaluation for log files from the Netherlands.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	43	73.2	61.4	93.4	32.0
2	72	110.3	56.3	95.5	39.2
3	78	85.8	83.1	90.3	7.2
Total	**193**	**269.3**	**66.2**	**93.3**	**27.1**

performance for the offline model in the Netherlands is over 90% for all log files. Log 3, which attained the worst performance for the Netherlands is illustrated in Figure 4.3. The manual annotations on the bottom graph have some discontinuities due to invalid data at some time steps. These time steps are not used when calculating the TP_D. The offline model makes some miss-predictions between minute 20 to 30 which is explained by a combination of incorrect readings from the camera and the Here map not being updated for variable speed limits on highways, as discussed in Section 3.3.4. Between minute 42-45, the camera is correct while the two map data sources are incorrect leading to an incorrect prediction by the offline model. This is explained by a temporary speed change that is not accounted for by the map sources.

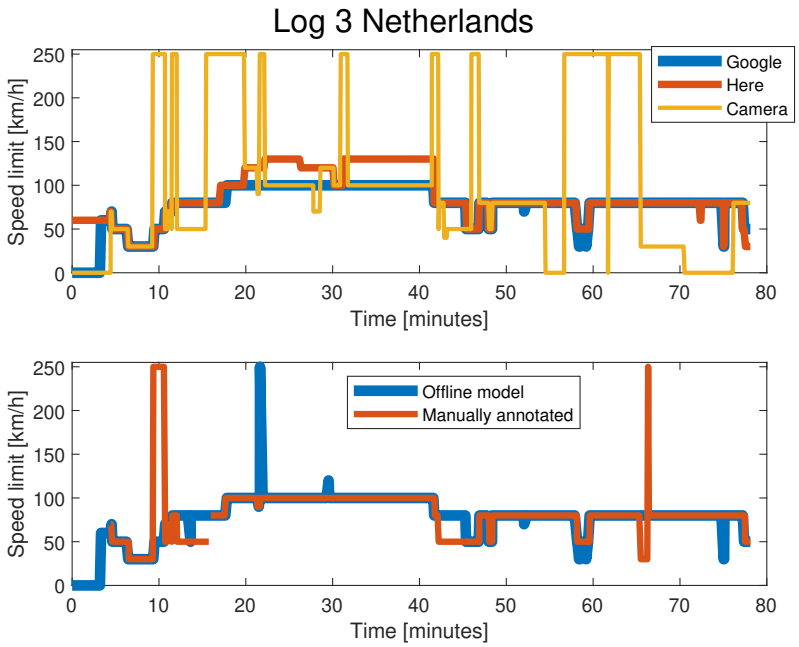


Figure 4.3: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth. Data collected in the Netherlands.

4.3 Luxembourg

Table 4.4 shows the specifications of the tuning log for Luxembourg, which constitutes 32.5% of the total driving distance for the country. Note that only a few log files, containing short driving distances, are available. This makes it difficult to draw general conclusions based on the results.

Table 4.5: Log files from Luxembourg for parameter tuning.

Log	Time [min]	Distance [km]
1	34 min	42.3

The tuning parameters for Luxembourg are presented in (4.4).

$$\alpha_{cam} = 0.15, \qquad \alpha_{Google} = 0.49, \qquad \alpha_{Here} = 0.52, \qquad \alpha_{off-ramp} = 0.54 \tag{4.4}$$

The two map data sources are the most reliable data sources for Luxembourg. The Here map gets a slightly higher tuning parameter value than the Google map. The true positive distance correctness for Luxembourg is shown in Table 4.6. The offline model performs better for all log files with a total performance of 91.9% compared to 62.1% for the RSI function. The AHR used by the RSI had similar software issues as for the Netherlands which is an explanation for the poor performance of the RSI function.

Table 4.6: Performance evaluation for log files from Luxembourg.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	71	56.3	76.9	92.1	15.2
2	19	16.4	44.4	96.4	52.0
3	23	15.2	26.3	86.3	60.0
Total	**113**	**87.9**	**62.1**	**91.1**	**29.0**

The offline model's output for log 1 is illustrated in Figure 4.4. The offline model has good accuracy for most time steps but appears to make incorrect predictions between minute 23 to 32, as shown in Figure 4.4. During this period, the predicted speed limits from all data sources disagree with the manual annotation. By inspecting the front camera video for this log file, it is possible to conclude that the annotation for this time period is incorrect. The video from the forward-looking cameras showed driving in a small village with a true speed limit of 50 km/h, not 120 km/h as indicated by the manual annotations. Errors like these are not corrected when calculating the performance, since it is too time consuming to analyze the videos from all log files.

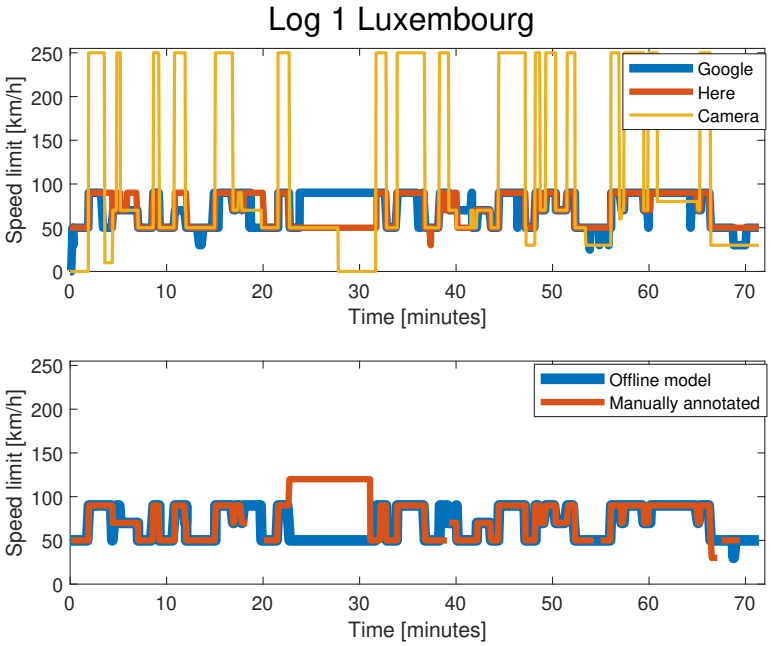


Figure 4.4: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth.
The discontinuities of the manual annotations are explained by invalid data.
Data collected in Luxembourg.

4.4 France

In France, one log containing 11.2% of the total driving distance is used for tuning the parameters. The specification of this log is shown in Table 4.7. The test data for France is limited and divided into only three log files, where only a small amount of data is used for tuning purposes. More data could be used for tuning, but this would mean that less data is available for evaluation.

Table 4.7: Log files from France for parameter tuning.

Log	Time [min]	Distance [km]	Country
1	60	42.0	France

The tuning parameters for France are presented in (4.5)

$$\alpha_{cam} = 0.76, \qquad \alpha_{Google} = 0.80, \qquad \alpha_{Here} = 0.82, \qquad \alpha_{off-ramp} = 0.08 \tag{4.5}$$

It can be concluded that all three data sources have similar reliability. The off-ramp tuning parameter is however very low ($\alpha_{off-ramp} = 0.08$) which means that the offline model relies only on map data when an off-ramp is detected. It is also observable that the Here map works slightly better than the Google map in France, since the reliability tuning parameter of the former is higher.

Two log files are used for evaluation. The total performance of the offline model is 88.9% which is higher than the RSI performance of 80.1%.

Table 4.8: Performance evaluation for log files from France.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	24	18.3	88.2	86.9	-1.3
2	194	316.2	79.6	89.1	9.5
Total	**218**	**334.5**	**80.1**	**88.9**	**8.9**

The offline model output for log 2, which consist of most evaluation data for France, is illustrated in Figure 4.5. The offline model shows a good accuracy for most time steps with some discrepancies around time 5-12 min. When the front camera video is inspected, it is possible to verify that the true speed limit over this road segment is 130 km/h and not 90 km/h which was indicated by the manual annotations. The offline model is however making the correct speed limit prediction.

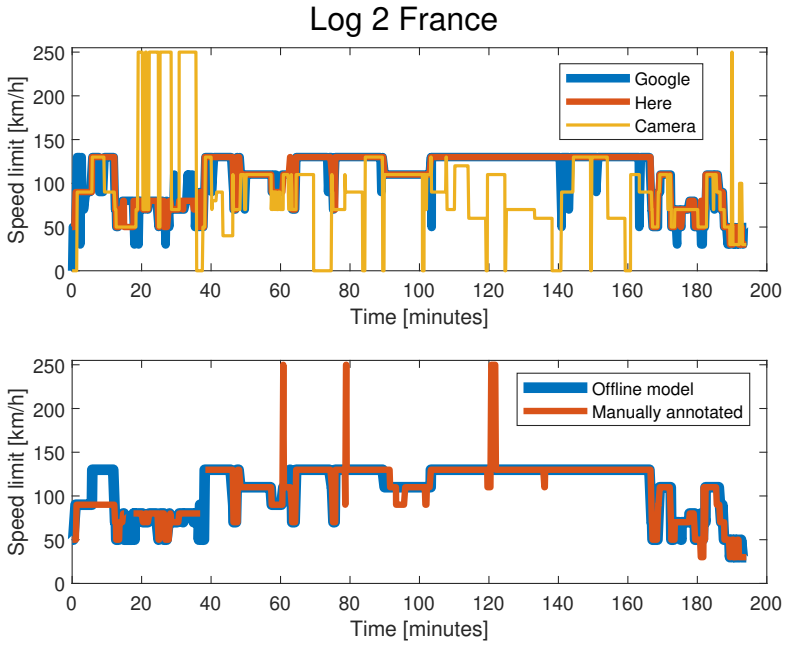


Figure 4.5: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth.
The discontinuities of the manual annotations are explained by invalid data.
Data collected in France.

4.5 Germany

For Germany, two log files containing 21.6% of the total driving distance are used for tuning. The specifications of the tuning log files are illustrated in Table 4.9.

Table 4.9: Log files from Germany for parameter tuning.

Log	Time [min]	Distance [km]
1	98	141.5
2	62	120.5

According to the expedition report of the log collection in Germany, a lot of road work was encountered which explains the high camera reliability seen in (4.6).

$$\alpha_{cam} = 0.90, \qquad \alpha_{Google} = 0.39, \qquad \alpha_{Here} = 0.29, \qquad \alpha_{off-ramp} = 0.89 \tag{4.6}$$

The tuning parameter for the Google map has a slightly higher value compared to the Here map. This indicates that the Google map is more reliable than the Here map in Germany.

Table 4.10 shows the result for the evaluation-logs in Germany. The offline model has a total accuracy of 72.1% in Germany compared to 45.7% of the RSI function. Both the RSI and the offline model are affected by the high number of road works

since it limits the performance of the map data sources. For some of the log files, the same software issues as for the Netherlands and Luxembourg were encountered, causing the RSI to obtain an incorrect reconstruction of the Google map. The offline model uses the offline reconstructor which bypasses this problem.

Table 4.10: Performance evaluation for log files from Germany.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	72	103.6	49.2	67.6	18.4
2	151	199.4	41.0	73.4	32.4
3	165	192.8	47.2	53.7	6.5
4	68	104.1	28.5	69.9	41.4
5	155	266.0	43.3	84.3	41.0
6	73	85.9	76.7	81.2	4.5
Total	**684**	**951.8**	**45.7**	**72.1**	**26.4**

Figure 4.6 illustrates the offline output for log 5 which is the longest log file for Germany. It can be seen that the two map data sources agree in most time steps while the vision deviates. This is explained by the number of road works encountered when collecting the log files. The tuning parameters are tuned such that the vision is trusted in a large extent. This is notable at time 72-102 min where the two map data sources are aligned but the offline model still bases its speed limit prediction on the vision output.

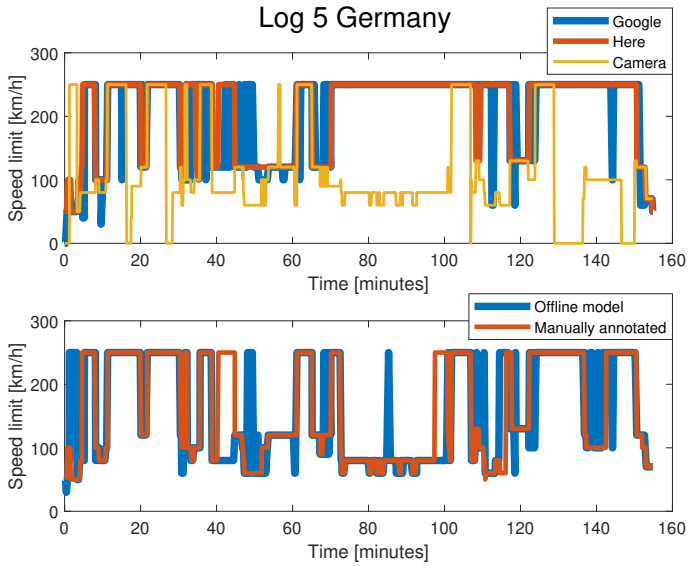


Figure 4.6: Top: speed limit data from the different data sources.
Bottom: predicted speed limit by the offline model compared with the ground truth. The discontinuities of the manual annotations are explained by invalid data.
Data collected in Germany.

4.6 Belgium

In Belgium only four log files are available. One of these was allocated for parameter tuning, which can be seen in Table 4.11, and the other three for testing.

Table 4.11: Log files from Belgium for parameter tuning.

Log	Time [min]	Distance [km]
1	34.1	42.3

The tuning parameters are shown in (4.7).

$$\alpha_{cam} = 0.48, \qquad \alpha_{Google} = 0.35, \qquad \alpha_{Here} = 0.61, \qquad \alpha_{off-ramp} = 0.26 \tag{4.7}$$

it can be seen that the Here map appears to be a reliable data source in Belgium, suggested by the high reliability tuning parameter value in (4.7). Some interesting results are obtained for the third testing log file, where the offline model performs significantly worse than the RSI function. In Figure 4.7, the result of plotting the speed limit readings from the different data sources is shown, along with the output of the offline model. Here, the three data sources at times completely disagree with one another, causing the offline model to struggle. The reason for this could be the fact that the different regions in Belgium apply different general speed limits [65]. An example of this could be that a highway sign implies different speed limits in different regions. This might explain the difference between speed limit values from the Google Maps and the Here map, illustrated in Figure 4.7.

Table 4.12: Performance evaluation for log files from Belgium.

Log	Time [min]	Distance [km]	RSI TP_D [%]	Offline model TP_D [%]	Δ TP_D [%]
1	18.6	17.8	90.8	98.6	7.8
2	22.1	18.3	79.2	94.2	15
3	143.6	163.8	86.0	66.2	-19.8
Total	**184.3**	**199.9**	**85.8**	**71.6**	**-14.2**

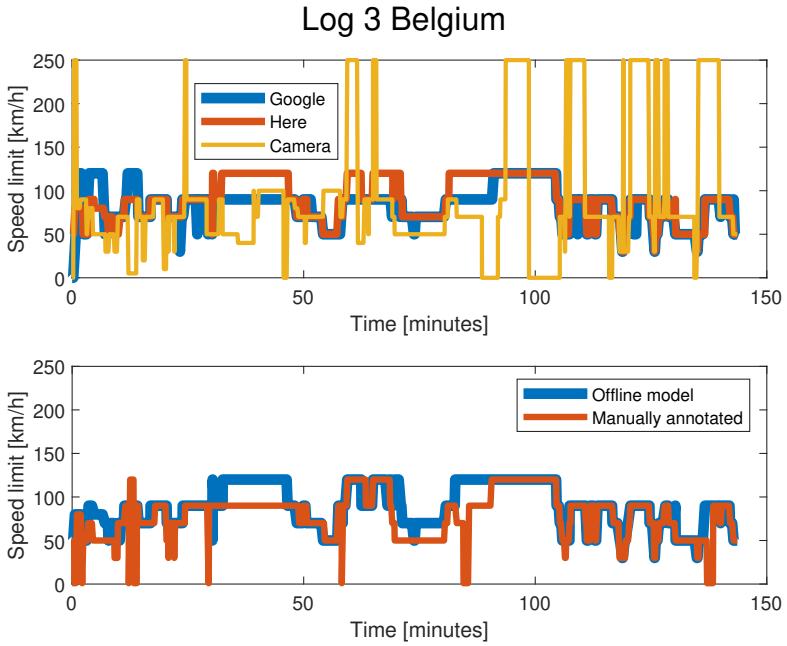


Figure 4.7: Top: speed limit data from the different data sources. A speed limit value of 250 km/h indicates unlimited speed limit.
Bottom: predicted speed limit by the offline model compared with the ground truth. Data collected in Belgium.

5
Discussion

In the following chapter the offline model's performance and the model architecture are discussed. Secondly, the parameter tuning method is analyzed in terms of how well the model adapts to different countries with their respective data source qualities. Thereafter the off-ramp detection system is discussed in terms of its effect on the model. Finally, suggested future work is presented, where suggestions on how to improve the model is given.

5.1 Model performance

In total, the model was evaluated on 25 log files. On 21 of those, the offline model's true positive distance was higher than the RSI function. The reason why the relative difference to the performance of the RSI function is interesting is because the quality of the data sources varies between countries, as well as the test drive conditions. This explains why both the offline model and the RSI function obtains higher correctness in some countries than others. The results for Sweden, Germany and Belgium are discussed further. The reason for this is because Sweden and Germany have the highest amount of log data available, and because Belgium is the only country where the RSI function performs better than the offline model.

The high true positive distance values for both the offline model and the RSI function in Sweden can be explained by high quality digital map data, a high density of speed signs, as well as the fact that all speed limit changes are marked by speed signs. The RSI function obtains a true positive distance corresponding to a correctness of 96.1%, whereas the offline model obtains a 97.8% correctness. The high values suggests that explicitly marked speed limits are favorable for both the RSI function and the offline model to work well.

In Germany, the results are rather different. The RSI function only achieves a correctness of 45.7%, suggesting that the data sources are incorrect, and possible non-ideal driving conditions. This could be explained by road work in progress, meaning temporarily decreased speed limits. These changes are generally not captured by the digital maps, which would result in incorrect speed limit data. The offline model handles this by setting the tuning parameter of the camera to a high value in relation to the two map reliability tuning parameters. Since it is common practice to put out speed signs during road work, high reliance on the camera is reasonable. This makes the offline model obtain a correctness of 72.1%, a significant improvement over the RSI function's performance.

In Belgium, the RSI function obtains a correctness of 85.8%, and the offline model a correctness of 71.6% when the results are computed for the total distance. In particular, log file 3 decreases the average performance of the offline model. It could be the case that the test drive conditions are different for that log file, meaning that the tuning parameters selected are not suitable. This log file also showed a large discrepancy between the two map data sources, illustrated in Figure 4.7, indicating that one of them is incorrect.

One limitation of the evaluation of the offline model's performance is the correctness of the manual annotations. Errors in the manual annotations were detected in several cases when observing the front camera videos. This was for example encountered for log file 1 in Luxembourg, discussed in section 4.3 and for log file 2 in France, discussed in section 4.4. Since manual annotations are used for tuning, incorrect ones can definitively impair the performance of the offline model. Besides, incorrect annotations deteriorate the results when the model is evaluated. On the other hand, scenarios like the one shown in Figure 4.4 shows a major advantage of the offline model. Since no data source agree with the manual annotation, this is a strong indication that the annotation might be incorrect.

5.2 Model architecture

The number of data sources used is a key element which affects the performance of the offline model to a large extent. A high number of data sources increases the probability that the correct speed limit is provided by at least one of the data sources. More data sources also increase the probability that the data sources agree. If the data sources agree they provide the same, often correct, speed limit.

The RSI function in the car uses two data sources, a road sign detection system and speed limit data from Google Maps. Data from these two sources were used as the foundation of the offline model developed in this work. To improve the performance beyond what would be possible by just fusing the onboard data sources, an additional map data source was incorporated in the offline model. This was found to be an appropriate way to improve the speed limit annotation accuracy.

5.3 Parameter tuning method

Another factor which greatly impacts the performance of the offline model is the way in which it combines the data sources. Since data sources varies in quality between different countries, tuning parameters in the data fusion algorithm are used. These tuning parameters act as a bias toward the data sources that prove to be the most reliable and are found by allocating log files solely for the purpose of parameter tuning. The parameter tuning was done separately for each country because of the country-wise variation of data source quality. For example, if one of the map suppliers proves to be very reliable in a country, it is reasonable to trust this data source to a large extent. From the result shown in Chapter 4, the

difference in reliability tuning parameter values between countries is significant. As a consequence, the model should adapt well to the data source quality in different countries.

A weakness of the parameter tuning method is that it requires manually annotated data which is representative for the test log files. If the data source reliability changes in one country, for example if the digital maps are updated, the model might require re-tuning.

5.4 Off-ramp detection system

The off-ramp detection system does not likely affect the overall performance of the model to the same extent that the model architecture and the reliability tuning parameters do. The reason for this is that in the context of error sources in the offline model, such as choosing the wrong data source, speed signs belonging to adjacent roads are likely a minor issue. The benefit of incorporating the off-ramp detection system into the offline model is therefore considered to be limited. However, on its own, the system works well in terms of detecting if the car is close to an off-ramp.

5.5 Industrial implementation

The purpose of this project was to develop an offline annotation model, whose output is as similar to the manual annotations collected during test drives as possible. However, by inspecting the videos from the front camera, it could be concluded that the onboard manual annotations are not always correct and can therefore not be used as ground truth when validating the RSI. To obtain ground truth speed limit annotations it would therefore be required to refine the annotations, for example by using videos from the forward-looking camera. Offline refinement using the videos from the forward-looking camera gives the opportunity to rewind, pause and slow down the video to get more correct annotations. The total process of gathering ground truth data would therefore consist of two steps, first by collecting manual annotations during test drives and secondly by manually refining the output using videos from the forward-looking camera.

The first step can be replaced with the offline model which automatically annotates speed limits. This model is however not perfect which means that the second step of refining the output is still needed. The advantage of using the offline model is that the process is automatic which decreases the manual labor and decreases unpredictable human errors. It is also easier to identify errors in the annotations, by observing the alignment of the data sources in the offline model.

5.6 Further development

To improve the offline model further, another additional map data source could be added. A map data source is preferred over a vision-based one because it is

simple to incorporate into the model. An offline vision-based data source would require processing of videos from the entire driving sequence, a computationally heavy process. In contrast, adding another map data source can be as simple as retrieving speed limit values at a set of coordinates which make up the path of the vehicle.

Another potential improvement is to make the offline model context aware to a higher extent, as suggested by Nienhüser *et al.* [14]. This means that environmental factors such as time of day, road work and weather can be used to temporarily reduce the mass values of the data sources. An example of this could be if a road work is detected. In this scenario temporary speed limits are commonly used which generally is not captured by the map data sources. This makes it reasonable to reduce the mass values of the map data sources temporarily. Another example could be to use unfavorable weather conditions such as rain as an indicator of when to reduce the mass value for the camera [14].

A third improvement is to use signal processing to refine the speed limit from the offline model. This could be done by filtering speed limit values which only apply for a few time steps. The reason is because the temporary detections are often incorrect. As can be seen in Figure 4.1, the camera system incorrectly gives the speed limit value 50 km/h after around 19 minutes. It would be desirable to filter out detections like this.

6
Conclusion

By incorporating an additional data source, the Here map, the offline model was able to outperform the RSI function on 21 out of the 25 log files it was evaluated on. On average for all log files, the offline model's true positive distance is 84.2%, compared to 69.8% for the RSI function. This illustrates that combining the data sources in the test car with an offline map data source, a promising model for automatic offline annotation of speed limits can be developed. The tuning parameters used in the fusion algorithm provide a useful method to tune the model depending on the reliability of the different data sources.

The offline model gives an automatic proposal of the correct speed limit, replacing manually annotated data collected during test drives. The advantage of using the offline model is that the process is automatic, which decreases the need of manual labor. This also makes it easier to identify errors by inspecting the alignment of the data sources, and decreases the risk of unpredictable human errors. However, the speed limit annotations proposed by the offline model are not without inaccuracies, and therefore manual refinement is needed to obtain perfect annotations.

Bibliography

[1] T. Vaa, T. Assum, and R. Elvik, "Estimating road safety effects at varying levels of implementation," in *Driver support systems*, 10 2014.

[2] WHO. (2021) Road traffic injuries. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

[3] M. Peden, *World Report on Road Traffic Injury Prevention.* WHO and UNICEF, 01 2004.

[4] E. Adell, "Driver experience and acceptance of driver support systems - a case of speed adaption," Ph.D. dissertation, Lund University, 12 2009.

[5] Council of European Union, "Commission delegated regulation (EU) no 2021/1958," 2021. [Online]. Available: https://eur-lex.europa.eu/eli/reg_del/2021/1958/oj

[6] J. Wang, S. Schroedl, K. Mezger, R. Ortloff, A. Joos, and T. Passegger, "Lane keeping based on location technology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 351–356, 2005.

[7] A.-S. Puthon, F. Nashashibi, and B. Bradai, "A complete system to determine the speed limit by fusing a GIS and a camera," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 1686–1691.

[8] M. Schreier and R. Grewe, "A high-level road model information fusion framework and its application to multi-lane speed limit inference," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1201–1208.

[9] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2110–2118.

[10] Zhu, Xi, Gu, Zhiqiang, and Wang, Zhen, "Ethical challenges and countermeasures of autonomous vehicles," *E3S Web Conf.*, vol. 233, p. 04016, 2021. [Online]. Available: https://doi.org/10.1051/e3sconf/202123304016

[11] Bundesministerium für Digitalisierung und Wirtschaftsstandort. (2017) Ethics commission's complete report on automated and connected driving. [Online]. Available: https://www.bmvi.de/SharedDocs/EN/publications/report-ethics-commission.html?nn=187598

[12] L. Zhu, J. Gonder, E. Bjärkvik, M. Pourabdollah, and B. Lindenberg, "An automated vehicle fuel economy benefits evaluation framework using real-world travel and traffic data," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 3, pp. 29–41, 2019.

[13] K. Faceli, A. de Carvalho, and S. Rezende, "Combining intelligent techniques for sensor fusion," in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02.*, vol. 4, 2002, pp. 1998–2002 vol.4.

[14] D. Nienhuser, T. Gumpp, and J. M. Zollner, "A situation context aware dempster-shafer fusion of digital maps and a road sign recognition system," in *2009 IEEE Intelligent Vehicles Symposium*, 2009, pp. 1401–1406.

[15] J. Daniel and J.-P. Lauffenburger, "Conflict management in multi-sensor Dempster-Shafer fusion for speed limit determination," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 987–992.

[16] G. Shafer, "A mathematical theory of evidence," in *A mathematical theory of evidence*. Princeton university press, 1976.

[17] M. Kłopotek, "Mathematical theory of evidence versus evidence," 2018. [Online]. Available: https://arxiv.org/abs/1811.04787

[18] J. Dezert, A. Tchamova, and F. Dambreville, "On the mathematical theory of evidence and Dempster's rule of combination," May 2011, 11 pages (2 columns). [Online]. Available: https://hal.archives-ouvertes.fr/hal-00591633

[19] M. Wahde, *Biologically Inspired Optimization Methods: An Introduction*. Southhampton: WIT Press, 2008.

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–511.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, nov 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[22] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*, C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 1. San Diego, United States: IEEE Computer Society, Jun. 2005, pp. 886–893. [Online]. Available: https://hal.inria.fr/inria-00548512

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[24] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[25] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. Lopez-Ferreras, "Road-sign detection and recognition based on support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 264–278, 2007.

[26] Ö. Kaplan and E. Saykol, "Comparison of support vector machines and deep learning for vehicle detection," in *3rd International Conference on Recent Trends and Applications in Computer Science and Information Technology (RTA-CSIT'18) at Tirana*, 11 2018.

[27] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks : the official journal of the International Neural Network Society*, vol. 32, pp. 323–32, 02 2012.

[28] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[29] I. Oztel, G. Yolcu, and C. Oz, "Performance comparison of transfer learning and training from scratch approaches for deep facial expression recognition," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 1–6.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[32] ——, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[33] D. Meek and D. Walton, "A note on finding clothoids," *Journal of Computational and Applied Mathematics*, vol. 170, no. 2, pp. 433–453, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042704000925

[34] J. Sánchez-Reyes and J. Chacón, "Polynomial approximation to clothoids via s-power series," *Computer-Aided Design*, vol. 35, no. 14, pp. 1305–1313, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448503000459

[35] Y.-B. Jia. (2020) Curvature (Computer Science 477/577 Notes). [Online]. Available: https://faculty.sites.iastate.edu/jia/files/inline-files/curvature.pdf

[36] F. Janda, S. Pangerl, and A. Schindler, "A road edge detection approach for marked and unmarked lanes based on video and radar," in *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 871–876.

[37] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich, "Viewpoint-aware object detection and pose estimation," *2011 International Conference on Computer Vision*, pp. 1275–1282, 2011.

[38] C. J. Hegarty, "GNSS signals — an overview," in *2012 IEEE International Frequency Control Symposium Proceedings*, 2012, pp. 1–7.

[39] A. E. Süzer and H. Oktal, "PRN code correlation in GPS receiver," *2017 8th International Conference on Recent Advances in Space Technologies (RAST)*, pp. 189–193, 2017.

[40] J. S. Subirana, J. J. Zornoza, and M. Hernández-Pajares. (2011) GNSS basic observables. [Online]. Available: https://gssc.esa.int/navipedia/index.php/GNSS_Basic_Observables

[41] Z. Xiuqiang, Z. Xiumei, and C. Yan, "Implementation of carrier phase measurements in GPS software receivers," in *2013 International Conference on Computational Problem-Solving (ICCP)*, 2013, pp. 338–341.

[42] K.-M. Cheung and C. Lee, "A trilateration scheme for relative positioning," in *2017 IEEE Aerospace Conference*, 2017, pp. 1–10.

[43] R. B. Langley, "Time, clocks, and GPS," *GPS World*, vol. 2, no. 10, pp. 38–42, 1991.

[44] A. El-Rabbany, *Introduction to GPS: The Global Positioning System*. Artech, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/9106103

[45] A. Kleusberg and R. B. Langley, "The Limitations of GPS," *GPS World*, vol. 1, no. 2, pp. 50–52, 1990.

[46] D. Wells, N. Beck, D. Delikaraoglou, A. Kleusberg, E. Krakiwsky, G. Lachapelle, R. Langley, M. Nakiboglu, K. Schwarz, J. Tranquilla, and P. Vanicek, *Guide to GPS Positioning*. Canadian GPS Associates, and University of New Brunswick, 1986.

[47] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*. New York: Springer-Verlag, 1994.

[48] C. Hay and J. Wong, "Enhancing GPS: Tropospheric delay prediction at the master control station," *GPS World*, vol. 11, no. 1, pp. 56–62, 2000.

[49] A. Leick, L. Rapoport, and D. Tatarnikov, *GPS satellite surveying*. John Wiley & Sons, Incorporated, 2015.

[50] V. Blervaque, K. Mezger, L. Beuk, and J. Loewenau, "ADAS horizon — how digital maps can contribute to road safety," in *Advanced Microsystems for Automotive Applications 2006*, J. Valldorf and W. Gessner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 427–436.

[51] C. Ress, A. Etemad, D. Kuck, and J. Requejo, "Electronic horizon-providing digital map data for ADAS applications," *Madeira*, vol. 3, pp. 40–49, 2008.

[52] A. Armand, J. Ibanez-Guzman, and C. Zinoune, *Digital Maps for Driving Assistance Systems and Autonomous Driving*. Cham: Springer International Publishing, 2017, pp. 201–244. [Online]. Available: https://doi.org/10.1007/978-3-319-31895-0_9

[53] J. Requejo, C. Ress, A. Etemad, and D. Kuck, "Using predictive digital map data to enhance vehicle safety and comfort," in *The Second International Workshop on Intelligent Vehicle Control Systems, Madeira, Portugal*, 2008.

[54] B. Kahl, "Autonomous driving data chain & interfaces," 2021. [Online]. Available: https://arxiv.org/abs/2104.01252

[55] R. Toledo-Moreo, D. Betaille, F. Peyret, and J. Laneurit, "Fusing GNSS, Dead-Reckoning, and Enhanced Maps for Road Vehicle Lane-Level Navigation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 5, pp. 798–809, 2009.

[56] C. S. Jensen and N. Tradišauskas, *Map Matching*. Boston, MA: Springer US, 2009, pp. 1692–1696. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_215

[57] Here Technologies. (2022) Here technologies. [Online]. Available: https://www.here.com/

[58] ——. (2022) Here route matching v8. [Online]. Available: https://developer.here.com/documentation/route-matching/dev_guide/index.html

[59] SWOV (Dutch Institute of Road Safety Research). (2022) What are the speed limits in the Netherlands? [Online]. Available: https://swov.nl/en/fact/speed-what-are-speed-limits-netherlands

[60] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[61] F. Larsson and M. Felsberg, "Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition," in *Image Analysis*, A. Heyden and F. Kahl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 238–249.

[62] F. Warburg, S. Hauberg, M. López-Antequera, P. Gargallo, Y. Kuang, and J. Civera, "Mapillary street-level sequences: A dataset for lifelong place recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2623–2632.

[63] Mapillary. (2022) Mapillary app. [Online]. Available: https://www.mapillary.com/app/

[64] Google. (2022) Google Maps satellite image showing the crossing Sörredsvägen-Björlandavägen, Gothenburg. [Online]. Available: https://www.google.com/maps/@57.7511001,11.8713612,150m/data=!3m1!1e3

[65] European Commission. (2021) Going abroad - Belgium. [Online]. Available: https://ec.europa.eu/transport/road_safety/going_abroad/belgium/speed_limits_en.htm

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY