Interpreting Complex Scenes using a Hierarchy of Prototypical Scene Models

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt an
der Technischen Fakultät der Universität Bielefeld

von
Sarah Bonnin

14.10.2014

Abdruck der genehmigten Dissertation zur Erlangung des akademischen Grades Doktor der Ingenieurwissenschaften (Dr.-Ing.)

Der Technischen Fakultät der Universität Bielefeld

- am 14.10.2014 vorgelegt von Sarah Bonnin
- am 23.01.2015 verteidigt und genehmigt

Gutachter:

- Prof. Dr. Franz Kummert,
  Universität Bielefeld
- Dr. Christian Goerick,
  Honda Research Institute Europe

Prüfungsausschuss:

- Prof. Dr. Ralph Möller,
  Universität Bielefeld
- Dr. Thorsten Jungeblut,
  Universität Bielefeld

Abstract

To drive safely, a good driver observes her surroundings, anticipates the actions of other traffic participants and then decides for a maneuver. But if a driver is inattentive or overloaded, she may fail to include some relevant information. This can then lead to wrong decisions and potentially result in an accident. In order to assist a driver in her decision making, Advanced Driver Assistance Systems (ADAS) are becoming more and more popular in commercial cars. The quality of these existing systems compared to an experienced driver is relatively low, because they purely rely on physical observation and thus react only shortly before an accident. To fully avoid a collision, a driver needs more time to react, therefore the driver should receive an early warning. For an earlier warning of the driver, behaviors of other traffic participants would have to be predicted. We classify existing research in this area with respect to two aspects: quality and scope. Quality means the ability to warn a driver early before a dangerous situation. Scope means the diversity of scenes in which the approach can work. In general we see two tendencies, methods targeting for broad scope but having low quality and those targeting for narrow scope but high quality.

Our goal is to have a system with high quality and wide scope. To achieve this, we propose a generic framework, called Context Model Tree (CMT), that combines multiple high quality classifiers to predict if an entity is coming into the way of the ego-vehicle for many scenarios. This framework is a tree structure in which context based models are ordered according to their context specificity, from the generic ones in the top nodes to the most specific ones in the leaves. We have designed a set of activation rules to activate the nodes fitting to the current situation, using sensory information like GPS, digital maps or vision.

To show that a combination of general and specific classifiers is a solution to improve quality and scope, this thesis introduces the generic concept of our system followed by a concrete implementation for predicting if an entity is coming into the way of the ego-vehicle when changing lane for highway scenarios. On the highway, a driver usually changes lane for a reason. Our models use complex features based on contextual information and relations between entities. On the highway, one of the most influential indicators to predict if a vehicle is going to change lane is a slow predecessor. A CMT for highway contains in the top node a model that uses such general indicators. Two models to predict lane changes at entrance and giveway lanes are placed as sub-nodes. These models make use of the specific information inherent to these contexts. We will provide a comparison of the quality of the three models separately and the combination of the models using a CMT and show

that, in general, the CMT performs better in terms of prediction time horizon and prediction errors.

In order to show the flexibility and adaptability of the CMT, we also present an extension of the framework for pedestrian crossing prediction in inner-city scenarios. In inner-city, a pedestrian who wants to cross a road without having the priority to do so and decide not to is usually influenced by its surroundings, for example a vehicle approaching too fast and not having enough time to cross. A CMT for inner-city contains in the top node a model that uses such general indicators to predict crossing behaviors at an early time for any road, in particular roads where pedestrians do not have the priority to cross. However, there are specific locations such as zebra crossings, where based on expert driving experience, one would expect that a prediction can be done even earlier. Therefore, we have developed an additional specific model fitted to the context of zebra crossings. This model makes use of the specific information inherent to this context. The experiments show that this model produces both, better and earlier predictions in this specific context. Because our goal is to build a generic behavior prediction system, we finally apply the framework of the CMT to combine the two models.

We demonstrate that this multi-model system is well suited to provide early predictions for realistic data, including both, generic inner-city situations and zebra crossings. This work could therefore be a step towards better Advanced Driver Assistance Systems (ADAS), through the generation of earlier warnings to increase the reaction time of a driver.

Contents

Acknowledgements

I would like to thank all the people that have made this PhD possible and have transformed it into a wonderful experience. This concerns especially: Dr.-Ing. Christian Goerick, Dr.-Ing. Jens Schmuedderich and Dr.-Ing. Thomas Weisswange, my supervisors who helped me and who have given me a remarkable support all along this project.

I am very thankful to my supervisor from the university Professor Franz Kummert who was always giving relevant advices and an important feedback throughout the project.

Additionally, I would like to thank my scientific advisor Dr.-Ing. Stephan Hasler for his availability and his valuable support.

Thanks to the Honda Research Institute (HRI-EU) for funding the PhD project giving me this great opportunity and of course all my colleagues at HRI-EU.

Finally, I would like to thank my family, my friends and my boyfriend Mr. Ferrandon who have been supporting me and encouraging me over these four years.

1 Introduction

On January 29, 1886 Karl Benz patented the first true, modern automobile [Eckermann, 2001]. Since then, the use of the automobile did not stop growing, mostly due to the increase of the population. Over the years, we can observe many improvement of cars in terms of safety, comfort, etc., due to the progress of technology. Today, cars are the most used means of transport in Europe as shown in Fig.1.1 [Directorate-General Transport and Statistical Office of the European Communities, 2012]. It is also the transportation mode which causes the most fatalities. Around 1.2 million people are killed each year on our roads, a further 50 million are injured [Peden et al., 2004].

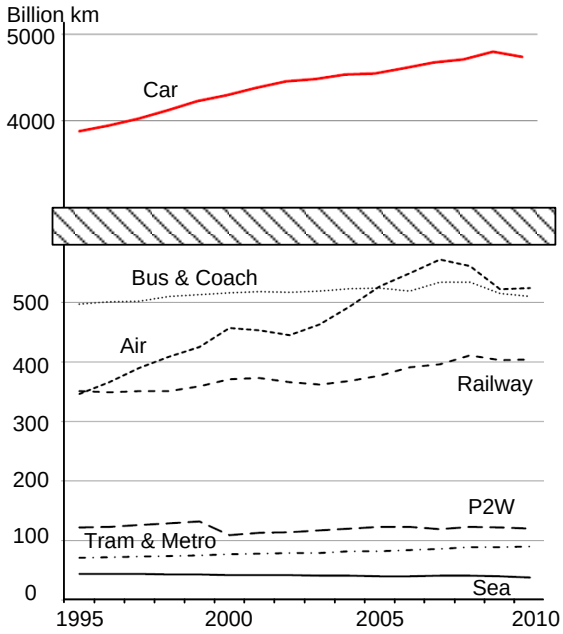


Figure 1.1: EU-27 Performance by mode for passenger transport. Each line represents the use of a transportation mode within the European Union in terms of distance travelled over the last 15 years. The solid line at the top represents the car.

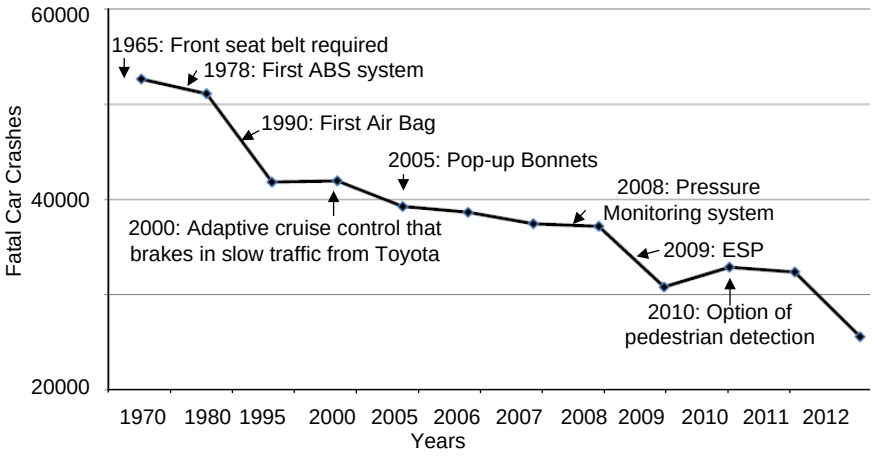


Figure 1.2: Total fatal car crashes by year in United States [United States. Dept. of the Treasury. Bureau of Statistics et al., 2010], [National Highway Traffic Safety Administration (NHTSA), 2010]. Some examples of safety systems appear together with the year they became commercially available.

In the last 60 years, car companies have used science and technology to increase safety and reduce accidents. Fig.1.2 includes some exemplary safety systems such as the seat belt, ABS, Air bag or electronic stability program which are nowadays mandatory in each car and reduce the risk of mortality during a collision. Until 2009, most of these systems (except adaptive cruise control) are considered to promote passive safety for a driver during a collision. The figure shows that since 1970, the number of fatalities on the road keeps reducing. Nevertheless, the number of dead people caused by car accidents is still too high. Most accidents occur because of human mistakes. In 2009, in the United States, 84% of fatalities were caused by driver's inattentiveness (cell phones, talking, etc.) [National Highway Traffic Safety Administration (NHTSA), 2010]. In Great Britain, between 2005 and 2009, 67.5% of accidents were due to a misinterpretation or a late reaction [Institute of Advanced Motorists (IAM), 2011]. Most of accidents caused by driver late reaction could be avoided by using more advanced safety systems. The next section will present the new generation of these safety systems, whose role is to assist the driver by sending a warning shortly before a collision is happening, to allow him to react earlier.

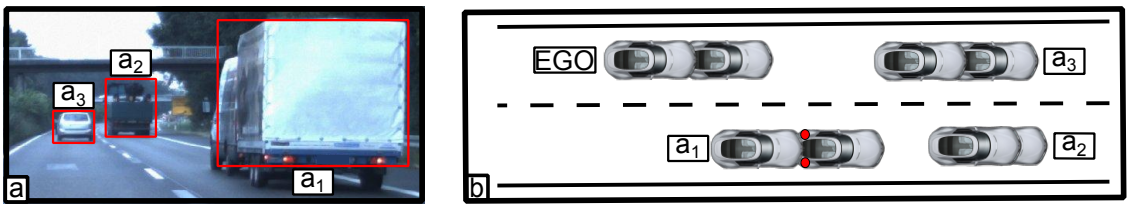


Figure 1.3: (a): Exemplary scene in which three vehicles are detected: a_1, a_2 and a_3. (b): Solid cars represent the current position of vehicles. Cars are projected to their future position using their current position and velocity. Silhouette cars represent the future position of vehicles. The red circles indicate that a_1 is going to brake because its estimated future position is too close from its slower predecessor.

1.1 Motivation for Behavior Prediction

A driver is naturally aware of the scene and can anticipate the future behavior of her surroundings to decide about her own behavior. Nevertheless, behavior prediction is a very complex problem.

Today, many vehicles are equipped with a variety of sensors (cameras, radars, lasers, GPS, etc.) that collect information about the environment, e.g. localize other traffic participants and static infrastructure. For example, radars can detect the position and velocity of a vehicle. A laser can only detect the position of a vehicle and a tracking algorithm can be applied to get the velocity. The image shown in Fig.1.3.(a) represents an exemplary scene as it could be observed by a camera inside the ego-vehicle. Once the different scene entities have been detected, a recognition algorithm is then used to identify which of the entities is e.g. a vehicle, or a pedestrian. In this scene, three entities have been recognized as vehicles driving at different speeds on the highway.

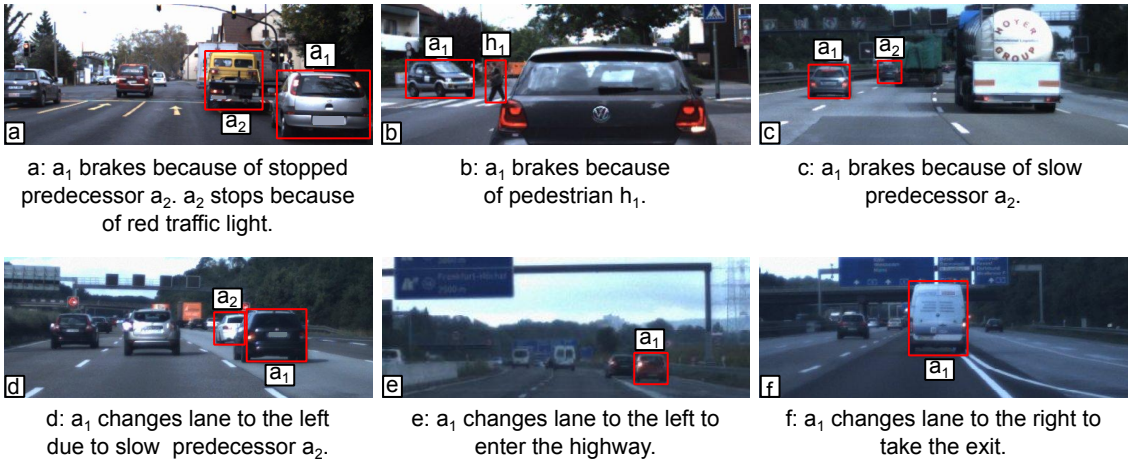
The most used prediction method uses information given by the sensors, e.g. headway distance and speed of traffic participants, to compute a projection of their future state. To evaluate the potential risk of a collision, they compute a Time-to-Collision (TTC) between two projected states. In Fig.1.3.(b), a_1 drives faster than a_2. If vehicles keep their respective velocities, a_1 will be too close to a_2 and they will collide. To prevent a collision, the driver of a_1 should be aware that she has to brake. In this thesis, a prediction method, which uses only sensory observation will be called 'physical' prediction.

In order to improve safety on roads and reduce car accidents, car manufactur-

ers use such technologies to create active safety systems called Advanced Driver Assistance Systems (ADAS). Current ADAS can react to dangerous situations by steering, accelerating or braking or by sending an early warning to the driver to avoid a collision. Existing ADAS estimate a risk of collision based on the current sensing observations. For example, in Autonomous Emergency Braking (AEB), the ego-car warns the driver when there is a low TTC with its predecessor or TTC is infinite but the predecessor strongly decelerates. When a dangerous situation is already occurring, if the driver does not react to the warning and a collision cannot be avoided, the system automatically brakes the car to reduce impact speed [Jansson et al., 2002]. A recent comparison between AEB performances from different suppliers showed that such systems help to significantly lower the number of road deaths and seriously injured accident victims [Allgemeiner Deutscher Automobil-Club e.V. (ADAC), 2011]. However, because current ADAS rely on the observation of the current state, it can only react when a dangerous situation is occurring. The time of the start of the prediction of ADAS depends on the complexity of the scene. Traffic density, different possibilities of behaviors and quantity of contextual information influence the complexity of the scene. For example, it is more difficult to drive at high speed on highways with many cars that change lane than when driving on an empty road. Recent studies emphasize the shortcomings of existing ADAS in complex scenes, such as junctions or intersections [Grover et al., 2013].

A driver can avoid an accident only if she has detected and recognized it as a dangerous situation. This phase is called the human perception time. It is followed by the reaction time, which is the time the driver needs to start reacting to avoid the accident. On average, the perception-reaction time is 1.5s [Lerner, 1993][Taoka, 1989]. Due to the complexity of the scene and the degree of attention of the driver, her perception-reaction time varies (also depending on the age of the driver, physical conditions, weather conditions...). Studies [National Highway Traffic Safety Administration (NHTSA), 2000] suggests that 60% of the rear-end crashes could potentially be avoided if the driver had an extra 0.5s to this 1.5s perception reaction time. Herrero Zarzosa et al. [2007] did an experiment where they predict the relative dynamics of surrounding vehicles on the road and warn the driver in case of a collision with one of them, and they showed that, an accident need to be anticipated at least 2s before a collision to ensure that the driver has time to react.

The main goal of this thesis will be to propose a system that can warn a driver earlier than existing ADAS. An earlier warning increases the reaction time of a driver and gives her more time to react. The solution proposed in this thesis to increase the reaction time of a driver is to use behavior prediction. The following



a: a_1 brakes because of stopped predecessor a_2. a_2 stops because of red traffic light.

b: a_1 brakes because of pedestrian h_1.

c: a_1 brakes because of slow predecessor a_2.

d: a_1 changes lane to the left due to slow predecessor a_2.

e: a_1 changes lane to the left to enter the highway.

f: a_1 changes lane to the right to take the exit.

Figure 1.4: Complex scenes in which vehicles brake or change lane due to different contextual reasons.

section explains the difficulties of such a method.

1.2 Challenge of Early Behavior Prediction

It is a challenging issue to predict the behavior of others early enough, so that it increases the time for a reaction. A traffic participant has many possibilities for behaviors with many different ways to execute them. Cars, for example, can, among others, drive straight, change lane to the right or to the left, brake, turn left or right. Additionally, the reason for a behavior, which can be used for the prediction, is strongly influenced by the local context [Montel et al., 2013]. For example, a driver brakes when approaching a red traffic light, but also to let a pedestrian cross or because of a slow predecessor as shown in Fig.1.4.(a-c). A driver will change lane because of a slow predecessor, or to enter or leave the highway as shown in Fig.1.4.(d-f). The applicability and interpretation of traffic rules also depends on the context (lane markings, traffic signs, zebra crossings, etc.). Behavior at an unsigned intersection differs from that at a traffic light controlled intersection. The first difficulty of behavior prediction lies in the diversity and complexity of the different possible behaviors according to the different scenes.

One major behavior influencing factor within the context is the scenario. A scenario defines a set of scene elements and restricts the possible behaviors. An example scenario can, for example, consider a rural road with two opposing lanes with drivers driving within their lane. Some scenarios, we can call generic because

Figure 1.5: Pedestrian h_2 is going to cross at the zebra crossing shown in the red rectangle. h_1 is walking on the pavement. The white arrows show the distance between the zebra crossing and both pedestrians, the green arrows show their motion direction.

they apply over a larger period of time and space. For example, on highways, a driver should follow her lane, but she can change lane to the left to overtake a slower predecessor and change lane to the right to return to her previous lane. Other specific scenarios are more limited in duration or spatial extend. For example, at entrances on highways, vehicles have to change lane to the left before the end of the lane, even though they do not have a slow predecessor. To ensure fluent traffic, vehicles being driven in the left neighboring lane of the entrance lane often change lane to the left to free the lane for the new entering vehicle. In a specific scenario, the reason of doing a behavior has changed due to a more specific context. In the related work chapter, we show the difficulty of dealing with predicting behavior in many different scenarios. To cope with the large variety of scenarios, the solution proposed in this thesis will be to create a system that can combine multiple methods for predicting behavior in different generic scenarios, e.g. on highways, and in more specific ones like entrances. The system formulation should be general enough to be applicable to any type of behavior in any combination of contexts.

A driver is able to perceive her environment to understand the state of her surroundings and anticipate their future actions in order to modify her own behavior accordingly. In contrast to physical prediction, more advanced prediction requires a better understanding of what a scene is consisting of. In this thesis, this step is called scene reconstruction, which consists of creating the relations between traffic entities and contextual information with the goal to understand how they influence

each other. For example, Fig.1.5 shows a scenario with two pedestrians h_1 and h_2 and a zebra crossing. The creation of relations between entities uses their position to localize them in the street and recreates the neighborhood of each traffic participant, e.g. which pedestrian is in the zebra crossing area. With such scene understanding, we can compute some specific features such as the distance between a pedestrian and a zebra, the distance between the ego-vehicle and a pedestrian, etc, to predict which of the pedestrians want to cross at the zebra. The importance of a feature depends on the current context. For example, on highways, the time-to-contact to a predecessor correlates with the lane change decision, whereas on an entrance, it is almost meaningless. To build a system for behavior prediction, the idea presented here is to find those scene specific features most correlated with the traffic participants behavior decision. In Fig.1.5, pedestrian h_2 might not be aware of the ego-vehicle arriving because she is looking at the other side of the road. In such a situation, there is a risk of collision between the arriving vehicle and h_2 if the driver is inattentive or distracted. In such a case, a predictive system could send a warning to the driver before the pedestrian steps on the road at all.

To do behavior prediction, the system needs to get, for each context, its predefined set of features. For that reason, we propose to design a set of context specific models: each model represents one generic or specific context and exploits all relevant features belonging to it. For example, a model for the zebra scenario uses features like the distance to a zebra crossing or the motion direction as shown in Fig.1.5. A classifier can be trained to predict future behavior from this set of features. We will provide some more details on which features have been defined to predict lane change maneuvers on highways in chapter 4 and to predict the behavior of crossing pedestrian in inner-city in chapter 5.

1.3 Goal of the Thesis

The general goal of this thesis will be to develop a method that can provide early warning to a driver of a car, to increase her available reaction time beyond what is currently done by commercial ADAS. The solution proposed in this thesis is to use behavior prediction.

Fig.1.6 shows a typical processing flow for doing behavior prediction. Existing ADAS use vehicles equipped with several sensors that can detect and recognize the different entities in the scene. The prediction module uses these information to do a projection of the current state of traffic participants into the future. A decision module computes the TTC between two projected states to evaluate the risk of

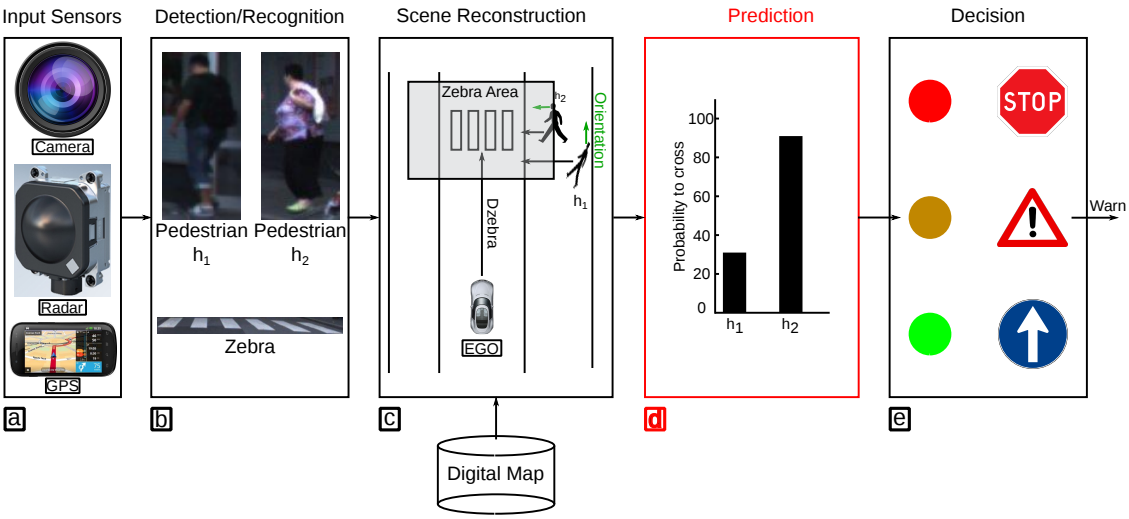


Figure 1.6: System overview of a behavior prediction ADAS. (a): A collection of sensors that could be mounted on the ego-vehicle. (b): Entities that have been detected and recognized. (c): Scene reconstruction, creation of relations between traffic participants. Black arrow is the distance between the zebra crossing and traffic participants. Green arrow is the moving orientation of pedestrians. (d): Context specific models are organized in a tree to do the behavior prediction. (e): Decision module to decide about the interaction between the driver and the system.

collision and send a warning to the driver. Our system will have an additional module which role is to reconstruct the scene before doing the prediction. This scene will be then used by the prediction module. Although this thesis does not focus on scene reconstruction, we will provide some more details on it in chapters 4 and 5, because of the big impact it has on the quality of the prediction algorithm.

The work presented in the following also aims at providing a generic system capable of predicting behaviors for numerous complex scenes. To cope with the large variety of context, context-specific models are organized in a tree. By activating only those models fitting the present context, a system applicable to different context is achieved. To have a system that performs well over a longer time, we need to combine these models. In the related work chapter, we will show that combining models is not an easy task. Therefore, in this thesis, we develop a general framework on how to combine models for multiple scenarios.

To demonstrate the benefits of this framework, we also applied it to ADAS problems from two different domains. For this, we restrict behavior prediction to those

entities surrounding the ego-vehicle, which might directly interact with it. We train classifiers to predict whether a surrounding entity will come into the way of the ego-vehicle. We picked the example of predicting if a vehicle on the highway will change into our lane, and as a very different one, if a pedestrian in inner-city will cross the road in front of us. This prediction should be early enough for the driver to react and avoid the collision. In this thesis, we target for a prediction that starts up to 2 seconds before the behavior changes.

1.4 Outline

The remainder of this thesis is divided into six chapters. The first chapter introduces the topic of this thesis. It describes the problem we want to solve and sketches our solution. The second chapter gives an overview of related work, it outlines the differences and similarities, strength and weaknesses of existing prediction methods and explains why existing approach cannot solve the problem described in the introduction. The third chapter presents the formal concept of our system designed to predict surrounding traffic participant behavior and describes a generic method to build a Context Model Tree (CMT). Our focus in this work is to provide a system, generic in terms of creation and application. It should be easily reproducible and easily expandable. The two following chapters will show two very distinct examples of an application of this concept. The fourth chapter demonstrates the benefit of using our system when applied it to cut-in prediction on highways. The fifth chapter outlines an application of our system for inner-city scenarios for predicting if a pedestrian is going to intersect the path of the ego-vehicle. Finally, the sixth chapter gives a conclusion and presents the possible future extensions.

2 Related Work

Behavior prediction has been a main topic of research in the last 10 years, but the understanding of meaning and purpose varies. For example, in biology, behavior can be regarded as any action of an organism that changes its relationship to its environment [Schwarz, 1978]. In automotive, we consider a behavior as a category of future movements, like walking/stopping or changing lane. On a different level of abstraction, trajectories are described as a succession of physical states. Multiple different trajectories can still be instances of the same behavior. A behavior is usually caused because of some contextual influences. Many approaches propose to learn the reasons why a behavior changes based on a collection of data.

Early ADAS relied on pure detection. Once an entity such as a pedestrian is detected, it is possible to estimate a risk of collision with the ego-vehicle. For example, the Collision Warning with Full Auto Brake and Pedestrian Detection system, that is commercialized by Volvo, alerts the driver or starts an automatic emergency braking when the time-to-collision (TTC) with a pedestrian in front drops below a threshold [Coelingh et al., 2010][MobilEye Ltd., 2007]. However, such reactive systems, which use detection only, are often not sufficient to fully avoid a collision, because the defined threshold is usually very low and does not consider the reaction time of the driver. For effective collision prevention, the detection of pedestrians should be followed by the prediction of the possibility of collision between pedestrians and vehicles using pedestrian dynamics and behavior analysis [Gandhi and Trivedi, 2006]. With the prediction, it is possible to evaluate the risk of collision before it occurs, which can be used to send appropriate warnings to the driver early enough that she has time to react to it [Gandhi and Trivedi, 2007]. Our approach aims to increase the reaction time of the driver by using behavior prediction. Several studies about the prediction of behaviors exist, focusing on scope or quality to varying extends. Quality refers to the accuracy of the prediction as well as the prediction time horizon which corresponds to the time the prediction starts. Scope refers to the diversity of scenarios in which an approach can work.

In this chapter, we will give an overview of the different existing methods that have been developed in order to recognize and predict behaviors for either the ego-vehicle or the other traffic participants. We will provide details about approaches that combine models for different contexts, but will detail the prediction models

themselves in chapters 4 and 5, where we can relate them directly to our applications.

2.1 Behavior Recognition

In the literature, the term behavior recognition is strongly linked to intention recognition, situation assessment and even prediction. In this thesis we separate approaches which target for the recognition of driving maneuvers from those that target for intention recognition or prediction, because methods to recognize behaviors do not do any prediction. Behavior recognition consists of recognizing the current state of an entity at the present moment, for example to classify the state of a traffic participant and analyze its behavior to evaluate a potential danger [Kumar et al., 2005]. A driver's vehicle control action, like turning the steering wheel, can provide for example, information about a lane change maneuver [Kuge et al., 2000]. The availability of an increasing number of sensors enlarge the scope of the behaviors to be recognized. Also, the behavior of a driver can be inferred more reliably by combining multiple information sources [Gerdes, 2006][Meyer-Delius et al., 2009]. Usually, a traffic participant has a certain reason to change its behavior. For collision mitigation, it might be very interesting to identify these reasons in order to assist a driver in her maneuver. For example, recognizing the scene elements that cause a braking maneuver like a red traffic light or a static predecessor can be used to initiate a braking maneuver [Platho and Eggert, 2012]. Finding the causes why a behavior is happening is closely related to behavior prediction.

Behavior recognition is used to recognize a behavior while it occurs, the driver is warned late and might not have time to react.

2.2 Ego-Vehicle Behavior Prediction

A large number of studies is targeting the prediction of the ego-vehicle behavior. Maneuvers such as emergency braking, lane changing or turning at a junction can be predicted with a good accuracy by measuring the steering angle, steering velocity [Hermes et al., 2009][Hermes et al., 2010][Liu and A.Pentland, 1997][Salvucci et al., 2007][Taniguchi et al., 2012][Kumar et al., 2013][Dogan and Edelbrunner, 2008], and also considering the action of the driver on the accelerator or the brake pedal [Hülnhagen et al., 2010][Kumagai and Akamatsu, 2006][Naito et al., 2007][Pentland and Liu, 1999]. Several studies revealed the importance of head motion and a driver's eye gaze in the prediction of the ego-vehicle behavior [Mccall et al.,

2005][Oliver and Pentland, 2000][Ohn-Bar et al., 2014]. Before a lane change, a driver will first look in the inside and outside mirrors of her vehicle to make an estimation of the best moment to change lane. When the driver sees that she has the time and the space to maneuver, she starts turning the steering wheel. The head motion of a driver as well as her eye gaze give many information about where she plans to go and when. A lane change can be predicted up to two seconds before it starts [Morris et al., 2011], which is much earlier than the one second prediction horizon using only monitoring data [He et al., 2012]. Such systems can be useful to assist the driver in her maneuver. They can also be used for example to detect the physical state of the driver to keep her driving lane in case she is falling asleep [Liu et al., 2009].

These studies show that reliable behavior prediction is possible based on driver monitoring to predict what a driver will do. However, in this thesis, we are aiming at predicting the behavior of surrounding traffic participants, for which monitoring information like the head motion or the action on the brake pedal are difficult to obtain.

2.3 Traffic Entity Behavior Prediction

Many different approaches to predict behaviors exist already. These methods can be grouped according to what they want to predict and how they do it. Vasquez et al. propose the prediction categories “descriptive models”, “behavioral models”, and “hybrid models”. In the following we will consider the additional class of “next state prediction”, define and group the related work into these classes [Vasquez et al., 2009].

2.3.1 Next State Prediction

One simple method for prediction uses the current position and velocity to estimate the most probable next location of a dynamic object. This method is mostly used in the prediction step of tracking algorithms [Barth and Franke, 2010][Musleh et al., 2010][Ma et al., 2009][Barth and Franke, 2008][Kaempchen et al., 2004][Wakim et al., 2004]. The estimation of the next position of a moving entity can then be used to compute a probability of collision with the ego-vehicle [Abramson and Steux, 2004][Jin et al., 2013][Bahram et al., 2014]. Two vehicles might be in conflict when approaching a merging situation. The next state estimation helps to take a decision

about which of the vehicles should stop to let the other one pass [Agamennoni et al., 2011].

Because predictions are based on physical observation only, they can be expected to be accurate for objects with a low dynamic range and/or for a very short time horizon only.

2.3.2 Descriptive Models

A traffic participant usually follows a particular path, for example to cross a road, walk along the pavement or turn at an intersection. Trajectories of other traffic participants can be estimated with the car-to-car technology [Ammoun and Nashashibi, 2009][Batz et al., 2009]. The different paths are all composed of a succession of states that can be learned [Large et al., 2004][Makris and Ellis, 2002][Kim et al., 2012][Saunier et al., 2007][Kooij et al., 2014b][Schneider and Gavrila, 2013]. A matching algorithm can then be used to compare the current trajectory (in terms of spatial location, velocity or heading angle) with the learned ones [Hu et al., 2003]. When the best match is found, the corresponding trajectory serves to predict future positions of the pedestrian and to estimate the risk of collision [Chen and Yung, 2009][Quintero et al., 2014]. A similar approach learns trajectories and divides the conflicting areas into regions. A risk of collision is computed by checking the overlap between regions of two vehicles [Sekiyama and Watanabe, 2009].

Tracking and trajectory prediction methods define the state (in terms of position and velocity) of an entity for any point in time. This makes these methods suitable for low level control, they provide a very detailed spatial and temporal information of an entity. However, pedestrians can change their walking direction very suddenly or vehicles can change maneuvers very quickly, therefore the prediction error will grow fast with increasing time horizon. The benefit these methods is, that they are usually working in a variety of simple scenarios. One could say that these methods trade off prediction quality at longer time horizons for a broad scope. An alternative to have a longer time horizon would be to look at the problem from a higher level, that is, to predict behavior.

2.3.3 Behavioral Models

Usually, a collision between two traffic participants occurs when one of them changes its dynamic very fast [Martin, 2006] and the other does not have time to react to it. For example a vehicle which suddenly brakes to avoid a pedestrian might also collide with the vehicle behind who might be inattentive or brake too late. In a dangerous

situation a driver should be warned early to fully avoid a collision. A solution to warn the driver earlier is to use behavior prediction. A behavior describes a category of future movements, for example walking versus stopping. Behavior prediction tries to classify a category without defining every single predicted state. This is often enough to decide if two entities will collide. In contrast to the descriptive models that can tell with a good accuracy where an entity will be and when in the next few steps, behavioral models recognize the intention of the entity which allow for a much earlier reaction of the driver. The methods described below can be called model-based approaches. The principle is to represent the scene by creating relations between vehicles and static infrastructure [Polychronopoulos et al., 2004][Hülsen et al., 2011] and look for features indicating a maneuver like a lane change. These indications are derived from contextual information, e.g. knowing which lane a vehicle is driving in, knowing that it will move to a faster lane if approaching another car, etc. Such information is used to define specific features, feed them to a classifier and obtain labels that describes the future behavior of other traffic participants. Models are designed with different level of specificity depending on what they have to do.

The easiest way to start reducing the complexity of the prediction consists in differentiating entities based on their possible behavior options. Entities have different behavior repertoires, for example a pedestrian does not execute the same behaviors as a vehicle. Also, even if two entities have similar behaviors, they will have different dynamics, for example a bicycle differs from a motorcycle in terms of size or dynamics. It seems very difficult to use the same model to predict all the different behaviors of all the different entities. For that reason, the same behavioral model is restricted to predict only one type of entity.

The same behavior can happen in different contexts due to different reasons, for example a vehicle can change lane on highways when approaching a slow predecessor or in inner-city to change direction. It is very challenging to build one generic model that can predict the same behavior in all the different possible situations because it would need to learn all the different contexts in which the behavior can occur. To overcome this problem, a behavioral model is most of the time restricted to a certain context, for example to predict behaviors on highways [Tsogas et al., 2008][Dagli et al., 2004] or in inner-city [Klingelschmitt et al., 2014][Aoude et al., 2011] [Käfer et al., 2010][Koehler et al., 2012][Koehler et al., 2013].

The more a model integrates contextual data, the more accurate it becomes and the earlier it can predict a behavior. However, this comes at the cost of reducing the scope. The model is then very limited in its application [Kasper et al., 2012][Schlechtriemen et al., 2014][Tomar and Verma, 2012][Reichel et al., 2010].

For example, a model that can predict lane changes on highways with a very good accuracy will fail at predicting lane changes at entrances or exits due to contextual variations [Garcia Ortiz et al., 2012]. Similarly, to be accurate at predicting turn maneuvers at intersections, a model will need to have very detailed information of the infrastructure which make the model very specific to the scenario [Lefèvre et al., 2011].

2.3.4 Hybrid Models

Models that belong to the fourth category combine physical information with semantics to improve the prediction. The first group of hybrid approaches focuses on predicting a behavior in order to improve trajectory prediction. An early recognition of a maneuver such as turning left or going straight at intersections provides useful information about the future trajectory of vehicles [Tran and Firl, 2014]. The second group of hybrid approaches focuses on first predicting a trajectory in order to improve the prediction of a behavior. For example, different trajectories are learned at intersections to predict a turn maneuver or lane following [Liebner et al., 2012]. A similar example defines two separate models to learn typical trajectories for walking and stopping behaviors. A trajectory matching can then be used to predict the future behavior of a pedestrian as well as its future state [Keller et al., 2011]. Alternatively, driving behaviors can often be considered as a sequence of basic maneuvers, e.g., an overtaking maneuver may consist of the following segmentation: approach leading vehicle, lane change to the left, passing, and lane change to the right. By recognizing the starting sub-maneuvers, one can predict behaviors early and estimate the future path of a vehicle [Gindele et al., 2010].

Many methods perform complex behavior prediction with high quality. These approaches show very good accuracy in the prediction for a long time horizon but their situation specificity prevents application to a wider scope of situations.

2.4 Prediction With Multiple Models

The above examples show the difficulty of targeting both, wide scope and high quality at the same time. One possibility to overcome this limitation is to use the approaches used by the robot which won the DARPA challenge. It has different situational models for different scenarios and uses its position in a digital map to activate the correct one [Urmson et al., 2008][Ferguson et al., 2008]. The DARPA

challenge is a competition where autonomous vehicles have a mission to perform, which consists of going from a starting point to an ending point via a series of waypoints. To localize itself and its surroundings in the environment, the robot uses a very precise digital map. Based on the local position of the robot, a model corresponding of the road structure is activated, for example a normal road model, an intersection model or a non structured area model. Because the different models used by some of the participants in the DARPA Urban challenge are designed specially to perform the tasks of the challenge, they are able to perform very well, profiting from the clearly defined and well distinguishable scenarios. One model at a time is activated using a digital map. Once a model is activated, the autonomous vehicle predicts its future trajectory as well as the one of the moving vehicles using its dynamic and the structure of the road. For example, at an intersection, the robot will predict each possible trajectory of the other vehicles according to the possible lanes they can drive in. The robot computes a risk of collision comparing the overlap between its own predicted trajectory and each future trajectory of the others in order to estimate the safest path to reach the next waypoint of its mission. In the real world, scenarios can become more complex and overlap with each other. In such a case, it can happen that the model activated based on the current location is not suitable to the current scene. For example, a model to predict cut-ins on the highway is not accurate at entrances or exits.

Graf et al. [Graf et al., 2013] also build a system based on the idea of improving prediction by using multiple models, but with a much finer granularity. They use a case-based reasoning method inspired by the human decision making mechanism. The principle is to compare a new situation with those stored in a database and find the most similar one. They first compute the local context of the car that is predicted, which includes its relative position to the ego-vehicle and the position/existence of all other cars in its surrounding. They build separate models for each local context. One model uses typical velocities and distances between the predicted vehicle and its surroundings for a specific maneuver. Each velocity and distance is mapped into one discretized 2D space that takes into account the time series properties. In a test case, they compute a similarity measure with all these 2D models in order to find the best fitting model and predict the behavior related to it. For example, one model predicts a cut-in maneuver of the right predecessor of the ego-vehicle using the relations to the front predecessor of the predicted vehicle as well as the ego-vehicle and its front predecessor. Additionally, for the same local context, there are different models for different global contexts, e.g. highway, rural road or inner-city. Therefore, they sort the models into a tree structure with the global context at the top, followed by the local context and with the models for

each behavior at the leaves. In order to evaluate their approach on real data, they did some recordings on German highways with a car equipped with a radar and a mono camera to detect vehicles and lane markings. The models target at predicting cut-in maneuver of the right predecessor of the ego-vehicle. Their dataset contains about 127 cases and half of it is stored into the database and the other half is used for the testing. On average, their approach can predict a cut-in 2.1s before it starts. In the case where no matching is found between the current case and the learned ones, they update the database by adding a new model. This high model specificity allows the use of very simple classifiers for good quality, but requires a very large training dataset. The authors mention another disadvantage of that approach: it will fail in situations which have never been seen before.

Berndt et al. use a related idea of combining multiple models and apply it to the recognition of the ego-vehicle's driver's intention [Berndt et al., 2008]. The authors designed a set of models, each of which is able to predict a single type of maneuver like changing lane to the left or to the right, turn left or right and normal lane following. To save computation time and improve the scope and quality, only a subset of the models is active at a time based on the position of the ego-vehicle in a digital map which limits the set of possible behaviors. For example, the models to predict turn maneuvers are activated only 30m in front of a crossing (based on the digital map position/annotation). Each maneuver is learned with a Hidden Markov Model [Rabiner, 1989]. The authors assume that driving maneuvers consist of a series of consecutive actions that are characteristic for the maneuver. Based on this assumption, they model each action as a state of the markov chain. The model learns the probability to change from the current state to the next one. Depending on the complexity of the maneuver the number of states of the model can vary. Because the last states of a model represent the end of a maneuver, the evaluation of a model focuses on the first states of the chain in order to have an early prediction. The different models are trained and tested on real data extracted from the CAN bus of the ego-vehicle: throttle pedal position, brake pressure, steering wheel angle, steering wheel angle velocity, yaw rate and velocity, as well as distances to the next left and right turn possibility, street curvature and street type from digital map. In order to evaluate the relevance of the features, the authors created one model per behavior per input sensor. They found that the steering wheel angle and the steering wheel angle velocity were the features giving the most accurate prediction. To train e.g. the lane change maneuver models, the authors use a dataset with 100 lane changes (50 left and 50 right). The models are then tested on one hour of driving on freeways with 50 lane changes. To find the optimal number of transitions within a model, they also compared models with different number of states. The

best model contained 9 states and covered 8s to 10s. With the three first states of the model, they reach 71% and 74% correct predictions for left and right lane change maneuvers respectively. The maneuver is predicted after the vehicle has started to move, shortly before it touches the lane markings. The models to predict left and right turn maneuvers are trained on 60 turns and tested on one hour of recordings with 35 turns. Once again the steering wheel revealed to be the best feature. The best model contained 9 states and the turn maneuver can only be predicted on states 3 to 6 which is shortly before the end of the maneuver. Furthermore, the authors also mention a limitation of their approach: the same maneuver can happen in many different scenarios and for different reasons. Therefore, the quality of the prediction might decrease when a model designed to predict a behavior in a specific context is applied to an inappropriate ones. For example, the quality of the prediction decreases when a model designed to predict lane changes on highways is applied to entrances.

All these approaches combine multiple models in order to predict behaviors in different situations. Additionally, Graf et al. propose a generic way of constructing a system that allows for the combination of multiple models. However, because they try to find the best model of their database to predict the current situation, only one model is used for the prediction. To overcome this problem, we will propose a formalism that can be used to build any kind of multi-model system for behavior prediction allowing several models to predict the same entity at the same time. To define the set of rules, we take advantage of the ideas already mentioned above. Our approach allows the combination of models with different levels of specificity which are activated based on the current context. We want to have both, multiple models active at the same time predicting different subsets of the scene, as well as groups of models that cannot be activated at the same time because they are mutually exclusive. We therefore choose a hierarchical tree structure. We will demonstrate the usefulness of our approach by some example systems and also show that this can be easily extended.

Similar to approaches discussed in the last paragraph, we believe that by combining multiple classifiers in a meaningful way, we can get the best of both worlds. The location (to access more detailed context information) appears to be an important piece of information to choose which classifiers should be combined and when they should be applied. Our goal is to define a model combination formalism which allows the construction of systems that produce the best possible prediction in any scenario.

3 Context Model Tree Framework

As explained in the related work section, a very specific model can predict a behavior with a high accuracy and a long time horizon, but this comes at the cost of a very limited scope. We could increase the scope of the prediction of a behavior by using multiple context specific models. However, when combining these, three problem cases can occur:

- Case 1: Mutually Exclusive Context Elements:
  Two contexts that never occur at the same time or space, e.g. highway and inner-city.

- Case 2: Dependent Context Elements:
  One context always occurs when the other one also occurs, e.g. highway and entrances on highways.

- Case 3: Competing Context Elements:
  The same scene indicates different behaviors depending on the context, e.g. at entrances on highways, a vehicle in the giveway lane will either change lane due to a slow predecessor or to free the lane to an entering vehicle.

To cover these cases, we will sort the context elements into nodes of a tree structure. The layers of what we will call Context Model Tree (CMT) represent the abstraction level, with the most generic context at the root and the most specific ones at the leaves. Each node of the tree is a model that is tuned to context information, an example is shown in Fig 3.2. Models can be combined only if they predict the same behavior and generate a single output per predicted entity. There is no limitation on the type of behavior to be predicted. The example in Fig.3.2 shows a CMT that could predict a very generalized behavior: "Is the entity coming my way?" (we could therefore even combine models that predict pedestrians or cars in the same CMT). A CMT will only predict the behavior of one entity at a time.

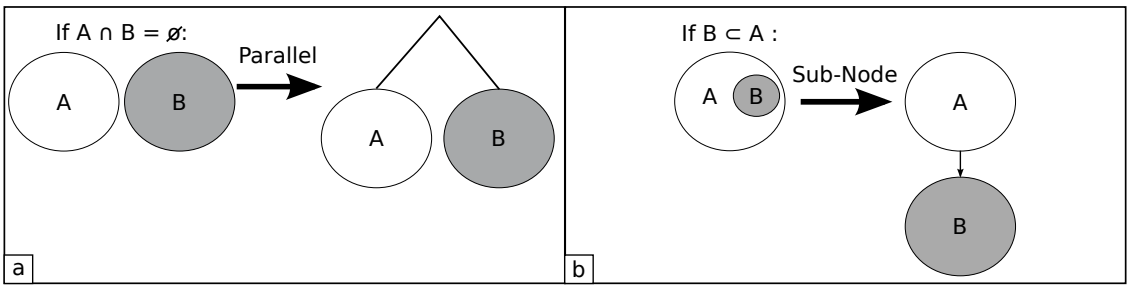


Figure 3.1: Context Definition. Red and green circles represent context A and B respectively. (a): If A and B have no elements in common, the two contexts are mutually exclusive and placed in parallel in the tree. (b): If B is contained in A, B is a sub-node of A.

CMT, solution to case 1, Mutually Exclusive Context Elements:
If A and B are two contexts and $A \cap B = \varnothing$, the contexts are mutually exclusive and placed in parallel in the tree, as shown in Fig.3.1.(a).

Generic context elements are usually mutually exclusive because of a clear spatial separation. A model designed to predict behaviors in the highway scenario will perform poorly at predicting vehicles in inner-city and vice versa. Lane changes on highways or in inner-city can happen due to many different reasons. For example, on highways, a model will predict that a vehicle is going to change lane to overtake a slow predecessor. In inner-city, a vehicle approaching a slow predecessor will not necessarily change lane. Instead, it might brake if the slow predecessor is approaching a red traffic light, or it might adapt its speed to its predecessor in case it wants to turn right at the next intersection. Since mutually exclusive models have nothing in common, it is important that they do not get active at the same time. To ensure that only one generic model can be active at a time, we have ordered the models into a tree structure, and mutually exclusive models are positioned in parallel into the tree. Additionally, we define a node activation mechanism. Each node has its own unique activation rule. A node gets activated based on features in the current scene, e.g. the ego-vehicle's GPS position as shown in Fig.3.2.

CMT, solution to case 2, Dependent Context Elements:
If A and B are two contexts, $A \cap B \neq \varnothing$ and $A * B$ and $B * A$, the contexts are dependent and B is placed as a sub-node of A in the tree, as shown in Fig.3.1.(b).

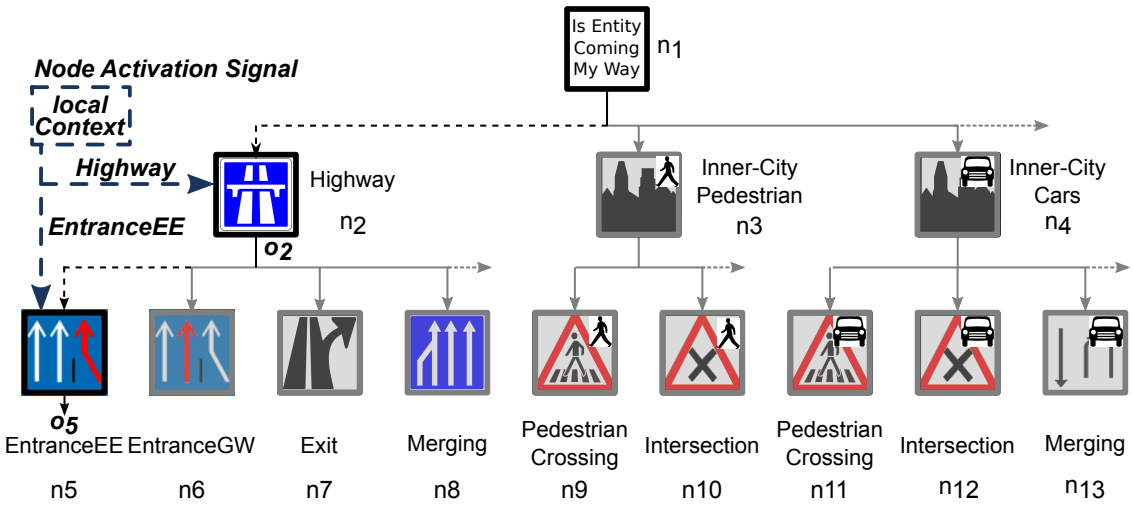


Figure 3.2: Visualization of a context model tree to predict if an entity is coming into my way. Each rectangle represents a node of the tree. Nodes with black frames have been activated by the digital map. Black dashed edges represent the active path. In this example, the "Highway" node is active and sends its output $\mathbf{o}_2$ to the active sub-node entranceEE which combines it with its own output to produce the final output $\mathbf{o}_5$.

To make sure to have a prediction in any context, indicators are defined for a global context which cover the main reason for a behavior. However, there are some specific contexts for which the quality of the prediction decreases because the indicators are not adapted or missing. It might be worth considering additional indicators to better predict behaviors in the more specific context. A more specific context is usually connected to a more generic one in the sense that it uses some additional context information. Depending on each traffic participant's local context, the model for one or the other context will produce the best predictions. For example, at a highway entrance, vehicles in the left most lane do not behave differently from any other part of the highway and could therefore be predicted by the model of the generic highway scenario. On the other hand, the prediction for vehicles in the entrance lane will profit from taking into account information about the entrance scenario. Due to a hierarchical ordering of the models, a specific node will always be activated together with its parent.

CMT, solution to case 3, Competing Context Elements:
In some cases, two contexts might both influence a traffic participant behavior at the same time. As in case 2, these contexts overlap in time, but in addition they

have a combined effect. For example, a vehicle being driven on the highway might get a high lane change prediction by a highway model if it has a slow predecessor. This is also true for vehicles that drive in the lane to the left of the entrance lane. But in addition, those vehicles should also be predicted to change lane by an entrance model if it can give way to an entering vehicle. When two dependent contexts are predicting the same entity, it can be that both predict the behavior correctly, but it can also be that both contexts have opposite predictions. In some cases, only one model can be right, whereas in other cases the prediction of both models should be combined to improve the results. To ensure accurate results, the final behavior prediction will be a combination between the outputs of all active contexts based on a combination function.

This tree structure automatically provides a fall-back in case of localization errors and the non-activation of a specific node, because prediction based on the generic parent node will always cover the most usual behaviors.

In the following we will define the set of rules needed to create our tree structure.

3.1 Definition Of The CMT

In a CMT, every traffic participant is processed separately.

- A CMT is composed of a set of nodes $N = \{n_i | i = 1...I\}$ where I is the total number of nodes N as shown in Fig.3.2.
- Each node n_i has exactly one parent n_j except for the root node.
- A node $n_i = (f_i, b_i)$ has a prediction function f_i, and a combination function b_i.
- The prediction function f_i of an active node delivers an intermediate output c_i for the expected behavior of a traffic participant for the specific context of the node.
- The combination function b_i of an active node n_i delivers the final output o_i, which is the result of the combination between c_i and the output o_j of the parent node n_j.

3.2 Activation Principle

The CMT uses an activation mechanism to decide which nodes are to be activated at time t. This decision requires information about the current context of the traffic participant that is currently predicted. This can be obtained from different sources.

For example, in Fig.3.2, the "Inner-City" node for pedestrian will be activated when a pedestrian is detected, and vice versa for a vehicle being driven in inner-city. An other example is using GPS localization on annotated digital maps to activate the system, e.g. when the ego-vehicle is approaching an exit lane, similar to [Hold et al., 2010][Wang et al., 2012][Mccall et al., 2005]. Alternatively, it is possible to acquire this information with sensory detection algorithms (e.g. highway exit detection [Hold et al., 2010] or detection of specific road environments [Takagi et al., 2006][Thuy and Puente Leon, 2010][Kuehnl et al., 2013]). It might actually be beneficial to use multiple sources to be robust against positioning/detection errors. One could for example confirm information from the digital map by sensory detection, as it is done for current product level street sign recognition [Volvo Car Corporation, 2012]. Certain information sources might be better suited for certain cases, temporary scenarios like roadwork are easier to be covered by visual detection [Mathibela et al., 2012]. In cases where information is missing or ambiguous with respect to which node to activate, one could decide to only activate the parent node if its more generic context is better supported ("fall-back mechanism"). Alternatively, one could introduce an additional node covering this specific scenario or design a more detailed activation principle.

In general, the activation principle of a CMT should always be designed to ensure that at any time, there is only one node active at the same level. In addition, the activation of a given node has to always activate its parent node, which should be true if nodes are ordered according to specificity.

3.3 Prediction and Combination Functions

The prediction function f_i of a node models the expected behavior of a traffic participant and delivers a confidence c_i, as shown in Fig.3.3.(a) for node n_i. The simplest prediction function can be a constant value, for example $f_i : c_i = 1$, like in Fig.3.3.(a) for the node n_j. This could be useful if for example in a certain context a given behavior always takes place (e.g. stopping at a red traffic light). In a more general case, f_i depends on some input $\mathbf{x}_i$, $f_i : \mathbf{x}_i \mapsto c_i$. For example, $f_i(\mathbf{x}_i)$ can be a classifier where $\mathbf{x}_i = [x_{i,1}, ..., x_{i,L_i}]$ contains L_i features from situational context and relevant relations between traffic participants. An other example for an f_i would be

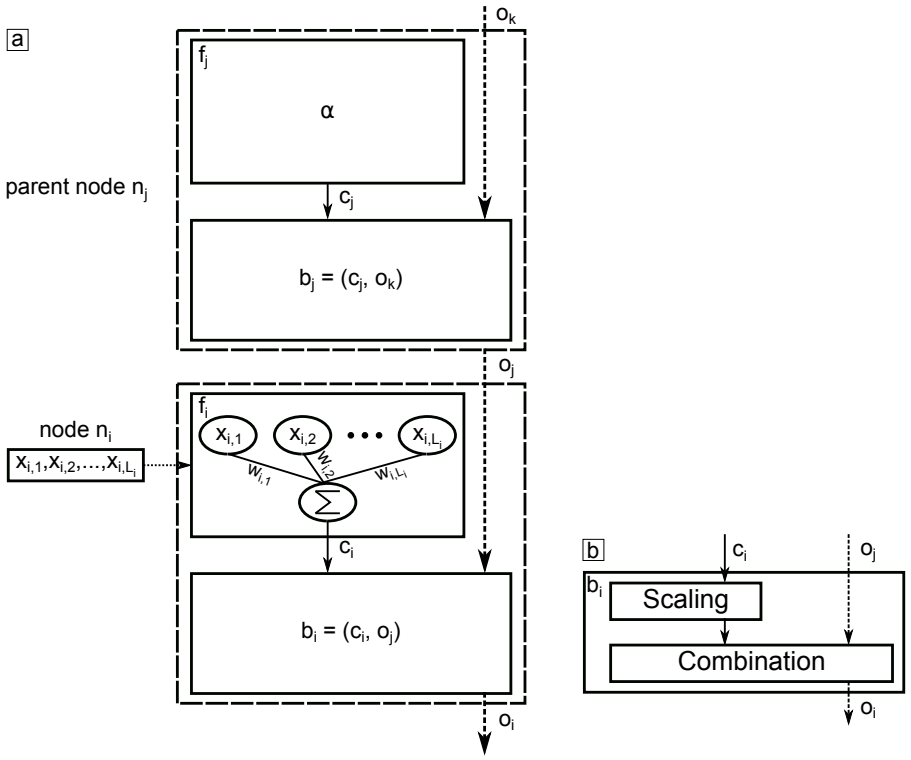


Figure 3.3: General structure of a node (a) and its combination function (b). Solid arrows represent intermediate confidences outputs from a prediction function. Dashed arrows represent the final confidence output from a combination function. (a): Node n_j is the parent of n_i. Both nodes contain a prediction function and a combination function. (b): The combination function b_i of a node n_i first scales the intermediate confidence c_i, then applies the combination between o_j and c_i after it has been scaled and outputs o_i.

to simply map one input value to a confidence using a sigmoid function. The final output o_i of n_i is the result of the combination function between the intermediate c_i and the output o_j of the parent node.

For each traffic participant, the parent node n_j produces an output o_j. The output o_j is sent to the active sub-node n_i. The combination function b_i of node n_i uses its own intermediate confidence c_i and the confidence o_j to compute the final output $o_i = b_i(c_i, o_j)$ as shown in Fig.3.3.(b). Each node has its own combination function based on context and prior knowledge.

The meaning of a confidence value is bound to its classifier. Therefore, it is not

Table 3.1: Contextual Errors with Examples

Context Error Types	Examples
Location	Intersection (50.106619°N, 8.758126°E)
Situation	Entrances on the highway, traffic lights, general intersections
Traffic participant type	Car, truck, motorcycle, bicycle, pedestrian
Environmental condition	Rain, shadow, snow, night
State of the ego-vehicle	Low speed, high speed, static
General environmental structure	Trees, houses, corn field[1]

possible to compare the confidence values between two classifiers. To make a fair comparison between two confidences, they both need to be normalized. One way to do the normalization is to consider a meaningful threshold (e.g. in terms of optimized ratio between true positive rate (TPR) versus false positive rate (FPR)) for each classifier and divide the confidence by it. As shown in Fig.3.3.(b), the combination is done between the normalized output o_j and the normalized confidence. The final confidence output o_i of the most specialized active node n_i can then be used in a the decision module (see Chapter 1 Fig.1.6), to determine how to interact with the driver.

3.4 Context Model Tree Design Guide

A CMT is a generic framework that is used to combine multiple prediction models in order to improve the prediction in terms of performance and prediction time horizon. This section explains in which situation it might be interesting to build a CMT and how to build it.

3.4.1 When to consider a CMT solution

A common method to evaluate the performance of a model designed to predict one behavior is to look at the performance of the model, like the TPR and/or the FPR. Additionally, the prediction time horizon (PTH), describing how early the prediction can start, is an important criterion to evaluate the quality of a model. One possibility to improve the performance of a model is to reduce the prediction

[1]For more details see [Graf et al., 2014].

errors. Errors caused by perception, like wrong lane assignment or false pedestrian detections are difficult to avoid without changing, e.g., the sensor setup or the detection algorithm. However, as it has been outlined in the previous chapter, some of the errors might be context dependent. To identify the causes of these errors it is beneficial to perform an error analysis, looking for clusters of meaningful structural errors like situation dependent or traffic participant type dependent errors, see Table 3.1 for more examples. If such error clusters are found, the idea is to verify if it is possible to reduce them.

A potential reason why the original model produces errors when predicting behaviors in some specific contexts can be that the model is trained on a large and unbalanced dataset, it has too few data related to the specific context. Then the classifier cannot learn the differences between the different contexts and produces errors. One example is the prediction of behavior on normal weather condition and under rain. On highways, vehicles keep larger safety distances when it is raining, and such variations are reflected on the features and can influence the classifier. A first possibility is to create an additional model similar to the original one and train it only on the data corresponding to one of the specific clusters that causes errors.

It can be that the model requires additional features that were not considered during the training of the original model, or which had a negative influence on the training. For example, at entrances on highways, a vehicle being driven in the left neighboring lane of the entrance lane might change lane to the left to give space to the new entering vehicle. A model designed to predict lane changes to the left on highways because of a slow predecessor can be retrained on the data related to the give way context without giving any better results because the model needs additional contextual information about the new entering vehicles and the new lane. In such a case, a second possibility is to build a similar model but adapt the features to the specific context.

Some contexts are more difficult to learn than others. For example, a stopping maneuver can happen due to many different reasons and it is not obvious why it happens [Platho and Eggert, 2012]. A third possibility is then to construct a complete new model specific to the context.

To make the original model and the new model comparable, the idea is to compare them on the same dataset that caused errors and check if the new model can correctly predict the error case. A second type of evaluation is to compare the PTH to see if the new model can predict the behavior earlier.

Using an additional model is worth considering if it correctly predicts cases that were not predicted in the original model or if, generally, it has a better PTH for the

given context. If it is possible to find a combination function that does not result in additional prediction errors, models should be combined using a CMT.

3.4.2 How to construct a CMT

A CMT consists of a tree structure with activation rules and combination functions.

Each model will be a node of the tree. A model should be positioned in the tree as a sub-node of an existing node if its context exists only inside its parent node. A sub-node is a specific model targeting at predicting a behavior for a context for which the parent node produces errors. For example, an error analysis shows that a model designed to predict lane changes on highways produces errors at exits scenarios. A model especially designed to predict lane changes at exits is then a sub-node of the "Highway" node. An example of a three level CMT would be a generic node on top to predict pedestrian crossing behavior. A sub-node would predict crossing behavior at intersections. A node specific to intersections with traffic lights could be then on a third level. Two models that are built for mutually exclusive contexts should be placed in parallel if they are in the same sub-tree. For example, a model to predict lane changes at entrances on highways and a model to predict lane changes at exits have very different contexts and should be placed parallel into the tree, as sub-nodes of the "Highway" node.

The most difficult configuration is when two models are built for contexts which can appear separately but can also overlap. For example a model to predict behaviors during rain and a model to predict behaviors at night. If the overlapping case (rain at night) is not covered by any model, both models can be placed in parallel and use an activation rule that makes them mutually exclusive. For example, the night model could be activated only at night without rain and the rain model only for rain at daylight. The overlapping case will be left to the parent node. If at least one model covers the overlapping case, the activation rule has to be formulated to favor one model. For example, one is active when it rains and the other in nights without rain.

Each node has its own unique activation rule. Different sensor information can be used to define these rules, e.g. digital map data to determine if the ego-vehicle is approaching an intersection and a camera to detect traffic lights. When constructing the tree, an important constraint is to make sure that only one node of a level can be active at a time. To respect this constraint, it is possible to add a dummy node as explained above between the top node and the specific one that just passes the information from the top node to the sub-node in order to make sure that two models that can be active at the same time are at a different level of the tree.

When it is activated, a node produces a confidence output for the predicted entity. If any of its sub-node is activated, the confidence output is sent to the active sub-node. The sub-node produces a confidence output and applies a combination function between both confidences. In order to make the confidences comparable, each node will first scale its confidence output by dividing it by the best threshold of its associated model. The combination function is then applied between both scaled confidences. The best threshold of a model is fixed during its evaluation according to some criteria, for example by selecting the threshold that has the best ratio between TPR and FPR. A fixed threshold is required in order to optimize the combination function. However, with a little bit more time, it is possible to choose a combination function and optimize the thresholds of the parent and sub-node together for this combination function by checking all combinations of thresholds between both nodes and choose the best pair of thresholds.

A method to find the most suited combination function is to plot the output confidence values of each model against each other in a 2D plot. The idea is to find the function that best discriminates the positive from the negative examples. For example, the combination function can use logical operators like (AND/OR/XOR), use a maximum/minimum function or use combinations like mean and variance. The final confidence value is above one if the threshold has been reached, below one otherwise. In the case of probabilistic models, it is also possible to multiply the probabilities together in case of statistical independence.

In the following, we propose two practical examples for the construction of a CMT to predict if an entity is coming into the way of the ego-vehicle by predicting lane changes on highways and crossing pedestrians in inner-city.

4 Context Model Tree for Highways

In the previous chapter, we presented a generic framework to combine multiple models to predict behaviors. In this chapter, to demonstrate the benefit of using such a framework, we propose a concrete application in which models are combined in order to predict maneuvers on highways. Existing ADAS such as adaptive cruise control (ACC) are developed to assist the driver on highways. However, as explained in the previous chapter, some of these systems have a limited time horizon and in some cases this can cause a late reaction. Therefore, we have chosen to develop our system to predict behaviors on highways to improve the time horizon of those systems and allow for an earlier reaction. Common ACC systems are particularly challenged by cars changing to our lane. When a slow vehicle wants to enter the lane of faster vehicles, a collision might happen. This is why it might be helpful to predict when a lane change occurs in order to react early and avoid critical situations. Usually, a lane change maneuver happens for a reason. For example, on highways, a driver might change left when approaching a slow predecessor, whereas a vehicle entering the highway might change lane before the end of the entrance. To allow for an accurate behavior prediction, we propose to create a model per context to learn the different context elements that cause a lane change maneuver. In the next section, we give an overview of the prediction methods that are the most related to our work. In this chapter, in order to predict if an entity is coming into the way of the ego-vehicle, we present a model to predict lane change maneuvers of the right predecessor of the ego-vehicle on highways as well as two models to predict cut-in maneuvers at entrances. We apply a Context Model Tree (CMT) to combine these models. The last section will present an evaluation to compare the performances of the single models separately versus the performances of the combination of the models with the CMT.

4.1 Existing Methods for Lane Change Prediction

As described above, many approaches target at predicting lane change maneuvers on highways. In this section, we give a detailed description of four of them that we consider as the most similar to our work as well as the most relevant ones.

Schlechtriemen et al. propose a method to predict lane change maneuvers to the left, to the right and lane following of the ego-vehicle's direct neighbors on highways [Schlechtriemen et al., 2014]. The prediction is the result of a four step process.

The first step consists of building the scene. Vehicles are connected with each other as well as with lanes. Each vehicle is connected to its 8 neighbors, the left, front, right predecessors and successors and the vehicles located on its left and right sides. Based on this reconstructed scene, complex features such as the distance to the start of an entrance, the time gap between two vehicles or the time-to-collision with the predecessor are computed for each neighbor of the ego-vehicle.

In a second step, the probability distribution of each feature for each of the three behavior classes is learnt separately using a Gaussian mixture model. To compute the probability distribution of a feature, a DB-Scan algorithm is used to apply a Gaussian to the values of a similar range. The DB-Scan algorithm returns the maximum number of possible Gaussians. The Expectation Maximization (EM) algorithm is then applied to minimize the number of Gaussians. The EM algorithm is an iterative method for finding maximum likelihood parameters for statistical models given incomplete samples. The Bayesian information criterion (Schwarz Criterion) based on the likelihood function is used as a condition to stop iterating when the optimum number of Gaussian has been found.

In a third step, the authors evaluate each feature for each class to find a small subset that maximizes the predictive power of the classifier. The authors also investigate how early a feature can predict a behavior. Based on ROC curves, a feature is considered important if the value of the area under the curve (AUC) is above a threshold.

In a fourth step, a naïve Bayes classification algorithm is applied. Based on the assumption that features are independent from each other, the authors multiply the probability output of all features for each class and apply the maximum function to get the final prediction.

To evaluate the system on real data, a vehicle is equipped with a front-facing stereo camera, front- and back-facing long range radar sensors and two sensors at the left and right side of the vehicle. Thirteen hours of driving are recorded on German highways and the database consists of a total of 160781 samples. A lane change maneuver is annotated positive from the moment the middle of the vehicle crosses the lane markings to two seconds before. After the computation of the AUC of each feature, the three most relevant features are found: the relative velocity of the preceding car could predict a maneuver 2.2s before it starts, the lateral velocity relative to the lane has a horizon of 2.0s and the lateral offset to the lane center

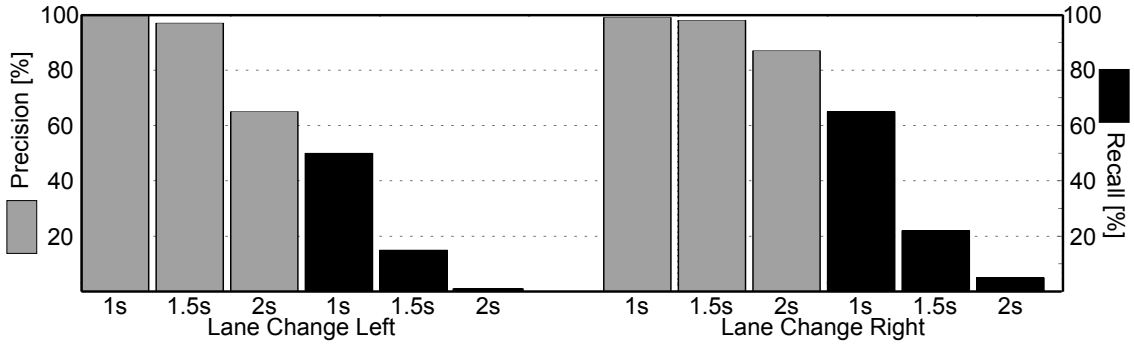


Figure 4.1: Lane change classifier performance evaluation. Precision in black and recall in grey for lane changes to the left and to the right on highways at different time horizons [Schlechtriemen et al., 2014].

with a time horizon up to 1s. To evaluate the performance of the classification, only these three features are used. The authors compute the precision and the recall [Fawcett, 2006] to evaluate the performance of the classifier for different time horizons as shown in Fig.4.1. A high precision means that most of the positive predictions are correct and a high recall means that an algorithm predicts correctly most of the positive events.

The authors have defined a large set of features and analyzed them separately in order to find out the most relevant ones. The lack of infrastructure features in the final feature selection shows that the dataset focuses on general highway situations. For those, the model shows an accurate prediction for a prediction time horizon up to 1s. Above 1s horizon, the accuracy decreases and above 1.5s less than 20% of the vehicles that change lane are correctly predicted.

Garcia et al. propose a method to predict lane changes of the ego-vehicle on highways [Garcia Ortiz et al., 2012]. In order to make the approach generic and applicable to predict other traffic participants, the authors have intentionally chosen not to use information about the driver of the ego-vehicle.

A Locally Weighted Projection Regression classifier [Vijayakumar et al., 2005] which is fast and suitable for online learning is used to predict if the driver of the ego-vehicle is going to change lane in a future prediction horizon. The authors consider the careful design of the input features the most important part for a successful classifier. To predict a lane change of the ego-vehicle, several indicators are computed considering the influence of the left, frontal and right predecessor of the ego-vehicle: time-to-collision with the front predecessor, the difference to the

desired speed, measured by calculating the average past speed in a time window and comparing it to the speed of the frontal predecessor, the distance to the left predecessor, the difference between left and frontal predecessor speed and the lateral speed of the right predecessor.

To test the model on real data, a vehicle is equipped with a radar and a camera. The combination of radar and video allows for the detection of other traffic participants and their position relative to the ego-vehicle. The ego-vehicle position, speed and acceleration are extracted from the CAN bus and data are filtered with a Kalman filter. The camera detects lane markings in order to assign a lane to each traffic participant. The recordings consist of 50 minutes with 20 lane change situations that are used to train and test the method. The evaluation focuses on the prediction of lane change to the left maneuvers. A lane change maneuver is annotated positive from the moment the driver of the ego-vehicle starts turning the steering wheel to two seconds before. Timesteps between 2s and 4s are not used for training in order to have clear separation between positive and negative examples. Everything else is annotated as negative examples. The system is trained and tested with N-fold cross validation.

The authors propose a frame-based evaluation to show the direct measurement of the learning capability of the system, but they also introduce an event-based evaluation in order to make a functional evaluation of the system. The principle is to group consecutive predictions that are above a threshold into one event. Then, an event is compared to the annotated groundtruth to compute false positives and true positives. Since we use the same evaluation method, we will give a more detailed description of its mechanism later in this chapter. The system can predict a lane change 2s before it starts with 80% of correct predictions. Additionally, the system produces 45 wrong predictions per hour. In order to understand the reasons why the system creates so many errors, the authors propose an error analysis that shows that many of the errors come from the perception like wrong lane and vehicle detection. Additionally, they show that the context influences the behavior of the driver. At exits and entrances on highways, the system is inaccurate at predicting lane changes of the ego-vehicle. The model also produces errors at predicting lane changes in congested traffic. In contrast to the model presented in [Schlechtriemen et al., 2014] which has a very short time horizon, this model can predict lane changes up to 2s horizon. However, this comes at the cost of a narrow scope. The model produces many errors when applied to an inappropriate scenario.

Reichel et al. propose a method to predict cut-in maneuvers in merging scenarios [Reichel et al., 2010]. The authors use a scenario based Random Forest algorithm

which is an extension of the Random Forest algorithm to predict if the ego-vehicle participates at a convoy merging on the absorbing convoy lane. Given a set of input vectors and the corresponding targets, the idea underlying the Random Forest algorithm [Breiman, 2001] is to construct a large number of different classification trees and then to take a majority vote among the individual classifiers. To make sure that classifiers are trained with different sets of features, the authors apply a bootstrapping technique which assures that each tree sees only about 64% of the inputs in the training set. A scenario based Random Forest first equalizes the datasets used to train and test the classifiers in terms of having the same number of positive and negative samples. Different classifiers are evaluated with different combinations of features using an estimate of the generalization error. The best classifier, minimizing the number of features, uses four features out of twenty. The features are computed using relations between the ego-vehicle and its dynamic and static environment.

The algorithm is tested on real data recorded with a vehicle equipped with a radar, a laser and the CAN bus. The dataset contains 4424 negative and 809 positive examples and is divided into 8 subsets, with seven subsets recorded on highways and one subset recorded on rural road. Four of the highway subsets contain merging situations occurring at the end of a highway and the others contain merging situations occurring at construction sites. The rural road subset contains merging situations that occur at construction site. To evaluate their approach, the authors use a 8-fold cross validation. After an equalization of the negative and positive samples in the dataset and the optimization of the features, the situation based random forest produces 9.08% false positives and a cut-in situation is predicted shortly before it starts. However, the start of a maneuver is not clearly defined in this paper.

Schmuedderich et al. propose a method to predict lane changes on highways [Schmuedderich and Rebhan, 2011]. An approach similar to [Schlechtriemen et al., 2014] is used to create the relations for each vehicle and their six surroundings, as well as to the corresponding lane. The authors also design specific contextual features including relations between vehicles. On the other hand, they do not consider any features related to the static infrastructure. More details about the features will be provided in the next chapters of the thesis.

Instead of learning the probability distribution for each feature and doing a feature reduction as proposed in [Schlechtriemen et al., 2014][Reichel et al., 2010], they use an approach similar to [Garcia Ortiz et al., 2012]. Two classifiers are applied

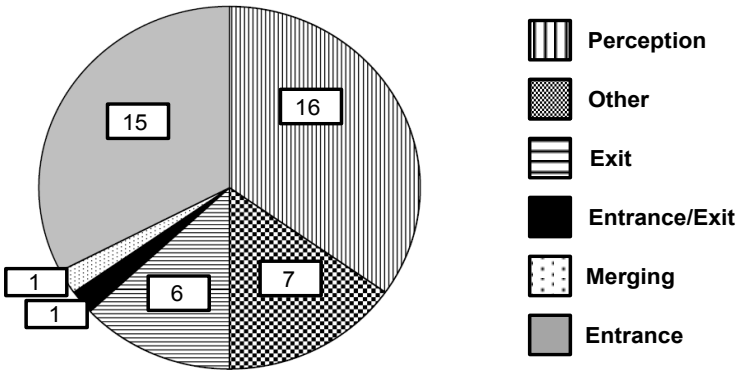


Figure 4.2: Prediction errors produced by the "Highway" model alone classified into 6 categories. The category 'Other' contains errors that cannot be attributed to systematic failures.

to represent the distribution of features for lane changes to the left and to the right respectively. A lane change maneuver is annotated positive from the moment the first wheels of the vehicle touch the lane markings to two seconds before.

This work is only published as a patent and consequently, it does not include an evaluation of the performance of the classifiers. Therefore, we evaluated this model for the prediction of lane changes to the left for our dataset using a threshold that guarantees a maximum of 4 false positives per hour. We analyzed the quality of the prediction using the same event based evaluation as [Garcia Ortiz et al., 2012]. More detailed explanations concerning the event-based evaluation will be provided in the next chapters. This model usually performs well on the highway even in the presence of occlusions or errors in low level perception. A cut-in maneuver is predicted approximately 2s before it starts. Fig.4.2 shows an error analysis which has also been published in [Bonnin et al., 2014b]). For each right predecessor of the ego-vehicle that is predicted, we counted the false positives and false negatives and categorized them with respect to their causes. This figure confirms the conclusion of [Garcia Ortiz et al., 2012], that many errors arise from perception (e.g. wrong lane assignments or limited sensor range), but also that a large proportion are caused by the context. This model predicts vehicles irrespective of the static environment. The behavior of a driver, in contrast, will change drastically for a short period of time while a specific scenario occurs. Such specific behaviors, e.g., lane changes at entrances, are stereotypical and should be predictable with a specialized classifier.

The different approaches designed to predict lane changes on highways show

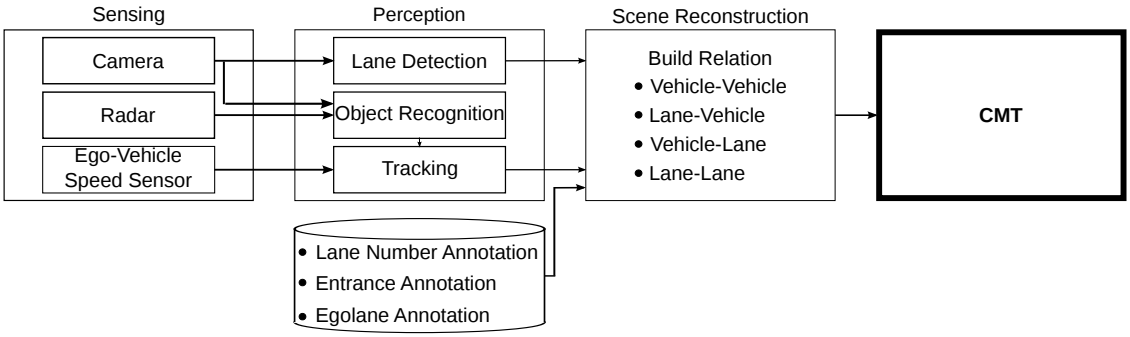


Figure 4.3: Overview of the system architecture for lane change behavior prediction.

that the behavior of a driver is influenced by her context. A vehicle changes lane because of a slow predecessor [Schlechtriemen et al., 2014], [Schmuedderich and Rebhan, 2011] or because of contexts like merging lanes [Reichel et al., 2010]. These models learn which context variables cause which behavior via the computation of contextual features and prior knowledge. Such scenario-specific models show a very good quality in terms of accuracy of the prediction and length of prediction horizon. However, this comes at the cost of a narrow scope: the quality of these models decrease when they are applied on a non appropriate context. A solution to extend the scope of the prediction would be to design different models for different specific contexts and combine them. In the previous chapter, we introduced the Context Model Tree (CMT), especially designed to combine multiple models. In this chapter, we use the highway as an example to demonstrate the value of this framework. We build a CMT for the highway to predict if an entity comes into the way of the ego-vehicle when changing lane to the left, see also [Bonnin et al., 2013][Bonnin et al., 2014b].

In the following, we use the model from [Schmuedderich and Rebhan, 2011] as the top node. In this chapter, we have chosen to limit the CMT to predict lane changes to the left, which means that we will not consider the prediction of the model for changing lane to the right in the following.

As an example for additional nodes, we implement two models to predict lane changes at entrances. Both are used as sub-nodes of the “Highway” model. These three models are separately presented in the following sections after the presentation of the system that has been built to pre-process the data.

4.2 System Overview

Fig.4.3 provides an overview of the system architecture that has been built to extract the required information from the recorded data and do the prediction of the surrounding vehicles on highways. Part of this system where reused from previous projects.

4.2.1 Sensing

The sensing module represents the sensory inputs of the system.

Color images are captured by a stereo camera system mounted behind the windshield of the recording vehicle and stored with a frame rate of 10Hz. The stereo cameras have a baseline of 24cm with a resolution of 1280×1024 and an opening angle of 31°. The images are rectified and sent to the perception module.

A radar (Bosch LRR3) is mounted at the front of the vehicle in a way so that its sensing area is overlapping with the camera. The radar data consists of lateral and longitudinal positions of radar targets and velocities.

The speed of the ego-vehicle is extracted from an inertial sensor.

4.2.2 Perception

The system uses the camera based lane detection component of the Honda production lane keep assistance system (LKAS) (2009 Accord).

Vehicles are detected on the highway with the radar. The object detection component synchronizes the radar and camera images, i.e. all radar targets from one radar time frame are visible (if not occluded by other objects) in the corresponding camera image frame. Radar data and camera images are then fused to have a better estimation of the lateral position of the detected objects. More details about this component can be found in [Nishigaki et al., 2012].

The speed of the ego-vehicle and the detected object position information are sent to the tracking component which applies a Kalman Filter to further stabilize them over time and to estimate object velocity and acceleration.

4.2.3 Database of Annotations

To predict a lane change at entrances on highways we need additional information about the static infrastructure of the road, like number and spatial alignment of lanes and the start and end of an entrance. In this work this information has been manually annotated. Alternatively, it could be obtained from a Digital Map

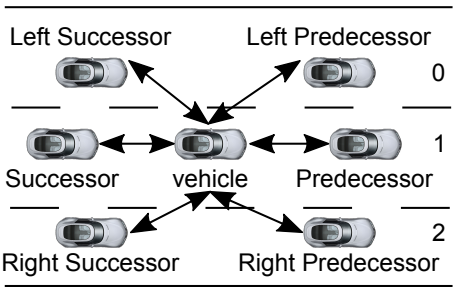


Figure 4.4: Car-to-car relations. Relations between a vehicle and its surroundings.

[Mccall et al., 2005][Wang et al., 2012] or computed based on sensor data [Michael and Fernando, 2010] [Liu et al., 2008][Takagi et al., 2006] [Thuy and Puente Leon, 2010][Kuehnl et al., 2013]. We used manual annotations to focus the evaluation on the quality of the prediction rather than that of the sensory systems.

4.2.4 Scene Reconstruction

The scene reconstruction module uses all the information provided by the perception module and the database of annotations to create relations between traffic entities and contextual information.

On highways, a driver behavior is influenced by her direct surroundings. For example, a driver might change lane to the left when approaching a slower predecessor. To recreate these relations of neighborhood, this module first connects lanes to each other in order to know which lane a driver can enter or not. Then, each vehicle is assigned to its corresponding lane to know in which lane a driver is driving. Finally, this module connects each vehicle to its left predecessor (LP), predecessor (P), right predecessor (RP), right successor (RS), successor (S), left successor (LS) and vice versa as shown in Fig.4.4.

4.2.5 Context Model Tree

The prediction module uses this reconstructed scene to compute features which are used to train classifiers in order to predict if a vehicle is going to change lane. This chapter is mainly focusing on this last module. This module contains multi-prediction models ordered in a CMT as shown in Fig.4.5. In the following we present the three different models to predict lane change maneuvers on highways that are used in the CMT to predict if a vehicle is coming into the way of the ego-vehicle.

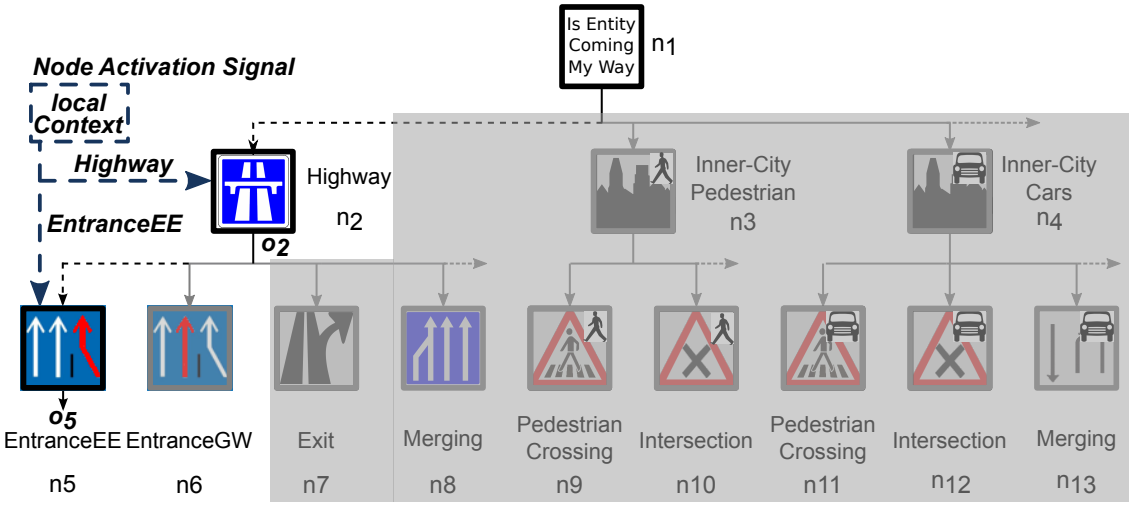


Figure 4.5: Visualization of the CMT for the highway. Each rectangle represents a node of the hierarchy. The "Highway" node n_1 is active to predict any entities driving on highways. Grey lines connect inactive sub-nodes to the "Highway" node and black dotted lines the active ones. The "Entrance Enter" node n_5 is activated when an entity drives on the entrance lane. It receives the output o_i from the "Highway" node and returns its own output. The nodes that are in the grey area are not presented in this chapter.

The top node should be a generic model which is always activated. In our application, the "Highway" model will be the top node of the CMT. The "Entrance Enter" model and the "Entrance Giveway" model are more specific models and will be sub-nodes of the "Highway" node. These nodes get activated using the local context of a traffic participant. For example, the "Highway" node is activated for each entity driving on the highway. The "Entrance Enter" node is activated when the ego-vehicle is approaching an entrance and the predicted vehicle is driving in the entrance lane. For the same traffic participant, only one node of the same level can be active at a time. An example of when to combine two nodes is the combination between the "Highway" model and the "Entrance Enter" model to predict the RP of the ego-vehicle and a combination between the "Highway" model and the "Entrance Giveway" model to predict the RP when the ego-vehicle is approaching an entrance. In case an entrance is not detected or a vehicle is assigned to a wrong lane, the specific node will not be activated. By always having the top node active, the CMT ensures that a traffic participant will always get predicted.

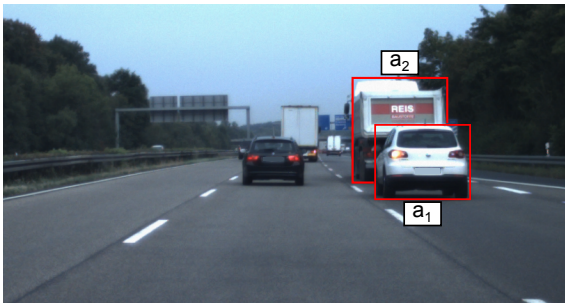


Figure 4.6: Vehicle a_1 is going to change lane to the left in front of the ego-vehicle because of a slow predecessor a_2.

There can be situations where for the same traffic participant, two active nodes provide opposite prediction. In such a case, the combination functions ensure an accurate final prediction. For each vehicle that is predicted by the "Highway" model, the classifier produces a confidence output o_j. When the sub-node is activated, it applies the maximum function between the confidence c_i of its classifier and the confidence o_j sent by the top node to get the final prediction output o_i.

The output of the last active module could then be used to decide about, e.g. warning the driver or reducing velocity.

4.3 "Highway" Node

4.3.1 Scenario

We are interested in predicting cut-in maneuvers on highways. We have chosen to use the context based model designed by Schmuedderich et al. because this model can predict a cut-in maneuver up to 2s before it starts without making too many errors as we will show in the experiment section.

The authors, after looking at videos of vehicles driving on highways, have observed that most of the time, a vehicle changes lane when approaching a slower predecessor in order to overtake it, as shown in Fig.4.6. However, even if the most important feature of the model uses the relation of a vehicle to its predecessor, there are many elements that a driver has to consider before changing lane which could influence her decision. For example, the driver must check if there is enough space in the neighboring lane, if the left successor is not approaching too fast and if she has time to change lane. The authors have designed a model which takes into account all these elements that influence the decision of a driver in order to

predict if a driver is going to change lane when approaching a slower predecessor. They define relations between a vehicle and its different surrounding vehicles as well as the static infrastructure like lane markings. These relations are used in the design of a set of indicators that provide variables which are observable before the predicted behavior starts. We present some of the different indicators:

1- $fitting-left-gap$ uses the current position, velocity and length of a vehicle a_i to estimate if a gap on the left neighboring lane is now or soon available, depending of the size of the gap between its left successor and predecessor, as well as their corresponding position and velocity.

2- $approaching-predecessor$ computes the TTC between a vehicle a_i and its predecessor.

3- *tailgating* analyzes the evolution of the safety distance between a_i and its predecessor, computing the ratio of the longitudinal distance between a_i and its predecessor, divided by the velocity depended safety-distance.

4- $evade-to-gap$ evaluates if the reduction of the safety distance is due to the possibility of fitting the left gap.

5- $accelerate-to-gap$ computes the acceleration or deceleration of a_i that is necessary to enter the left gap.

6- $accelerate-despite-ttc$ indicates that a driver is accelerating despite a high risk of collision with its predecessor.

7- $successor-approaching$ computes the TTC between a vehicle a_i and its successor.

8- $free-lane$ indicates if a_i is following its predecessor or if the lane ahead is free.

9- $free-lane-and-let-overtake$ indicates that a vehicle a_i which does not have any predecessor might change lane because of a free gap on the left and a successor approaching too fast.

The values of these indicators are normalized between $[0; 1]$ by means of a Fermi function to allow for the combination of indicators and train a classifier on annotated data to learn the different configurations for predicting a lane change maneuver.

This model is a generic model designed to predict cut-in behaviors targeting for a wide scope and a long prediction time horizon. However, the error analysis that is presented in the introduction of this chapter, see Fig.4.2, shows that the model is inaccurate for very specific contexts. For that reason, we use this model as the top node of the CMT, which is by default always active, and design different models for the different specific contexts and place them as sub-nodes. The model serves as basis for an ongoing product development, therefore it can not be described in more details here. The models that are described in the following are based on a similar idea and give a better understanding of how this model is built. For more information, the interested reader is kindly referred to the patent [Schmuedderich and Rebhan, 2011].

4.3.2 Model Conditions

The "Highway" node n_1 is active for all vehicles driving on the highway in the ego-vehicle's surrounding within 100m distance. In this CMT the "Highway" node is, by default, always active.

4.4 "Entrance Enter" Node

4.4.1 Scenario

The "Entrance Enter" node represents an entrance scenario where, for a short period of time, there is an additional lane and an entering vehicle has to leave this lane before it ends, even if there is a dense traffic and not much free space, as shown in Fig.4.7. To find out if the vehicle will enter the highway before or after its left successor, we have designed a specific model [Bonnin et al., 2012]. We have designed a set of complex features that are normalized to a similar range in order to then train a classifier.

4.4.2 Feature Definitions

A vehicle with ID i has a currently measured position $p_i = (p_{i_x}, p_{i_z})$, a velocity $v_i = (v_{i_x}, v_{i_z})$, an acceleration $acc_i = (acc_{i_x}, acc_{i_z})$[1] and a length $len_i = 4m$.

We compute the distance between two vehicles a_i and a_j according to the following equation:

[1] In this thesis, x is directed to the right, z directed to the front and y directed upwards.

Figure 4.7: Three exemplary scenes where vehicles being driven in the entrance lane on the highway and change lane to the left in front of the ego-vehicle.

$$d(a_i, a_j) = \begin{cases} (p_{i_z} - p_{j_z}) + len_i & \text{if } (p_{i_z} - p_{j_z}) < 0 \\ (p_{i_z} - p_{j_z}) - len_i & otherwise \end{cases} \tag{4.1}$$

The features are designed to answer two questions: first, does the vehicle a_i have to enter the highway, secondly can it enter according to the free space and the other vehicles on its left. In the following, we present the different features.

1- *TimeToContact* (TTC) between a_i and its LS a_j allows to estimate the time before the two vehicles collide. This feature is manually set to 1 after the scaling if $TTC(a_i, a_j) < 0$:

$$TTC(a_i, a_j) = d(a_i, a_j)/(v_{j_z} - v_{i_z}) \tag{4.2}$$

2- *TimeToGap* is the time needed for a_i to reach the left gap between its LP and LS (a_k, a_j). If $TTG = 0$ or no LP and LS exist, timeToGap (TTG) is manually set to 1 after the scaling:

$$dFront = p_{k_z} - (p_{i_z} + len_i)$$
$$velGapFront = v_{i_z} - v_{k_z}$$

$$TTG(a_i, a_j, a_k) = \begin{cases} 0 & \text{if } dFront >= 0 \\ dFront/velGapFront & \text{if } dFront < 0 \end{cases} \tag{4.3}$$

3- *DiffVelocity* is the difference of velocity between a_i and its LS a_j. Before there is any significant TTC, the feature gives early information about a

possible lane change. If $DiffVelocity <= 0$ or cannot be computed because no LS, the feature is manually set to 1 after the scaling:

$$DiffVel(a_i, a_j) = v_{j_z} - v_{i_z} \text{ if } TTC(a_i, a_j) > 0 \tag{4.4}$$

4- $SizeLeftGap$ describes if a_i fits into the left gap between LP and LS (a_k, a_j) according to the length of the gap. This feature is manually set to 1 after the scaling if there is no LP or LS.

$$SizeGap(a_i, a_j, a_k) = p_{k_z} - p_{j_z} - len_i \tag{4.5}$$

5- $AccelerateToGap$ is an indicator to compute the acceleration required by a_i to enter the gap between LP and LS (a_k, a_j). It is a multiplication between the scaled TTG, the scale $SizeGap$ and the scaled acceleration of a_i.

$$\begin{aligned} AccToGap(a_i, a_j, a_k) = TTG_{scaled}(a_i, a_j, a_k). \\ SizeGap_{scaled}(a_i, a_j, a_k).acc_{scaled}(a_i) \end{aligned} \tag{4.6}$$

6- $PrevPredGW$ is the prediction from the previous timestep of the LS a_j by a copy of an "Entrance Giveway" model, which describes if a new gap will appear that would allow a_i to change lane:

$$previousPrediction(a_j) = f_{giveway}(a_j, t-1) \tag{4.7}$$

7- $TimeToEnd$ (TTE) is the time of a_i before reaching an end of the entrance.

$$TTE(a_i, T_{end}) = \triangle t(a_i, T_{end}) \tag{4.8}$$

4.4.3 Feature Scaling

The features of the "Entrance Enter" model described above have been designed based on expert knowledge. For example, a vehicle being driven in the entrance lane has to change lane before the end of the lane. This behavior is described by the $TimeToEnd$ feature. A feature alone does not suffice to predict a lane change behavior. For example, a very small value of the $TimeToEnd$ does not necessary mean that the vehicle is going to change lane if there is a left successor approaching

too fast or the size of the left gap is too small. To be able to handle many situations we use a multitude of features. Because the combination between features is not an easy task, we use a learning classifier.

We chose to train a single layer perceptron (SLP). We decided to use a simple classifier because the features used in inputs are very complex, they use many contextual scene information, which should allow a simple classifier to perform well. Additionally, adding models to the CMT might not come at a high cost when using very simple classifiers.

To fit the characteristics of an SLP, all features are scaled to a similar range $[0; 1]$. A feature like the TTC can be within [inf,+inf], but after manual inspection of the values, we observed that the TTC of a vehicle which does change lane is always positive and below 8s. For that reason, it makes sense to only preserve the variability of values inside this range. To define this range, for all features a minimum and a maximum threshold have been defined. The maximum threshold defines the point with 90% activity and the minimum threshold the point for 10% activity. After defining these thresholds, we choose a Fermi function to scale the features because it limits the impact of the choice of the thresholds compared to, e.g., a piecewise linear function, see equation (4.9).

$$\text{fermi}(x,min,\, max) = 1/(\exp((x - \mu)/k) + 1) \tag{4.9}$$

The parameter $\mu = \min + (\max - \min)/2.0$ is used to center the results in the interval defined by the thresholds and the parameter $k = (\max - \mu)/log(1/0.9 - 1)$ changes the slope of the curve to reach 90% at the maximum value and 0.1% at the minimum.

4.4.4 Single Layer Perceptron

We want to investigate if it is possible to choose features which are linearly separable by carefully designing them in order to use a simple classifier such as a SLP [Bishop, 2007]. An SLP is an algorithm for supervised classification. It is a linear classifier which uses a linear predictor function combining a set of weights with the feature vector to compute a confidence output. Fig.4.8 gives an example of the SLP we use in our models. Each input x_j is multiplied with a weight w_j and a sigmoid is applied to the sum of the product of input and weight, to get a confidence output c, see equation (4.10).

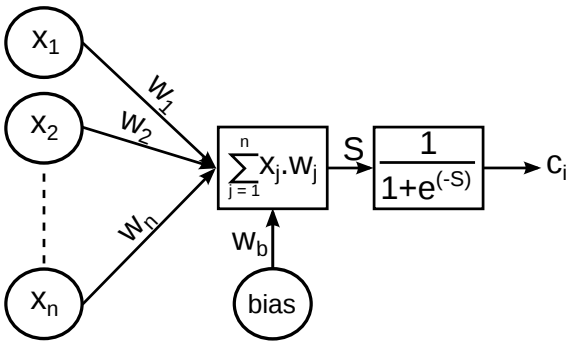


Figure 4.8: Single Layer Perceptron. x_1 to x_n and the bias are the input neurons used to train the classifier. Each input x_j is connected to a weight w_j. Each input value is multiplied with its corresponding weight value. The classifier does the sum of all the products and applies a sigmoid function.

$$c = 1/(1 + e^{-\sum_{j=0}^{j=n} w_j.x_j}) \tag{4.10}$$

During the training, the SLP must learn decision boundaries between positive and negative examples of the dataset in order to do correct classification. The aim is to get the network to generalize to classify new inputs appropriately.

We first randomly initialize all the weights. Then, the first step consists of iterating over all the examples of the dataset, and compute the confidence c using the equation (4.10). At each iteration, we update the weights according to the equation (4.11). If the confidence output c does not correspond to the annotated one $label$, a weight is updated by multiplying a learning rate α with the prediction error $(label - c)$. The learning rate α determines how smoothly we shift the decision boundaries. The member $c.(1 - c)$ of the equation (4.11) is used to dynamically update the value of the learning rate α. It increases the learning rate for results around 0.5 and decreases the learning rate for results close to 0 or 1.

$$\triangle w = c.(1 - c).\alpha.(label - c).x \tag{4.11}$$

The second step consists of computing the cost function according to the equation (4.12). The cost function value represents how well the neural network performed to map training examples to correct output.

$$E = 1/2 \sum_{m=0}^{m=numExamples} (label_m - c_m)^2 \tag{4.12}$$

The value $numExamples$ is the number of examples present the training dataset.

Finally, we normalize the cost function by computing the root mean square error (RMS) according to the equation (4.13):

$$finalError = \sqrt{E/numExamplesDataset} \tag{4.13}$$

The goal of an SLP is to optimize the cost function. The optimization process consists of repeating the first and second steps until the number of epochs (iterations) has reached 1000 or the output of the RMS is below a fixed threshold $\theta = 0.0001$.

4.4.5 Model Conditions and Combination Function

The "Entrance Enter" node is active for all vehicles being driven in an entrance lane in the ego-vehicles surrounding within 100m distance.

For this scenario, the combination function shown in equation (4.14) is applied to all predicted vehicles being driven in the entrance lane. All predictions of the "Highway" node for vehicles not being driven in this lane will be preserved in the final confidence value. The final prediction for changing lane to the left will be the output of the maximum function, applied on the confidence value of both nodes[2]. In this situation, the results of the "Highway" node and the "Entrance Enter" node have to be combined in order to compensate for prediction errors of the "Highway" node. For example, the "Entrance Enter" node contains features indicating that there will be an additional lane where the vehicles will enter. The "Highway" node will produce an error and will predict that the vehicle is not going to change lane if there is no slower predecessor.

The error analysis shown in Fig.4.2 revealed that the "Highway" model produces false negative errors when predicting lane changes at entrances. This means that the model does not predict lane changes for this specific context. This reason motivates the choice of using the max function for the combination, in order to increase the number of true positives without increasing the false positives.

$$o_i = max(o_j, c_i) \tag{4.14}$$

[2]We found that the confidences of two classifiers have a similar range, therefore we used an identity scaling function.

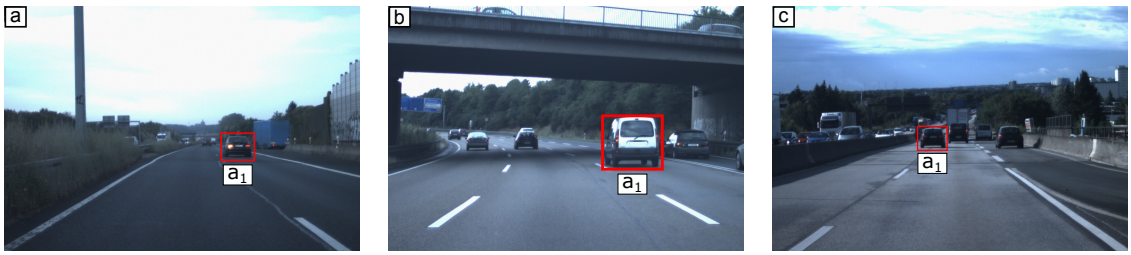


Figure 4.9: Three exemplary scenes where vehicles being driven in the left neighboring lane of the entrance on the highway and change lane to the left because of entering vehicles.

4.5 "Entrance Giveway" Node

4.5.1 Scenario

The "Entrance Giveway" node represents a scenario where a vehicle being driven in the left neighboring lane of an entrance lane might give way to free the lane for the entering vehicle, as shown in Fig.4.9. To find out if the vehicle will give way to the left before or after its LS, we have designed a specific model [Bonnin et al., 2012]. As for the "Entrance Enter" model, we have designed a set of complex features that are normalized to a similar range in order to then train a classifier.

4.5.2 Feature Definitions

The features are designed to answer two questions, does vehicle a_i have to change lane due to the RP, and can it change lane to the left according to the free space and the other vehicles on its left. The model uses the same features 1-6 as the "Entrance Enter" model.

1- $TimeToContact$, eq. (4.2)

2- $TimeToGap$, eq. (4.3)

3- $DiffVelocity$, eq. (4.4)

4- $SizeLeftGap$, eq. (4.5)

5- $AccelerateToGap$, eq. (4.6)

6- $PrevPredGW$, eq. (4.7)

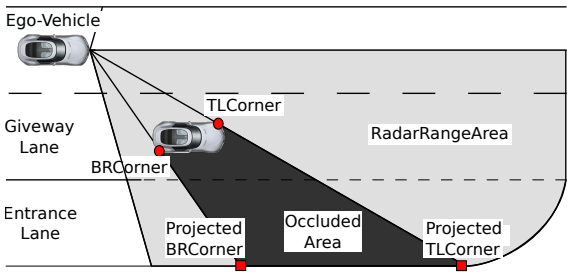


Figure 4.10: Computation of Feature VisCoef. "RadarRangeArea" is the right half of the area visible to the radar. "OccludedArea" is the area occluded by the vehicle in the giveway lane.

7- $VisCoef$, the Coefficient of Visibility used to approximate the percentage of the entrance lane hidden by a_i as shown in Fig.4.10. We assume that the behavior of a vehicle is influenced by its surroundings. However, in some cases, it can be very challenging to predict a lane change behavior, if none of the surroundings are detected because hidden by other vehicles. This feature is used to evaluate the risk of not detecting a vehicle by computing the occluded area. To compute the occluded area, we compute the projection from the radar passing the top left corner and bottom right corner of a_i onto the right edge of the entrance lane in the range of the radar.

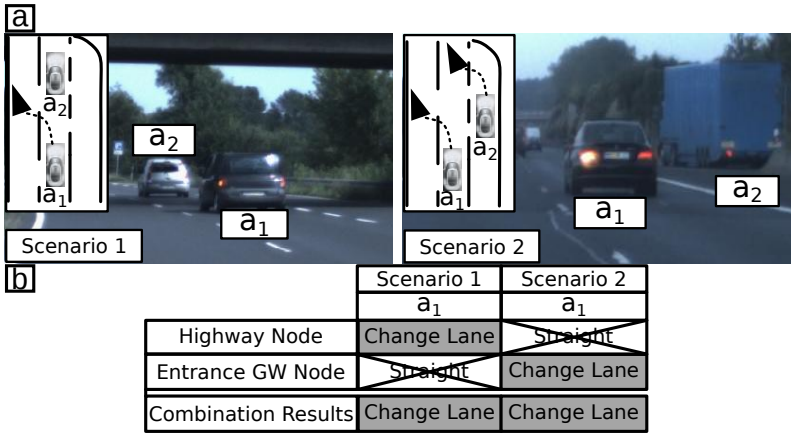
$$VisCoef(a_i) = Size(RadarRangeArea) - Size(OccludedArea) \tag{4.15}$$

8- $PrevPredE$ is the prediction from the previous timestep of the RP a_l by a copy of an "Entrance Enter" model, which describes if a new gap will appear that would allow a_i to change lane:

$$PrevPredE(a_i) = f_{entranceEnter}(a_l, t-1) \tag{4.16}$$

4.5.3 Feature Scaling

The features of the "Entrance Giveway" model have been designed based on expert knowledge. For example, a vehicle being driven in the left neighboring lane of the entrance lane might change lane to give the space to the entering vehicle. This behavior is described by the $TimeToContactRP$ feature. One feature alone does



	Scenario 1	Scenario 2
	a_1	a_1
Highway Node	Change Lane	~~Straight~~
Entrance GW Node	~~Straight~~	Change Lane
Combination Results	Change Lane	Change Lane

Figure 4.11: Two entrance scenarios on the highway. (a): Typical highway and giveway scenarios occurring in the giveway lane. Dashed arrows represent the future behavior of the vehicles. (b): Prediction results. The Table visualizes the prediction result of the "Highway" node, the "Entrance Giveway" node and the combination results of both models for the example scenarios. Grey fields indicate a correct prediction, crossed out fields indicate wrong predictions

not suffice to predict a lane change behavior. For example, a very small value of the $TimeToContactRP$ does not necessary mean that the vehicle is going to change lane if there is a left successor approaching too fast or, as for the "Entrance Enter" model, the size of the left gap is too small. To be able to handle many situations we use a multitude of features. In analogy to the "Entrance Enter" model, all features are scaled to a similar range with the equation (4.9) to fit the characteristics of an SLP, as shown in Fig.4.8.

4.5.4 Model Conditions and Combination Function

The "Entrance Giveway" model is active for all vehicles being driven in the left neighboring lane of an entrance lane within 100m distance and in the ego-vehicles surroundings.

For this scenario, the combination function is applied to all predicted vehicles being driven in the left neighboring of an entrance lane. All predictions of the "Highway" node for vehicles not driving in this lane will be preserved in the final confidence value. The final prediction will be the output of a maximum function, as shown in equation (4.14), which corresponds to the prediction of the node which

has the highest confidence for changing lane to the left. In this situation, the results of the "Highway" node and the "Entrance Giveway" node have to be combined in order to compensate for prediction errors of the "Highway" node. Fig.4.11 displays two exemplary situations at an entrance which motivate this choice for the combination function. In scenario 1 the lane change intention of a_1 arises from the slower vehicle a_2. In this case, the "Highway" node predicts a lane change with a high confidence which will win over the low confidence of the "Entrance Giveway" node prediction. In scenario 2 the lane change intention of vehicle a_1 is caused by a_2 entering the highway. Here, the maximum selection of the combination function makes the "Entrance Giveway" node override the wrong prediction of the "Highway" node.

4.6 Experiments

In this section, we present an evaluation of the performance and prediction time horizon of the generic "Highway" node, the "Entrance Enter" node and the "Entrance Giveway" node using real data. We show that the "Highway" model performs generally well at predicting lane change behaviors, but in specific locations, the prediction is less accurate. We show that the "Entrance Enter" model and the "Entrance Giveway" model, which use specifically tuned contextual information, perform better at those locations than the generic "Highway" model. However, such specific models do not generalize well to other scenarios. Despite getting a good performance for a single generic node, we see that this might no be the best possible performance for all situations. Therefore we want to combine the three models. In the previous chapter, we introduced the CMT framework to combine multiple models of different specificity. In this section, we will present an example of a CMT to predict lane change maneuvers of the right predecessor (RP) of the ego-vehicle. The "Highway" model is placed in the top node of the tree. The two entrance models are placed as sub-nodes under the "Highway" node as shown in Fig.4.5. We will show the benefit of using a CMT with an evaluation comparing the performances of the single models separately versus the performances of the combination of the models with the CMT.

The evaluation focuses on predicting if the RP of the ego-vehicle is going to enter the lane of the ego-vehicle for various scenarios and a long prediction time horizon.

Figure 4.12: Exemplary scenes recorded on German highways under varying weather and traffic conditions. (a): Rain Condition. (b): Fog condition. (c): Sunny condition. (d): Overexposure due to the difference between low and high lighting. (e): Dense traffic condition. (f): Artificial light in a tunnel.

4.6.1 Dataset

All tests have been done using real data recorded on several German highways (curved and straight) under varying weather (overcast, rain, fog, low sun) and traffic conditions (light, medium, and dense traffic).

We use a supervised learning method to train a classifier to predict behaviors, therefore we need to build one dataset which is used for training the classifiers and one dataset which is used only for testing.

The dataset used to train the "Entrance Enter" model and the "Entrance Giveway" model consists of 2 hours of recordings. The dataset is divided into two subsets, one sub-set to train the "Entrance Enter" model which contains 18 vehicles with 14 of them changing lane, and one sub-set to train the "Entrance Giveway" which contains 40 vehicles with 6 of them changing lane. Both models are trained with K-fold cross validation technique [Bishop, 2007]. The technique consists of dividing the dataset into K subsets to train the classifier on K-1 subsets and test it on the subset that was excluded during the training.

The dataset used for testing consists of 4 hours of driving with 783 vehicles, 40 of them perform a lane change on the highway and 33 do a lane change at entrances. To evaluate the effect of the prediction system on a realistic ADAS function, we

Table 4.1: Feature Definition

N°	Feature Name	[min;max] Scaling Thresholds
1	$TimeToContact$	EE:[0;8]
2	$DiffVelocity$	EE:[0;20]
3	$TimeToGap$	EE:[0;15]
4	$SizeLeftGap$	EE:[0;8]
5	$TimeToEnd$	EE:[0;20]
6	Acceleration a_i	EE:[0;0.8]

restrict the prediction to vehicles within a certain distance, respectively within a certain TTC window. Vehicles that have a TTC (distance divided by velocity difference) larger than 10s or a TTG (distance divided by ego velocity) larger than 2s will not be predicted, because they do not represent a danger for the ego-vehicle. This leads to a dataset of 695 vehicles with 29 lane change maneuvers on the highway and 17 at entrances.

To train and test a classifier we need to compute these features and provide the ground truth in each timestep. That means each timestep has a corresponding annotation as a negative or positive example. On the highway we are interested in predicting if a vehicle will change lane to the left. A vehicle is considered as changing lane to the left when its left wheel touches the lane marking. When a lane change is detected, each timestep from that moment to three seconds before is annotated as a positive example. Two seconds before and two seconds after the lane change are ignored in the evaluation because there is no obvious start or end point for a prediction, everything else is annotated as negative examples.

4.6.2 "Entrance Enter" Node

In this experiment, we present a frame based evaluation of the performance of the "Entrance Enter" model.

Scaling Features

All features are scaled to a similar range [0; 1] with a Fermi function, see equation (4.9). For each feature, Table 4.1 shows the corresponding min and max thresholds for scaling.

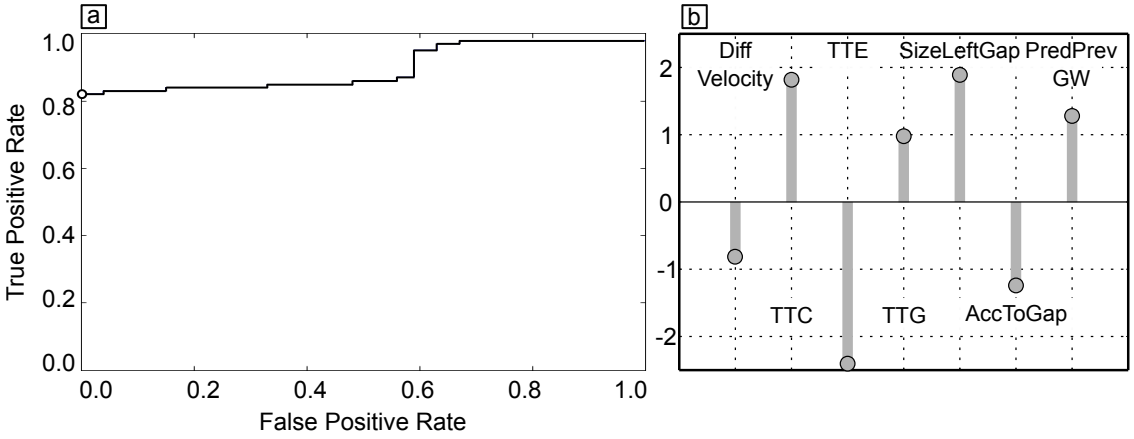


Figure 4.13: Results of the model "Entrance Enter" classifier. (a): ROC curve for the model "Entrance Enter" with 82%TPR at 0% FPR for a frame based evaluation with a 3-fold cross validation method. (b): Weight values after the training of the SLP.

Single Layer Perceptron

For this model, we used an SLP with 7 input neurons and a bias set to 1, a learning rate and a sigmoid function to activate the output. After some tests, we fixed the learning rate value α to 0.002. We trained and tested the model with a 3-fold cross validation method on the training dataset of the "Entrance Enter" model.

Fig.4.13.(a) presents the ROC curve for the model "Entrance Enter". The result for this model is 80% TPR at 0% FPR. Fig.4.13.(b) shows the values of the weights for the different inputs after the classifier has been trained. The weight of a feature indicates the degree of importance attached to it. A feature, for which the weight has a high absolute value has a big influence during training, whereas a weight with a value close to zero has almost no influence, and the feature might not be used at all.

Fig.4.13.(b) shows that the feature that has the biggest impact on the training is the TTE with a high negative value. A positive weight indicates a positive correlation between the scaled feature value and the frequency of cut-in maneuvers in the training data. A negative weight indicates an inverse correlation. For example, a high TTC between the predicted vehicle and its LS maps to a high scaled input value which most of the time occurs for positive examples (high TTC means that the LS is far away from the cut-in vehicle), whereas a low TTE maps to a high scaled input value which most of the time occurs for positive examples (the vehicle

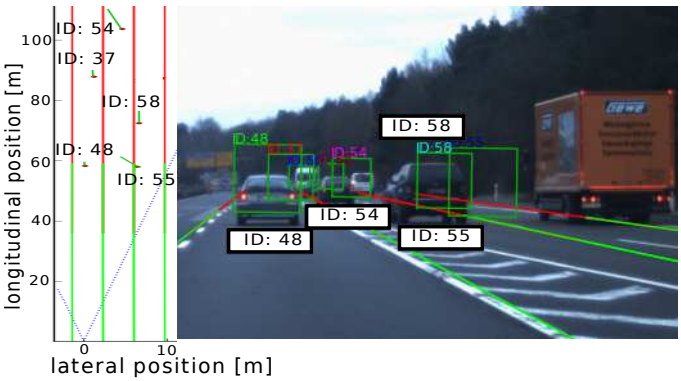


Figure 4.14: Example of a scene for which the model "Entrance Enter" is wrongly active. The left part of this image shows a bird's-eye view (BEV) with the ego-vehicle at the x-position 0, the lane markings and the radar targets with their relative velocity. The scene on the right shows the current scene in which vehicles 55 and 58 are assigned to the wrong lane because of the zebra area.

has to change lane because the lane is ending). We can also see that all other features have a very similar impact.

One common prediction error of the model is visualized in Fig.4.14. In this scene, the entrance is made of two lanes, the right most lane is the entrance lane and the left neighboring lane is merging with the highway. The vehicle 55 is driving in the merging lane but the picture shows that it is detected on the entrance lane by our perception system. The ego-lane and the merging lane are separated by a dashed area which has the same width of a lane, and because there are only two lanes on the right of the ego-vehicle, the dashed area is considered as the giveway lane and the real giveway lane as the entrance lane. The model cannot handle errors caused by the perception and the scene reconstruction.

4.6.3 "Entrance Giveway" Node

In this section, we present a frame based evaluation of the performance of the "Entrance Giveway" model trained for the giveway scenario.

Table 4.2: Feature Definition

N°	Feature Name	[min;max] Scaling Thresholds
1	$TimeToContactRP$	GW:[0;15]
2	$TimeToContactLS$	GW:[0;15]
3	$DiffVelocityRP$	GW:[0;15]
4	$DiffVelocityLS$	GW:[0;20]
5	$TimeToGap$	GW:[0;15]
6	$SizeLeftGap$	GW:[0;15]
7	Acceleration a_i	GW:[0;0.8]
8	$VisCoef$	GW:[0;75]

Scaling Features

As for the model "Entrance Enter", all features have been scaled to a similar range using a Fermi function, see equation (4.9). For each feature, Table 4.2 also shows the corresponding min and max thresholds for scaling.

Single Layer Perceptron

The model uses an SLP with 10 input neurons and a bias set to 1, a learning rate and a sigmoid function, see equation (4.10), to activate the output. After some tests, we fixed the learning rate α to 0.001.

Fig.4.15.(a) introduces the ROC curve for the model "Entrance Giveway". The result for this model is 57%TPR at 0% FPR.

Fig.4.15.(b) shows the values of the weights for the different inputs after the classifier has been trained. We observe that some of the weights are negatives and some are positives. As we explained in the model "Entrance Enter", a positive weight indicates a positive correlation between the scaled feature value and the frequency of cut-in maneuvers in the training data and a negative weight indicates an inverse correlation. We can also see that three of the features have a very small weight value. However, even with a low value, the features might still have a negative influence during the training. The number of features gives the dimensionality of the classifier. The more features are used, the more the dimensionality increases, and the more complex it gets for the classifier. This is why, for such a case, it might be worth considering using a feature optimization method [Guyon, 2003] in order to reduce the dimensionality of the classifier with for example principal component analysis (PCA) methods [Tipping and Bishop, 1999]. We chose to not apply such

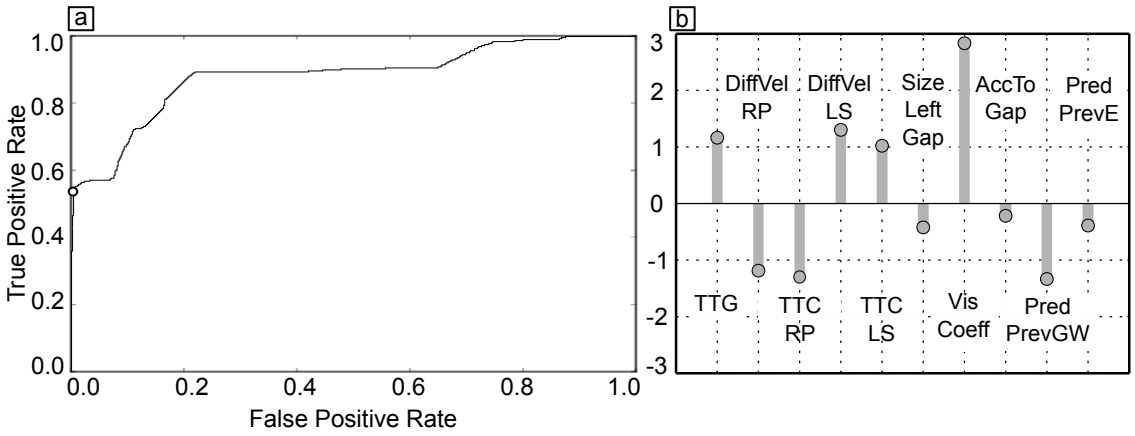


Figure 4.15: Results of the model "Entrance Giveway" classifier. (a): ROC curve for the model "Entrance Giveway" with 57%TPR at 0% FPR for a frame based evaluation with a 3-fold cross validation method. (b): Weight values after the training of the SLP.

methods because the prediction errors of the classifier were all explainable, and we did not expect the results to be better without them.

One common prediction error is visualized in Fig.4.16. In this situation vehicle 268 driving in the giveway lane is faster than truck 262 driving in front and it will change lane to overtake the truck. There is no vehicle being driven in the entrance lane. This example gives false negatives because this is not a scenario the model "Entrance Giveway" has been trained on. Such scenarios can be predicted by the "Highway" model being the top node of the CMT.

4.6.4 Context Model Tree

Event Based Evaluation

We use an event based evaluation in order to show how the system would interact with a driver. When building a system, it is important to consider user's acceptance. As our system should warn the driver when there is a risk of collision with a right predecessor (RP) that wants to enter its lane, we want to know how often the system will send an incorrect warning to the driver. We will show that our CMT produces fewer errors than the "Highway" model alone. In the following we compare the performances of the "Highway" model alone to the CMT.

In contrast to the commonly applied frame-based evaluation, an event-based evaluation considers the temporal grouping of system signals and the relevance of

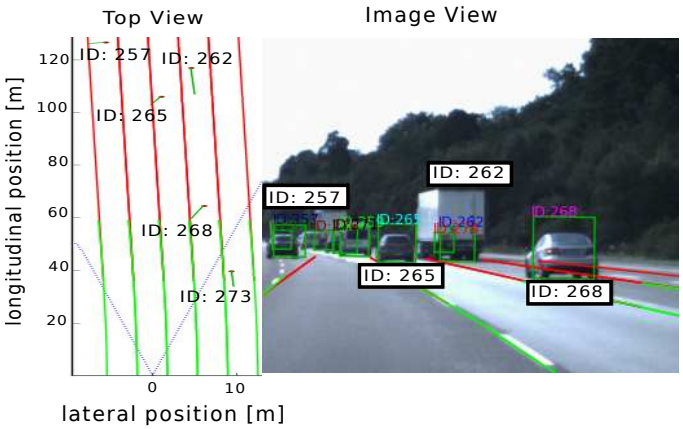


Figure 4.16: Example of a scene for which the model "Entrance Giveway" is wrongly active. The left part shows the BEV, the lane markings and the radar targets with their relative velocity. The scene on the right shows that vehicle 268, being driving in the giveway lane, will change lane to overtake the truck in front of it, not because of a vehicle on the entrance lane.

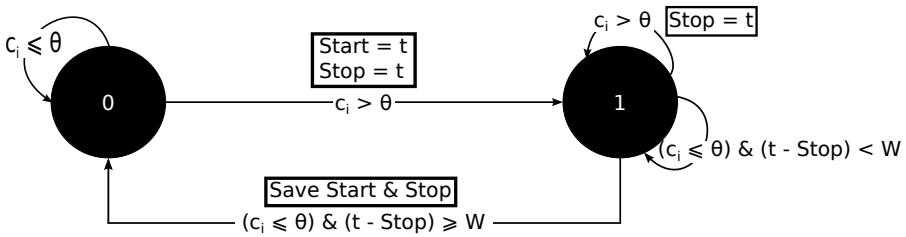


Figure 4.17: State machine built to count events. This state machine is used to count the number of events according to different thresholds θ. States are black circles and transitions are black lines connecting the states with each other. c_i is the confidence and θ the threshold. The white rectangles represent the different actions done during the different transitions. The current timestep is symbolized by letter t and W is a time window.

a predicted vehicle for the driver of the ego vehicle. This evaluation consists of two parts. First we count events. We consider the consecutive timesteps in which for each vehicle the prediction is above a threshold θ as one event (prediction gaps of size smaller than 1s are grouped into the same event). Then, we compute the events and then compare those with the groundtruth annotation at the same timesteps.

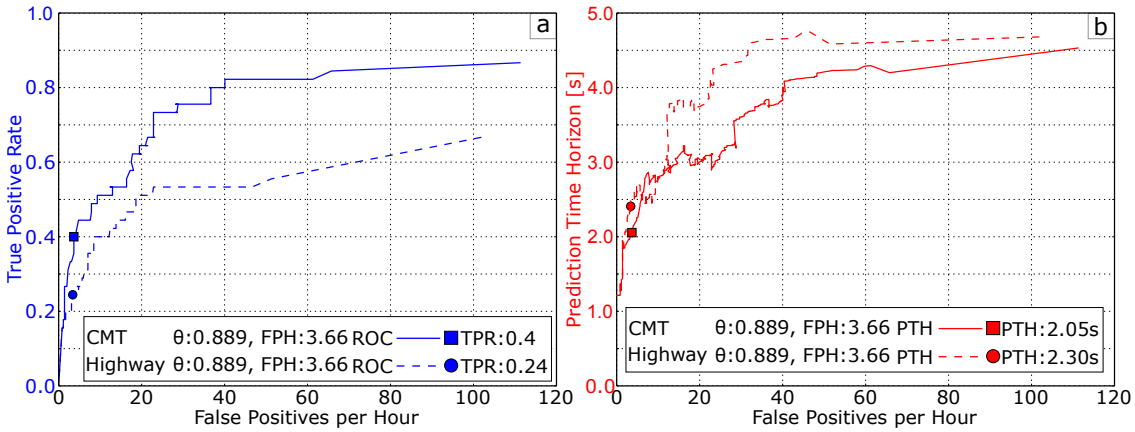


Figure 4.18: Performance Comparison between the CMT and the "Highway" model. a): Event based analysis. Blue solid and dashed lines show the True Positive Rate against false positives per hour (FP/H) for predicting that the right predecessor of the ego-vehicle will change lane to the left, for the CMT and "Highway" model respectively. Blue rectangle and circle show the best FP/H for both curves according to our quality criterion and their corresponding threshold theta. b): Prediction time horizon. Red solid and dashed lines show the average prediction time horizon (PTH) against FP/H. Red rectangle and circle represent the PTH at the same threshold for the CMT and the "Highway" node respectively.

If the event overlaps with any timestep labeled to 1, it is counted as a true positive (TP). If it overlaps with any timestep labeled to 0, it is counted as a false positive false positive (FP).

1- Grouping System Signals To Events

For each vehicle at each time t the system provides a confidence value for a lane change.

An event starts for one vehicle whenever its "lane change left" confidence c_i exceeds a threshold θ and an event finishes when it falls back below θ.

To count the events, we use the state machine presented in Fig.4.17. A new state machine is initialized with state 0 for each predicted vehicle. For each time t, the confidence value c_i is compared to the threshold θ. If the confidence is above the threshold, the current time t is saved in the variables "start" and "stop" and state 1 is active. While the confidences are above the threshold

they are considered to belong to the same event. As soon as a confidence value is below the threshold, the variable "stop" is updated with the new time t and state 1 counts the number of confidences which are below the threshold. If the number of confidences below the threshold reaches the limit value W, the variables "start" and "stop" are saved and state 0 is active. Otherwise it stays in state 1 until the prediction of the vehicle ends.

Low values for W could lead to multiple events triggered by a single lane change which would trigger multiple successive warnings. Therefore it is important to find the time window W within which we will fuse multiple events. In this thesis we use $W = 1s$, which showed the best trade off between reducing the number of warnings and still allowing for the correct behavior in cases where a vehicle changes lane multiple times.

2- Classifying An Event

An event is a FP when the timesteps of the prediction event overlap only with timesteps annotated to 0 in the groundtruth. This means that a possible driver assistant system sends a warning but the right predecessor does not change lane to the left within the following three seconds. An event is a TP when the timesteps of the prediction event overlap with timesteps annotated to 1 in the groundtruth. This means that the system sends a warning and the right predecessor changes lane within the next three seconds. In the case where the system does not send any warning, an event is a true negative (TN) if no vehicle changes lane, a false negative (FN) otherwise. We plot the TPR ($TPR = TP/(TP + FN)$) against the number of FP/H for a number of different values of the threshold θ. In contrast to the commonly applied frame-based evaluation for which an ROC curve shows the TPR versus FPR, an event-based evaluation considers the temporal grouping of system signals and the relevance of a predicted event for the driver of the ego vehicle. We chose to compute the FP/H for evaluating the functionality of our system which might be more relevant for the acceptability of such a system for a driver.

Table 4.3: Prediction errors "Highway" model and CMT

Prediction Errors	"Highway" model	CMT
Perception	16	23
Other	7	11
Exit	6	4
Entrance/Exit	1	1
Merging	1	1
Entrances	15	0
Total Errors	45	40

3- Results

Fig.4.18 (blue) presents the TPR against the FP/H for the "Highway" model alone and for the CMT[3]. We observe that the CMT generally produces fewer errors than the "Highway" model. On the ROC curve, we have chosen a threshold for the "Highway" model and the CMT that guarantees a maximum of 4 false positives per hour. The ROC curve of the CMT shows for a threshold of 0.89 a FP/H of 3.66. On the highway, the FP/H of 3.38 is obtained for a θ of 0.75.

Table 4.3 shows an error analysis of the CMT at threshold 0.89. The 15 errors produced by the "Highway" model alone due to the specificity of the entrance scenario have disappeared (see Table 4.3). A number of events (6) are now predicted correctly by the CMT. However, the perception errors have slightly increased, which is due to the fact that the CMT uses additional sensor information. We also found a few cases that cannot be predicted well by the CMT ('Other'), mostly caused by non typical behaviors of traffic participants. Some false positives caused by the exit scenario have disappeared, because the CMT allows us to work with a higher threshold since the combination of specific models results in higher prediction confidence.

Fig.4.19 presents an example of an entrance scene perceived by the ego-vehicle, driving on a highway and approaching an entrance. Vehicle a_7 is in the entrance lane and will have to change lane to the left.

Our event-based analysis confirms that a combination of specific classifiers produces more accurate predictions and fewer errors compared to using only

[3]The ROC curve shows small artifacts because the grouping of events (and through this their number) changes with the threshold.

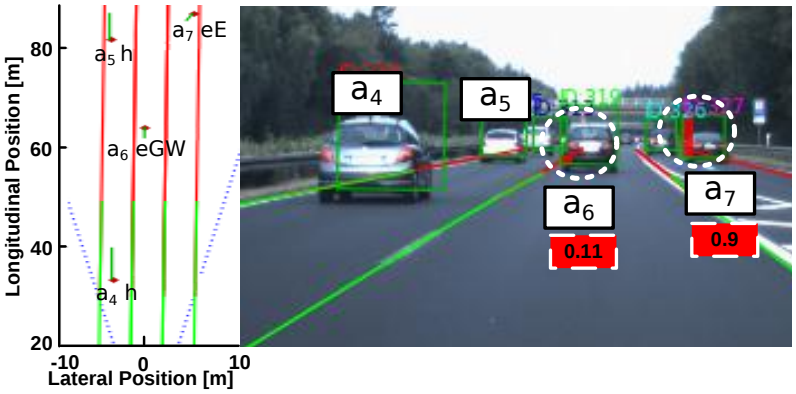


Figure 4.19: Visualization of a scene in which vehicles are predicted with the CMT. The left part shows the ego-vehicle at the origin, the lane markings and the radar targets with their relative velocity. The letter "h" denotes that the vehicle has been predicted by the "Highway" node, letters "eE" and "eGW" mean that vehicles have been predicted by node "Entrance Enter" and node "Entrance Giveway" which won the combination. The right part shows the detected vehicles. The white dashed rectangles contain the probability of a lane change for the vehicles driving in the left neighboring lane and the entrance lane. The probability is also represented by a red bar in the white circles. Vehicle a_7 will change lane to the left with a high probability and a_6 will change lane with a low probability.

a single classifier.

Time Horizon Evaluation

As mentioned above, we labeled a window of three seconds before our right predecessor has moved into our lane for the prediction target.

To evaluate the prediction horizon, we averaged the earliest points of lane change prediction over all predicted vehicles at various thresholds. The resulting prediction horizon is plotted in Fig.4.18 (red). The plot shows a general decrease of prediction horizon with increasing threshold. One reason is that the prediction of a lane change most of the time starts early with a low confidence value which slowly increases with time until reaching a maximum. This maximum is usually reached shortly before the vehicle changes lane, which explains the low value of the PTH at a high threshold.

The observable saw-tooth pattern is due to the fact that the number of predicted vehicles decreases while the threshold increases.

In the plot, we highlighted the thresholds of equal False Positive Rate where the "Highway" model alone can predict a lane change slightly earlier than the CMT. One reason is that the "Highway" model alone does not predict any lane changes at entrances, whereas the CMT predicts both, lane changes on highways and at entrances. However, vehicles at entrances are only visible for a short time due to the limited length of the lane and often lower entering speed.

Model Activation Robustness

In our current system, the "Entrance Enter" node of the CMT becomes activated via a digital map when the ego-vehicle is approaching an entrance scenario and if a vehicle is driving in the entrance lane. The "Entrance Giveway" node becomes activated when a vehicle is driving in the left neighboring lane of the entrance lane. The start and end of the entrance have been manually annotated. In future work, the manual annotation should be replaced by a commercial digital map. The accuracy of the GPS to get the position of the ego-vehicle and the accuracy of the digital map will influence the performance of our system. We evaluated the effect of an inaccurate digital map or GPS signal by adding noise, varying the start and the end of an entrance between $[-10m; 10m]$ from the original annotations. We observed a variability of the TPR, e.g. 0.42 TPR at $-10m$, 0.4 TPR at $0m$ and 0.46 TPR at $+10m$ when the "Entrance Enter" node or the "Entrance Giveway" node is activated. We also observed an increased number of FP/H, 10.44 for $-10m$, 9.87 for $-10m$ compared to 3.66 for $0m$. These numbers show that the quality of the prediction will deteriorate with a GPS error which means that we would probably use additional features in addition, for example using the algorithm of [Kuehnl et al., 2013] for lane assignment.

Frame-Based Analysis

We also evaluated our system using a frame-based evaluation. Table 4.4 shows the number of TP, FP, TN and FN produced by the "Highway" model alone and the CMT for their given thresholds (0.75 and 0.89).

The "Highway" model alone predicts the right predecessor independent of its lane (this includes entrance and left neighboring lane). While most of the FP are caused by perception, many of the FN happen because the "Highway" model cannot predict vehicles at entrance scenarios (as shown in the error analysis Table 4.3).

Table 4.4: Confusion Matrix

	Highway	Entrance Enter	Entrance Giveway	Total
TP (Highway)	151	0	0	151
TP (CMT)	113 (-38)	105 (+105)	20 (+20)	238 (+87)
FP (Highway)	149	0	0	149
FP (CMT)	54 (-95)	29 (+29)	8 (+8)	91 (-58)
FN (Highway)	683	173	100	956
FN (CMT)	722 (+39)	67 (-106)	80 (-20)	869 (-87)
TN (Highway)	33213	294	3366	36874
TN (CMT)	33304 (+91)	265 (-29)	3363 (-3)	36932 (+58)

Table 4.4 shows that the CMT produces more TP and fewer FN compared to the "Highway" model. The majority of improvements can be seen for predictions in the entrance scenario. We observe two additional effects: the number of FP on the highway decreases because we can now set a higher threshold without changing our quality criterion (see Fig.4.18). The second effect is that the number of FP at entrances increases, which is due to the fact that the entrance models suffer from an additional type of perception error. However, the positive effects are more prominent.

Computation Time Analysis

The generic computational complexity of the CMT depends on the number of nodes n, the number of vehicles v detected and the number of neighboring vehicles s of a vehicle but not on the number of models since our activation rules guarantee that only one model is active per n and v. This means $O(n.v.s)$.

For our highway system we have a constant number of nodes and the maximum number of surrounding vehicles that are considered is three. Therefore the system's complexity is only $O(v)$. On average, the "Highway" node has a computation time of 20 ms and both entrance nodes together have a computation time of 4 ms on a single core of a Xeon X5550 (2.6GHz), 8 GB RAM. The overall computation time will be between that of the "Highway" node alone and the combination of both nodes (24ms).

4.7 Discussion

As we explained in the introduction, it is very complex to design a model that has both, a wide scope and a high quality. A compromise to extend the scope is to use different context-based models with a high quality but a limited scope and combine them. In this chapter, we present a model designed to predict lane changes of the right predecessor of the ego-vehicle on highways for different context specificity. As we showed in the experiments, this model can predict a cut-in maneuver very early in advance. However, the model produces more errors when predicting cut-in maneuvers in specific locations. In that sense, we can say that this model has a limited scope. To reduce prediction errors for some of these specific locations, we have designed two context-based models to predict cut-in maneuvers at entrances on highways. To extend the scope of the prediction, we use a CMT specially designed to combine multiple models. A CMT is a tree-like structure in which models are ordered according to their context specificity, and combination rules between nodes are applied in order to obtain the most accurate prediction to know if an entity is coming into the way of the ego-vehicle. This structure allows combining specific models with high quality but narrow scope to one overall system featuring high quality and a broad scope. In the evaluation, we have shown the performance and the prediction time horizon for each of the three models separately and compare them with the performance of the CMT combining them. The evaluation demonstrates that the combination of multiple models using a CMT outperforms the single models.

The framework proposed in this thesis is designed to allow flexibility when building systems with a high quality and a wide scope. In order to show the flexibility of a CMT, the next chapters presents a concrete implementation of a CMT to predict if a pedestrian is coming into the way of the ego-vehicle when crossing the road in inner-city.

5 Scenario Model Tree for Inner-City

In the previous chapter, we showed the benefit of using a Context Model Tree (CMT) to combine multiple models to predict lane change maneuvers on highways in terms of wide scope and long prediction time horizon (PTH). To demonstrate that a CMT is a generic framework which can be used for very different scenarios, we extend the scope of the existing CMT with a concrete example to predict if an entity is coming into the way of the ego-vehicle by predicting crossing behaviors of pedestrians in inner-city in this chapter [Bonnin et al., 2014a].

Many methods exist which detect and recognize pedestrians in a scene [Enzweiler and Gavrila, 2009][Dollar et al., 2009]. Reactive systems like the system proposed by Mobileye and used by Volvo [Coelingh et al., 2010][MobilEye Ltd., 2007] have been designed to improve the protection of pedestrians. However, in some situations, such systems, that use detection only, might send a warning to a driver too late – the pedestrian is already on the road and the driver does not have time to fully avoid a collision.

To evaluate the risk of collision with a vehicle in advance, a simple method, usually used in tracking, consists of using the current dynamics of a pedestrian to estimate his future state [Musleh et al., 2010][Abramson and Steux, 2004]. Since a pedestrian usually follows a particular path, and a path is a succession of states that can be learned [Large et al., 2004], there are some methods that are developed to predict the path of a pedestrian [Chen and Yung, 2009]. Tracking and trajectory prediction methods define a position of a pedestrian for any point in time. Because pedestrians can change their walking direction very suddenly, the prediction error will grow fast with increasing time horizon. An alternative would be to look at the problem from a higher level, that is, to predict pedestrian behavior. However, very few methods target at predicting the behavior of a pedestrian [Koehler et al., 2012][Koehler et al., 2013]. For that reason, we choose to predict the behavior of pedestrians to show the benefit of using a CMT.

In the following section, we provide an overview of two prediction methods, that are the most relevant for our work. Then we will present the architecture of the system we built to do behavior prediction. To allow for an accurate behavior prediction, we propose to create one model per context to learn the different elements which are used to predict a crossing behavior. We present a model to predict cross-

ing behaviors of pedestrian on normal roads, where the pedestrian does not have the priority to cross, as well as one model to predict crossing behaviors at zebra crossings, where the pedestrian has the priority.

We use the same Context Model Tree (CMT) as the one presented in the previous chapter to combine these models. The top node contains the "Inner-City" model and will be placed parallel to the "Highway" node. The "Zebra Crossing" model will be placed as a sub-node of the "Inner-City" node. The last section will present an evaluation to compare the performances of the single models to the performances of the combination of the models with the CMT.

5.1 Existing Methods for Pedestrian Crossing Prediction

As described above, not many approaches target at predicting crossing behaviors of pedestrians. In the related work section, methods were ordered into four categories. The most relevant work concerning the prediction of pedestrians belongs to the fourth category consisting of hybrid models using the prediction of the trajectory to improve the prediction of the behaviors.

Keller et al. propose two methods to predict the lateral path of pedestrians that might cross the road or stop before it [Keller and Gavrila, 2014]. The first method uses Probabilistic Hierarchical Trajectory Matching (PTHM). This model uses optical flow to extract information about the pedestrian's legs and upper body. At each timestep, trajectories consisting of the position and motion features for a given pedestrian are stored in a database. The database contains crossing and stopping trajectories, and the model uses a matching algorithm to find which of the stored trajectories fit best the current one. The future position of the pedestrian is then derived by looking ahead on the best matched trajectory. To find the best match in the database, the authors use a probabilistic search framework. Trajectories are split into snippets of constant length, and then are transformed into a feature vector. PCA is then performed on all training vectors. A binary tree is then constructed, where at each level of the tree the value of one feature dimension, with features ordered by their eigenvalue, is tested. Each training trajectory traverses the tree and is assigned to one defined leaf. During test, a new trajectory will be mapped to a certain leaf with the same procedure. The full trajectory corresponding to the snippet assigned to this leaf is used for prediction. Given the matched trajectory, the probability of the future position of the pedestrian is computed with

a particle filter.

The second approach predicts the trajectory of a pedestrian and further uses this for behavior prediction. From successive positional information and motion features, the authors build trajectories which are then classified into crossing or stopping behavior using two separately trained classifiers. Then, a Gaussian Process Dynamical Model (GPDM) is used for path prediction. The prediction is a four step process. First, the GPDM is used for feature dimensionality reduction. It maps the learned real trajectory features into a latent space. In a second step, the model uses the latent features to do a dynamics position prediction of the pedestrian. In a third step, the model combines the estimated position with the observed one in the real world to get an estimate of the current position using a particle filter. Finally, the position predicted in the latent space is mapped back to the real world.

To evaluate the models on real data, the authors use a vehicle equipped with a stereo camera system. The dataset consists of various sequences of stopping at the curbstone and crossing behaviors, involving four different unoccluded pedestrians in three different locations at a distance range of 5-34m from the ego-vehicle. The moment of the last placement of the foot at the curbstone has been annotated as a stopping behavior, and the moment the foot touches the curbstone for the last time has been annotated as a crossing behavior. Models are trained with a leave-one-out cross-validation method. The authors compare the accuracy of the prediction of the two models to an interacting multiple model Kalman filter (IMM-KF) and a simple Kalman filter. In general, the two models described above can predict a position with better accuracy in terms of a lower lateral localization error. Thanks to the optical flow information, the models can adjust more quickly to a change of pedestrian motion. The models are also evaluated on their ability to classify a behavior, i.e. whether the pedestrian is crossing or stopping. For this evaluation, a probability for stopping is computed for a given trajectory and compared to a fixed threshold, which has been tuned for each model during training to minimize the classification error. Fig.5.1 shows the accuracy of classification versus the prediction time horizon for the PTHM, the GPDM, the IMM-KF and a human prediction. To get the human prediction, the authors asked several candidates to look at different moments in a video and classify a pedestrian as walking or stopping with an associated confidence. For an accuracy of 80%, the figure shows that the Kalman filter does a correct behavior prediction 0.09s in advance, the PTHM and the GPDM models perform similarly and predict a behavior 0.27s in advance, whereas the human prediction achieves better performance with a prediction starting 0.57s in advance. This evaluation shows that distinguishing stopping from crossing behavior is not an easy task, even for a human, but he still generally

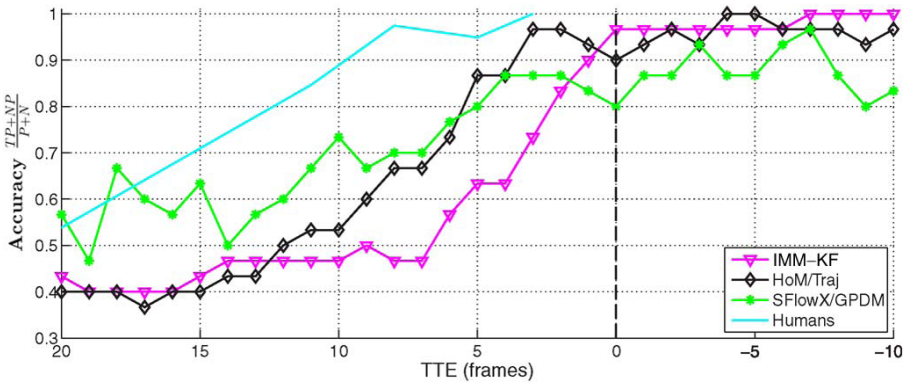


Figure 5.1: Classification accuracy of the different models over time [Keller and Gavrila, 2014].

outperforms the model. In contrast to a motion based model, humans also analyze the current situation, deducing what the pedestrian will do, based on knowledge and contextual information.

Kooij et al. propose a model to predict the lateral path of pedestrians that have the intention to cross the road, but who might stop at the curbstone or continue walking [Kooij et al., 2014a]. The authors assume that a pedestrian who intends to cross the street but decides to stop is influenced by three factors: the existence of an approaching vehicle on collision course, the pedestrian's awareness thereof, and the spatial layout of the environment. A Dynamic Bayesian Network (DBN) is trained to learn which of the current context features are involved in the stopping behavior of a pedestrian and predict its future position. The DBN uses observable information provided by external sensors to estimate the values of four discrete latent variables of the network: if the pedestrian sees or has seen the vehicle based on his head orientation, if there is a risk of collision between the pedestrian and the vehicle, and if the distance between the pedestrian and the curbstone is below a certain threshold. The DBN outputs a probability for stopping or walking. This probability is then used in a Switching Linear Dynamical System to predict the future position of the pedestrian. For each timestep, a new position is predicted via a three step process, doing "predict", "update" and "collapse". The "prediction" step consists of predicting the next behavior of the pedestrian as well as its future position using its previous behavior and position estimation. The "update" step combines the estimated position with the observed one to get an estimate of the current position. The "collapse" step marginalizes out the previous timesteps to

allow for easier computation.

To evaluate the model on real data, the authors use a vehicle equipped with a stereo camera for the recordings. Pedestrians are detected with a HOG/linSVM algorithm [Dalal and Triggs, 2005]. From the detected pedestrians, the likely position of the head of the pedestrian is determined. The angular domain of the observed head orientation is divided into eight bins and a neural network is trained for each of the eight bins. The ego-vehicle speed is provided by on-board sensors. Pedestrian speed is an average of the first order derivative of the position estimate. The curbstone is annotated as a region of interest (ROI) in a digital map. The ego-vehicle localizes itself with GPS data and an algorithm based on Hough transformation to recognize the curbside is applied. The dataset is made out of various short sequences, involving four different instructed unoccluded pedestrians with the intention to cross the street in eight different locations. The moment of the last placement of the foot at the curbstone has been annotated as a stopping behavior, and the moment the foot touches the curbstone for the last time has been annotated as a crossing behavior. The model is trained and tested with a leave-one-out cross validation method. The evaluation shows that a model with the proposed context information has a better estimate of the probability of a stopping behavior and produces more accurate position prediction compared to models that use less or no context at all. The comparison between the accuracy of the prediction of the lateral position between the proposed models and the Probabilistic Hierarchical Trajectory Matching model (PTHM) [Keller and Gavrila, 2014] that is described above shows that the proposed models are more accurate. The DBN model outperforms the PTHM in terms of mean and variance when predicting critical crossing behaviors in which the pedestrian has not seen the vehicle. However, the PTHM performs slightly better when predicting critical stopping behaviors in which the pedestrian has seen the approaching vehicle. This method uses behavior prediction as a mean to improve position prediction in a tracking framework. It demonstrates the benefit of including contextual information and behaviors in the prediction model.

Montel et al. and Tian et al. show that distinguishing stopping from crossing behavior is not an easy task. Montel et al. propose some experiments to identify the environmental features that influence pedestrians to explain their road crossing decision. The authors show that the crossing decision significantly varies according to the environment [Montel et al., 2013]. Tian et al. propose an analysis of the influence of the context on the risk of collision. They show that the risk of collision increases for various road conditions where many pedestrians cross the road [Tian

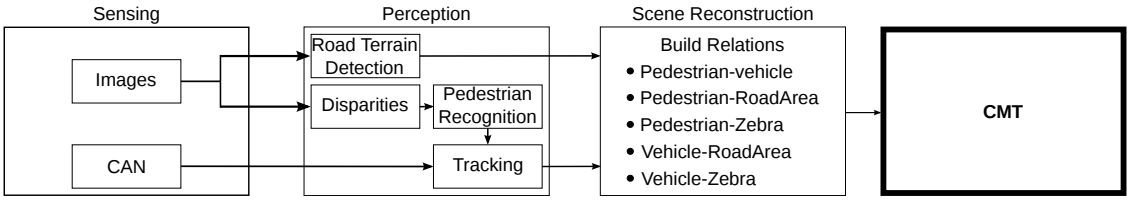


Figure 5.2: Overview of the system architecture for crossing behavior prediction.

et al., 2014]. For example, a pedestrian walking towards a zebra crossing is more likely to cross. On the other hand, a pedestrian approaching the road at an intersection might turn left or right instead of crossing the road. A pedestrian with the intention to cross might not do so if there is a car approaching fast, but might cross if the car is still far away. However, even though the models presented above, which uses different contextual information performs well, a human can still predict a behavior with a higher accuracy and longer time horizon [Keller and Gavrila, 2014]. In contrast to trajectory prediction models, which are very useful for low level control because they target for an explicit position prediction at each timestep, behavioral models focus more on predicting if a pedestrian is going to cross or not, without providing any knowledge about his exact future position. This is usually enough for ADAS to warn a driver in case of a risk of collision with a crossing pedestrian. As we shown in the previous chapter, a model that provides an accurate prediction with a long time horizon is usually very limited to the context. This is why it might make sense to build separate models for different contexts and combine them.

In the previous chapter, we presented an example of a Context Model Tree (CMT) to predict lane changes on the highway. In this chapter, we extend the framework to perform a very different task which is predicting crossing behaviors in inner-city [Bonnin et al., 2014a] as an example to demonstrate its value.

5.2 System Overview

Fig.5.2 provides an overview of the system architecture that has been built in order to do pedestrian behavior prediction. Part of this system was taken from previous work [Weisswange et al., 2013].

5.2.1 Sensing

The sensing module provides the inputs of the system. Color images are taken by a stereo camera system mounted behind the windshield of a vehicle with a frame rate of 20Hz. The stereo cameras have a baseline of 24cm with a resolution of 1280×1024 and a large opening angle (63°). The images are rectified and sent to the perception module.

The speed of the vehicle as well as the yaw rate are extracted from the CAN bus of the vehicle. The CAN data, used to estimate the movement of the ego vehicle is sent to the perception module as well.

5.2.2 Perception

The stereo-images are used to compute disparities. The rest of the perception module uses single images.

The pedestrian recognition component uses a pedestrian detection algorithm [Gepperth et al., 2013] that applies the sliding window technique to detect pedestrians. First, the size of a window is defined in which histograms of oriented gradients (HOG) features are computed [Dalal and Triggs, 2005]. The feature vector is then sent to a linear support vector machine (SVM) trained to classify an image region as a pedestrian or non-pedestrian. The characteristic of the approach is to compare the output of the linear SVM to its corresponding threshold. If the output is above a defined threshold, they localize the position of the detected pedestrian in the sliding window and define a patch around it of the size of a pedestrian. Then, the same feature vector is used in a non linear SVM which is trained only on the patch that is supposed to contain the pedestrian. The detection windows contains a pedestrian if the output of the non linear SVM is above its corresponding threshold. The second classifier acts like a filter to reduce the false positives. The algorithm returns a 2D position of a pedestrian in the image. The disparities are used to convert the 2D image coordinates into 3D egocar-centered coordinate system. The origin of the coordinate system is the center of the rear axle projected to the ground, with x directed to the right, z directed to the front and y directed upwards. In the following, we work in a bird's-eye view (BEV), therefore we assume that the y coordinate of an object is always 0 and consider only the x and z directions.

The tracking module filters the BEV coordinates checking for the width and position of the bounding box. Then it assigns an ID to each detected pedestrian and computes its velocity. To allow the tracking of these entities, an algorithm which can associate multiple simultaneous detections to the same ID is applied. The association between the detection and the prediction is possible computing prob-

abilities with linear multi-target integrated probabilistic data association [Mušicki and La Scala, 2008]. A non linear unscented Kalman filter (UKF) is used as the tracking algorithm, while propagation is done using a linear model. Because BEV positions are computed in the egocar-centered coordinate system which moves with the ego vehicle, all tracked entities are subject to ego motion compensation in each timestep. Ego motion is estimated from speed and yaw rate information provided by the vehicle via the CAN bus assuming constant movement.

The road terrain detection module uses a spatial image recognition algorithm to detect different types of road terrain [Fritsch et al., 2014]. In a first stage, the system represents visual properties of the road surface, the boundary, and lane marking elements in confidence maps based on analyzing local visual features. From these confidence maps SPatial RAY (SPRAY) features that incorporate properties of the global environment are calculated. After this two-staged extraction process, a decision is taken by applying a boosting classifier for road terrain detection that is trained on annotated ground truth data. The approach is trained to recognize the egolane which is the area where the ego-vehicle drives, and the road area which is the terrain where vehicles drive. The road area is limited by a polyline which represents the border of the road [Köster, 2012]. We use an extension of their algorithm to also detect zebra crossing areas.[1]

5.2.3 Scene Reconstruction

The scene reconstruction module uses all the information provided by the perception module to create the relations between traffic entities and contextual information.

For example, a pedestrian who wants to cross the road might first approach the curbstone. In contrast to Kooij et al. [Kooij et al., 2014a] who used a direct detection of the curbstone, we infer the border of the curbstone on both sides of the road, by considering the polyline representing the border of the road area provided by the perception module and extract the left and right border.

The perception module provides an image in which areas that might be zebra crossings are detected. In order to make sure that the detection corresponds to a real zebra crossing and extract the borders, we have designed a filter. We first filter the image in which zebra crossings are detected by the perception module by setting all pixel values above a fixed threshold to 1, to 0 otherwise. We then define a bounding box of the size of a zebra in which all pixels are set to 1. We then do

[1] This basically means training the same algorithm with annotated images including zebra crossing areas.

a convolution between the filtered image and the bounding box in order to isolate the areas with a maximum value. We consider a zebra crossing an area in which a certain number of values are above a fixed threshold. Once a zebra crossings is localized, we estimate a bounding box around in order to get the borders of the zebra.

In some specific locations such as zebra crossing areas, a pedestrian has the priority to cross the road, and vehicles in the vicinity must reduce their velocity. However, not all the zebra crossings detected in the scene might be relevant for a vehicle. For example, at an intersection, a vehicle going straight will not stop if the zebra crossing is on the street that turns left or right. For that reason, a car is related to the closest zebra crossing located in front of it. A pedestrian is related to the closest zebra crossing in its vicinity.

To safely cross a road without any priority to do so, a pedestrian might evaluate the risk of crossing considering the distances and velocities of arriving vehicles [Schmidt and Faerber, 2009]. To present such influence, each pedestrian is related to its closest car, and each car is related to its closest pedestrian.

5.2.4 Context Model Tree

We extend the CMT for highways to the inner-city as shown in Fig.5.3 to predict crossing behaviors. In the following, pedestrians are only related to the ego-vehicle as the closest vehicle, mainly because our recordings do not have situations where a pedestrian is crossing in front of an other vehicle than the ego-car.

The prediction module uses this reconstructed scene to compute features which are used to train classifiers in order to predict if a pedestrian is going to cross in front of the ego-vehicle. This chapter is mainly focusing on this last module. It proposes an example of an application of the CMT on inner-city pedestrian prediction [Bonnin et al., 2014b]. This framework is designed to combine models of different specificity hierarchically by ordering them from generic to specific. The top node is a generic model which is always activated as long as the ego-vehicle is localized in inner-city. In our application, an “Inner-City” model will be in the top node of the CMT, parallel to the “Highway” node. A “Zebra Crossing” model is a more specific model and will be a sub-node of the “Inner-City” node. This node gets activated only when a zebra crossing is detected at up to 40m distance. By always having the “Inner-City” node active, the CMT ensures that a traffic participant will always get predicted.

There can be situations where for the same traffic participant, the two nodes provide opposite predictions. In such a case, the CMT uses a combination function

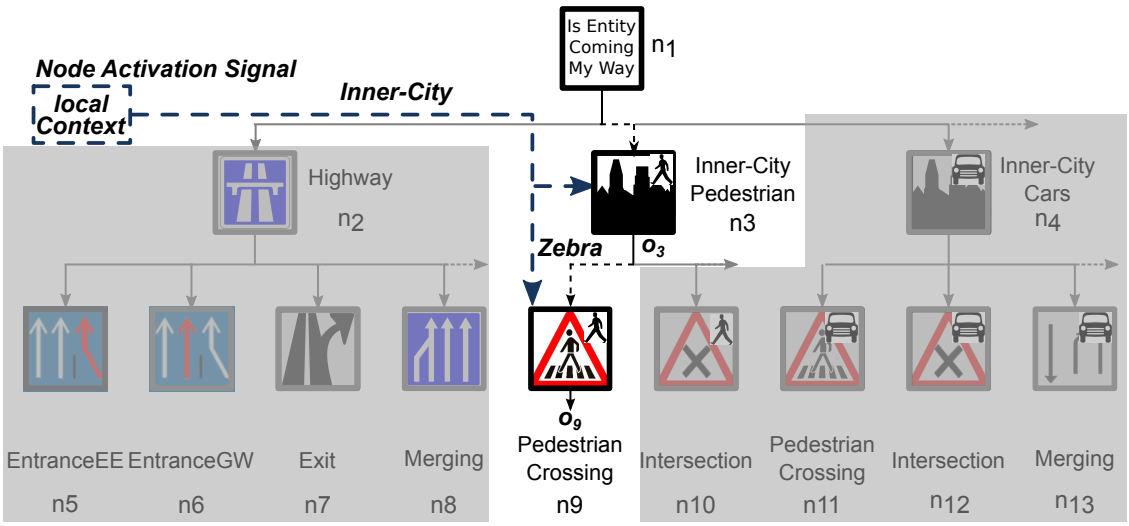


Figure 5.3: Visualization of the CMT for the inner-city. Each rectangle represents a node of the tree. The "Inner-City" node n_3 is active to predict any pedestrians in inner-city. The "Zebra Crossing" node n_9 is a sub-node specific to the inner-city. Black lines connect the active sub-node to the inner-city. The "Zebra Crossing" node is activated when a pedestrian approaches a zebra crossing. It receives the output o_i from the "Inner-City" node and returns its own output. The nodes that are in the grey area are not presented in this chapter.

to ensure an accurate final prediction. For each pedestrian that is predicted by the "Inner-City" model, the classifier produces a confidence output o_j. When the sub-node is activated, it applies the combination function between the confidence c_i of its classifier and the output o_j sent by the top node.

The confidence values c_j and c_i do not have the same meaning because they are produced by independent classifiers. To make the two models comparable, we first scale each confidence output by dividing it by the best threshold of the corresponding model. The combination function is then a maximum function between both scaled confidences. The sub-node outputs the result of the maximum function as the final prediction as shown in equation (5.1).

$$o_i = max(o_j, (c_i/\theta_i)) \tag{5.1}$$

We predict that a pedestrian is crossing if $o_i >= 1$.

The output of this module could then be used to decide about an appropriate action, e.g. warning the driver or reducing velocity.

5.3 "Inner-City" Node

5.3.1 Scenario

When driving in inner-city, a driver has to care for pedestrians that can cross the road at any time. To prevent a car-pedestrian collision, we want to develop a generic "Inner-City" model that covers usual behaviors of pedestrians in inner-city based on physical motion with additional knowledge of the context. We have designed a set of complex features that are normalized to a similar range in order to then train a classifier.

5.3.2 Feature Definitions

A pedestrian with ID i has a currently measured position $p_i = (p_{i_x}, p_{i_z})$ and velocity $v_i = (v_{i_x}, v_{i_z})$. Equation (5.2) uses the current position and velocity of the pedestrian to estimate his future position:

$$\begin{aligned} \tilde{\mathrm{p}}_{i_x} &= v_{i_x}. \triangle t + p_{i_x} \\ \tilde{\mathrm{p}}_{i_z} &= v_{i_z}. \triangle t + p_{i_z} \end{aligned} \tag{5.2}$$

Fig.5.4 shows a scene in inner-city where a pedestrian h_1 is going to cross in front of the ego-vehicle. The arrows 1-9 represent features that are used by the "Inner-City" model. The features are designed to answer two questions: first, will a pedestrian h_i cross the road and secondly, is he going to intersect the path of the ego-vehicle. To answer these questions we have designed the following features:

1- *globalOrientation* represents the angle α which is the angle between the moving direction of a pedestrian h_i and the ego-coordinate axes:

$$\alpha(h_i) = tan(v_{i_z}, v_{i_x}) \tag{5.3}$$

2- *relativeOrientation* relates the orientation of h_i as the angle θ between h_i and the ego-vehicle:

$$\theta(h_i) = tan(\tilde{\mathrm{p}}_{i_z}, \tilde{\mathrm{p}}_{i_x}) \tag{5.4}$$

We can use $\tilde{\mathrm{p}}_{i_x}$ and $\tilde{\mathrm{p}}_{i_z}$ directly because they are expressed in the ego-vehicle coordinate system.

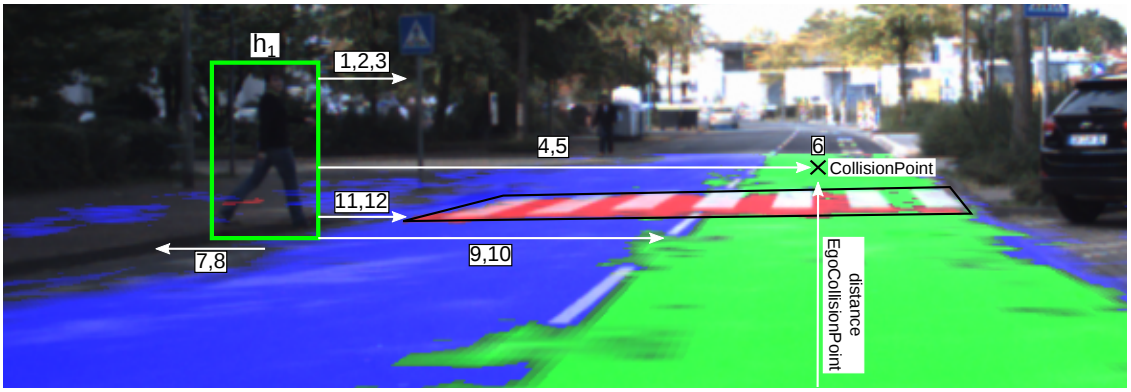


Figure 5.4: Features computed to predict if pedestrian h_1 is going to cross in front of the ego-vehicle. The green area represents the egolane. The blue area represents the road area. The red area within the black rectangle is the zebra crossing area. Each number is associated to a specific feature that is described in the text.

3- *isFacingRoad* uses the *globalOrientation* α from equation (5.3) to compute if h_i is facing the road:

$$isFacingRoad(h_i) = \begin{cases} 1 & (p_{i_z} > 0.5 \text{ and } \text{abs}(\alpha) >= 90°) \text{ or} \\ & (p_{i_z} < 0.5 \text{ and } \text{abs}(\alpha) <= 90°) \\ 0 & otherwise \end{cases} \tag{5.5}$$

4- *distanceCollisionPoint* is the lateral distance p_{i_x} of h_i in the ego-car coordinate system. The collision point (CP) is the point where pedestrian and ego-vehicle will intersect assuming that the ego-vehicle goes straight and the pedestrian walks orthogonal to the ego-vehicle a_{ego}:

$$CP_{static} = (p_{Ego_x}, p_{i_z}) \tag{5.6}$$

$$d(CP_{static}, h_i) = p_{i_x} \tag{5.7}$$

5- *timePedCollisionPoint* is the time for h_i to reach the collision point as shown in equation (5.9). We assume that the ego-vehicle goes straight, and the pedestrian walks according to its moving direction. We compute the collision point as the point where both linear trajectories will intersect, see equation (5.8). This feature is computed only for moving pedestrians that are facing

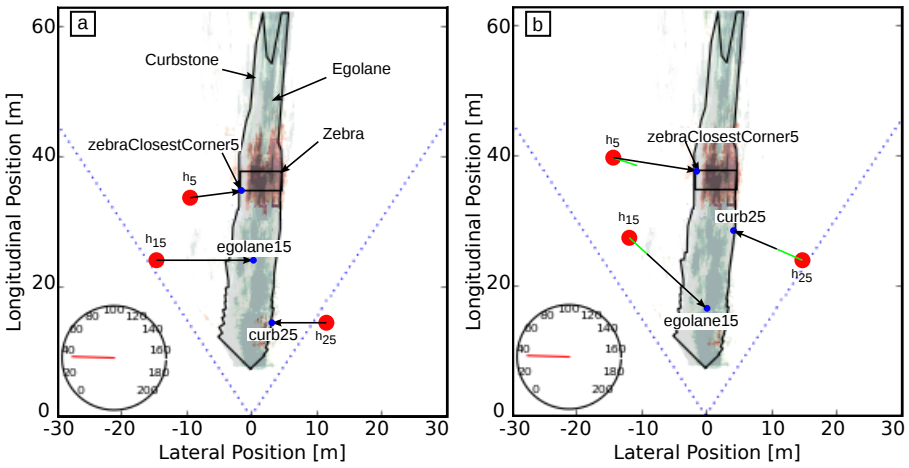


Figure 5.5: Bird's-eye view representation of an inner-city scene for feature computation. Red dots are pedestrian and green lines represent their velocity. The solid black lane is the polyline representing the curbstone. The black rectangle around the red area represents the zebra crossing. Dark grey is the egolane and pale green is the road area. Blue dots represent the point of the context elements that are used for computing the distance and time to pedestrians. (a): Computation of the distance between pedestrians and the different context elements using only the pedestrian position. (b): Computation of the time needed by the pedestrians to reach the different context elements using the moving direction of the pedestrians.

the road. For static pedestrians or moving ones that are not facing the road, of if the CP_{moving} does not exist, the feature value is set to 0 after the scaling:

$$CP_{moving} = ((\tilde{\mathrm{p}}_{i_z}.p_{i_x}) - (p_{i_z}.\tilde{\mathrm{p}}_{i_x}))/(p_{i_x} - \tilde{\mathrm{p}}_{i_x}) \tag{5.8}$$

$$dCP_{moving} = \sqrt{{p_{i_x}}^2 + (CP_{moving} - p_{i_z})^2}$$

$$|v_i| = \sqrt{{v_{i_x}}^2 + {v_{i_z}}^2}$$

$$timePedCP(h_i) = dCP_{moving}/|v_i| \tag{5.9}$$

6- $diffTimeColision$ is the difference between the time for the ego-vehicle to reach the CP_{moving} minus the time for h_i to reach the CP_{moving} as shown in equation (5.10). This feature is computed only for moving pedestrians that

are facing the road. For static pedestrians or moving ones that are not facing the road, the feature value is set to 0 after the scaling:

$$\begin{aligned} timeEgoCP &= CP_{moving}/v_{Ego_z} \\ diffTimeColision &= timeEgoCP - timePedCP \end{aligned} \tag{5.10}$$

7- *distanceToCurbstone* is the distance between h_i and the left or right border of the curbstone depending if the position p_{i_x} is negative or positive accordingly, assuming that the pedestrian moves orthogonal to the ego-vehicle. This feature is computed only for pedestrians facing the road, otherwise the feature is set to 0 after the scaling:

$$\begin{aligned} curb_i &= (curb_{i_x}, curb_{i_z}) \\ d(h_i, curb_i) &= \sqrt{(curb_{i_x} - p_{i_x})^2 + (curb_{i_z} - p_{i_z})^2} \end{aligned} \tag{5.11}$$

Pedestrian h_{25} in Fig.5.5.(a) is walking towards the curbstone orthogonal to the ego-vehicle. The left and right border of the polyline which limits the road area corresponds to the left and right border of the curbstone respectively.

8- *timeToCurbstone* is the time needed by h_i to reach the curbstone using his moving direction. We assume that the pedestrian is facing the road and moving towards it. *timeToCurbstone* is computed by dividing the distance between pedestrian and curbstone by the current velocity of the pedestrian. For static pedestrians or moving ones that are not facing the road, the feature value is set to 0 after the scaling:

$$t(h_i, curb_i) = d(h_i, curb_i)/|v_i| \tag{5.12}$$

Pedestrian h_{25} in Fig.5.5.(b) is walking towards the curbstone $curb_{25}$.

9- *timeToEgoLane* is the time needed by h_i to reach the left or right border of the egolane depending if the pedestrian is on the left or right of the ego-vehicle accordingly. We assume that the pedestrian is facing the road and moving towards it. *timeToEgoLane* is computed by dividing the distance between pedestrian and egolane by the current velocity of the pedestrian. For static

pedestrians or moving ones that are not facing the road, the feature value is set to 0 after the scaling:

$$egolane_i = (egolane_{i_x}, egolane_{i_z})$$
$$d(h_i, egolane_i) = \sqrt{(egolane_{i_x} - p_{i_x})^2 + (egolane_{i_z} - p_{i_z})^2} \quad (5.13)$$
$$t(h_i, egolane_i) = d(h_i, egolane_i)/|v_i| \quad (5.14)$$

Pedestrian h_{15} in Fig.5.5.(b) walks towards $egolane15$.

5.3.3 Feature Scaling

The features of the "Inner-City" node have been designed based on expert knowledge. For example, a pedestrian who is going to cross will be close to the curbstone before entering the road. This behavior is described by the *distanceToCurbstone* feature. A feature alone does not suffice to predict a crossing behavior. For example, a very small value of *distanceToCurbstone* does not necessary mean that the pedestrian is going to cross, it could be just waiting for a bus. On the other hand, the same feature combined with the knowledge that the pedestrian is moving towards the road (*timeToCurbstone*) indicates a crossing behavior. To be able to handle many situations (including static pedestrians starting to cross) we use a multitude of features. As we did on the highway, we used a Fermi function to scale the features as shown in equation (5.15).

$$\text{fermi}(x, min, max) = 1 - (2/(\exp((x - \mu)/k) + 1)) \quad (5.15)$$

The parameter $\mu = \min + (\max - \min)/2.0$ is used to center the results in the interval and the parameter $k = (\max - \mu)/log(1/0.9 - 1)$ changes the slope of the curve.

We chose to train an SLP, see Fig.4.8 and equations (4.10, 4.11, 4.12, 4.13).

The "Inner-City" model is active for all pedestrians in the ego-vehicles surrounding within 60m distance.

5.4 "Zebra Crossing" Node

5.4.1 Scenario

This model is designed to predict crossing behaviors of pedestrians at zebra crossings in inner-city, where pedestrians have the priority to cross the road. As for the "Inner-City" model, we have designed a set of complex features that are normalized to a similar range in order to then train a classifier.

5.4.2 Feature Definitions

Fig.5.4 shows a scene where a pedestrian h_1 is going to cross in front of the ego-vehicle at a zebra crossing. This model uses features 1-2 and 5-9 of the "Inner-City" model. Additionally, it uses the following features:

10- *distanceToEgoLane* is the distance between a pedestrian and the left or right border of the egolane depending if the pedestrian position is negative or positive respectively, see equation (5.13). This feature is computed only for pedestrians facing the road. Otherwise, the feature is set to 0 after the scaling. In the "Inner-City" model, this feature is used to compute the $timeToEgolane$, but is not used as an input of the classifier because the distribution between positives and negatives was very similar to the $distanceToCurbstone$. An example is shown in Fig.5.5.(a) with pedestrian h_{15} who walks towards $egolane15$.

11- *distanceToZebra* is the distance between h_i and the closest corner of the zebra crossing. In order to know where the pedestrian is located according to the zebra crossing, we compute the distance between a pedestrian and the four corners of a zebra crossing:

$$\begin{aligned} zebraCC_i &= (zebraCC_{i_x}, zebraCC_{i_z}) \\ d(h_i, zebraCC_i) &= \sqrt{(zebraCC_{i_x} - p_{i_x})^2 + (zebraCC_{i_z} - p_{i_z})^2} \end{aligned} \tag{5.16}$$

Pedestrian h_5 in Fig.5.5.(a) walks towards $zebraClosestCorner5$.

12- *timeToZebra* is the time for h_i to reach the closest corner of the zebra crossing. This feature is computed only for moving pedestrians facing the road, otherwise the feature is set to 0 after the scaling:

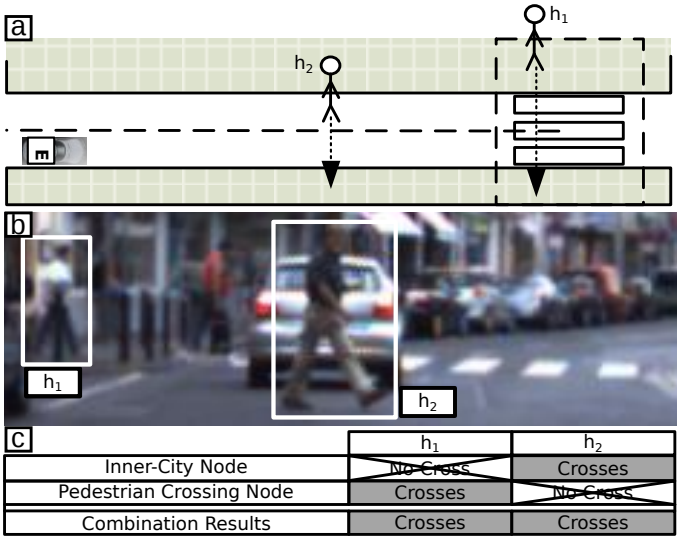


Figure 5.6: Scenario typical for inner-city and zebra crossings occurring at the same time. (a): The ego-vehicle E is approaching a zebra crossing represented by the dashed rectangle. Dashed arrows represent the future behavior of the pedestrians. h_1 and h_2 are pedestrians about to cross the road. (b): h_1 and h_2 in a real world image. (c): The prediction result of the "Inner-City" model, the "Zebra Crossing" model and the combination results of both models exemplary for both pedestrians. Grey fields indicate a correct prediction, crossed out field indicate a wrong prediction.

$$t(h_i, zebraCC_i) = d(p_i, zebraCC_i)/|v_i| \tag{5.17}$$

Pedestrian h_5 in Fig.5.5.(b) walks towards $zebraClosestCorner5$.

In analogy to the "Inner-city" model, the features are scaled with a Fermi function, see equation (5.15), and we chose to train an SLP, see Fig.4.8.

5.4.3 Model Conditions and Combination Function

This model will be used in the "Zebra Crossing" node as a sub-node of the "Inner-City" node. The model is active for all pedestrians located within 10m distance of a zebra crossing.

For this scenario, the combination function is applied to all predicted pedestrians located in the zebra area. All predictions of the "Inner-City" node for other

pedestrians will be preserved in the final confidence value. The final prediction for crossing the road will be the output of the maximum function applied to the scaled confidence value of both nodes, see equation (5.1).

Fig.5.6 displays an exemplary zebra crossing scenario which motivates this choice for the combination function. In this scenario, the intention of the two pedestrians to cross the road can be inferred. h_2 is already on the road walking toward the ego-lane. The "Inner-City" model predicts that h_2 is going to intersect the path of the ego-vehicle. However, h_2 is very far from the zebra crossing area and not walking towards it, so the "Zebra Crossing" model predicts that h_2 is not going to cross. h_1 who is very far from the road is not going to cross according to the "Inner-City" model. However, the "Zebra Crossing" model predicts the contrary due to the knowledge of the pedestrian approaching the zebra area to cross the road. The final prediction will be the output of the maximum function applied to the scaled confidence of the "Inner-City" node and the output of the "Zebra Crossing" model, as shown in equation (5.1).

5.5 Experiments

In this section, we present an evaluation of the performance and prediction time horizon of the generic "Inner-City" model and "Zebra Crossing" model using real data. We show that the "Inner-City" model performs generally well at predicting crossing behaviors, but in specific locations, the prediction starts later than what we could expect. We show that a "Zebra Crossing" model, that uses specifically tuned contextual information, performs better at those locations than a generic "Inner-City" model. However, such a model does not generalize well to other scenarios. Despite getting a good performance for a single generic model, we see that this might no be the best possible performance for all situations. Therefore we want to combine the two models. We will therefore also test a multi-model system based on the previously introduced CMT framework to combine multiple models of different specificity [Bonnin et al., 2014b].

5.5.1 Recordings

All tests have been done using real data recorded by a camera mounted on the windshield of a vehicle on German inner-city roads (curved and straight). We recorded a single stream of one hour length in areas where pedestrians often cross

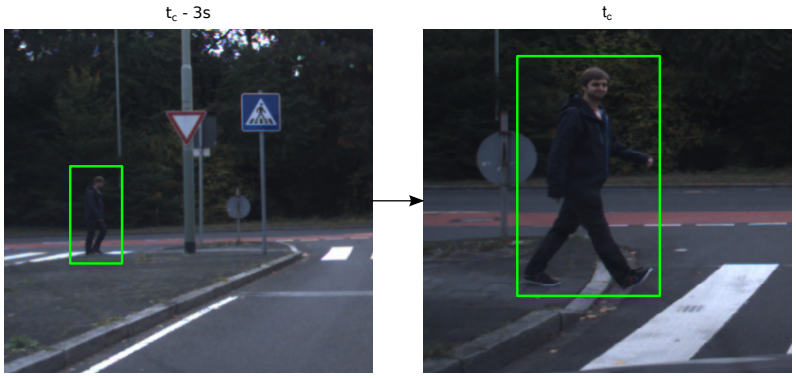


Figure 5.7: Example of a crossing pedestrian. The right image shows a pedestrian entering the egolane. This point in time t_c is defined as the beginning of the crossing behavior. The left image shows the same pedestrian at time $t_c - 3s$ which we define as the start of positive prediction event annotation.

the road[2], with schools, shops, bus stops and zebra crossings. An initial recording showed that in one hour, only 5 to 6 pedestrians were crossing (out of 83 overall). To increase the size of the positive dataset, we have instructed seven people with a list of tasks to perform while the ego-vehicle was approaching. Every time we were approaching one of them, they freely chose a task of the list. They could e.g. walk on the pavement, wait for the bus, cross at a zebra crossing, or cross in a non zebra crossing area, but we never knew in advance what they were going to do. They have been told to act naturally, without taking unrealistic risk when crossing the road. Our recording contains normal pedestrian crossings as well as several crossings of these instructed people.

In this thesis, we define the start of a pedestrian crossing when its foot touches the egolane[3] as shown in Fig.5.7 and our goal is to predict a crossing behavior before this point. Fig.5.8 shows that on average, 15% of the pedestrians in our dataset start crossing at below 10m from the ego-vehicle, 18% of the pedestrians start crossing between 10m and 20m, and 63% of the pedestrians start crossing above 20m. Due to the camera's opening angle and resolution used for recording our dataset, the detection range of the pedestrian recognition system is limited to approximately 20m in longitudinal and 15m in lateral distance. On average, a

[2]Offenbach-Bieber, Germany, latitude 50.084697°N, longitude 8.827502°E

[3]We use the egolane and not the road area to concentrate on those cases that really cross the path of the ego-vehicle.

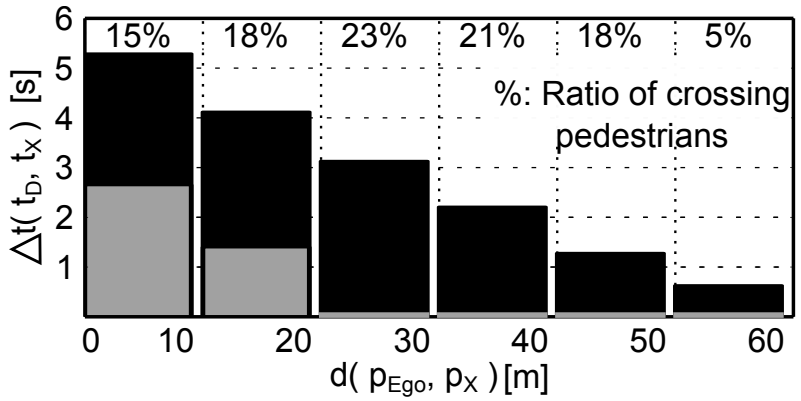


Figure 5.8: Time between the first pedestrian detection t_D and the start of a crossing behavior t_X versus the distance between the ego-vehicle's position p_{Ego} and the crossing point p_X. Grey: pedestrians that are tracked by the original system. Black: pedestrians that are detected by the algorithm merged with manual pedestrian annotations. Percentage values on top: ratio of pedestrians that cross in the corresponding distance range.

pedestrian that starts crossing the road below 10m can be detected 2.65s before a crossing. Between 10m and 20m, the first detection of a crossing pedestrian starts 1.4s before a crossing. 67% of the crossing pedestrians are not detected at all due to the limited detection range. To be able to predict pedestrians earlier and at a further distance, we have decided to manually annotate pedestrians that are above 20m. The annotated 2D positions of pedestrians were mapped to 3D using the disparity map and then fed to the tracking algorithm. The black bars in Fig.5.8 show that with annotated pedestrians it is possible to have a longer detection, up to 5.3s and up to 60m distance. This "assisted" pedestrian detection lies within the range of state of the art laser scanners [Lages, 2013].

5.5.2 Dataset

We use a supervised learning method to predict behaviors, so we need to build one dataset for training the classifier and one dataset for testing. To train and test a classifier we need to compute the input features and provide the groundtruth at each timestep. That means that each ID has a corresponding annotation as a negative or positive example for each timestep.

In inner-city, we are interested in predicting if a pedestrian will cross the future

path of the ego-vehicle. A pedestrian is considered as crossing when its foot touches the egolane as shown in Fig.5.7. When a crossing is detected, each timestep from that moment to three seconds before is annotated as a positive example (annotated to 1).

The timesteps are annotated to -1 while the pedestrian is on the egolane because it is not clear what should be the outcome of the prediction when the behavior already takes place. Three seconds before the start of the prediction are annotated to -1 to make a clear separation between a positive and negative example to not confuse the classifier during training. Everything else is annotated as negative example (annotated to 0).

The dataset contains 322 pedestrians with 36 of them crossing the road. We excluded wrong pedestrian detections (e.g. bikes, trees, traffic signs) from the dataset in order to focus the evaluation on the prediction module rather than the general system.

Out of the dataset, we created a subset 'zebra' containing all pedestrians that are within 10m distance of a zebra crossing (18% cases). It might be interesting to separately test on this dataset, because we observed that the ratio of crossings versus non-crossings is very different from the rest of the dataset (26% of pedestrians crossing at zebra crossings versus 9% crossing in inner-city without priority). We also constructed a subset 'intersection' containing all pedestrians at intersections (26%). These cases will be very difficult, if the ego-vehicle turns, because they do not cross the ego-path despite being in front of the car. To deal with these pedestrians, we expect additional information about the ego-path to be required, therefore we do not use them in the following. We created a third subset 'inner-city' containing all other pedestrians.

5.5.3 "Inner-City" Model

In this experiment, we present an event based evaluation of the performance and prediction time horizon of the "Inner-City" model.

Scaling Features

As we did on the highway, all features have been scaled a similar range. For each feature, we have separated the values in training set into two groups: crossing and non crossing as shown in Fig.5.9. The four plots in Fig.5.9 have different ratio of positives and negatives. This occurs because we chose to draw only the meaningful values. A feature that cannot be computed is considered as invalid and does not appear in this figure. When looking at the distribution between both

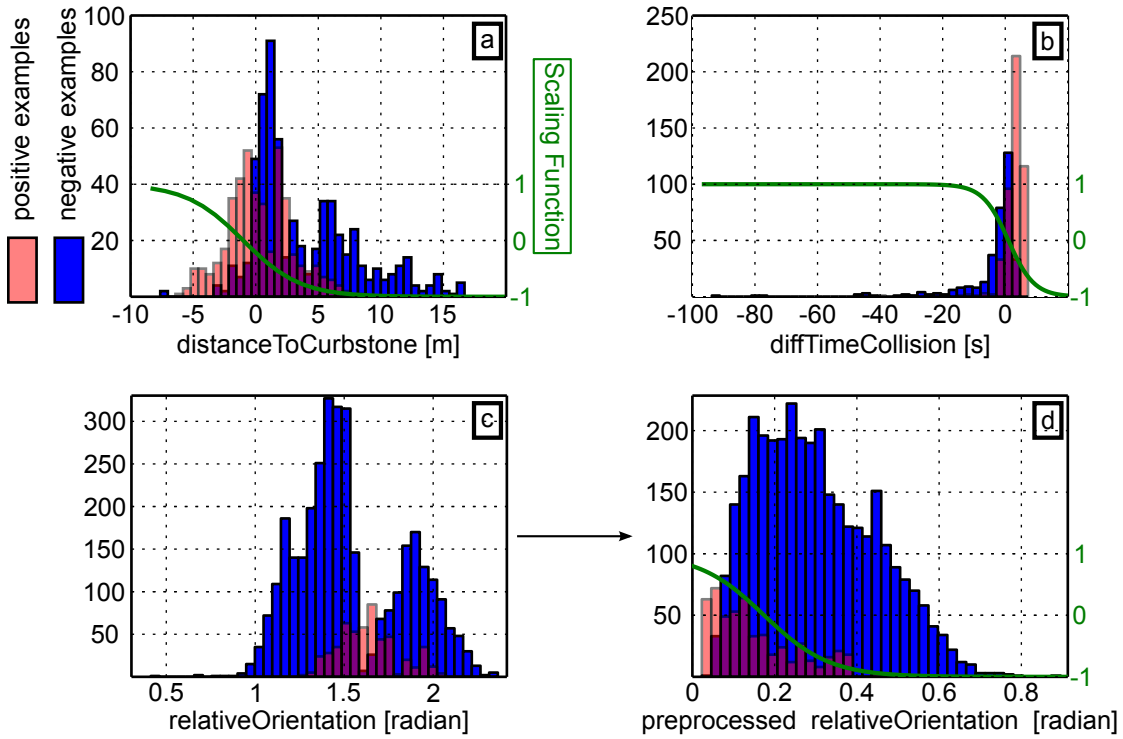


Figure 5.9: Distribution of values for three example features computed on the inner-city set. Blue bars are non crossing pedestrians and pink bars the crossing ones. The green line is the chosen scaling function for the corresponding feature. (a): Feature *distanceToCurbstone*. (b): Feature $diffTimeColision$. (c): Feature *relativeOrientation* before any processing. (d): New distribution of negatives and positives of feature *relativeOrientation* after processing the data.

groups, we observe that in some ranges the two groups overlap, whereas in some others they do not. It is easy to separate the two groups when they do not overlap. For that reason, we have chosen a function that preserves the value differences in the overlapping area and compresses the representation of the non-overlapping ones. A minimum and maximum threshold are defined for each feature as shown in Table 5.1. Afterward, we choose a Fermi function (see equation (5.15)) to scale the features 2 and 4-9.

Fig.5.9.(a) shows that positive examples will be scaled between 0 and 1 and negatives between -1 and 0. Values that are scaled to 0 are those that cannot be well interpreted by the classifier because it is hard to differentiate between crossing

Table 5.1: Feature Definition

N°	Feature Name	[min;max] Scaling Thresholds
2	*relativeOrientation*	IC:[0; 3.14]
4	*distanceCollisionPoint*	IC:[-6;10]
5	*timePedCollisionPoint*	IC:[0,6]
6	$diffTimeColision$	IC:[-8;10]
7	*distanceToCurbstone*	IC:[-6;4]
8	*timeToCurbstone*	IC:[-2;4]
9	*timeToEgoLane*	IC:[-1;4]

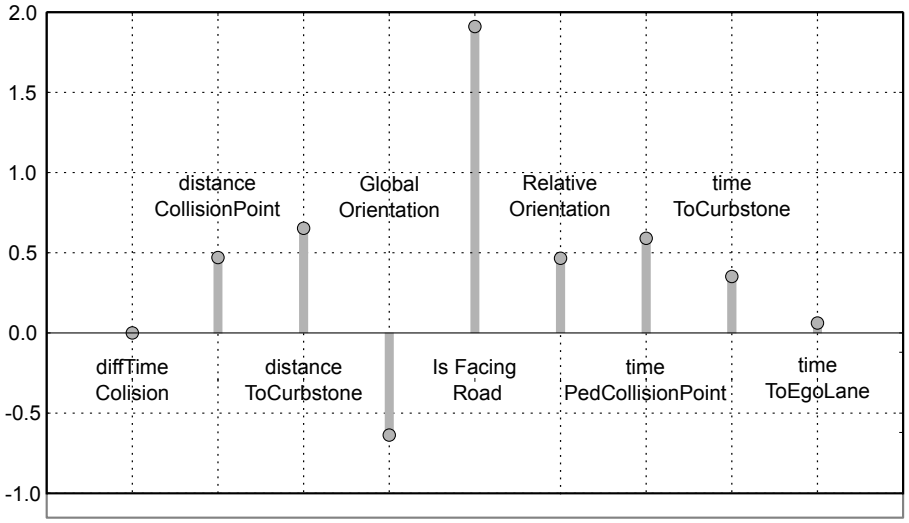


Figure 5.10: Weight values after the training of the Single Layer Perceptron used in the "Inner-City" model.

and non crossing pedestrians. The classifier will not be able to learn from the values that are mapped to 0 during the training.

Features 1-2 are angles defined on $[-\pi, \pi]$. To extract meaningful information taking into account the cyclic dependencies, they have to be preprocessed. Fig.5.9.(c) shows the feature *relativeOrientation*. Before applying the Fermi function to it, we have applied the following transformation: $p = abs(90 - x)$. Fig.5.9.(d) shows the feature values that have been preprocessed, as well as the scaling function. We did not apply the Fermi function to the feature *globalOrientation*, instead we divide the feature value by π because there was no clear separation between the two groups. Feature 3 is a boolean as shown in equation (5.5) and does not need

to be scaled.

Single Layer Perceptron

We use the same SLP as for the models presented in the previous section (see Fig.4.8). For the "Inner-City" model, we use 9 input neurons plus bias and a learning rate of 0.002. Fig.5.10 shows the values of the weights for the different inputs after the classifier has been trained. Because some of the features had a very low weight, we have compared the performance of the classifier trained with all the features with a model trained with features in a PCA space keeping only the features that explained 90% of the variance. We did not observe any difference between both methods. For that reason, we chose to train the classifier with all the features.

Event Based Evaluation

The "Inner-City" model is trained and tested on the 'inner-city' and the 'zebra' dataset (the 'ICZ dataset').

We consider consecutive timesteps in which for each pedestrian the prediction is above a threshold θ as one event (prediction gaps of size smaller than 1s are grouped into the same event). We use the state machine shown in Fig.4.17 in the previous chapter to count the events. Following the principle of a ROC curve, theta is sampled from 0 to 1 and the TPR and FPR are computed for each setting of theta. To do so, we compute the events and then compare those with the groundtruth annotation at the same timesteps. If the event overlaps with any timestep labeled to 1, it is counted as a TP. If it overlaps with any timestep labeled to 0, it is counted as a FP. For more details on event based evaluation see the evaluation of the CMT for the highway presented in the previous chapter.

We compute the PTH as the difference between the time of the first prediction of a TP event with the time t_c when the pedestrian starts crossing (Fig.5.7).

Fig.5.11 shows a ROC curve with the TPR versus the FPR of the "Inner-City" model tested on the 'ICZ dataset'. The ROC curve shows that for a threshold $\theta_j = 0.717$, the model has a TPR of 31% for a FPR of 0.0%. At this threshold, the model can predict a crossing 0.72s before it starts[4].

[4]The PTH versus FPR curve sometimes shows a reduction of time horizon despite lowering the threshold. This occurs because we use only true positive events to compute the time horizon, and a new TP usually starts with a small PTH.

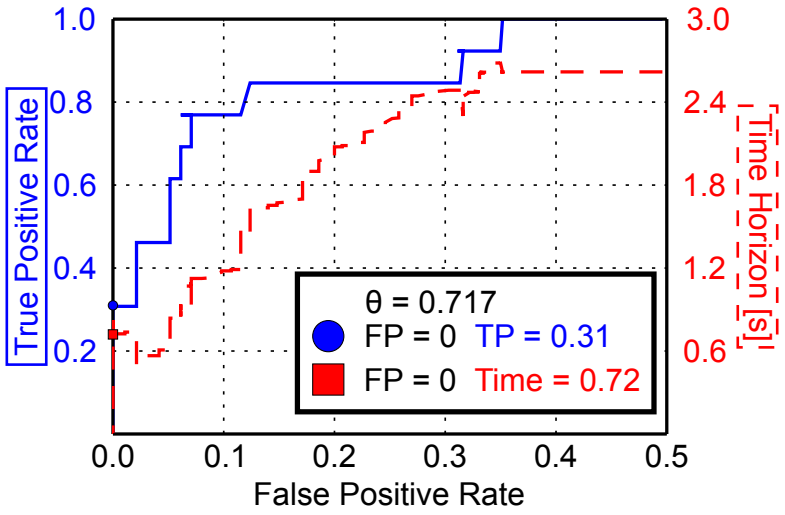


Figure 5.11: Event based ROC curve of the generic "Inner-City" model tested on the 'ICZ dataset'. The solid line shows True Positive Rate versus False Positive Rate. The dashed line represents an average of the PTH versus the False Positive Rate.

This ROC curve shows that by lowering the threshold, it would be possible to have a higher TPR with a better prediction time horizon. However, this comes at the cost of making prediction errors by increasing the FPR. We can see in the dataset that there are much fewer people crossing than non crossing (only 16% with 9% in inner-city). For that reason, we target for a minimum of errors, and choose $\theta_j = 0.717$ for the following experiments.

We were interested if the model shows differences in performance between the two types of context. We therefore tested it with the 'inner-city' and the 'zebra' dataset separately. Fig.5.12 shows that the "Inner-City" model, can predict non priority inner-city crossings with a TPR of 29% on average 0.67s in advance without making errors. It can predict zebra crossing behaviors with a TPR of 33%, a FPR of 0% on average 0.77s in advance.

These numbers show that the "Inner-City" model can predict both types of crossings equally well. However, we expect that prediction at zebra crossings should be easier because a pedestrian at zebras is usually detected for a much longer time due to the lower speed of the ego-vehicle and the intention to cross is often more visible for example because of fewer parked cars around a zebra crossing area. We have developed a model specific to zebra crossings to test if it can produce even

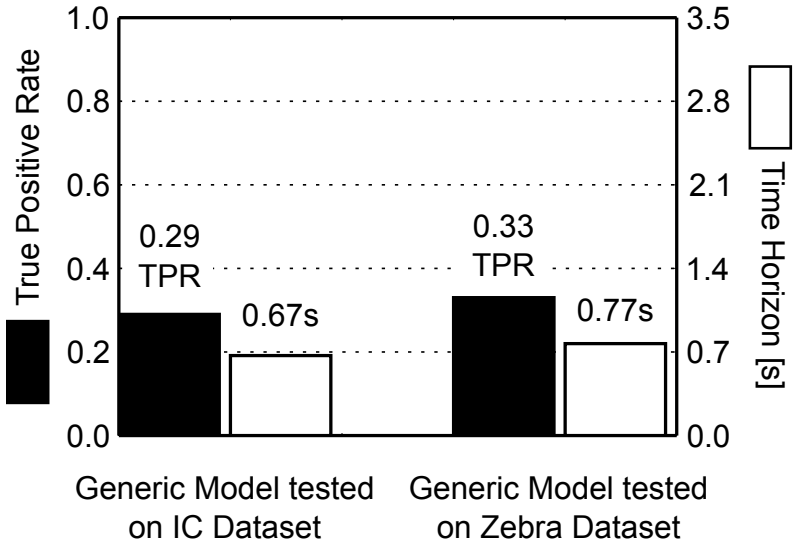


Figure 5.12: Performance of "Inner-City" model. Black and white bars show the TPR and PTH respectively. The left bars show the performance on the 'inner-city' set. The right bars show the performance on the 'zebra' set.

Table 5.2: Feature Definition

N°	Feature Name	[min;max] Scaling Thresholds
2	*relativeOrientation*	Z:[0;3.14]
5	*timePedCollisionPoint*	Z:[0;12]
6	$diffTimeColision$	Z:[-6;4]
7	*distanceToCurbstone*	Z:[-2;8]
8	*timeToCurbstone*	Z:[-1;11]
9	*timeToEgoLane*	Z:[0;12]
10	*distanceToEgoLane*	Z:[-7;9]
11	*distanceToZebra*	Z:[-4;10]
12	*timeToZebra*	Z:[-3;6]

better results.

5.5.4 "Zebra Crossing" Model

In this experiment, we present an event based evaluation of the performance and prediction time horizon of the "Zebra Crossing" model.

Scaling Features

The features used by the "Zebra Crossing" node are similar to the ones in the "Inner-City" node, however, the feature ranges learned by the classifier are different. For example, pedestrians crossing at zebras are usually closer to the ego-vehicle. A pedestrian standing at the side of the road is more likely to cross in the vicinity of a zebra crossing than in other areas. Fig.5.13.(a) shows that pedestrians located in a zebra area which do not cross stay further away from the curbstone than in the "Inner-City" node. This might be to not confuse drivers about their intention and not make them stop for no reason. Fig.5.13.(b) shows a similar distribution as in the "Inner-City" node except that the range is smaller. Fig.5.13.(c) and Fig.5.13.(d) show the $distanceToZebra$ and $timeToZebra$ that are features specific to the "Zebra Crossing" model. As for the "Inner-City" model, we use a single layer perceptron as classifier and a Fermi function to scale the features, see equation (5.15). For each feature, Table 5.2 shows the corresponding min and max thresholds for scaling.

Single Layer Perceptron

We use the same SLP as for the "Inner-City" model (see Fig.4.8). For this model, we used an SLP with 10 input neurons plus bias and a learning rate of 0.002. Fig.5.14 shows the values of the weights for the different inputs after the classifier has been trained.

Event Based Evaluation

We did an event based evaluation of the "Zebra Crossing" model tested on the 'zebra' dataset. For this model, we chose a threshold which provides a good TPR and a long prediction time horizon. We have chosen the threshold ($\theta_i = 0.34$) that allows the model to have a good prediction quality for no prediction errors. Fig.5.15 (left) shows that the "Zebra Crossing" model, tested at zebra crossings, can predict all crossing pedestrians on average 3.23s in advance without doing any FPs. These numbers confirm our hypotheses that pedestrians at zebra crossings can be predicted more easily and much earlier.

We have also tested the "Zebra Crossing" model on the 'inner-city' dataset[5]. Fig.5.15 shows that the model, at θ_i, can predict crossings in inner-city with a

[5]For this evaluation, the features specific to the "Zebra Crossing" model were set to 0 when no zebra crossing was detected.

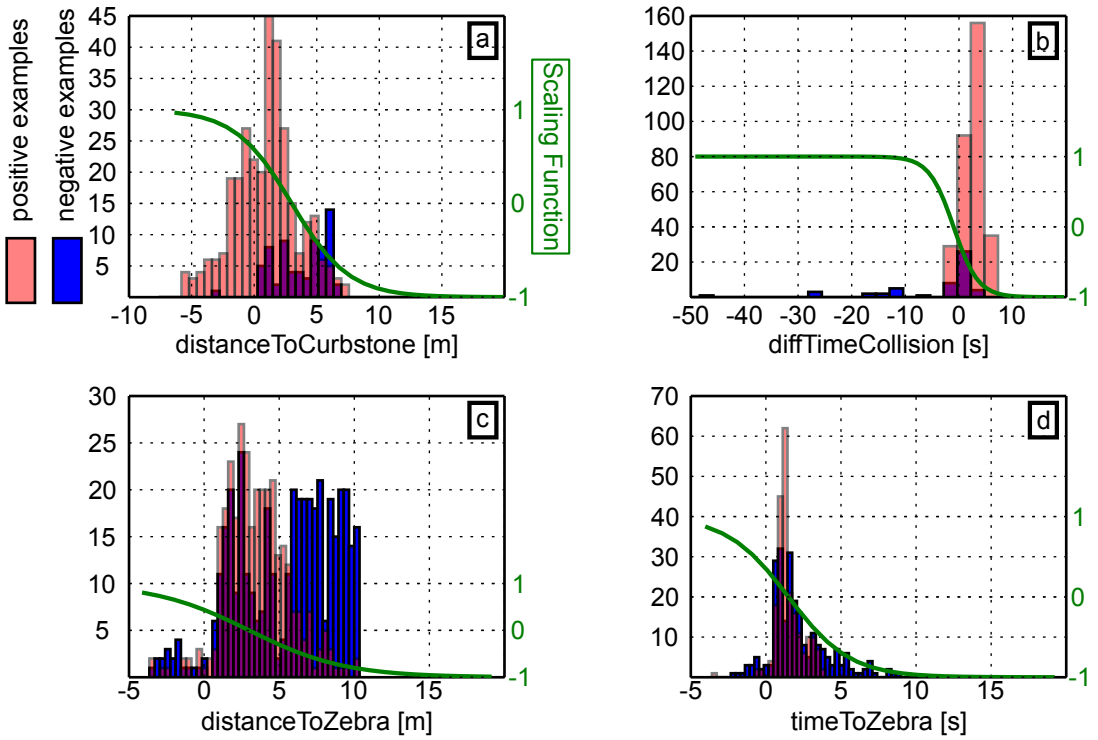


Figure 5.13: Distribution of values for four example features computed on the zebra set. Blue bars are non crossing pedestrians and pink bars crossing ones. The green line shows the scaling function for the corresponding feature. (a): Feature *distanceToCurbstone*. (b): Feature *diffTimeColision*. (c): Feature *distanceToZebra*. (d): Feature *timeToZebra*.

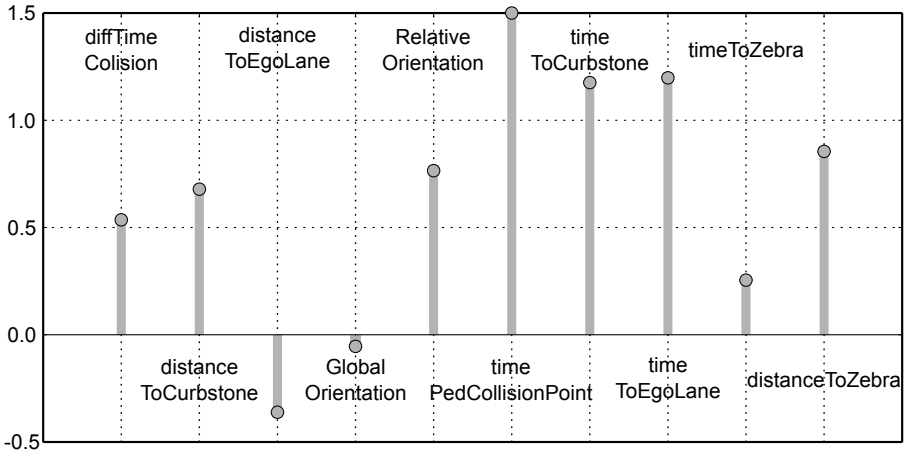


Figure 5.14: Weight values after the training of the Single Layer Perceptron used in the "Zebra Crossing" model.

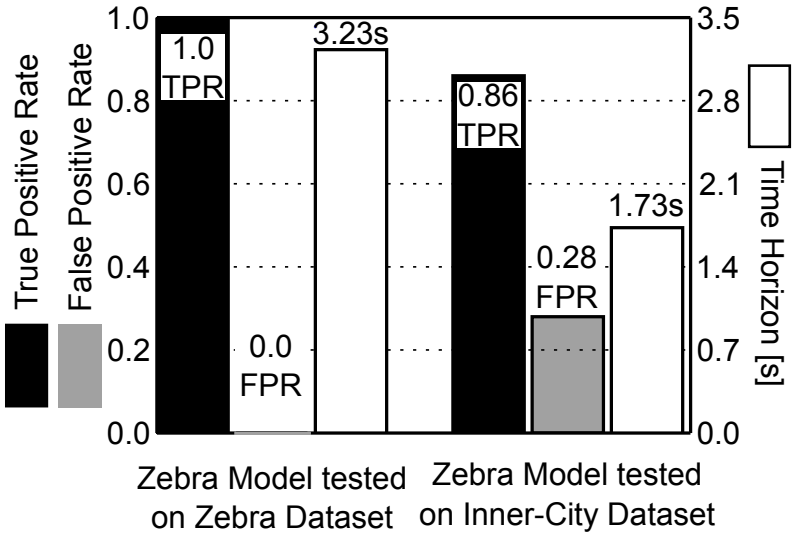


Figure 5.15: Performance of "Zebra Crossing" model. Black, grey and white bars show the TPR, FPR and PTH respectively. The left shows the performance on the 'zebra' set. The right shows the performance on the 'inner-city' set.

TPR of 86% for an average of 1.73s in advance. But, the model also produces a FPR of 28%. For a FPR of 0%, the model does not predict any pedestrian correctly, the TPR is 0%. We can see that the model performs well at predicting non priority crossings in inner-city with a very good PTH, but it does many errors when predicting non crossing pedestrians. At zebras, many more pedestrians cross the road. The "Zebra Crossing" model has therefore learned a tendency to over-predict crossing behaviors and produces many FP errors when applied to a different scenario.

As we showed in the previous chapter, combining multiple models can really improve the quality of the prediction. For that reason, the next section proposes to combine the "Inner-City" model with the "Zebra Crossing" model using the CMT framework.

5.5.5 Context Model Tree

The pedestrian crossing CMT consists of the "Inner-City" model as top node and the "Zebra Crossing" model as sub-node. When both nodes are active, each node scales its corresponding confidence and the sub-node applies equation (5.1) for the

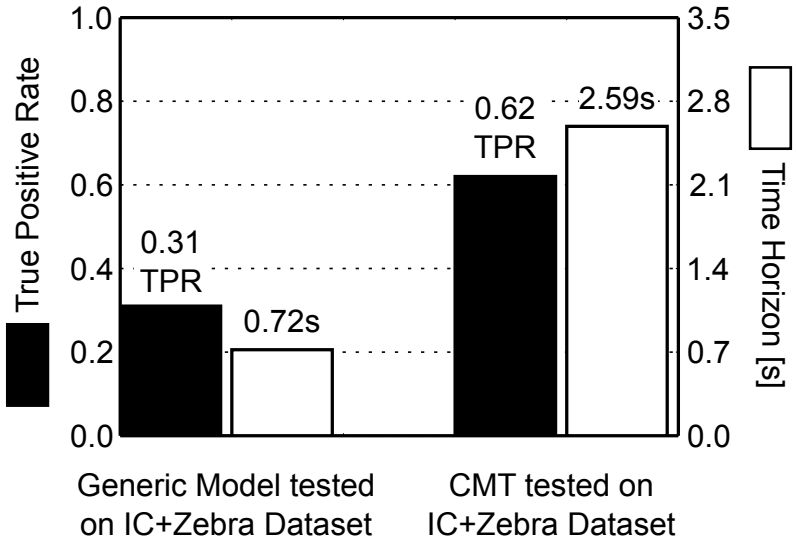


Figure 5.16: Performance of "Inner-City" model (left) versus CMT (right). Black and white bars show the TPR and PTH on the 'ICZ dataset'.

combination function in order to determine the best prediction. To evaluate the CMT, we again do an event based evaluation. For this evaluation, both models are already trained. The CMT is tested on the full 'ICZ dataset'.

Fig.5.16 shows that, at its best threshold, the generic "Inner-City" model has a TPR of 31% and can predict crossing behavior 0.72s in advance (see also Fig.5.11). In comparison, for the same test dataset, the CMT has a TPR of 62% and can predict crossing behavior 2.59s in advance without any errors.

As explained earlier, perception errors and intersection cases have been excluded from the dataset for these experiments. However, we have also evaluated our CMT on the entire dataset containing all excluded examples. The CMT has a TPR of 44% and on average it can predict crossings 2.72s in advance. It also produces prediction errors with a FPR of 6%, with 60% of the false positives due to a wrong pedestrian detection (tree, traffic sign), and 40% of the FPs occurring at intersections. To reduce the latter errors, one possibility would be to create a model specific to intersections and place it as a sub-node of the "Inner-City" node in the CMT.

Fig.5.17 shows four scenes from our dataset with the corresponding perception results and pedestrian predictions. In the scenes shown in Fig.5.17.(a), Fig.5.17.(b), and Fig.5.17.(d) pedestrians are detected by the pedestrian recognition algorithm. In Fig.5.17.(c), the pedestrian which is about 40m away from the ego-vehicle was manually annotated. In Fig.5.17.(a), pedestrians are walking towards the road but

they will not cross because they are waiting for a bus to arrive. The CMT correctly predicts that the pedestrians are not going to cross the road. In Fig.5.17.(b), the CMT correctly predicts that pedestrian 685 is going to cross the road in front of the ego-vehicle at a zebra crossing and pedestrian 665 is not. Pedestrian 685 is an instructed person. In Fig.5.17.(c), the CMT predicts that pedestrian 4615 is going to cross the road without priority. This is an example which shows that pedestrians that do not have any priority usually cross the road at a further distance of the ego-vehicle. In Fig.5.17.(d), the CMT predicts that pedestrian 145 is going to cross the road. In this scene, the CMT does a wrong prediction because it is an intersection where the ego-vehicle is turning left before the pedestrian intersects the ego-path.

These numbers show that using a CMT to combine multiple contextual models is a very suited solution. It can predict crossing pedestrians in inner-city with very few errors as well as predicting pedestrian at zebra crossings much earlier.

In reality, people do not cross often where they do not have the priority to do so. The CMT allows the combination between a generic model that is only predicting clear cases due to the higher impact of a FP with one that really focuses on predicting crossing behaviors as early as possible.

5.6 Discussion

In this chapter, we proposed a generic model designed to predict crossing behavior in inner-city and show tests on real inner-city data. Because the generic model had a limited prediction time horizon in specific areas where many people usually cross the road, we have additionally developed a "Zebra Crossing" model using more specific contextual information. We have shown that this specific model can predict crossing behaviors with a very good accuracy and a long prediction time horizon. However, it has a limited scope.

We applied our CMT framework to combine both models to predict if an entity is coming into the way of the ego-vehicle. The results of this combination of context based models show the benefit of using this framework to improve the accuracy of the prediction as well as the prediction time horizon.

Compared with the state of the art, we show an improved time horizon for a good performance. On similar data as [Keller and Gavrila, 2014], we can reach a performance comparable to humans (PTH 0.71s versus 0.5s for the "inner-city" dataset).

We have shown an event based evaluation of the different models. In contrast to the commonly applied frame-based evaluation, an event-based evaluation considers

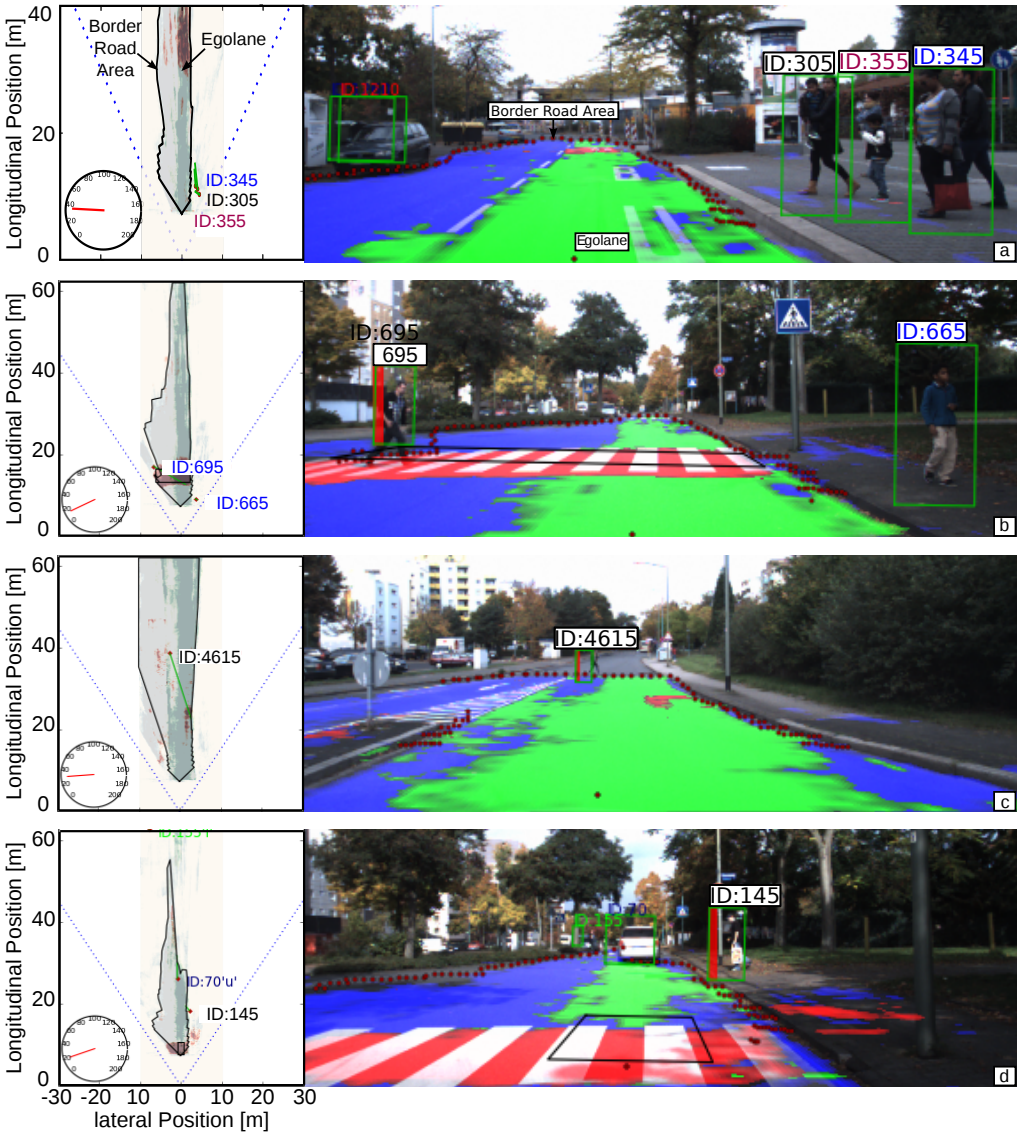


Figure 5.17: Four examples of prediction results. The top view (left) shows the ego-vehicle at the origin with its corresponding velocity. Dark blue represents the egolane. The black lane represents the border of the road area. The black rectangle around the red area represents the border of zebra crossings. Red dots represent pedestrians and the green line attached to them their relative velocity. The image view (right) shows the detected pedestrians inside rectangles. The vertical bar in a rectangle means that the pedestrian is predicted to cross the road.

the temporal grouping of system signals and the relevance of a predicted event for the driver of the ego vehicle. Our evaluation was based on the fraction of true positives/false positives. However, in order to analyze the online performances of our system and the acceptability to a driver, we plan to do an evaluation using the frequency of events (FP/TP per hour) (see also [Bonnin et al., 2014b]). For such experiments, we need to do additional recordings in order to build a dataset that contains the real frequency of crossing pedestrians. We might also use a different sensing setup to be able to detect pedestrians at a further distance without the need to manually annotate them.

In order to reduce prediction errors, we could develop a model to predict crossing behaviors at intersection and add it to the CMT. With an extended dataset, we might find other additional contexts for which prediction could be improved by designing specific models.

As the authors explain in [Schmidt and Faerber, 2009], pedestrians also take into account their distance to surrounding vehicles as well as their respective velocities. In our feature set, we only include the relations to the ego-vehicle, but in the future we want to extend this to other cars in the scene. Additionally, pedestrians strongly influence each other when crossing the road. People are up to 2.5 times more likely to cross if one of their neighbors does [Faria et al., 2010]. For that reason, it might be interesting to relate pedestrians that are in the same neighborhood.

6 Conclusion

The scientific community and the automotive industry have been working closely with each other for many years in order to provide us the safe vehicles that we use in our daily life. Studies show that current Advanced Driver Assistance Systems (ADAS) help to significantly lower the number of fatalities and seriously injured accident victims. To further reduce accidents and also assist the driver in preventing dangerous situations at all, those situations have to be recognized earlier. Currently, a warning is sent after a dangerous situation has started, therefore a driver cannot react early enough to fully avoid the collision. This thesis proposes a solution to warn drivers earlier and increase their reaction time. It proposes a generic framework to predict behaviors before they start.

As explained in the introduction, it is a challenging issue to predict the behavior of others early enough, so that it increases the time for a reaction, because a traffic participant has many possibilities for behaviors with many different ways to execute them for many different contexts. From related work we can see that context-based models are the most suitable to predict behaviors with a long prediction time horizon. However, this comes at the cost of a narrow scope. Such a model is specific to a scenario and becomes less accurate when applied to a different one.

The goal of this thesis was to provide a framework that allows us to construct systems which can predict behaviors for a long prediction time horizon for various scenarios. A solution to extend the scope is to combine context based-models. However, as we explained in chapter 3, combining models is not an easy task, and none of the existing related work provides a generic way of combining models. In this work, we propose a concept that allows for multiple model combination in order to have a prediction for a wide scope and a high quality. The proposed framework, called Context Model Tree (CMT), is a tree-structure in which models are ordered according to context specificity. Models designed to predict behaviors for a generic context are placed in the top of the tree and more context specific models are in the leaves. We have designed a mechanism to activate the model corresponding to the current context, as well as a solution based on a combination function to solve conflicts between two models that are active at the same time and produce different prediction outputs. We also provide a user's guide in order to allow anyone interested in doing prediction to build its own CMT. We explain the different cases

for which it can be relevant to build a CMT as well as how to build it.

We have demonstrated the benefit of using such a framework by implementing a Context Model Tree (CMT) to predict if a vehicle is coming into the way of the ego-vehicle when changing lane to the left on highways. For the top node of the CMT, we have used a model from a previous project designed to predict that a vehicle is going to change lane when approaching a slow predecessor. Additionally, we have implemented two context-based models to predict lane changes at entrances on highways. We could demonstrate that the CMT outperformed single models in terms of fewer false predictions and an increased prediction time horizon. We have explained the importance of combining models on the quality of the prediction.

We have shown that a CMT is a generic, reproducible and expandable framework in chapter 5, by extending the existing CMT for very different different scenarios. We have extended the CMT to predict if a pedestrian is coming into the way of the ego-vehicle when crossing the road in inner-city. We have implemented a model for the top node to predict crossing behaviors for inner-city and placed it parallel to the "Highway" node. Additionally, we have implemented a context-based model to predict pedestrians at zebra crossings and placed this as a sub-node. Again, this demonstrated the benefit of using such a framework to combine multiple-models and improve the prediction.

The four models that we have implemented ourselves have a similar structure. They contain several complex features based on context, relations between traffic participants and prior knowledge. The features are scaled to fit the characteristics of a single layer perceptron which is trained on annotated data. We have shown that with well defined features, it is possible to use a simple classifier which has the advantage to save computation time.

We chose to design the models to predict lane changes at entrances and giveway lanes on highways as well as crossing pedestrians at zebra crossings in inner-city because there are no existing models in the literature that are designed for these tasks. We also designed a model to predict crossing behaviors in inner-city because the existing models did not have the expected prediction time horizon. Otherwise, we would have used external models as we did in the top node of CMT for highways.

A potential ADAS application of our system would be an improved advanced cruise control (ACC) system. Using the prediction of a lane change maneuver of our right predecessor would allow us to maintain a safe distance before it enters our lane, and therefore smooth the deceleration of the ego-vehicle. We constructed our experimental setup with such an application in mind, for example by only evaluating those cars that would enter our lane below the safety distance. Tests in chapter 4 were done on German highways because they can be seen as a worst-case for

ACC due to the high range of relative speeds. Additionally, we could think about an "Advanced Pedestrian Crossing Warning" system that uses the prediction of a crossing pedestrian to warn a driver earlier. Tests in chapter 5 were done on German inner-city in areas well frequented by pedestrians, with many zebra crossings, bus stops and schools.

However, further steps towards a product would probably require additional adaptations. It might be necessary to include more scenarios on both, highways and inner-city, to adapt models to e.g. driving style in different countries, or tune the activation function to a given set of sensors. The framework proposed in this thesis was designed to allow such flexibility when building systems with a high quality and a wide scope.

References

Y. Abramson and B. Steux. Hardware-friendly pedestrian detection and impact prediction. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 590–595, 2004.

G. Agamennoni, J. I. Nieto, and E. M. Nebot. A bayesian approach for driving behavior inference. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 595–600, 2011.

Allgemeiner Deutscher Automobil-Club e.V. (ADAC). Comparative test of advanced emergency braking systems. Technical report, 2011.

S. Ammoun and F. Nashashibi. Real time trajectory prediction for collision risk estimation between vehicles. In *Proc. IEEE Intelligent Computer Communication and Processing*, pages 417–422, 2009.

G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How. Behavior classification algorithms at intersections and validation using naturalistic data. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 601–606, 2011.

M. Bahram, A. Lohrer, and M. Aeberhard. Generatives praediktionsmodell zur fruehzeitigen spurwechselerkennung. In *Proc. 9. Workshop Fahrerassistenzsysteme*, pages 47–54, 2014.

A. Barth and U. Franke. Where will the oncoming vehicle be the next second? In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1068–1073, 2008.

A. Barth and U. Franke. Tracking oncoming and turning vehicles at intersection. In *Proc. IEEE Intelligent Transportation Systems*, pages 861–868, 2010.

T. Batz, K. Watson, and J. Beyerer. Recognition of dangerous situations within a cooperative group of vehicles. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 907–912, 2009.

H. Berndt, J. Emmert, and K. Dietmayer. Continuous driver intention recognition with hidden markov models. In *Proc. IEEE Intelligent Transportation Systems*, pages 1189–1194, 2008.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

S. Bonnin, F. Kummert, and J. Schmuedderich. A generic concept of a system for predicting driving behaviors. In *Proc. IEEE Intelligent Transportation Systems*, pages 1803–1808, 2012.

S. Bonnin, T. H. Weisswange, F. Kummert, and J. Schmuedderich. Accurate behavior prediction on highways based on a systematic combination of classifiers. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 242–249, 2013.

S. Bonnin, T. H. Weisswange, F. Kummert, and J. Schmuedderich. Pedestrian crossing prediction using multiple context-based models. In *Proc. IEEE Intelligent Transportation Systems*, 2014a.

S. Bonnin, T. H. Weisswange, F. Kummert, and J. Schmuedderich. General behavior prediction by a combination of scenario-specific models. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1478–1488, 2014b.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Z. Chen and N. H. C. Yung. Improved multi-level pedestrian behavior prediction based on matching with classified motion patterns. In *Proc.IEEE Intelligent Transportation Systems*, pages 1–6, 2009.

E. Coelingh, A. Eidehall, and M. Bengtsson. Collision warning with full auto brake and pedestrian prediction - a practical example of automatic emergency braking. In *Proc. IEEE Intelligent Transportation Systems*, pages 155–160, 2010.

I. Dagli, G. Breuel, H. Schittenhelm, and A. Schanz. Cutting-in vehicle recognition for ACC systems-towards feasible situation analysis methodologies. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 925–930, 2004.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 886–893, 2005.

Directorate-General Transport and Statistical Office of the European Communities. *EU Transport in Figures: Statistical Pocketbook*. 2012.

U. Dogan and H. Edelbrunner. Towards a driver model: Preliminary study of lane change behavior. In *Proc. IEEE Intelligent Transportation Systems*, pages 931–937, 2008.

P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 304–311, 2009.

E. Eckermann. *World History of the Automobile*. 2001.

M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009.

J. J. Faria, S. Krause, and J. Krause. Collective behavior in road crossing pedestrians: The role of social information. *Behavioral Ecology*, 21(6):1236–1242, 2010.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters In ROC Analysis in Pattern Recognition*, 27(8):861–874, 2006.

D. Ferguson, M. Darms, C. Urmson, and S. Kolski. Detection, prediction, and avoidance of dynamic obstacles in urban environments. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1149–1154, 2008.

J. Fritsch, T. Kuehnl, and F. Kummert. Monocular road terrain detection by combining visual and spatial information. *IEEE Transactions on Intelligent Transportation Systems*, 15(4): 1524–9050, 2014.

T. Gandhi and M. M. Trivedi. Pedestrian collision avoidance systems: A survey of computer vision based recent studies. In *Proc. IEEE Intelligent Transportation Systems*, pages 976–981, 2006.

T. Gandhi and M. M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. In *Proc.IEEE Intelligent Transportation Systems*, pages 413–430, 2007.

M. Garcia Ortiz, F. Kummert, and J. Schmuedderich. Prediction of driver behavior on a limited sensory setting. In *Proc. IEEE Intelligent Transportation Systems*, pages 638–643, 2012.

A. Gepperth, M. Garcia Ortiz, and B. Heisele. Real-time pedestrian detection and pose classification on a GPU. In *Proc. IEEE Intelligent Transportation Systems*, pages 348–353, 2013.

A. Gerdes. Automatic maneuver recognition in the automobile: The fusion of uncertain sensor values using bayesian models. In *Proc. IEEE Intelligent Transportation*, pages 129–133, 2006.

T. Gindele, S. Brechtel, and R. Dillmann. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In *Proc. IEEE Intelligent Transportation Systems*, pages 1625–1631, 2010.

R. Graf, H. Deusch, M. Fritzsche, and K. Dietmayer. A learning concept for behavior prediction in traffic situations. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 672–677, 2013.

R. Graf, F. Seeliger, H. Deusch, M. Fritzsche, and K. Dietmayer. A learning concept for behavior prediction at intersections. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 939–945, 2014.

C. Grover, I. Knight, F. Okoro, I. Simmons, G. Couper, P. Massie, and B. Smith. Automated emergency brake systems: Technical requirements, costs and benefits. Technical report, 2013.

A. Guyon, I. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

L. He, C. f. Zong, and C. Wang. Driving intention recognition and behaviour prediction based on a double-layer hidden markov model. *Journal of Zhejiang University - Science C*, 13(3): 208–217, 2012.

C. Hermes, C. Wöhler, K. Schenk, and F. Kummert. Long-term vehicle motion prediction. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 652–657, 2009.

C. Hermes, J. Einhaus, M. Hahn, C. Wöhler, and F. Kummert. Vehicle tracking and motion prediction in complex urban scenarios. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 26–33, 2010.

J. I. Herrero Zarzosa, M. E. G. Biraud, S. Damiani, J. P. Magny, J. Merino, and E. Koren. REPOSIT: Relative positioning for collision avoidance systems. Technical report, 2007.

S. Hold, S. Görmer, A. Kummert, M. Meuter, and S. Müller-Schneiders. ELA - an exit lane assistant for adaptive cruise control and navigation systems. In *Proc. IEEE Intelligent Transportation Systems*, pages 629–634, 2010.

W. Hu, X. Xiao, D. Xie, and T. Tan. Traffic accident prediction using vehicle tracking and trajectory analysis. In *Proc. IEEE Intelligent Transportation Systems*, pages 220–225, 2003.

T. Hülnhagen, I. Dengler, A. Tamke, T. Dang, and G. Breuel. Maneuver recognition using probabilistic finite-state machines and fuzzy logic. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 65–70, 2010.

M. Hülsen, J. M. Zöllner, and C. Weiss. Traffic intersection situation description ontology for advanced driver assistance. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 991–997, 2011.

Institute of Advanced Motorists (IAM). Licensed to skill contributory factors in road accidents great britain 2005-2009. Technical report, 2011.

J. Jansson, J. Johansson, and F. Gustafsson. Decision making for collision avoidance systems. In *SAE 2002 World Congress and Exhibition*, 2002.

S. Jin, D.-H. Wang, C. Xu, and D.-F. Ma. Short-term traffic safety forecasting using gaussian mixture model and kalman filter. *Journal of Zhejiang University - Science A: Applied Physics & Engineering*, 14(4):231–243, 2013.

N. Kaempchen, K. Weiss, M. Schaefer, and K. C. J. Dietmayer. IMM object tracking for high dynamic driving maneuvers. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 825–830, 2004.

E. Käfer, C. Hermes, C. Woehler, H. Ritter, and F. Kummert. Recognition of dangerous situations within a cooperative group of vehicles. In *Proc. IEEE Robotics and Automation*, pages 3960–3965, 2010.

D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel. Object-oriented bayesian networks for detection of lane change maneuvers. *IEEE Intelligent Transportation Systems Magazine*, 4(3):19–31, 2012.

C. G. Keller and D. M. Gavrila. Will the pedestrian cross? A study on pedestrian path prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):494–506, 2014.

C. G. Keller, C. Hermes, and D. M. Gavrila. Will the pedestrian cross? Probabilistic path prediction based on learned motion features. In *Proc. Pattern recognition*, pages 386–395, 2011.

S. Kim, S. J. Guy, W. Liu, R. W. H. Lau, M. C. Lin, and D. Manocha. Predicting pedestrian trajectories using velocity-space reasoning. In *Algorithmic Foundations of Robotics*, pages 609–623, 2012.

S. Klingelschmitt, M. Platho, M.-M. Gross, V. Willert, and J. Eggert. Combining behavior and situation information for reliably estimating multiple intentions. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 388–393, 2014.

S. Koehler, M. Goldhammer, S. Bauer, K. Doll, U. Brunsmann, and K. Dietmayer. Early detection of the pedestrian's intention to cross the street. In *Proc. IEEE Intelligent Transportation Systems*, pages 1759–1764, 2012.

S. Koehler, B. Schreiner, S. Ronalter, K. Doll, U. Brunsmann, and K. Zindler. Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 519–526, 2013.

J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila. Context-based pedestrian path prediction. *Computer Vision - ECCV*, 8694:618–633, 2014a.

J. F. P. Kooij, N. Schneider, and D. M. Gavrila. Analysis of pedestrian dynamics from a vehicle perspective. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1445–1450, 2014b.

N. Köster. Improved spatial features for road terrain detection. Master's thesis, Technische Fakultät, Universität Bielefeld, 2012.

T. Kuehnl, F. Kummert, and J. Fritsch. Visual ego-vehicle lane assignment using spatial ray features. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1101–1106, 2013.

N. Kuge, T. Yamamura, and O. A. L. Shimoyama. A driver behavior recognition method based on a driver model framework. In *Proc. IEEE Intelligent vehicles Symposium*, pages 47–54, 2000.

T. Kumagai and M. Akamatsu. Prediction of human driving behavior using dynamic bayesian networks. *IEICE Transactions on information and systems*, E89(2):857–860, 2006.

P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier. Learning-based approach for online lane change intention prediction. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 797–802, 2013.

S. Kumar, P. Ranganath, H. Weimin, and K. Sengupta. Framework for real-time behavior interpretation from traffic video. *IEEE Transactions on Intelligent Transportation Systems*, 6(1): 43–53, 2005.

U. Lages. Automatic scenario generation by advanced offline processing for groundtruth evaluation. Technical report, 2013.

F. Large, D. Vasquez, T. Fraichard, and C. Laugier. Avoiding cars and pedestrians using velocity obstacles and motion prediction. In *Proc.IEEE Intelligent Vehicles Symposium*, pages 375–379, 2004.

S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán. Exploiting map information for driver intention estimation at road intersections. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 583–588, 2011.

N. D. Lerner. Brake perception-reaction times of older and younger drivers. In *Proc. Human Factors and Ergonomics Society*, pages 206–210, 1993.

M. Liebner, M. Baumann, F. Klanner, and C. Stiller. Driver intent inference at urban intersections using the intelligent driver model. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 1162–1167, 2012.

A. Liu and A.Pentland. Towards real-time recognition of driver intentions. In *Proc. IEEE Intelligent Transportation Systems*, pages 236–241, 1997.

C. C. Liu, S. G. Hosking, and M. G. Lenné. Predicting driver drowsiness using vehicle measures: Recent insights and future challenges. *Journal of Safety Research*, 40(4):239–245, 2009.

W. Liu, H. Zhang, B. Duan, H. Yuan, and H. Zhao. Vision-based real-time lane marking detection and tracking. In *Proc. IEEE Intelligent Transportation Systems*, pages 49–54, 2008.

G.-S. Ma, J. Yao, and X.-G. Yang. Prediction of the position of pedestrian crossing road section based on kalman predictor. In *Measuring Technology and Mechatronics*, pages 541–544, 2009.

D. Makris and T. Ellis. Spatial and probabilistic modelling of pedestrian behaviour. In *British Machine Vision*, pages 557–566, 2002.

A. Martin. Factors influencing pedestrian safety: A literature review. Technical report, 2006.

B. Mathibela, M. A. Osborne, I. Posner, and P. Newman. Can priors be trusted? Learning to anticipate roadworks. In *Proc. IEEE Intelligent Transportation Systems*, pages 927–932, 2012.

J. C. Mccall, D. Wipf, M. M. Trivedi, and B. Rao. Lane change intent analysis using robust operators and sparse bayesian learning. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 431–440, 2005.

D. Meyer-Delius, J. Sturm, and W. Burgard. Regression-based online situation recognition for vehicular traffic scenarios. In *Proc. IEEE Intelligent Robots and Systems*, pages 1711–1716, 2009.

T. Michael and P. Fernando. Lane detection and tracking based on lidar data. *Metrology and Measurement Systems*, XVII(3):311–322, 2010.

MobilEye Ltd. C2-270 collision prevention system user manual. Technical report, 2007.

M. C. Montel, T. Brenac, M. A. Granié, M. Millot, and C. Coquelet. Urban environments, pedestrian-friendliness and crossing decisions. In *Transportation Research Board 92nd Annual Meeting*, pages 1–13, 2013.

B. Morris, A. Doshi, and M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 895–901, 2011.

B. Musleh, F. Garca, J. Otamendi, J. M. Armingol, and A. De la Escalera. Identifying and tracking pedestrians based on sensor fusion and motion stability predictions. *Sensors*, 10(9): 8028–8053, 2010.

D. Mušicki and B. La Scala. Multi-target tracking in clutter without measurement assignment. 44(3):877–896, 2008.

T. Naito, H. Morimoto, T. Ito, T. Yamamoto, and H. Fukumoto. Driver turning behavior prediction at intersection improving prediction performance with driver behavior adaptation. In *Review of Automative Engineering*, pages 115–122, 2007.

National Highway Traffic Safety Administration (NHTSA). Automotive collision avoidance systems (ACAS) program. Technical report, 2000.

National Highway Traffic Safety Administration (NHTSA). Distracted driving 2009. Technical report, 2010.

M. Nishigaki, S. Rebhan, and N. Einecke. Vision-based lateral position improvement of radar detections. In *Proc. IEEE Intelligent Transportation Systems*, pages 90–97, 2012.

E. Ohn-Bar, A. Tawari, S. Martin, and M. M. Trivedi. Predicting driver maneuvers by learning holistic features. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 719–724, 2014.

N. Oliver and A. P. Pentland. Driver behavior recognition and prediction in a smartcar. In *Proc. of SPIE*, pages 280–290, 2000.

M. Peden, R. Scurfield, D. Sleet, D. Mohan, A. A. Hyder, E. Jarawan, and C. Mathers. *World report on road traffic injury prevention.* 2004.

A. Pentland and A. Liu. Modeling and prediction of human behavior. *Neural Computation*, 11 (1):229–242, 1999.

M. Platho and J. Eggert. Deciding what to inspect first: Incremental situation assessment based on information gain. In *Proc. IEEE Intelligent Transportation Systems*, pages 888–893, 2012.

A. Polychronopoulos, M. Tsogas, A. Amditis, U. Scheunert, L. Andreone, and F. Tango. Dynamic situation and threat assessment for collision warning systems: The EUCLIDE approach. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 636–641, 2004.

R. Quintero, J. Almeida, D. F. Llorca, and M. A. Sotelo. Pedestrian path prediction using body language traits. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 317–323, 2014.

L. R. Rabiner. A tutorial on hidden markov models and selected aapplications in speech recognition. In *Proc. IEEE*, pages 257–286, 1989.

M. Reichel, R. Botsch, M.and Rauschecker, K. H. Siedersberger, and M. Maurer. Situation aspect modelling and classification using the scenario based random forest algorithm for convoy merging situations. In *Proc. IEEE Intelligent Transportation Systems*, pages 360–366, 2010.

D. D. Salvucci, H. M. Mandalia, N. Kuge, and T. Yamamura. Lane change detection using a computational driver model. *Human Factors*, 49(3):532–542, 2007.

N. Saunier, T. Sayed, and C. Lim. Probabilistic collision prediction for vision-based automated road safety analysis. In *Proc. IEEE Intelligent Transportation Systems*, pages 872–878, 2007.

J. Schlechtriemen, A. Wedel, J. Hillenbrand, J. Breuel, and K.-D. Kuhnert. A lane change detection approach using feature ranking with maximized predictive power. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 108–114, 2014.

S. Schmidt and B. Faerber. Pedestrians at the kerb recognising the action intentions of humans. *Transportation Research Part F: Traffic Psychology and Behaviour*, 12(4):300–310, 2009.

J. Schmuedderich and S. Rebhan. A method and system for predicting movement behavior of a target traffic object. Patent number: EP20110178252, 2011.

N. Schneider and D. M. Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *Proc. IEEE Pattern Recognition*, pages 174–183, 2013.

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

N. Sekiyama and T. Watanabe. A method for calculating future behaviors of vehicles toward effective collision prediction. In *Proc. IEEE Engineering and Computer Science*, pages 2–7, 2009.

K. Takagi, K. Morikawa, T. Ogawa, and M. Saburi. Road environment recognition using on-vehicle lidar. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 120–125, 2006.

T. Taniguchi, S. Nagasaka, K. Hitomi, N. P. Chandrasiri, and T. Bando. Semiotic prediction of driving behavior using unsupervised double articulation analyzer. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 849–854, 2012.

G. T. Taoka. Brake reaction times of unalerted drivers. *Journal Institute of Transportation Engineers*, 59(3):19–21, 1989.

M. Thuy and F. Puente Leon. Lane detection and tracking based on lidar data. *Metrology and Measurement Systems*, 17(3):311–322, 2010.

R. Tian, L. Li, K. Yang, S. Chien, Y. Chen, and R. Sherony. Estimation of the vehicle-pedestrian encounter/conflict risk on the road based on tasi 110-car naturalistic driving data collection. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 623–629, 2014.

M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

R. S. Tomar and S. Verma. Safety of lane change maneuver through a priori prediction of trajectory using neural networks. *Network Protocols and Algorithms*, 4(1):4–21, 2012.

Q. Tran and J. Firl. Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non- parametric regression. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 918–923, 2014.

M. Tsogas, X. Dai, G. Thomaidis, P. Lytrivis, and A. Amditis. Detection of maneuvers using evidence theory. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 126–131, 2008.

United States. Dept. of the Treasury. Bureau of Statistics, U. S. D. of Commerce, L. B. of Statistics, U. S. B. of Foreign, D. Commerce, U. S. B. of the Census, and U. C. Bureau. *Statistical Abstract of the United States*. 2010.

C. Urmson, J. Joshua Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(1):425–466, 2008.

D. Vasquez, T. Fraichard, and C. Laugier. Incremental learning of statistical motion patterns with growing hidden markov models. *IEEE Transactions on Intelligent Transportation Systems*, 10 (3):403–416, 2009.

S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17(2):2602–2634, 2005.

Volvo Car Corporation. Road and traffic information system, v40, s60, v60,xc60, v70, xc70 & s80. Technical report, 2012.

C. Wakim, S. Capperon, and J. Oksman. A markovian model of pedestrian behavior. In *Proc. Systems, Man and Cybernetics*, pages 4028–4033, 2004.

C. Wang, Z. Hu, and R. Chapuis. Predictive lane detection by interaction with digital road map. *Journal of Information Processing*, 20(1):287–296, 2012.

T. H. Weisswange, B. Bolder, J. Fritsch, S. Hasler, and C. Goerick. An integrated adas for assessing risky situations in urban driving. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 292–297, 2013.