ETH zürich

ETH Library

Situation Understanding at Stop Intersections

Master Thesis

Author(s):
Wyder, Thierry

Publication date:
2014

Permanent link:
https://doi.org/10.3929/ethz-a-010434060

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Berkeley
UNIVERSITY OF CALIFORNIA

Thierry Wyder

Situation Understanding at Stop Intersections

Master Thesis

Model Predictive Control Lab
University of California, Berkeley

Institute for Dynamic Systems and Control
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Prof. Dr. Francesco Borrelli
Prof. Dr. Lino Guzzella
Dr. Georg Schildbach
Dr. Stéphanie Lefèvre

October 2014

Preface

I would like to thank Professor Lino Guzzella for enabling me the incredible opportunity to compose my Master Thesis here in Berkeley, California.

I would also like to thank Professor Francesco Borrelli for welcoming me to his lab and providing me with an interesting and enjoyable working environment. It is my hope that my work will prove to be useful for the continuation of the autonomous car project.

Finally, I would like to thank Georg Schildbach, Stéphanie Lefèvre and the MPC Lab members for providing me with aid and support throughout my Master Thesis.

Contents

Abstract

All-way stop intersections are a widely used traffic management instrument. However, previous research to understand them in the context of autonomous vehicles is rare. The focus of this project is to comprehend other vehicles intentions at arbitrary all-way stop intersections, with limited sensor data available. This understanding is necessary for making a decision when to enter the intersection. The problem is split up into two parts. The first part deals with predicting the right of way. The second part with estimating the maneuvers of tracked vehicles in the intersection. The difficulties that arise when solving this problem originate from driver behavior and noisy sensor data. Real world data collected with a LIDAR system installed on a test car is used for cross-validation. The right of way prediction is solved with a kinematic feedback model to predict the arrival time at the intersection. It is successful in 90% of the validation datasets. However, it became clear that some scenarios need a more complex model. To tackle the maneuver estimation, a naïve Bayesian filter is implemented. It is designed to be applicable to arbitrary stop intersections of different layouts. This filter showed successful generalization to intersections of different sizes and geometries and it reduced the tracking distance necessary to recognize a maneuver by 30% compared to a deterministic reference model.

Nomenclature

Symbols

ϕ	Heading angle	$[deg]$
$\triangle\phi$	Relative angle to trajectory	$[deg]$
ω	Turning rate	$[^{\mathrm{deg}}/_{\mathrm{s}}]$
a	Acceleration	$[^{\mathrm{m}}/_{\mathrm{s}^2}]$
d	Distance to trajectory	$[m]$
i	Object ID	$[-]$
k	Discrete time index	$[-]$
l	Length	$[m]$
L	Length along a trajectory	$[m]$
m	Maneuver	$[-]$
s	Segment	$[-]$
t	Time	$[s]$
v	Velocity	$[^{\mathrm{m}}/_{\mathrm{s}}]$
v_x	Velocity in x direction	$[^{\mathrm{m}}/_{\mathrm{s}}]$
v_y	Velocity in y direction	$[^{\mathrm{m}}/_{\mathrm{s}}]$
w	Width	$[m]$
x	x Position	$[m]$
y	y Position	$[m]$
z	Vector of measurements	$[-]$

Indices

act	Actual
est	Estimated
seg	Segment
veh	Vehicle

Acronyms and Abbreviations

CAN	Controller Area Network
CC	Correct Classification
CRM	Constant Rate Model
CTRA	Constant Turn Rate and Acceleration

DBN	Dynamic Bayesian Network
DOF	Degree of Freedom
ETH	Eidgenössische Technische Hochschule
JOSM	Java Open Street Map
UCB	University of California at Berkeley
HMM	Hidden Markov Model
MLP	Multi Layer Perceptron
MPC	Model Predictive Control
NBF	Naïve Bayesian Filter
SVM	Support Vector Machine

Chapter 1

Introduction

This section first outlines the problem that was tackled within the scope of this thesis. Then, a literature review is presented showing what research has previously been done on this matter. Finally, the proposed solution is introduced.

1.1 Problem Formulation

The goal of this project is to find a procedure to understand the situations that arise at all-way stop intersections. Ultimately, this will allow autonomous vehicles to navigate them successfully. All-way stop intersections require vehicles to stop whether conflicting traffic is present or not. They are widely used in North America, especially in residential neighborhoods and rural areas and they provide efficient, economical and safe traffic management under certain traffic volumes and geometric constraints. However, even though they are considered safe, there are approximately 700,000 reported vehicle crashes at stop signs annually, according to Retting et al. [22]. A further understanding of the situations arising is therefore in the interest of safety as well. The California Vehicle Code [1] defines the right of way at stop intersections:

> (a) The driver of a vehicle approaching an intersection shall yield the right-of-way to any vehicle which has entered the intersection from a different highway.

Essentially, this means that one has to yield to vehicles that arrived earlier at the intersection. Because stop intersections are traffic zones of heavy interaction between the different traffic participants, even more than at traffic lights, it is of great interest to predict their actions and movements. As an example, Figure 1.1 and Figure 1.2 show two possible situations. In both, there are four cars present. Cars number 1 and 2 arrive first and second at the intersection, ahead of our ego car. The ego car is marked with an E and we arrive third at the stop line with the intention to go straight. Car number 4 is still in the process of approaching the intersection in both situations. We have to recognize that this car arrives after us and that we therefore have the right of way over it. In the first situation, car 1 is going straight and will cross neither our path nor the one of car 2. Car 2 realizes that and therefore enters the intersection as well. As soon as we see that car 2, which has the right of way because it arrived earlier, turns right, we can enter the intersection as well without causing a conflict. In the second situation however, car 1 turns left and car 2 wants to go straight. This forces car 2 to wait until its path is clear before it enters the intersection. In our ego car, we have to detect that on one hand, our own path is blocked by car 1 turning left and that on the other hand, car 2 has the right of way and can set off ahead of us. When car 2 sets off, we wait until we can detect that it is going straight. After it cleared our side of the intersection, both conflicts have been resolved and our ego car can enter the intersection as well.

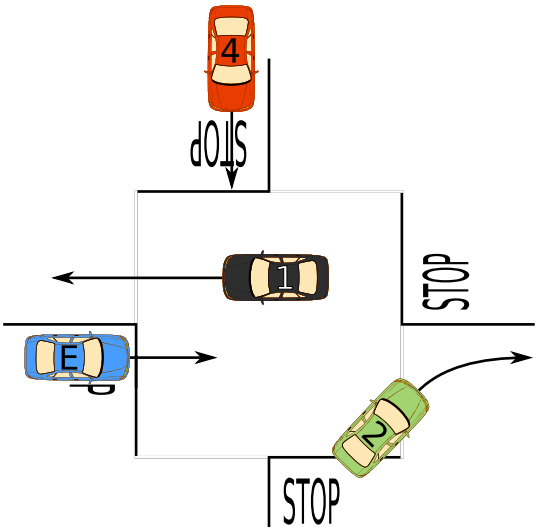

Figure 1.1: Complex Traffic Situation without Conflict.

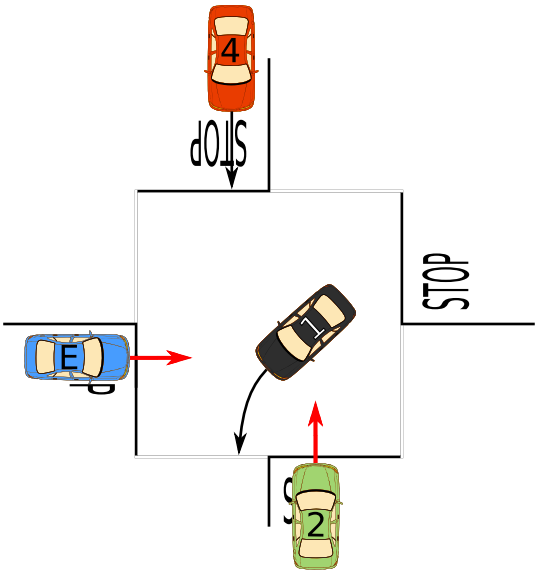

Figure 1.2: Complex Traffic Situation with Conflicts.

From these examples, we can see that the problem at hand can be split up into two parts. In the first one, the arrival time of vehicles at the intersection has to be estimated to create an order according to which the vehicles will have the right of way to enter the intersection. And in the second part, the maneuver performed by a vehicle entering the intersection has to be estimated in order to predict its future trajectory. Solving these two problems will allow an autonomous vehicle to answer the question: "When can I enter an all-way stop intersection?". The goal of this work was to develop a solution to both of these problems. While doing this, a special focus was set on developing an approach that is valid for arbitrary intersections of different geometries and sizes. The main difficulties that arise when solving these problems originate from noisy sensor data and driver behavior. Driver behavior sums up that different drivers take very different trajectories through intersections, their stopping behavior at the sign might be abrupt, they might stop early or not stop at all. Interestingly, significantly less research has previously been done on stop intersections compared to intersections controlled by traffic lights. This might be because traffic

light intersections are a more controlled environment with a reduced complexity.

The available information to tackle this problem is provided by a laser sensor system installed on a test car. Further on, a road map is used to provide the data needed to specify the geometry of intersections. The algorithms were implemented and tested offline in a simulation environment with real world data. It was assumed that all cars comply with the traffic rules. Therefore, no car would run the stop line or stop in the middle of the intersection. However, it was taken into account that most vehicles do not stop completely but perform a rolling stop. This fact could not be ignored because Woldeamanuel [26] showed that it was the case for 65% of all drivers.

1.2 Literature Review

An up to date overview on motion prediction in intelligent vehicles has been done by Lefèvre et al. [17]. The authors propose a classification of the different motion models and prediction approaches into three groups:

1. **Physics-based:** The movement of the tracked vehicle is only modeled based on the laws of physics.
2. **Maneuver-based:** These models contain the physics-based model but also take into account that the future motion of the vehicle depends on the maneuver that the driver is about to perform.
3. **Interaction-aware:** The interactions and reactions between multiple vehicles are also incorporated.

Purely physics-based trajectory predictions have been widely researched and applied. Common representations are dynamic models such as the bicycle model given by Rajamani [21] or models similar to the constant turn rate and velocity change rate model described by Batz et al. [5]. However, these models ignore traffic rules, road geometry and other external influences on the vehicle.

Maneuver-based models go a step further and try to identify the high-level behavior that the driver is performing or about to perform. For example, when considering a vehicle approaching a stop sign, one might want to know if the vehicle will be compliant with the traffic rules and stop at the sign or just run across it. In this case, a possible set of maneuvers might be $M \in \{\text{stop}, \text{rolling stop}, \text{no stop}\}$. These maneuvers are usually represented by unique prototype trajectories or by a stochastic representation as in Vasquez and Fraichard [25]. They can be learned in advance using training data or they can be constructed based on road geometry. The challenge for these models is to estimate in what maneuver the observed vehicle is currently in. Lefèvre et al. [17] mention several techniques that have previously been applied. There are context based or heuristic algorithms as found in the publication of Greene et al. [10]. For more complex problems, increasingly advanced techniques have been applied. For example Multi-Layer Perceptrons by Ortiz et al. [20], Support Vector Machines by Aoude et al. [2] or Hidden Markov Models by Aoude et al. [3].

The last group mentioned in the list above are the interaction-aware motion models. They overcome the limitations of the previous group which made the assumption that all vehicles move independently from each other. Vehicles on the road obviously influence each other and the maneuvers they perform can alter other vehicles maneuvers. These models therefore lead to a better understanding and a better reproduction of the real world behavior. However, only few models of this type can be found in literature. The main reason for this is the increased computational cost due to more complex models. Examples of implementations can be found in the works of Gindele et al. [9] and Lefèvre et al. [14], [15], [16].

As part of this literature review, a closer look at maneuver estimation was taken since this is one of the problems that was addressed in this project. One approach used in previous research are hidden Markov models (HMM). Meyer-Delius et al. [18] propose a model based on HMMs with underlying dynamic Bayesian networks (DBN) to classify a passing maneuver on the highway. They

introduce a three-layer model with a state-space model at the bottom, an abstract state description in the middle and the situational model at the top. The state-space model is based a DBN. It is a compact representation of a complex stochastic process which describes the system at each point in time t. An introduction to these dynamic probabilistic networks is given by Friedman et al. [8]. The states from this system are then transformed into so called abstract states. An example for this transformation is given in Table 1.1 for the distance between the ego vehicle driving on a highway and the vehicle in front of it.

Relation	Distances
equal(R,R')	$[0m, 1m)$
close(R,R')	$[1m, 5m)$
medium(R,R')	$[5m, 15m)$
far(R,R')	$[15m, \infty)$

Table 1.1: Conversion from Physical to Abstract States by Meyer-Delius et al. [18].

Those abstract states are then used in a Bayesian filter to estimate the maneuver in the HMM. The transition probability matrix defines the allowed transitions between the states in the HMM, and therefore between the maneuvers. This HMM is depicted in Figure 1.3.

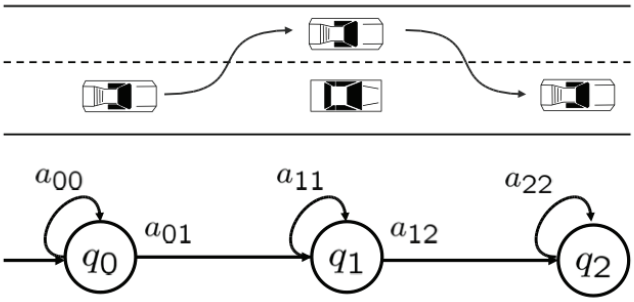

Figure 1.3: A passing maneuver, in which the reference vehicle (square car in the middle) is being passed on its left hand side by another vehicle. In this case, the maneuver was divided into three stages and, thus, the 3-state HMM shown at the bottom was used. Meyer-Delius et al. [18]

In a later publication, Meyer-Delius et al. [19] presented an improved solution where they removed the conversion from physical to abstract states. They replaced it with an observation model based on Gaussian mixtures.

Further on, Aoude et al. [3] propose two algorithms for behavior classification at intersections. The first one using HMMs and the second one support vector machines (SVM). However, the classification in this publication is limited to estimating if a vehicle is compliant with the traffic rules or if it is violating them. A different approach to the vehicle behavior classification problem was presented by Ortiz et al. [20]. The defined problem was to predict behavior primitives of a vehicle approaching a traffic light. The possible states were defined as $x \in \{\text{Constant Velocity}, \text{Braking}, \text{Stopped}\}$ and the inputs as $u \in \{\text{Distance to Intersection}, \text{Speed}, \text{State of Traffic Light}\}$. The prediction was made using a multi layer perceptron (MLP) with one hidden layer consisting of 100 nodes. Furthermore, Hulnhagen et al. [11] describe an approach to maneuver recognition using a fuzzy rule base to identify basic maneuver elements and probabilistic finite state machines to model the temporal sequence of a particular driving maneuver. This method was successful in distinguishing turn maneuvers from similar maneuvers using real world data. Gindele et al. [9] propose a filter that estimates the behaviors of traffic participants and predicts their future trajectories. The filter is based on a dynamic Bayesian network which consists of a finite set of behaviors. Lefèvre et al. [13] use a Bayesian network to predict vehicle maneuvers at road intersections. This method includes topographical information from a map.

The prediction was made based on information on what road and what lane the vehicle entered the intersection and on the velocity and orientation of the vehicle. Finally, Armand et al. [4] modeled stop intersection approaches with the use of Gaussian processes.

1.3 Proposed Solution

As explained in Section 1.1, there are two problems that were tackled in the scope of this project. The main problem is to estimate which exit road a vehicle will take, after it entered a stop intersection. The second problem is to predict the time, at which an approaching vehicle will arrive at the stop intersection.

For the first part, the different maneuver estimation algorithms found in the literature where evaluated. The first approach up for discussion was a HMM. These models are often used by decomposing maneuvers into steps that chronologically follow each other. This was the case for the model in Figure 1.3 where HMMs provide a good solution. The maneuvers of the problem at hand however can not be split up chronologically. A HMM is therefore not applicable. Further on the MLP approach was neglected due to limited knowledge and complex implementation. The fuzzy logic methodology mentioned in the literature review was investigated further at first but then dropped because there was no clear benefit from it in sight. As a result, a Bayesian filter would be used to solve the maneuver estimation problem.

For the second part, the arrival time estimation, a physics-based model was evaluated. The only other model for this task that was found in the literature was of a complexity that is beyond the scope of this project.

Chapter 2

Hardware Setup and Data Collection

The two algorithms introduced in this project were both trained and then validated with real world data. This chapter explains how and where this data was collected and preprocessed. The current vehicle platform used by the MPC lab is a Hyundai Azera 2013. It is equipped with the following sensor systems:

- An inertial navigation system with GPS. It provides six DOF information such as accelerations and velocities as well as the vehicles global position.
- A camera vision sensor from Mobileye that is able to detect lane information.
- A laser scanner system from IBEO Automotive Systems able to detect objects up to 200 meters away.
- The cars own proprioceptive sensors, providing wheel speeds, steering angles and so on.

The system used for this project was the IBEO LIDAR. All sensor measurements were taken with the ego vehicle stationary. The vehicles position was determined manually with the use of a map.

2.1 IBEO LIDAR System

2.1.1 LIDAR Technology

The term LIDAR is a blend of the words light and radar and stands for a remote sensing technology where a target is illuminated with lasers. The reflected light is analyzed in order to measure the distance to that target. The IBEO sensors installed on the test car use time of flight technology in order to get this distance and they also analyze the echo pulse width. Every sensor scans the surroundings with four layers of lasers set at different vertical angles as it can be seen in Figure 2.1. This setup allows to capture objects at different heights and distances. The sensors return a cloud of points from the scanned surfaces. The internal post processing unit from IBEO matches a box to that point cloud. An example of such a point cloud of an observed vehicle and the fitted box is shown in Figure 2.2.

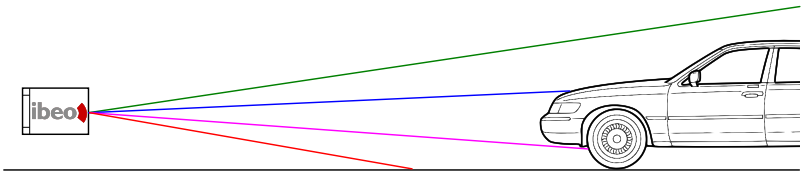

Figure 2.1: Vertical Laser Setup.

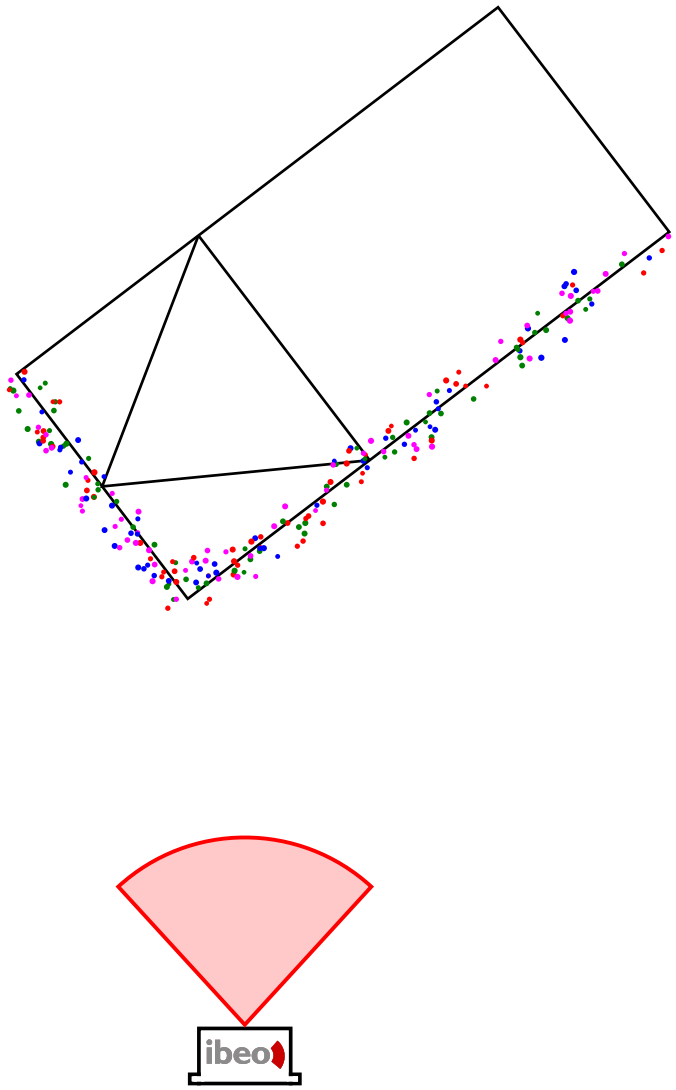

Figure 2.2: Object Box Generation.

2.1.2 Sensor Placement

There are six of these LIDAR sensors installed on the test car in the locations shown in Figure 2.3. The data received from them is passed to a fusion box. Each of the sensors has a horizontal field of view of 85 degrees. The fields of view of adjacent sensors overlap each other in order to allow better sensor fusion. This results in a 360 degree all around view as visualized in Figure 2.4.

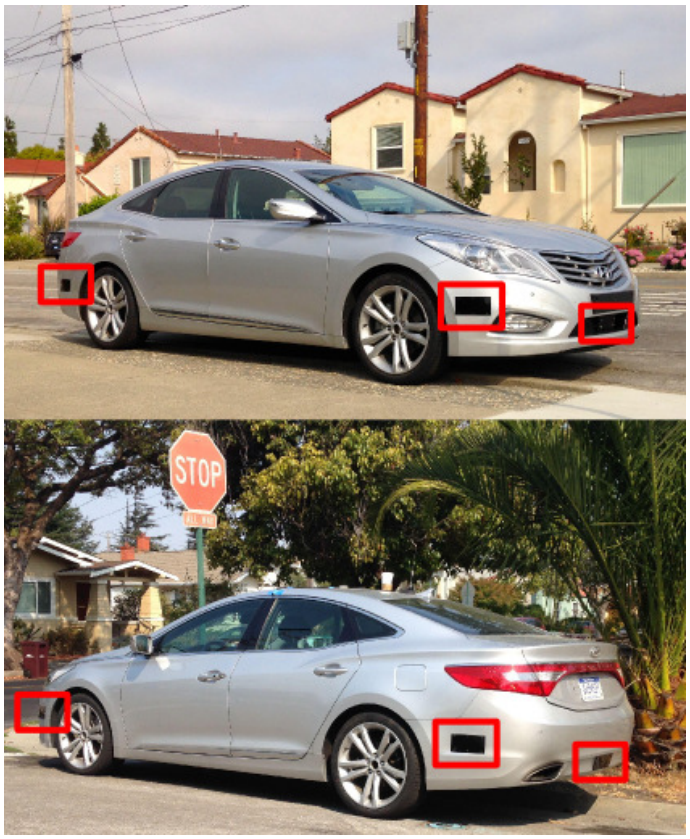

Figure 2.3: Sensor Locations on the Car.

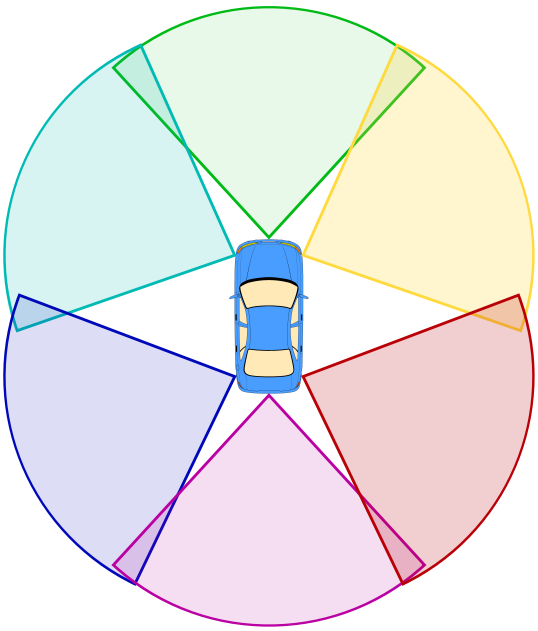

Figure 2.4: Field of View and Blind Spots.

2.1.3 Data Retrieval

The data is retrieved from the sensor fusion box over the vehicles CAN bus. The abbreviation CAN stands for Controller Area Network and it is a standardized vehicle bus. It is designed to allow the communication between micro controllers without a host computer. Our vehicles communication setup is given in Figure 2.5. The blue boxes on the left hand side represent the CAN bus system on the car. CAN bus two and three are primarily used to collect data from the respective sensors. The vehicles own CAN bus, number one, was made accessible in order to connect to the cruise control and active steering control systems as well as the vehicles proprioceptive sensors. A dSPACE box is used to run the MPC controller. The CANape data logging software from Vector Informatik collects and processes the available data. For this project, all data was processed offline. Table 2.1 shows the primary data variables that are provided by the IBEO fusion box for every tracked object.

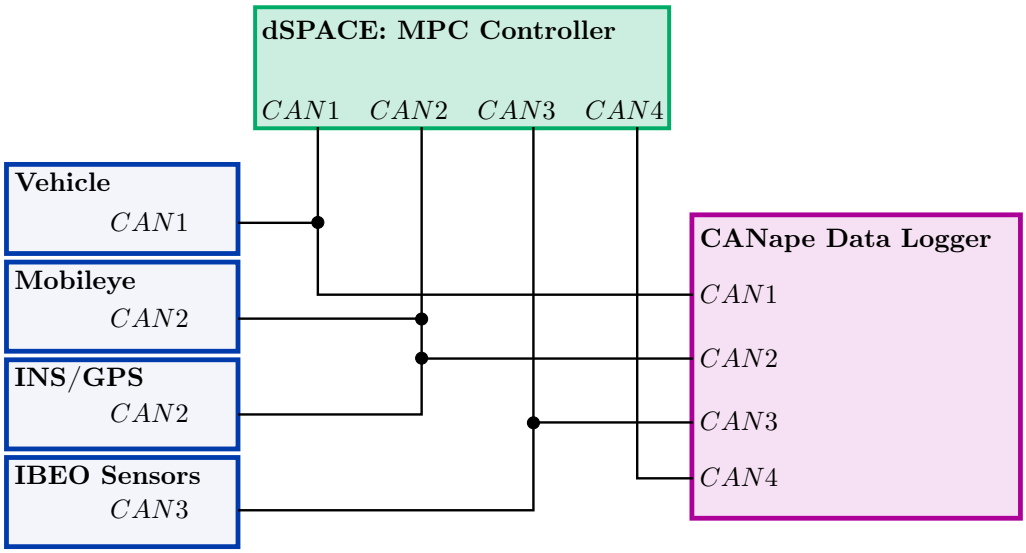

Figure 2.5: Communication Setup on the Test Car.

Content	Symbol	Description	Unit
Object ID	i	Unique ID for a tracked object, used to separate objects from each other.	[–]
Time	t	Timestamp at which the measurement was taken.	$[s]$
x Position	x_i	x Position of the object (center of box) in the reference coordinate system.	$[cm]$
y Position	y_i	y Position of the object (center of box) in the reference coordinate system.	$[cm]$
Velocity	v_i	Object velocity in the reference coordinate system.	0.1 [m/s]
Orientation	ϕ_i	Angle of the fitted box.	0.01 $[deg]$
Size	w_i, l_i	Width and Length of the fitted box.	$[cm]$
Classification		Object classification into different vehicle categories.	[–]

Table 2.1: Tracking Information from IBEO Sensors.

The sensor system returns the position $(x_i(t), y_i(t))$ of an object with its ID and the timestamp when it was scanned. It also returns the velocity and orientation of the object and the size of

the box fitted to it. The coordinate system in which those measurements are taken is defined in Figure 2.6. This coordinate system has been used throughout the project. The ego car taking the measurements is shown in blue while the tracked vehicle is colored green. The reference point of the coordinate system is in the center of the rear axle of the ego car. The box fitted to the tracked object is shown in the figure as a dashed line around it. The center of this box defines the location of the object.

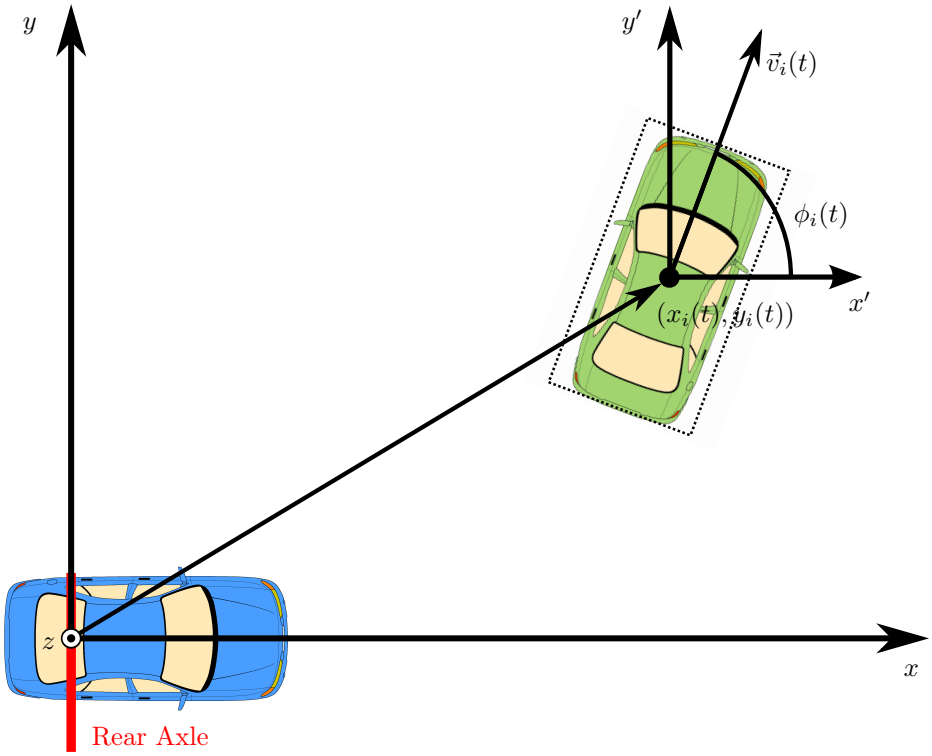

Figure 2.6: Sensor Coordinate System. (Ego Car in Blue, Tracked Vehicle in Green)

There is also high level information available from the sensors such as a variable specifying the classification of the object. Classification is meant in the sense that objects are categorized as either Pedestrians, Bikes, Cars, Trucks, Unknown Small, Unknown Big or Unclassified. All of this high level information seems very helpful at first. However, most of it is too noisy and too uncertain to be used in maneuver prediction. An example of this problem is shown in Figure 2.7. The two plots show orientation data for roughly two seconds long trajectories of a tracked car, collected with the IBEO sensor system. The blue lines show the box orientation directly taken from the IBEO post processing unit. The red lines were calculated as the heading angle of the object given by

$$\phi_i(k) = \frac{180}{\pi} \arctan\left(\frac{y_i(k+1) - y_i(k)}{x_i(k+1) - x_i(k)}\right) \tag{2.1}$$

based on the position data. One can clearly see that the box orientation from the sensor system has very large fluctuations due to flatlining at zero for long periods of the time. Also, there is even data missing at the start of the second plot. Obviously, given that the box angle represents the vehicle orientation, one would not necessarily expect it to align perfectly with the heading angle of the the vehicle. However, the variations of the box orientation are clearly misreadings from the sensors and sensor noise. As a consequence of this problem, only the position and velocity information along with their timestamp and object ID are used directly from the sensors for this project. All other variables such as the heading angle are calculated from them.

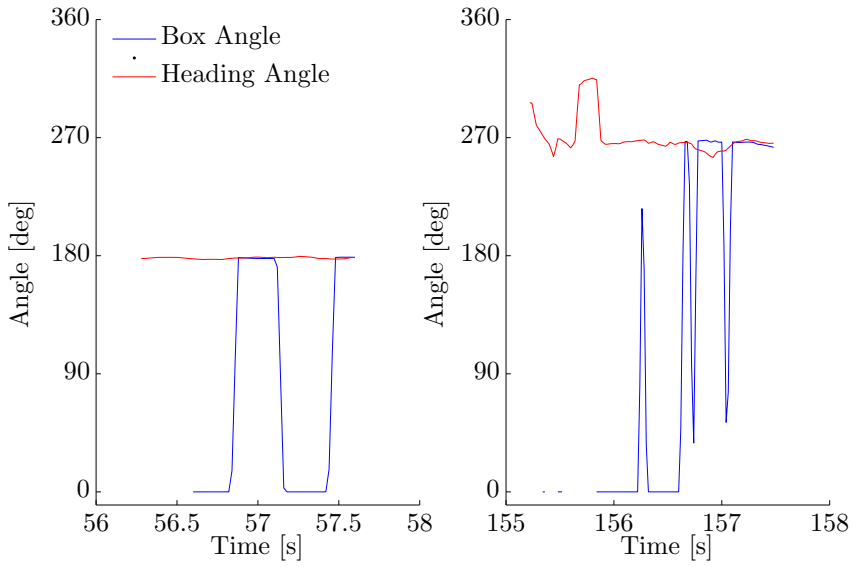

Figure 2.7: Comparison of the IBEO Box orientation versus the heading angle calculated from the position data.

2.2 Data Collection

The main task of the data collection was to track vehicles while they were approaching or navigating within an all-way stop intersection. The procedure for this was to keep our test car stationary next to an intersection. In order to get good sensor readings, the test car had to be as close as possible to the intersection with a good field of view into all roads that form the intersection. However, the test vehicle was not allowed to be in the way of, or otherwise distract, the other road users in order to avoid falsified sensor readings. Further on, the test car had to be positioned as horizontal as possible to capture all objects on the road. This was an issue with roads dropping off heavily towards the sidewalk. It would lead to sensors pointing into the ground and not capturing any objects that a are driving by.

In order to get the first data set, used for the training and the verification of the algorithm, a standard intersection was selected. The choice fell on the crossing of Santa Fe Avenue and Portland Avenue in Albany, CA. This is the single lane, symmetric, 4-way Stop intersection shown in Figure 2.8 and in the schematic on the left hand side of Figure 2.10. It is located in a residential neighborhood where these type of intersections are very common. This crossing sees a comparatively large amount of traffic which was beneficial as it allowed the recording of more vehicle trajectories in a shorter amount of time. Figure 2.9 shows two different locations where the test car was parked at this intersection while recording passing vehicles.

Figure 2.8: Intersection 1: Santa Fe Avenue and Portland Avenue. Google Maps™.

Figure 2.9: Two different test car locations for data collection at the intersection of Santa Fe Avenue and Portland Avenue.

At this intersection, over 200 passing vehicles were recorded during a 30 minute time window of data collection. Half of these trajectories were used for training the algorithm, and the other half for verifying it. Later on, the verification data set was extended by collecting data at different intersections. This data was used to show that the maneuver estimation algorithm would work on intersections with arbitrary geometries as well. The following intersections were used for this:

Washington Ave & Key Rte Blvd: A large 4-way intersection with a vegetated lane separator on the main street. It is shown in top middle schematic of Figure 2.10 and in Figure 2.11a.

Portland Ave & Pomona Ave: This is a small 3-way intersection, shown at the top right of Figure 2.10 and in Figure 2.11b.

Washington Ave & Neilson St: A 4-way intersection where the cross street running from north to south if significantly less wide than the other street. Also, one of the entrances joins the intersection at an angle smaller than 90 degrees. This makes the intersection asymmetric. It is shown at the bottom middle of Figure 2.10 and in Figure 2.11c.

Washington Ave & Pomona Ave: The last intersection is characterized by a shift of the two entrances of the street running from east to west. Even though this shift is larger than the street is wide, is is still considered as one intersection and also driven as such by all motorists. It is shown at the bottom right of Figure 2.10 and in Figure 2.11d.

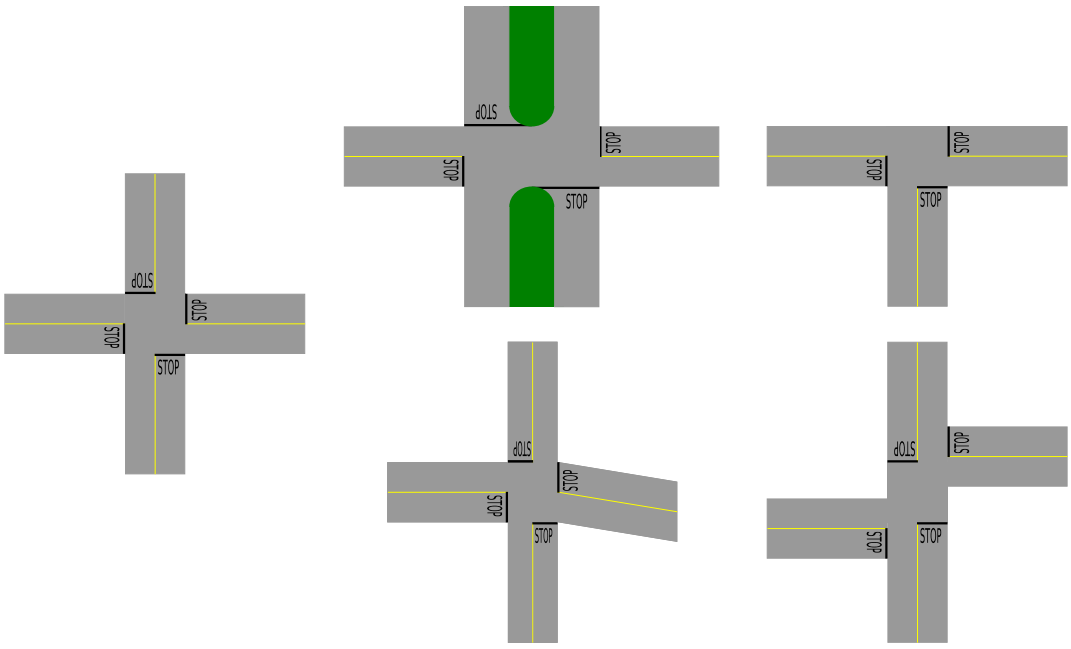

Figure 2.10: Schematics of all intersections where data was collected.

(a) Intersection 2: Washington Ave & Key Rte Blvd

(b) Intersection 3: Portland Ave & Pomona Ave

(c) Intersection 4: Washington Ave & Neilson St

(d) Intersection 5: Washington Ave & Pomona Ave

Figure 2.11: Futher intersections used for data collection as seen on Google Maps™.

2.3 Generation of Reference Paths

In order to implement the naïve Bayesian filter, a fast and realistic method to generate reference paths was necessary. The inputs available for it are the start and end point of a path as well as the angle at those points. According to van der Molen [24] and Labakhua et al. [12], sets of successive clothoids following each other can be used for trajectory planning tasks for passenger vehicles. With clothoid curves, it is possible to produce smooth paths with smooth changes in curvature. Rajamani [21] defines a clothoid to be a spiral whose curvature is a linear function of its arc length. It can be written in parametric form as

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = b \begin{pmatrix} C(t) \\ S(t) \end{pmatrix} \tag{2.2}$$

given the scaling factor b and the time parameter t. The two terms $C(t)$ and $S(t)$ are the Fresnel integrals given by:

$$C(t) = \int_0^t \cos\left(\frac{\pi\tau^2}{2}\right) \mathrm{d}\tau \tag{2.3a}$$

$$S(t) = \int_0^t \sin\left(\frac{\pi\tau^2}{2}\right) \mathrm{d}\tau. \tag{2.3b}$$

A fast and accurate clothoid fitting algorithm was implemented and published by Bertolazzi and Frego [6]. An example trajectory produced by it can be seen in Figure 2.12. The task was to connect the first point at $(0, 0)$ with the second one at $(0, 3)$. However, the orientation of the trajectory at those points was set to be the opposition of each other. As one can see, the blue path solves this problem.

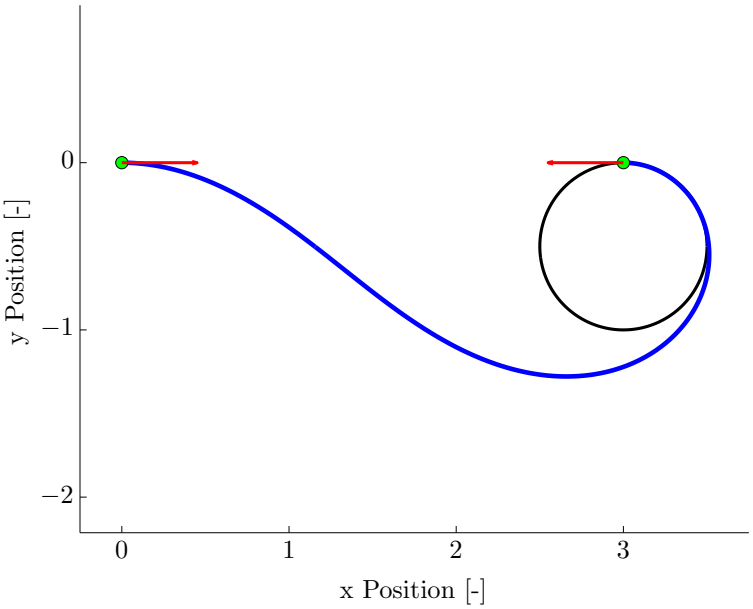

Figure 2.12: Clothoid path connecting two points with opposed directions.

2.4 Intersection Geometry Specification

In order to apply the model trajectory generation procedure described in Section 2.3, a standardized description of the intersection geometry had to be defined. The following information is needed:

1. Center of the entry lanes to the intersection.
2. Center of the exit lanes from the intersection.
3. Center of the roads at the intersection.
4. Center of the roads further away from the intersection.
5. Ego car position.

All of these points except the ego car position have to be read out of a map. In a real world application, the ego car position would be given by the localization system of the car. For the project at hand, the position was simply determined at the test site. The first two points from above are used to determine where the vehicles would enter or exit the intersection. The third and fourth category of points in the list are used to determine the angle for the roads entering the intersection. Figure 2.13 shows these points on the model intersection. All of them were found with JOSM, the Java OpenStreetMap Editor, in longitudes and latitudes and with a resolution of 90 millimeters. They were then converted into the ego car coordinate system given by Figure 2.6.

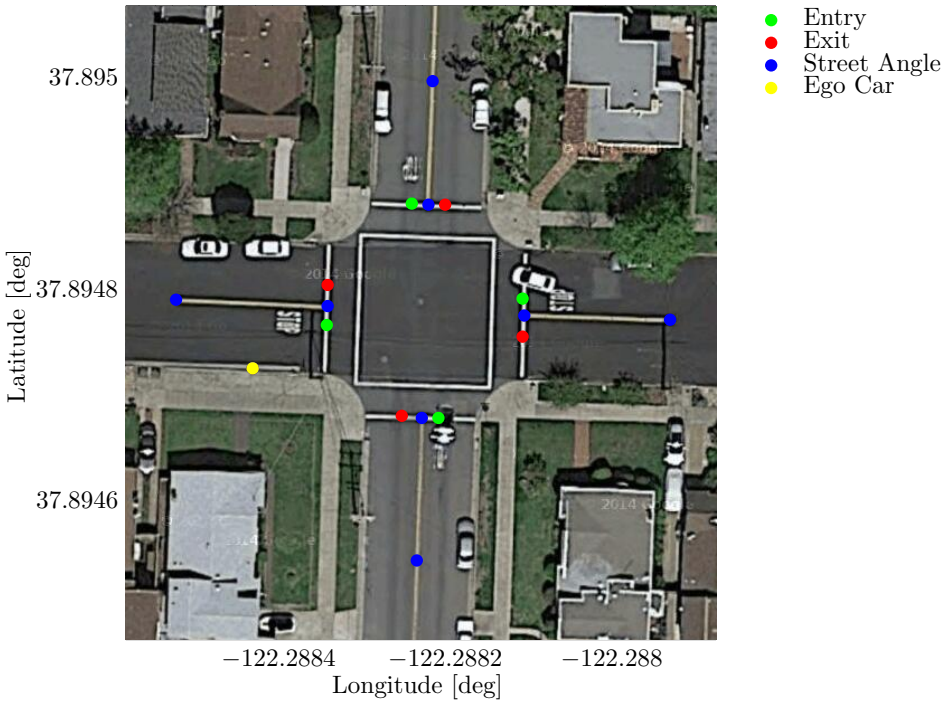

Figure 2.13: Geometric Points for Reference Path Generation.

Further on, the problem that most drivers cut the corners had to be assessed. It means that they start tuning in before actually crossing the stop line and that they have completely finished the turn after exiting the intersection. This driving behavior can be reproduced by moving the start and the end point of the path further away from the intersection, back into the road. It is done by shifting the respective start and end points along a straight line at the angle of the road. An empirical study of the data at hand showed that a shift of -4 meters for the entry points and a shift of -6 meters for the exit points represents the actual driving behavior well. Given those start and end points and the road angles, the reference paths can be generated using the clothoid algorithm described in Section 2.3. The result is shown in Figure 2.14 and Figure 2.15 for intersection number one and two. In these Figures, the inner black square represents the intersection area drivable for vehicles. The outer square is constructed by extending the four stop lines until they join each other. Those squares are also painted on the road as seen in Figure 2.13 and Figure 2.11a.

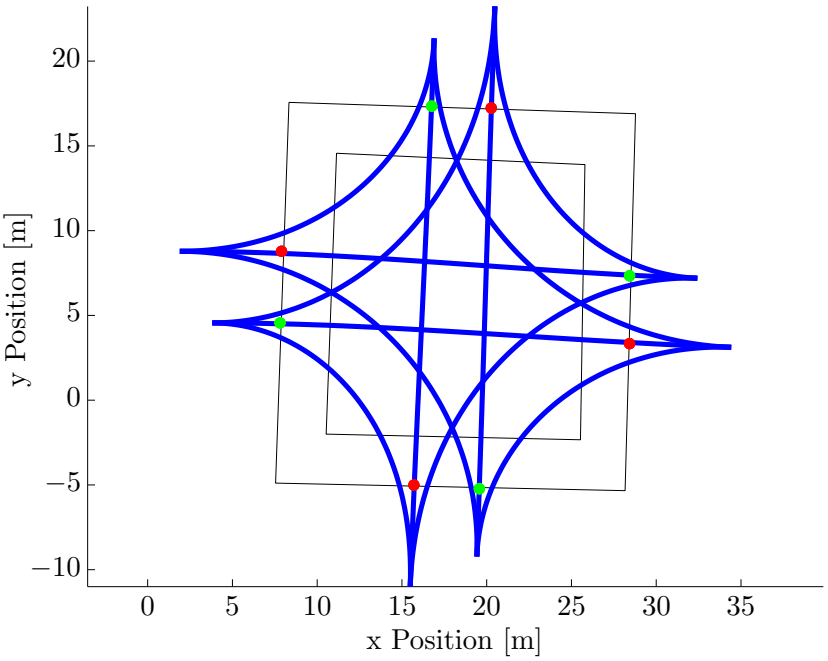

Figure 2.14: Generated Reference Paths.

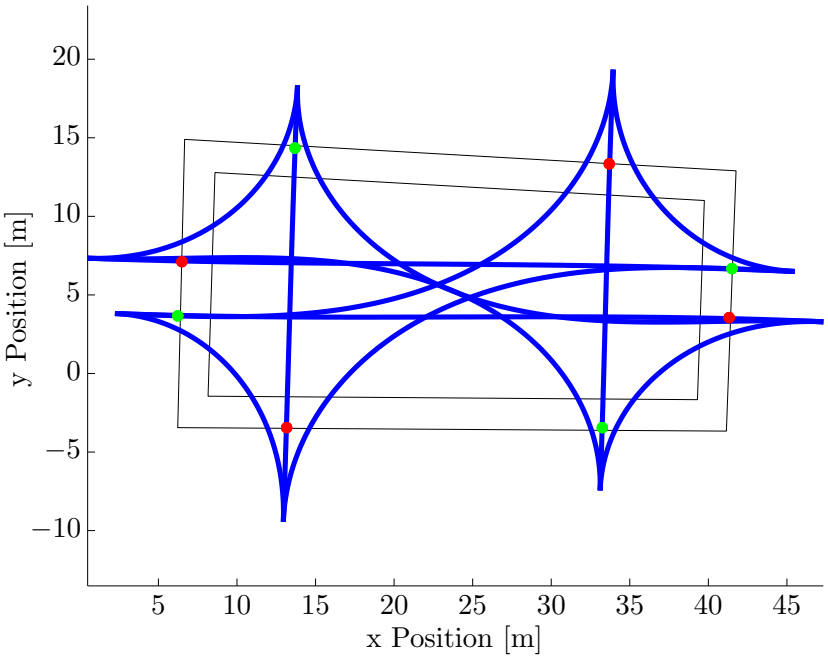

Figure 2.15: Generated Reference Paths.

2.5 Data Preparation

2.5.1 Raw Data

All the real world data collected had to be preprocessed offline in several steps before it could be used to train the naïve Bayesian filter. The first 15 minutes of collected data at the intersection of Santa Fe Avenue and Portland Avenue are shown in Figure 2.16. The blue square in the center of the figure represents the ego car recording the data. The car was parked as shown in the bottom image of Figure 2.9 facing towards the intersection.

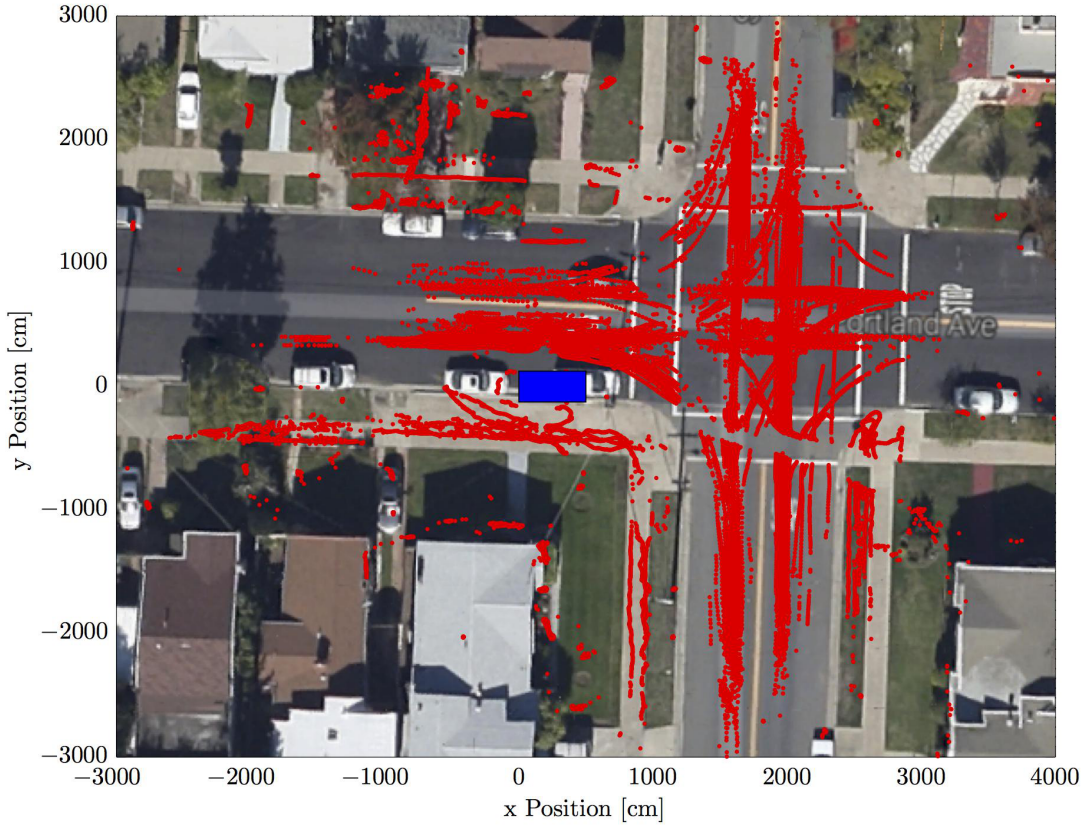

Figure 2.16: Raw sensor data after 15 minutes of recording. The location of the test car is shown as a blue square in the center of the figure.

2.5.2 Preprocessing of Sensor Data

The first step of preprocessing the raw sensor data was to group data packages with the same object ID together. The data packages are structured as shown in Table 2.1. Unfortunately, these packages are sent over the CAN bus at random times for any given object ID. It is therefore necessary to run through all the data and add them to packages with the same ID. However, because there are only a limited number of IDs available, they get reused. Therefore, one has to make sure not to group packages together which, although they have the same ID, belong to different physical objects. Analyzing the data has shown that a threshold of 0.1 milliseconds was large enough to group data of the same physical object together. But if an object ID has not been accessed for longer than that time and it gets reused, it will belong to a newly detected physical object.

Further on, static objects had to get filtered out. Because the test car was standing still, it would constantly track static objects such as other parked cars, vegetation like bushes and trees or even buildings. Due to the test setup and the static nature of these objects, they actually amount to the majority of collected data. However, it is useless information and has to be removed. Filtering these objects was done by removing all data groups that covered a trajectory of less then two meters in length and also all data groups that contained less than ten data points. Both those thresholds have been found empirically and have been proven to work as desired for the problem at hand. Then, the data is interpolated onto an equitemporal time vector. And finally, the data points are filtered by their geometric location. This means that moving objects outside of the road and intersection area, such as pedestrians on the sidewalk, are removed.

2.5.3 Labeling the Data

After all these filtering steps, the roughly 200 cars that have driven through the first intersection result in about 900 trajectories. There are more trajectories than actual vehicles because the vehicle paths get split up into several fragments. The first reason for this is that there are objects blocking the line of sight between the sensors and the tracked vehicle. The second reason, which is more common than the first, is that the sensor often loses track of an object and regains it shortly afterwards. Unfortunately, it assigns a new object ID to the same physical object in this process. This means that the two trajectories, from before and afterwards, will be handled like they are from two different objects. However, apart from the increased number of trajectories, this does not pose a problem for the algorithms implemented during this project.

All of these trajectories had to be labelled manually in order to have ground truth about the maneuver that the vehicle was performing. Therefore they had to be inspected by eye and a decision had to be made about the maneuver being performed. An interface was set up to handle this task efficiently. In uncertain cases, video footage recorded by the test vehicle was used for determination. From this process, every trajectory received two numbers, specifying the entry and the exit roads. The filtered and labelled data from Figure 2.16 is shown in Figure 2.17. For the given intersection, the maneuvers can be classified as left turns, going straight or right turns and have been coloured accordingly. Trajectory snippets of the approach phase, which lie entirely before the intersection, have been removed. After the labeling stage, the data was ready to be used for training or validation of an algorithm.

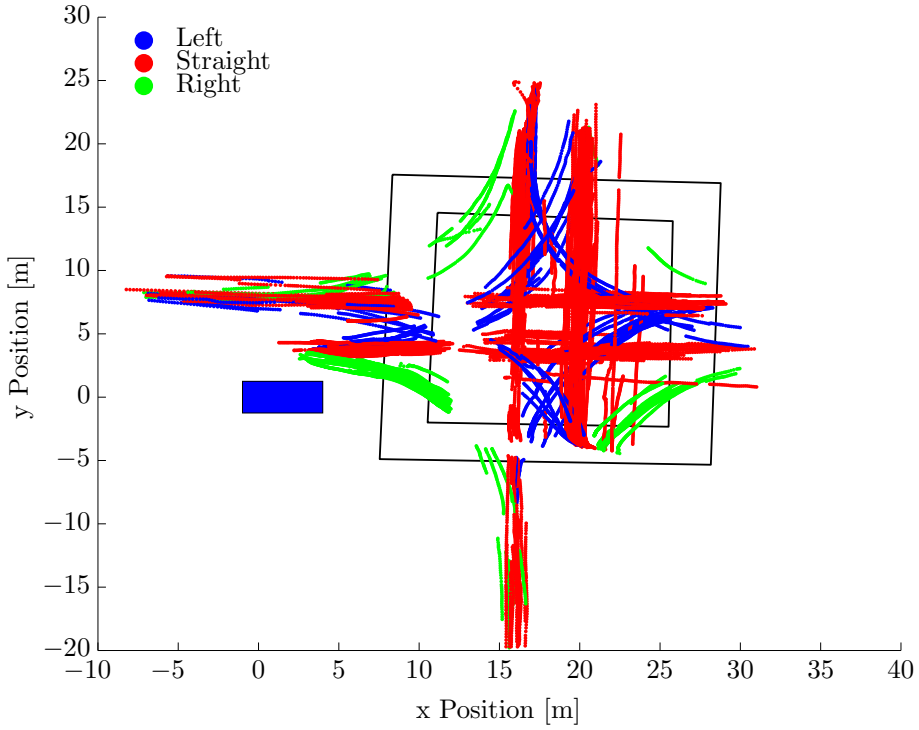

Figure 2.17: Data points after filtering and labeling, coloured according to the performed maneuver.

Chapter 3

Implementation of the Maneuver Prediction

This chapter explains the idea behind the naïve Bayesian filter for maneuver prediction, its implementation and its training. Further on, a reference model for comparison is introduced as well. The fundamental task assessed here is to estimate what maneuver an observed vehicle is performing as it navigates along its path through an intersection. The algorithm is explained at a simplified example based on the four-way stop intersection introduced in Figure 2.8. All explanations in this chapter are in reference to this intersection. However, the algorithm can handle arbitrary intersections of different geometry or size as well. For easier understanding, we assume that all vehicles enter the intersection from the western entrance and that they have the possibility to either go left, straight or right.

3.1 Idea

The basic idea to solve this problem is to create model trajectories as representations for each possible maneuver given an entry road. These reference trajectories all start from the same point and they are split up into equidistant segments. Doing that, every segment on a trajectory gets assigned to one segment on each of the other trajectories. This is given by the fact that they have the same curvilinear distance from the starting point, measured along the curved path to the respective segments.

When a vehicle is driving through our model intersection, its path is tracked and the curvilinear distance that the vehicle has travelled into the intersection is determined from it. This distance L is calculated as an arc length along the vehicles path. With this length, the segment on a reference trajectory that the vehicle would be in when assuming it had travelled along that ideal path can be determined. Calculating this for all three reference trajectories, one ends up with three segments whose properties can be compared against each other in order to find out which one of them the vehicle is most likely to be in. And therefore the maneuver that the vehicle is most likely performing at the moment can be found. This concept is visualized in Figure 3.1. The drawing on the left hand side shows the three reference trajectories, for a vehicle approaching from the western entrance of the intersection, split up into segments. The drawing on the right hand side points out the tree segments that will get compared to evaluate the maneuver for a vehicle that has advanced into the intersection. All of these three segments have the same curvilinear distance to the starting point as the vehicle which is equal to L.

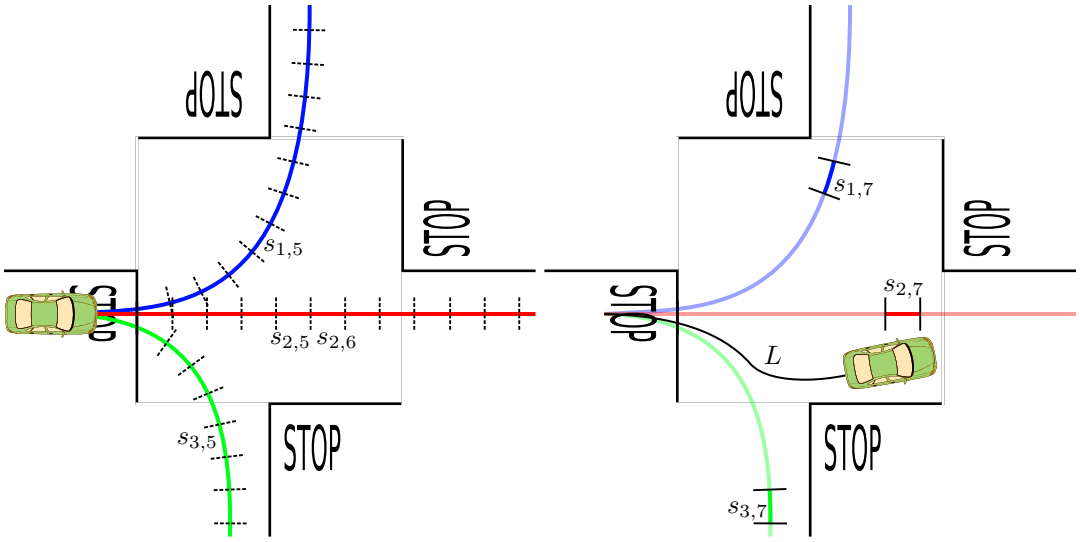

Figure 3.1: Segmentation of the Reference Trajectories.

3.2 Distinctive Properties

Based on a given set of measurements z_k, distinctive properties have to be emerged that are characteristic for maneuvers and allow the comparison of segments from different trajectories with each other. They should be as discriminatory as possible for the different maneuvers, but also general enough so that they are applicable to different intersection sizes and geometries. From these properties, the maneuver is estimated.

The first choice is the heading angle. It defines where the vehicle is headed in the near future and is therefore a good indication of the current maneuver. In order to use the orientation as a discriminant for arbitrary intersections, with possibly fewer or more roads joining or intersections with asymmetric layouts, the relative orientation in respect to the current segment on the reference trajectory has to be used:

$$\triangle\phi = \phi_{segment} - \phi_{vehicle}. \tag{3.1}$$

The second measure is based on the location of the vehicle. For example, it is clear that a vehicle located close to the upper exit of the intersection is more likely to be turning left rather than right. However, this measure has the same problem of generalization as the heading angle above. In order to be applicable to arbitrary intersections, the distance to the current segment on the reference trajectory given by

$$d = sign((x,y)_{segment},(x,y)_{vehicle})\sqrt{(x_{segment} - x_{vehicle})^2 + (y_{segment} - y_{vehicle})^2} \tag{3.2}$$

has to be used rather than the absolute position. The term $sign((x,y)_{segment},(x,y)_{vehicle})$ returns a positive or negative sign, depending on which side of the trajectory the vehicle is on. Figure 3.2 shows this distance to the trajectory d and the two angles $\phi_{segment}$ and $\phi_{vehicle}$ used to calculate the relative angle $\triangle\phi$.

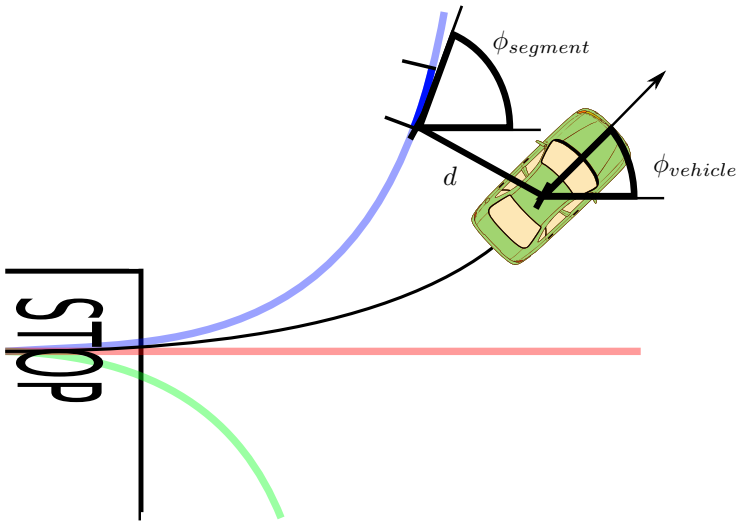

Figure 3.2: Calculation of the Relative Angle and Distance.

Further on, the vehicle velocity was analyzed as a possible distinctive property. This was on the basis that a vehicle navigating a sharp corner would be slower than one following a wide turn. And obviously, a vehicle going straight would have the highest velocity of all. However, after evaluating the real world data, it became clear that the velocity does not have the distinctive properties as it was hoped for. When considering a vehicle driving through an intersection, the velocity only comes in to play towards the exit. But by this time, it is already highly certain what exit it will choose based on the heading angle and location. During the most interesting phase, right when the vehicle enters the intersection, the velocity information is inconclusive due to two reasons: a) the vehicles have not had the time to actually accelerate to different velocities and b) a lot of vehicles do not make a full stop but rather a rolling stop which means passing the stop line at a low velocity. Therefore the velocity data has a large variance at the beginning of the intersection compared to the other measures. For those reasons, the velocity was dropped as a distinctive property.

3.3 Naïve Bayesian Filter

The method of choice to estimate the current maneuver intention from measurements of the vehicles position and orientation is a Bayesian filter. A Bayes Filter is the most general algorithm for propagating a probabilistic model for a system state which is a function of the time and a noisy sensor measurement. The algorithm can be found in the book of Thrun et al. [23]. The prefix "naïve" results from independence assumptions made in the measurement model. This technique is widely used in spam filtering for emails. A good introduction to it is given by Durrant-Whyte and Henderson [7].

3.3.1 Bayesian Inference

The Bayes rule plays a predominant role in probabilistic inference. It provides an instrument for combining the information about an object, given by the state x, with an observation z of that object. Based on the chain rule of conditional probability, it is commonly stated as

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)} \tag{3.3}$$

with $P(x|z)$ being the posterior probability and $P(x)$ the prior probability. $P(z|x)$ is the likelihood of observing z given x and $P(z)$ is the marginal likelihood.

3.3.2 Naïve Bayes

Under the naïve Bayes assumption, we state that conditioned on a state x, the components of an observed vector $z = [z^1, z^2, ..., z^n]^{\{T\}}$ are independent. This allows to write:

$$P(z|x) = P(z^1|x) * P(z^2|x) * ... * P(z^n|x) = \prod_{i=1}^{n} P(z^i|x). \tag{3.4}$$

Graphically, this is represented by Figure 3.3: z^1, z^2 and so on all depend on x. However, they do not depend on each other. This makes the vector z conditionally independent.

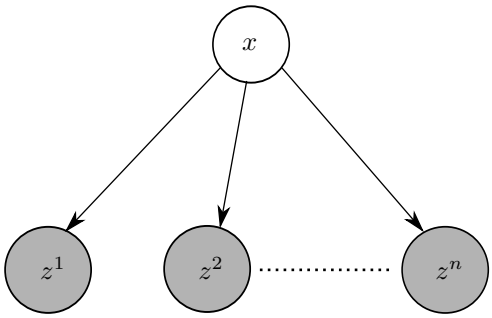

Figure 3.3: Conditional Independence.

3.3.3 Setup

To set up the filter, the state of the system at time k and the observations of that system at time k need to be specified. We therefore define the hidden state variable m. It is an element of the set of all possible maneuvers:

$$m_k \in M, \tag{3.5a}$$
$$M = \{\text{left}, \text{straight}, \text{right}\}. \tag{3.5b}$$

The goal of the filter is to estimate this state. The available measurements are the position in x and y direction, the heading angle ϕ and the segment number s. They are given at every time step k:

$$z_k = [x_k, y_k, \phi_k, s_k], \tag{3.6a}$$
$$x, y, \phi \in \mathbb{R}, \tag{3.6b}$$
$$s \in \mathbb{N}. \tag{3.6c}$$

This setup is represented by the Bayesian network shown in Figure 3.4. The segment number s_k is an integer and can not be measured directly. It is calculated according to Section 3.1 by integrating the distance travelled by the vehicle. It specifies which segments the vehicle can be in and is represented in Figure 3.1 as the second index.

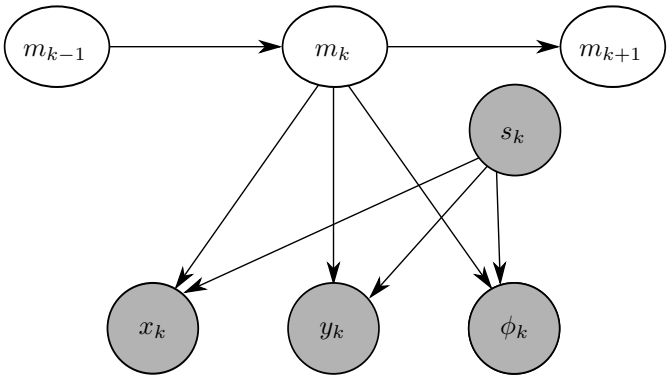

Figure 3.4: Bayesian Network.

Because this is a recursive filter, the initial probability distribution needs to be specified. A state of complete initial ignorance is assumed which is represented by a uniform initial distribution. For three states we write

$$p(m_0) = \frac{1}{3} \tag{3.7}$$

for every state. The filter is divided into two steps that are performed for every time step k: the prediction step and the measurement step.

3.3.4 Prediction Step:

For general problems, the prediction step is written as

$$P(x_k|z_{0:k-1}) = \int P(x_k|x_{k-1})P(x_{k-1}|z_{0:k-1})\,\mathrm{d}x_{k-1}. \tag{3.8}$$

The prediction probability $P(x_k|z_{0:k-1})$ is calculated by the integral over the product of the prior assigned to x_{k-1} with the probability of a state transition from x_{k-1} to x_k. Based on the assumption that the state transition follows a Markov process, the integration is only done over the states from the last previous time step x_{k-1}. For the problem at hand, the integral in Equation 3.8 is replaced by a sum over the discrete states m_{k-1}:

$$P(m_k|z_{0:k-1}) = \sum_{m_{k-1}} P(m_k|m_{k-1})P(m_{k-1}|z_{0:k-1}). \tag{3.9}$$

The prior probability $P(m_{k-1}|z_{0:k-1})$ is given by the initial probability distribution in the first step or results from the previous measurement update step. The transition probability $P(m_k|m_{k-1})$ is modeled as a discrete-time Markov chain, depicted in Figure 3.5. In this drawing, the capital letters in the nodes represent the states according to Equation 3.5. At every time step, the probability of staying in the same state is given by a. The probability of switching to another state is equal for all other states. For the three states at hand, this yields:

$$b = 1/2(1-a). \tag{3.10}$$

Using this, we can build the matrix of transitional probabilities:

$$P(m_k|m_{k-1}) = \begin{array}{c} \\ L \\ S \\ R \end{array} \begin{array}{c} \begin{array}{ccc} L & S & R \end{array} \\ \begin{pmatrix} a & b & b \\ b & a & b \\ b & b & a \end{pmatrix} \end{array}. \tag{3.11}$$

The value of the parameter a was determined by a parameter search documented in Section 3.4.3. The idea behind this Markov chain is not to be a model of the intersection. It is simply a filter to prevent excessive switching between the different states. Otherwise, more parameters would have to be fitted from data which would make the model more prone to overfitting and less applicable to arbitrary intersections.

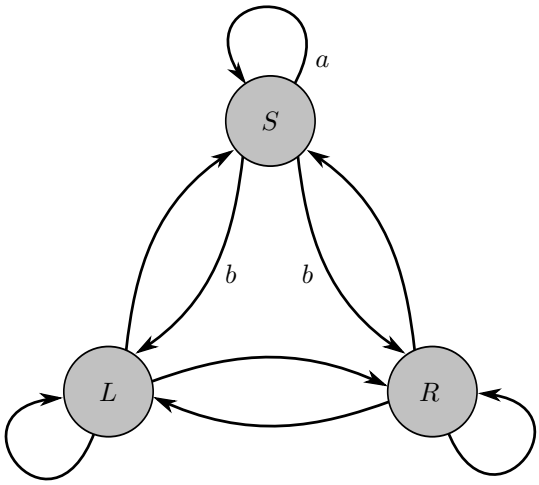

Figure 3.5: Markov Chain showing Transition Probabilities.

3.3.5 Measurement Step:

The measurement step follows after the prediction step. It is given by

$$P(m_k|z_k) = \eta P(z_k|m_k)P(m_k|z_{0:k-1}). \tag{3.12}$$

In this step, the prediction probability from above is multiplied with the probability that the measurement z_k is observed. The marginal probability is the same for all z_k and it is therefore replaced by the constant η. The measurement probability $P(z_k|m_k)$ is evaluated by assuming conditional independence of the measurements as shown in Section 3.3.2. We can therefore write

$$P(z_k|m_k) = P(d_k|m_k = M, s_k = S) * P(\triangle\phi_k|m_k = M, s_k = S). \tag{3.13}$$

The two probabilities on the right hand side of this equation are given by evaluating the probability density function of the distributions in Equation 3.16 at the values given by the measurements. The parameters of these distribution functions are learned from data. This is explained in detail in Section 3.4.

3.3.6 Probability Evaluation

After the measurement step, a decision has to be made about the maneuver that the vehicle is performing. For this, the probabilities $P(m_k|z_k)$ of each possible maneuver m_k are compared with each other and the one with the largest probability is said to be the current maneuver. The evaluation of the state probabilities concludes the maneuver estimation for the time step k.

3.4 Learning from Data

3.4.1 Variance Fitting

The data used to train the algorithm is half of all the data collected from intersection number one shown in Figure 2.8. There are two reasons why this intersection was chosen as the one to provide

training data. The first is that this is the simplest possible standard intersection: four streets with one driving lane each join each other at orthogonal angles. It is therefore a good general basis that can later be extended to more complex intersections of different geometries. The second reason behind this choice is that this is simply the most common intersection type due the chessboard layout of many residential and urban areas. At first, all data points were assigned to a segment, given their curvilinear distance from the start and the maneuver they are on, which is known from the manual labeling done earlier. Therefore, every segment of every reference trajectory had two data vectors assigned to it, one for the distance to trajectory d_s and on for the relative angle $\triangle\phi_s$. Then, the assumption was made, that the data in these vectors is normally distributed. To verify this, a Kolmogorow-Smirnow test was performed on the null hypothesis that the data, shifted by its mean and scaled by its standard deviation, stems from a standard normal distribution:

$$H_0 : F_X(x) = \mathcal{N}(0, 1). \tag{3.14}$$

The alternative hypothesis is that it does not come from such a distribution:

$$H_1 : F_X(x) \neq \mathcal{N}(0, 1). \tag{3.15}$$

For the relative angle $\triangle\phi_s$, 11.2% of the segments rejected the null hypothesis at the significance level of 5%, with an average p-value of 0.577. And for the distance to trajectory d_s, only 8.4% of the segments rejected the null hypothesis at the same significance level, with an average p-value of 0.521. It was therefore concluded that $\triangle\phi_s$ and d_s are normally distributed random variables.

For fitting the data to the reference trajectories, the further assumption was made that the distributions of d_s and $\triangle\phi_s$ have a mean equal to zero. In other words, we assume that the actual trajectories are normally distributed around the reference trajectory. This will simplify the adaption of the algorithm to arbitrary intersections later on. We therefore write:

$$d_s \sim \mathcal{N}(0, \sigma^2_{d,s}) \tag{3.16a}$$

$$\triangle\phi_s \sim \mathcal{N}(0, \sigma^2_{\triangle\phi,s}). \tag{3.16b}$$

Under the assumption that the mean is zero, the variance of the training data is calculated as follows:

$$\sigma^2(X) = \frac{1}{n-1}\sum_{i=1}^{n} x_i^2 \tag{3.17}$$

for d_s and $\triangle\phi_s$ respectively. The resulting standard deviations σ are shown in Figure 3.6. The upper plot shows the calculated values for the distance to trajectory and the lower one for the angular difference. There are twelve lines, one for each reference trajectory and the values are plotted against their position on the respective trajectory.

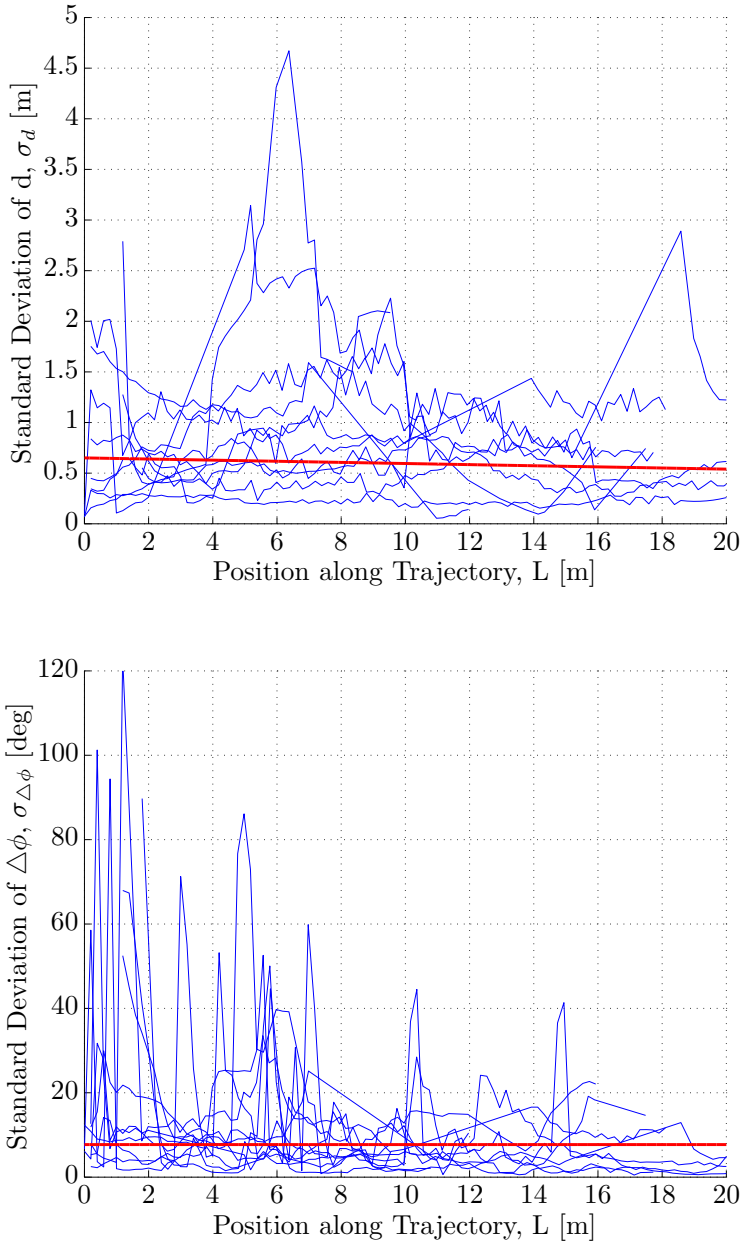

Figure 3.6: Standard Deviations from the Training Intersection.
(Linear and Constant Fit in Red)

In order to make these values valid for arbitrary trajectories, a more general representation had to be developed. First, a linear regression model was fitted to each of the twelve standard deviation plots. Then, the assumption was made that the deviations of the real world trajectories from the reference trajectory are the same for all reference trajectories. Therefore, these models were averaged to create one linear representation, valid for all reference trajectories. For the standard deviation of the distance to trajectory, σ_d, this fit is shown as a red line in the upper plot of Figure 3.6. Unfortunately, the linear fit of $\sigma_{\triangle\phi}$ had a negative slope and would lead to negative standard deviations for segments further away than about 17 meters. This makes this model infeasible for larger intersections with reference trajectories longer than that. Therefore, a constant model was fitted instead. The result is shown in the lower plot of Figure 3.6. To conclude,

we now have a lookup table that returns the variances of the angular difference and the distance to a reference trajectory, given the length L that the vehicle has progressed into the intersection. The numeric values are:

$$\sigma_d(L) = -5.5 * 10^{-6} * L + 0.6507 \ [m] \tag{3.18a}$$
$$\sigma_{\triangle\phi}(L) = 7.7193 \ [deg]. \tag{3.18b}$$

3.4.2 Comparison to a Second Intersection

As mentioned, these parameters were calculated from data of one intersection. In order to gain an understanding if these values are valid for other intersections as well, the same fitting was also conducted with data from a second intersection. It is the crossing of Washington Avenue and Key Rte Avenue. However, these values were only used as a comparison to assess the quality and accuracy of the fit and not in the verification process later on. They are given in Equation 3.19:

$$\sigma_d(L) = -1.6 * 10^{-4} * L + 0.6299 \ [m] \tag{3.19a}$$
$$\sigma_{\triangle\phi}(L) = 6.0351 \ [deg] \tag{3.19b}$$

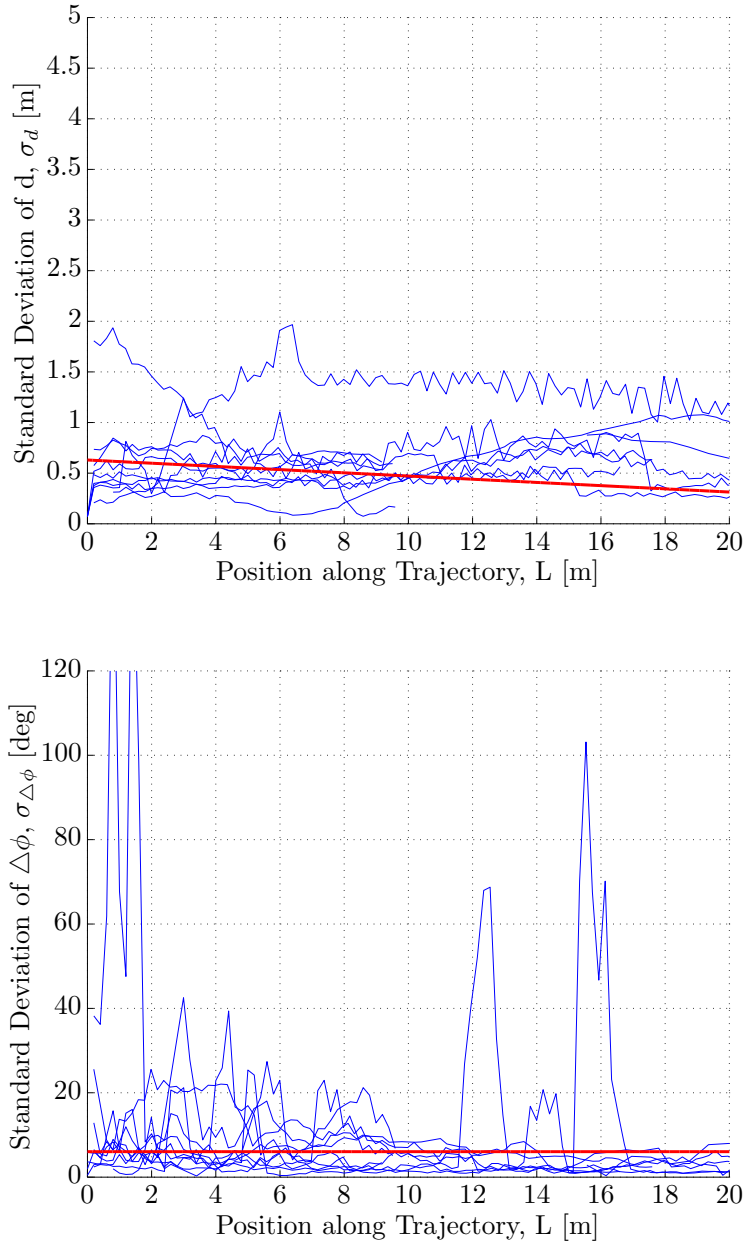

Figure 3.7: Standard Deviations from a Verification Intersection.
(Linear and Constant Fit in Red)

and shown in Figuer 3.7. To compare the two fits, the constant approximation of σ_d was calculated as well. This allows to compare the absolute difference of the constant parameters fitted for both intersections. These values are shown in Table 3.1. As a reference, the standard deviations of the fitted standard deviations are also in that table. First of all one can see that these standard deviations are in the same dimension for both intersections. Further on, it also shows that the absolute differences between the two constant fits are roughly a magnitude smaller for $\triangle\sigma_{\triangle\phi}$ and half a magnitude for $\triangle\sigma_d$ smaller than the standard deviations of the fitted values. Therefore, the difference between the fitted values from the two intersection is smaller than the standard deviation of the fitted values. This leads to the conclusion that the fitted model will transfer well onto other intersections.

	STD of Intersection 1	STD of Intersection 2	Absolute Difference between Intersections	
σ_{σ_d} $[m]$	0.57	0.34	0.12	$\triangle\sigma_d$ $[m]$
$\sigma_{\sigma_{\triangle\phi}}$ $[deg]$	12.70	11.91	1.68	$\triangle\sigma_{\triangle\phi}$ $[deg]$

Table 3.1: Standard Deviations of the Fitted Standard Deviations.

3.4.3 Tuning Parameter Optimization

The naïve Bayesian filter has a tuning parameter that can not be learned directly from data. A feedback from the performance metrics which are defined in Section 4.1 is necessary to optimize it. The value in question is the probability a that specifies the transitional probabilities in the Markov chain, introduced in Figure 3.5 and Equation 3.11. This value defines the probability that the state stays the same from one time step to the next. If this value is large, the system is conservative and will take a longer time to adjust to new measurements, but it can filter out noisy measurements. If it is small, the Markov chain will be more dynamic and it can adjust faster to wrong initializations. In order to find a value close to optimality, a grid search was performed on the training data set. The grid was set as $a \in \{0.05, 0.1, \ldots, 0.95\}$. The resulting correct classification rate and the 95% quantile of the distance to correct classification are shown in Figure 3.8. The value $a = 0.45$ was chosen to be a good compromise between the two performance metrics. Equation 3.10 allows the calculation of the ratio $^{b}/_{a} = 0.6111$. With this, the parameter a can be calculated for intersections with n roads joining together as:

$$a = \frac{1}{1 + 0.6111 * (n-2)}. \tag{3.20}$$

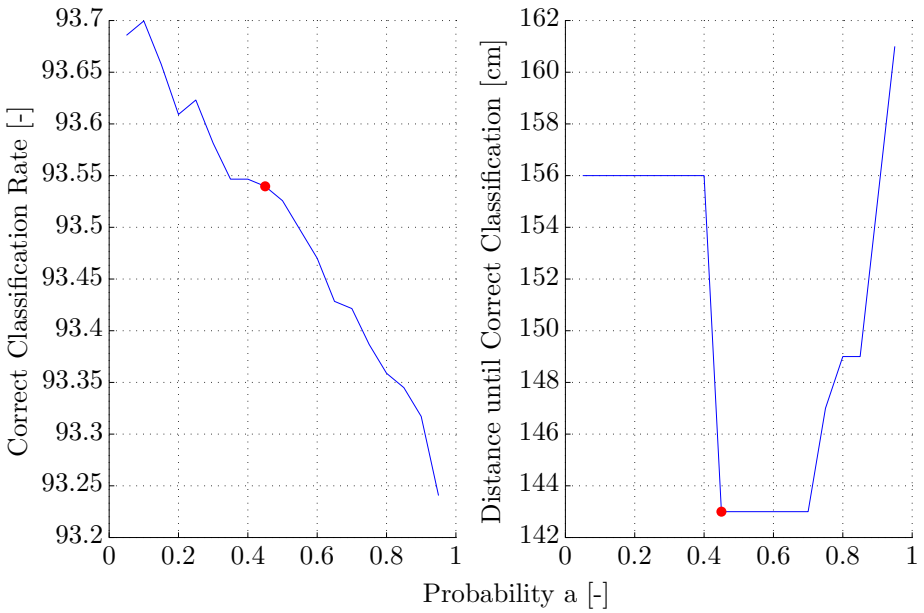

Figure 3.8: Parameter search for the transitional probabilities of the Markov chain.

3.5 Reference Model

In order to evaluate the performance of the naïve Bayesian filter, an alternative and reasonably simple solution to the problem was developed to serve as a reference model and benchmark. The idea of this approach is to take the sensor measurement of the current state of a vehicle, use a simple deterministic prediction model to predict its future trajectory and then decide which maneuver this trajectory corresponds to based on a threshold.

3.5.1 Constant Rate Model

Constant rate models are commonly used in target tracking algorithms. They predict the future movement of an object assuming the current change rates stay constant. The model applied here is a constant turn rate and constant acceleration (CTRA) model according to Zhai et al. [27]. This means that for each point on a trajectory, we assume the current acceleration and angular velocity to stay constant. We can therefore use them to integrate a kinematic point model over a given time t. The resulting trajectory describes a prediction of the objects motion for that integration time. We define the velocity vector of an object as

$$\vec{v}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix} = \begin{pmatrix} v(t)\cos(\phi(t)) \\ v(t)\sin(\phi(t)) \end{pmatrix} \tag{3.21}$$

and then find, due to the acceleration a and the turn rate ω being constant, the velocity along the path and the orientation as

$$v(t) = v_0 + \int_0^t a(\tau)\,\mathrm{d}\tau = v_0 + a_0 t \tag{3.22a}$$

$$\phi(t) = \phi_0 + \int_0^t \omega(\tau)\,\mathrm{d}\tau = \phi_0 + \omega_0 t. \tag{3.22b}$$

The position of the object can be written as

$$x(t) = x_0 + \int_0^t v_x(\tau)\,\mathrm{d}\tau \tag{3.23a}$$

$$y(t) = y_0 + \int_0^t v_y(\tau)\,\mathrm{d}\tau. \tag{3.23b}$$

Combining Equations 3.23 with 3.22 and 3.21 leads to the model describing the trajectory of an object given its initial conditions $(x_0, y_0, v_0, a_0, \phi_0, \omega_0)$:

$$x(t) = x_0 + \int_0^t (v_0 + a_0\tau)\cos(\phi_0 + \omega_0\tau)\,\mathrm{d}\tau \tag{3.24a}$$

$$y(t) = y_0 + \int_0^t (v_0 + a_0\tau)\sin(\phi_0 + \omega_0\tau)\,\mathrm{d}\tau. \tag{3.24b}$$

3.5.2 Maneuver Prediction

With Equation 3.24, the position of the vehicle after t seconds can be calculated with the initial conditions. These are given from sensor measurements. The turn rate ω and the acceleration a are calculated by the numerical derivatives of the orientation and the velocity. The predicted positions are then compared to threshold lines marked red in Figure 3.9. Those lines separate the intersection areas into three zones which determine the estimated maneuver depending on where the predicted final position lies. For example, if the position (x_t, y_t) lies below the lower red line, a

right turn is assumed. The threshold lines are created by constructing the midpoint mark, shown in the figure by a red dot, between the end points of two adjacent reference trajectories, marked as blue points. These midpoint marks are then connected with the entry point to the intersection to form the lines.

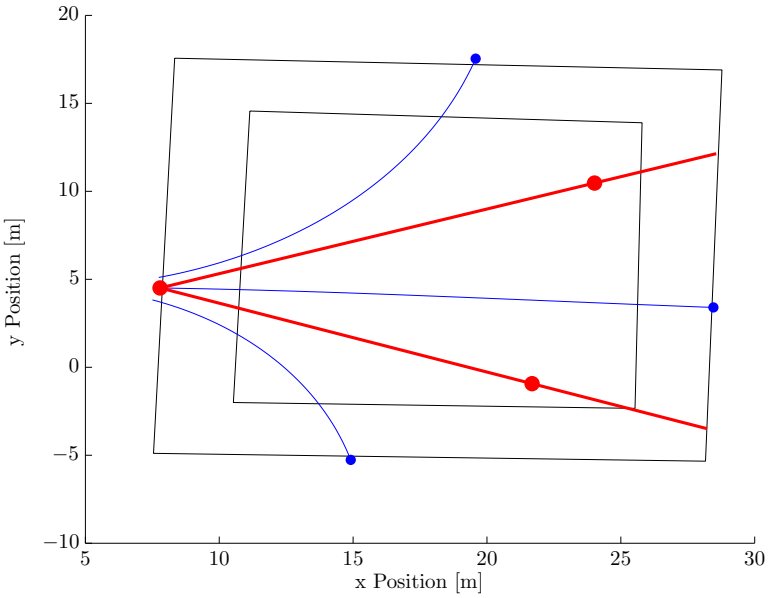

Figure 3.9: Construction of Threshold Lines.

3.5.3 Integration Time Optimization

The integration time t is a tuning parameter for this model. It can not be learned directly from data. A feedback from the performance metrics which are defined in Section 4.1 is necessary to optimize it. If the integration time is too long, the effect of the noise in the measurement will be increased strongly. However, if it is too short, the predictor loses performance and takes longer to make the correct classification. A grid search for t was conducted over the training data set. The calculated performance metrics are shown in Figure 3.10. The grid was set as $t \in \{0.2, 0.4, \ldots, 2.0\}$. The goal is to get a large correct classification rate and a small distance to correct classification. Resulting from that, the integration time was set to $t = 0.6$ seconds.

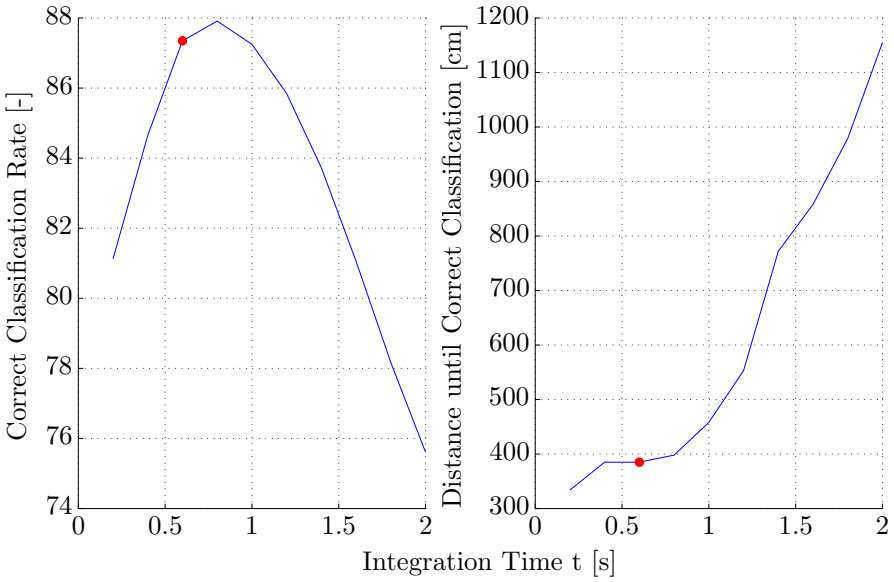

Figure 3.10: Parameter search for the integration time of the constant rate model.

Chapter 4

Evaluation of the Maneuver Prediction

4.1 Performance Metrics

In order to evaluate the performance of a prediction method, objective criteria have to be defined. The two used in this project are illustrated in this section.

4.1.1 Correct Classification Rate

When applying the maneuver estimation algorithms on a verification data set, each data point of a trajectory receives the estimated maneuver at that point as an additional piece of information. We can therefore check, how many of the data points have the correct estimated maneuver as we know the actual maneuver from the manual labeling process done in advance. The sum of all these correctly classified points, divided by the total number of points yields the correct classification rate for one algorithm:

$$\text{Correct Classification Rate} = \frac{100}{n} \sum_{k=1}^{n} CC_k \tag{4.1}$$

with the correct classification index:

$$CC_k = \begin{cases} 1, & \text{if } m_{k,pred} = m_{k,act} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

This is a simple method to assess the performance of an algorithm. However, it does not take into account where the wrong classifications were made. For example, wrong classifications further into the intersection have the same weight as errors right at the start. In real life however, the further into an intersection a mistake is made, the more severe it is. Because towards the exit, the performed maneuver becomes more and more obvious. Meanwhile, mistakes right at the start have to be tolerated simply because there is not enough distinctive information available that separates the maneuvers. Therefore, a different approach for performance evaluation was developed as well.

4.1.2 Distance until Correct Classification

For this measure, every trajectory is analyzed as whole and not point by point. First of all, the point on the trajectory where the classification was wrong for the last time is found. Then, the distance along the trajectory from its start to that point is calculated. This distance can be described as the

length from the first available piece of information to the point where the algorithm has classified the trajectory correctly, without any switching back afterwards. It is shown in Figure 4.1 and marked as L.

These distances from all trajectories are then put into a cumulative histogram. Out of this histogram, we can for example read the 95% quantil, which is the distance, after which 95% of all trajectories have been classified correctly. Figure 4.2 shows an example of this cumulative histogram. The naïve Bayesian filter is shown in blue. It starts at 80% which means that 80% of all trajectories are already classified correctly after the first measurement. The 95% benchmark is crossed after 1.41 meters. The maneuver prediction using the constant rate model can only reach that after 2.90 meters.

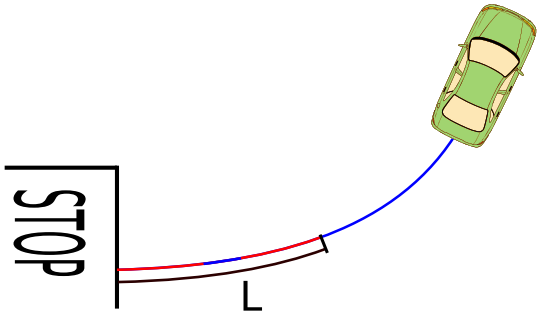

Figure 4.1: Distance until Correct Classification.
(Blue = Correct Classification, Red = Incorrect Classification)

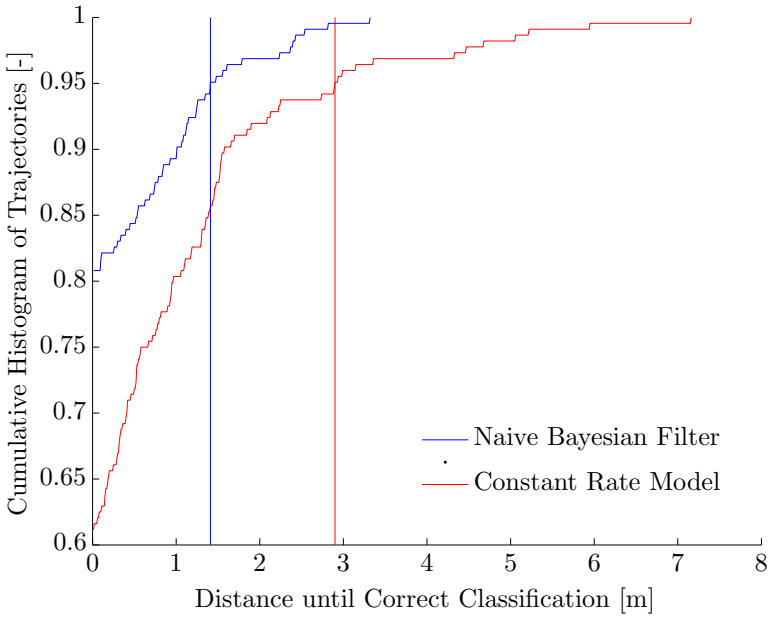

Figure 4.2: Cumulative Histogram of the Distance until Correct Classification.

4.2 Results

The following section presents all the results calculated by applying the concepts explained above to real world data that was collected at different types of all-way Stop intersections.

4.2.1 Standard 4-way Intersection

This is the intersection of Portland Avenue and Santa Fe Avenue shown in Figure 2.8. The first dataset was used to train the naïve Bayesian filter and to estimate the tuning parameters. A second set of trajectories from the same location is now used to evaluate the two estimation algorithms. Both maneuver estimation methods assign the right maneuver to over 90% of all data points. However, the Bayesian filter makes less errors further down the trajectories. It therefore only needs 1.41 meters to classify 95% of all trajectories correctly. This is a reduction of over 51% compared to the distance needed by the deterministic constant rate model. The mean is even reduced by almost 60%. Figures 4.3 and 4.4 visualize this results for the respective prediction approaches and entry streets. Correctly classified points are coloured blue and incorrect classifications are marked red. The reference trajectories and the threshold lines are shown in green as well.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	91.66 [%]	94.68 [%]	+3.29 [%]
95% Quantile of Distance until Correct Classification	2.90 [m]	1.41 [m]	-51.38 [%]
Mean of Distance until Correct Classification	0.55 [m]	0.22 [m]	-59.78 [%]
Standard Deviation of Distance until Correct Classification	1.12 [m]	0.57 [m]	-49.30 [%]

Table 4.1: Performance Evaluation at the Intersection of Santa Fe Avenue with Portland Avenue.

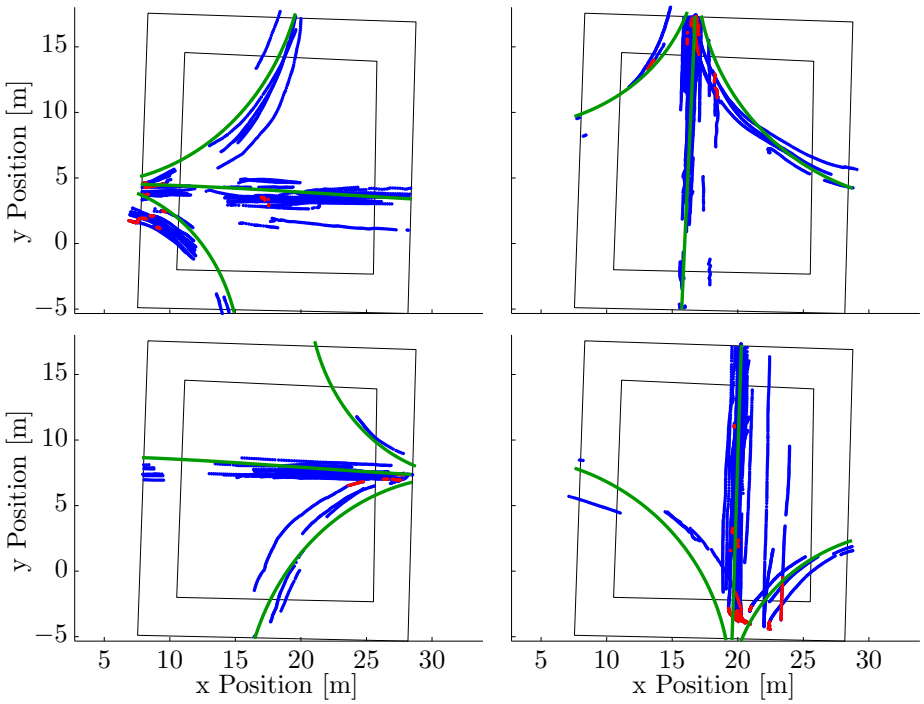

Figure 4.3: Maneuver Prediction using the Naïve Bayesian Filter.
(Blue = Correct, Red = Incorrect)

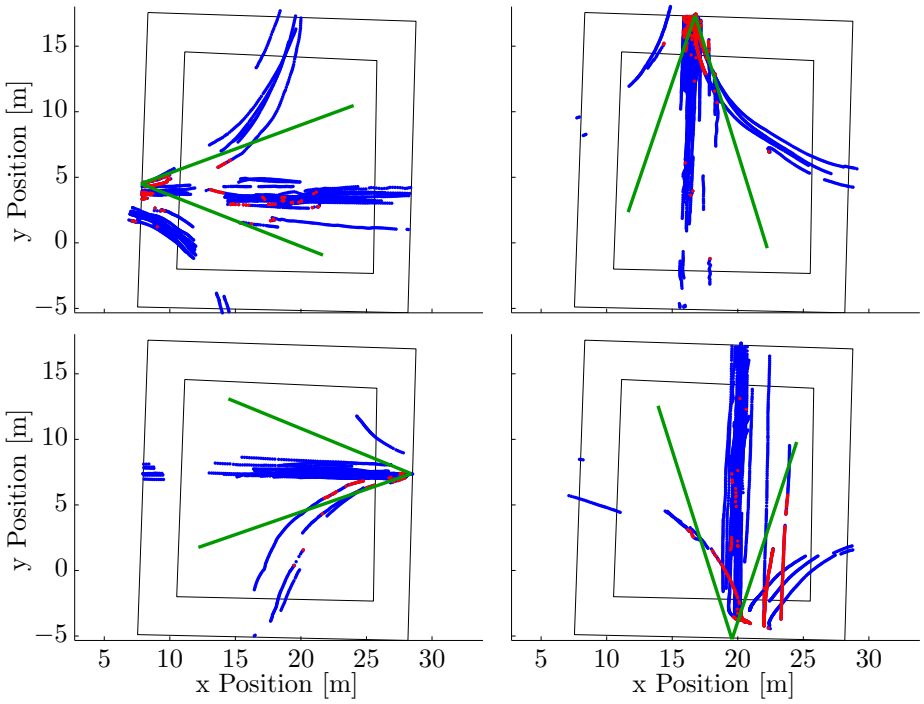

Figure 4.4: Maneuver Prediction using the Constant Rate Model.
(Blue = Correct, Red = Incorrect)

4.2.2 Small 4-way Intersection

This intersection, shown in Figure 2.11b, is a smaller but asymmetric 4-way intersection. It is located at the crossing of Washington Ave and Neilson St. Due to the asymmetry, the inner black square that shows the actual driving area is not rectangular but skewed. It is interesting to see that the correct classification rate is actually slightly higher for the deterministic prediction than for the Bayesian filter. However, the Bayesian filter still performs better overall as it reduced the classification distance to about two thirds.

For this dataset, it has to be said that there are not a lot of trajectories shown and also, that they are often split up into small pieces. The reason for this is that there was a problem with the data logging on the test car at the time of recording. This resulted in an overflow on the CAN bus and random discharge of sensor data. However, the data that was logged is still valid. Further on, this is also a small intersection in an urban settlement with very limited traffic in some directions.

One can see that for this intersection, the benefits of the probabilistic estimation is smaller compared to the first intersection tested. The reason for this lies mainly in the area size of the intersection. As there is less drivable space, vehicles do not have a lot of free room to chose their driving path. They rather have to follow a given path determined by the obstacles around it. This forces all vehicles to a very similar path through the intersection and it therefore reduces their variances. And the smaller the variances of the verification data get, the smaller is the benefit of a probabilistic model over a deterministic model.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	95.48 [%]	93.18 [%]	-2.41 [%]
95% Quantile of Distance until Correct Classification	1.98 [m]	1.33 [m]	-32.83 [%]
Mean of Distance until Correct Classification	0.42 [m]	0.28 [m]	-32.90 [%]
Standard Deviation of Distance until Correct Classification	0.89 [m]	0.51 [m]	-42.94 [%]

Table 4.2: Performance Evaluation at the Intersection of Washington Avenue with Neilson Street.

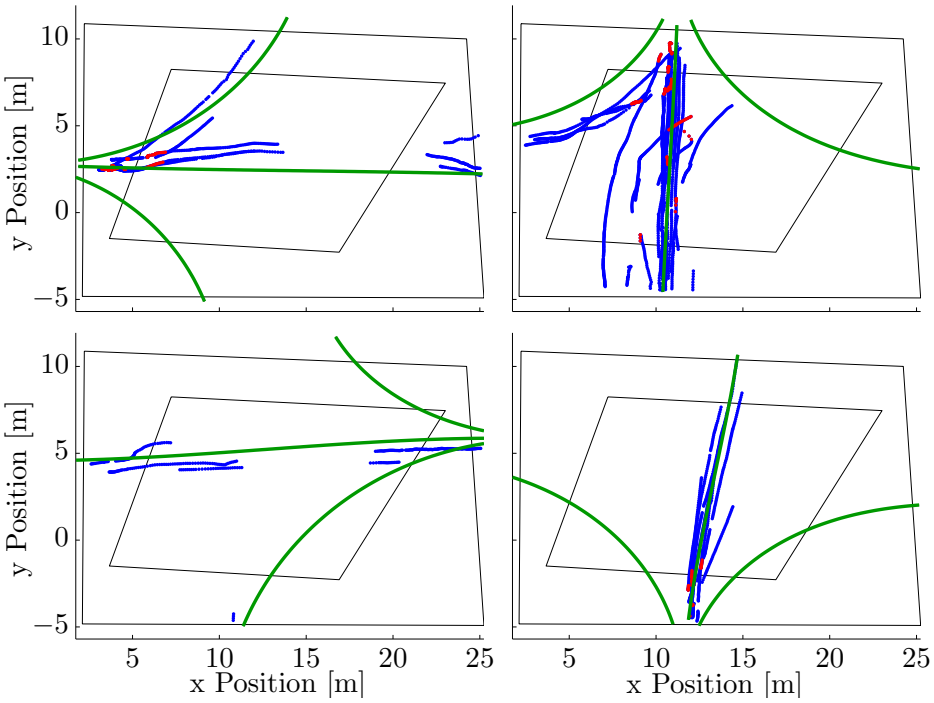

Figure 4.5: Maneuver Prediction using the Naïve Bayesian Filter.
(Blue = Correct, Red = Incorrect)

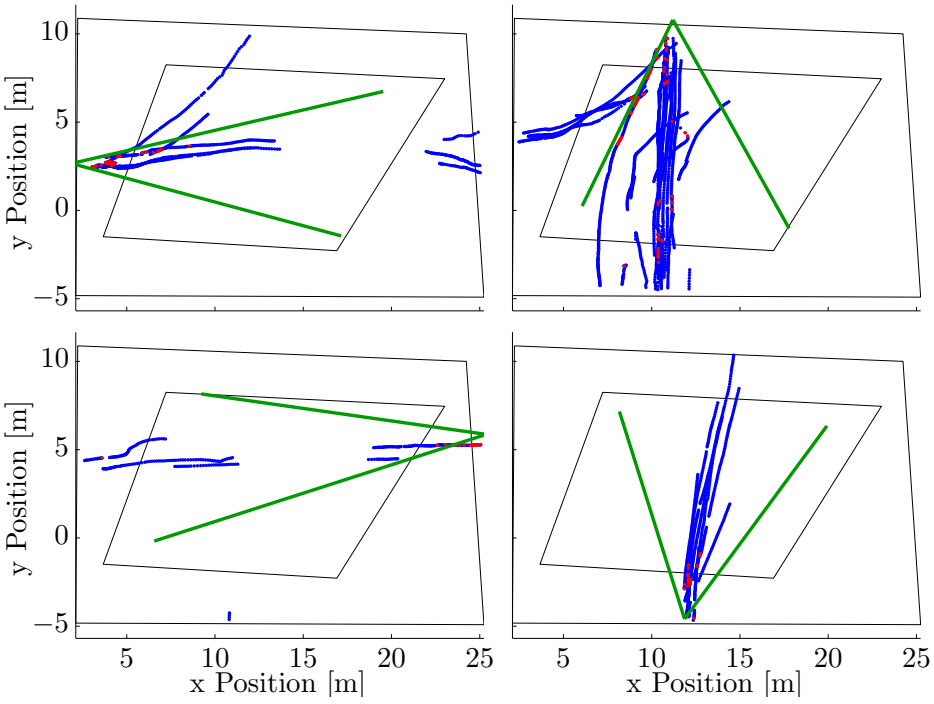

Figure 4.6: Maneuver Prediction using the Constant Rate Model.
(Blue = Correct, Red = Incorrect)

4.2.3 3-way Intersection

This is a classic T intersection where a minor street is discontinued and split up into the major street in cross direction. This intersection type is a subset of a standard 4-way intersection. Unfortunately, the data collection process at this intersection suffered the same setbacks as explained for the small 4-way intersection above: due to sensor issues, the trajectories are segregated into many small pieces. Also, there was very little traffic on the minor street. Nevertheless, on the trajectories that got recorded, the naïve Bayesian filter reduced the classification distance by over 50%.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	94.38 [%]	95.41 [%]	+1.09 [%]
95% Quantile of Distance until Correct Classification	1.39 [m]	0.68 [m]	-51.08 [%]
Mean of Distance until Correct Classification	0.20 [m]	0.15 [m]	-22.87 [%]
Standard Deviation of Distance until Correct Classification	0.55 [m]	0.77 [m]	-6.84 [%]

Table 4.3: Performance Evaluation at the Intersection of Pomona Avenue with Portland Avenue.

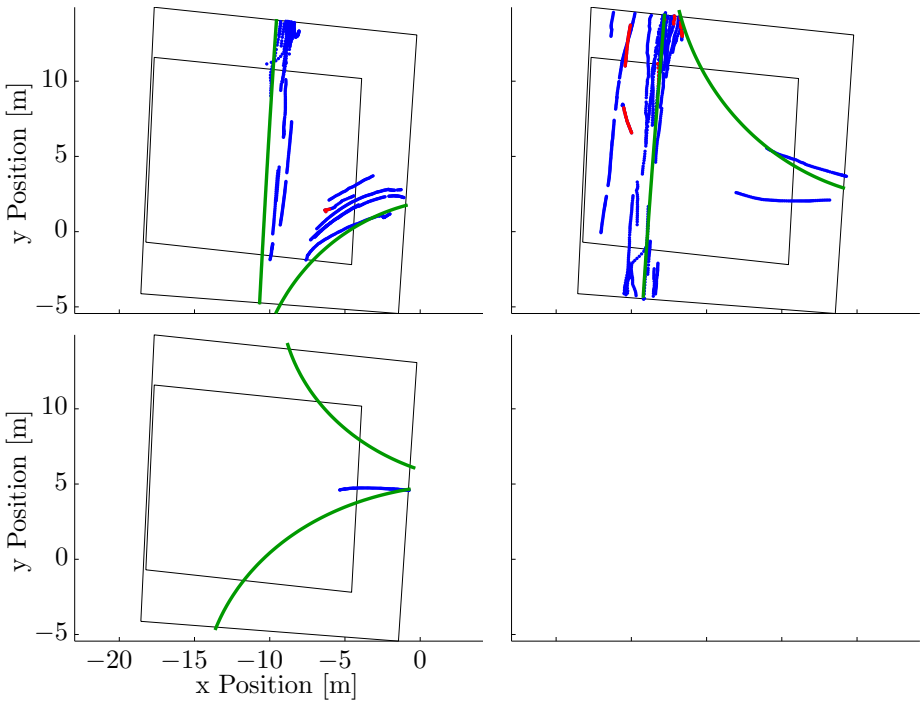

Figure 4.7: Maneuver Prediction using the Naïve Bayesian Filter.
(Blue = Correct, Red = Incorrect)

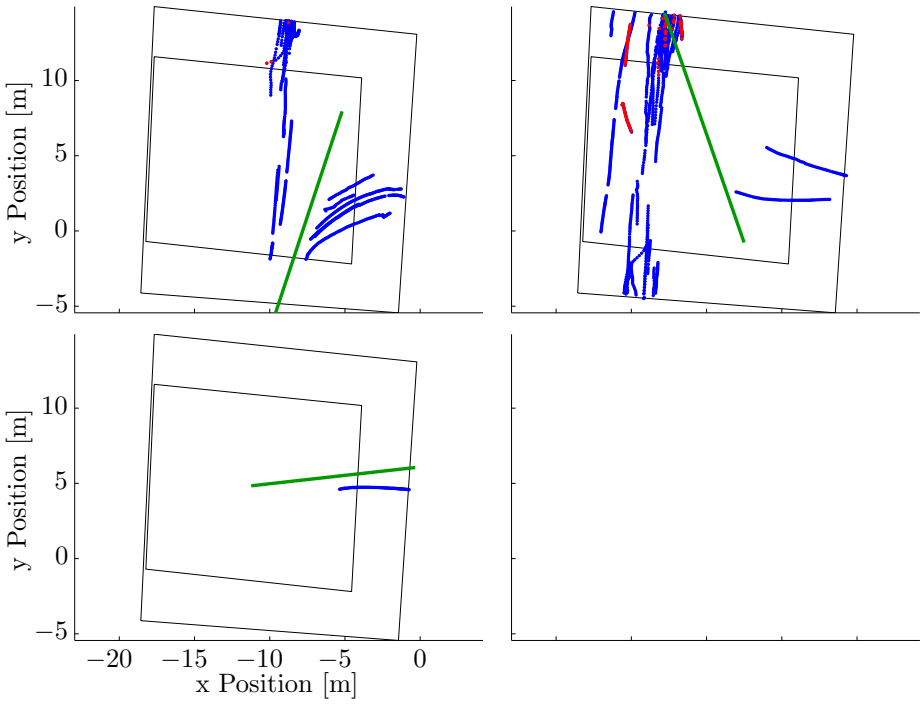

Figure 4.8: Maneuver Prediction using the Constant Rate Model.
(Blue = Correct, Red = Incorrect)

4.2.4 Large 4-way Intersection with Lane Division

The main road of this intersection is the Key Rte Boulevard running along the north-south direction. The the two lanes of this street are physically separated from each other by a vegetated strip which also allows parking on both sides of it. The cross street is a normal one lane road. This leads to an interesting intersection geometry characterized by its large area and the non quadratic, rectangular shape. Furthermore, this large intersection sees a lot of traffic. The simulation results are interesting because they show the lowest correct classification rate of all analyzed intersection types. This results form the elongation of the intersection in x direction: vehicles approaching from the west or east with the intention of going straight or left will travel along the same trajectory until they are almost half way into the intersection. During this phase, a lot of mispredictions occur because there is no significant difference in the vehicles behaviors that would allow to make a correct distinction between the two maneuvers. Nevertheless, the Bayesian filter still manages to classify over four percent more data points correctly compared to the constant rate model. However, the real performance benefit is shown by the 95% quantile of the correct classification distance: it is reduced to less than two thirds of the distance. In absolute numbers, this is a reduction of almost five meters.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	84.76 [%]	88.68 [%]	+4.62[%]
95% Quantile of Distance until Correct Classification	14.49 [m]	9.58 [m]	-33.89 [%]
Mean of Distance until Correct Classification	2.74 [m]	2.09 [m]	-23.76 [%]
Standard Deviation of Distance until Correct Classification	4.55 [m]	3.53 [m]	-22.41 [%]

Table 4.4: Performance Evaluation at the Intersection of Washington Avenue with Key Route Boulevard.

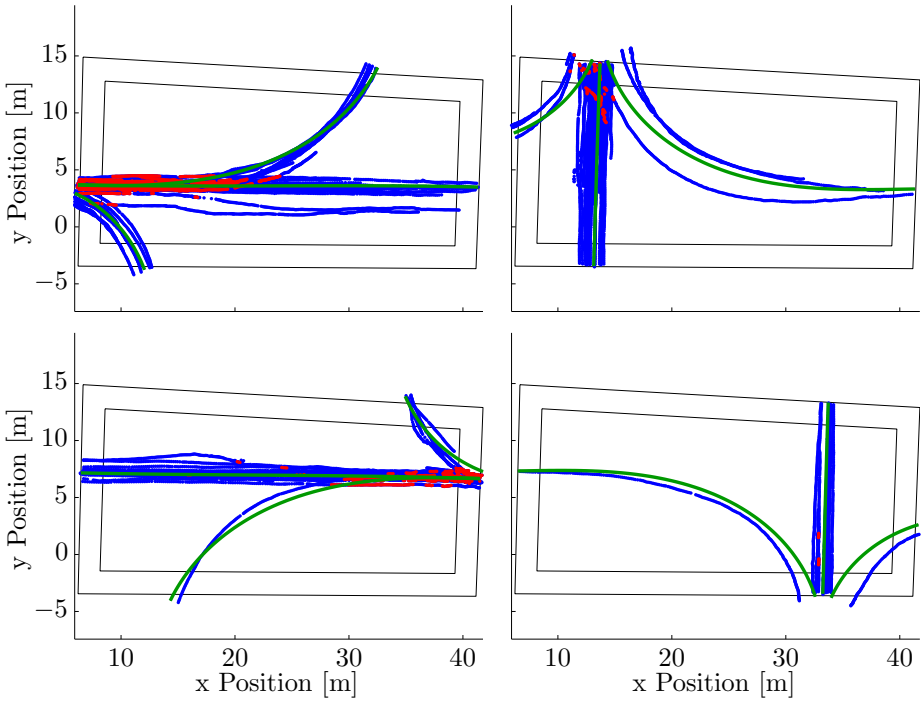

Figure 4.9: Maneuver Prediction using the Naïve Bayesian Filter.
(Blue = Correct, Red = Incorrect)

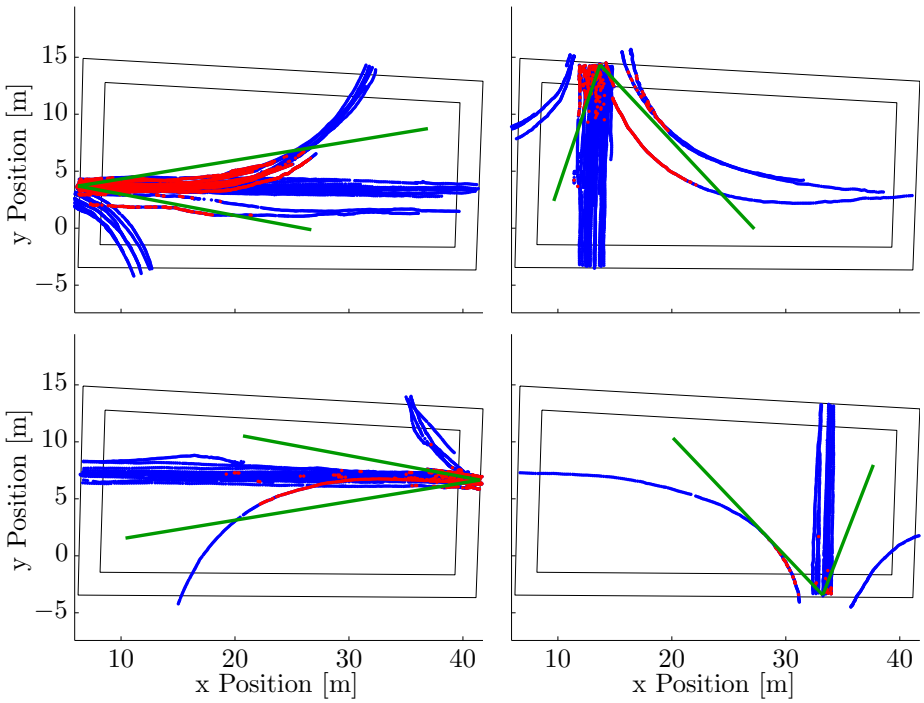

Figure 4.10: Maneuver Prediction using the Constant Rate Model.
(Blue = Correct, Red = Incorrect)

4.2.5 Highly asymmetric 4-way Intersection

The last intersection tested has the most complex geometry. Washington Avenue is the main road, crossing with Pomona Avenue. However, the entry of the main road running from east to west is shifted southwards by more than the width of the road itself. This leads to large curvatures in the reference trajectories that follow Washington Avenue. Even for this type of intersection, the Bayesian filter reduces the 95% quantile of the correct classification distance by over 17%. However, this is the worst performance improvement of all the intersections analyzed. The reason for this is that the openings of Washington Avenue into the intersection are kept very wide. This was probably done to make the navigation for larger vehicles easier. Unfortunately, it also allows normal vehicles to drive through the intersection on paths a lot different than the reference trajectory because they can cut the corners. An example of this problem can be seen in the top left plot of Figure 4.11 where the entry of the trajectories going straight and left are highly scattered. This scattering introduces many prediction errors.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	87.72 [%]	90.88 [%]	+3.60[%]
95% Quantile of Distance until Correct Classification	8.66 [m]	7.18 [m]	-17.09 [%]
Mean of Distance until Correct Classification	2.49 [m]	1.52 [m]	-38.96 [%]
Standard Deviation of Distance until Correct Classification	3.45 [m]	2.54 [m]	-26.34 [%]

Table 4.5: Performance Evaluation at the Intersection of Washington Avenue with Pomona Avenue.

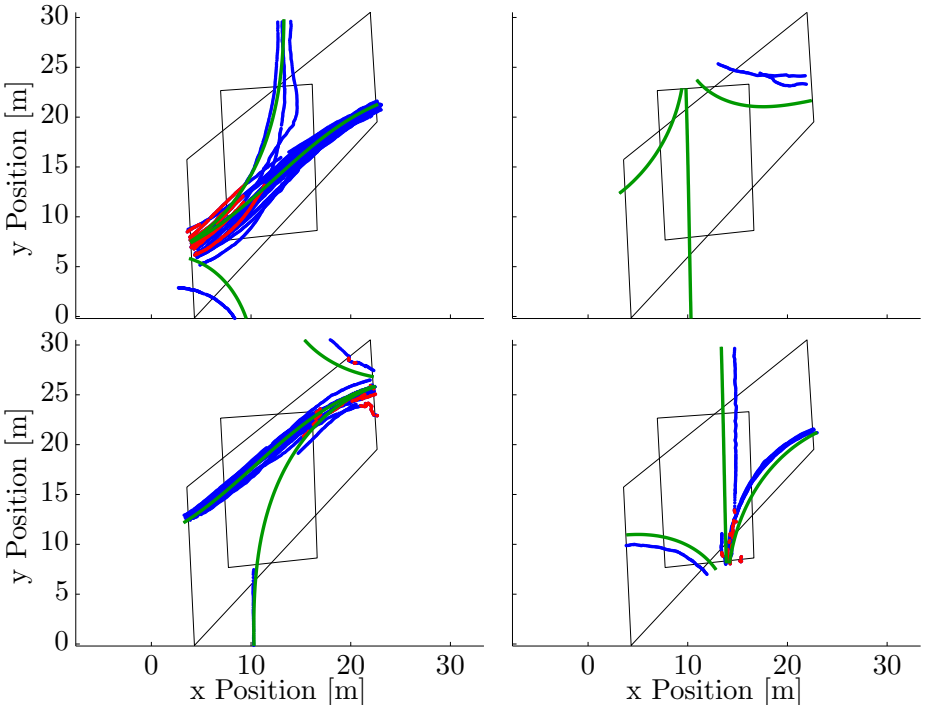

Figure 4.11: Maneuver Prediction using the Naïve Bayesian Filter.
(Blue = Correct, Red = Incorrect)

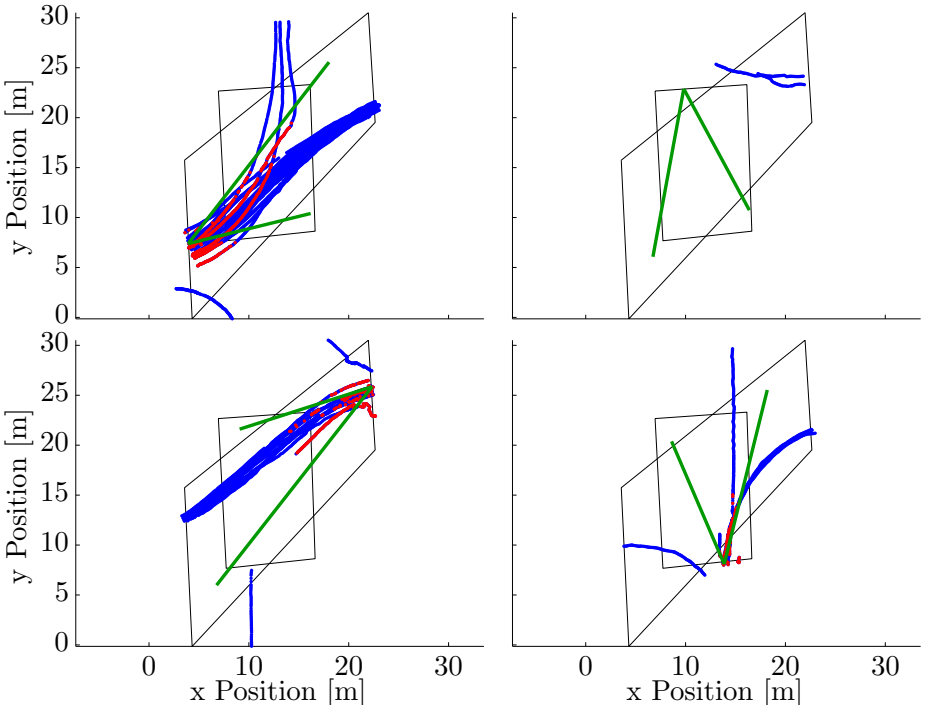

Figure 4.12: Maneuver Prediction using the Constant Rate Model.
(Blue = Correct, Red = Incorrect)

4.2.6 Combined Results

Overall, the data set used for the validation consists of 544 single trajectories. The numerical results from all these combined are given in Table 4.6. The naïve Bayesian filter manages to increase the correct classification rate by almost 2%. The more impressive result however is that it reduces the distance needed to achieve a correct classification by 30%. This is the case for the 90% and the 99% quantile and for the mean of this distance. We can therefore conclude that even though the Bayesian Filter increases the correct classification rate only slightly, it significantly reduces the distance needed to classify a trajectory correctly.

	Constant Rate Model	Naïve Bayesian Filter	Relative Improvement
Correct Classification Rate	89.90 [%]	91.69 [%]	+1.99 [%]
90% Quantile of Distance until Correct Classification	3.20 [m]	2.24 [m]	-30.00 [%]
99% Quantile of Distance until Correct Classification	14.02 [m]	9.87 [m]	-29.60 [%]
Mean of Distance until Correct Classification	1.07 [m]	0.72 [m]	-32.54 [%]
Standard Deviation of Distance until Correct Classification	2.42 [m]	1.97 [m]	-18.74 [%]

Table 4.6: Combined Results from all Validation Trajectories.

4.2.7 Decision Threshold

In order to finalize the decision when to enter an intersection, a measure had to be developed to decide when a maneuver prediction is accepted as correct. One can think of this as a threshold on the probability that a certain maneuver is being performed. If a tracked vehicle navigates through an intersection, it will arrive at a point where the probability of one maneuver is so high, that all other possible maneuvers can be excluded. The ego vehicle waiting at the intersection can therefore start to act based on that information and enter the intersection, if their paths will not cross. However, the probabilities returned by the naïve Bayesian filter show very abrupt switching behavior while staying at values close to zero or one respectively for most of the time. This behavior is shown in Figure 4.13 with probabilities calculated for maneuvers of a measured trajectory. A decision based on a threshold does not make sense here because that value would already be passed by the first few data points, when a certain decision clearly can not be made yet. Therefore, an alternative approach was necessary.

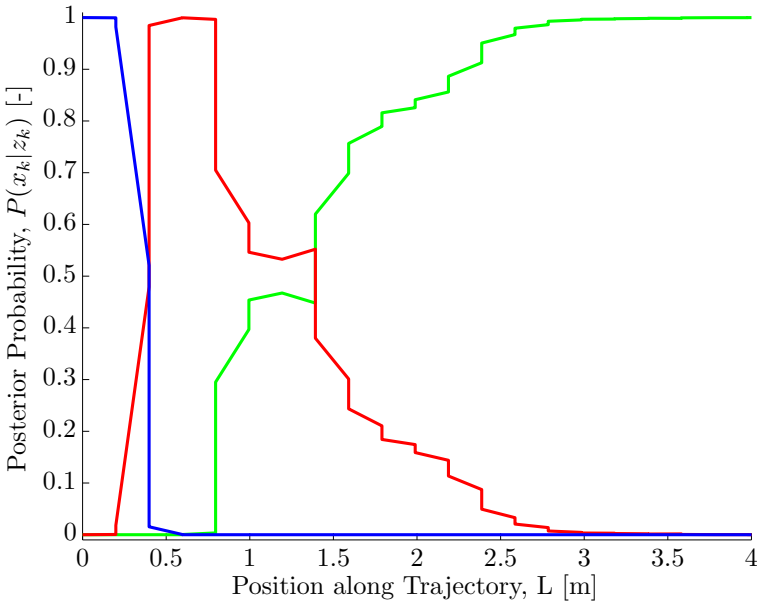

Figure 4.13: Example of three Posterior Probabilities.

While analyzing the data, it became clear that the point where this decision is possible depends primarily on the size of the intersection. At a smaller intersection, it is possible to make a decision on the maneuver a lot earlier than at a large intersection, because on one hand there is less distance to travel and on the other hand, there is less available room to deviate from the reference trajectory. The size of the intersection can be further specified as the length of the trajectory that the vehicle is on. Considering the intersection in Figure 4.9: a left turn performed after arriving from the western entrance, shown in the upper left hand plot, will take a longer distance to be classified correctly than a left turn after arriving from the northern entrance, shown in the plot on the upper right hand side. A more abstract definition of this length can be found by using the average length of all reference trajectories from the same entry road. This value is proportional to the size of the intersection and easily calculated as all necessary information about the intersection is already available from the reference trajectories. A more accurate representation would obviously be achieved by only taking the length of the one trajectory that the vehicle is on. However, this is inherently infeasible because that trajectory is the estimated state and using the estimated maneuver trajectory can lead to wrong thresholds in case of misclassifications.

Figure 4.14 shows the distances necessary to classify a trajectory correctly, plotted by a point for each trajectory. The values on the x axis are the average length of the reference trajectories for the given entry to the intersection. The points are therefore grouped by intersection entries and the trajectories from all validation data sets are shown. The red line shows a manual fit for a threshold, after which the predicted maneuver is correct. This line is defined by the equation

$$y = 1.38 * x - 16.3 \tag{4.3}$$

with x being the average curvilinear length of the reference trajectories of the respective intersection entry and y the length of a trajectory after which a certain statement can be made about the maneuver being performed.

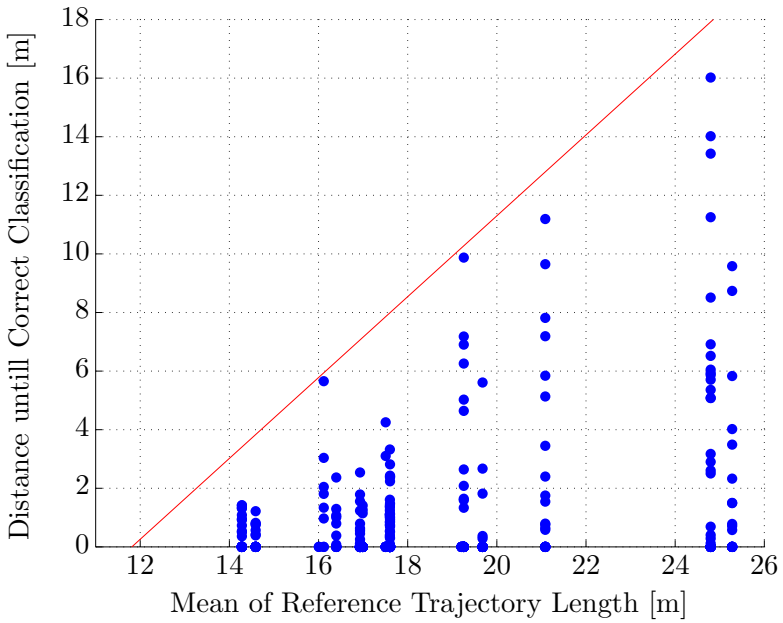

Figure 4.14: Required Distance to Correct Classification as a Function of Intersection Size.

Chapter 5

Right of Way Estimation

5.1 Model

As mentioned in the introduction, the right of way at a stop intersection is defined by the arrival time of vehicles at the intersection. The vehicle arriving later will yield the right of way to the vehicle arriving earlier. This concept is therefore based on an arrival time ranking and an approach had to be developed to estimate this arrival time. Estimating the arrival time of an observed vehicle is not a trivial task as it depends a lot on the driver and situation at hand: most drivers do not stop completely and pass the sign at a low speed while some stop very abruptly. Also, some vehicles have their point of lowest velocity far before the stop line while others have it after the sign, in the intersection. It is therefore difficult to specify where and when a vehicle has actually arrived at the intersection. The problem is defined in the scheme depicted in Figure 5.1. It shows a car approaching a stop line with the velocity $v(t)$. The distance to the line is measured backwards and denoted by $x(t)$.

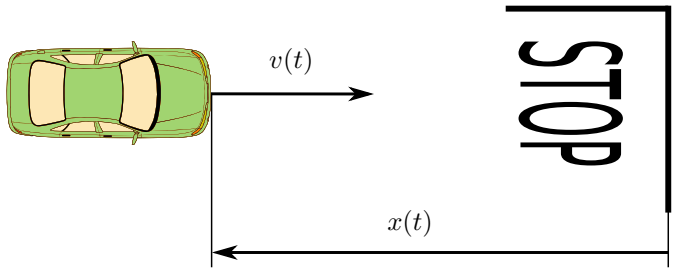

Figure 5.1: Vehicle Velocity and Distance to STOP.

Initial approaches based only on the distance to the stop line or the vehicle velocity were deemed to be to simple for the problem at hand. The former one because vehicles stop in a wide zone around the stop line. In the test data at hand, this zone was roughly defined from three meters before the stop line until one meter after it. Taking the distance to the same mark for all vehicle would therefore lead to inaccurate results. The latter approach, based on the velocity, did not work due to many vehicles not coming to a complete stop which makes it difficult to specify when a vehicle has actually arrived at the intersection. To outmaneuver these limitations, a more advanced approach was implemented. It is based on the following system of ordinary differential equations:

$$\dot{x}(\tau) = v(\tau) \tag{5.1a}$$
$$\dot{v}(\tau) = k_x * x(\tau) + k_v * v(\tau) \tag{5.1b}$$

with the initial conditions

$$x(0) = x(t) \tag{5.2a}$$
$$v(0) = v(t). \tag{5.2b}$$

This system is used to predict the future motion of a vehicle. The acceleration acting on the model is defined by the distance to the intersection plus the vehicle velocity and each multiplied by a gain factor. The model is integrated for every sensor measurement by using the measured vehicle position and velocity as initial conditions. The integration is stopped when the model crosses the stop line defined as $x(\tau) = 0$. The time that it takes the model from the initial position to arrive at point zero is the estimated time until stop. The estimation is rerun for every sensor measurement and it will get more accurate the closer the vehicle gets to the stop sign.

5.2 Parameter Optimization

The parameters k_x and k_v have been found by minimizing the difference of the predicted arrival times compared to actual arrival times given by a set of training data:

$$\min_{k_x,k_v} \sum_i |t_{i,est} - t_{i,act}|. \tag{5.3}$$

The set consists of twelve approach trajectories with a total of 2073 data points. For each of these points, the time that it took the actual vehicle from there to the stop line was calculated. This time was then compared to the time predicted by the model for that point. The resulting time difference for all data points was summed up. The gain parameters were found by minimizing this final time. The resulting parameters are stated in Table 5.1.

k_x	-1.5741 [1/s²]
k_v	-1.7820 [1/s]

Table 5.1: Fitted Gain Parameters.

5.3 Performance Verification

Based on this setup, the approach was verified on a set of twelve trajectories with a total of 1996 data points. Overall, an average absolute difference between the predicted and the actual arrival time over all data points of 0.7022 seconds was achieved. The standard deviation was 0.6618 seconds. Figure 5.2 shows such a time to intersection prediction for a trajectory of roughly nine meters. The upper plot displays the measured velocity profile for the vehicle approaching from the left hand side with the stop sign set at the zero mark of the x axis. The blue line in the lower plot shows the time it took the vehicle from that data point until the vehicle crossed the stop sign. And the red line is the time predicted by the model. One can see that apart from an initial offset at the start of the prediction, the model predicts the time within a reasonable boundary. However, the models limitations become clear when considering Figure 5.3. One can see that the predicted time to arrival is about 2.5 seconds lower than the actual time. This errant behavior is a result of the abrupt breaking maneuver performed in the middle of the approach which can obviously not be foreseen by the prediction. Later, the prediction flips into the other negative and over predicts the time by more than a second for almost the complete rest of the approach distance. This happens because at that point, the velocity of the vehicle, which is used as an initial condition for the model, is very low. Clearly, the model introduced here is not capable of handling this type of behavior. However, this is an exception. From our data set, the proposed model was able to handle 90% of all approach trajectories.

Average Absolute Error	0.7022 $[s]$
Standard Deviation	0.6618 $[s]$

Table 5.2: Resulting Performance.

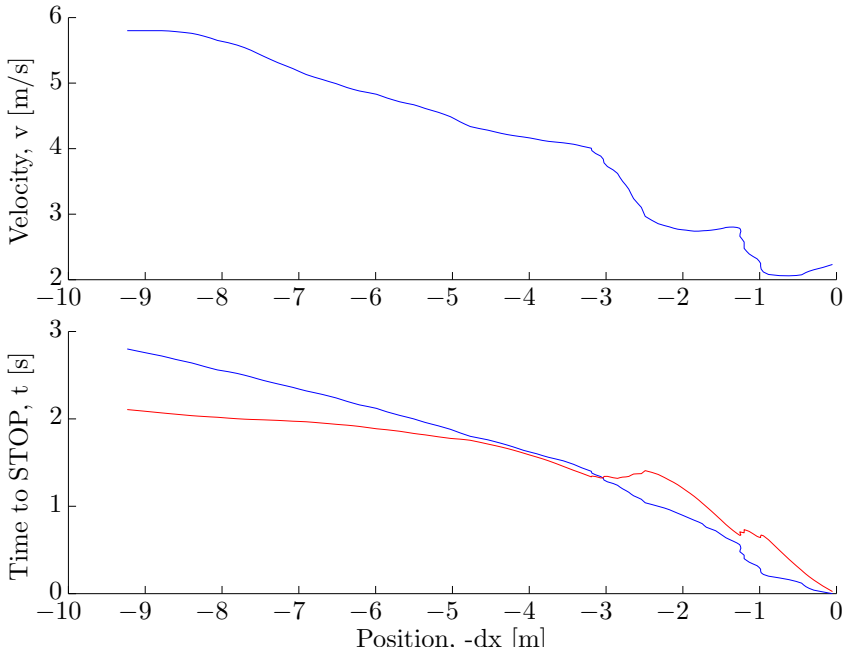

Figure 5.2: Time Prediction Performance I.

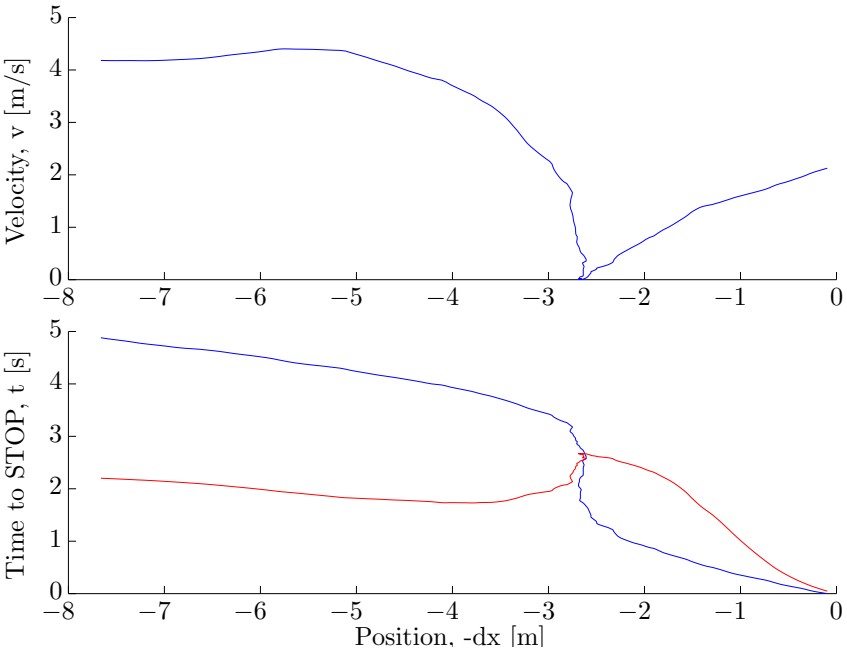

Figure 5.3: Time Prediction Performance II.

Chapter 6

Conclusion

6.1 Summary

During the course of this project, a method for maneuver prediction at stop intersections was developed. It was shown that this method is applicable to arbitrary intersections of different geometries and sizes. Further on, a right of way estimator was implemented for these intersections. Both of these methods were trained and later verified using real world data.

The data needed for this project was collected using a test vehicle equipped with LIDAR sensors. It was parked next to different stop intersections to track vehicles maneuvering the intersection. The data was then processed offline and labelled manually to have ground truth about the performed maneuvers.

The first and main part of the project focused on the implementation and verification of a maneuver prediction algorithm based on a naïve Bayesian filter. This filter was trained using data from a standard four-way intersection and was later verified with data from five intersections of different sizes and layouts. It was shown that this approach successfully adapts to these intersections. A comparison was made with a deterministic predictor as a reference model. Overall, the Bayesian filter reduces the distance necessary to classify trajectories correctly by 30% compared to the reference model. Further on, a linear threshold depending on the size of the intersection was introduced to decide when a prediction is final.

The second part of the project deals with the estimation of the right of way at a stop intersection. It is solved by predicting the arrival time of vehicles at the stop sign. For this, a kinematic feedback model was tuned with real world data of vehicles approaching stop intersections. The model was then validated successfully on 90% of the datasets. However, the model was unable to handle some scenarios. For those exceptions, a more sophisticated approach needs to be developed.

With these two parts combined, an autonomous vehicle is able to handle arbitrary all-way stop intersections according to the legal transportation code. It can create a ranking which defines the right of way between the different vehicles. Then, it will focus on the last vehicle that has the right of way over itself. For this vehicle, the performed maneuver is estimated as it navigates through the intersection. At the point where certainty about the intended action of the observed vehicle is reached, the autonomous car is able to make a decision when it can enter the intersection.

6.2 Key Issues

Two key issues that came up during this project can be named. On one hand, it is the fact that stop intersections have rarely been treated in the concept of environment prediction. There is next to no literature available about the treatment of stop intersections for autonomous vehicles. Most of the previous work on intersections deals with signal controlled intersections. These have less degrees of freedom than stop intersections and are a more controlled environment.

The second key issue arouse during the data collection and processing. This task demanded a

lot of overhead. On one hand, a large amount of post processing of the sensor data was necessary to retain useful datasets. This task consumed a lot more time than initially anticipated. On the other hand, there were also problems with the actual sensor system. An overload on the CAN bus led to uncontrollable data loss. At first, this passed unnoticed but it later required the rerun of several data collection cycles.

6.3 Outlook

The final goal will be to implement the presented methods on the test vehicle. The big challenge to achieve this goal will be to realize online data processing to transform the raw sensor data into useful information for the algorithm. In this project, all the processing was done offline. Further on, more data will be needed to attain a better calibration of the trained parameters. The actual maneuver prediction problem of this project however is considered solved. More work will need to be done on the prediction of the arrival time of vehicles at an intersection. This method does not cover all eventualities occurring in a real world environment and it therefore needs improvement.

Bibliography

[1] California Vehicle Code. Division 11, Rules of the road. Chapter 4, Right-of-Way, [21800]. `http://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=VEH&sectionNum=21800.`

[2] Georges S Aoude, Brandon D Luders, Kenneth KH Lee, Daniel S Levine, and Jonathan P How. Threat assessment design for driver assistance system at intersections. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1855–1862. IEEE, 2010.

[3] Georges S Aoude, Vishnu R Desaraju, Lauren H Stephens, and Jonathan P How. Driver behavior classification at intersections and validation on large naturalistic data set. *Intelligent Transportation Systems, IEEE Transactions on*, 13(2):724–736, 2012.

[4] Alexandre Armand, David Filliat, Javier Ibañez-Guzmán, et al. Modelling stop intersection approaches using gaussian processes. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems-ITSC*, 2013.

[5] Thomas Batz, Kym Watson, and Jiirgen Beyerer. Recognition of dangerous situations within a cooperative group of vehicles. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 907–912. IEEE, 2009.

[6] Enrico Bertolazzi and Marco Frego. Fast and accurate clothoid fitting. *arXiv preprint arXiv:1209.0910*, 2012.

[7] Hugh Durrant-Whyte and Thomas C Henderson. Multisensor data fusion. *Springer handbook of robotics*, pages 585–610, 2008.

[8] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. pages 139–147. Morgan Kaufmann, 1998.

[9] Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1625–1631. IEEE, 2010.

[10] Daniel Greene, Juan Liu, Jim Reich, Yukio Hirokawa, Akio Shinagawa, Hayuru Ito, and Tatsuo Mikami. An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):942–953, 2011.

[11] Till Hulnhagen, Ingo Dengler, Andreas Tamke, Thao Dang, and Gabi Breuel. Maneuver recognition using probabilistic finite-state machines and fuzzy logic. *IEEE Intelligent Vehicles Symposium*, 2010.

[12] Larissa Labakhua, Urbano Nunes, Rui Rodrigues, and Fátima S Leite. Smooth trajectory planning for fully automated passengers vehicles: spline and clothoid based methods and its simulation. In *Informatics in Control Automation and Robotics*, pages 169–182. Springer, 2008.

[13] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Exploiting map information for driver intention estimation at road intersections. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 583–588. IEEE, 2011.

[14] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Evaluating risk at road intersections by detecting conflicting intentions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4841–4846. IEEE, 2012.

[15] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Risk assessment at road intersections: Comparing intention and expectation. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 165–171. IEEE, 2012.

[16] Stéphanie Lefèvre, Christian Laugier, Javier Ibañez-Guzmán, et al. Intention-aware risk estimation for general traffic situations, and application to intersection safety. 2013.

[17] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal*, 1(1):1–14, 2014.

[18] Daniel Meyer-Delius, Christian Plagemann, Georg von Wichert, Wendelin Feiten, Gisbert Lawitzky, and Wolfram Burgard. *Data Analysis, Machine Learning and Applications*, chapter A Probabilistic Relational Model for Characterizing Situations in Dynamic Multi-Agent Systems, pages 269–276. Springer Berlin Heidelberg, 2008.

[19] Daniel Meyer-Delius, Christian Plagemann, and Wolfram Burgard. Probabilistic situation recognition for vehicular traffic scenarios. *IEEE International Conference on Robotics and Automation*, pages 459–464, 2009.

[20] Michaël Garcia Ortiz, Jannik Fritsch, Franz Kummert, and Alexander Gepperth. Behavior prediction at multiple time-scales in inner-city scenarios. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 1068–1073. IEEE, 2011.

[21] Rajesh Rajamani. *Vehicle dynamics and control*. Springer, 2011.

[22] Richard A. Retting, Helen B. Weinstein, and Mark G. Solomon. Analysis of motor-vehicle crashes at stop signs in four u.s. cities. *Journal of Safety Research*, 34:485–489, 2003.

[23] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[24] GM van der Molen. Trajectory generation for mobile robots with clothoids. In *Robotic Systems*, pages 399–406. Springer, 1992.

[25] Dizan Vasquez and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3931–3936. IEEE, 2004.

[26] Mintesnot Woldeamanuel. Stopping behavior of drivers at stop-controlled intersections: Compositional and contextual analysis. In *Journal of the Transportation Research Forum*, volume 51, pages 109–123. Citeseer, 2012.

[27] Guan Zhai, Huadong Meng, and Xiqin Wang. A constant speed changing rate and constant turn rate model for maneuvering target tracking. *Sensors*, 14(3):5239–5253, 2014.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
Prof. Dr. R. D'Andrea, Prof. Dr. L. Guzzella

Title of work:

Situation Understanding at Stop Intersections

Thesis type and date:

Master Thesis, October 2014

Supervision:

Prof. Dr. Francesco Borrelli
Prof. Dr. Lino Guzzella
Dr. Georg Schildbach
Dr. Stéphanie Lefèvre

Student:

Name: Thierry Wyder
E-mail: thwyder@ethz.ch
Legi-Nr.: 08-914-780
Semester:

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Declaration of Originality:

`http://www.ethz.ch/faculty/exams/plagiarism/confirmation_en.pdf`

Zurich, 14. 10. 2014: _____