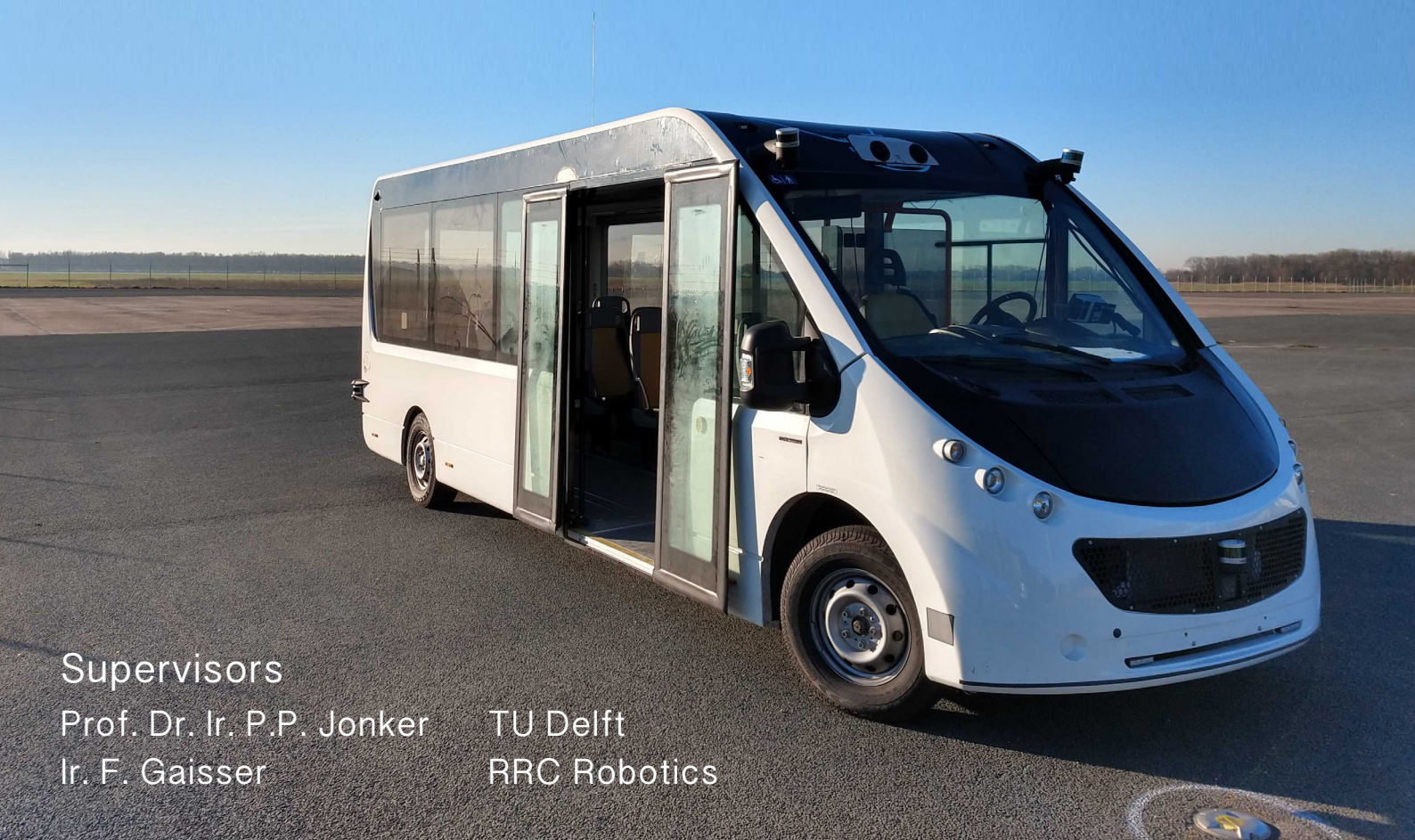
Vehicle motion prediction for autonomous driving

A deep learning model based on vehicle interaction and road geometry using a semantic map.

Tim Resink



Supervisors

Prof. Dr. Ir. P.P. Jonker TU Delft

Ir. F. Gaisser RRC Robotics

Vehicle motion prediction for autonomous driving

A deep learning model based on vehicle interaction and road geometry using a semantic map.

by

Tim Resink

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science at the Delft University of Technology

in

Mechanical Engineering, track BioMechanical Design

Department of BioMechanical Engineering, faculty 3ME,
TU Delft

Abstract

To be able to understand the dynamic driving environment, an autonomous vehicle needs to predict the motion of other traffic participants in the driving scene. Motion prediction can be done based on experience and recently observed series of past events, and entails reasoning about probable outcomes with these past observations. Aspects that influence driving behavior comprise many factors, such as general driving physics, infrastructure geometry, traffic rules, weather and so on.

Models that are used today are incapable of including many of these aspects. These deep learning models often rely heavily on the past driven trajectory as an information source for future prediction. Sparsely, some work has been done to include interaction between vehicles or some infrastructural features to the model. The lack of information regarding the driving scene can be seen as a missed opportunity, because it is a useful feature source to predict the motion of a vehicle. Especially when a map of the driving scene is available, reliable information regarding the road layout can be extracted.

In this thesis, a model architecture is sought that is aware of the interaction between vehicles, and that understands the geometry of the roads in the scene. Importantly, map information regarding the driving scene should be used as the primary source of information regarding the road geometry. First, a baseline deep learning model is constructed, that can generate predictions based on the past observed trajectory. To add interaction features to the model, a social pooling module is introduced. The social pooling method allows to efficiently include the driving behavior of other vehicles in the scene.

To introduce reliable, map-based road information to the model, two novel methods are proposed. In the first, the predictions from a model are used to extract features in a map. These features describe the road scene around the predicted location, and are used to update these predictions. In the second method, the semantic map is only used to extract a road segment ahead of the vehicle. A road-RNN is introduced to construct features regarding the road segment, and an attention mechanism to determine what part of the road segment is relevant for the predictions. These modules are referred to as road-refinement and road-attention respectively.

The importance of including both road geometry and interaction methods in the model is shown by constructing 5 different models that vary in their road-geometric and interaction awareness. A baseline deep learning model is used and extended with a road-geometry module, an interaction module, a combination of the two, or none. To test the prediction capabilities of the models, they are trained on two different datasets. The first dataset, called i80, consists of trajectory recordings from a straight highway with dense traffic. The other dataset is a curved version of the i80, called i80c, where the trajectories and road are transformed to introduce road-geometric variations in the data.

The prediction performances from the models clearly show the importance of both interaction-aware and road-geometry aware modeling. The road-refinement and road-attention models outperform the road-agnostic baseline model on the i80c data, showing better understanding of the road layout. Comparison between these models learns that the road-attention model is more effective for road-geometry inclusion. Additionally, the performance increase for interaction-aware models compared to the baseline clearly shows the importance of interaction in modeling on both datasets.The model that combines the interaction and road-attention modules shows outstanding prediction performance on the challenging i80c dataset compared to all other models. From the predictions it can be seen that the model understands the road layout ahead of the subject vehicle, as well as the interactive forces that are in play.

Contents

1 Introduction **1**

2 Related work **5**
2.1 Classical methods . . . 5
2.1.1 Physics-based models . . . 5
2.1.2 Maneuver recognition . . . 6
2.1.3 Graph models . . . 7
2.1.4 Deep learning-based prediction . . . 8
2.2 Interaction . . . 9
2.3 Road geometry . . . 12
2.4 Conclusion . . . 16

3 Methodology **19**
3.1 Sequence to sequence prediction . . . 19
3.1.1 Recurrent Neural Networks . . . 19
3.1.2 Encoder-decoder architecture . . . 21
3.1.3 Input embedding . . . 22
3.1.4 Normalization & Loss . . . 24
3.1.5 Summary . . . 25
3.2 Interaction . . . 25
3.2.1 Adjacent vehicles . . . 26
3.2.2 Social pooling . . . 27
3.3 Road geometry . . . 27
3.3.1 Semantic map . . . 28
3.3.2 Road refinement . . . 30
3.3.3 Road attention . . . 32
3.4 Conclusion . . . 35

4 Experiments & Results **37**
4.1 Experimental setup . . . 37
4.1.1 NGSIM . . . 38
4.1.2 Data inconsistencies . . . 40
4.1.3 Data augmentation . . . 41
4.1.4 Error metrics . . . 46
4.1.5 Training details . . . 46
4.2 Experiment 1: Baseline model . . . 47
4.3 Experiment 2: Interaction . . . 50
4.4 Experiment 3: road-refinement . . . 53
4.5 Experiment 4: road-attention . . . 55
4.6 Experiment 5: combined RNN . . . 57
4.7 Overview of results . . . 59

5 Discussion & Conclusion **63**
5.1 Discussion . . . 63
5.2 Conclusion . . . 71
5.3 Recommendations . . . 71

Bibliography **73**

Acknowledgements

This master thesis was conducted as a part of the Master Mechanical engineering, track BioMechanical Design, specialization BioRobotics at the faculty of 3ME, TU Delft. The study has been conducted in collaboration with RRC Robotics on the subject of autonomous driving in the project I-AT Interreg Automated Transport. This thesis has been completed with the support of some people to whom I would like to extend my gratitude.

First of all, I would like to thank RRC for facilitating this thesis, both in terms of presenting me this subject as well as all guaranteeing the involved support. By providing all the working facilities, actively involving me in their work flow, and offering their expertise whenever it was needed, an atmosphere was created which has been very positive both in terms of my personal development, as well as the thesis process.

My daily supervisor, Floris Gaisser, had a substantial part in this experience. Floris has all the academic and practical experience I could have wished for, aiding me tremendously both on the subject matter as well as the research process. Weekly meetings often turned into afternoons of discussions about subjects of interest, where Floris showed a genuine interest in my work and ideas, and contributed with valuable suggestions and positive stimulation. Floris, thank you sincerely for your council, interest, enthusiasm, and discipline with endless reviews.

I would like to express my appreciation to Prof. dr. ir Pieter P. Jonker, for providing this thesis possibility, and giving me valuable feedback on my work throughout the process. Furthermore I would like to thank the other members of my thesis committee, Julian Kooij and Dr. Javier Alonso Mora, for their effort to oversee this final stage of my education.

Finally, my parents are the most important reason that I got here. I truly believe that without their support this would not have been possible. My mother, who infects me with her drive and enthusiasm, who gives me a place to clear my head and stands ready when I need it. My dad, whose love and wisdom I always carry with me.

Tim Resink
Delft, February 2019

1 Introduction

At present, road accidents are one of the largest concerns of public health. According to the World Health Organization, approximately 1.3 million people die each year in traffic, claiming more victims than most diseases, and being the leading cause of death among young people aged 15-29 [1]. For this reason, the UN adopted the "2030 Agenda for Sustainable Development" in 2015, where a goal is set to reduce the amount of lethal accidents with 50% by 2030[3].

Safety-enhancing technology and measures have been developed over the years to make driving less dangerous. Seat belts for example have dramatically reduced the risk of death and serious injury. Seat belts reduces the risk of lethal consequences by 45% for drivers and front-seat passengers, and the risk of serious injury by 50% [2]. Furthermore, road safety has improved by reducing the number of accidents through infrastructural improvements for example [48]. Nevertheless, accidents are still a common occurrence, of which 90% is attributed to some critical human error [6, 41, 46].

Human errors that cause accidents can be attributed to several behavioral factors according to Petridou and Moustaki [37]. Factors that reduce driving capabilities on a short-term basis can be drowsiness, fatigue or acute alcohol intoxication. On a long-term basis factors such as aging, diseases or disabilities play a role. In order to reduce the number of human errors, regulatory policies have been introduced. Policies regarding driving under influence have decreased the number of alcohol-impaired fatalities with 65% between 1982 and 2016 [5]. However, not all factors can be counteracted through issuing and enforcing regulation or raising public awareness. Fully eradicating human error as the cause of driving fatalities can only be achieved by removing the human as the driving agent.

Both from industry and institutions there is an increased interest in autonomous driving, in order to ultimately eradicate fatalities in traffic caused by human error. Estimates indicate that autonomous vehicles could reduce accidents by 90% [4]. Companies from a range of industries are currently developing autonomous vehicles, among which some established non-automotive tech companies such as Google and Baidu, and emerging tech companies such as RRC, Argo and Aptiv. An abundant amount of academic institutes are also focusing on autonomous vehicles or specific challenges within this application domain. Such a global focus on autonomous vehicles greatly increases the development rate and will make them accessible to the public earlier in the future.

However, many challenges still have to be overcome to achieve new levels of automation. In the current situation, a limited amount of automation is present in vehicles. Advanced Driver Assistance Systems (ADAS) are widely used, aiming at assisting drivers by either simplifying a task for the human or taking it over completely. Technology such as the Night Vision System, Adaptive Front Lights and Surround View Cameras enhance the perception of the human driver, whereas Adaptive Cruise Control, Blind Spot Monitoring or Pedestrian Detection Systems replace some situation perception or action execution tasks. Parking assistance is a form of ADAS which fully takes over the task of perceiving the environment and controlling the vehicle accordingly. More recently the Tesla Autopilot was introduced which allows the vehicle to navigate autonomously on highways under human supervision.

Highway autopilots or Parking Assistance Systems are forms of partial automation, where a driver leaves specific tasks to the autonomous system, while monitoring the process and standing ready to intervene. In the Society of Automotive Engineers (SAE), 5 levels of automation are identified (figure 1.1). These levels range from assistance systems (ADAS) to fully autonomous driving capabilities without a human present to assist or intervene in any way[44]. Current technological automation capabilities are considered level 2 - partial automation, which requires attentive supervision by the driver. This level of automation poses new challenges due to the interaction level of human and machine. Human supervisory control and monitoring is associated with numerous hazards [26] such as failure to intervene when necessary, over-trust in computer tasks, loss of situational awareness and manual skill decay. It is therefore desirable to move further in the levels of autonomous driving.

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.1: The five levels of automation as defined by the SAE, adapted from [44]

Fully autonomous driving can only be achieved if the autonomous vehicle has a degree of understanding about the surrounding environment. The autonomous vehicle that perceives the environment is often referred to as the ego vehicle, being the center of the perceived traffic situation around it. To obtain a reliable perception of the environment, the ego vehicles relies on a number of different sensors, generally consisting of laser scanners (LIDAR), 3D cameras, regular cameras, radars and ultrasonic sonars. These sensors enable perception of the environment by combining complementary information, such as scene vision and radar distances, which is crucial for localization, object detection and mapping [47]. Moreover, some autonomous vehicles such as the WEpods have a map available, with information regarding the static driving scene that is gathered offline. By means of integration of sensory and map information, an understanding of the traffic situation can be formed.

Understanding the dynamic environment is a complex challenge, and is being actively researched. To be able to understand the dynamic environment, an estimation of the driving behavior of other traffic participants has to be obtained. Estimating driving behavior can be achieved by anticipating the intentions that surrounding traffic participant have, and the motion that follows from this intention. A motion prediction module computes the expected trajectory for each traffic participant over a desired time in the future, called the prediction horizon. With the predicted trajectories, an assessment can be made whether the planned path conflicts with other trajectories, or otherwise unwanted situations will arise.

Motion prediction can be done based on experience and recently observed series of past events, and entails reasoning about probable outcomes with these past observations. Aspects that influence driving behavior comprise many factors, such as general driving physics, infrastructure geometry, traffic rules, weather and so on. This also includes the dynamic interaction between traffic participants, which is particularly important in crowded areas. In an ideal situation, all of these factors are taken into account when predicting the trajectories of vehicles in the scene.

In reality, modeling approaches that are used today are incapable of including many of the before mentioned aspects. These models often rely heavily on the past driven trajectory as an information source for future prediction. Sparsely, some work has been done to include interaction or some infrastructural features to the model. The lack of information regarding the driving scene can be seen as a missed opportunity, because it is a useful feature source to predict the motion. For example, the shape of the road ahead of a vehicle gives a considerable amount of information on the future driving path of that vehicle. Moreover, when a map of the driving scene is available, reliable information regarding such a road segment can be extracted.

The aim of this thesis is to find a model architecture that can use information regarding multiple aspects for its prediction. More specifically, a model architecture is sought that is aware of the interaction between vehicles, and understands the geometry of the roads in the scene. Importantly, map information regarding the driving scene should be used as the primary source of information regarding the road geometry. After the design, an assessment will be made on the effectiveness of the modeling approaches and the inclusion of the interaction and road-geometric aspects in general. By finding an appropriate method to include these aspects in the prediction, the motion of vehicles in the scene can be more accurately predicted. Such a model brings motion prediction one step closer to practical use for autonomous vehicles.

The structure of this thesis is as follows: In Chapter 2, related work regarding motion prediction will be detailed. More specifically, related work that use the past observed trajectories, interaction between vehicles, or road geometric aspects will be discussed. In Chapter 3, the construction of the models that include a part or all of these aspects will be detailed. In Chapter 4, the experiments that are done to validate the approaches and assess their importance are presented. In Chapter 5, the results from the experiments will be discussed.

2

Related work

In chapter 1, an introduction is given to the importance and goal of motion prediction in autonomous driving. The goal is to predict a future trajectory of a vehicle, given observations of the past driven trajectory and its driving environment. This chapter will detail relevant methods from the field, that represent the current capabilities of motion prediction methods.

The past driven trajectory is often considered the main source of information to describe the future driving behavior. Common approaches that use this past driven trajectory will be detailed in the first part. A distinction will be made between well known classical methods, and more recent machine learning approaches. Solely using the past driven trajectory to predict the future is insufficient. More sophisticated predictions can be obtained when the driving environment is taken into account. The driving environment consists of dynamic and static aspects. The dynamic environment is formed by other vehicles, pedestrians or cyclists. The static environment consists of the infrastructure, traffic rules and obstacles that are present. Taking both the interaction and road geometry into account is challenging, and proposed methods that do so will be described in more detail in section 2.2 and 2.3.

2.1. Classical methods

Motion prediction is attempted ever since the interest in autonomous vehicles started. Predicting the motion of vehicles is a key aspect of autonomous driving, and many approaches have been proposed over the years. In the classical methods, the driving behavior of vehicles was created by manually defining the relations between features and the future driving behavior. In this section, the most common classical approaches are described.

2.1.1. Physics-based models

The process of driving in traffic can at the most fundamental level be seen as a physical process. A model of this physical process can predict the states of a vehicle over time. The first attempts at motion prediction are based on such a physical modeling. The more accurate the model describes the process, the more accurate predictions can be made over time. The models describe the relation between the states of the vehicle at the current time s_t and the next time step s_{t+1}. Simulations of the driving behavior into the future can thus be obtained by iterating the predictions over the model. These models are known as physics-based models or evolution models.

In vehicle motion prediction, simple models are often preferred. When observing vehicles from the ego vehicle's perspective, only a limited number of variables of the subject vehicle can be observed. To this end, physics-based models often are a strong simplification of the vehicle driving process. The physical processes that are modeled consist of dominant dynamic factors, and can be modeled with observable variables of the subject vehicle. Kinematic models are one such category of models that are used often. Rather than modeling the dynamics, the kinematic relations between states such as position, velocity and heading angle over time are used. An important kinematic model is the bicycle model [14, 23, 32], which takes into account the non-holonomic constraints that apply to vehicles. A variety of similar models are described in [31], such as

the Constant Steering Angle and Acceleration (CSAA) model. In these models, the states such as the velocity and position at a next time step can be obtained through the kinematic relations. These states can be iterated over the model to obtain a multiple time-step prediction.

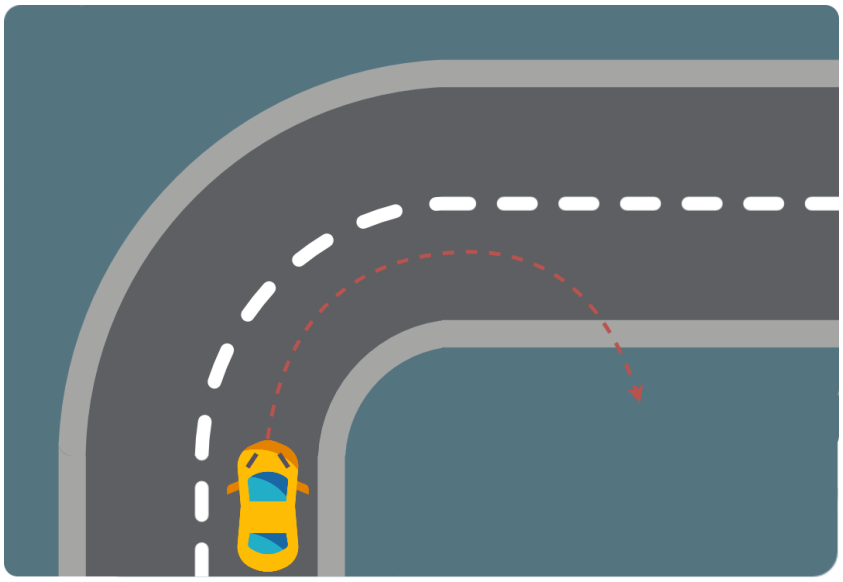


Figure 2.1: An example prediction of a physics-based model assuming a constant steering angle.

Physics-based models can predict the future only accurately over a small prediction horizon. They are unable to predict variables that are directly related to human input, such as acceleration (throttle) or steering angle. By lack of a model, these variables are assumed to take the constant value of the last observation over the entire trajectory, see figure 2.1. Alternatively, multiple models with different constant variables can be combined in an Interactive Multiple Model (IMM) filter [8]. The IMM is a much used modeling technique [15, 24, 25], where a weighted combination of predictions from various physics-based models is used. Other methods such as Switching Kalman Filters [9, 34] or heuristic methods [50] follow a similar approach. These methods have been popular and as of today often are still used to benchmark performances of state-of-the-art approaches. By combining multiple models, more complex driving behavior can be described as a combination of subtrajectories. However, methods to determine when and why the models should be switched for prediction is often lacking.

2.1.2. Maneuver recognition

To obtain better long-term prediction, methods were explored where a prediction is generated from a higher abstraction level. Maneuver recognition approaches try to find an appropriate trajectory with a recognized maneuver, rather than generating a sequence of states based on their previous states. The intuition behind maneuver recognition approaches is that all driving behavior that is observed can be categorized in a set of maneuver classes. If a maneuver is recognized, a maneuver-specific model can be deployed to get a prediction of the driving behavior. In this way, a part of the human influence of driving behavior can be taken into account by incorporating the maneuvers that are known to drivers in the motion prediction approach.

Recognition is done by classifying an observed trajectory as a maneuver, often by looking at a set of handcrafted features or comparing it to class-specific prototypical trajectories. In the latter case is often focused on a sophisticated feature that can accurately resemble the important aspects of a maneuver, making it straightforward to compare observed trajectories with prototypes, such as in [20].

Once the maneuver is recognized, there are two approaches to generate a prediction. The first simply takes the most resembling prototypical trajectory of the recognized maneuver class. The level of similarity between

the observed driving behavior and the prototype can also be used to construct a prediction as a combination of multiple prototypes. The second approach is to use maneuver-specific models per maneuver class to generate a new trajectory. A set of physics-based models can be used with different values for the human-input related variables [22]. The maneuver recognition approach thus combines prior knowledge, e.g. distinguishing between known maneuvers, with an online prediction that is tailored to the specific class.

2.1.3. Graph models

Many approaches are based on the combination of maneuver recognition and subsequent trajectory simulation with a set of physics-based models. Over time, more sophisticated methods are proposed, that better deal with the many varieties in driving behavior. For example, in maneuver recognition one can construct elaborate sets of logic rules that dictate which maneuver model should be deployed [16]. Alternatively, the uncertainty of classifications and predictions can be taken into account with stochastic models. Such methods attempt to better handle the large set of possible future driving behavior that exists.

Stochastic models are useful to indicate and process the level of uncertainty in the predictions. Probabilistic classification methods often rely on graph models. Graph models such as Dynamic Bayesian Networks can use heuristic features to obtain a probabilistic estimate of the maneuver class that a maneuver belongs to. Additionally, these heuristic features offer the possibility to use environment information to estimate the maneuver. For a feature f and a class c, the probability that a maneuver belongs to a class c is obtained with bayesian inference by

$$P(c) = P(c|f)P(f). \tag{2.1}$$

In the DBN, the conditional probability between all features and classes $P(c_i|f_j)$ is defined, indicating the relation between features and maneuvers. These conditional probabilities can be learned and updated online, through a machine learning technique called Expectation-Maximization. In such approach a trajectory is first assigned to a class, and then all class definitions are updated according to the new distribution including this new trajectory. This means that the model learns throughout the process of classifying the trajectories, and improves with the amount of classifications it does.

Such work is done in [29], where a DBN is used to classify the driving intention that a driver has at an intersection, based on a heuristic feature. Due to the separation of classification, and class-specific prediction works can focus on either aspect of the problem. DBNs can also be used to incorporate both the classification and the prediction step [42]. Many works have focused on a similar approach, where a variety of features describing the past driven trajectory and the environment are used to classify the maneuver and generate a prediction.

Methods based on maneuver classification in combination with predictions from physics-based models or stochastic prototypical trajectories have made motion prediction approaches more accurate and versatile. The classification can assess the driving behavior from a semantic level, where desired features can be incorporated in the reasoning about the high level driving behavior. The physics-based models can then be deployed to provide the resulting predictions.

However, these methods are limited in their performance. When classifying driving behavior from a high level, a balance has to be found between the level of detail of individual classes, and the amount of classes that are used to describe all possible driving behavior. Over-generalization leads to poor definitions of maneuvers, and reduces the effectiveness of maneuver-specific models. On the other hand, giving too detailed distinctions between maneuvers requires an extensive set of classes. This makes it challenging to clearly define subtle differences, yet cover the entire set of possible driving behavior that can be encountered, as acknowledged in [35]. Furthermore, maneuvers are difficult to mathematically describe. This makes it even more difficult to generate trajectories that accurately simulate the driving behavior with respect to a found maneuver. As of today, long-term prediction performance of such models is limited.

2.1.4. Deep learning-based prediction

An alternative approach to motion prediction is to create models that look for relations in the data. Deep learning represents a family of methods where a parametrized model learns to model a process or find patterns in data through supervised learning, thus based on example input-output pairs. Deep Neural Networks (DNN) are such models that are becoming increasingly dominant in several fields, such as computer vision, speech recognition and natural language processing. In recent years, these methods have made their entrance in motion prediction literature.

Deep learning-based approaches do not require the designer to define the relations between features and driving behavior. Instead, these models try to find the relations in the data that can explain the driving behavior, by looking at example data. There is no need to create truthful physics-based models or graph models where the relations between states and future driving behavior have to be well understood and defined manually. Rather, given a sufficient amount and diversity of training data for the desired task, these relations can be learned by the model. The amount of training data is determined by the complexity of the model, where complex tasks that need larger models need more data to train the model.

The type of data and complexity of the task dictate what model architecture will be most effective, rather than the physical meaning of the data. For this reason, new models and techniques often arise in different academic fields that can prove to be useful for motion prediction if the data has a similar structure. In motion prediction, mostly sequence analysis tools are used, that analyze a past driven trajectory to predict the future driving behavior. A trajectory T is then regarded as any other time-series, a sequence of states over time with a set of features that describe the state of the vehicle at each time step.

Recurrent Neural Networks (RNN) are DNNs that have been designed for sequential data, such as sentences or time series. Modern RNNs, such as the Long-Short Term Memory (LSTM) [21], can handle long sequences relatively well by analyzing a trajectory step by step and building up an internal understanding in some memory parameters. At each time step, the RNN receives a state of the sequence x_t and its own memory parameters h_{t-1} from the previous time step. This is represented in figure 2.2. At the end of the trajectory, an internal representation of the observable trajectory is constructed in the memory, called the cell state. The entire sequence is then encoded in a number of cell state parameters. With this internal memory representation of the trajectory, a classification or trajectory prediction can be obtained. Patterns are recognized in the data by embedding the trajectory over time into the cell state and analyzing the cell state. The working of an RNN are explained in more detail in chapter 3.1.

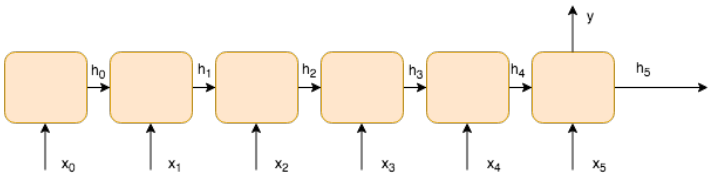


Figure 2.2: An RNN unrolled over time.

The RNN as depicted in figure 2.2 can generate some output at the last time step t_0. Such a task is called a many-to-one mapping, where multiple past sequence steps are analyzed to obtain an output at a single (time) step. However, for trajectory prediction, a new set of states is desired at N time steps into the future. To obtain a future sequence by analyzing a past sequence, the sequence to sequence models have been proposed [45]. In such a many-to-many model there are two RNNs being used, called the encoder and the decoder 2.3. The encoder is responsible for the analysis of the observable data, similar to the model from figure 2.2. The distinction is that rather than using the output at t_0, the cell state is used and passed on to a different RNN, called the decoder. This decoder uses the cell state of the encoder to unroll predictions into the future. The encoder thus encodes the observable trajectory, that can be of variable length, into a cell state of a fixed number of parameters. This cell state is then given to a decoder RNN, that has to 'decode' the cell state into a future trajectory over any desired number of future time steps.

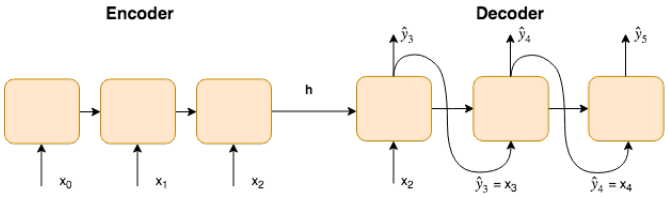


Figure 2.3: Sequence to sequence modeling with an encoder-decoder architecture.

Sequence-to-sequence prediction is used in various works [27, 35], where an LSTM analyses a sequence of observed states. In [35], data recordings from an ego vehicle is used regarding surrounding vehicles on a highway. The state of a subject vehicle n at time step t is given by

$$S_t^{(n)} = \{v_t, \dot{\psi}_t, x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)}\}, \tag{2.2}$$

where $v_t, \dot{\psi}_t$ are the ego vehicle's velocity and yaw rate, and $x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)}$ are the subject vehicle's cartesian position and velocities. The decoder can, based on the analyses of the LSTM encoder, make an estimate of the vehicle occupations in a grid. In [13], a similar encoder-decoder architecture is used to model the driving behavior. In this work, both maneuver recognition and trajectory prediction is attempted, with an a priori defined set of maneuver classes. The maneuver recognition and prediction tasks are both obtained from the same model. First, the encoder analyses a past trajectory, to build a representation of the trajectory in memory. The encoder is also asked to produce a classification of the maneuver at the last observed time step t_0. The cell state is then used for the decoder to generate maneuver-specific predictions. In this way, the separation between maneuver recognition and prediction can be maintained. Regardless of the task, the model is expected to understand the driving behavior both at the state level, and at the higher semantic level.

Prediction of the driving behavior based on the past observed trajectory can be obtained in various ways. A number of approaches are available to model driving behavior, where both the features that represent the data and the model of choice are important. In recent years, a trend emerged where the field moves away from models where the relations between features and driving behavior are explicitly defined, and move more towards approaches where these relations are learned from data. RNNs have shown to be capable of fairly accurate predictions, based solely on the past observed trajectories. As acknowledged in [35], it is assumed that RNNs can deduce the influence of latent factors such as driver style or lane-following without having explicit indications of the lane or preferred driver style. This capability is effective to a certain extend, but does not take away the need to incorporate such information in the predictions. Therefore, the modeling and indicating of the most important factors will be indicated in the next sections.

2.2. Interaction

To make a sophisticated guess of the future driving behavior of a vehicle, all factors that have a dominant influence should be regarded. One of the most dominant influences on driving behavior is the interaction among traffic participants. Particularly driving behavior in dense traffic situations such as peak hours highway traffic or urban scenarios such as crossroads, is dictated by interaction. Trajectories that are chosen in such situations can significantly differ from the preferred trajectory that would have been chosen in case no other participants are present.

To model interaction between traffic participants, a distinction has to be made between unilateral and bilateral interaction. Unilateral interaction occurs when one driver adapts its behavior to another, for example when a vehicle slows down to avoid collision with a truck in front of it. Bilateral interaction on the other hand occurs when multiple vehicles interact to jointly solve a traffic situation. For example, on unsignalized intersections are no traffic rules to explicitly define who has priority. Vehicles have to yield to one another by anticipating the future situation. This situations often demands multiple vehicles to interact with each other, making it bilateral.

The distinction between unilateral and bilateral interaction has major implications on the modeling of driving behavior. In unilateral interaction, the subject vehicle's driving behavior can be predicted by assuming that the surrounding vehicles are dynamic 'objects', that do not change behavior by the actions of the subject

vehicle. In bilateral interaction, this assumption does not hold. Predicting motion with bilateral interaction then becomes a joint optimization task, where the driving behaviors are coupled. In such a scenario, the possible set of trajectories grows exponentially with the number of traffic participants that are interacting. When considering all action combinations of the interacting traffic participants, a large solution space has to be searched. The assumptions on the type of interaction therefore have a large influence on the presented modeling challenge.

Modeling of bilateral interaction was attempted by coupling graph models that each predict the driving behavior of a single vehicle [11]. In such an approach, the probabilistic state transition of both graphs are dependent on the other. In graph models, section 2.1.3, the relation between features and future states is defined by a conditional probability: $P(s_{t+1}|s_t, f_t)$, where f_t are the features and s_t, s_{t+1} are the states at a specific time step. In a coupled graph, the feature vector f_t includes the state of the adjacent vehicle s_{at}, and vice versa. Two graph models thus contain variables that depend on each other, connecting the models.

The downside to this modeling approach is twofold. Firstly, the computational complexity increases significantly with each extra vehicle in the scene, because graph models have to be coupled. Secondly, due to the coupled driving behavior, the uncertainty of the predicted driving behavior of one vehicle also propagates to the predicted driving behavior of another vehicle. In this way, the uncertainty of the predictions can grow unacceptably large because the future predictions of other traffic participants are quite uncertain. This is known as the freezing-robot problem, where the robot or autonomous vehicle has to come to a halt due to the inability to estimate behavior of other agents [43]. These drawbacks render this approach unsuitable for real-time motion prediction.

To address this problem and handle the computational complexity of interaction modeling, assumptions are made. One such assumption is that the interaction is always unilateral. In situations such as depicted in 2.4 this can be a valid assumption, assuming only the blue vehicle adapts its behavior to the yellow one. Such assumption significantly reduces the possible set of solutions[31]. For unilateral interaction, asymmetric graph models can be used, such that only the transition probabilities from the past states of vehicle A to the current state of vehicle B have to be learned. These past states have been observed and are fixed, and thus introduce no prior uncertainty. The joint optimization process can be circumvented in this way, obtaining a more computationally efficient approach.

An alternative assumption that can be made is that only a limited amount of vehicles can be interacting. In [17], it is assumed that the interaction with a maximum of two vehicles is important to model, to reduce the solution space. In this work, rather than explicitly modeling the transition probabilities between vehicle states, relative states such as heading difference and time-to-collision are used. Here, lane sections are determined which traffic participants are planning to drive on. These lane sections are used to determine whether there is a possible interaction, computing only the important possible interactions. By limiting the maximum amount of vehicles and first looking at route intention to determine which vehicles are important to take into account, the solution space becomes much smaller. This allows to model bilateral interaction with the vehicles that is most intensely interacted with.

As stated before, deep neural networks can be convenient to learn relations from data that are difficult to explicitly define in models. In the case of interaction-aware motion prediction, the DNN is assumed to learn the interactive relations between vehicles in the scene, without making any assumptions regarding the type of interaction. Specifically in the field of pedestrian motion prediction, a great focus has been put on interaction with DNNs. Pedestrians have less movement constraints than vehicles on the road, and their motion consists of more complex and abrupt movements. In crowded areas, such abrupt movements are often caused by the interaction between pedestrians. These factors make pedestrian motion prediction extra challenging to model.

Pedestrian interaction modeling with deep learning is achieved with RNN trajectory analysis that is described in 2.1.4. The cell state of an RNN describes the past observed trajectory in a fixed amount of latent parameters, and can be used as a feature vectors. For each pedestrian in the scene such a feature vector can be obtained, representing all past movements that occurred. In [7], all these pedestrian cell states are used, to

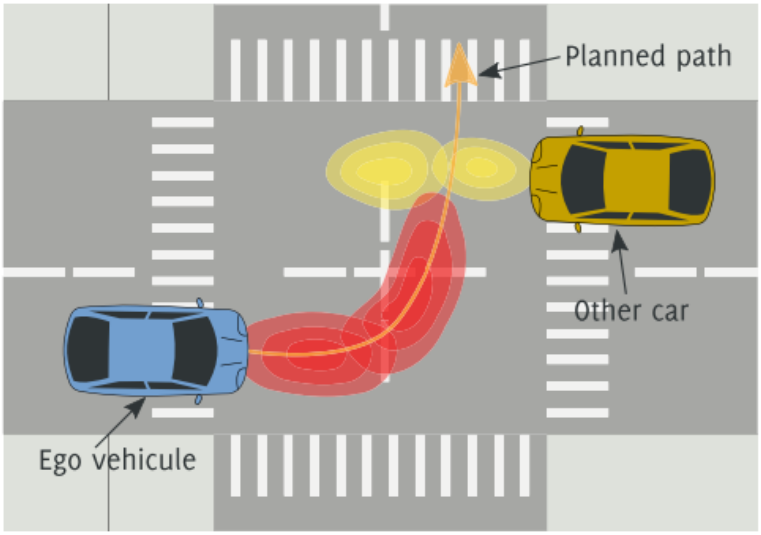


Figure 2.4: Vehicle interaction at an intersection. Adapted from [31]

take the interaction between them into account. These cell states are all organized in a grid map that represents the scene, where they are placed according to the spatial configuration among the pedestrians (see figure 2.5). Such a grid can be flattened to a 1D array, where each vector entry represents (part of) a grid cell. This vector contains information of all the pedestrians in the scene, organized to their spatial configuration, and can be seen as a interaction feature vector. Such a feature vector can be used by an RNN decoder to take the interactive forces into account for prediction.

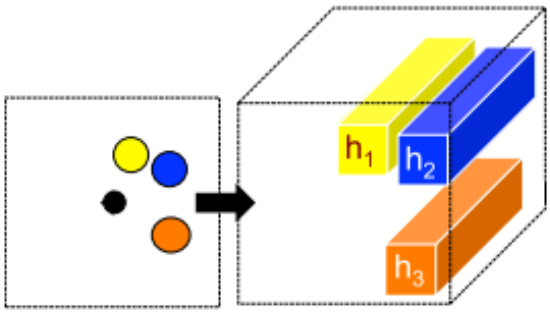


Figure 2.5: A 3D Tensor containing n cell states h_n of the pedestrian's motions. Adapted from [7]

One requirement of such an interaction feature vector is that it should always be of fixed size, whereas the number of neighbouring pedestrians may vary. To this end, multiple cell states that are assigned to the same grid cell are pooled into one new state (figure 2.6). By pooling the cell states of the individual pedestrians in a single cell, only the dominant effects of the pedestrians remain. The grid cells can be empty if no pedestrian is present, be filled with a single cell state, or with a pooled combination of multiple cell states. Usually, the resulting feature vector is first passed through an embedding layer before feeding it into the RNN decoder. The resulting feature vector can be created at each prediction time step of the decoder, to generate a future trajectory that is aware of the interactive forces in play. This technique is known as Social Pooling (SP), referring to the socially-driven interaction of pedestrians.

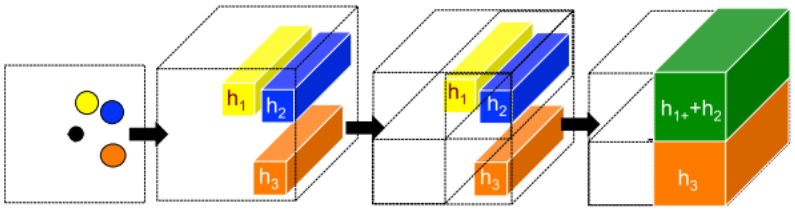


Figure 2.6: Pooling a 3D tensor, where multiple cell states are contained in a single grid cell. Adapted from [7]

Social pooling has been adopted in other work, such as pedestrian prediction in a museum [10]. It is shown that the pooling method also offers a feature entry point into the network for static objects, in this case the location and type of art in the museum. In [30], a social-pooling based method is first introduced for vehicle motion prediction. In an encoder-decoder setup, the interaction feature vector is reconstructed at every decoder time step and fed back to the decoder, to obtain more accurate interaction-aware predictions.

Recently, more effective methods that are inspired by social pooling have been proposed. In [13], a method called Convolutional Social Pooling (CSP) is introduced, for vehicle motion prediction. In the SP method from [7], the grid map is transformed to a 1D feature vector and passed through an embedding layer. In CSP, the conversion of the grid map to a interaction feature vector is done by a Convolutional Neural Network. It is argued that by flattening the grid map and embedding it with a fully connected layer, much of the spatial information is discarded. By using convolution networks, the grid map can be directly analyzed and the global interaction configuration can be better understood by the model. This work has been used for highway trajectory prediction with dense traffic, a situation where interaction is a dominant factor in the driving behavior.

Although a 3D grid representation is an intuitive representational form for spatial configurations of vehicles, it is computationally expensive. For the entire scenery, all occupied and unoccupied spaces have to be indicated and analyzed by the network. This becomes specifically redundant when interaction between vehicles that are further apart should be taken into account, such as at large intersections for example. Rather than addressing the limitations of the embedding step, the limitations of the grid representations are addressed in [18]. Here, the cell states are not placed in a grid, but instead their relative positions are embedded into the cell state. In this fashion, any vehicle can be taken into account for the interaction, regardless of the distance between the subject vehicle and the interacting vehicle.

An additional proposed improvement is using the obtained feature vector to initialize the decoder cell state, rather than using it as an input feature vector. This means that, by embedding the pooled features into a vector of the cell state size, an RNN decoder can be initialized with this cell state. This removes the need to re-obtain a feature vector at every decoding step, without conceding in prediction accuracy. Due to this approach, a 16x speedup is reported over the SP method from [7].

Social pooling methods have evolved into a beneficial deep learning approach to model driving behavior with the interactive factors taken into account. It eliminate the need to explicitly model the interaction, such as in the coupled graph models. In classical methods, predicting the vehicles' uncertain future trajectories introduced a complex optimization task, with some clear downsides. In social pooling, the detailed analyses of the past trajectories of the RNN encoder are utilized to determine the dominant aspects of the adjacent vehicles on the subject vehicle's driving behavior. More adjacent vehicles can be taken into account, as the dominant factors are filtered out by the pooling operation, regardless of the amount of vehicles. This technique is relatively new, but shows great promise for future interaction-aware motion prediction.

2.3. Road geometry

To obtain accurate predictions of the future trajectory, information on the past driven trajectory and interacting agents is often not sufficient. Additional information needs to be taken into account regarding the

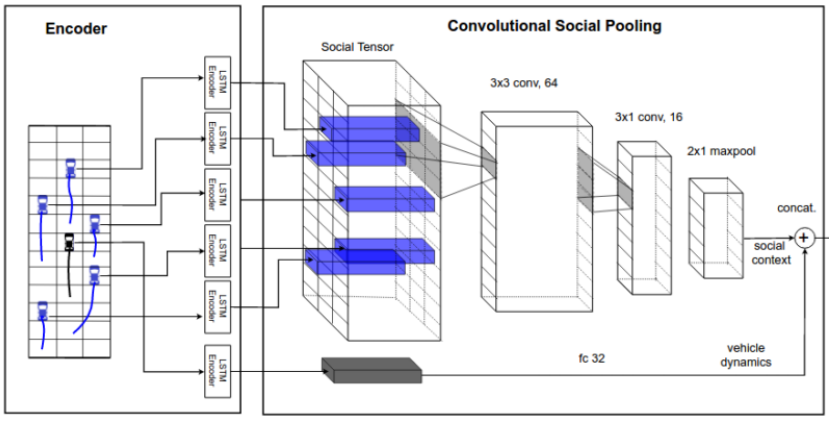


Figure 2.7: Convolutional social pooling over a hidden state tensor. Adapted from [13]

driving environment, because driving behavior can vary significantly depending on the location at which is being driven. Arguably the most dominant factors of the driving environment is the road geometry. The shape of the road and infrastructure determines where a driver can go, and acts as a constraint on the possible future locations of the vehicle. This requires the human driver to adapt to it, and change the driving behavior in order to safely navigate through traffic.

Therefore, it is advantageous to take information regarding the road geometry into account. It depends on the application and resources what information regarding the infrastructure is available. Often, only sensory information on the ego vehicle is present, such as camera images or LIDAR point clouds. In other applications, such as the wepods case, a priori information is present in the form of a semantic map. A semantic map is a map where locations on a map are labeled with semantics or features, that describe the environment. Such features can be regarding the road shape, traffic rules, expected traffic density etc. A semantic map can thus be used to extract reliable features regarding the environment, based on the ego vehicle's location. In this section, several information sources will be regarded, with a focus on semantic map-based motion prediction.

One way to represent the road geometry in a semantic map is by describing the centerline or boundaries of a lane as a sequence of points. These so-called waypoints can be defined in a global coordinate system, where the driving direction is indicated by defining the predecessor and successor of each waypoint (figure 2.8). For two successive waypoints, the road segment in between them can be described by a mathematical curve, such as a cubic hermite curve. A cubic hermite curve is a third-degree polynomial that can be specified in Hermite form; by its constraints on the values and first derivatives at the end points of the corresponding curve [49]. Hermite curves are particularly suitable if the end points of the curve have to define the shape. This is the case with with waypoints, where the curve describes the road segment in between them. The resulting curve is a polynomial

$$p(x) = \sum_{i=0}^{n} c_i^i, \tag{2.3}$$

parametrized with 4 coefficients that describe the curve that best satisfies the geometric constraints. In this way, the waypoints and accessory curves can describe a lane centerline or boundary in detail. Because the waypoints are defined in a global coordinate system, environment features can be extracted at a vehicle's global position. In [36], this information is used to obtain the projection of the vehicle position on the centerline of the most proximate lane, called the Active Lane Point (ALP). For a given vehicle position, the closest waypoint and its successor is found. By finding the closest point to the vehicle on the curve between the waypoints, the ALP is obtained. The distance between the ALP and vehicle is the orthogonal distance from the vehicle to the center of the lane. This semantic is used as a feature to classify lane-changing behavior. By determining the lane on which is driven, and the rate in lane change, appropriate physics-based models are used to predict the future trajectory. In this manner, the semantic map is used to extract the distance to lane, and classify maneuvers for motion prediction.

In [28], the waypoints are used to obtain context-specific features at intersections. The aim is to obtain a set

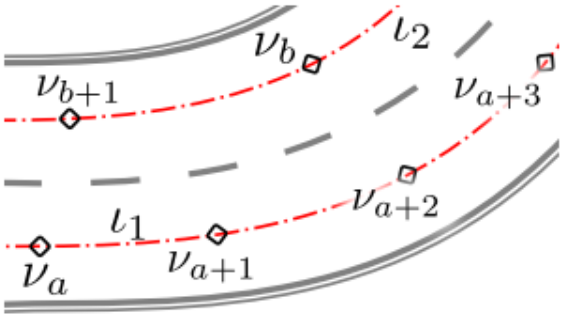


Figure 2.8: Sequential waypoints forming centerlines in two opposite directions. Adapted from [36]

of features that describes the important distinctions between the routes that can be taken on an intersection. The different routes at intersections can result in distinct velocity profiles, and obtaining accurate predictions for each is route is the goal. The waypoints on the driven lane are used to obtain route-specific features like *distance to the intersection entrance* and *distance to the intersection exit*. Additionally, the angle between the current centerline and the route centerline at the intersection entrance is determined. The distance features allow the model to find a give a velocity prediction, based on velocity profiles that are common for a certain route. Models can thus learn patterns such as vehicles decelerating at the start of a turn, and accelerating at the exit. The crossing angle is an indication of the curvature of the route, and is reported as a particularly important feature for prediction performance. These features are used for several models that classify either prototypical velocity profiles or longitudinal traveling distance and velocity directly. With the use of such semantic features, important aspects of the context ahead of the subject vehicle can be taken into account in the prediction.

Rather than appending the state of the vehicle with features regarding the context, they can be embedded in them. By describing the vehicle coordinates with the lane centerline as the reference frame, the lane geometry can be incorporated in the state of the vehicle. If a road segment is described by a single curve, the vehicle states can be expressed with respect to this curve. Given a vehicle's state consisting of its position in global coordinates, the ALP on the curve is obtained. The vehicle's position can then be indicated as the arclength of the curve from its origin to the ALP, and the orthogonal distance from the ALP to the vehicle. These are indicated as the longitudinal coordinate S, and lateral coordinate N, and the system is called a Curvilinear Coordinate System (CCS). The advantage of states expressed in CCS is that it relieves the prediction model from having to look at the road shape. Rather, all movement in longitudinal direction indicate a vehicle driving along the road, and all lateral movements indicate deviations from the centerline, such as a lane change. It thereby makes all lane-following driving behavior look approximately linear.

The CCS is used in [25], where the centerline of an entire road segment is described with a B spline, instead of a sequence of waypoints. Because of the apparent linearity of the vehicle motion, physics-based models are used for prediction that describe lane-keeping and lane-changing maneuvers, without regarding any rotational movement. A downside of the CCS is that no information is present regarding the road shape, and aspects such as the road curvature can not be taken into account when generating the future trajectory. This problem has been addressed by [39]. In this work, the CCS coordinates are used in a RNN encoder-decoder model, with the road curvature as extra state feature. By introducing the curvature, the model can still observe the influence of the road curvature on the velocity profile of the trajectories, while still being able to use curvilinear coordinates for the trajectory prediction.

There are two downsides using the local CCS. First, by looking up the curvature of the road with the predictions of the model, it fails to capture the global shape of the road ahead. Important information regarding roads with a curve, such as at the intersection prediction in [28], is not present in the model's reasoning mechanism. Another downside of CCS is that by defining the vehicle in a local coordinate system, information regarding the global traffic configuration with other vehicles is lost. For example, when another vehicle is present on a different road segment, both vehicle's states will be expressed in different coordinate systems.

Models using the CCS then will not be able to comprehend the spatial configuration between these two vehicles. For this reason, a CCS-based model can not be combined with the social-pooling based methods described in section 2.2.

Rather than extracting location-specific features manually, models can be presented with an entire section of the scene and extract the relevant information. In [30], front-view images of the driving scene are used in a RNN encoder-decoder model. The input data consists of the past observed trajectory, and an image of the subject vehicle and its surrounding. At first, a trajectory prediction is generated solely based on the past observed trajectory. These predictions are used to obtain location-specific features from the environment. A Convolutional Neural Network (CNN) analyzes the image to detect the relevant road segment in the scene, acting as a road geometry feature extractor. To obtain location-specific environment features, a feature pooling operation is used with the predicted position of the vehicle and the extracted features. In this way, the model finds environment features in the image, based on its own predictions. These features can then be used to update the predictions and tailor them to the road geometry of the scene.

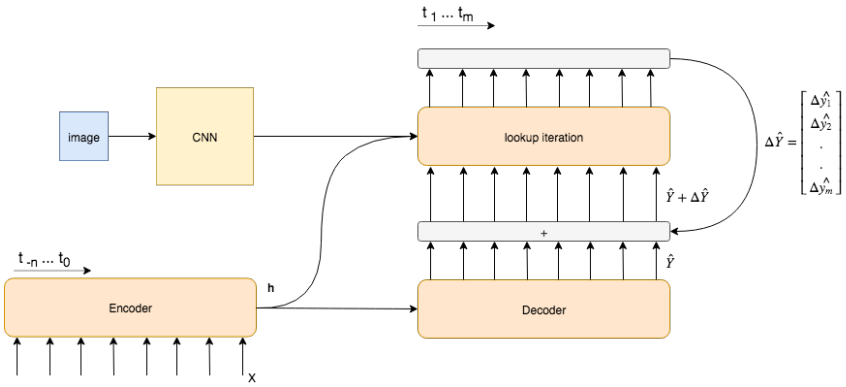


Figure 2.9: Iterative refinement of predictions $Y_{t_{1\,..\,m}}$, based on input trajectory $X_{t_{-n\,..\,0}}$ and environment images

However, this poses a chicken-and-egg problem. The features are obtained based on the predictions, and the predictions are updated based on the features. To deal with this, an iterative scheme is proposed. An additional decoder is deployed to update the predictions based on the current predictions and the obtained environment features. This second decoder takes in the original prediction and the pooled image features, and generates a regression vector ΔY. The prediction is then updated with this regression vector: $Y_{k+1} = Y_k + \Delta Y_k$. This iteration process can be repeated k number of times, until the prediction converged sufficiently. In their work, 4 iteration cycles are reported as sufficient to obtain accurate predictions. In this approach, the manual extraction of the environment features is replaced with a model that can autonomously look up its features from a visual information source with any prediction.

An alternative to this iteration method is by letting the model determine which part of the environment is important, rather than looking up the features around a predicted coordinate. This feature extraction automation step is taken in [40] to make the model road-geometry aware. In this work, top view images of the road network on which is being driven are analyzed with a CNN. A novel deep learning technique called an attention mechanism is used that allows the model to learn what parts of the image are important for the prediction. Therefore, this operation is a substitution of the location-based feature pooling from [30],removing the need for the iterative scheme. Two attention mechanism modules are used to look at parts of the road structure in detail. One module analyzes the road in front of the vehicle for short-term prediction. The second module analyzes the road network as a whole, having its attention scattered across the image. Based on the top view images, the model can thus assess the entire scene configuration and accurately analyze the road segment ahead of the subject vehicle.

The two approaches from [40] and [30] are promising methods to communicate the road geometry to a model. The problem with these approaches lies in the source of information, rather than the modeling approaches that are taken. The use of front view images is suboptimal, because parts of the road can be occluded or misunderstood, especially for longer prediction horizons. This is acknowledged in [30], where exits

on roundabouts and other occluded road segments are sometimes missed. In this regard, top view images are an improved source of information regarding the road geometry. However, such an information source is generally not available to autonomous vehicles and is thus of limited practical use.

With the observed methods in mind, it can be concluded that including road geometry into motion prediction has been attempted in literature, but has not satisfactorily been solved. The road geometry can be suitably represented with splines are sequences of waypoints, allowing to extract a rich set of features based on the location of the vehicle. On the contrary, some promising deep learning model architectures have been proposed for the inclusion of the road geometry. Yet these methods do not rely on map-based prediction, but use less reliable sensory information for road geometry feature extraction.

2.4. Conclusion

Vehicle motion prediction is a complex technological challenge, because of the many factors that influence the driving behavior of traffic participants. In this chapter, some of the main focuses of motion prediction have been highlighted. With the introduction of deep learning, novel long-term prediction performance is achieved with only a limited amount of features, reducing the design complexity for motion prediction models. Models for sequence prediction such as the encoder-decoder from [45] have achieved this, and in the rapidly developing field of deep learning, more techniques are proposed each year.

Besides the challenge of obtaining more efficient models, information regarding the driving environment has to be included in the predictions. The interaction between vehicles is a dominant factor in driving behavior, and should be taken into account. Originally, interaction was a challenging aspect to model due to the coupling of driving behavior between vehicles. Suboptimal assumptions had to be made or large, computationally expensive models had to be used. With the introduction of Social-Pooling based methods, the interaction among a theoretically unlimited set of vehicles can be taken into account. These methods exploit the analytical capabilities of RNNs to recognize the interaction aspects with adjacent vehicles that have the most impact on the driving behavior. Social pooling methods result in increased prediction performance, and bring the field a step further to accurate and generalizable motion prediction models.

Another fundamental aspect of driving behavior is the infrastructure. Drivers are mostly forced to adhere to the shape of the road and the traffic rules. In classical methods, road geometry is often used for accurate motion prediction, because the shape of the road is a good indication of the motion that is to be expected. Elaborate sets of features regarding the lane on which is driven, or the intersection that is approached, can be extracted based on semantic map information. In recent work, some road geometry-based methods have been translated to deep learning methods.

However, it is not straightforward to combine semantic map information with deep learning models. The approaches for road geometry inclusion today are suboptimal. Current map-based methods exploit the powerful features, but suffer from a lack of comprehension of the global scene. Because of local reference frames, the global environment configuration can not be indicated. This makes these methods unsuitable to include information regarding adjacent vehicles, to obtain interaction-aware motion prediction. Furthermore, the global layout of the road segment ahead of the subject vehicle is unknown to the model. These limitations are not present in works where information from images is used as infrastructure feature source. Some interesting methods have been proposed to include vehicle-specific environment information in deep learning models. A drawback is the use of images as a primary source of road geometry information.

Therefore, the field is currently lacking methods to reliably include both road geometry and interaction in deep learning prediction models. The global configuration that is needed for deep-learning based interaction models can not be obtained with current road geometric approaches. The road geometry is either absent or lacks a global reference frame, where the model can link environment features to positions of the vehicles in the scene. A combination of such methods has never been included.

A road-geometric and interaction-aware prediction model based on the previously mentioned requirements

should be obtained. Such a model can generate map-based motion prediction, where the descriptiveness of road features from a semantic map are combined with the effectiveness of deep learning approaches regarding interaction. This would exploit the reliability of map-based features, to obtain more robust road-geometry- and interaction-aware motion prediction.

3
Methodology

To safely navigate through traffic, an autonomous vehicle is required to predict the motion of surrounding vehicles in the driving scene. This driving behavior consists of many factors, yet current models are not capable of including some important ones in their predictions. As seen in the previous chapter, deep-learning based motion prediction approaches show promise, but fail to include both the road geometry and interaction among vehicles in a reliable way. To include both the road geometry and interaction, new road-geometric modeling approaches have to be designed, taking inspiration from the approaches seen in section 2.3. Additionally, interaction-aware modeling approaches have to be used that were discussed in section 2.2. These modeling approaches have to be compatible, so that they can be combined into a road-geometry and interaction aware model.

In this chapter, the design of these models will be discussed. In section 3.1, the baseline sequence to sequence model will be detailed. Moreover, the choices that have been made in the model architecture design will be explained. In section 3.2, the method will be highlighted with which the baseline model is extended for interaction-aware predictions. In section 3.3, the author's novel contributions will be highlighted. To extend the baseline model to generate road-geometry aware predictions, two methods are proposed. These methods have been designed to overcome the observed shortcomings of the current approaches, and the reasoning behind them will be thoroughly explained.

3.1. Sequence to sequence prediction

With the baseline model, the observed past trajectory of the vehicle is used to generate a prediction of the future trajectory. This domain is referred to as sequence to sequence prediction, where one sequence is analyzed to generate another. In deep learning, the crucial building blocks of sequence to sequence prediction models are Recurrent Neural Networks (RNNs). The intuition behind RNNs and the chosen variant will be highlighted in subsection 3.1.1. With these RNNs, an encoder-decoder model can be constructed for the sequence to sequence prediction, which will be detailed in subsection 3.1.2. The features that are used and operations on the data for improved performance will be highlighted in subsection 3.1.3. Finally in subsection 3.1.4, the loss function with which the model performance is measured will be examined.

3.1.1. Recurrent Neural Networks

For data-driven analysis of time-series or other data with a sequential relation, Recurrent Neural Networks (RNNs) have proven to be effective models. RNNs are a flavor of Neural Networks, a branch of models that are inspired by the neural mechanisms in the human brain. For more information regarding the working of Neural Networks and Deep Neural Networks in general, we refer the reader to [19].

A recurrent neuron has a memory state, that is passed on to itself over time. This allows the model to 'memorize' the observations from the past inputs, giving it the capability to find sequential relations in the data and describe the behavior of dynamic systems.

The memory state that the RNN passes on to itself is called the hidden state h_t, and this hidden state is updated at every time step based on the new observations x_t. In figure 3.1, an RNN is depicted twice. The first representation is the RNN, represented as a single node. The second representation is more often used for clarity, and shows the same single RNN 'unrolled' over time. The hidden state is obtained as

$$z_t = W_{hx}x_t + W_{hh}h_{t-1} \tag{3.1}$$
$$y_t = f(z_t) \tag{3.2}$$

where W are the parameters or weights of the model that are learned throughout the training process, and f is some nonlinear activation function such as a sigmoid, tanh or Rectified Linear Unit (RELU).

Originally, the output y_t is identical to the hidden state. In some methods, the output is passed through some postprocessing layer called a projection layer. This layer projects the hidden state to the desired dimensionality, allowing the RNN to have a memory of different size than the desired output size. For this reason, the output of an RNN is often denoted with h_t rather than y_t. For the remainder of this thesis, h_t will be used to denote the output of an RNN, and y_t to the final predicted trajectory.

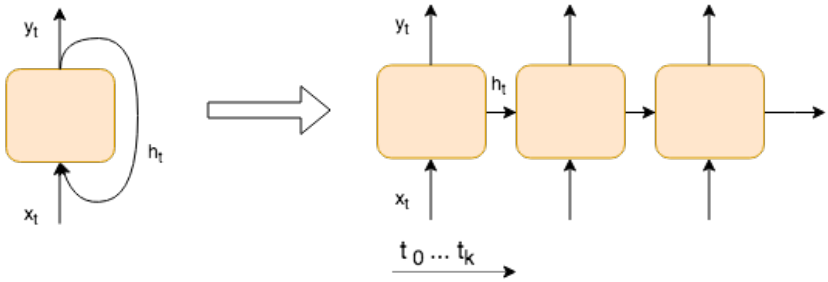


Figure 3.1: Left: an RNN with the recurring hidden state. Right: the same RNN, unrolled over time.

An RNN uses the same weight matrix to compute the hidden state over several time steps, depicted by the 'unrolled' RNN. This significantly reduces the amount of parameters, and allows the network to analyze a sequences of variable length. For RNNs, a special form of Back-Propagation (BP) is used called Back Propagation Through Time (BPTT). BPTT updates the weights of the model not just for every sequence of data, but for every input time step. The model parameters are thus still updated based on the error gradient at every input, similar to fully connected networks.

A downside of this model is that predictions that are further back in time hardly contribute to the weight updates. This phenomenon occurs because the weight update is proportional to the error gradient with respect to the weights. The error gradients earlier in the sequence are proportional to the error gradients later in the sequence as

$$\frac{\partial \varepsilon}{\partial W_t} \propto f'(z_t)f'(z_{t_1})..f'(z_{t+n}). \tag{3.3}$$

If the derivative of the activation function $f'(z) \ll 1$, which is the case for sigmoid and tanh activation functions, this error gradient vanishes for data early in the sequence. This means that the weights of the model will hardly be updated for early inputs. This is called the vanishing gradient effect, and prohibits the RNN from functioning effectively on long sequences.

To counteract the vanishing gradient phenomenon, alternative RNNs were designed. The first and currently most widely-used is the Long-Short Term Memory (LSTM)[21], and more recently the Gated Recurrent Unit (GRU) [12] was introduced. The core concept of both methods is similar, namely to let a memory state flow gradient-free through the model over time. Both the LSTM and GRU solve the vanishing gradient problem and improve general performance. In this thesis, the models are based on GRU RNNs for two reasons. The first is that both models perform similarly, but GRUs are more computationally efficient. The second reason is that the LSTM contains two memory states, the hidden state and the cell state. Contrarily, a GRU only has one hidden state, which is more convenient for the interaction method that will be detailed in subsection 3.2.

Gated Recurrent Units

A GRU is a modern RNN that can find patterns in long sequences, based on a hidden state. This hidden state is not directly obtained as the output of an activation function and weighted input, like in equation 3.2. Instead, there are multiple gates that update the hidden state at each time step.

First, the update gate decides what part of the past hidden state h_{t-1} is still relevant, based on the new observed input as

$$z_t = \sigma(W_{zx}\ x_t + W_{zh}\ h_{t-1}). \tag{3.4}$$

The sigmoid activation function maps all values between 0 and 1, meaning that z_t contains a 'relevancy' score for each hidden state entry. The second gate is called the reset or forget gate,

$$r_t = \sigma(W_{rx}\ x_t + W_{rh}\ h_{t-1}), \tag{3.5}$$

and is used to determine which parts of h_{t-1} is no longer relevant. Although equations 3.4 and 3.5 are identical, they serve a different purpose based on their usage. The reset gate is used to obtain a new hidden state estimate, based on the current input and the relevant part of the past input as

$$h'_t = tanh(W_{h'x}\ x_t + r_t \circ W_{h'h}\ h_{t-1}). \tag{3.6}$$

Here, the $\circ$ operation denotes an element-wise multiplication, degrading all the entry values that are deemed irrelevant by the reset gate. The tanh activation maps all values between -1 and 1, meaning that all hidden state values and output values will be within this range. Finally, this new hidden state is updated with the relevant part of the previous hidden state, as determined by the update gate h_t through

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h'_t. \tag{3.7}$$

These gates give the GRU an architecture that allows it to construct a hidden state by gathering the relevant information over time. The size of the hidden state of the model can be controlled. A larger hidden state can contain more information, but makes the model more prone to overfitting.

3.1.2. Encoder-decoder architecture

In motion prediction, a model is required that can translate a past sequence to a future sequence,

$$T_{future} = f(T_{observed},\ W), \tag{3.8}$$

where f is the RNN model, parametrized by the weights W. The trajectories are defined as

$$T_{observed} = \{s_{-k}, s_{1-k}, .., s_0\} \tag{3.9}$$

$$T_{future} = \{s_1, s_2, .., s_n\}, \tag{3.10}$$

where k and n are the number of past and future states s, respectively. To predict the future trajectory of a vehicle, more is needed than a single RNN cell. As seen previously, a GRU cell generates an output h_t, based on the previous output h_{t-1} and input x_t. This is insufficient, as a model is needed where the entire past sequence $T_{observed}$ is analyzed first, before generating any future predictions.

To this end, an encoder-decoder model can be used. At first, a GRU cell is used to analyze the past sequence. The output of the GRU cell is not of importance, except for the final hidden state at t_0. This hidden state is assumed to have encoded the entire trajectory in a fixed set of parameters; the size of the hidden state. This GRU is the model encoder. The encoded trajectory is used to initialize a second GRU cell at the beginning of the sequence. This second GRU cannot observe past states, but is instead asked to predict new future states based on the encoding of the past observed trajectory (figure 3.2). This GRU thus acts as a decoder, decoding the hidden state into a future trajectory. By generating a new state s_t at every time step and using this as the input x_{t+1} at the next time step, the decoder can generate a theoretically unlimited amount of future states.

The role of the encoder is especially interesting. Because the outputs of the encoder are not used except at t_0, it will not be penalized for the outputs it gives at previous time steps. This is due to the properties of back-propagation, where the model parameters are updated based on their contribution to the error. If the output

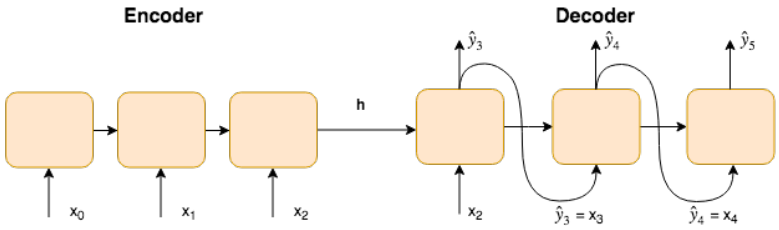


Figure 3.2: Sequence to sequence prediction by an encoder-decoder model.

is not used, it will have no contribution to the error and will thus be neglected in the weight updates. The encoder learns its 'role' in the model by the way its outputs are used. This means that the output of the encoder will have no physical meaning, but is designed to construct a memory state that represents the relevant parts of the observed trajectory accurately.

In the baseline model, the encoder consists of a single GRU cell with a hidden state size of 48 parameters. The decoder consists of two stacked GRU cells to improve the inference performance of the encoded trajectory. The first decoder cell passes its output on to the input of the second cell. Both GRU cells are initialized with the hidden state of the encoder (figure 3.3).

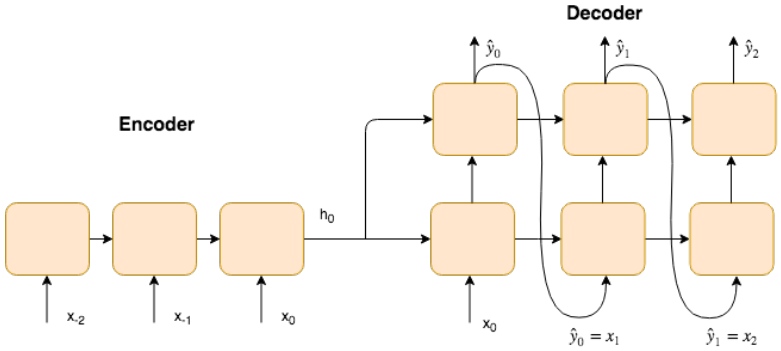


Figure 3.3: A encoder-decoder model with stacked decoder cells.

Classic RNN cells and GRU cells contain a single hidden state h that is the output of the model at the same time. Often, the number of parameters of the hidden state do not match the number of desired outputs, i.e. the features that describe a state s. In this case, the state consists of an (x,y) coordinate, whereas the hidden state consists of 48 parameters. To this end, a projection layer can be used,

$$y_t = f(W_{yh}\, h_t) \tag{3.11}$$

which is a fully connected layer that projects the hidden state into the output dimensionality. Often, the activation function of the projection layer is linear, meaning that equation 3.11 can be reduced to

$$y_t = W_{yh}\, h_t. \tag{3.12}$$

The output projection layer from equation 3.12 is used in all the models in this thesis.

3.1.3. Input embedding

The encoder-decoder model that is described in the previous subsection is sufficient to predict future trajectories based on the past observations. However, it is often beneficial to add a preprocessing step where the model learns to translate the input to a more effective representation. In this model, a Temporal Convolution Layer (TCL) is added for this purpose. This layer is added at the beginning of the model, taking in the original observed trajectory and outputting some higher dimensional input for the encoder.

Convolution layers are often used in Convolutional Neural Networks (CNN) for image analysis. Temporal convolution refers to a convolution over a single dimension (time), rather than a 2D pixel space for example. The convolution of a signal *f(t)* and system *g(t)* is defined as

$$[f * g](t) \equiv \int_0^t f(\tau) g(t-\tau) d\tau. \tag{3.13}$$

Essentially, the convolution of a signal and a system is the integral over time of the signal and the system mirrored over the y axis, see figure 3.5. In this thesis, $f(t)$ is the input trajectory and $g(t)$ is a so-called convolution filter or convolution kernel. The convolution filter is in essence a filter mask, that at each time step determines how much the data around the current time step are of importance. This is represented in figure 3.4 for a 2D case, where the matrix on the left (signal) is convolved with the filter on the right for the red pixel box. The filter indicates that the only part of the data that is interesting, is the entry above the red box. For the temporal convolution from equation 3.13, the kernel *g(t)* determines for each state in trajectory *f(t)* what past states are important and how important. The resulting feature can be seen as a weighted average of the input data over a number of time steps.

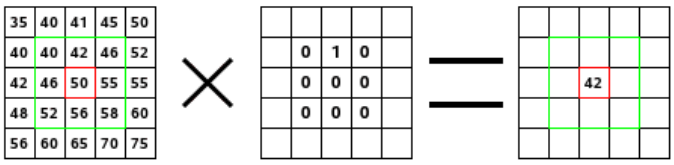


Figure 3.4: A convolution mask (right) convoluted with a matrix (left).

In the TCL, the filter values of *g(t)* are learnable. The TCL learns what past states $[s_{-n} .. s_0]$ are important for the feature at the current time step. Convolution operations are useful for a number of reasons:

- Find patterns in signals
- Filtering signals
- Find correlation between signals

In this particular application, the first two reasons are of importance. The main advantage of the TCL is that it allows the model to find velocity and acceleration patterns in the sequence more easily. Moreover, it acts as a filter on the input data.

There are two hyperparameters that can be controlled for the TCL. First, the convolution filter size has to be indicated. This filter size determines the amount of past time steps that are taken into account to obtain a feature at the current time. The second hyperparameter is the amount of convolution filters that are used. By using multiple filters, each can learn a different task e.g. smoothing the data or extracting different features from the same data. Because in this case the input sequence consists of multiple features, namely the x and y coordinates, each new feature is obtained by using the same convolution filter separately over both coordinates and adding the resulting features.

In this model, the kernel size is chosen as 3, similar to the kernel in figure 3.5. This includes enough time steps to extract information regarding accelerating behavior. Inspired by the TCL from [30], it was found that the models perform optimally when 16 kernels are used. This results in an embedded input trajectory containing states of 16 features at each time step.

Features

Traditionally, the states s_t at different time steps are described with an elaborate feature set. This was seen in section 2.1.4, where a feature set was given as

$$S_t^{(n)} = \{v_t, \dot{\psi}_t, x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)}\}. \tag{3.14}$$

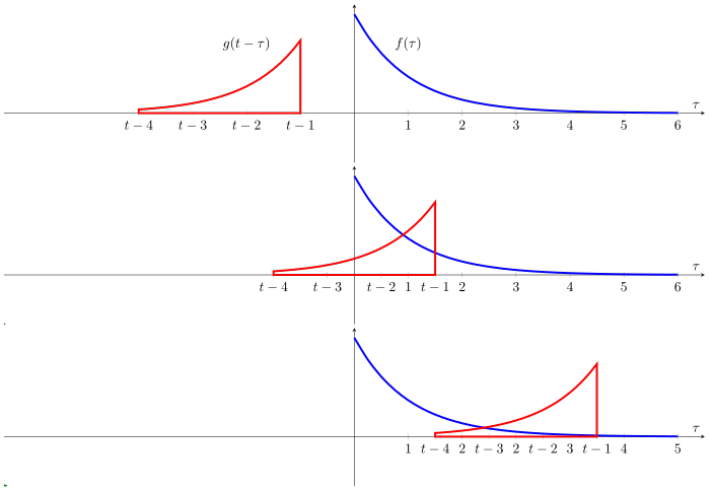


Figure 3.5: The convolution between a signal *f(t)* and filter *g(t)*

In this model, there is chosen to keep the input more sparse, including only the x and y coordinates, reducing $S_t^{(n)}$ to

$$S_t^{(n)} = \{x_t^{(n)}, y_t^{(n)}\}. \tag{3.15}$$

This is done because the TCL is assumed to extract time-related features such as velocities, accelerations, and orientation changes. Instead of manually differentiating the data and inputting it as extra state features, the TCL is able to estimate them if the kernel size covers a sufficient amount of time steps. The filtering properties of the convolution kernel is expected to make the time-dependent feature extraction extra effective, compared to manual differentiation of the data. Empirically, it was found that covering more than 3 time steps in the convolution does not further increase performance.

3.1.4. Normalization & Loss

The recorded trajectories that are used to train the model, are defined as 2D coordinates with respect to some coordinate frame. The values of these coordinates can take any value $(x, y) \in \mathbb{R}^2$. To obtain more consistent predictions invariant of the location, trajectories are often translated to start at the position $(0,0)$ as

$$T_{translated}(t) = T(t) - T(0) \tag{3.16}$$

for improved performance of the model.

Normalization

RNNs are especially sensitive to the value range of the data. In equations 3.4 - 3.8, all internal operation of the GRU cell are given. Here, it can be seen that all GRU gates use sigmoid activation functions, producing values between [0, 1]. The hidden state is updated by passing it through a tanh, mapping all values between [-1, 1]. Therefore, the GRU internally always operates with values on the interval [-1, 1]. Using data in different orders of magnitude can slown down learning and decrease convergence of the network. Therefore, the trajectories are not only translated to start at position 0, but also normalized so that majority of the values lie within either [0, 1] or [-1, 1], depending on what the fits the data best. The same normalization is applied to all the data, keeping all the data on an identical scale.

Huber loss

The loss function is an important part of the model, giving a performance measure for the output of the network. In trajectory prediction, the loss function should be based on the actual future trajectory (ground truth), and the prediction that is generated by the model. Since trajectory predictions amounts to a regression task, where the state at a time step has to be quantified, a regression loss is desired. The most frequently used

regression loss functions are the Mean Absolute Error (MAE), and the Mean Squared Error (MSE), given by

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.17}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.18}$$

where N represents the number of prediction steps, y_i the ground truth and $\hat{y}_i$ the prediction. In the case of motion prediction, y_i consists of the coordinates $(x, y)_i$, and the error is the sum of the MSE/MAE of each coordinate.

The loss function that is chosen represents the objective of the model, and choosing a different loss function leads to different performance. The MAE makes the model look for the median prediction, and is relatively insensitive to outliers. A downside of the MAE is that its gradient does not change, meaning the error gradient (and weight updates) stay relatively large for smaller errors. This leads to a decrease in convergence. In this regard, the MSE has the opposite properties. The error gradient decreases when the error decreases, improving convergence to a desired model. The downside of this is that errors larger than 1 will grow quadratically. This makes the MSE loss more prone to outliers.

To overcome these downsides, the Huber loss was introduced. The huber loss combines the robustness against outliers of the MAE, with the convergence properties of the MSE. Essentially, the huber loss is an adapted MAE loss for large errors, and a MSE loss for small errors. It is defined as

$$L_\delta(\epsilon) = \begin{cases} \frac{1}{2}(\epsilon)^2, & \text{for } |\epsilon| \le \delta. \\ \delta|\epsilon| - \frac{1}{2}\delta^2, & \text{otherwise.} \end{cases} \tag{3.19}$$

$$\epsilon = y - y_i \tag{3.20}$$

where δ is a hyperparameter to indicate at what size of the error the transition between the two losses should be. The Huber loss is particularly useful for RNNs with normalized data. Since the majority of the values will be within a range of [-1, 1], prediction errors both larger and smaller than $\epsilon = 1$ will occur. With the huber loss, there will be less inconsistent weight updates during training.

3.1.5. Summary

In this section, the components of the baseline prediction model have been outlined. The model is optimized to generate accurate future predictions, based on the past predictions. The GRU cell is the fundamental RNN building block of the model. With this building block, the encoder-decoder model can be constructed to obtain future sequence predictions. The input data is encoded into a fixed amount of parameters, called the hidden state. The hidden state is decoded by a decoder, and used by a projection layer to project its output into the 2-dimensional coordinate space.

To let the encoder better find patterns over time, the Temporal Convolution Layer embeds the states into a new feature space. It allows the model to find time-related features such as velocity and acceleration patterns, by learning a convolution filter over a fixed time interval. The input data to the TCL is normalized to put the data in the preferred value range for GRU RNNs. The performance of the model is then assessed in a specific loss function called the huber loss, for optimal performance convergence.

With this model, several techniques are present to optimally extract the relevant information from the input trajectory, and generate a new prediction. This model will serve as the baseline prediction model, showing what performance can be obtained by looking solely at past observations. The modules that will be detailed in the next sections are additions to this model. These modules will add functionality but the baseline architecture will stay unaltered, so that the observed changes in performance are completely due to the additions to the model.

3.2. Interaction

In the previous section the baseline prediction model is detailed, that generates a prediction on the past observed trajectory. As explained in Chapter 2, motion prediction is heavily influenced by interaction. In this

section, the social pooling method is detailed that will be used to obtain more accurate predictions. In the first subsection, details will be given on the adjacent vehicles that are important to take into account, and how these vehicles can be included in the prediction. This includes the analysis of individual adjacent vehicles, as well as their global configuration. In the second subsection, an elaborate explanation will be given on how the information can be used with social pooling to obtain a prediction.

3.2.1. Adjacent vehicles

When driving in an environment, interaction can occur with vehicles that are in the vicinity, or further away. Vehicles that are important for interaction are often the preceding vehicle, vehicles on adjacent lanes and close to the subject vehicle, or vehicles whose future path is intersecting the subject vehicle. As described in section 2.2, with social pooling it is the model that determines which vehicles are important for prediction. If it is unclear which adjacent vehicles are the most important for prediction, then all of these adjacent vehicles can be taken into account.

In this thesis, 5 adjacent vehicles will be used; the preceding vehicle and two vehicles on each adjacent lane (figure 3.6). An adjacent lane vehicle is only selected if there is an adjacent lane, and if the vehicle is slightly behind the subject vehicle or in front of it. In a driving environment consisting of multiple lanes, this should capture all the necessary interaction influences that exist.

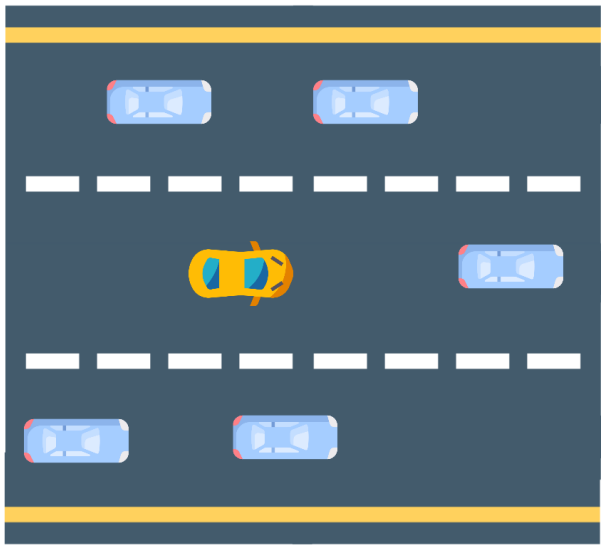


Figure 3.6: All vehicles that are taken into account for motion prediction of the subject vehicle (yellow).

The adjacent vehicle analysis is identical to that of the subject vehicle. The past observed trajectory of the adjacent vehicle T_{adj} is used to capture the past motion. This is accomplished by using the same encoder from the encoder-decoder model for all adjacent vehicles. For each adjacent vehicle n, a hidden state $h_{adj}^{(n)}$ containing the past encoded trajectory is obtained. These hidden states are in essence feature vectors, containing the necessary information of the adjacent vehicle for the prediction.

Because the trajectories are translated to start in the origin, the spatial configuration of these vehicles is lost. To the model, it seems as if all the trajectories started from the same point in space, being the origin. To include the spatial configuration, multiple solutions have been proposed, as shown in chapter 2.2. In this thesis, the relative position embedding is used instead of the grid-based Convolutional Social Pooling (CSP). Whereas CSP requires to build a grid of fixed size of the entire scene, the relative position embedding is a more efficient approach that does not require a separate Convolutional Neural Network to consume a grid map with redundant information. This reduces model complexity and computational cost and is expected to be the more effective modeling approach. In this method, the relative position between two interacting vehi-

cles is taken, and embedded in their hidden state. Since we are interested in the subject vehicle, the relative position of an adjacent vehicle with respect to the subject vehicle is embedded in the adjacent vehicle state. The relative positions are obtained from the original untranslated trajectories, normalized, and concatenated to $h^{(n)}_{adj}$. This concatenation is passed through a fully connected embedding layer, which transforms it back to an array of the hidden state size. Thus, for each adjacent vehicle the hidden state is updated as

$$h^{(n)}_{emb} = \left[h^{(n)}; \Delta x; \Delta y\right]^T \tag{3.21}$$

$$h'^{(n)}_{adj} = f\left(h^{(n)}_{emb}, W_{emb}\right) \tag{3.22}$$

where $h'^{(n)}_{adj}$ is the spatial hidden state of size M, $[\Delta x, \Delta y]$ are the relative positions, *f(x)* is the embedding layer and W_{emb} are the layer parameters. Therefore, after this step we have 5 spatial hidden states for the adjacent vehicles, and the original hidden state from the subject vehicle. With these updated hidden states, all information is present to obtain interaction-aware motion prediction.

3.2.2. Social pooling

Now that a hidden state is obtained for every adjacent vehicle of interest, the interaction features can be obtained. The hidden states can be concatenated to obtain an interaction tensor $P_i \in \mathbb{R}^{MxN}$

$$P_i = \left[h'^{0}_{adj}, .., h'^{n}_{adj}\right] \tag{3.23}$$

The goal is to summarize P_i in a smaller representation, giving a vector I containing the relevant interaction features. A naive implementation would be to simply pass P_i through an embedding layer, projecting the n features in each row to a single value. This is a poor approach, because a neural network learns weights for each specific entry, making the order in which the hidden states h'^{n}_{adj} are placed important.

Therefore, it is preferred to use a symmetric function that obtains the same result, regardless of the order in which the n features are placed. Examples of symmetric functions are multiplication, averaging or taking the extreme values. In [38] it is shown that a neural network can learn to be symmetric, i.e. produce the same output regardless of the input order. This can be achieved by using one or multiple fully-connected embedding layers, followed by a symmetric function. The result is a symmetric embedding operation, where the embedding layer extracts the useful features from the adjacent hidden states and returns a single feature vector of size M, containing the relevant information.

In social pooling, two symmetric functions are particularly useful. The first is taking the average of the features, and the second is taking the maximum value. There exists no consensus on which function should be the preferred one, but here we have chosen for the MAX operation. The intuition is that in social pooling, the feature vector should contain all aspects that have a strong impact on the driving behavior of the subject vehicle. By choosing the max function, the dominant features are extracted, rather than the average of all vehicles in the scene. This makes the model more robust to redundant information, about adjacent vehicles that have no influence on the driving behavior. Their information will not obfuscate the relevant information, because their hidden states will be omitted in the max social pooling. This intuition has been confirmed empirically.

Now that an interaction feature vector I is obtained, interaction-aware motion prediction is within hand's reach. Feature vectors in an encoder-decoder model can be added as auxiliary inputs to the decoder. This means that at every time step, the feature vector gets appended to the input of the decoder. However, in [18] a more efficient method is proposed. The interaction features can be embedded in the subject vehicle hidden state, again by means of concatenation and passing it through a new embedding layer. If the resulting state is of equal size to the original subject hidden state, the decoder can be initialized with this new interaction-aware state. This does not decrease performance, and removes the need to feed the interaction features to the decoder at every prediction time step. The entire interaction-aware architecture is represented in figure 3.7.

3.3. Road geometry

With the encoder-decoder model described in section 3.1, and the social-pooling method from section 3.2, improved predictions can be obtained. In this section, the proposed methods for road-geometry aware mo-

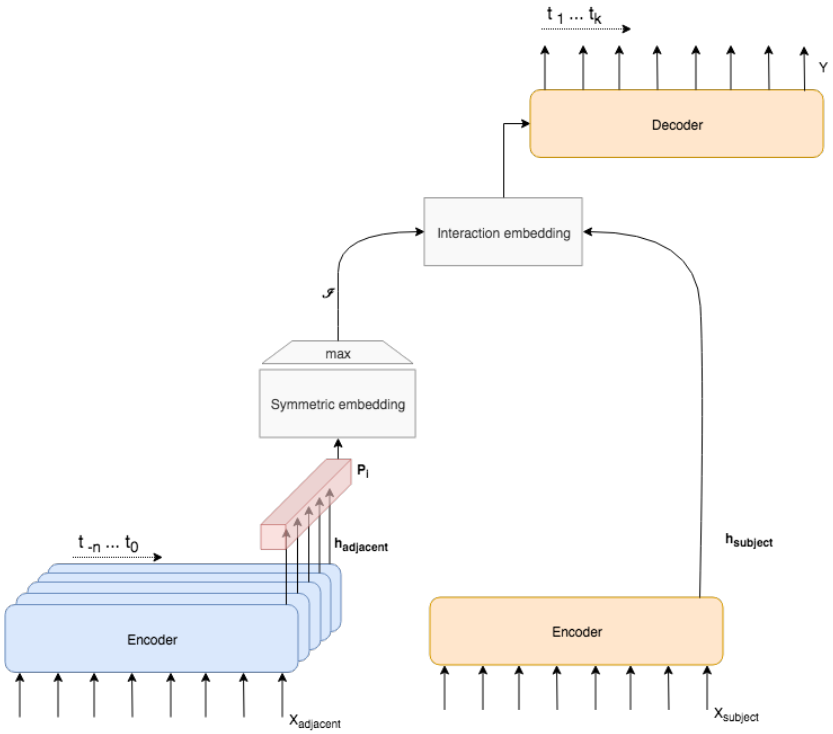


Figure 3.7: An encoder-decoder model with social pooling. Blue represents encoding of the adjacent vehicles, yellow for the subject vehicle.

tion prediction will be proposed. Similar to the Social Pooling method, these two methods are extensions to the baseline model. First, the used semantic map will be detailed. This includes the chosen road representation and available features that can be used by the model.

Thereafter, the road geometry methods will be detailed. As explained in chapter 2, there are two ways of communicating the road information to a model. The entire road segment can be exposed to the model before generating a prediction, making the model responsible for the extraction of the relevant features prior to the prediction. We refer to the proposed method as road-attention. Alternatively, the predicted vehicle states can be used to find the location-specific features manually, and update the predictions accordingly. This method will be referred to as road-refinement, and will be detailed in the last subsection of this chapter.

3.3.1. Semantic map

As mentioned in chapter 2, map-based motion prediction models extract information from a semantic map. Such a semantic map can be constructed offline, and can provide more detailed information regarding the driving environment. In this thesis, a semantic map is constructed that will primarily include information regarding the road geometry. There is chosen to represent the lanes with a waypoint representation. To keep the interpolation error minimal, the centerlines should be represented by a sufficiently dense sequence of waypoints, see figure 3.8. Therefore, the waypoints are placed at a distance of 1.5 [m] from one another. They are defined as a point in 2D space with a global x and y coordinate.

To be able to classify the driven lane, a waypoint sequence will be defined for each adjacent lane in the scene, rather than one sequence for all lanes. These waypoints are annotated with the ID of the lane that they represent. Each lane contains a feature that indicates whether there is an adjacent lane left and right that can be entered. By finding the waypoint that is closest to the vehicle, a feature can be obtained that indicates on what lane is driven and whether it is possible to change lanes left or right.

To obtain features regarding the road geometry itself, the road segment between the waypoints should be described. This is done by fitting a single parametric B-spline over the entire lane centerline. A parametric

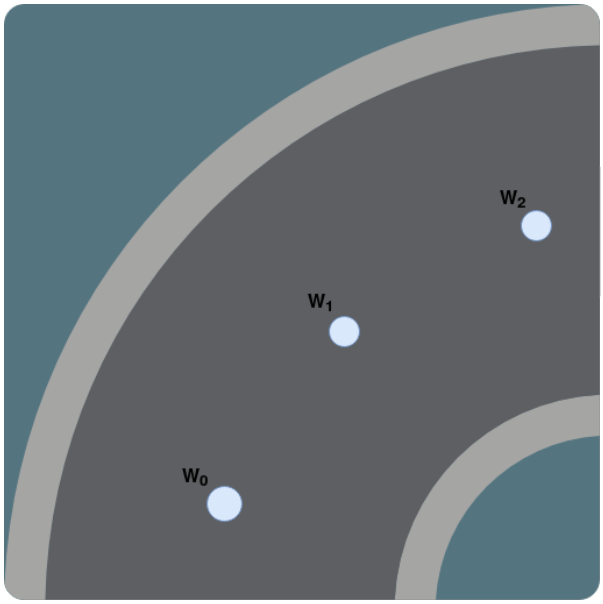


Figure 3.8: Waypoints representing the centerline of a single lane.

B-spline is defined as

$$f(t) = g(t)i + h(t)j \tag{3.24}$$
$$g(t) = A_x t^3 + B_x t^2 + C_x t + D_x \tag{3.25}$$
$$h(t) = A_y t^3 + B_y t^2 + C_y t + D_y \tag{3.26}$$

where $t \in [0,1]$ is the parameter that indicates the location on the B-spline, with $t = 0$ being the origin and $t = 1$ the end of the spline. The B-spline can first of all be used to find the Active Lane Point (ALP) on the centerline. Given a predicted location of the vehicle, the ALP is obtained as the point on the spline between the two waypoints closest to the vehicle on that spline.

With the ALP, the Distance To Centerline (DTC) can be obtained. This is the euclidean distance from the centerline to the vehicle position. To obtain a more effective feature, the DTC is mapped to a value range [-1, 1]. In this way, the DTC is invariant to the size of the lane, and $DTC < -1$ and $DTC > 1$ indicate lane changes. To obtain the DTC, the orthogonal distance is first obtained and scaled with the distance from the centerline to the lane boundary, as

$$DTC = \lambda \frac{1}{D_{lane}} \sqrt{\left(x_{alp} - x_{vehicle}\right)^2 + \left(y_{alp} - y_{vehicle}\right)^2}, \tag{3.27}$$

where D_{lane} is the distance from the centerline to the lane boundary, and $\lambda \in \{-1, 1\}$ denotes the side of the lane. It is positive if the vehicle is on the left side of the lane, and negative if it is on the ride side. λ can be obtained with the cross product between the normalized spline derivative $u = \frac{\partial f}{\partial t}$ at the ALP, and the normalized direction vector from the ALP to the vehicle position $v = [x_{vehicle} - x_{alp}, y_{vehicle} - y_{alp}]$, as

$$\lambda = \frac{u \times v}{||u \cdot v||} \tag{3.28}$$

meaning that if the orthogonal vector of the plane spanned by u and v contains a positive z coordinate (points upwards), the vehicle is left of the centerline.

The final feature that can be obtained from the spline is the curvature of the road at the ALP. The curvature is a clear indication of the shape of the road segment and can be used to improve prediction accuracy, as shown in [39]. The road curvature of a parametric curve is given as

$$\kappa = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{\frac{3}{2}}}. \tag{3.29}$$

Summary
The road geometry in the semantic map is described per lane, to obtain lane-specific features as well as features describing the amount of lanes and lane-changing capabilities. Defining all the lanes in the scene is done with a sequence of waypoints, over which a B spline is fit. With this representation, the following set of features can be extracted from the semantic map:

- Road shape
- Active Lane Point
- Distance To Centerline
- Road curvature
- Current occupied lane
- Adjacent lane presence

3.3.2. Road refinement

With the features from the semantic map, several aspects of the road can be described. In this proposal, the predictions of the baseline model will be refined to better match the road geometry. To this end, the baseline model described in section 3.1 is taken as starting point and extended with the road-refinement module.

Looking up information in a semantic map is done based on the predicted geolocation of the vehicle. In the baseline model, the decoder generates a trajectory $T_{future}(t)$ that contains the locations of the subject vehicle at K future time steps. Assuming a state $s_t = \{x_t,\ y_t\}$ at future time step t, the waypoint v_c that is closest to s_t can be found. By assessing whether this waypoint is behind or in front of the vehicle, the two waypoints that describe the road segment on which s_t lies can be obtained. At every prediction step in the decoder, the features ϕ are extracted from the map, and appended to the next input of the GRU, see figure 3.9. In this way, the decoder looks up the appropriate features during prediction.

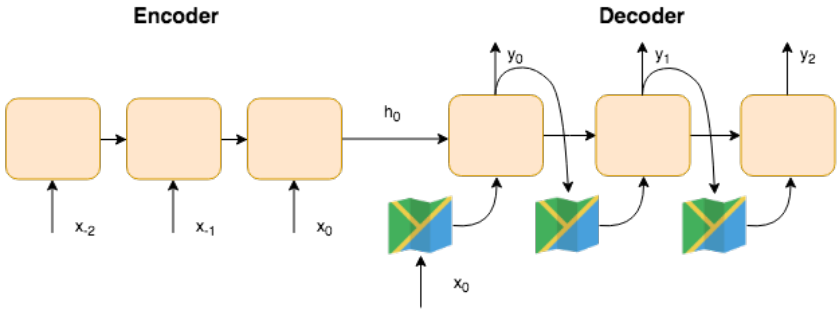


Figure 3.9: encoder decoder model with a lookup map.

This straightforward lookup approach fails to deal with one issue. The features are obtained with the predictions from the decoder, and the predictions of the decoder are based on the features that are looked up: $y \rightarrow \phi \rightarrow y' \rightarrow \phi'$, see figure 3.10. Thus, this method introduces a mutual dependency between map features and predicted state. The predictions of the decoder y_t are used to obtain the features ϕ_{t+1}, and the concatenation $[y_t; \phi_{t+1}]$ is the GRU input to the next time step to obtain y_{t+1}. This implies that the map features that are used to update the predictions are no longer valid, because the predictions themselves have changed.

To resolve this, an iterative scheme can be used, as described in section 2.3. In this iterative scheme, the process of obtaining the predictions and features is repeated until they have satisfactorily converged to a stable prediction. In such an iterative scheme, the predictions based on the past observed trajectory $T_{observed}(t)$ and map features are decoupled. First, a road-agnostic prediction Y is generated by the baseline model. This prediction is used to look up features ϕ_t in the map, belonging to the state at each time step t. A regression vector ΔY can then be generated, based on the initial prediction and the map features, and used to update the prediction.

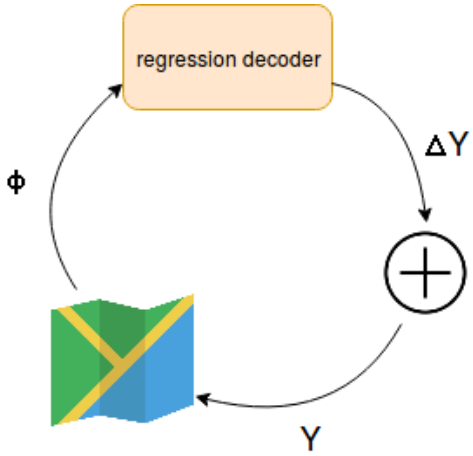


Figure 3.10: Iteration between the map features and predictions.

In this work, there is chosen for a second RNN decoder to obtain the regression vector ΔY, similar to [30]. This regression decoder is initialized with the same hidden state as the original decoder. It takes the original predictions as input and looks up the features to generate the regressed state Δy_t at each time step. In this way, the regression decoder is aware of the original encoded trajectory, the predictions of the road-agnostic decoder and the road of the scene. This process can be repeated for a number of iterations, where the updated output $Y^{'}$ is fed back into the regression decoder to generate a new regression vector $\Delta Y^{'}$ and update the predictions accordingly. A schematic representation of this model is given in figure 3.11. By repeating the iteration until the prediction converges, a correct road-aware prediction should be obtained.

As explained before, a set of features is looked up to describe the local road scene. The normalized Distance To Centerline (DTC), as well as the centerline projection (ALP) are given. The ALP is a relatively continuous feature, indicating where the centerline of the current driven lane is regardless of any lateral deviations. The DTC is an indication of this lateral deviation, showing the normalized lateral position on the lane. The ALP and DTC are considered complementary features that will both increase the capability of the network to predict. During prediction, the model is expected to use these features to observe when the predicted trajectory deviates from the centerline. The task is then to determine whether this is due to a prediction error or because of lane changing behavior, and update the prediction accordingly.

For the more global information regarding the road the curvature is given, as well as the lane position. The lane position is considered important when multiple adjacent lanes are used in the map. When prediction s_{t+1} lies on a different lane than s_t, the features will be looked up in the adjacent lane, changing the ALP and DTC to the respective lane. By including the lane position vector, the network can observe a lane change and can comprehend the sudden feature changes. The lane position is given as a one-hot encoding of the current lane. This means that it consists of a 1D array with one entry for each lane. The entire array is set to 0, except for the driven lane, which is set to one. The lane position feature for the third lane on a 6 lane highway would look like $[0,\ 0,\ 1,\ 0,\ 0,\ 0]^T$. It thus represents a simple global overview of the amount of lanes that are present.

Convergence loss

The map features that are detailed above describe the road geometry both locally and globally. When using these iteratively with the regression decoder, the predictions can be refined to match the road geometry.

Because the map features are based on the predictions, the predictions should have some level of accuracy. If the predictions are completely false, the road scene around this false location will be predicted. The map features then have no correlation to the desired ground truth output, and will be noise to the model. This will have a negative impact on the convergence of the model. Therefore, it is important that the predictions that are refined have some initial accuracy to them.

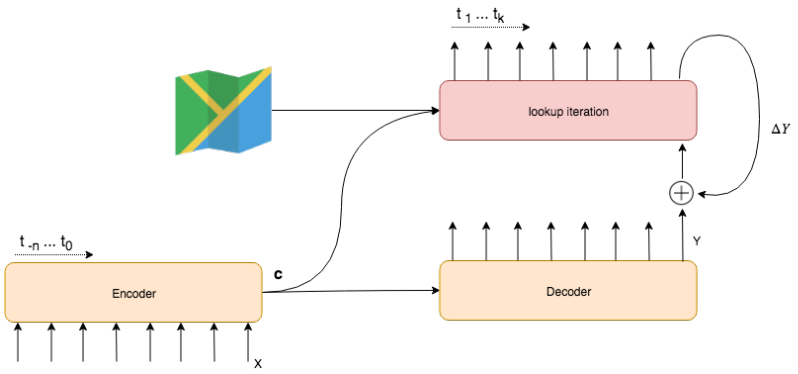


Figure 3.11: The road-refinement model.

One method to counteract this, is to pretrain the baseline model without the regression decoder. The model is then initially trained on the same dataset, learning to predict solely on the past observed trajectory. When learning converged to a satisfactory degree, the parameters can be used to initialize the encoder and decoder in the road-refinement model. There are two main drawbacks to such an approach. Firstly, this is a cumbersome method, because the model has to be trained twice on the same data. Secondly, this reduces the coherence between the different components of the model. As explained in section 3.1, different components of the model obtain a 'role' by learning from their contribution to the prediction error, by means of backpropagation. In essence, the components generate their output in a way that is most convenient for the next parts of the model. In this model, the hidden state from the encoder is used in both the original decoder, as well as the regression decoder. The encoder will thus converge to an encoder that generates hidden states that can most effectively be used by both decoders. By pretraining part of the model, this will not be the case.

Instead, we introduce a new loss function to counteract the convergence problems. In all other models, the loss function consists of the Huber loss of the predicted trajectory $\hat{Y}$ and the ground truth Y. In this model, this would imply the Huber loss of the refined prediction. Instead, the loss consists of both the Huber loss of the original prediction, as well as the refined prediction, defined by

$$L_{total} = L_{huber}(Y, \hat{Y}) + L_{huber}(Y, \hat{Y}'). \tag{3.30}$$

The model is penalized for both the predictions of the baseline model, as well as the refined predictions. It is thus asked to generate meaningful predictions, based on just the observed trajectory as well as the road geometry. In this way, the baseline part of the model can converge to a state where it generates meaningful predictions, even though the lookup features are still without meaning.

Conclusion

The road-refinement model introduces a new map-based approach to include the road geometry in predictions. The features from the semantic map describe the road both locally and globally. To ensure convergence and effective use of the features, a convergence loss is introduced that is robust to lookup functionality with poor prediction performance.

Besides the fact that the road-refinement method introduces the road-geometry, it is fully compatible with the interaction method. The regression decoder can refine the original predictions, regardless of what they are based on. Furthermore, the regression decoder is also initialized with the hidden state from the encoder. If social pooling is added to the baseline model, the hidden state that is received by the regression decoder thus also contains information regarding the interaction. In such a setup, it would be fully aware of the interactive forces that determined the original prediction.

3.3.3. Road attention

In this section, the second road-geometry approach will be detailed. The second approach to communicating the road geometry is to present a road segment that is not specific to the prediction. Rather than iteratively

updating the prediction, the initial prediction is road aware. In this method, the model is responsible for selecting the appropriate information from the map and obtaining the right features.

In this approach, the road geometry should be known to the model prior to any predictions. As a consequence, the lookup approach can not be used, and all the location-based features can not be extracted. Instead, for every observed trajectory $T_{obs}(t)$, an entire road segment is selected. This road segment is the lane centerline ahead of the vehicle at the last observed state. It consists of N waypoints $R = [v_0, v_1, .., v_N]$, where N is a fixed number for all vehicles, to provide a consistent information source. These waypoints are defined in the same scale as $T_{obs}(t)$, meaning that the waypoint coordinates are defined with respect to the origin at $T_{obs}(0)$ and normalized.

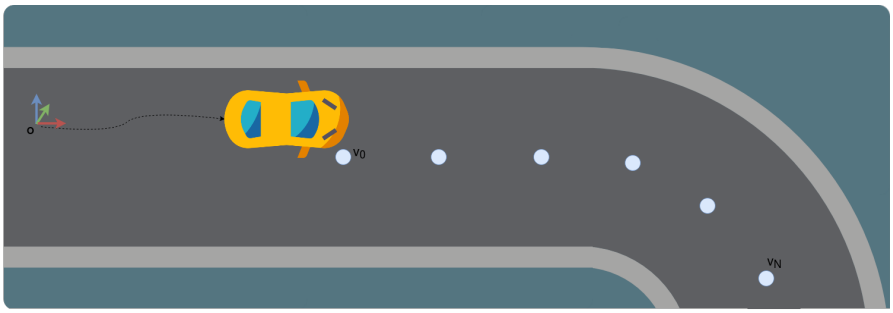


Figure 3.12: Selecting a road segment ahead of the vehicle.

Because both $T_{obs}(t)$ and R are defined in a global coordinate system, their configuration is preserved. The waypoint coordinates of R are first embedded in a more effective representation. Because of the sequential structure of R, a RNN is used to analyze the road ahead of the vehicle. Therefore, the road-RNN outputs a road feature vector for every waypoint location, containing information of the road up to and including the current waypoint.

As explained in section 3.1, the descriptive power of RNNs can be controlled by setting the hidden state size. In the road-RNN, the size of the hidden states is equivalent to the road feature vector size, and thus to the level of detail with which the road is described. To this end, the road-RNN can be chosen to be smaller than the trajectory encoder. Firstly, because the road RNN is not an encoder where only the final hidden state is used. Instead, all outputs that are produced throughout the sequence are considered. Secondly, the amount of variation in the shape of a road is significantly less than that of a driven trajectory. The set of features that is needed to describe the facets of the sequence can be assumed to be lower for the road. Emperically it is found that the hidden state size reduction is effective up to 25%, giving a road-RNN of size 36 for optimal results. With this road-RNN, the model is expected to more effectively analyze the shape of the road and extract all necessary features.

So far, we obtained a different representation of the same road segment, without knowing what part of it is relevant for the prediction. The relation between the road segment features and the future trajectory is still unknown. For this reason, simply providing the road-RNN output as input to the decoder is impossible. A solution could be to use the road-RNN as a second encoder, embedding the final road hidden state and trajectory hidden state into a new state. This includes redundant information regarding the entire road, because a road-encoder is unaware of what section is important. Such redundancy could obfuscate the important information that is already contained in the hidden state on the past driven trajectory, and possibly adjacent vehicles.

For this reason, in this thesis is chosen to use an attention mechanism for the road information (figure 3.13). An attention mechanism learns to focus on specific input data, by determining what part of a data source is important for a given task. It does so by reflecting on the input data with the current output of the decoder. Multiple variants of attention mechanisms have been proposed, but here we use the Bahdanau Attention. The initial output of the decoder is denoted as h_t, the target hidden state or target output. The source of

information is referred to with h_s, the source hidden state. In the case of the road-RNN, h_s is the output at each waypoint and h_t is the output of the decoder *before* the output projection layer.

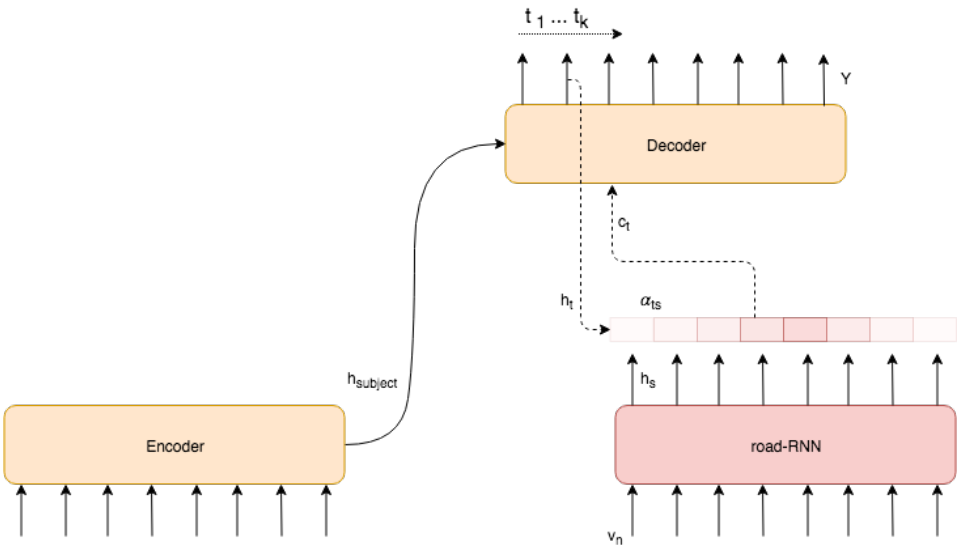


Figure 3.13: The encoder-decoder model with road attention.

First, an 'alignment score' a_{ts} is obtained. This alignment score is an indication of the relevancy of each source sample $h_{s_1..s_N}$ for the current output h_t. The alignment scores are then transformed to probability scores α_{ts} on a probability distribution as

$$a_{ts} = W_b \, tanh\big(W_a \, [h_t; h_s]\big) \tag{3.31}$$

$$\alpha_{ts} = \frac{e^{a_{ts}}}{\sum_s e^{a_{ts}}}, \tag{3.32}$$

where W_a, W_b are weight matrices that can be learned to map the concatenation $[h_t; h_s]$ to the alignment score. The softmax layer in 3.29 maps the alignment scores to a probability function. With α_{ts}, a feature vector called the context vector can be created with all the relevant source information for the current time step,

$$c_t = \sum_s \alpha_{ts} h_s. \tag{3.33}$$

The context vector c_t is thus a weighted combination of all the source states. What happens with this context vector depends on the attention mechanism that is chosen. In Bahdanau attention, the context vector c_t is not directly used to compute a new output $\tilde{h}_t$. Instead, it is appended to the output h_t, so that it is first processed by the GRU and used for output h_{t+1}.

For the sequence to sequence model with the road-RNN, the attention mechanism overcomes the problems that are mentioned earlier. The alignment process autonomously learns which part of the road is relevant at each time step. In this way, no redundant information is passed to the decoder, as would be the case in the embedding approach. Furthermore, the alignment scores provide a useful insight in the reasoning mechanisms of the model, indicating which part of the road input it finds relevant. If the alignment scoring layer can learn to properly score the source, the attention mechanism should be able to incorporate the road geometry in the predictions.

Conclusion

The road-attention model introduces a novel way of including the road geometry to a sequence to sequence prediction model. Instead of manually crafting features with the predicted locations of the vehicle, the road segment ahead of the vehicle is given. By embedding this information and using an attention mechanism,

the model is made fully responsible for extracting the right features with a minimal amount of information.

The road-attention model can be extended with social pooling. No extra information is embedded in the hidden state from the social pooling module. Therefore, no obfuscation of the interaction information takes place, when it is used to initialize the decoder. Instead, the road-RNN features are obtained at the decoder prediction time, by obtaining the context vector c_t from the attention module. By including the context vector in the gating mechanism of the GRU, the model can determine what information to include in the hidden state and subsequently use for the output.

3.4. Conclusion

To obtain more accurate predictions of the driving behavior, deep learning methods for road-geometry aware and interaction-aware prediction have to be improved and combined. First, a baseline deep learning model is constructed, that can generate predictions based on the past observed trajectory. The sequence to sequence prediction model is a popular approach to deep-learning based motion prediction. The model will be used to benchmark the performance, and show the compatibility of the proposed methods with existing approaches. To introduce the role of interaction in driving behavior, a social pooling module is introduced to the baseline model. The social-pooling allows to efficiently include the driving behavior of other vehicles in the scene, and is robust in handling redundant information.

To introduce reliable, map-based road information to the model, two methods are proposed. In the first, the predictions from a model are used to extract features in a map. These features describe the road scene around the predicted location, and are used to update these predictions. In the second method, the semantic map is only used to extract a road segment ahead of the vehicle. A road-RNN is introduced to construct features regarding the road segment, and an attention mechanism to determine what part of the road segment is relevant for the predictions.

Both road-geometry models include all information that is necessary for road-aware prediction. Furthermore, the models are extensions of the baseline model, and are fully compatible with the social pooling method. Either the interaction-aware predictions are refined to the road, or already include the road context vector from the attention mechanism. With these approaches, all facets are present to model interaction and road-aware driving behavior.

4

Experiments & Results

In the previous chapter, the proposed models for interaction and road-geometry aware motion prediction are detailed. In this chapter, the experiments in which these models are used will be described. First, the experimental setup will be highlighted, where two different datasets are used to assess performance of the models. This includes details regarding the model hyperparameters, the original data, modifications of the data and the constructed semantic map. Next, the experimental results will be examined of each model on the datasets, highlighting all noteworthy observations.

4.1. Experimental setup

For autonomous driving, more and more datasets are becoming available. These datasets are often recordings of sensory data, and differ in size and type of data that is recorded. To clearly assess the impact of the proposed methods on map-based motion prediction, a suitable dataset has to be chosen. Concretely, a number of demands for the dataset have been formulated to properly test the models.

Dataset size Deep learning is a machine learning technique, where a sufficient amount of data needs to be present. Training on a sufficient amount of diverse data prevents overfitting of the model. This implies that the model generalizes better, learning the actual important patterns in the observed data. For this reason, the dataset should be of significant size, to properly train the model.

Data type Many autonomous vehicle datasets focus on perception, providing extensive sets of visual information regarding the driving scene. For motion prediction, the dataset should contain tracklets, being the tracked states of vehicles in the scene over time. These can be given in global GPS coordinates, in a local reference frame or relative to an ego vehicle. Furthermore, the vehicles in the scene should be tracked for a sufficient amount of time. One advantage of trajectory prediction is that it is a semi-supervised learning task, meaning that the data does not have to be labeled. Rather, the recorded trajectories can be cut up in a part that is used as the observed trajectory and a part that is the desired output, the future trajectory. This requires a sequence that is sufficiently long to provide both the input and the ground truth.

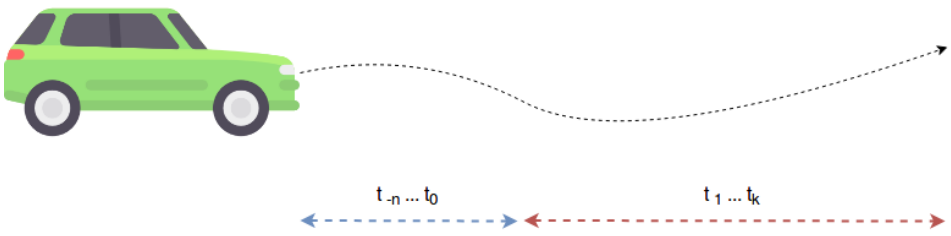


Figure 4.1: A vehicle recording, split up in the observed input trajectory and ground truth future trajectory.

Map information A semantic map of the scene has to be constructed. This means that map information of the area where the data is recorded is available. This could either be by directly supplying map features, or have some detailed map that can be used to craft a semantic map. Open source maps such as Open Street Maps do not deliver sufficiently accurate map information to extract features based on geolocations.

Environment isolation This thesis focuses on a subset of the challenges of motion prediction. The outline is to improve motion prediction by inclusion of interaction and road geometry in the predictions. To properly assess whether the predictions are actually improved, the dataset has to qualify regarding two aspects. First, it should be sufficiently challenging dataset with respect to the interactive and road geometric aspects. For interaction, traffic should be sufficiently dense to obtain a strong level of interaction in the driving behavior. To see whether the road-aware models can obtain good prediction performance regardless of the road scene, several scenarios with a challenging road layout should be available. Second, it is desirable to keep influences in the scene that are outside of the scope of this work limited. This includes traffic lights, pedestrians, obstacles on the road and more. By isolating a driving environment that contains only the factors that are modeled, the effects of the proposed methods can be best assessed.

4.1.1. NGSIM

The dataset that best meets the requirements is the Next Generation Simulation (NGSIM) dataset. NGSIM is a US government project from 2003-2006, where the traffic in 4 traffic scenarios is tracked over a fixed amount of time. With a set of cameras positioned next to the road (figure 4.3), vehicles in the scene are tracked over time. Shapefiles with accurate vectorized representations of the tracked scenes are given with the data.

Figure 4.2: A digital video camera mounted on top of a building that overlooks I-80 is recording vehicle trajectory data.

From the 4 possible sets, the NGSIM i80 dataset is chosen. The i80 is the interstate 80 freeway in the San Francisco Bay area in Emeryville, CA. There are several unique properties characteristics that make this dataset particularly suitable. Some characteristics are summarized in table 4.1. First of all, the NGSIM i80 dataset contains the tracking of vehicles in dense traffic. In the dataset, approximately 6000 vehicles are tracked in 45 minutes over a road of approximately 500 meters. The recording times are segmented in three 15 minute interval; 4.00 - 4.15 pm, 5.00 - 5.15 pm, 5.15 - 5.30 pm. These intervals represent three traffic density levels in the build up of traffic towards the peak hour (5.15 - 5.30 pm). Unlike the other datasets, the i80 does not contain any traffic lights or crossings in the driving scene. The recorded freeway contains 6 adjacent lanes and

an onramp where vehicles are obligated to merge into existing traffic. Interaction thus is thoroughly present in the dataset.

Table 4.1: i80 characteristics

Amount of vehicles	5567
Amount of lanes	7
Vehicle velocities	0 - 105 km/h
Average tracking time	90 s
Sampling frequency	10 Hz

The NGSIM datasets are particularly suitable for the creation of a semantic map. Often, the recordings are obtained by equipping a vehicle with sensors and driving through traffic. This makes the map creation laborious, because the entire driven route has to be mapped. In the NGSIM data, recordings are obtained by monitoring a fixed scene, limiting the size of the of the area that has to be mapped. Furthermore, the shapefiles of the scene can be used for the semantic map, because they are sufficiently accurate. With the shapefiles, the centerline of the lanes can be extracted and discretized into waypoints. In figure 4.3 the centerline waypoints are projected on a satellite image. One can observe that there is a small offset between the waypoint centerlines and image centerlines, due to inaccuracies in the satellite image locations. For this reason, it is important that accurate shapefiles are provided, and can not be extracted from existing public map data such as google maps or open street maps satellite images.

Figure 4.3: Waypoints defining adjacent lanes in the NGSIM i80 dataset, with a small offset due to satellite image location inaccuracy.

Concluding, we can say that the NGSIM i80 dataset qualifies on nearly all aspects. It is a dataset of considerable size, containing tracklets of vehicles in a global coordinate frame. The vehicles are tracked over a sufficient period of time, providing sequences that can be used as input data and ground truths. Furthermore, map information is present in the form of shape files of the driving scene, which can be used to create a semantic map. Moreover, the driving scene isolates the features of importance. No traffic lights or pedestrians are present in the scene, and the driving behavior is highly interactive.

4.1.2. Data inconsistencies

The trajectory recordings in the data contain errors. The original NGSIM data was quite noisy, and many filtering techniques have been proposed. The velocity and acceleration in the original dataset are obtained from derivations of the position data. This causes noise amplification, and in the original dataset accelerations of up to $400\frac{m}{s^2}$ are reported. The most recent work on NGSIM preprocessing is detailed in [33], where velocities are determined by using filters. Furthermore, trajectory inconsistencies are smoothed out by determining a curve over a larger set of trajectory points and replacing outliers with data points on the curve. The obtained NGSIM dataset contains these smoothed operations

However, in the trajectory data there remain some peculiar patterns. Often, trajectory states will make a jump that seems entirely unnatural, see figure 4.4. Since these jumps can occur both in the input as well as the ground truth, they can obstruct the models from learning proper trajectory shapes.

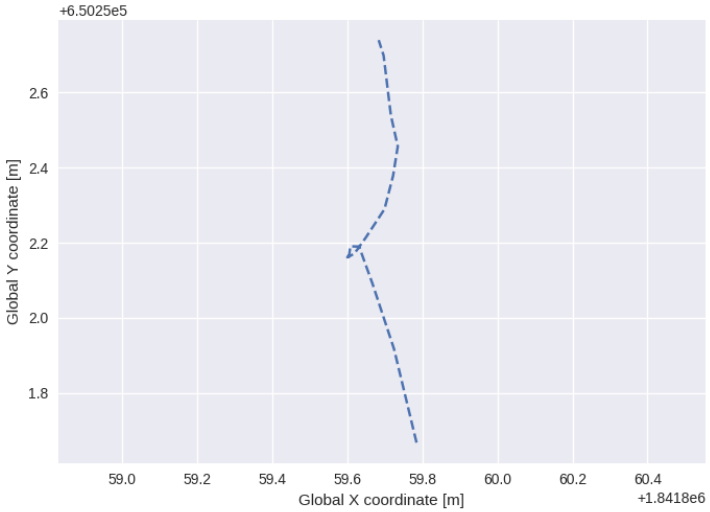


Figure 4.4: State jumps in the trajectory data.

To avoid sequences with these jumps, the following (new) data selection approach is taken. First, the directional vector u is obtained, describing the direction from one state to the next. In a single NGSIM trajectory, these will all approximately point in NW direction, along the highway. By computing the angle α between the unit vectors u and v of two states, a measure can be obtained for the change in direction

$$cos(\alpha) = \frac{u \cdot v}{\|u\| \cdot \|v\|}. \tag{4.1}$$

A large angle α indicates a large change of direction, where $\alpha = 0$ deg means no change, $\alpha = 90$ deg means a purely lateral translation, and $\alpha > 90$ deg means reverse driving. Naturally, any situation where $\alpha \geq 90$ can not occur. First of all, it can be assumed that vehicles are not driving in reverse on the highway. Moreover, lateral translations cannot occur die to nonholonomic constraints. Therefore, α is a good indicator of the noise in these trajectories.

To obtain sequences that are free of these jumps, all states with $\alpha \geq 20$ deg are flagged as bad. All trajectory sequences between two bad states that are of sufficient length can be used as proper data. The $\alpha \geq 20$ deg is obtained by plotting the shape of the trajectories and determining a threshold between natural and unnatural jumps. Since the sampling frequency is 10 Hz, α indicates the directional change in 0.1 second. Therefore, trajectories with a yaw rate of up to $20\frac{\text{deg}}{s}$ are still maintained in the dataset. Thus, in this way only noisy trajectories are avoided, while preserving trajectories that include a lane change.

4.1.3. Data augmentation

The NGSIM datasets have been augmented to make them more suitable for the focuses of this thesis. The main downside of these datasets is that they contain only limit road-geometric variations. Especially for the us101 and i80, which are highways that are quite straight and continuous. In figure 4.5, a subset of the trajectories on the i80 are plotted. Here, the lack of variation in road-geometry can be clearly observed. All trajectories are increasing in positive y and negative x direction. To make prediction more straightforward, the standard i80 dataset is mirrored over the y axis, making all coordinates positive increasing in x and y direction.

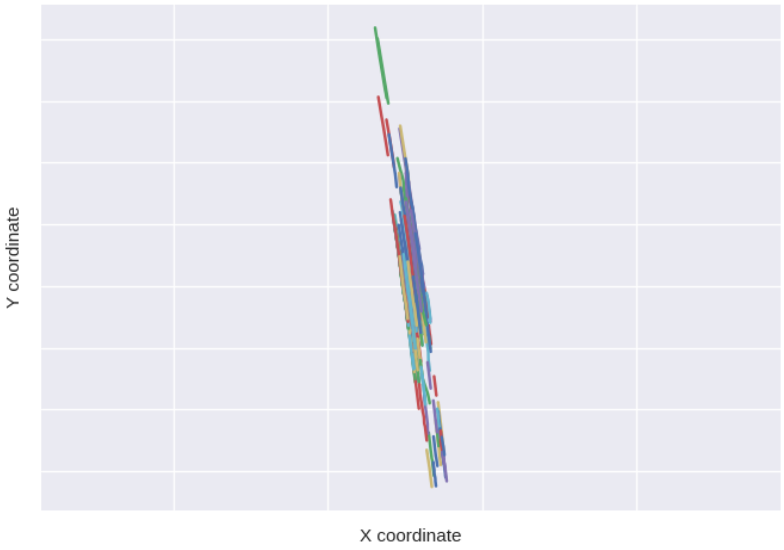


Figure 4.5: Some trajectories from the NGSIM i80 dataset, colorized per vehicle.

This i80 dataset clearly lacks variation in road geometry. A model that is trained on this dataset will not be required to take into account the road geometric variations, because there are none. To introduce more variety in the road geometry, the i80 dataset is augmented. By transforming both the recorded trajectories and the map waypoints, while preserving the relative distances, a new dataset can be obtained that better tests the models for their geometric-aware prediction capabilities. From here on, we will refer to this dataset as the i80 curved, or i80c. In this augmentation, multiple 'maps' will be created, where the road geometry is altered and the data is transformed to fit the new shape of the road. In the road geometries, a mix of roads with a gradually increased curve will be introduced, as well as very sudden turns and bends. Due to the sudden occurrence of the sharp turns, it will not be possible for a model to deduce the future shape of the road based on the past driven trajectory.

To augment the data while preserving the relative configurations of trajectories and map, the coordinates have to be redefined in a different coordinate system. For this purpose, the curvilinear coordinate system (CCS) is used. First, a curve is found that represents the original shape of the road. All coordinates can be defined with respect to this curve in the CCS. This road curve can be substituted by any other curve shape. When the coordinates are transformed back to the global coordinate system with respect to the new curve, the data will obtain the shape of the new curve.

First, all coordinates in the data have to be defined in the CCS. The road shape can be represented by a

parametric B-spline f defined as

$$f(t) = g(t)i + h(t)j \tag{4.2}$$
$$g(t) = A_x t^3 + B_x t^2 + C_x t + D_x \tag{4.3}$$
$$h(t) = A_y t^3 + B_y t^2 + C_y t + D_y, \tag{4.4}$$

where $t \in [0,1]$ is the variable that describes the spline from its origin at $t = 0$ to its end at $t = 1$, $g(t)$ and $h(t)$ describe the x and y coordinates as a function of t. The parameters of the curve are found by fitting it over the shape of the road. In this case, it is chosen to fit the curve over the left most centerline in the scene, defining all coordinates with respect to the centerline of the left most lane.

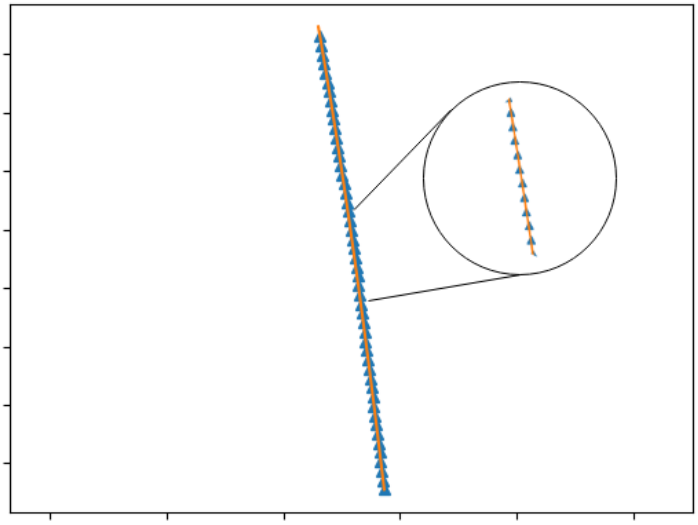


Figure 4.6: A spline curve, fitted over the centerline waypoints of the road.

In this case, the curve is represented by a nearly straight spline. With this original curve, all coordinates of the vehicle states and map waypoints can be defined in the CCS. This is done by defining every point as a certain longitudinal distance along the curve S, and lateral distance from the curve N. This means that every point A defined in carthesian coordinates X_a, Y_a, can also be defined as S_a, N_a with respect to some curve. For the CCS conversion of point A, first the closest point on the curve P_a has to be obtained. This is done by finding the parameter t, at which the least squares residual between point A and the spline is minimized

$$\underset{t}{\operatorname{argmin}} \|(g(t) - X_a)i + (h(t) - Y_a)j\|. \tag{4.5}$$

The obtained parameter t represents the point C_{p_a}, indicating a location on the spline that is closest to the point A. Because the point C_{p_a} is closest to A, it can be assumed that the line between A and P_a is orthogonal to the curve. The lateral coordinate N of the point is then simply the euclidean distance between A and P_a, given by

$$N = \|(Cp_x - A_x)i + (Cp_y - A_y)j\|\kappa, \tag{4.6}$$

Where κ is used to indicate on which side of the curve A is, similar to equations 3.25 and 3.26. Next, the length of the spline from its origin to point P_a has to be determined as the second curvilinear coordinate S_a. The arclength is defined as

$$L = \int_0^{s=1} dl, \tag{4.7}$$

where the change in arc length dl can be formulated as a function of the parametric coordinates as

$$dl = \sqrt{1+\left(\frac{dy}{dx}\right)^2}\,dx \tag{4.8}$$

$$= \sqrt{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2}\,dt \tag{4.9}$$

$$= \sqrt{(g'(t)^2+h'(t)^2}dt \tag{4.10}$$

$$= \sqrt{\left(3A_x t^2+2B_x t+C_x\right)^2+\left(3A_y t^2+2B_y t+C_y\right)^2}\,dt. \tag{4.11}$$

This form is notoriously challenging to analytically solve. Instead, the length can be obtained by numerical integration over equation 4.10. The obtained arclength S_a is the final step in the conversion of point A from $(X_a,\ Y_a) \rightarrow (S_a,\ N_a)$.

When all coordinates are defined in CCS, a new curve can be drawn to represent the new road shape. These road shapes are drawn by defining a sequence of points on which the curve has to be fitted. As said before, multiple road shapes have been defined with different levels of road geometric variety, ranging from slight turns to unnaturally sharp zigzags (figure 4.7).

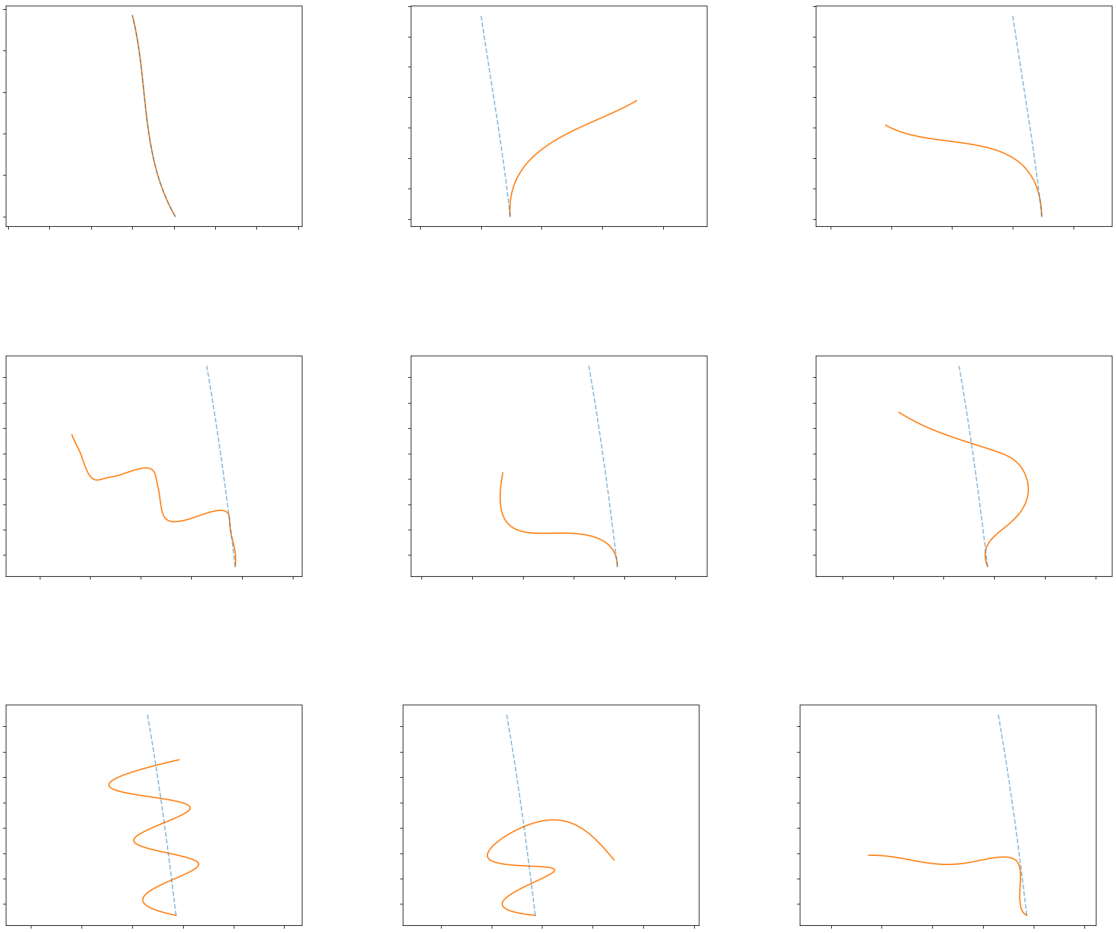


Figure 4.7: 9 curves representing 9 different road layouts represented in yellow, with the original road shape presented in blue. The top left curve shows the entire shape of the original road curve.

The road layouts represented in figure 4.7 differ from one another to a high degree, and may not all seem very truthful to road layouts that can be encountered in real traffic scenarios. The most important function of

these road layouts is to introduce sufficiently diverse road maps and turns. By introducing multiple different maps and multiple turns per map, the chance that the model is capable of memorizing the shape of all roads is greatly reduced. Not only are there 9 different maps in which the vehicle can be situated, the trajectory can start on any arbitrary segment of the map.

Furthermore, the sharp turns disallow the model to deduce the road shape ahead with only the observed trajectory. In a turn with approximately fixed radius, the future road layout can be deduced by looking at the curvature of the past trajectory. With a sudden turn to the left or right, this is impossible. This is an important aspect, because the road layout ahead of a vehicle in real traffic scenarios can not always be deduced with the observed trajectory either. For example, when a vehicle is approaching an intersection, it will drive on a straight road, giving no information about the road layout if the vehicle wants to turn right. Therefore, despite the artificial form of the road layouts, these maps will test the models on the road-aware prediction capabilities that are needed for satisfactory motion prediction.

The map and trajectories can now be transformed to match the road shapes above, by means of the CCS conversion explained earlier. For this road layout shaping, the inverse CCS transformation is to be obtained, mapping the CCS coordinates back to global cartesian coordinates with the new road layouts. That is, given a set of curvilinear coordinates S_a, N_a and a new curve C_{new}, we are looking for the cartesian coordinates X'_a, Y'_a. the new coordinates define the location of state or waypoint A along a new curve. Since $C_{original}$ and C_{new} are both parametrized with $t_a \in [0,1]$, but do not necessarily have the same length, the original t_a can not be used. Instead, the point P'_a on C_{new} has to be found where the length of the curve approximately matches S_a by

$$t'_a = \underset{t}{\operatorname{argmin}} \left| \int \sqrt{\frac{dx^2}{dt} + \frac{dy^2}{dt}}\, dt - S_a \right| \tag{4.12}$$

$$P'_a = f_{new}(t'_a). \tag{4.13}$$

This point is the projection of the desired transformation A' on C_{new}. To obtain A', we translate from the point P'_a with distance N_a in the direction orthogonal to the spline

$$u' = \left[\frac{\partial f}{\partial y}, -\frac{\partial f}{\partial x}\right] \frac{\kappa}{\left\| \left[\frac{\partial f}{\partial y}, -\frac{\partial f}{\partial x}\right] \right\|} \tag{4.14}$$

$$A' = P'_a + u' \cdot N_a. \tag{4.15}$$

With these steps, the data transformation is completed. With a 'forward' CCS step, all coordinates are defined along a curve that describes the original road shape. Then these coordinates are transformed back to cartesian coordinates with respect to the new road shapes, rather than the original one. An example of the original and resulting data is given in figure 4.8.

Error analysis

The data augmentation method mentioned above is a useful method to create multiple road-geometric scenarios from a single dataset. The augmentation method also introduces some error in the data, and these will be discussed here.

First and foremost, the new data is not entirely truthful to real-world data, since the velocity profiles do not match the turns and bends. In any real driving environment, drivers adapt their velocity in turns due to the centrifugal forces. This means that an upcoming turn can be a good indication of a vehicle decreasing its velocity. In this data, such patterns are not present, because the trajectories are recordings from a straight road. Therefore, no matter how sharp the turn is, the velocity profile will be independent of the road geometry.

Second, the transformation does deform the trajectories and centerlines to a degree. The trajectories are reshaped over a certain curve with a radius. The length of the projection of the trajectory on the curve will match the original length of the trajectory. However, the trajectories are most often a lateral distance from

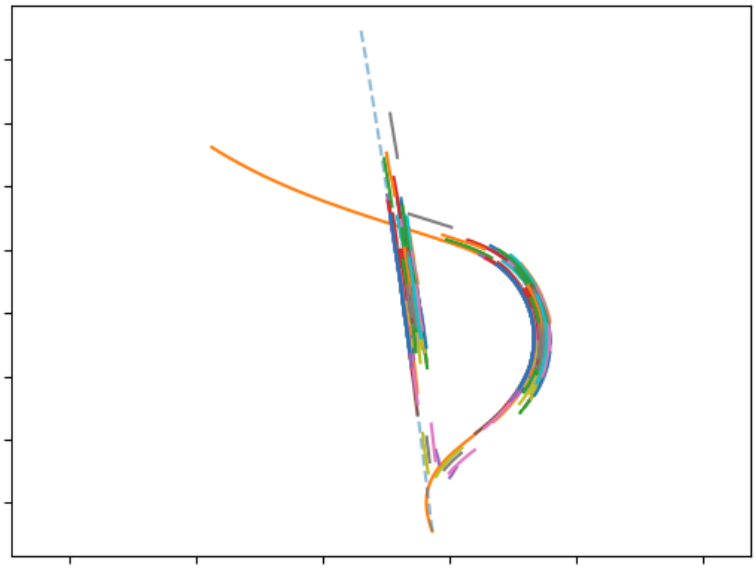


Figure 4.8: The original trajectories and road curve, and transformed trajectories along a newly drawn road curve.

the curve with which they are defined. Because the orthogonal distance from the point on the curve is taken to find the new location, the trajectories will be stretched if they are projected over a curve. This is shown in figure 4.9.

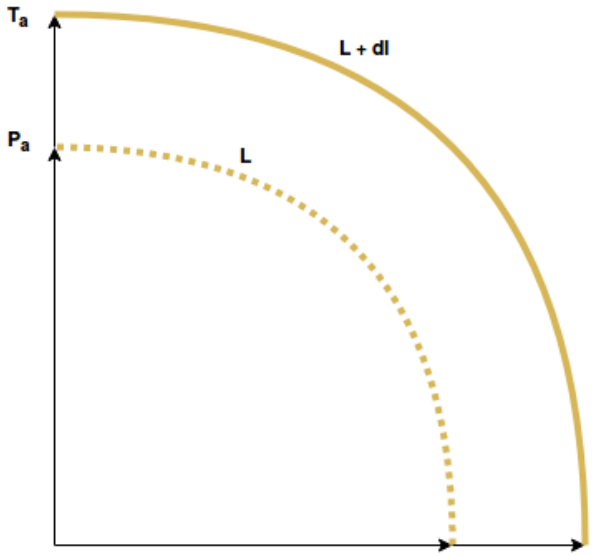


Figure 4.9: A trajectory T_a and its road centerline projection P_a.

In this figure, T_a is the transformed trajectory and P_a is the projection of T_a on the curve. The larger the lateral distance to the curve, the more the sequence gets stretched. For a curve with angle α and radius R, and a trajectory at a constant lateral distance N, the lengths of the trajectories are

$$dl = \alpha N. \tag{4.16}$$

Thus, the error scales with the orthogonal distance. This is disadvantageous because it shifts the relative position of vehicles on adjacent lanes. For interaction-aware motion prediction, where the relative position of the vehicle has to be indicated, this could have an impact on the performance.

Conclusion

Apart from the introduced errors that are mentioned before, the data augmentation is a useful solution to create data with more diverse road geometry. The selected trajectories in the i80c dataset will be evenly distributed among all maps, including the original map. Experiments will be conducted on both maps if the road geometric performance is of interest, to be able to compare results between models and between datasets.

4.1.4. Error metrics

In the experiments, the model performances are assessed on the i80 and i80c data. The models will obtain an input trajectory recording of 3 seconds (30 states), and predict over a prediction horizon of 6 seconds (60 states). The 6 second prediction horizon is needed for the autonomous vehicle to anticipate the driving behavior of other vehicles sufficiently well.

The first reported error metric will be the Mean Euclidean Distance (MED) between the predicted locations and the ground truth, in meters. The reported distance is the mean of the obtained distances of all N trajectories at the mentioned time step. At time t, the error is thus obtained as

$$\varepsilon_{l_2} = \frac{1}{N} \sum_{i=0}^{N} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}. \tag{4.17}$$

Additionally, the Mean Absolute Error (MAE) of the longitudinal and lateral predictions and ground truths will be given. These MAE_{lon} and MAE_{lat} are obtained by converting the predictions to the curvilinear coordinates: $(x,\ y) \rightarrow (s,\ n)$, and computing the mean absolute error between these prediction coordinates and the ground truth as

$$\varepsilon_{l_1-lon} = \frac{1}{N} \sum_{i=0}^{N} |s_i - \hat{s}_i| \tag{4.18}$$

$$\varepsilon_{l_1-lat} = \frac{1}{N} \sum_{i=0}^{N} |n_i - \hat{n}_i|. \tag{4.19}$$

Whereas the RMSE gives a general sense of the distance between the ground truth and predicted position of the vehicle, the MAEs dissect the error into a longitudinal and lateral component. Because the MAEs are obtained on curvilinear coordinates rather than linear coordinates they can not be directly compared to the RMSE, i.e. $\varepsilon_{l_2} \neq \sqrt{\varepsilon^2_{l_1-lon} + \varepsilon^2_{l_1-lat}}$.

4.1.5. Training details

Training of the model is done on 6000 trajectories that are randomly drawn from the datasets. These trajectories are recordings of 9 seconds, of which the first 3 seconds are used as input and the last 6 seconds as ground truth. The data is divided in a training set of 4500 trajectories and a test set of 1500 trajectories. The model is trained for 200 epochs with 16 trajectories per batch, amounting to a total of 56.2k iterations.

4.2. Experiment 1: Baseline model

In the first experiment, the model that is described in section 3.1 will be tested. This is an encoder-decoder model without any further extensions, solely using the past observed trajectories as input data. The encoder-decoder model is well known in literature, and its performance is used as a benchmark to compare other modeling performances to. As described before, the model performance will be reported as the Mean Euclidean Distance (MED) between the predictions and the ground truth at several prediction horizon times. Additionally, the longitudinal and lateral Mean Absolute Errors (MAE) will be reported. The modeling error will be reported for the normal i80 dataset, as well as the curved i80c dataset.

Table 4.2: Experiment 1 MED

Prediction Horizon [s]	Baseline i80 [m]	Baseline i80c [m]
1	0.75	1.11
2	1.55	2.11
3	2.50	3.41
4	3.70	5.13
5	5.12	7.06
6	6.59	9.36

The resulting performance is given in table 4.2. From this table, it can be observed that the error increases for a longer prediction horizon. Moreover, the prediction performance is worse for the i80c dataset than the i80 dataset. This can better be seen in figure 4.10, where the error on both datasets is represented over time.

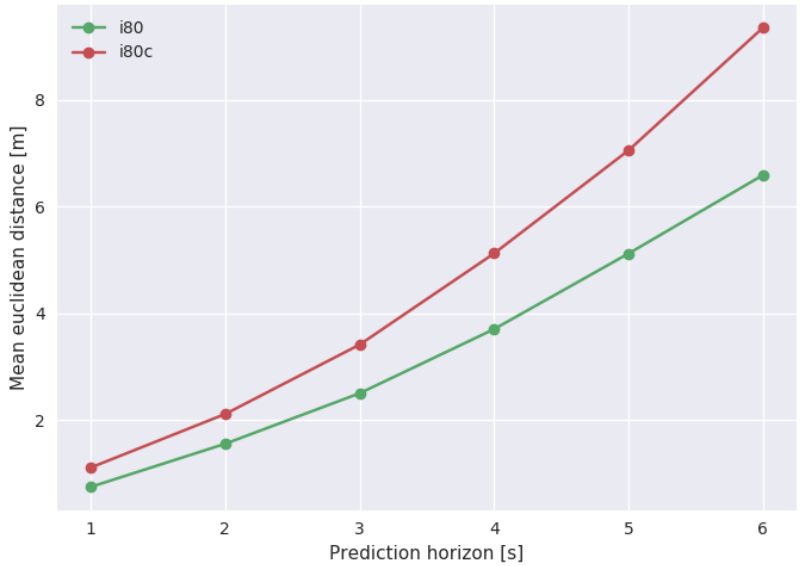


Figure 4.10: The baseline model prediction performance over the prediction horizon.

In addition to the RMSE, the predictions are converted to CCS to obtain an estimate of the longitudinal and lateral error. The results are given in tables 4.3 and 4.4, where the Mean Absolute Error (MAE) of the longitudinal and lateral coordinate predictions are given. In these tables, it can be seen that the error on the i80 dataset consists mainly of a longitudinal error, whereas the error on the i80c dataset is both in longitudinal and lateral direction.

Table 4.3: Experiment 1 Longitudinal MAE

Prediction Horizon [s]	Baseline i80 [m]	Baseline i80c [m]
1	0.71	0.84
2	1.50	1.74
3	2.44	2.82
4	3.63	4.11
5	5.04	5.67
6	6.51	7.60

Table 4.4: Experiment 1 Lateral MAE

Prediction Horizon [s]	Baseline i80 [m]	Baseline i80c [m]
1	0.13	0.56
2	0.23	0.96
3	0.32	1.37
4	0.42	1.98
5	0.49	2.77
6	0.56	3.69

In figure 4.11, some of the trajectories from the i80 test set are shown. It can be seen that all the vehicles traverse in a similar direction. The baseline model is able to accurately predict the lateral position of the vehicles. The predictions in 4.11a-b show that the main error is in longitudinal direction, which was also observed from tables 4.3 and 4.4. Lane changes that happen at later times in the prediction horizon remain challenging to predict for the model, as shown in 4.11c-d.

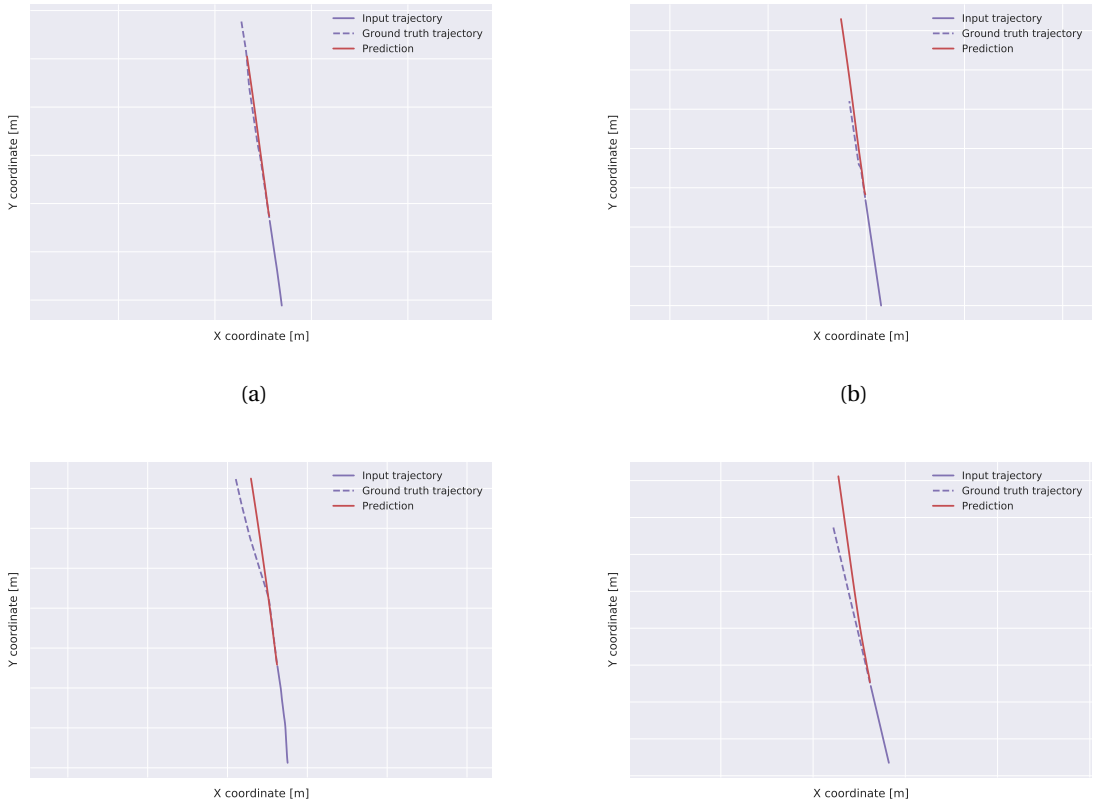


Figure 4.11: Predictions on the i80 dataset.

In figure 4.12, some of the trajectories from the i80c test set are shown. Here, the attempted strategy from the baseline model can be clearly seen. In 4.12a-b, two accurate predictions are shown. In these predictions, the model was able to deduce the road curvature from the past observed trajectory. This interpretation of road geometry is counterproductive in the predictions from 4.12c-d. The model prediction either misses the turn

completely (C), or memorized some road layout and tries to estimate what turn will come (D). This translates to the substantial lateral prediction errors from table 4.4.

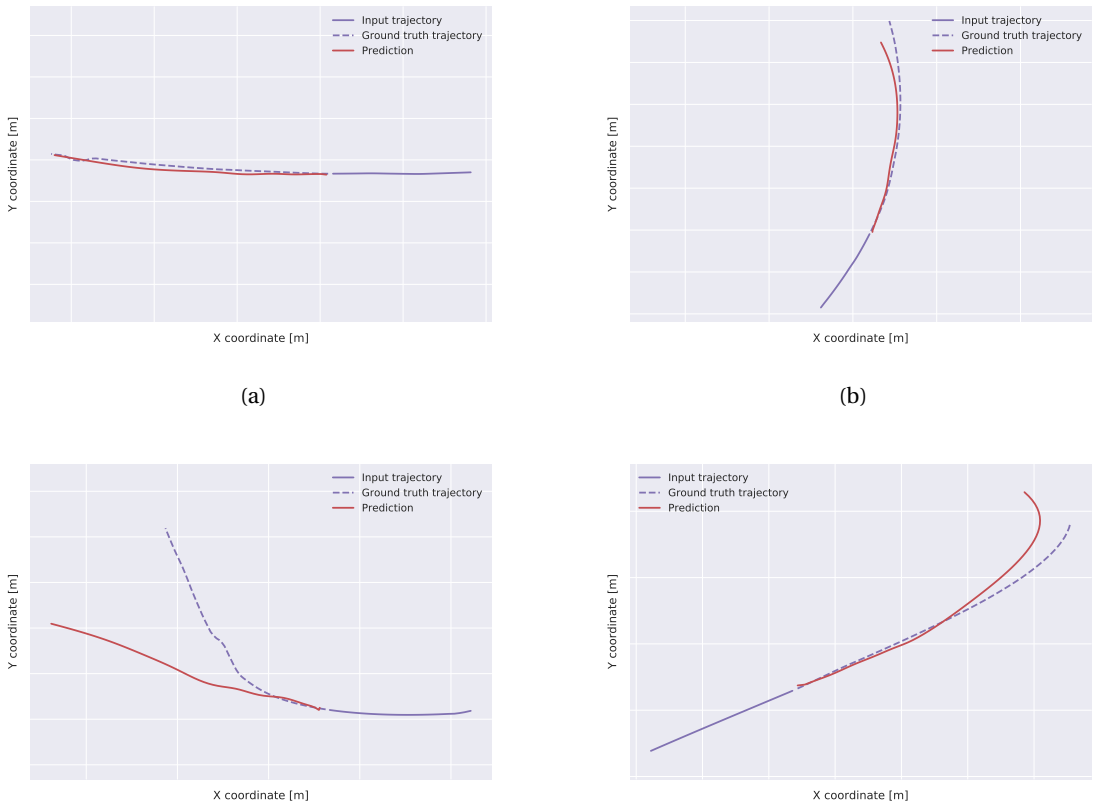


Figure 4.12: Predictions on the i80c dataset.

4.3. Experiment 2: Interaction

In the second experiment, the interaction module from section 3.2 will be added to the baseline model. A comparison can be made between the baseline model and the social pooling model. For every subject vehicle, the 5 most relevant vehicles around the subject vehicle will be taken into account. The resulting performance is given in table 4.5.

Table 4.5: Experiment 2

Prediction Horizon [s]	Baseline i80 [m]	Interaction i80 [m]	Baseline i80c [m]	Interaction i80c [m]
1	0.75	0.81	1.11	1.25
2	1.55	1.33	2.11	2.02
3	2.50	2.01	3.41	3.10
4	3.70	2.82	5.13	4.50
5	5.12	3.81	7.06	6.10
6	6.59	5.02	9.36	7.99

In this table, it can be seen that the interaction-aware model outperforms the baseline model on both datasets. Despite the improvement, there is still a major gap in prediction performance between both datasets, where predictions on the i80 dataset are considerably more accurate. The reported error in table 4.5 is plotted over the prediction horizon in figure 4.13.

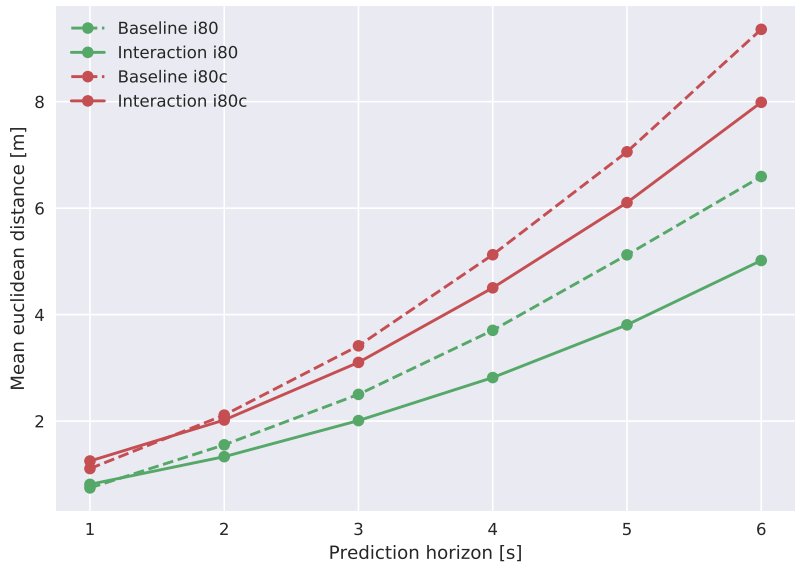


Figure 4.13: The model error of the baseline and interaction-aware model over time.

In tables 4.6 and 4.7, the longitudinal and lateral MAEs are given. It can be observed that on both datasets, the interaction-aware model increases the prediction performance mainly in longitudinal direction. The interaction module thus mainly increases estimation of the longitudinal movement of a vehicle along the centerline of the lane. Some lateral prediction performance increase is obtained on the i80c dataset.

Table 4.6: Experiment 2 Longitudinal MAE

Prediction Horizon [s]	Baseline i80 [m]	Interaction i80 [m]	Baseline i80c [m]	Interaction i80c [m]
1	0.71	0.88	0.84	1.00
2	1.50	1.47	1.74	1.73
3	2.44	2.17	2.82	2.68
4	3.63	3.09	4.11	3.77
5	5.04	4.13	5.67	5.06
6	6.51	5.32	7.60	6.64

Table 4.7: Experiment 2 Lateral MAE

Prediction Horizon [s]	Baseline i80 [m]	Interaction i80 [m]	Baseline i80c [m]	Interaction i80c [m]
1	0.13	0.17	0.56	0.57
2	0.23	0.24	0.96	0.80
3	0.32	0.30	1.37	1.16
4	0.42	0.35	1.98	1.67
5	0.49	0.41	2.77	2.24
6	0.56	0.48	3.69	3.06

In figure 4.14, some trajectories from the i80 test set are highlighted. These predictions confirm what can be concluded from the MAEs, namely that the interaction module realizes an increase in longitudinal performance. Due to the observation of surrounding vehicles and their driving behavior, the model can better estimate the driven trajectories along the centerline.

Figure 4.14: Predictions on the i80 dataset.

In figure 4.15, some trajectories from the i80c test set are highlighted. From table 4.5, it can be seen that the prediction accuracy increases compared to the baseline model on the i80c. However, the highlighted trajectories in figure 4.15 show similar prediction characteristics as the baseline model. For the more straight trajectories, accurate predictions are possible. For predictions that are near sharp turns, the road geometry is often anticipated by memorizing several road layouts.

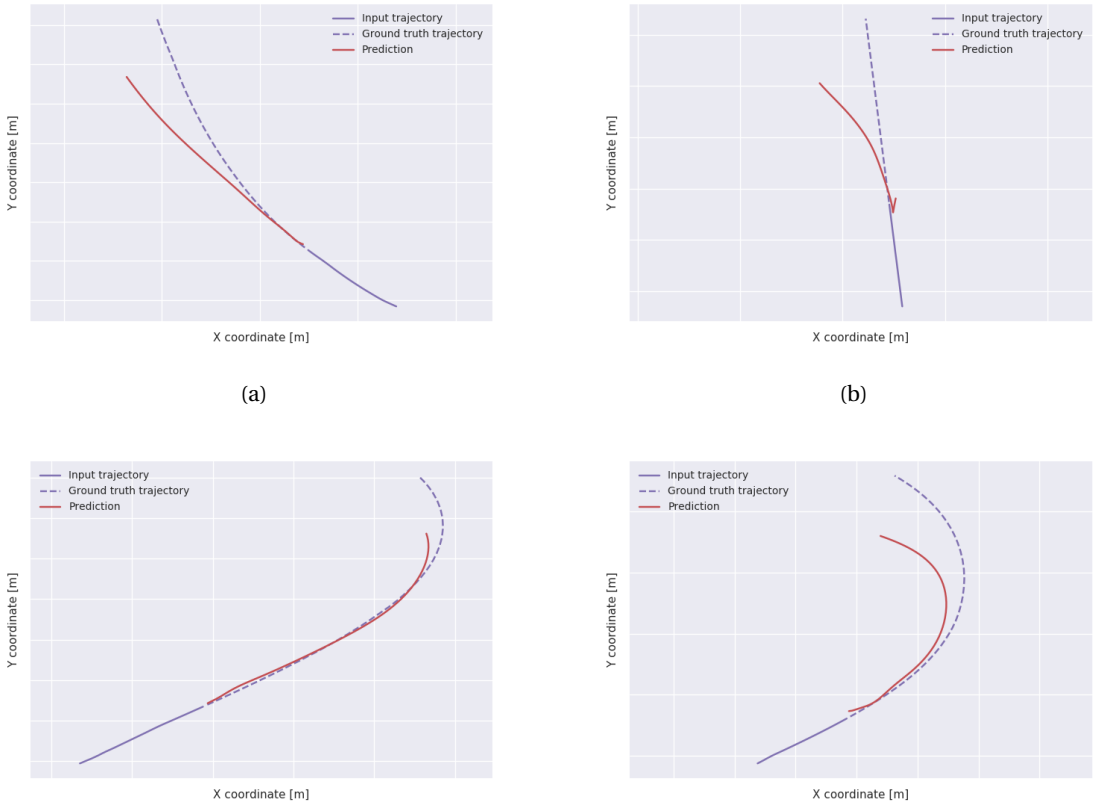


Figure 4.15: Predictions on the i80c dataset.

4.4. Experiment 3: road-refinement

In the third experiment, the baseline model will be extended with the road-refinement module, described in 3.3.2. The model uses the normalized Distance To Centerline, Active Lane Point and Global Lane Position features to determine where the road is located. The resulting performance is given in table 4.8.

Table 4.8: Experiment 3

Prediction Horizon [s]	Baseline i80 [m]	Refinement i80 [m]	Baseline i80c [m]	Refinement i80c [m]
1	0.75	1.16	1.11	1.35
2	1.55	2.05	2.11	2.17
3	2.50	2.93	3.41	3.36
4	3.70	4.09	5.13	4.69
5	5.12	5.57	7.06	6.29
6	6.59	7.10	9.36	8.20

The reported error in table 4.8 is plotted over the prediction horizon in figure 4.16. It can be seen that performance on the i80c dataset is slightly lower than on the i80 dataset. This model performs better than the baseline model on the i80c dataset, but worse on the i80 dataset.

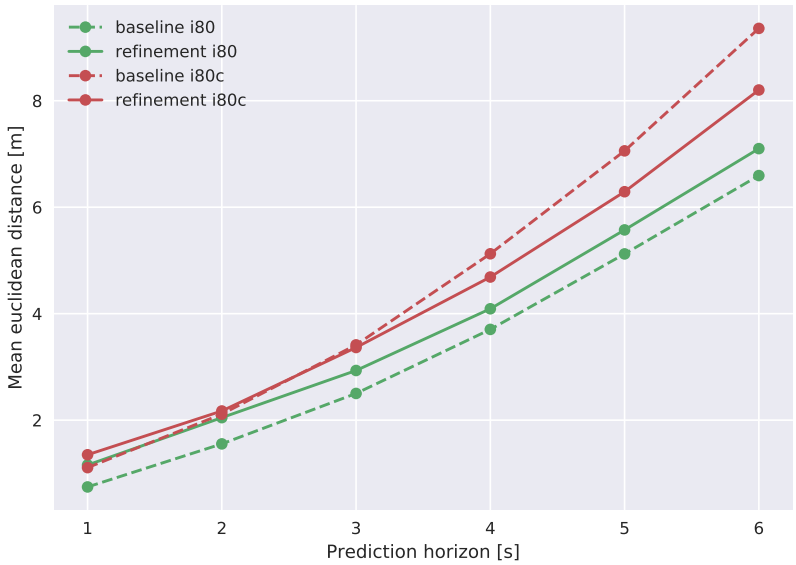


Figure 4.16: The model error of the baseline and interaction-aware model over time.

In the tables 4.9 and 4.10, the longitudinal and lateral errors are given. There, it can be seen that the refinement model obtains its increase in performance on the i80c model, mainly caused by a decreased lateral error. On the i80 dataset, prediction performance gets slightly worse over both the longitudinal and lateral errors.

In figure 4.17, some trajectories from the i80c test set are highlighted. It can be observed that the predictions all show some discontinuity. The model has some road-aware prediction capabilities, that can be seen in predictions 4.17b-c, where the ground truth shapes are followed. In other predictions such as in 4.17d, the model fails entirely to grasp the road geometry.

Table 4.9: Experiment 3 Longitudinal MAE

Prediction Horizon [s]	Baseline i80 [m]	Refinement i80 [m]	Baseline i80c [m]	Refinement i80c [m]
1	0.71	1.11	0.84	1.04
2	1.50	1.98	1.74	1.86
3	2.44	2.84	2.82	2.89
4	3.63	4.00	4.11	3.97
5	5.04	5.49	5.67	5.37
6	6.51	7.02	7.60	7.21

Table 4.10: Experiment 3 Lateral MAE

Prediction Horizon [s]	Baseline i80 [m]	Refinement i80 [m]	Baseline i80c [m]	Refinement i80c [m]
1	0.13	0.20	0.56	0.68
2	0.23	0.30	0.96	0.87
3	0.32	0.39	1.37	1.17
4	0.42	0.46	1.98	1.49
5	0.49	0.52	2.77	1.92
6	0.56	0.58	3.69	2.34

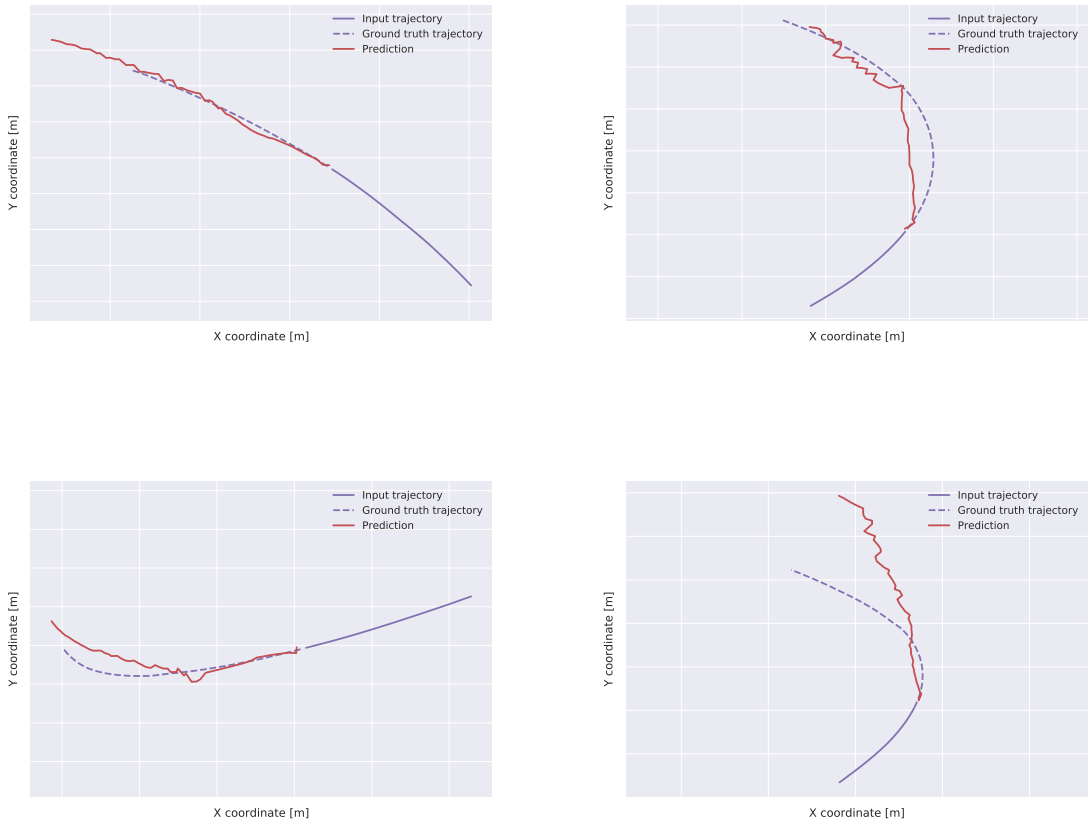


Figure 4.17: Predictions on the i80c dataset

4.5. Experiment 4: road-attention

In the fourth experiment, the baseline model will be extended with the road-attention module, described in section 3.3.3. The model uses an attention mechanism and road RNN to extract relevant road information ahead of the vehicle. Because this model is road-geometry aware, the model will be used on both the i80 and i80c dataset. The resulting performance is given in table 4.11.

Table 4.11: Experiment 4

Prediction Horizon [s]	Baseline i80 [m]	Attention i80 [m]	Baseline i80c [m]	Attention i80c [m]
1	0.75	0.73	1.11	1.18
2	1.55	1.38	2.11	1.82
3	2.50	2.36	3.41	2.67
4	3.70	3.45	5.13	3.80
5	5.12	4.70	7.06	5.03
6	6.59	6.16	9.36	6.52

The reported error in table 4.11 is plotted over the prediction horizon in figure 4.18. It can be seen that the road-attention model outperforms the baseline model on both datasets. What's more, the prediction performance of the road-attention model is nearly equal on both datasets, due to a sharp increase in prediction performance on the i80c dataset. The attention model outperforms the i80 baseline model performance on both datasets over the 6 second prediction horizon.

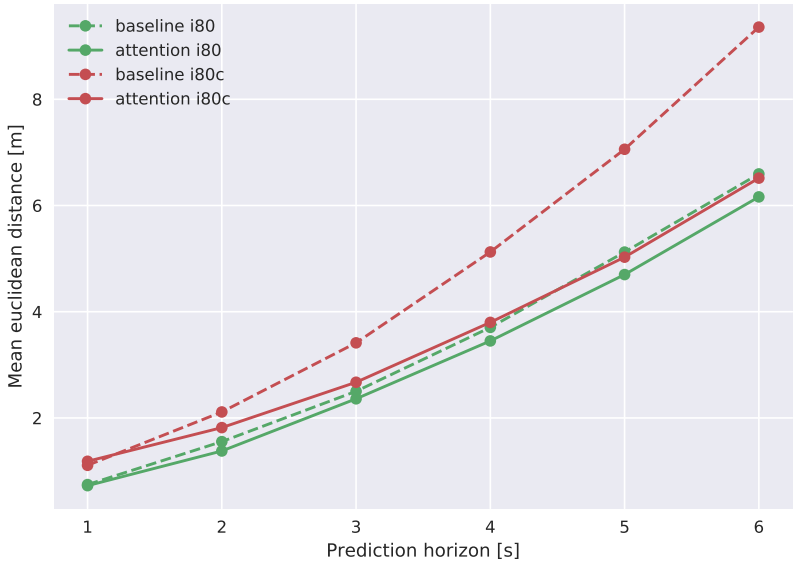


Figure 4.18: The model error of the baseline and interaction-aware model over time.

In the tables 4.12 and 4.13, the longitudinal and lateral errors are given. There, it can be seen that the road-attention model performance increase is especially large in lateral direction, due to a better understanding of the road geometry.

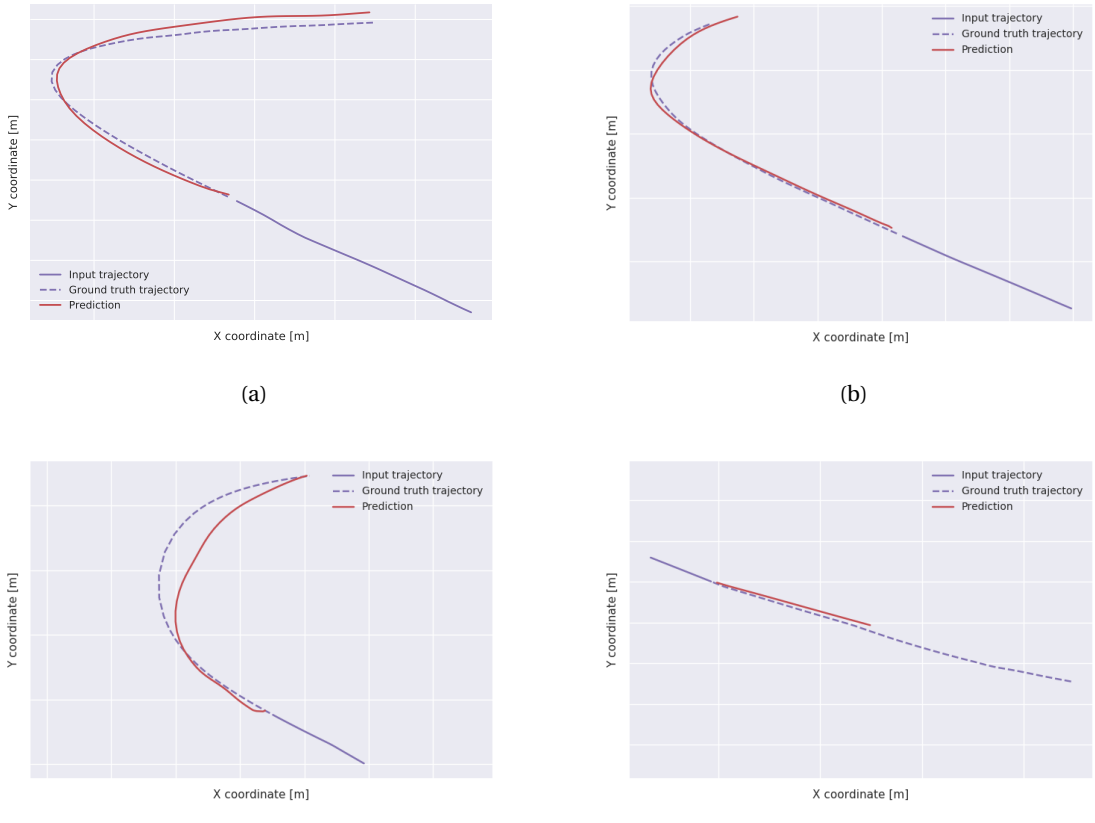
In figure 4.19, some trajectories from the i80c test set are highlighted. The predictions show that the model is capable of interpreting the road geometry for prediction. The trajectories in 4.19a-b show accurate predictions over sharp turns. In figure 4.19d, it can be seen that the model occasionally struggles with longitudinal prediction, as observed in the baseline model predictions in figure 4.11b.

Table 4.12: Experiment 4 Longitudinal MAE

Prediction Horizon [s]	Baseline i80 [m]	Attention i80 [m]	Baseline i80c [m]	Attention i80c [m]
1	0.71	0.69	0.84	0.83
2	1.50	1.33	1.74	1.54
3	2.44	2.31	2.82	2.51
4	3.63	3.40	4.11	3.55
5	5.04	4.65	5.67	4.88
6	6.51	6.11	7.60	6.32

Table 4.13: Experiment 4 Lateral MAE

Prediction Horizon [s]	Baseline i80 [m]	Attention i80 [m]	Baseline i80c [m]	Attention i80c [m]
1	0.13	0.14	0.56	0.51
2	0.23	0.20	0.96	0.58
3	0.32	0.27	1.37	0.69
4	0.42	0.34	1.98	0.76
5	0.49	0.40	2.77	0.87
6	0.56	0.47	3.69	1.04



(a) (b)

(c) (d)

Figure 4.19: Predictions on the i80c dataset.

4.6. Experiment 5: combined RNN

In the final experiment, the interaction-aware model and the road-attention model are combined. This model is assumed to understand the interactive forces in the scene, as well as the road geometry that lies ahead of the subject vehicle. Because this model is road-geometry aware, the model will be used on both the i80 and i80c dataset. The resulting performance is given in table 4.14.

Table 4.14: Experiment 5a: attention-combination

Prediction Horizon [s]	Baseline i80 [m]	Combined i80 [m]	Baseline i80c [m]	Combined i80c [m]
1	0.75	0.77	1.11	1.34
2	1.55	1.47	2.11	1.85
3	2.50	2.25	3.41	2.54
4	3.70	3.12	5.13	3.44
5	5.12	4.17	7.06	4.53
6	6.59	5.42	9.36	5.82

The reported error in table 4.14 is plotted over the prediction horizon in figure 4.20. Again, the prediction performance on the i80 dataset is slightly higher than on the i80c dataset. As with the road-attention model, the combined model outperforms the baseline model on both datasets.

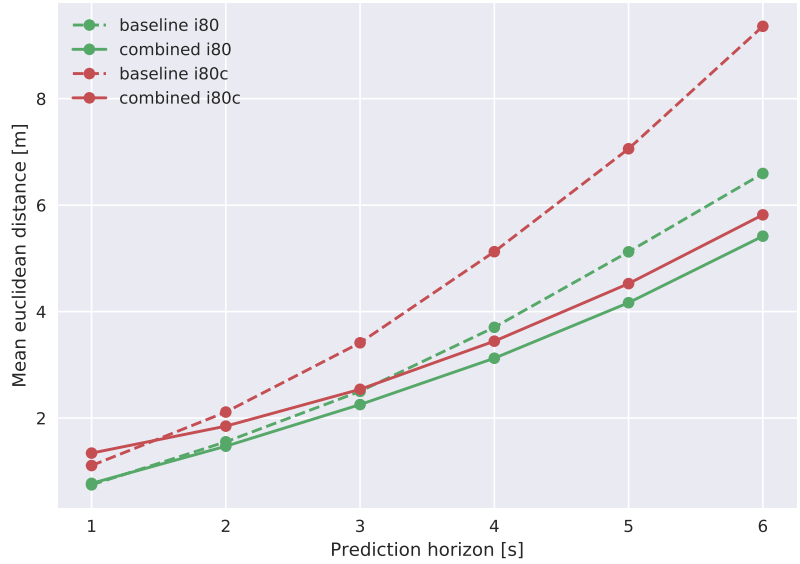


Figure 4.20: The model error of the baseline and interaction-aware model over time.

In tables 4.15 and 4.16, the longitudinal and lateral MAEs are given. It can be seen that on the longitudinal errors, the combined model strongly outperforms the baseline model on both datasets. On the i80c, the lateral prediction performance is also increased considerably.

In figure 4.21, some trajectories from the i80c test set are highlighted. It can be seen that all highlighted trajectories conform to the shape of the road. Regardless of the road curvature, the predictions keep following the ground truth direction. In figure 4.21b, the prediction starts with an offset, but follows the shape of the ground truth. This is reflected in the strongly reduced lateral error presented in table 4.16. What's more, the predictions contain a higher longitudinal accuracy compared to the predictions from experiment 4.5.

Table 4.15: Experiment 5 Longitudinal MAE

Prediction Horizon [s]	Baseline i80 [m]	Combined i80 [m]	Baseline i80c [m]	Combined i80c [m]
1	0.71	0.72	0.84	0.95
2	1.50	1.41	1.74	1.55
3	2.44	2.20	2.82	2.27
4	3.63	3.06	4.11	3.10
5	5.04	4.10	5.67	4.21
6	6.51	5.34	7.60	5.43

Table 4.16: Experiment 5 Lateral MAE

Prediction Horizon [s]	Baseline i80 [m]	Combined i80 [m]	Baseline i80c [m]	Combined i80c [m]
1	0.13	0.18	0.56	0.80
2	0.23	0.25	0.96	0.81
3	0.32	0.30	1.37	0.83
4	0.42	0.35	1.98	0.89
5	0.49	0.41	2.77	0.99
6	0.56	0.48	3.69	1.23

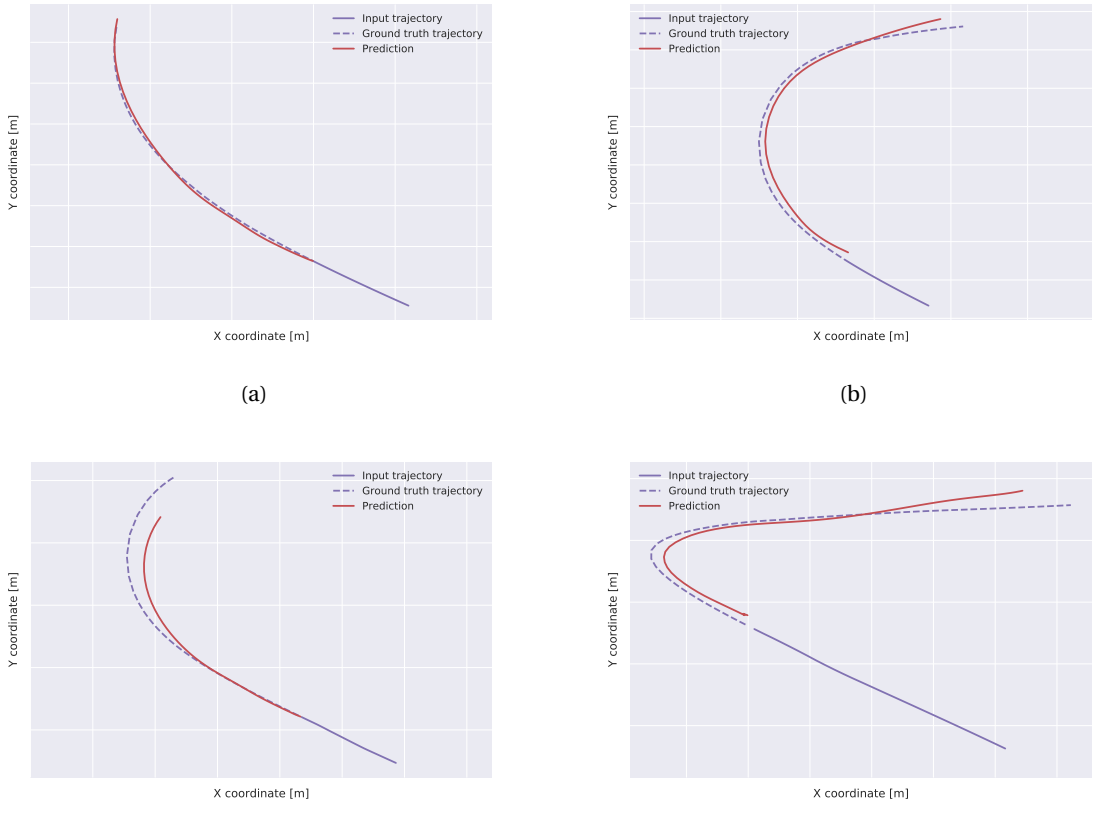


Figure 4.21: Predictions on the i80c dataset.

4.7. Overview of results

In this section, an overview of the results is given. The 5 experiments that are detailed show the results of different models on the i80 dataset. These results will be shown here to accomodate comparisons. The performances of the models on the i80 and i80c datasets are summarized in tables 4.17 and 4.18.

In table 4.17, it can be seen that long term performance on the i80 dataset is best for interaction-aware models from the experiments in 4.3 and 4.6. The road-geometry aware models show little improvement over the baseline model, with the road-attention model being slightly more accurate than the baseline model and the road-refinement model slightly less accurate. The model performances are plotted in figure 4.22 for better comparison.

Table 4.17: Summarized results on i80

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	0.75	0.81	1.16	0.73	0.77
2	1.55	1.33	2.05	1.38	1.47
3	2.50	2.01	2.93	2.36	2.25
4	3.70	2.82	4.09	3.45	3.12
5	5.12	3.81	5.57	4.70	4.17
6	6.59	5.02	7.10	6.16	5.42

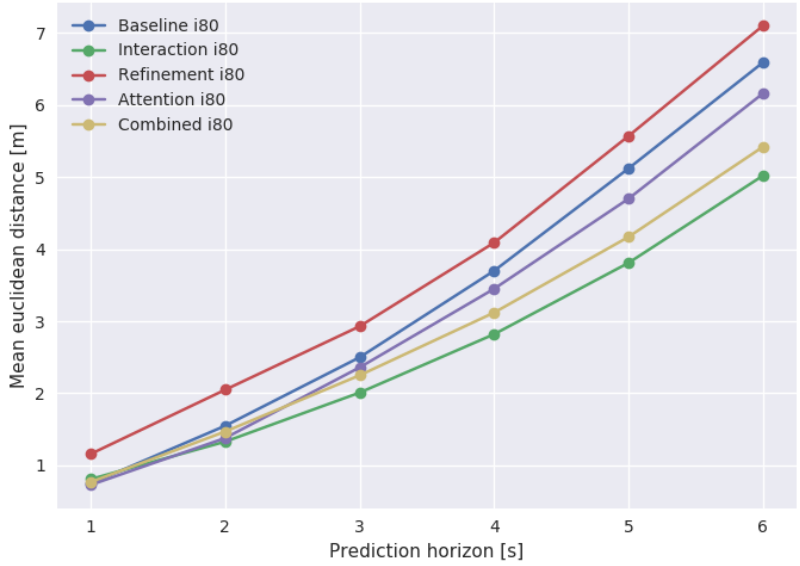


Figure 4.22: The prediction performances on the i80 dataset.

in table 4.18, the road-geometry aware models show an increase in performance compared to the baseline. The baseline model performs poor on the i80c dataset compared to its performance on the i80 dataset. Especially the road-attention model shows a significant increase in performance compared to the baseline. The highest accuracy is obtained with the road and interaction-aware model.

Table 4.18: Summarized results on i80c

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	1.11	1.25	1.35	1.18	1.34
2	2.11	2.02	2.17	1.82	1.85
3	3.41	3.10	3.36	2.67	2.54
4	5.13	4.50	4.69	3.80	3.44
5	7.06	6.10	6.29	5.03	4.52
6	9.36	7.99	8.20	6.52	5.82

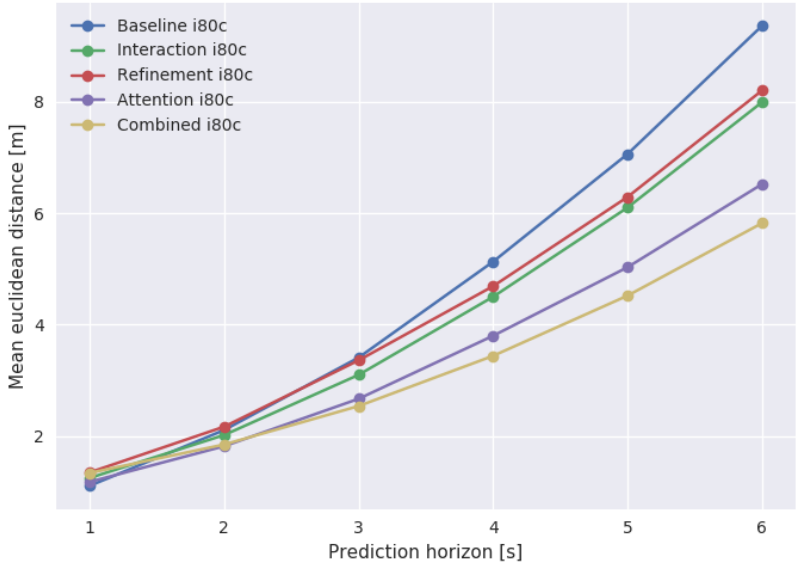


Figure 4.23: The prediction performances on the i80c dataset.

The longitudinal and lateral errors are given for the i80 dataset in tables 4.19 and 4.20 below. Here. it can be seen that the interaction-aware model and the combined model perform best on the i80 dataset. The lateral error remains quite constant over all the models, showing the capability of all models to estimate the lateral position on the i80 dataset with relative ease. The longitudinal error gets greatly improved, especially by introducing social pooling to the baseline model.

Table 4.19: Summarized longitudinal results on i80

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	0.71	0.88	1.11	0.69	0.72
2	1.50	1.47	1.98	1.33	1.41
3	2.44	2.17	2.84	2.31	2.20
4	3.63	3.09	4.00	3.40	3.06
5	5.04	4.13	5.49	4.65	4.10
6	6.51	5.32	7.02	6.11	5.34

Table 4.20: Summarized lateral results on i80

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	0.13	0.17	0.20	0.14	0.18
2	0.23	0.24	0.30	0.20	0.25
3	0.32	0.30	0.39	0.27	0.30
4	0.42	0.35	0.46	0.34	0.35
5	0.49	0.41	0.52	0.40	0.41
6	0.56	0.48	0.58	0.47	0.48

Finally, the longitudinal and lateral errors are given for the i80c dataset in tables 4.21 to 4.22 below. Here, it can be observed that both in longitudinal and lateral direction, the baseline model obtains large errors. A longitudinal improvement can be observed both by adding the interaction as well as adding road attention to the model. The combined model strongly outperforms all other models in this sense. For the lateral error, the most important addition is the road attention model, reducing the error by more than a factor 3 compared to the baseline and interaction model.

Table 4.21: Summarized longitudinal results on i80c

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	0.84	1.00	1.04	0.83	0.95
2	1,74	1.73	1.86	1.54	1.55
3	2.82	2.68	2.89	2.51	2.27
4	4.11	3.77	3.97	3.55	3.10
5	5.67	5.06	5.37	4.88	4.21
6	7.60	6.64	7.21	6.32	5.43

Table 4.22: Summarized lateral results on i80c

Prediction Horizon [s]	Baseline [m]	Interaction [m]	Refinement [m]	Attention [m]	Combined [m]
1	0.56	0.57	0.68	0.51	0.80
2	0.96	0.80	0.87	0.58	0.81
3	1.37	1.16	1.17	0.69	0.83
4	1.98	1.67	1.49	0.76	0.89
5	2.77	2.24	1.92	0.87	0.99
6	3.69	3.06	2.34	1.04	1.23

5

Discussion & Conclusion

The aim of this thesis is to improve the prediction of driving behavior of vehicles in a driving scene. To overcome current limitations, the objective is to include features regarding the road and the neighbouring vehicles in the prediction model. The performance of the resulting method has been observed in chapter 4. These results will be discussed and used to conclude the effectiveness of the taken approach, whereafter future recommendations will be given.

5.1. Discussion

To take into account the interaction between vehicles and road geometry, additions to existing modeling techniques have been proposed. These modules can extend existing deep learning sequence to sequence prediction models, to make these road-geometry or interaction-aware. For this purpose, one recently proposed interaction module has been constructed and two novel road-geometry modules have been proposed. These road-geometry modules use information from a semantic map, allowing more reliable features to be exploited for the road-aware prediction.

To validate the addition of road-geometric or interaction awareness, 5 distinct models have been constructed. These models all have different levels of this awareness, containing different combinations of the before mentioned modules. This allows to assess the importance and effectiveness of the modeling approaches. All models have been trained and tested on two different challenging datasets, one containing no variation in road geometry (i80) and another containing a strong variation in road geometry (i80c).

Baseline model

The baseline model is a sequence to sequence RNN model that generates predictions based on the past observed trajectory. It obtains no information regarding the road geometry or vehicles that are in the vicinity of the subject vehicle. The results of the baseline model on the i80 dataset show that it can reach a decent level of prediction performance. over a time horizon of 6 seconds, the Mean Euclidean Distance (MED) between predictions and ground truth trajectories is 6.59 meters. The model does not need to take into account any road-geometrical variations. The main inaccuracies lie in longitudinal direction, as shown in table 4.3, showing the challenge of correctly estimating the longitudinal trajectory of the subject vehicle with only a past observed trajectory. This is reflected well in some of the predictions in figure 5.1. The prediction overshoots or undershoots in a longitudinal sense, but keep a minimal lateral deviation from the ground truth.

On the i80c data, the results are considerably less. The MED for a 6 second prediction horizon is close to 50% higher than on the i80 dataset. The reason for this can be clearly observed in the highlighted trajectories (figure 5.2). The model is unaware of any road geometry in the scene. The road geometry causes a variety of rapid changes in direction in the trajectories that cannot be anticipated by the model. The baseline model attempts to memorize some shapes and deduce from the past observed trajectory what lies ahead. However, this model has no information as how far the subject vehicle is from a turn, or what the shape of this turn would be. Therefore, some guesses of memorized trajectory predictions are given based on the past observed

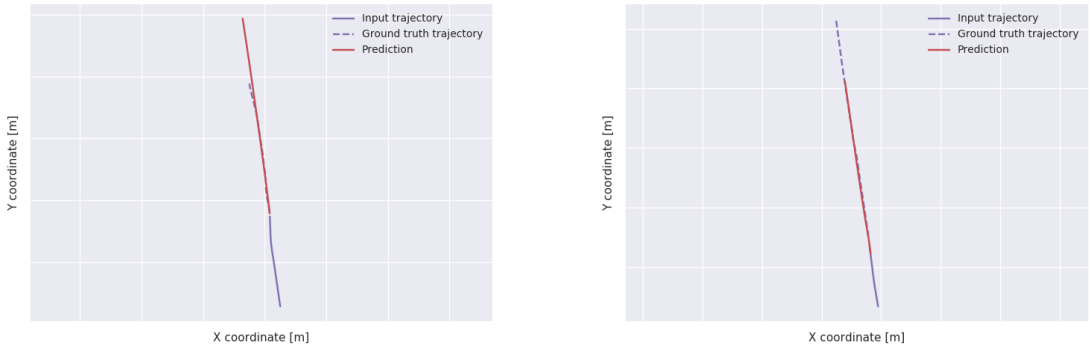


Figure 5.1: Predictions on the i80 dataset. Left: A prediction overshoot in longitudinal direction. Right: A prediction undershoot.

trajectories. The inability to comprehend the road geometry is mainly reflected in an increased lateral deviation, show in table 4.4, where the average lateral error after 6 seconds is substantial compared to the one on the i80 dataset.

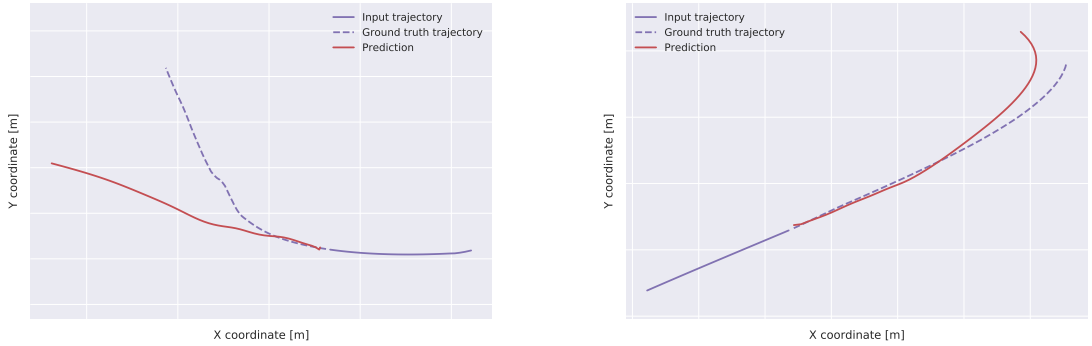


Figure 5.2: Predictions on the i80c dataset. Left: The model completely misses the road geometry. Right: the model memorized a road layout and uses it for prediction.

What is interesting is that the short-term prediction performance also significantly decreases on the i80c dataset. It could be expected that for a prediction horizon of 1 second, most of the future trajectories could still be forecasted as a continuation of the past trajectory. However, the model obtains decreased performance from the start of the prediction, showing the contrary.

Interaction

The interaction-aware model uses a technique called social pooling, extending the baseline model to become aware of the influence of vehicles in the vicinity. The impact of this extension to the baseline model can be clearly seen in the results. Compared to the baseline model, the long-term performance significantly increases to a MED of 5.02 meter for a 6 second prediction horizon on the i80 dataset. From the longitudinal and lateral errors, it can be seen that this increase is obtained in longitudinal direction (table 5.1). The baseline model assumes the road geometry to be constant in the i80 and predicts all the trajectories in identical direction. The velocity along the centerline is more challenging to guess, and adding the interaction module adds a boost to the longitudinal prediction accuracy.

In figure 5.3, one of the overshot predictions of the baseline model in figure 5.1 is compared to the prediction of the interaction model on the identical input trajectory. As can be seen, the interaction model achieves superior prediction accuracy in longitudinal direction due to its comprehension of interactive forces in play.

Table 5.1: Longitudinal performance on the i80 dataset

Prediction Horizon [s]	Baseline i80 [m]	Interaction i80 [m]
1	0.71	0.88
2	1.50	1.47
3	2.44	2.17
4	3.63	3.09
5	5.04	4.13
6	6.51	5.32

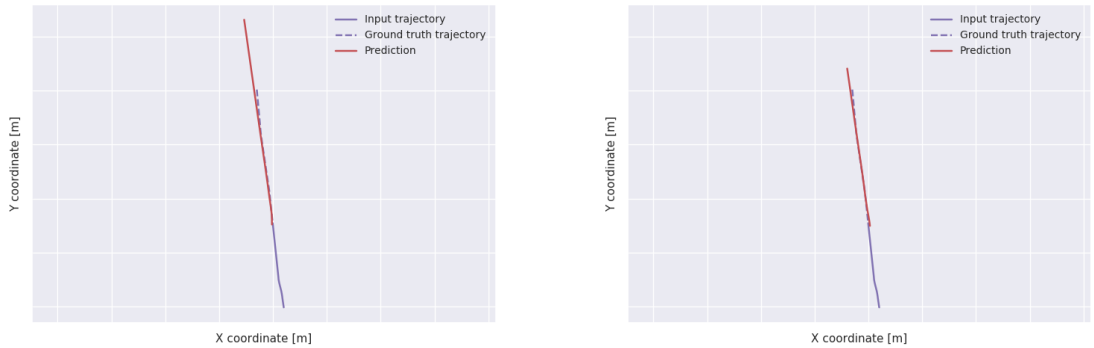


Figure 5.3: Predictions of the same vehicle on the i80 dataset. Left: the baseline model prediction with an overshoot. Right: The interaction-aware model with a more accurate prediction.

The performance of the model drops on the i80c dataset, similar to the baseline model. In figure 5.4 some predictions clearly show that the model still has no comprehension of the road geometry and thereby cannot estimate the future trajectory. The predictions show some similar characteristics to that of the baseline model, where the model tries to memorize the road shapes and often estimates it falsely.

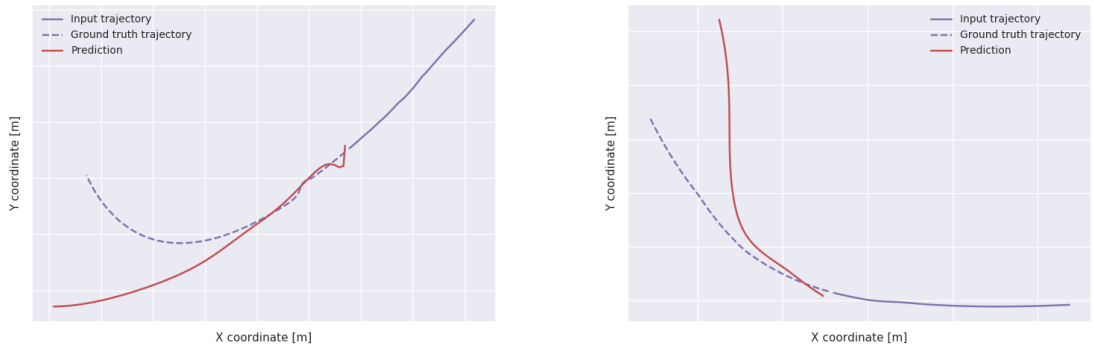


Figure 5.4: Predictions of the interaction-aware model on the i80c dataset. Left: the model prediction misses the turn. Right: the model memorized a road shape and gambles.

However, some performance increase is observed compared to the baseline model. This can be explained by the fact that the trajectories of vehicles ahead of the subject vehicle reveal some information about the shape of the road. This is shown in figure 5.5, where the input data to the model of the subject vehicle and

preceding vehicle are shown. The preceding input data clearly indicates the shape of the road ahead of the subject vehicle. As such, the recordings of the vehicles in the vicinity indirectly include some road-geometric information in prediction. However, this is not a sufficiently reliable source to obtain decent predictions on the i80c dataset.

Figure 5.5: Indications of the road geometry by means of the preceding vehicle.

Road-geometry

To include the road geometry, two methods have been identified in literature and subsequently incorporated in the models. In one approach, a set of road features is found with the predictions to refine these same predictions. In the other approach, the shape of an entire road segment is communicated to the model prior to prediction, making the model responsible for extracting the relevant information.

Both road-geometry models lead to an increase in accuracy on the i80c dataset, compared to the baseline model. From the predictions it can be seen that the models are aware of the layout of the road and can take it into account in their predictions to some degree. In this sense, the road-attention approach strongly outperforms the road-refinement approach on both datasets.This becomes more clear when comparing the MED error over the prediction horizon of both approaches (figure 5.6).

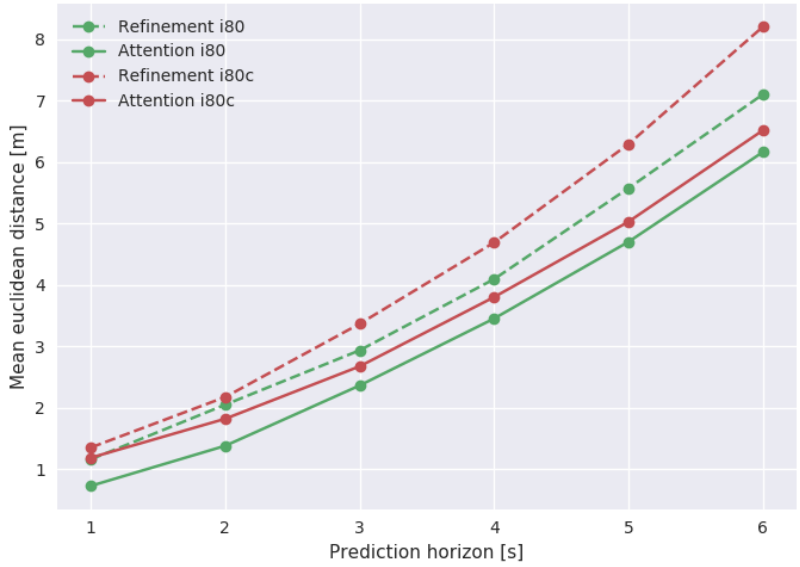


Figure 5.6: The prediction performances of the road-refinement model vs the road-attention model.

The road-geometry aware prediction is reflected largely in the increased lateral prediction accuracy on the i80c dataset, as shown in table 5.2. On the i80c data, knowing the road geometry is quite important. The baseline model fails to grasp this and obtains an average error of 3.69 meter after 6 seconds. This amounts to an average estimation error larger than the entire width of a lane. The road-refinement model brings this error down to 2.34 meter after 6 seconds, showing a clear improvement because this model has some idea of the road. It is the road-attention model that brings this error further down to 1.04 meters, showing that this model is capable of keeping the lateral error minimal over long prediction horizons.

The road-attention model understands the road layout best, and this is also shown in its predictions in figure 5.7. The model understands the road geometry and the position of the vehicle with respect to the road. Over a long prediction horizon, the road-attention model on the i80c even outperforms the baseline predictions on the straight i80 highway data. Furthermore, the road-attention model adds predictive performance on the i80 dataset. Despite the road understanding, the short-term prediction performance of the baseline and road-attention models on the i80c dataset remain the same.

The road-refinement model outperforms the baseline model on the i80c, yet prediction performances are still far from desirable. From the predicted trajectories, it can be seen that the model generates very discontinuous predictions. This could be caused by the lookup functionality. As stated before, the lookup functionality generates features based on predictions. These features cause a discontinuity in the learning, clarified by the

Table 5.2: Lateral performance on the i80c dataset

Prediction Horizon [s]	Baseline i80c [m]	Refinement i80c [m]	Attention i80c [m]
1	0.56	0.68	0.51
2	0.96	0.87	0.58
3	1.37	1.17	0.69
4	1.98	1.49	0.76
5	2.77	1.92	0.87
6	3.69	2.34	1.04

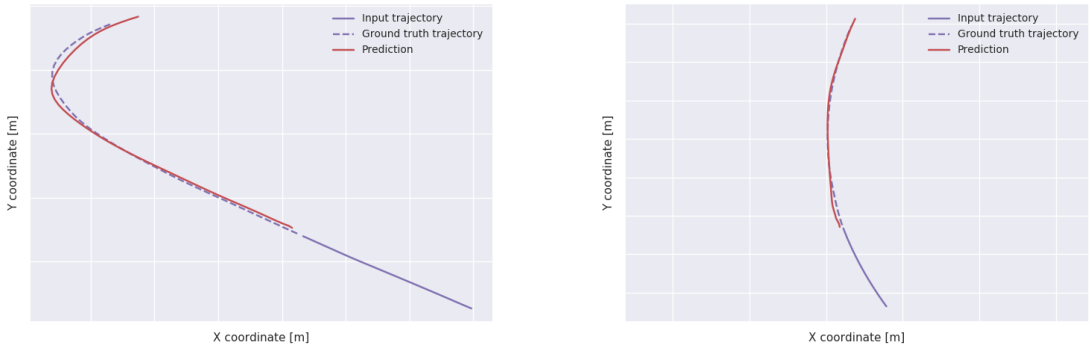


Figure 5.7: Road-attention predictions on the i80c dataset.

following example. The model is training on the i80c data and generates a road-refined prediction. It does so by generating an initial prediction, looking up the map features and refining the initial prediction. There is a prediction error, and the weights get updated according to this error. The next time the model encounters the same training data, it will give a new (possibly improved) initial prediction. This initial prediction generates a new set of map features, that are *not identical* to the map features that are looked up in the previous training round. The model thus updated its weights based on the first set of features, but because of its change in prediction behavior it gets different features at this time. This is a general drawback of the iterative approach. As such, a refinement method with lookup features from a semantic map is an unsuitable method to include road geometry in motion prediction.

Combined

The goal of this thesis was to find a road-geometry aware prediction approach that is compatible with the social pooling methods for interaction. Now that it is seen that adding both interaction and road-geometric awareness is beneficial for prediction, the two are combined. For this experiment, the road-aware prediction model that is used is road-attention, because it showed to be the most effective for road geometry modeling.

The combined model shows outstanding prediction accuracy, predicting quite more accurately on the i80c dataset than the baseline model on the i80 dataset. From the highlighted trajectories, similar predictions to that of the road-attention model are shown. In a lateral sense, the predictions stay within range of the ground truth trajectory, showing that the model is well aware of the road layout. As shown in table 4.18, the combined model generates the most accurate predictions on the i80c dataset of all models. It contains the advantages of the interaction-aware model and the road-attention model and that is reflected in the prediction results (figure 5.8).

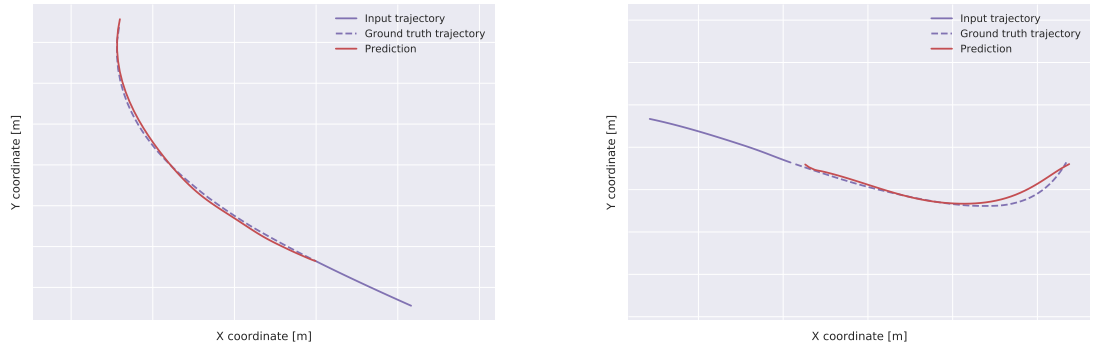


Figure 5.8: Two accurate predictions from the combined model on the i80c dataset.

The advantages of combining the road-attention and social pooling models is shown in more details in tables 5.3 and 5.4 below. Here, the baseline model, the road-attention model and the combined model's longitudinal and lateral performance are shown. In the lateral prediction performance, the advantage of adding the road attention to the model can clearly be observed, reducing prediction error from 3.69 to 1.04 meter. This also increases longitudinal performance from 7.60 to 6.32 meter. In the longitudinal performance, the advantage of adding the social pooling can additionally be seen, further increasing the performance compared to the road-attention model.

Table 5.3: Longitudinal prediction performance on the i80c dataset.

Prediction Horizon [s]	Baseline i80c [m]	Attention i80c [m]	Combined i80c [m]
1	0.84	0.83	0.95
2	1.74	1.54	1.55
3	2.82	2.51	2.27
4	4.11	3.55	3.10
5	5.67	4.88	4.21
6	7.60	6.32	5.43

Table 5.4: Lateral prediction performance on the i80c dataset.

Prediction Horizon [s]]	Baseline i80c [m]	Attention i80c [m]	Combined i80c [m]
1	0.56	0.51	0.80
2	0.96	0.58	0.81
3	1.37	0.69	0.83
4	1.98	0.76	0.89
5	2.77	0.87	0.99
6	3.69	1.04	1.23

On the i80 dataset, the interaction-aware model outperforms the combined model. This is interesting, because both the road-attention model and interaction model have shown to increase performance separately on the i80 dataset. When these two models are combined, this does not result in a further increase in performance. A possibility for this is that the interaction-aware model already introduces some information regarding the road geometry. As shown in the experiment in section 4.3, interaction also increases performance significantly on the i80c dataset. Because the interaction-aware model obtains information regarding the driven trajectories of vehicles ahead of the subject vehicle, it also obtains information regarding the road

geometry ahead of the vehicle. For the i80c, this is not a sufficiently reliable road-geometric information source, but for a straight road layout such as with the i80 this could be sufficient. The combined model then contains redundant information regarding the road geometry on i80, making the model more complex than necessary and showing reduced effectiveness compared to the interaction-aware model.

The results of the combined model on the i80c dataset are slightly worse compared to the combined model on the i80 dataset. This could be due to the distorted relative position of adjacent vehicles in the i80c dataset. As explained in section 4.1.3.1, the projection of the original trajectories onto the new road layouts stretches the trajectories, where trajectories that are further away from the reference curve get stretched more. This shifts the relative position of vehicles on adjacent lanes, which is important information to estimate the level of interaction between vehicles. Such noise is absent in the original i80 dataset, and could explain the reduced performance on the i80c data.

Another interesting phenomenon is the reduced accuracy on short-term prediction performance. All models show a superior short-term performance on the i80 dataset compared to the i80c dataset. Often, the predicted trajectories start with an offset compared to the past observed trajectory. Some examples of this are shown in figure 5.9. In such predictions, no road-geometric surprises can explain the offset that is given. Moreover, this performance degradation increases with the complexity of the model. For a 1 second prediction horizon on the i80c, the baseline model is slightly more accurate than the road-attention model, which is slightly more accurate than the combined model. This phenomenon can thus be related to the complexity of the model, indicating that a more complex model needs more data to obtain similar performance on these baseline prediction tasks.

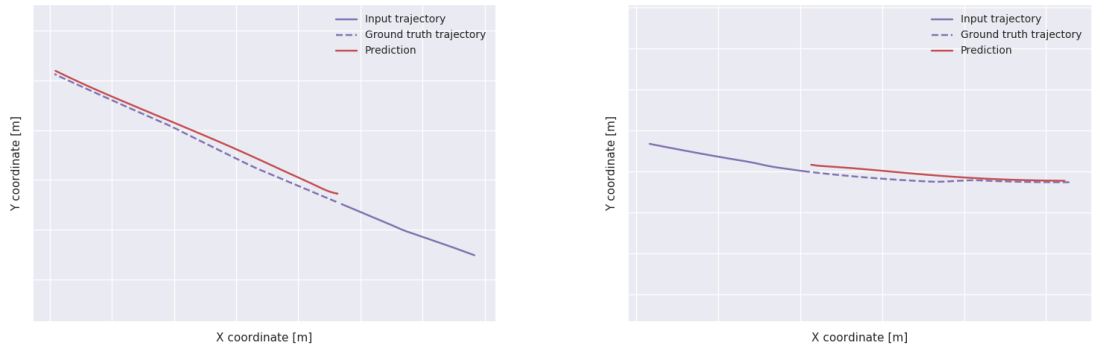


Figure 5.9: Some predictions with an initial offset in the future trajectory prediction.

The results of the combined model can be compared to the Convolutional Social Pooling (CSP) model from [13]. This work, as detailed in section 2.2, uses an alternative convolutional social pooling method on the NGSIM dataset. The data consists of a mix of the i80 and us101 datasets, being two continuous freeways without any road-geometric variations. The results are shown below in table5.5, and compared to the interaction-aware model on the i80 and the combined model on the i80c. The interaction-aware model is directly comparable to the CSP model, being two interaction-aware models trained on the original NGSIM data. The combined model should be capable of achieving similar results on the i80c as these models, because it understands the road geometry and the interactive forces.

The CSP method is reported for a prediction horizon up to 5 seconds. It can be seen that the interaction model on the i80 performs better for longer prediction horizons than the CSP method. The combined model shows very similar prediction accuracy to the CSP model, being only slightly less accurate. This could be due to the distorted relative position of adjacent vehicles, induced by the CCS transformation.

Table 5.5: Comparison of results between the CSP model from [13] and our models.

Prediction Horizon [s]	CSP i80 + us101 [m]	Interaction i80 [m]	Combined i80c [m]
1	0.61	0.81	1.34
2	1.27	1.33	1.85
3	2.09	2.01	2.54
4	3.10	2.82	3.44
5	4.37	3.81	4.52
6		5.02	5.82

5.2. Conclusion

In this thesis, motion prediction based on the road geometry and interaction with other vehicles is attempted. A major drawback of current state-of-the-art approaches is their inability to include information regarding both the interaction among vehicles, and the shape of the road. This limitation has been addressed by proposing new modeling techniques to can include road-geometric information and are compatible with some important existing models. A focus of this compatibility is with interaction-based predictions methods, due to the importance of both interaction-aware and road-aware prediction.

To show this importance in modeling, 5 different models are constructed that vary in their road-geometric and interaction awareness. A baseline deep learning model is used and extended with a road-geometry module, an interaction module, a combination of the two, or none. Because the existing road-geometric models are inadequate, two new approaches have been designed that exploit map information of the scene. These modules are referred to as road-refinement and road-attention.

To test the prediction capabilities of the models, they are trained on two different datasets. The first dataset, called i80, consists of trajectory recordings from a straight highway with dense traffic. The other dataset is a curved version of the i80, called i80c, where the trajectories and road are transformed to introduce road-geometric variations in the data.

The prediction performances from the models clearly show the importance of both interaction-aware and road-geometry aware modeling. The road-refinement and road-attention models outperform a road-agnostic model on the i80c data, showing better understanding of the road layout. Comparison between these models shows that the road-attention model obtains superior prediction accuracy on both datasets. The performance increase for interaction-aware models on both datasets also clearly shows the importance of interaction in modeling, compared to the baseline.

The model that combines the interaction and road-attention modules shows outstanding prediction performance on the challenging i80c dataset compared to all other models. From the predictions it can be seen that the model understands the road layout ahead of the subject vehicle, as well as the interactive forces that are in play. With the road-attention and interaction module, a model is obtained that can predict the motion of vehicles by considering the road geometry and interaction. The model utilizes reliable map information to assess the shape of the road ahead of the vehicle, and shows that it can accurately predict in situations with different road layouts. Meanwhile, the model is capable of understanding the interaction between vehicles within these road scenarios. Therefore, with the road-attention module, a model is obtained that overcomes some of the most important limitations in current motion prediction approaches.

5.3. Recommendations

Motion prediction is a challenging task, and future research is needed to obtain satisfactorily accurate predictions across all possible driving scenarios. This applies both to the discussed road-geometric and interaction-aware motion prediction, as well as a more general sense. Some recommendations for these future works are given here.

For motion prediction with road-attention, a dataset with real recordings of vehicles on curvy roads will be beneficial. The current model is aware of the shape of the road, but can not conclude any change in longitudinal driving behavior when a vehicle is in a turn, because there is none. From real observations, we know that vehicles decelerate before a turn and accelerate after a turn. Such longitudinal behavior is absent in the current data, whereas it will be interesting to see if the road-attention model can also comprehend this influence of the road geometry on driving behavior.

The road-attention module itself is an effective method to communicate the shape of the road. It is based on a well known attention mechanism. Every year, new variants on attention mechanisms arise with domain-specific advantages. For future research, more exotic variants for road-attention could be sought, exploring multiple attention mechanisms and RNN setups.

For motion prediction more broadly, it will be interesting to see how well the road-geometry and interaction aware model can be used for multimodal prediction. In the current model, a prediction can be seen as the average of all possible trajectories, given the current set of features. Multimodal predictions generates an entire set of future trajectories, rather than a single most probable one. Obtaining multiple predictions for a specific route can improve prediction accuracy.

Bibliography

[1] World health organization annual safety report 2017.

[2] Traffic safety facts: Children. Dept of Transportation (US), 2010. URL http://www-nrd.nhtsa.dot.gov/Pubs/811387.pdf.

[3] Transforming our world : the 2030 agenda for sustainable development. UN General Assembly, 2015. URL http://www.refworld.org/docid/57b6e3e44.html.

[4] Ten ways atuonomous driving could redefine the automotive world. McKinsey, 2015. URL https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world.

[5] Drunk driving fatalities. Responsibility.org, 2016. URL https://www.responsibility.org/wp-content/uploads/2018/02/2016-State-of-Drunk-Driving-Fatalities.pdf.

[6] National Highway Traffic Safety Administration et al. National motor vehicle crash causation survey: Report to congress. *National Highway Traffic Safety Administration Technical Report DOT HS*, 811:059, 2008.

[7] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.110. URL http://ieeexplore.ieee.org/document/7780479/.

[8] Anthony F Genovese. The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets. *Johns Hopkins APL Technical Digest*, 22(4):614—-623, 2001. ISSN 02705214.

[9] David Barber. A stable switching kalman smoother. 2004.

[10] Federico Bartoli, Giuseppe Lisanti, Lamberto Ballan, and Alberto Del Bimbo. Context-Aware Trajectory Prediction. pages 1–12, 2017. URL http://arxiv.org/abs/1705.02503.

[11] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1997)*, pages 994–999, 1997. doi: 10.1109/CVPR.1997.609450.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] Nachiket Deo and Mohan M Trivedi. Convolutional Social Pooling for Vehicle Trajectory Prediction. 2018. URL https://arxiv.org/pdf/1805.06771.pdf.

[14] Andreas Eidehall and Lars Petersson. Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):137–147, 2008. ISSN 15249050. doi: 10.1109/TITS.2007.909241.

[15] Geetank. Situational Awareness in Intelligent Vehicles. *literature survey*, pages 61–80, 2017. doi: 10.1007/978-0-85729-085-4_4. URL http://link.springer.com/10.1007/978-0-85729-085-4{_}4.

[16] Xinli Geng, Huawei Liang, Biao Yu, Pan Zhao, Liuwei He, and Rulin Huang. A Scenario-Adaptive Driving Behavior Prediction Approach to Urban Autonomous Driving. *Applied Sciences*, 7(4):426, 2017. ISSN 2076-3417. doi: 10.3390/app7040426. URL http://www.mdpi.com/2076-3417/7/4/426.

[17] Tobias Gindele, Sebastian Brechtel, and Rudiger Dillmann. Learning context sensitive behavior models from observations for predicting traffic situations. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, number Itsc, pages 1764–1771, 2013. ISBN 9781479929146. doi: 10.1109/ITSC.2013.6728484.

[18] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. 2018. URL http://arxiv.org/abs/1803.10892.

[19] Kevin Gurney. *An introduction to neural networks*. CRC press, 2014.

[20] Christoph Hermes, Christian Wöhler, Konrad Schenk, and Franz Kummert. Long-term Vehicle Motion Prediction. *IEEE Intelligent Vehicles Symposium*, pages 652–657, 2009.

[21] Sepp Hochreiter and Jurgen Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1–32, 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735{%}5Cnfile:///Files/F0/F0267A41-D807-4137-A5AC-8D9A84E9E12C.pdf{%}5Cnpapers3://publication/doi/10.1162/neco.1997.9.8.1735{%}5Cnfile:///Files/B1/B10E2649-D486-4D93-B71B-80023681156B.pdf.

[22] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle Trajectory Prediction based on Motion Model and Maneuver Recognition. (61161130528):4363–4369, 2013.

[23] Jihua Huang and Han-Shue Tan. Vehicle future trajectory prediction with a {DGPS/INS-based} positioning system. *Proc. American Control Conference*, pages 5831–5836, 2006. ISSN 07431619. doi: 10.1109/ACC.2006.1657655.

[24] Shau Shiun Jan and Yu Chun Kao. Radar tracking with an interacting multiple model and probabilistic data association filter for civil aviation applications. *Sensors (Switzerland)*, 13(5):6636–6650, 2013. ISSN 14248220. doi: 10.3390/s130506636.

[25] Kichun Jo, Minchul Lee, Junsoo Kim, and Myoungho Sunwoo. Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints. *IEEE Transactions on Intelligent Transportation Systems*, 18(2):460–476, 2017. ISSN 15249050. doi: 10.1109/TITS.2016.2605163.

[26] David B Kaber and Mica R Endsley. Out-of-the-loop performance problems and the use of intermediate levels of automation for improved control system functioning and safety. *Process Safety Progress*, 16(3):126–131, 1997.

[27] ByeoungDo Kim, Chang Mook Kang, Seung Hi Lee, Hyunmin Chae, Jaekyum Kim, Chung Choo Chung, and Jun Won Choi. Probabilistic Vehicle Trajectory Prediction over Occupancy Grid Map via Recurrent Neural Network. 2017. URL http://arxiv.org/abs/1704.07049.

[28] Stefan Klingelschmitt and Julian Eggert. Using Context Information and Probabilistic Classification for Making Extended Long-Term Trajectory Predictions. 2015.

[29] Stefan Klingelschmitt, Matthias Platho, Volker Willert, and Julian Eggert. Combining Behavior and Situation Information for Reliably Estimating Multiple Intentions. (Iv), 2014.

[30] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip H S Torr, and Manmohan Chandraker. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. 2017. doi: 10.1109/CVPR.2017.233. URL http://arxiv.org/abs/1704.04394.

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

[32] Chiu Feng Lin, A. Galip Ulsoy, and David J. LeBlanc. Vehicle dynamics and external disturbance estimation for vehicle path prediction. *IEEE Transactions on Control Systems Technology*, 8(3):508–518, 2000. ISSN 10636536. doi: 10.1109/87.845881.

[33] Marcello Montanino and Vincenzo Punzo. Making ngsim data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction. *Transportation Research Record*, 2390(1):99–111, 2013.

[34] Kevin P Murphy. Switching Kalman Filters 1 Introduction. *Dynamical Systems*, 1(August):1–16, 1998. URL `https://www.cs.ubc.ca/{~}murphyk/Papers/skf.pdf`.

[35] SeongHyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture. 2018. URL `http://arxiv.org/abs/1802.06338`.

[36] Dominik Petrich, Thao Dang, Dietmar Kasper, Gabi Breuel, and Christoph Stiller. Map-based long term motion prediction for vehicles in traffic environments. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, (Itsc):2166–2172, 2013. ISSN 2153-0009. doi: 10.1109/ITSC.2013.6728549.

[37] Eleni Petridou and Maria Moustaki. Human factors in the causation of road traffic crashes. *European journal of epidemiology*, 16(9):819–826, 2000.

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[39] Geetank Raipuria, Floris Gaisser, and Pieter P Jonker. Road infrastructure indicators for trajectory prediction. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 537–543. IEEE, 2018.

[40] Amir Sadeghian, Ferdinand Legros, Maxime Voisin, Ricky Vesel, Alexandre Alahi, and Silvio Savarese. CAR-Net: Clairvoyant Attentive Recurrent Network. 2017. URL `http://arxiv.org/abs/1711.10061`.

[41] Paul M Salmon, Michael A Regan, and Ian Johnston. *Human error and road transport: Phase one–Literature review*. Number 256. 2005.

[42] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. Interaction-Aware Probabilistic Behavior Prediction in Urban Environments. 2018. URL `https://arxiv.org/pdf/1804.10467.pdf`.

[43] Wilko Schwarting, Javier Alonso-mora, and Daniela Rus. Survey on Planning and Decision-Making for Autonomous Vehicles. (January):1–26, 2018. ISSN 2573-5144. doi: 10.1146/annurev-control-060117.

[44] SAE Standard. J3016,". *Sae international taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," levels of driving automation*, 2014.

[45] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. pages 1–9, 2014.

[46] John R Treat. Tri-level study of the causes of traffic accidents: an overview of final results. In *Proceedings: American Association for Automotive Medicine Annual Conference*, volume 21, pages 391–403. Association for the Advancement of Automotive Medicine, 1977.

[47] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow. *Transportation Research Part C: Emerging Technologies*, 2018.

[48] FCM Wegman, R Roszbach, JAG Mulder, CC Schoon, and F Poppe. Road safety impact assessment: Ria. Technical report, Report R-94-20, Leidschendam: SWOV Institute for Road Safety Research, 1994.

[49] Jun-Hai Yong and Fuhua Frank Cheng. Geometric hermite curves with minimum strain energy. *Computer Aided Geometric Design*, 21(3):281–301, 2004.

[50] Guan Zhai, Huadong Meng, and Xiqin Wang. A constant speed changing rate and constant turn rate model for maneuvering target tracking. *Sensors (Switzerland)*, 14(3):5239–5253, 2014. ISSN 14248220. doi: 10.3390/s140305239.