Context-based Cyclist Path Prediction

Crafted and Learned Models for Intelligent Vehicles

Pool, E.A.I.

DOI
10.4233/uuid:a5689f32-6eed-4949-9527-60723e16c8b5

Publication date
2021

Document Version
Final published version

Citation (APA)
Pool, E. A. I. (2021). *Context-based Cyclist Path Prediction: Crafted and Learned Models for Intelligent Vehicles*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a5689f32-6eed-4949-9527-60723e16c8b5

CONTEXT-BASED CYCLIST PATH PREDICTION

CRAFTED AND LEARNED MODELS FOR INTELLIGENT VEHICLES

Ewoud Alexander Ignacz Pool

CONTEXT-BASED CYCLIST PATH PREDICTION:

CRAFTED AND LEARNED MODELS FOR INTELLIGENT VEHICLES

CONTEXT-BASED CYCLIST PATH PREDICTION:

CRAFTED AND LEARNED MODELS FOR INTELLIGENT VEHICLES

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 7 juni 2021 om 17:30 uur

door

Ewoud Alexander Ignacz POOL

Master of Science in Systems and Control,
Technische Universiteit Delft, Nederland,
geboren te Amsterdam, Nederland.

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotie commissie bestaat uit:

Rector Magnificus,	voorzitter
Prof.dr. D.M. Gavrila	Technische Universiteit Delft, promotor
Dr. J.F.P. Kooij	Technische Universiteit Delft, copromotor
onafhankelijke leden:	
Prof.dr. R. Babuska	Technische Universiteit Delft
Prof.dr. M.Á. Sotelo	Universiteit van Alcalá, Spanje
Prof.dr. A. De La Fortelle	Technische Universiteit Mines ParisTech, Frankrijk
Dr. G. Dubbelman	Technische Universiteit Eindhoven
Prof.dr.ir. M. Wisse	Technische Universiteit Delft, reservelid

TU Delft Delft University of Technology NWO Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Keywords: Context modeling, Predictive models, Intelligent vehicles

Printed by: Ridderprint

Front & Back: Sandra Tukker

ISBN 978-94-6416-489-3

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Getting funky on the mic like a old batch of collard greens.

Snoop Dogg

CONTENTS

SUMMARY

This thesis addresses the problem of path prediction for cyclists. Instead of solely focusing on how to predict the future trajectory based on previous position measurements, this thesis investigates how to leverage additional contextual information that can inform on the future intent of cyclists. This thesis does this with the application of intelligent vehicles in mind. That means all measurements come from the point of view of a vehicle on the road. Additionally, the resulting predictions must be usable by a motion planner. In practice, this means the predictions are a probability distribution over the future position rather than a single point in space.

This thesis starts with an investigation of one of the modules that allow path prediction in the first place: 3D object detection. Two existing state-of-the-art 3D object detectors that exploit Lidar data are evaluated beyond the standard metrics of 3D object detection. 3D object detectors predict an oriented 3D bounding box. The standard metric determines a correct detection based on the accuracy of the position, extent, and orientation of the bounding box all at once. By loosening the requirements for when a detection is considered correct, the accuracy of the estimated position, extent, and orientation can be evaluated separately. The results show that a large number of detections are considered incorrect largely because of inaccurate bounding box extent rather than bounding box position, which is arguably a more important aspect for path prediction. As a result, the performance of these 3D object detectors when used for path prediction can be considered to be higher than what the common metrics suggest.

After this, this thesis investigates how knowledge of the road topology can be used to improve the accuracy of cyclist path prediction. The trajectories of cyclists near an intersection are extracted from a naturalistic cyclist detection dataset. These are categorized and grouped based on the action taken by each cyclist (hard left/right, slight left/right, or straight). A Linear Dynamical System (LDS) is fitted on each group. These LDSs are used together to create a Mixture of Linear Dynamical Systems (MoLDS). During online inference, the relative probability of each underlying LDS allows the MoLDS to evaluate which direction the cyclist is most likely to take. This chapter demonstrates that the highest prediction accuracy is obtained when this model is additionally given prior knowledge on which directions are available for the cyclist to take.

Next, context cues related to a specific scenario are considered. In the scenario, a cyclist in front of the ego-vehicle approaches an intersection and has the option to either continue straight or turn left. The three context cues considered are the distance of the cyclist to the intersection, whether the cyclist is raising their arm, and the criticality of the situation. This last context cue is based on the time it will take the ego-vehicle to overtake the cyclist: the lower this is, the more risk a left turn brings. This scenario is first modeled with a Switching Linear Dynamical System (SLDS) with two motion models that represent "cycling straight" and "turning left", respectively. This model does not yet use any context cues. Still, the SLDS is shown to outperform a baseline model that represents the scenario with a single

motion model. By letting the context cues inform the SLDS whether switching from one motion model to the other is likely to happen the performance is increased even further. The resulting model is referred to as a Dynamic Bayesian Network (DBN).

The context-based path prediction methods described so far have been designed with specific motion models and interplay of context cues in mind: the overall state representation has been hand-crafted. The advantage of this approach is that the state representation is then interpretable, making it easy to understand why a model predicts what it does, even when it fails to predict something correctly. However, methods with a learned state representation often attain higher performances. The next point of investigation of this thesis is then to compare a model with a crafted state representation to a model with a learned one. Specifically, the DBN is compared to a Recurrent Neural Network (RNN), using the cyclist scenario from before. To level the playing field as much as possible two actions are taken. First, the contextual cues are supplied to the RNN as well, and experiments assert that the performance of the RNN does in fact improve when it incorporates these cues. Secondly, the optimization method used in the RNN is applied to the DBN as well, but in such a way that the interpretation of its crafted state representation remains the same. Of the two methods, the RNN attains the highest performance. Still, optimizing the DBN largely closes the performance gap between the two.

Finally, this thesis determines whether the DBN is not only performant but also useful in practice: it is integrated in an intelligent vehicle. The cyclist scenario is performed live, in which the intelligent vehicle extracts the relevant context cues directly from sensor data. The resulting predictions are used to create an early warning system for the driver, to warn them if the cyclist intends to turn left. The model is also used for predictions in an autonomously driving intelligent vehicle, but due to safety reasons on a different scenario that contains comparable contextual cues. An automated dummy plays the role of a pedestrian on the sidewalk who walks towards the curbside in order to cross the road. The intelligent vehicle is driving on this road towards the pedestrian and has right of way. In this scenario, a pedestrian is only expected to cross the road if they are unaware of the approaching vehicle. Furthermore, if they will stop, they are expected to only stop at the curbside. The intelligent vehicle determines whether the pedestrian is aware of it by estimating the head orientation of the pedestrian. Additionally, it measures the distance between the pedestrian and the curbside, and predicts the future trajectory of the pedestrian accordingly. With the model in place, the vehicle can autonomously follow a planned trajectory and evade the pedestrian if the pedestrian does indeed cross the road. The real-world experiments confirm the feasibility of the system. By evaluating the entire pipeline at once, from detections to motion planning, this chapter is able to propose future work that bridges these various disciplines and shows what intelligent vehicles can already realistically achieve.

SAMENVATTING

Deze thesis adresseert het voorspellen van het toekomstige pad van fietsers. Dit wordt niet alleen gedaan aan de hand van de positie van de fietser in het verleden: deze thesis onderzoekt hoe relevante contextuele informatie gebruikt kan worden om de nauwkeurigheid van de voorspellingen te verbeteren. Dit wordt gedaan met intelligente voertuigen als uiteindelijke toepassing. Dat betekent dat zowel de positiemetingen als alle gebruikte contextuele informatie meetbaar moeten zijn met de sensoren van een intelligent voertuig dat op de weg rijdt. Daar bovenop moeten de resulterende voorspellingen bruikbaar zijn voor de module die het voertuig autonoom bestuurt. In de praktijk betekent dit dat alle voorspellingen een kansverdeling over de toekomstige positie zijn.

Deze thesis begint met het onderzoeken van een van de modules zonder welk het voorspellen niet mogelijk zou zijn: de detectie van objecten in 3D. Twee van de best presterende 3D object detectors die gebruik maken van Lidar data worden geëvalueerd op de standaard metriek van 3D object detectie. 3D object detectoren leveren hun detecties aan in de vorm van een *3D bounding box*: een balk gedefinieerd aan de hand van zijn 3D positie, zijn dimensies en een oriëntatie. De standaard metriek bepaalt of een voorspelde 3D bounding box correct is aan de hand van een combinatie van alle aspecten van die 3D bounding box. Door de standaard definitie van een correctie detectie te versoepelen is het mogelijk om de correctheid van de geschatte positie, afmetingen en oriëntatie los van elkaar te evalueren. Uit deze evaluatie blijkt dat een groot aantal detecties voornamelijk als incorrect wordt gezien door een onnauwkeurige schatting van de dimensies van de 3D bounding box en niet zo zeer de locatie, terwijl redelijkerwijs de locatie gezien kan worden als een belangrijker aspect voor pad predictie. Het resultaat hiervan is dat de competentie van deze 3D object detectoren specifiek bij het gebruik van pad predictie hoger is dan wat de officiële metriek suggereert.

Hierna onderzoekt deze thesis hoe kennis van de layout van een kruispunt gebruikt kan worden om de nauwkeurigheid van pad predictie van fietsers te verbeteren. De afgelegde wegen van fietsers in de buurt van een kruispunt worden geëxtraheerd uit een naturalistische dataset gemaakt voor het detecteren van fietsers. Deze worden gecategoriseerd en gegroepeerd op basis van de richting die de fietser beweegt (hard naar links of rechts, licht naar links of rechts, of rechtdoor). Een *Linear Dynamical System* (LDS) wordt gepast op elke groep. Deze LDS'en worden samengevoegd om een zogenaamd *Mixture of Linear Dynamical Systems* (MoLDS) te creëren. Dit model kan vervolgens online afleiden welke richting de fietser het meest waarschijnlijk op zal gaan, gebaseerd op de relatieve waarschijnlijkheid van de onderliggende LDS die bij die richting hoort. Dit hoofdstuk laat zien dat de meest nauwkeurige voorspellingen van de toekomstige locatie van de fietser gemaakt kunnen worden als in het model voorkennis mee wordt genomen over welke kant de fietser op het kruispunt op kan gaan.

Vervolgens wordt een specifiek scenario onderzocht: een fietser fiets rechtdoor voor het intelligente voertuig, en beide bewegen zich richting een kruispunt. Op dit kruispunt kan de fietser ofwel rechtdoor gaan, ofwel linksaf slaan. In dit scenario zijn er meerdere contextu-

ele informatiebronnen relevant. Er worden er drie beschouwd: de afstand van de fietser tot de kruising, of de fietser zijn arm opsteekt, en hoe kritiek de situatie is. Dit laatste wordt bepaald aan de hand van hoe lang het duurt totdat de auto de fietser ingehaald zal hebben. Dit scenario wordt eerst gemodelleerd met een *Switching Linear Dynamical System* (SLDS), waarin twee dynamische modellen respectievelijk "rechtdoor fietsen" en "naar links afslaan" representeren. Deze SLDS neemt nog geen contextuele informatie mee, maar laat al betere resultaten zien dan een LDS dat het complete scenario met maar één dynamisch model representeert. Door de contextuele informatie in de SLDS te gebruiken als informatiebron of de dynamiek mogelijk gaat wisselen van "rechtdoor" naar "linksaf" (dit model wordt een *Dynamic Bayesian Network* (DBN) genoemd) wordt een nog nauwkeurigere voorspelling gemaakt.

De hierboven beschreven methodes zijn handmatig ontworpen. Ze beschrijven specifieke dynamische modellen en specifieke manieren hoe de contextuele informatie deze modellen kunnen beïnvloeden. Het voordeel van deze aanpak is dat hun toestandsrepresentatie vervolgens interpreteerbaar is, bijvoorbeeld in termen van "positie" en "snelheid". Dit maakt het makkelijk om te begrijpen waarom een model een specifieke voorspelling maakt, en het maakt het ook makkelijk te begrijpen waar de fout zit als de voorspelling niet klopt. Modellen waar de toestandsrepresentatie niet ontworpen is maar automatisch geleerd wordt zijn echter vaak nauwkeuriger in het voorspellen. Het volgende punt dat deze thesis onderzoekt is een zo eerlijk mogelijk vergelijk tussen een model dat een onderworpen toestandsrepresentatie heeft (het DBN van hiervoor) en een model dat een geleerde toestandsrepresentatie heeft, een *Recurrent Neural Network* (RNN). Om er voor te zorgen dat de vergelijk alleen gaat over hun representatie worden de modellen zo gelijk mogelijk behandeld. Als eerst wordt dezelfde contextuele informatie in het RNN verwerkt, en wordt er vastgesteld dat dit RNN er daadwerkelijk nauwkeuriger mee kan voorspellen, net als hoe het DBN dit kan. Ten tweede wordt de optimalisatie methode die het RNN gebruikt om zijn toestandsrepresentatie te leren ook gebruikt voor de DBN. Dit wordt gedaan op zo'n manier dat de toestandsrepresentatie nog steeds interpreteerbaar is. Uit de vergelijk blijkt dat het RNN de nauwkeurigste voorspellingen maakt. Door het DBN te optimaliseren loopt de nauwkeurigheid van het DBN echter een stuk minder achter op die van het RNN.

Als laatste achterhaalt deze thesis of de DBN ook daadwerkelijk bruikbaar is in de praktijk door het te integreren in een intelligent voertuig. Het fiets scenario wordt in het echt uitgevoerd, waarbij het intelligente voertuig live de relevante context informatie moet extraheren uit sensor data. De voorspellingen van het DBN worden gebruikt om een waarschuwingssysteem te ontwerpen dat de bestuurder vroegtijdig waarschuwt als de fietser van plan is om af te slaan. Het model wordt ook ingezet om een volledig autonoom voertuig te helpen een mogelijke botsing te vermijden. Om veiligheidsredenen wordt dit gedaan op een ander scenario met typen van vergelijkbare contextuele informatie. Een geautomatiseerde dummy speelt de rol van een voetganger die richting de stoeprand loopt om een weg over te steken. Op deze weg komt het zelfrijdende voertuig aanrijden, en deze heeft voorrang. In dit scenario wordt er verwacht dat voetgangers alleen zullen oversteken als zij zich niet gewaar zijn van het aanrijdende voertuig. Mochten zij van plan zijn te stoppen, dan zullen ze dit doen dichtbij de rand van de stoep. Het intelligente voertuig bepaalt of deze voetganger gewaar is van het aankomend voertuig door middel van het schatten van de oriëntatie van het hoofd van de voetganger. Verder wordt er gebruik gemaakt van de afstand tussen de stoeprand en

de voetganger om te voorspellen tot waar de voetganger door zal lopen als deze van plan is te stoppen. Met dit model kan het voertuig autonoom een pad volgen en de voetganger ontwijken mocht deze inderdaad oversteken. De evaluatie van het systeem als geheel leidt tot aanbevelingen voor verder onderzoek dat de verscheidene disciplines beter aan elkaar zal laten aansluiten, en laat zien waar intelligente voertuigen al toe in staat zijn.

1

INTRODUCTION

A bicycle ride around the world begins with a single pedal stroke.

Scott Stoll

1

EVERY day, millions of people get in their car to drive to work, friends, home, or any other destination. The combination of speed and flexibility that a car has is virtually unrivaled by any other transportation method. Arguably the most complex component of this transportation method is the driver itself. We, the drivers, are what makes it possible to take the car from one bustling inner city, on the highway, and into another.

However, that does not mean that the human driver is perfect. The World Health Organization estimated that 1.35 million people died in traffic-related accidents in 2018 worldwide [1]. A large portion of these deaths is a result of driver error such as inattentiveness, drowsiness, or drunk driving. Additionally, people do not necessarily *want* to drive. Many of the hours driven are seen as hours wasted: traffic jams being the obvious culprit, but in many cases, people solely drive to get to the destination rather than for the journey.

To alleviate both issues, universities and companies around the world are working on the development of intelligent vehicles [2–6]. Some of these are vehicles that assist the driver with Advanced Driver Assistance Systems (ADAS) such as lane guidance and emergency brake assistance. Others are intelligent vehicles that go one step further and require no driver at all, for example, those of Waymo in Phoenix, USA.

In the development of intelligent vehicles, many difficult situations arise in the urban centers of the cities that were never designed for vehicles in the first place. Here, vehicles drive close to (or even in the same space as) pedestrians and cyclists who may quickly change direction – or not. An intelligent vehicle must be able to reason about that uncertainty inherent in the behavior of others around it. At the same time, the uncertainty cannot be so big that no maneuver seems safe.

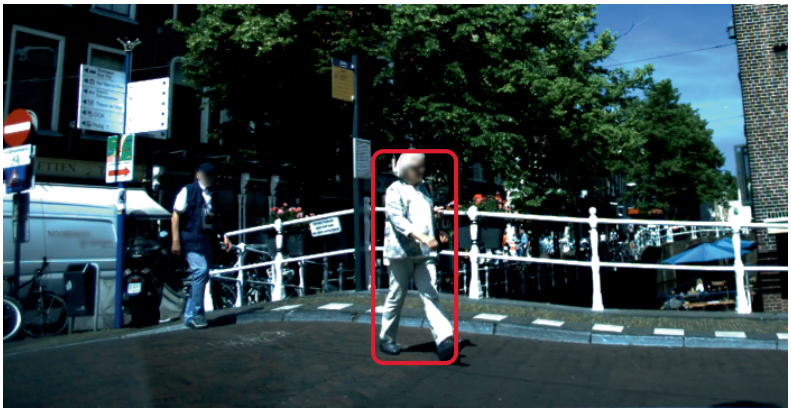
Take for example the following situation in fig. 1.1, recorded in the city of Delft in the Netherlands. As the driver approaches a T-junction to turn left onto a bridge, the driver stops in fig. 1.1a to let the couple coming from the right cross. The driver knows the couple has the right of way: both the sign of the zebra crossing as well as the zebra stripes themselves indicate this. As the couple crosses, several cyclists pass behind them in fig. 1.1b. These cyclists are looking straight forward rather than into the road that the ego-vehicle is coming from. This indicates they will likely continue straight, so the driver knows that they will no longer be in the way the moment the couple has crossed the road. However, the driver must be aware of what is happening even further away from them, as on the left side of fig. 1.1b, a woman is looking to the right, instead of in front of her. Could she be planning to cross the road? Indeed, in fig. 1.1c she has moved onto the road but has left enough space for the driver to continue onto the bridge.

In this example, one of the essential concepts that allows the driver to navigate this situation safely is *anticipation*. Instead of solely relying on what is happening now, the driver tries to predict what will happen in the near future and uses that to make a more informed decision. To ensure the planned maneuvers are safe, the driver takes the *context* of the situation into account. In other words, solely looking at the physical properties of the people around (position, velocity, etc.) is not enough. Instead, various other factors come into play to provide context to the situation. These factors are called context cues.

In fig. 1.1a, the zebra crossing is an example of a context cue that indicates the couple will continue to cross the road, as they will expect the ego-vehicle to stop. In fig. 1.1b, the context cue that the woman is looking over her shoulder shows she intends to move onto the road. At the same time, the context cue that her posture is more parallel than perpendicular to

(a)

(b)



(c)

Figure 1.1: Three different situations occurring in succession while driving in Delft. (a): Two people plan to cross the road. (green box) (b): Behind them, several cyclists continue straight (blue boxes). (c): The person behind the cyclist with a child on the back steps onto the road (red box).

the sidewalk is indicating that she will most likely move in such a way that does not require the driver to come to a full stop (fig. 1.1c).

1.1. ANALYZING THE COMPLEXITY OF EVERYDAY DRIVING

THE examples above showcase two complexities for intelligent vehicles. The first: context cues can come in many forms. Moreover, different context cues are relevant in different situations. Secondly, human drivers are adept at assessing what is relevant at what time. The descriptions above only contain the key context cues required for the reasoning of what will happen in the future and omit many details. Because human drivers can easily convey why they took certain actions, it is easier to put trust in their driving abilities. Somehow, an intelligent vehicle must be able to do the same.

From these two complexities arise the two main directions in this research. The first direction is to improve context-based path prediction for everyday driving. Here, the challenge lies in finding prediction methods and context cues that are applicable in a diverse set of situations that arise in inner cities. The second direction is to ensure the models are interpretable so it is easier to trust their results, while maximizing their prediction accuracy. The challenge in this direction lies in finding prediction methods that can easily convey why they predict the trajectory that they do, yet also have the ability to learn as much as possible from available datasets.

In this thesis, these two main directions are investigated with an additional emphasis on cyclists over pedestrians. For one, cyclists move faster than pedestrians and often share the same drivable space with the intelligent vehicle, putting them at a greater risk. Furthermore, pedestrian path prediction has already been studied to a greater extent (e.g. [7–11]). However, that is not to say that the methods described in this thesis could not be applied to pedestrians or other road users as well. Some chapters will address both cyclists and pedestrians, who are together referred to as Vulnerable Road Users (VRUs).

The purpose of improving path prediction is to facilitate safer intelligent vehicles. To ensure the compatibility of these methods for use in intelligent vehicles, all the information that the models in this thesis use must be accessible or measurable from a vehicle. For example, the position information of a cyclist must be extracted from sensors mounted on the intelligent vehicle. Furthermore, the predictions must account for the uncertainty present in the measurements as well as the uncertainty in the model itself. To that end, all methods used in this thesis predict the future path as probability distributions, rather than single points in space. To limit the scope, this thesis considers only path prediction for the near future, up to one second ahead.

1.2. THESIS OUTLINE AND CONTRIBUTIONS

CHAPTER 2 will cover the related work. Next, chapter 3 will look into 3D VRU detection, one of the more important inputs to any prediction method. After this, chapters 4 to 6 present prediction methods for cyclist path prediction. To showcase the compatibility of these methods for use in intelligent vehicles, one is implemented on such a vehicle as well, and tested on real-world sensing conditions. This is described in chapter 7, where one of the proposed models is interfaced directly with both a motion planning module as well as its required perception modules. More detailed topic outlines and contributions of chapters 3

to 7 are as follows:

EVALUATION OF LIDAR-BASED 3D PERSON LOCALIZATION.

As mentioned, before the rest of this thesis concerns itself with path prediction chapter 3 evaluates a part of the pipeline that precedes it: 3D VRU detection. For much of the past two decades, vision has been the dominant sensor modality for intelligent vehicles to detect VRUs. Strong progress has been made on 2D image-based VRU detection facilitated by novel (deep learning) methods, faster processors, and more data (including benchmarks, e.g. [12–14]). These 2D detections still need to be converted to 3D detections, though. This is usually done through disparity matching [15], which can result in noisy 3D positions.

The Lidar sensor is an attractive sensor for intelligent vehicles, and in particular for 3D VRU detection, stemming from its capabilities to directly and accurately measure distances and to deal with low-light environments. Chapter 3 therefore investigates the efficacy of two current state-of-the-art 3D object detectors based on Lidar measurements specifically for intelligent vehicles. It does so by evaluating additional metrics that are relevant to intelligent vehicle research, on top of the metrics commonly used by 3D detection benchmarks. This chapter also provides an overview of 3D object detectors together with available datasets.

The contributions of this chapter are twofold. The first is a performance analysis of two state-of-the-art methods (PointPillars [16] and AVOD [17]) on the KITTI 3D object detection benchmark [18], to determine whether 3D bounding box location, extent, or orientation influences the performance the most. Secondly, it provides results from domain transfer experiments between KITTI and the EuroCity Persons 2.5D (ECP2.5D) dataset [19].

USING ROAD TOPOLOGY TO IMPROVE CYCLIST PATH PREDICTION.

Chapter 4 investigates cyclist path prediction using a generic context cue that is relevant in a wide set of scenarios. The main location where the paths of vehicles and cyclists intersect is at intersections. Depending on the angle from which the vehicle and cyclist approach the intersection, the cyclist can be at a collision course with the approaching vehicle either by intending to turn or by intending to continue straight. Whether a cyclist is actually able to make either maneuver depends on the specific layout of the intersection. This chapter, therefore, investigates how knowledge of the road layout can improve the accuracy of cyclist path prediction at intersections. The trajectories used in the experiments are extracted from a naturalistic cyclist dataset [13] which covers a wide range of intersections and cyclist behaviors.

The contributions of this chapter are twofold. First, it supplies an extension to the Tsinghua-Daimler Cyclist (TDC) benchmark [13] in the form of cyclist trajectories, which are made available to the scientific community. Secondly, it provides a study of cyclist path prediction using probabilistic filters and a mixture model. It shows that this approach can exploit prior information on the topological road layout.

CYCLIST PATH PREDICTION USING CONTEXT-BASED SWITCHING SYSTEMS.

Chapter 5 zooms in on a specific subscenario of the previous chapter: a cyclist who is aware of the ego-vehicle driving behind them and who might turn left at an upcoming intersection. In this scenario, the dynamics of the cyclist can suddenly change from cycling straight to a left turn. Predicting whether the change might happen, as well as predicting when it will happen is facilitated by looking at various contextual cues. For one, cyclists are more likely

to turn when they raise their arm. Additionally, how far the ego-vehicle is from the cyclist in combination with how quickly the ego-vehicle is catching up influences whether the cyclist feels the need to raise an arm. If the cyclist intends to turn, however, then there is a specific area at the intersection where the cyclist is likely to do so. This chapter describes how to incorporate these three context cues into a Dynamic Bayesian Network (DBN) [20]. Because this section investigates a specific scenario, it does not use a naturalistic dataset as these do not contain large numbers of scenario examples as a consequence of their generality. Instead, it uses a dataset that is recorded especially for this scenario.

The main contribution of this chapter is the extension of the DBN from [20] onto the cyclist domain. It provides explicit equations for inference with the model in general terms and shows how these can be adapted to fit a specific scenario. Additionally, the cyclist dataset recorded for this experiment is made public for the scientific community.

CRAFTED VS. LEARNED REPRESENTATIONS IN PREDICTIVE MODELS.

The methods used in the two previous chapters, chapters 4 and 5, are both examples of models with a crafted state representation. On the other side of the spectrum are methods with learned representations such as Recurrent Neural Networks (RNNs), which have shown state-of-the-art performance in context-based path prediction [21, 22]. The downside of these data-driven approaches is they do not provide an intuitive explanation of their output: the learned state representation essentially renders them black-box models. The lack of interpretability complicates understanding why they fail when they do, which is disadvantageous for safety-critical domains such as intelligent vehicles.

Models with a crafted state representation on the other hand explicitly capture the causal relationships between context cues and future actions. However, as their crafted representations are an abstraction of the real world, they might not encode all the useful information that is available in the data. Additionally, the parameters for these methods are often not optimized, but instead individually estimated from ground truth annotations (see chapters 4 and 5) or tuned manually (e.g. [23]). Estimating parameters individually does not necessarily optimize the predictive performance of the complete model directly.

Chapter 6 therefore compares the context-based path prediction performance of a model with a learned state representation, an RNN, to that of a model with a crafted state representation, the DBN from chapter 5. The main contribution of this chapter is the evaluation of these two methods on a leveled playing field. This is made possible thanks to the other two contributions: first, the chapter describes how to integrate the context cues into an RNN (as is done for the DBN), and conversely, how to optimize the DBN with gradient descent by utilizing back-propagation (as is done for the RNN) while keeping its state representation interpretable.

INTEGRATED PATH PREDICTION FOR INTELLIGENT VEHICLES.

The intent of the methods developed in this thesis is to improve the performance of intelligent vehicles as a whole. Comparing path prediction methods to one another tells you which outperforms the others, but it does not tell you whether they would aid an intelligent vehicle in practice, either with driver assistance or with full autonomy. Chapter 7 therefore further builds on the DBN and evaluates its effectiveness when used online on an intelligent vehicle. Here, intelligent vehicles are assessed both as a driver-in-the-loop system in which the goal

is to assist the driver, as well as a fully autonomous vehicle, where it must follow a trajectory and evade obstacles without any human intervention.

As contributions, this chapter shows that the path prediction method can be readily applied in a larger pipeline in intelligent vehicles. It explains how the DBN can act as an early warning system for the cyclist scenario. Furthermore, it describes how to connect the DBN to a motion planner, which uses the predicted path and uncertainty to autonomously follow a trajectory while evading dynamic obstacles. Finally, it makes observations on the workings of the entire pipeline which would not become apparent when these components are evaluated in isolation.

2

Related Work

The more that you read, the more things you will know.
The more that you learn, the more places you'll go.

Dr. Seuss

2

Detection and tracking of Vulnerable Road Users (VRUs) have made great progress in recent years. Ohn-Bar and Trivedi [24] indicate that VRU tracking is becoming increasingly robust, and research is shifting to high-level tasks of predicting future traffic situations to inform automated decision making in Advanced Driver Assistance Systems (ADAS). As a consequence, there are now survey papers that focus solely on path prediction for VRUs [6, 25].

Path prediction methods require VRU positions as input. Prediction methods often incorporate additional semantic information, also called context cues, related to the VRU and their environment. The prediction methods themselves differ in how they model the VRU's dynamics, which in turn affects how the model parameters are estimated or optimized. The following sections will describe each of these topics (detection, prediction methods, context cues, and parameter estimation) in detail. Finally, this chapter concludes with related work on what comes after path prediction: motion planning.

2.1. Detection

Ground plane positions relative to a vehicle reference frame can be obtained from detections in various sensors, such as camera [14, 26], radar [27], or Lidar [16]. In the case of camera-based detections, the 3D location is often extracted using depth information retrieved from stereo camera images [15, 28]. However, there is a current trend towards obtaining 3D detections by incorporating Lidar (e.g. [17, 29]). A more in-depth overview of Lidar-based detection methods along with available datasets can be found in section 3.1. If ground plane positions relative to a global reference frame are needed, then vehicle ego-motion compensation is necessary as an additional pre-processing step.

To separate the performance of the prediction module from the accuracy of the VRU detector, it is common to use datasets where the VRUs are annotated, e.g. [4, 30–32]. Similarly, as ego-motion compensation will never perfectly transform the location to a global world frame, some datasets capture VRU motion from static viewpoints. However, these viewpoints are most of the time sufficiently different (e.g. a top-down view filmed with a drone [31]) from the viewpoint of a vehicle that VRU specific cues might not be as easily recognized.

2.2. Motion models

Motion models for path prediction can be categorized into physics-based, pattern-based, and planning-based [6].

In physics-based methods, motion is predicted by the forward propagation of a set of explicit dynamics equations with a physical interpretation. This category contains the single-motion model case, as in Linear Dynamical Systems (LDSs) (e.g. a plain Kalman filter) and extensions to the non-linear case (e.g. unscented or extended Kalman filters or particle filters). This category also contains more advanced approaches with multiple motion models, either as a mixture [8, 33] or with switching dynamics, e.g. Interacting Multiple Models (IMM). Context cues can guide the switch in dynamics, leading to a more general Dynamic Bayesian Network (DBN) [20, 34, 35].

Pattern-based methods instead derive predictions from previously seen data. One way of doing this is to match the current (partial) track to previously seen (complete) tracks in

a database and use the best matching exemplar for extrapolation [9]. An alternative is to perform non-linear regression by means of Gaussian Process Dynamic Models (GPDMs) [9, 36], Quantile Regression [37], or Recurrent Neural Networks (RNNs) [38–44]. Popular instantiations of RNNs are Long Short Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). The latter uses fewer parameters than the former while it may keep a similar performance [45]. An RNN can predict not only a future state but also its uncertainty (e.g. Gaussian distribution [38], or similar to an IMM filter, a mixture of Gaussians [39]). RNNs cannot inherently handle missing data (e.g. a frame where the VRU was not detected), and several methods have been proposed to overcome this (e.g. [40]). Posner and Ondrúška [40] add an extra binary input to each measurement whether the measurement has data.

Some approaches blur the line between pure physics-based and pattern-based methods. Fraccaro *et al*. [41] model the dynamic latent state of an RNN with a Kalman filter, allowing them to use the exact inference, prediction, and smoothing of a Kalman filter for the dynamics. Li *et al*. [42] propose to make separate predictions with both a DBN and RNN, and fuse these afterward in an online adaptive weighting scheme.

Planning-based methods model road users as agents that perform a sequence of decisions or actions in order to accomplish some goal. The specific behavior of an agent is guided by a reward function, which captures progress towards the goal and encodes certain agent preferences (e.g. a pedestrian might rather walk on the sidewalk than on the street). This reward function might not be known, and can be learned off-line from training data by Inverse Reinforcement Learning (IRL) [31, 46–50]. The agent's goal is typically not known either, but it can be jointly inferred online together with the agent's behavior. These possible actions of an agent can be described by a dynamics model, which can be either physics-based (e.g. [47]) or pattern-based (e.g. [48]).

2.3. CONTEXT CUES

OBJECT context cues are those that are directly linked to the object of interest, in addition to point target kinematics (positions, velocities, and orientation [51]). For example, Keller and Gavrila [9] use dense optical flow features to improve pedestrian path prediction. Xiong *et al*. [52] incorporate a learned feature representation of the VRU related cues, either through the feature representation of a re-identification network or through the last layer feature representation of the YOLO object detector [53]. Quintero *et al*. [36] recover a full 3D articulated pose of a pedestrian.

Static context cues refer to the influence of the world surrounding the VRU on their path. These are static effects such as an expectation on where VRUs plan to walk to [23], or their specific location within the scene [10, 54]. For road users, the topological and geometric layout of crossings can be a powerful cue for future behavior, especially for crossings. This is the case for both pedestrians [55–57] as well as cyclists [58, 59] A more implicit static context cue is found by identifying the VRU's preference to traverse certain kinds of semantic areas (sidewalks, grass, zebra crossings, etc.). One way of implementing this is through IRL [46, 47], or with neural networks [60]. Ballan *et al*. [31] learn preferred routes directly on top-down image data rather than on a semantic map and show that the learned knowledge is transferable to new locations. Saleh *et al*. [61] forego the need for a goal by using IRL only to learn the reward map of a static scene. Another approach is to directly encode the structure

of the road ahead [21], or to predict the trajectory along the curvature of the road [62].

Dynamic context cues include whether the VRU is aware of his or her surroundings. Kooij *et al*. [20] incorporate both whether the vehicle and the pedestrian are on a collision course as well as the pedestrian's awareness thereof into a DBN to predict the future position of a pedestrian who might cross the road. Neogi *et al*. [63] leverage the interaction between ego-vehicle and pedestrian for path prediction near an intersection as well. Other dynamic objects or VRUs can also influence the future path of VRUs. Social Force Models [43, 44, 50, 64] model the influence that nearby VRUs have on each other.

A closely related field that also uses context cues is intent recognition for VRUs, where inferring the intent is the goal, rather than the exact future trajectory. Here, many context cues have been examined as well, such as the pose [65], image data [66, 67], or physical properties: positions [57], as well as velocity and heading [68]. Intent recognition is sometimes used as an intermediate goal, where the predicted intent specifies what kind of motion model is used [9, 11].

2.4. Parameter Estimation

Methods with learned state representations optimize their parameters directly by performing gradient descent of an objective loss using training data. This has been greatly simplified thanks to frameworks such as PyTorch [69] and TensorFlow [70]. The main requirement is that this loss is differentiable. Similarly, the quantile regression forest-based approach of Völz *et al*. [37] optimizes all parameters at once. The downside is that while the learned representation fits the data, it is not necessarily possible to interpret the hidden state of the learned representation. Being able to interpret *why* such a model predicts what it does is an active field, both in path prediction [40] as well as in detection [71]. Attentive neural networks [72] improve the interpretability of a neural network by forcing the network to make predictions on only a subset of all available information, such that the "attention" of the network points to specific areas or moments in time.

Methods with a crafted state representation on the other hand often explicitly fix certain parameters a priori which ensures that the latent state is interpretable. Kooij *et al*. [20] fix the dynamic models in a DBN to a constant-velocity model as well and estimate the other parameters for the context cues by annotating all context variables at each frame. A similar approach can be found in [42]. Hashimoto *et al*. [35] use a DBN and fix its dynamic model to be a constant-velocity model while optimizing the other parameters through maximum likelihood estimation. Batkovic *et al*. [23] specifically structure their model so the few parameters can be tuned by hand. If the goal is to optimize the DBN for estimating the current state (i.e. filtering) and the DBN only has discrete hidden variables, both the optimal parameters and structure can be computed [73]. If it has both discrete and continuous hidden variables, parameter optimization can be done by Expectation-Maximization [74] or gradient descent [75, p. 169].

2.5. Planning

The sections above indicated that a large body of work has focused specifically on VRU path prediction. A similar amount of attention has gone to motion planning for intelligent vehicles, e.g. [3, 5, 28, 76–78]). One categorization here is whether the goal is ADAS, i.e.

to reduce the risk of injury in dangerous situations, or full autonomous planning, i.e. to drive similar to a human driver in a diverse set of situations. As an example of the former, Rosado *et al*. presented a pedestrian Automatic Emergency Braking (AEB) analytical model based on analyzing the pedestrian lateral behavior [76]. As an example of the latter, Ziegler *et al*. drove an autonomous vehicle along the 103 km long Bertha Benz Memorial Route, where they had to deal with VRUs along the way [3].

On the side of ADAS, evasive steering maneuvers are necessary if there is no longitudinal space to brake. In [28], the authors provide a driver-assistant design to decide whether to brake or evade the crossing pedestrian based on the information provided by the perception module. Alternatively, Köhler *et al*. focus on a scenario where there is no time to brake and propose an autonomous lane-keeping evasive maneuver that relies on the road infrastructure [78].

3

Evaluation of Lidar-based 3D Person Localization

You can observe a lot by just watching.

Yogi Berra

Parts of this chapter have been published in [79].

THIS chapter investigates the performance of 3D object detectors in the context of intelligent vehicles, as accurate detections are key for successful path prediction: if you do not detect that someone is there, you cannot predict where they will go. To that extent, this chapter investigates what these detectors can estimate accurately and what they cannot. This is done through an experimental study on the 3D localization of pedestrians and cyclists in traffic scenes, using monocular vision and Lidar data. Two 3D object detection methods are considered, PointPillars [16] and AVOD [17], which are among the top performers on the KITTI benchmark [18]. With these object detectors, this chapter investigates the effect of varying Intersection over Union (IoU) settings on detection performance and quantifies the errors in terms of 3D bounding box location, extent, and orientation.

Given that the KITTI benchmark contains relatively few 3D person instances, additional experiments are done on a large subset of the EuroCity Persons 2.5D (ECP2.5D) dataset [19]. Apart from being one order of magnitude larger than KITTI, ECP2.5D has advantages in terms of diversity (e.g. geographical coverage, time of day/season, weather conditions) and by being devoid of privacy-driven image blurring. Finally, domain transfer experiments between KITTI and ECP2.5D examine how these datasets relate to each other.

3.1. 3D OBJECT DETECTION

THIS section focuses on previous 3D object detection methods that use neural network architectures, as they are the current best performers in the various benchmarks.

One way to categorize these is by sensor modality, i.e. either a single modality or a fusion of multiple modalities. The commonly used sensors used are (monocular) camera and Lidar. However, the RGB-only methods (e.g. Shift R-CNN [80]) are generally outperformed by methods that instead use Lidar information. These Lidar-only networks map the point cloud to either a 2D or a 3D representation. Examples of 2D representations are Birds Eye View (used by *e.g.* HDNet [81]) and Range View (*e.g.* LaserNet [82]). Networks can also map the point cloud to 3D representations like Voxels (*e.g.* Voxelnet [83]), Pillars (*e.g.* PointPillars [16]), or Stixels (*e.g.* SCNet [84]).

Multi-sensor modality networks, also called fusion networks, use both camera and Lidar. Here, all the previously mentioned Lidar mappings can be used to fuse with the camera data. How they are fused exactly falls into four categories. The first category is early fusion, where the modalities are concatenated before being passed into a neural network. An example of early fusion is MVX-Net PointFusion [85] where the pointcloud is projected onto an RGB-image and then concatenated. Secondly, deep fusion networks fuse the modalities after they have already been processed by a part of the network, for example, PointFusion [86]. Here, the features from a PointNet [87] and a ResNet-50 are concatenated. With deep fusion, it is also possible to fuse the various modalities at multiple stages, as is done with AVOD [17]. Within such a deep fusion network, the performance is dependent on the feature encoder used [88]. Thirdly, late fusion takes the output of two or more independent networks and fuses the class probabilities [89]. Lastly, sequential fusion processes the sensor modalities in sequence. For example, Frustum PointNets [29] and Frustrum Convnet [90] use a 2D image detector to select frustums in a pointcloud, which are then processed separately.

Another way of categorizing previous 3D object detection methods is by the number of stages used by the network. Two-stage approaches utilize a Region Proposal Network (RPN) to generate bounding boxes that are individually evaluated (*e.g.* STD [91]). Single-state

Figure 3.1: An example of the predicted bounding boxes of PointPillars [16] and AVOD [17] on a scene from the EuroCity Persons 2.5D [19], along with the annotated ground truth.

Table 3.1: Comparison of AVOD and PointPillars.

	AVOD	PointPillars
Modality	Lidar + image	Lidar
Stages	Two-stage	Single-stage
Bounding box regression	four corners, heights, orientation	3D center point, length, width, height, orientation

Table 3.2: Overview of traffic-related 3D persons datasets. A dash denotes that the information could not be determined.

Dataset	Waymo [4]	nuScenes [93]	Argoverse [32]	Lyft [94]	KITTI [18]	ECP2.5D [19]
# Countries	1	2	1	1	1	12
# Cities	2	2	2	1	1	30
# Imgs	800K	34K	350K	55K	15K	46K
# Peds	2.8M	222K	132K	25K	9.4K	123K
# Riders	67K	24K	11K	22K	3.3K	13K
# Seasons	-	-	1	1	1	4
Weather	dry, rain	dry, rain	dry	dry	dry	dry, rain
Time of day	day, night	day, night	day, night	-	day	day, night
Unblurred	✗	✗	✗	✗	✓	✓

approaches instead evaluate predetermined bounding boxes (*e.g.* PointPainting [92]), also called anchor boxes.

Table 3.1 highlights the differences between PointPillars [16] and AVOD [17], two of the best performing Lidar and fusion networks, respectively, with code available at the time of writing. These will be used later in the experiments.

In terms of existing datasets, one of the first 3D object detection benchmarks was an extension to KITTI [18], released in 2017, which contains around 9400 pedestrians (of which half in the publicly available training set). Since then, KITTI has become the de facto standard for 3D object detection. However, because of the relatively small dataset size, performances can differ a lot on the validation and test set. More recent dataset additions to KITTI are significantly larger and more diverse, see table 3.2.

3.2. METHODOLOGY

THE goal of 3D person detectors is to detect the bounding boxes of Vulnerable Road Users (VRUs) in the scene. In KITTI, these bounding boxes have seven degrees of freedom (fig. 3.2). The 3D position is given in a coordinate system with respect to the ego-vehicle, where x is the position of the bounding box center lateral to the vehicle, z is the position longitudinal to the vehicle (i.e. depth), and y determines the altitude of the bounding

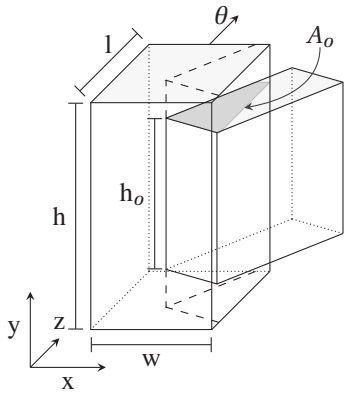


Figure 3.2: A visualization of the parameters relevant for computing the IoU of a ground truth and predicted bounding box. The darker shaded area indicates the overlap area A_o. In this figure, the overlapping height h_o is equal to the height of the smaller bounding box.

box center. The bounding box dimensions are specified by a width w, length l, and height h. Finally, each bounding box has a yaw rotation θ. The top and bottom faces of the bounding box are assumed to be parallel to the y = 0 plane. The predicted bounding boxes will also have a detection score d related to them.

3.2.1. INTERSECTION OVER UNION

To evaluate the performance of an object detector, one needs to count a predicted bounding box as valid or non-valid (i.e. true positive or false positive). In 3D (as well as 2D) object detection, the method to assess if a proposed bounding box is a true- or false-positive is based on Intersection over Union (IoU). It is defined as the intersection (or overlap) of a 3D bounding box prediction (B_p) and ground truth (B_{gt}) divided by the union of the prediction and ground truth. When both bounding boxes only have a yaw rotation, this can be written as [95]:

$$\text{IoU} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} = \frac{A_o \times h_o}{V_{gt} + V_p - A_o \times h_o} \tag{3.1}$$

Where V_p and V_{gt} are the volumes of the predicted and ground truth bounding box. The overlap of volumes can be computed from the overlapping top-view area A_o and the overlapping height (h_o), see fig. 3.2. In the KITTI benchmark, a predicted bounding box is seen as a true positive if it has an IoU of more than 0.5. Only one predicted bounding box can be marked as a true positive for any ground truth bounding box.

3.2.2. PERFORMANCE METRICS

After the true positives have been determined, it is possible to compute the two metrics as defined in the KITTI benchmark for 3D object detection: 3D Average Precision (AP_{3D}) and Average Orientation Similarity (AOS) [18].

The AP_{3D} averages the maximum attained precision s with at least a recall r for a fixed

range of recall values [96]:

$$AP_{3D} = \frac{1}{40} \sum_{r \in \{\frac{1}{40}, \frac{2}{40}, \ldots, 1\}} \max_{\tilde{r}:\tilde{r} \geq r} s(\tilde{r}) \tag{3.2}$$

As precision and recall both depend on the number of true positives, the AP_{3D} strongly depends on the IoU threshold.

Where the AP_{3D} verifies whether the bounding boxes are in the correct place, the AOS additionally verifies the correctness of their orientations:

$$AOS = \frac{1}{40} \sum_{r \in \{\frac{1}{40}, \frac{2}{40}, \ldots, 1\}} \max_{\tilde{r}:\tilde{r} \geq r} \tilde{s}(\tilde{r}) \tag{3.3}$$

$$\tilde{s}(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \frac{1 + \cos \Delta_\theta^{(i)}}{2} \delta_i \tag{3.4}$$

Where $\mathcal{D}(r)$ denotes the set of all objects at a specific recall rate r and $\Delta_\theta^{(i)}$ the difference between the estimated and the real orientation. The indicator δ_i is one if the predicted bounding box is seen as a true positive, and zero otherwise. If every true positive predicted bounding box has an orientation error of 0, eq. (3.4) reduces to the precision at that recall rate.

3.3. EXPERIMENTS

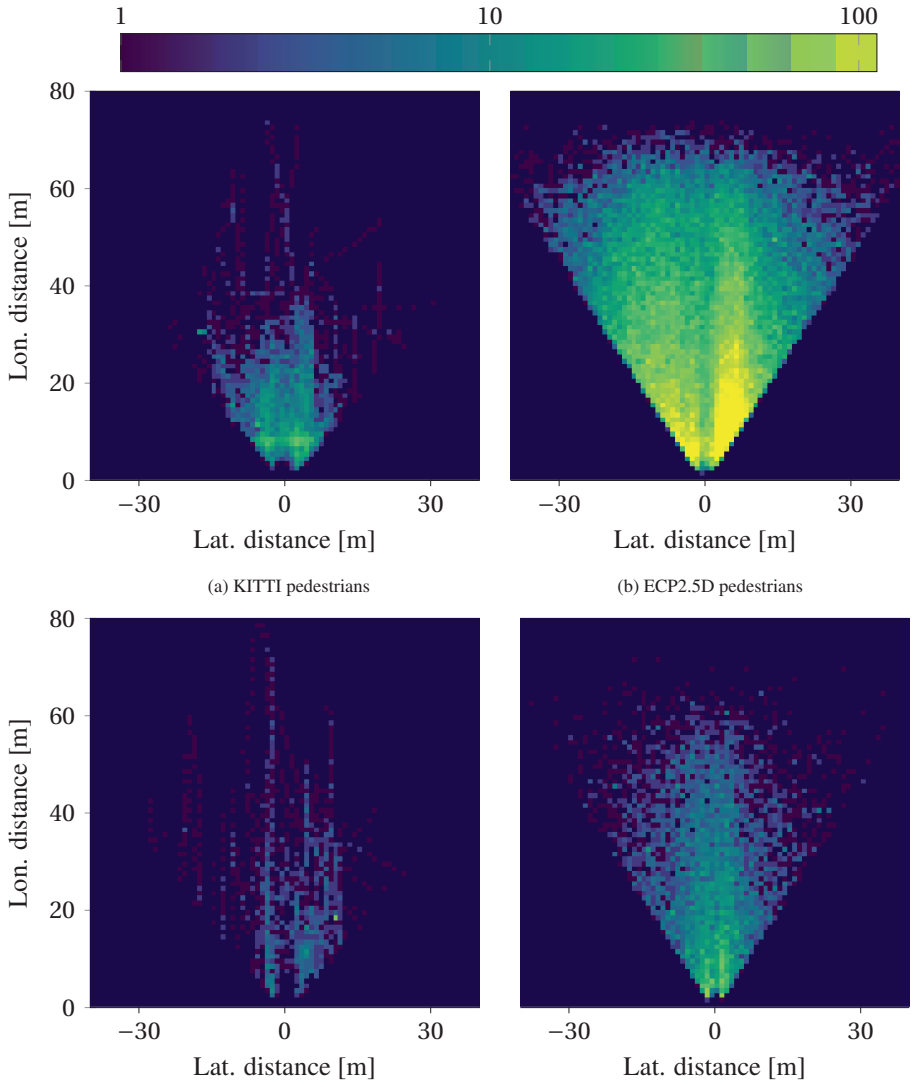
EXPERIMENTS are performed with the codebase of the authors of AVOD[1] and the codebase recommended by the authors of PointPillars[2] as is, using the best performing network as reported in their papers. Thus for AVOD, the specific version used is AVOD-FPN, and PointPillars uses a spatial resolution of 0.16×0.16 m^2.

3.3.1. DATASETS OVERVIEW

Figure 3.3 shows the distribution of the VRU locations relative to the vehicle for the publicly available part of both KITTI and ECP2.5D. The bulk of the detections in the KITTI dataset lies within 30 m distance of the ego-vehicle. Both datasets have a bias towards VRUs being on the right side of the ego-vehicle.

This chapter uses the same KITTI 1:1 train/validation split as specified by the AVOD and PointPillars codebases. The KITTI dataset contains 2.2K/0.7K and 2.3K/0.9K pedestrian/cyclist annotations for the train and validation split respectively. The validation split is divided into three parts, “easy”, “moderate”, and “hard”, as defined by KITTI. The ECP2.5D dataset has a larger amount of annotations for the 3D position and orientation but lacks width, length, and height annotation. Instead, the median bounding box dimensions of the train split of the KITTI dataset are used, so both networks can still regress a full bounding box. This chapter uses the "Day" subset of ECP2.5D as its basis. Additionally, the underlying Eurocity Persons (ECP) dataset misses an orientation label for 386 pedestrians and 144 riders, these

[1] https://github.com/kujason/avod
[2] https://github.com/traveller59/second.pytorch



(a) KITTI pedestrians

(b) ECP2.5D pedestrians

(c) KITTI cyclists

(d) ECP2.5D cyclists

Figure 3.3: The overall distribution over location of pedestrians and cyclists the KITTI dataset as a logarithmic plot. In all figures, the ego-vehicle is positioned at $(0,0)$, looking upwards. Each pixel in the image corresponds to a $1x1$ square meter area. The color indicates the number of ground truth occurences in that location according to the colorbar on top. The darkest blue region indicates areas with zero occurences.

Table 3.3: AOS and AP_{3D} performance of PointPillars (PP) and AVOD, trained on KITTI and evaluated on the moderate part of the KITTI validation split.

	Pedestrian		Cyclist	
IoU	AP_{3D}	AOS	AP_{3D}	AOS
PP				
0.5	55.8	27.0	58.5	5.8
0.4	71.5	34.5	63.7	6.9
0.3	76.5	37.1	64.9	7.1
0.2	77.1	37.4	66.0	7.2
0.1	77.2	37.5	66.0	7.2
AVOD				
0.5	41.2	32.3	35.1	34.8
0.4	50.0	38.3	36.3	35.9
0.3	52.5	40.1	36.3	35.9
0.2	52.7	40.2	36.3	35.9
0.1	52.7	40.3	36.3	35.9

are set to "Don't Care". This results in 62.3K/7.3K pedestrian/cyclist annotations in the training split, and 12.6K/1.3K pedestrian/cyclist annotations in the validation split. The test set ground truth annotations of both datasets are not made public, so all evaluations done in the rest of this chapter are done using the validation splits of either dataset as mentioned here.

Both datasets use the Velodyne HDL-64E (Lidar) sensor. The intensity of the Lidar points in KITTI fall in 100 discrete bins of between 0 and 1. ECP2.5D has an intensity on a continuous range between 1.0 and 255.

3.3.2. EFFECT OF IOU ON PERFORMANCE AND ERROR ANALYSIS

PERFORMANCE WITH LOWER IOU CONSTRAINTS

Table 3.3 shows the performance of PointPillars and AVOD on KITTI for the cyclist and the pedestrian classes. PointPillars has a higher AP_{3D} than AVOD, even though their scores on the moderate test split on the KITTI benchmark differ less than one percent. However, the results for AVOD are comparable to those found on the validation split in the comparison study of [88]. Lowering the IoU threshold increases the AP_{3D} by a large margin. For example, the AP_{3D} of PointPillars on pedestrians increases from 55.8 to 77.2 (21 %).

This is further visualized in fig. 3.4, which shows a histogram of the IoU found for all true positive detections at an IoU threshold of 0.1. This histogram shows that for pedestrians more than 15% of the detections of PointPillars and 10% of the detections of AVOD had an IoU between 0.4 and 0.5, just outside the normal IoU threshold. A similar effect is seen for cyclists, albeit less strongly.

The upper bound of the AOS is the AP_{3D}, as mentioned in section 3.2.2. Table 3.3 shows

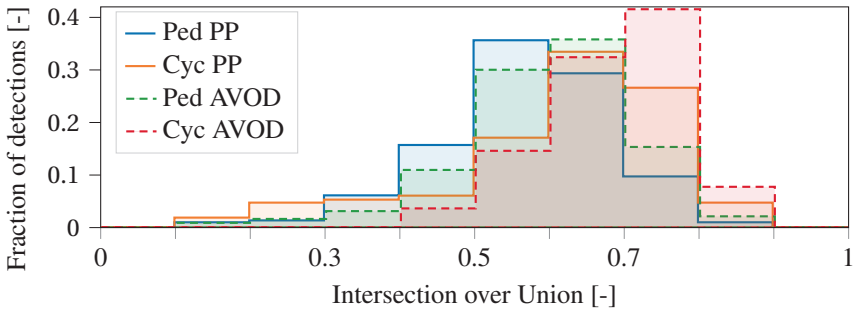


Figure 3.4: PointPillars and AVOD trained on KITTI: a histogram of what fraction of true positive detections had what IoU (IoU threshold of 0.1).

that even though the general detection accuracy of AVOD is lower than PointPillars, its AOS is almost perfect, especially for cyclists. The AOS of PointPillars is far worse than the AOS noted on the online KITTI benchmark. A closer inspection of the distribution of the orientation error (fig. 3.5) shows that for PointPillars, the orientation error peaks around 0 or 180 degrees. In the paper of PointPillars, the authors state that the orientation loss used cannot distinguish between flipped boxes, for which they use an additional binary classification loss. The orientation errors of PointPillars shown in fig. 3.5 seem to indicate that while the original overall orientation loss works as expected, there might be an implementation issue with the binary classification loss in the codebase of SECOND. As for AVOD, almost all of the orientation estimates indeed have an error closer to 0 degrees as was expected from their AOS.

ERROR ANALYSIS OF BOUNDING BOX ESTIMATION

Figure 3.6 shows the error made in position and size of the predicted bounding boxes on pedestrians by PointPillars. The smallest errors are made on the x and the z estimation: the lateral and longitudinal position. The largest error is made on the width and length estimation. These depend on the stride of a pedestrian, as well as the location of their arms, which can be difficult to infer at larger distances.

The relatively small error in x and the z position (essentially a top-down position estimate) is visualized in fig. 3.7. It shows the x and z position error made for the true positive detections for the original IoU threshold, as well as the error for the detections between an IoU of 0.1 and 0.5. A lot of the detections with an IoU below 0.5 are still accurate at estimating the position. For an IoU threshold of 0.5, nearly all of the true positive detections (1462 of the 1494) lie within a radius of 15 cm. When looking at the detections found with an IoU threshold of 0.1, a total of 1811 detections lie within a radius of 15 cm. In other words, using a radius of 15 cm as a metric to determine true positives instead of an IoU of at least 0.5 shows a 23 % increase in the number of detections.

The same data is put more succinctly in fig. 3.8, with cyclists added as well. It shows the amount of true positive detections that fall below a specific Euclidean position error. Cyclists see a smaller benefit, but as their annotated bounding boxes are larger, it is possible to make a larger position error without affecting the IoU as much.

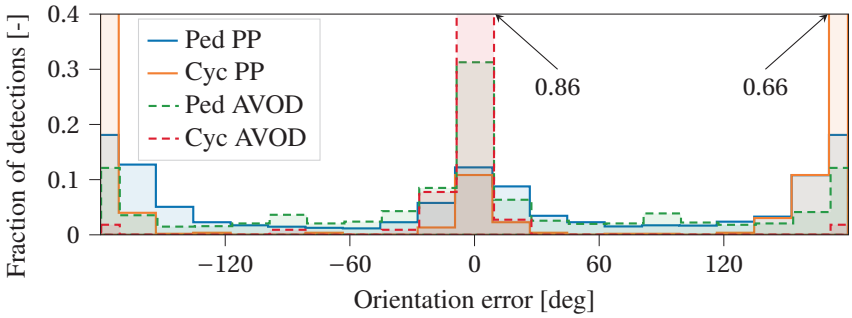


Figure 3.5: PointPillars and AVOD trained on KITTI: A histogram of the orientation error. The arrows indicate the fraction of detections of the two bars outside of the y axis range. Most orientation errors lie either between -40 and 40 degrees, or between 140 and -140 degrees.

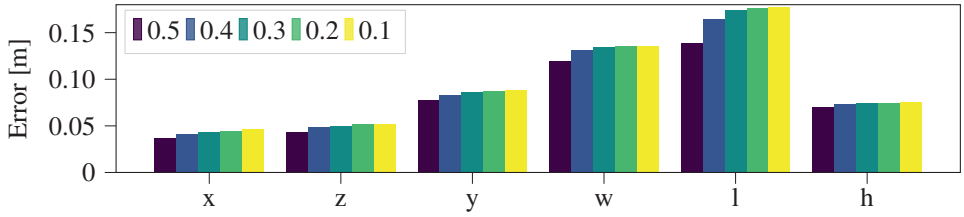


Figure 3.6: PointPillars trained on KITTI: the average error between the prediction and the ground truth for the pedestrian detections on x, z, y, w, l, and h, at different IoUs thresholds. The largest error is made on the altitude estimation, together with the bounding box width and length.

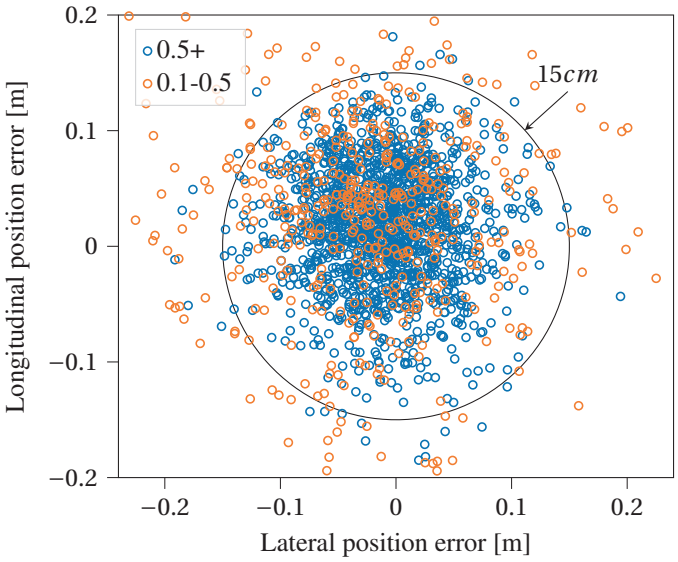


Figure 3.7: PointPillars trained on KITTI: The localization error made by true positive detections of **pedestrians**, from a bird's eye viewpoint. Of the true positive detections with an IoU of over 0.5, 1462 out of 1494 detections lie within a radius of 15 cm. Of the true positive detections with an IoU between 0.1 and 0.5, 349 out of 478 lie within that radius.

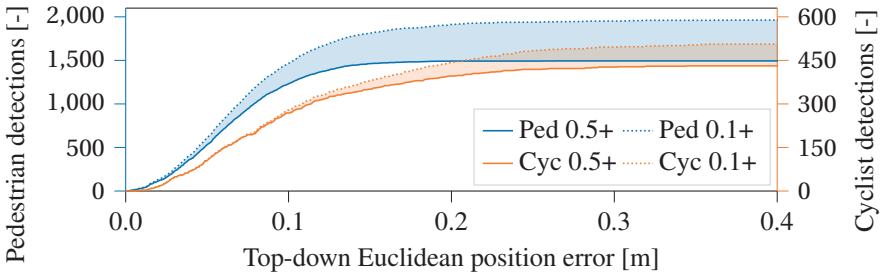


Figure 3.8: PointPillars trained on KITTI: given a certain Euclidean position error threshold, how many detections would be inside. The solid line shows what Euclidean error is made by detections using the current default IoU threshold of 0.5. The dotted line shows the number of detections within a given Euclidean error for an IoU threshold of 0.1. The shaded area then shows the number of detections added.

Table 3.4: AP_{3D} performance of PointPillars (PP) and AVOD for two IoU thresholds, evaluated on the moderate part of the KITTI validation split. The networks were trained on the original KITTI ground truth or on the ground truth with fixed bounding box dimensions.

	Pedestrian		Cyclist	
IoU	Original	Fixed	Original	Fixed
PP				
0.5	55.8	54.6	58.5	62.6
0.1	77.2	73.3	66.0	68.1
AVOD				
0.5	41.2	46.0	35.1	35.5
0.1	52.7	59.6	36.3	38.8

Table 3.5: AP_{3D} performance of PointPillars (PP) and AVOD for an IoU of 0.1 on the moderate validation split of KITTI and ECP2.5D. Bold indicates the highest performance in that column.

	AP_{3D}	
Trained network	ECP2.5D	KITTI
with intensity:		
PP on ECP2.5D	**34.1**	46.7
PP on KITTI	6.9	**77.2**
w/o intensity:		
PP on ECP2.5D	32.8	55.4
PP on KITTI	26.0	67.5
AVOD on ECP2.5D	26.8	34.0
AVOD on KITTI	5.0	52.7

ACCURACY EVALUATION USING FIXED BOUNDING BOXES DURING TRAINING

The relatively large errors in width and length suggest that these two 3D object detectors are not able to properly estimate these. To investigate the influence of the dimensions of the bounding boxes, the model is trained on a version of the KITTI dataset train split where the dimensions of each VRU have been set to the median dimensions of their respective class. The resulting network is evaluated on the original KITTI dataset validation split with the correct dimensions (see Table 3.4). At an IoU of 0.5, the performance of PointPillars on the pedestrian class drops by 1.2 %. Surprisingly, the performance of the cyclist class even increases by 3.9 %. Next to that, AVOD shows an increase in both the pedestrian and the cyclist class.

3.3.3. CROSS-DATASET EVALUATIONS

To see how well each dataset generalizes, both networks are trained on the one dataset and evaluated on the other. Because the original PointPillars uses the intensity information of the points in the point cloud as well, it is trained once *with* this intensity information present, and once *without*. AVOD does not use the intensity information, and therefore only needs to be trained once on each dataset. To ensure the datasets are compatible, the Lidar intensity values in each dataset are linearly rescaled to the same range.

Table 3.5 shows the resulting AP_{3D}. PointPillars using Lidar intensity data and the ("native") training sets, corresponding to the datasets tested, has the best performance on both datasets. When not using Lidar intensity data, PointPillars' performance slightly drops, but still clearly outperforms AVOD on both datasets, when using the native training sets. Performance of both methods was significantly lower on ECP2.5D vs. KITTI,

When non-native training sets are used, performances degrade significantly for both methods, both when moving from KITTI to ECP2.5D and vice versa. The performance degradation for PointPillars is less severe when the Lidar intensity data is not used.

3.4. DISCUSSION

THIS chapter presented an experimental study on 3D person localization in traffic scenes, on the basis of monocular vision and Lidar data. Experiments on KITTI showed that whereas headline results (AP_{3D}) results might seem low, the 3D box center localization accuracies are in fact quite high. The errors lowering AP_{3D} are mostly related to the estimates of the bounding box extents (especially, width and length). The path prediction methods discussed so far do not utilize these bounding box extents, and as such these errors are not as relevant as the 3D box center localization.

PointPillars clearly outperformed AVOD (AP_{3D} of 68% vs. 53% and 33% vs. 27%, for KITTI and ECP2.5D respectively, when not using Lidar intensity information). Performance of both methods was significantly lower on ECP2.5D vs. KITTI, this is attributed to a larger prevalence of distant persons with fewer Lidar points in ECP2.5D.

The domain transfer experiments indicated the two datasets have quite different biases, in the sense that training on one and testing to the other leads to significantly degraded performance (upwards of AP_{3D} of 6.8%). One could then expect a similar degradation in performance if the target Lidar of the intelligent vehicle is different from the one used in such a dataset. As such, further research is needed on cross-domain adaptation.

Additionally, the two detection methods discussed here estimate an orientation alongside the location, as is the standard in 3D object detection. Future work can extend the motion models used in this thesis to include this as an additional measurement, as suggested in [51], without needing an additional orientation detector pipeline.

One remaining question is how Lidar-based 3D object detectors compare to pure stereo camera-based detectors. While Lidar is expected to be more accurate, stereo camera detection pipelines are shown to be adequate for path prediction [3, 28]. Additionally, current Lidar sensors are more expensive than stereo camera setups, where the latter are already implemented in consumer vehicles. Consequently, a prediction method that generates good results with stereo camera detections will be broadly applicable and is expected to have an even higher performance when used with Lidar detections. The next chapters therefore opt for using detections from a stereo camera setup to ensure they are broadly applicable.

4

Using Road Topology to Improve Cyclist Path Prediction

Alice came to a fork in the road.
'Which road do I take?' she asked.
'Where do you want to go?' responded the Cheshire Cat.
'I don't know,' Alice answered.
'Then,' said the Cat, 'it doesn't matter.'

from Alice in Wonderland by Lewis Carroll

Parts of this chapter have been published in [97].

THIS chapter describes various methods to learn motion models for cyclist path prediction at intersections. Their efficacy is evaluated using real-world tracks obtained from a moving vehicle, extracted from the Tsinghua-Daimler Cyclist (TDC) benchmark [13]. The trajectories are spatially aligned in the vicinity of curves and crossings where important changes in dynamics can occur, through a coordinate system related to the road topology. These tracks are used to learn viewpoint invariant cyclist dynamics. This chapter evaluates standard motion models for path prediction and proposes an extension to leverage prior knowledge on the road topology to improve its predictions.

The focus lies on intersections for two reasons. First, this is the location where cyclists and (intelligent) vehicles interact the most. This makes it a high-risk scenario, calling for the importance of accurate path prediction. Secondly, cyclists can showcase various kinds of dynamic behavior at intersections, i.e. they have the option to continue along any of the exit roads available to them at that intersection. This means it is a difficult scenario, which is why specialized path prediction methods such as the one laid out in this chapter are needed.

4.1. CYCLIST TRACK DATASET

THE publicly available real-world TDC benchmark [13] is recorded with a stereo-camera setup in a moving vehicle in the Tsinghua city area. It contains annotated bounding boxes for cyclists, together with dense disparity maps of each frame, and camera parameters. Since the objective is to study predictive motion models instead of object detection, ground truth track data is extracted from the bounding boxes. Furthermore, all tracks are spatially aligned based on the road topology to ensure that all tracks have a more similar initial state, which could aid in path prediction. These tracks are made available for the community[1].

4.1.1. EXTRACTING TRACKS FROM THE TDC BENCHMARK

By taking the median disparity in each bounding box, one obtains 2D ground plane positions (lateral, longitudinal) relative to the ego-vehicle. Since the annotated bounding boxes also contain track ids, cyclist tracks can be extracted as sequences of 2D positions.

This chapter combines tracks from both the training and test set of the detection benchmark. The experiments will use Leave-One-Out cross-validation to separate training and test tracks. In the test set bounding boxes are provided at 5 fps, but in the training set at 2.5 fps. The training tracks are therefore interpolated to 5 fps to ensure constant time intervals for all tracks. All occluded bounding boxes, bounding boxes smaller than 30×30 pixels, or with a distance greater than 60 meters from the ego-vehicle, are removed.

To learn cyclist motion models, their position and velocities should be expressed in a ground plane coordinate system independent of vehicle egomotion. Unfortunately, the TDC benchmark does not provide egomotion information. The vehicle egomotion is therefore estimated once in an offline process by applying the Iterative Closest Point (ICP) algorithm (using the Point Cloud Library [98]) on the disparity maps of each pair of subsequent frames, and accumulating the resulting 3D transformations. For reference, in recording "2014-11-20_074640" the vehicle starts and finishes at the same spot (i.e. 'loop closure') after driving 1.55 km. The traveled distance found by ICP was 1.55 km, with a deviation of only 12.8 m

[1] This dataset is available for non-commercial research purposes. Follow the links from `http://www.gavrila.net` or contact the author.

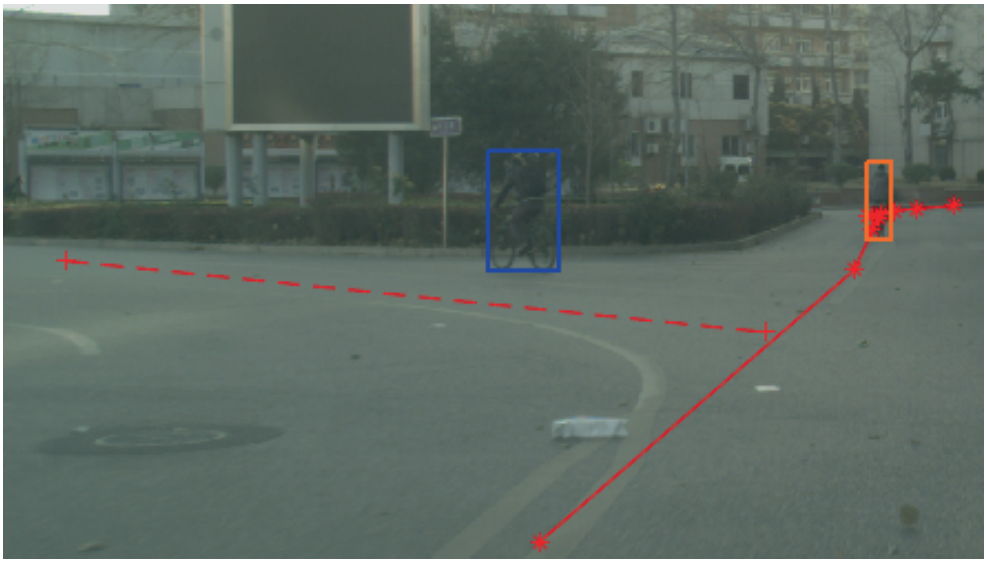


Figure 4.1: An example of two cyclists, together with the road annotation. The dotted line shows the center lane of the road that the cyclist in the blue rectangle is cycling onto, while the solid line shows the center lane of the road that both the ego-vehicle and the 'orange' cyclist follow. Every star is an annotated point on the main road, the dashed line is the sideroad annotation.

between the start and end points.

4.1.2. ALIGNING TRACKS WITH ROAD TOPOLOGY

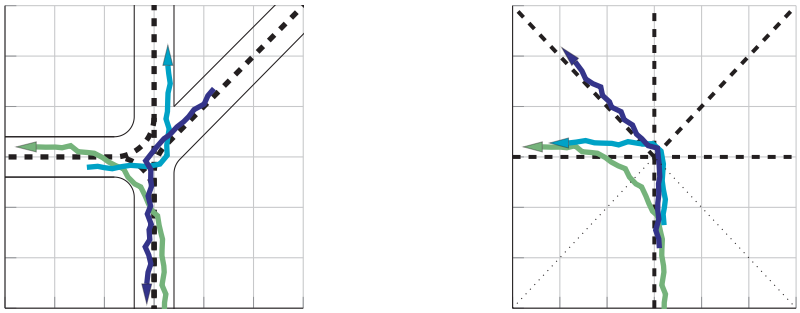
The road layout and intersection topology of the driven routes is manually annotated. This was done by marking points in the video along the centerline of the driven road throughout the sequences, as well as any side- or crossroads that a cyclist's path followed. For each crossing on a cyclist's path, a label is added to define which directions are available and which direction the cyclist takes, as five canonical direction classes: a 90° left and right bend, a 45° left and right bend, and straight. Finally, each track is labeled as to which main and sideroad segments it follows, hence it is known where each track passes a crossing. An example of the annotated scene is shown in fig. 4.1. Section 4.2 shall propose a method to exploit this topological prior knowledge for path prediction.

All locations where a track follows a bend in the road are located, either by taking a turn at an intersection, or by following a curved road. If a cyclist track had more than one turn in it, the tracks are cut into two segments, one for each turn. If a cyclist travels straight longer than 10 seconds (50 frames), it is cut into smaller segments. Track segments shorter than 1 second are discarded. For the remainder of this chapter, the term 'tracks' shall now refer to the processed track segments.

All tracks are then categorized by the direction that they take on their respective crossings. The total track count per class label is given in table 4.1. There are much more straight than bending tracks, therefore bending tracks are extracted from the full TDC benchmark, but straight tracks only from the TDC test set.

Table 4.1: The total amount of tracks extracted from the dataset. In total, there are 119 tracks, extracted from 108 cyclists.

	90° left	45° left	straight	45° right	90° right
Track count	16	8	68	10	17
Frame count	136	99	1128	135	167



(a) (b)

Figure 4.2: Three tracks, before (4.2a) and after (4.2b) they have been transformed to the general coordinate system. The general coordinate system ensures a much more similar initial state between all tracks. The frame where a track is closest to the thin dotted line is where the Time To Event (TTE) of that track is defined to be 0.

Tracks are then aligned with respect to their local road topology through translation and rotation. For all curved tracks, the translation is done based on the intersection point of the center lanes of the incoming and outgoing roads. This point is selected to be the origin point for the curved track. For straight tracks, the origin is the point at the center lane that is closest to the average of the track's start and end position. After the translation, the tracks are rotated such that the direction from the incoming road towards the intersection point is pointed directly upward when viewed in a 2D x-y graph. The process is illustrated in fig. 4.2. The resulting spatially aligned real-world tracks are shown in fig. 4.3.

For temporal alignment, existing literature [8, 20] is followed by expressing frames in Time To Event (TTE), where the frame with TTE = 0 is when the track crosses the line of equal lateral and longitudinal distance to the origin (see dotted lines in fig. 4.2b). Earlier frames have negative TTE (e.g.. cyclist approaching intersection), later frames a positive TTE (e.g. cyclist leaving intersection).

4.2. METHODOLOGY

THis chapter compares three probabilistic motion models based on linear dynamics for path predictions. As in [8], observations are filtered online using a recursive Bayesian filter with the selected motion model. At any frame, a predictive distribution for future positions is obtained by executing a filter's 'predict' step several times without any 'update'.

Below, section 4.2.1 will first introduce the considered motion models and explain how to exploit road topology. Then section 4.2.2 will explain how model parameters are estimated from the track data, section 4.2.3 will detail online path prediction. Finally, section 4.2.4

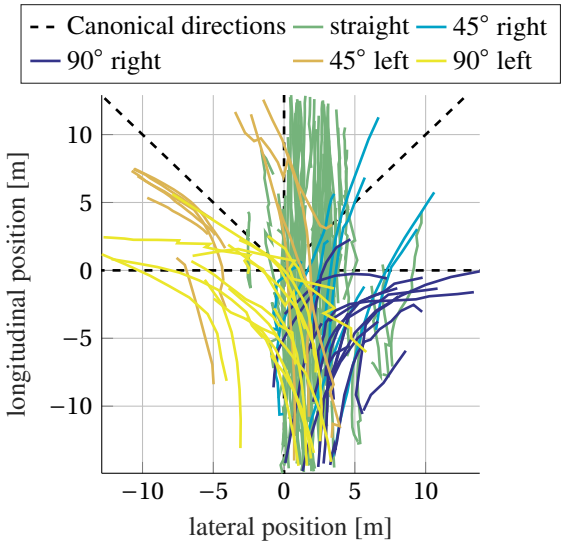


Figure 4.3: Extracted real-world cyclist tracks, aligned with their local road topology which distinguishes five canonical directions. All tracks start at the bottom and move upward. The figure shows that most (but not all) cyclists drive on the right side of the road. Note that some cyclists who plan to turn to the left are seen to cycle on the left side of the road, even before the crossing. Most tracks move straight.

defines the metrics on which the learned models are evaluated.

4.2.1. MOTION MODELS FOR PATH PREDICTION

The following three probabilistic motion models are considered:

Linear Dynamical System (LDS) Previous research on pedestrian path prediction found no significant benefit of higher-order or constant turn motion models over a constant velocity model with white noise acceleration [8]. Hence the picked baseline is the constant velocity LDS. The used recursive Bayesian filter is therefore a common Kalman filter. The predictive distribution is Gaussian.

Uninformed MoLDS (U-MoLDS) Consider that the cyclist has a latent *intent* to move into one of the five canonical directions that could occur in the road topology (see section 4.1.2). Given this intent, more specific dynamics might be applied. The baseline model is extended to a five component Mixture of Linear Dynamical Systems (MoLDS), one for each of the five canonical directions.

Since the cyclist's intent is unobserved, it must be estimated online. The prior intent is assumed to be a uniform distribution. This model is called *uninformed* with respect to the latent intent. During online inference, both a distribution on the continuous state and the latent intent is inferred from the past observations.

Informed MoLDS (I-MoLDS) The third model is similar to the U-MoLDS but includes prior information on which canonical directions are present in a track's local road topology (e.g. obtained from map data). Namely, the intent prior is set to zero for road directions that are not in a track's local topological layout, the other directions have an equal prior probabil-

ity.

More formally, at every time step t, the model is defined by two variables: the hidden state $h_t \in \mathbb{R}^{|h|}$ and the observation $y_t \in \mathbb{R}^{|y|}$. Their relations are defined by linear dynamics and Gaussian noise, namely

$$h_t = Ah_{t-1} + B\epsilon_t \qquad \epsilon_t \sim \mathcal{N}(\mu_\epsilon^M, \Sigma_\epsilon^M) \tag{4.1}$$

$$y_t = Ch_t + \eta_t \qquad \eta_t \sim \mathcal{N}(0, \Sigma_\eta^M). \tag{4.2}$$

Here the vector $\epsilon_t \in \mathbb{R}^{|h|}$ is an unknown noise signal affecting the system and matrices $A \in \mathbb{R}^{|h|\times|h|}$ and $B \in \mathbb{R}^{|h|\times|h|}$ define the linear state transition. The measurements $y \in \mathbb{R}^{|y|}$ are related to the state through matrix $C \in \mathbb{R}^{|y|\times|h|}$. The variables ϵ and η are Gaussian noise, and assumed to be dependent on the intent $M \in [1, \cdots, |M|]$ of the cyclist. Here, $\mu_{(\cdot)}^M$ and $\Sigma_{(\cdot)}^M$ show that the process mean and covariance of the noise is dependent on M. In the remainder of this chapter, the LDS baseline is considered a special case with only one possible intent, i.e. $|M| = 1$, while $|M| = 5$ for the U-MoLDS and I-MoLDS.

The observations are positions in 2D, and the model is a constant velocity model. The noise that acts on the system is modeled as acceleration as given in eq. (4.3):

$$B = \begin{bmatrix} \frac{1}{2}\Delta T^2 & 0 & \Delta T & 0 \\ 0 & \frac{1}{2}\Delta T^2 & 0 & \Delta T \end{bmatrix}^\top. \tag{4.3}$$

Here, ΔT is the time difference between consecutive frames.

4.2.2. OFFLINE PARAMETER LEARNING

For each motion model, its parameters consist of the initial state distribution $(\mu_{h_1}^M, \Sigma_{h_1}^M)$, process noise parameters $(\mu_\epsilon^M, \Sigma_\epsilon^M)$, and observation noise Σ_η^M for each intent M. During training, the model parameters must be determined from the training data, but each track's intent M is set to its class label (i.e. the canonical direction the cyclist actually takes).

Due to the difficulty of obtaining large amounts of track data, rare motion patterns could only have a few examples, and maximum likelihood parameter estimation could overfit the more complex models with more parameters. Therefore, this chapter follows the approach in [10] and uses fully Bayesian approximate inference to integrate out the model parameters in the experiments. More precisely, conjugate priors distributions are placed on the parameters for regularization, namely Normal-Inverse-Wishart (NIW) distributions on $(\mu_{h_1}^M, \Sigma_{h_1}^M)$ and on $(\mu_\epsilon^M, \Sigma_\epsilon^M)$, and Inverse-Wishart (IW) on Σ_η^M. Given the training data, Gibbs sampling is used to sample several probable parameter combinations from their joint posterior. The sampling procedure is explained in appendix 4.A. The same priors will be used for all motion models.

4.2.3. ONLINE PATH PREDICTION

For a given set of sampled model parameters, online path prediction can proceed by running 'predict' steps, as outlined at the start of this section. Conditioned on the intent, all models reduce to a Kalman filter for which prediction is straightforward. However, since a track's

intent is unknown during test time, the on-line method must consider the posterior distribution on M given all past observations.

At every time step t, the posterior state distribution $p\left(h_t|y_{1:t}, M\right)$ is computed separately for each LDS M, using $|M|$ separate Kalman filters. The posterior on M can be computed from the past observations and the prior distribution $p\left(M\right)$,

$$p\left(M|y_{1:t}\right) \propto p\left(y_{1:t}|M\right) p\left(M\right). \tag{4.4}$$

Recall that the U-MoLDS assumes a uniform prior $p\left(M\right)$ over all 5 intents and I-MoLDS over only the possible intents. The LDS has only one intent, which always has probability 1.

The posterior state distribution for time t is a mixture of $|M|$ Gaussians, $p\left(h_t|y_{1:t}\right) = \sum_{M=1}^{|M|} p\left(h_t|y_{1:t}, M\right) p\left(M|y_{1:t}\right)$. To get a prediction n time steps in the future at time t, the Kalman prediction step is applied n times to all $|M|$ filters, each resulting in a predictive distribution $p\left(h_{t+n}|y_{1:t}, M\right)$. The complete predictive distribution for future time step $t+n$ is thus again a mixture of $|M|$ Gaussians with weights $p\left(M|y_{1:t}\right)$:

$$p\left(h_{t+n}|y_{1:t}\right) = \sum_{M=1}^{|M|} p\left(h_{t+n}|y_{1:t}, M\right) p\left(M|y_{1:t}\right). \tag{4.5}$$

Applying the observation model once more to each filter results in the predictive distribution $p\left(y_{t+n}|y_{1:t}\right)$ for the future observation y_{t+n}.

4.2.4. EVALUATION

The results on the dataset of this chapter are evaluated with two metrics, each based on the distribution on the future position given the measurements up to time t: $p\left(y_{t+n}|y_{1:t}\right)$. For both metrics, the prediction n is set to five time steps into the future, which equals one second. The first metric is the Euclidean distance error between the expected future observations $\mathbb{E}_{y_{t+n}}\left[p\left(y_{t+n}|y_{1:t}\right)\right]$ and the actual observations. Let $\hat{y}_{t+n}$ be the true future observation of the position n time steps ahead of the current time t. For a particular model, a track's Euclidean distance at a certain time step is then

$$error\left(t+n|t\right) = \left\|\mathbb{E}_{y_{t+n}}\left[p\left(y_{t+n}|y_{1:t}\right)\right] - \hat{y}_{t+n}\right\|. \tag{4.6}$$

For this measure, a lower score indicates a better performance. When the Euclidean error of entire tracks is given, it shows the average Euclidean error of every time step.

The second evaluation metric is the log-likelihood of the predictions, which is a unitless measure but is indicative of both accuracy and certainty. The log-likelihood considers the probability of the actual observation at time step $t+n$. For each track i, the log-likelihood is defined as

$$ll\left(t+n|t\right) = \log\left(p\left(y_{t+n} = \hat{y}_{t+n}|y_{1:t}\right)\right). \tag{4.7}$$

For this measure, a higher score indicates a better performance. When the log-likelihood of entire tracks is given, it shows the summed log-likelihood of every time step. The two measures are also shown how they evolve over time, based on their TTE which was explained in section 4.1.2.

To better assess how the model improves path prediction, the underlying assumption that cyclist dynamics are distinct for different intents is also tested. The U-MoLDS and I-MoLDS have a separate intent for the five given directions because it is expected that these five directions have distinct dynamics. This assumption will be tested as a classification problem: the most likely intent (i.e. the intent whose motion model has the highest likelihood over a track's observations) is compared to the track's class label. If the assumption of linear dynamics is reasonable, and the dynamics are distinct, then the classification results should be good.

4.3. EXPERIMENTS

THE models explained in section 4.2.2 are trained and evaluated using Leave-One-Out cross-validation with the measures from section 4.2.4. For each model, and each leave-one-out iteration, the Gibbs sampler was run for 300 iterations, and every tenth parameters sample of the last 100 was selected. All predictions are 1 second in the future, $n = 5$, and performance measures are computed for all sampled parameters. The performance results of the sampled parameters are then averaged at each time step.

The same priors are used for all models. The hyperparameters of these prior distributions (see appendix 4.A) are as follows:

$$\kappa^{M,h_1} = 1 \qquad \mu^{M,h_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^\top \tag{4.8}$$

$$\nu^{M,h_1} = 4 \qquad \Psi^{M,h_1} = \nu^{M,h_1} \mathrm{diag}\left(\begin{bmatrix} 8 & 8 & 0.2 & 0.2 \end{bmatrix}\right) \tag{4.9}$$

$$\kappa^{M,\epsilon} = 1 \qquad \mu^{M,\epsilon} = \begin{bmatrix} 0 & 0 \end{bmatrix}^\top \tag{4.10}$$

$$\nu^{M,\epsilon} = 4 \qquad \Psi^{M,\epsilon} = \nu^{M,\epsilon} \mathrm{diag}\left(\begin{bmatrix} 0.01 & 0.01 \end{bmatrix}\right) \tag{4.11}$$

$$\nu^{M,\eta} = 5 \times 10^5 \qquad \Psi^{M,\eta} = \nu^{M,\eta} \mathrm{diag}\left(\begin{bmatrix} 0.2 & 0.4 \end{bmatrix}\right) \tag{4.12}$$

Here 'diag' is a shorthand for a diagonal matrix with the given entries on the diagonal. The parameters $\Psi^{(\cdot)}$ for both the IW and NIW are given as a matrix, m, multiplied by $\nu^{(\cdot)}$. Interpret these priors as if v samples are a-priori known, and their expected covariance is m.

The values for ν in eqs. (4.9), (4.11), and (4.12) mean that there is a weak prior on initial state distribution ξ^{M,h_1} and the system noise $\xi^{M,\epsilon}$, while there is a strong prior for the observation noise $\xi^{M,\eta}$. This encodes the notion that different models should have similar observation noise, though their dynamics may be distinct.

4.3.1. MODEL EVALUATION

The assumption is that the dynamics of each intent are distinct. This is assessed by evaluating the classification performance given all observations. The U-MoLDS, which does not take the road topology into account, classifies 82% of all tracks correctly. To compare this with the I-MoLDS, one should consider that a part of the tracks in the available dataset has only one destination in their road topology, and as such the I-MoLDS cannot fail on these tracks. To make a fair comparison between the classification of the I-MoLDS and U-MoLDS, only the tracks with multiple destinations are considered in table 4.2. On these tracks, the U-MoLDS classifies 76% correctly which shows that it is reasonable to assume the dynamics are distinctive for their respective intent. However, the I-MoLDS classifies 90% correctly, which means the model can still benefit from additional prior knowledge.

Table 4.2: The confusion matrix for all tracks with multiple destinations. The value on the left/right shows the result for the U-MoLDS/I-MoLDS, respectively. The **bold** values highlight the best scoring model. Overall, the U-MoLDS classifies 76% correctly, whereas the I-MoLDS classifies 90% correctly.

		Estimate				
		90° left	45° left	straight	45° right	90° right
Ground truth	90° left	14 / **17**	3 / **0**	0 / 0	0 / 0	0 / 0
	45° left	0 / 0	2 / 2	0 / 0	0 / 0	0 / 0
	straight	**1** / 2	2 / **1**	13 / 13	1 / **0**	**0** / 1
	45° right	1 / 1	0 / 0	1 / 1	3 / **6**	3 / **0**
	90° right	**0** / 1	0 / 0	0 / 0	3 / **0**	13 / **15**

Table 4.3: The average Euclidean distance error in meters over all tracks, grouped by true class label. The best performance is shown in **bold**.

	90° left	45° left	straight	45° right	90° right
LDS	1.75	1.15	**1.19**	1.23	2.36
U-MoLDS	1.59	1.11	1.38	1.16	1.99
I-MoLDS	**1.51**	**1.10**	1.20	**1.08**	**1.88**

4.3.2. PATH PREDICTION

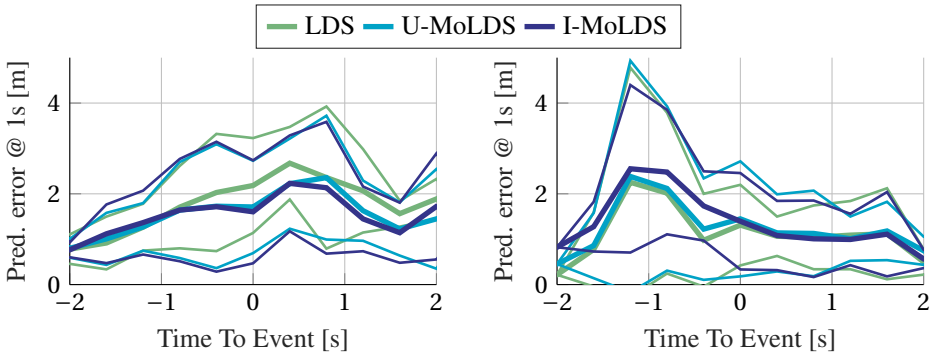
Path prediction is evaluated on the two metrics explained in section 4.2.4, with the results shown in tables 4.3 and 4.4. The I-MoLDS has the lowest Euclidean distance error for all intents except straight. This is most evident for the 90-degree angles, where the average error decreases by 24 cm (14%) and 48 cm (20%) for left and right, respectively. On straight tracks, the LDS outperforms the I-MoLDS, although only minimally (1%).

On the log-likelihood, the I-MoLDS performs best on all directions, except for 45-degree turns to the left. Here, the U-MoLDS performs best. Furthermore, when the I-MoLDS performs best, it outperforms the LDS by a large margin, whereas the difference in performance is not so large for the 45-degree turn to the left. A closer inspection of the likelihoods also shows that where the log-likelihood for 90-degrees left and right are roughly the same, there is a large discrepancy between the log-likelihood of 45-degrees left and right.

A more complete picture is painted by plotting the error over time. Figure 4.4a shows the error over time for tracks bending at a 90-degree angle to the left. At $TTE = -1$ s, where the models are predicting the state for $TTE = 0$ s, the performance diverges. This shows that the I-MoLDS can predict the change in dynamics that is related to the 90-degree turns. Consequently, the I-MoLDS improves at a time where it matters the most. The same result

Table 4.4: The mean log-likelihood for all tracks, grouped by true class label. The best performance is shown in **bold**.

	90° left	45° left	straight	45° right	90° right
LDS	-26.66	-27.27	-29.35	-21.91	-24.09
U-MoLDS	-21.72	**-26.57**	-26.24	-21.93	-19.99
I-MoLDS	**-20.62**	-28.05	**-23.65**	**-20.78**	**-19.73**



(a) Prediction error over time on 90° left turn.

(b) Prediction error over time on 45° left turn.

Figure 4.4: The mean error (thick line) and standard deviation(thin line) over time for all tracks, with respect to the moment they were predicted. The tracks turning 90 degrees to the left are shown in fig. 4.4a. The tracks turning 45 degrees to the left that show an anomaly (see text) are shown in fig. 4.4b.

was seen for the 90-degree turns to the right. The results for all classes can be found in appendix 4.B.

The same cannot be said for the 45-degree left turn, however, as is shown in fig. 4.4b. For this angle, the LDS and U-MoLDS show more accurate predictions. A large spike in the standard deviation can also be seen around this time, indicating that the large error is not present for all tracks. Together, this indicates that there are tracks present in the 45 degrees left class whose dynamics are not represented by the others in their group during training. This is further illustrated by the anomaly seen in the log-likelihood in table 4.4, where, even though the I-MoLDS did not perform well on the 45-degree left turn, neither did the others. This suggests that more data is needed for this class.

4.4. DISCUSSION

THIS chapter presented a complementary dataset to enrich the TDC Benchmark. On this dataset, a Mixture of Linear Dynamical Systems that can take the road topology into account was trained and evaluated. The Euclidean prediction error of the cyclist position one second into the future is shown to be comparable to a Linear Dynamical System on straight tracks, and improve when the cyclist is turning left or right. Because this dataset adds to an existing dataset, future research can consider incorporating the visual features from the video frames to further improve prediction.

Because the trajectories were extracted from a naturalistic dataset, are representative of what a real intelligent vehicle would encounter. However, to ensure the evaluation in this chapter showed the performance of the methods rather than the surrounding software infrastructure, three assumptions have been made, and the first two warrant further investigation.

The first assumption is each cyclist is perfectly tracked. Specifically, each detection is assigned to the correct trajectory, rather than sometimes being either incorrectly assigned to a different existing trajectory, or incorrectly assigned to a new trajectory. Allowing such misassignments would harm prediction performance, but they would not shed additional light

on which prediction methods are preferred to others. Still, an in-depth analysis of what the effects of realistic tracker performance are on path prediction and the motion planning afterward would give important insight into the applicability of intelligent vehicles in real-world situations.

The second assumption concerns the bounding box detections. These are currently based on human annotations, rather than the output of an object detector. Using an object detector would increase the measurement noise, especially as the 2D bounding boxes need to be converted to 3D using disparity: an offset in the 2D bounding box causes the wrong disparity values being queried, which will result in an offset in the final 3D position. As mentioned in chapter 3, this can also be done with 3D object detectors based on Lidar. While these require an additional sensor, they remove the need for stereo processing and 2D-to-3D conversion and have a high 3D localization accuracy.

The third and last assumption is that the road topology is assumed known. This is a valid assumption, as road layout databases do exist (some datasets include these as well [32]). Still, the road topology – that is the available directions at an intersection – can sometimes be different for a cyclist and a vehicle: some alleyways might be too narrow for the intelligent vehicle, but accessible for the cyclist. Additionally, the traffic rules might disallow entrance to a road (e.g. a one-way street), but grant an exception to cyclists. Worse yet, cyclists are prone to disregard traffic rules altogether, resulting in a mismatch between what the map information states is possible and what is actually possible. Such effects can be integrated into the I-MoLDS by for example making the intent priors depend on information sources such as signage. However, as these situations do not arise very often, future work will have to evaluate the efficacy of such approaches with datasets much larger than the one used here.

Further motivated by the fact that interesting critical situations do not occur that often naturally, the next chapter will move away from naturalistic data and instead focus on a single scenario. This also makes it possible to exploit a larger amount of scenario-specific context cues and investigate what type of prediction method is most suited to easily allocate for this larger amount of context cues.

APPENDIX

4.A. GIBBS SAMPLING

THE distributions over the unknown initial state, $\mu_{h_1}^M$ and $\Sigma_{h_1}^M$, and each type of dynamics, μ_ϵ^M and Σ_ϵ^M, have a prior NIW distribution, eqs. (4.13) and (4.14). The observation noise Σ_η^M has an IW distribution as prior, eq. (4.15).

$$\left\{\mu_{h_1}^M, \Sigma_{h_1}^M\right\} \sim \mathcal{NW}^{-1}(\xi^{M,h_1}) \tag{4.13}$$

$$\left\{\mu_\epsilon^M, \Sigma_\epsilon^M\right\} \sim \mathcal{NW}^{-1}(\xi^{M,\epsilon}) \tag{4.14}$$

$$\Sigma_\eta^z \sim \mathcal{W}^{-1}(\xi^{M,\eta}) \tag{4.15}$$

The NIW and IW distributions are parametrized by $\xi^{(\cdot)} = \left\{\mu^{(\cdot)}, \kappa^{(\cdot)}, \Psi^{(\cdot)}, \nu^{(\cdot)}\right\}$ and $\xi^{(\cdot)} = \left\{\Psi^{(\cdot)}, \nu^{(\cdot)}\right\}$ respectively. The advantage of these two distributions is that when they are updated with new measurements, their posterior is the same type of distribution. So, from the prior $\xi_-^{(\cdot)}$, one can compute the posterior $\xi_+^{(\cdot)}$ after taking N more samples $(q_1, \ldots, q_N)$ from the normal distribution as given in eqs. (4.16) to (4.18). Here, $\bar{q}$ is the mean of all samples, and S is the scatter matrix created from all samples.

$$\mu_+^{(\cdot)} = \frac{\kappa_-^{(\cdot)}\mu_-^{(\cdot)} + N\bar{q}}{\kappa_-^{(\cdot)} + N}, \quad \kappa_+^{(\cdot)} = \kappa_-^{(\cdot)} + N \tag{4.16}$$

$$\Psi_+^{(\cdot)} = \Psi_-^{(\cdot)} + S + \frac{\kappa_-^{(\cdot)} N}{\kappa_-^{(\cdot)} + N}\left(\bar{q} - \mu_-^{(\cdot)}\right)\left(\bar{q} - \mu_-^{(\cdot)}\right)^\top \tag{4.17}$$

$$\nu_+^{(\cdot)} = \nu_-^{(\cdot)} + N \tag{4.18}$$

The intuitive explanation of the parameters is that $\Psi^{(\cdot)}$ is the sampled scatter matrix, taken from $\nu^{(\cdot)}$ samples. Similarly, $\mu^{(\cdot)}$ is the sampled mean, taken from $\kappa^{(\cdot)}$ samples. For the IW distribution, the same equations apply but it could be said that the prior taken samples $\kappa^{(\cdot)}$ of the mean $\mu^{(\cdot)}$ is infinite, thereby ensuring that additional measurements will not change the mean $\mu^{(\cdot)}$. In this chapter, an IW or NIW distribution that is updated with additional measurements will be written as a function of both its initial parameters $\xi^{(\cdot)}$ and the measurements $\eta_{(\cdot)}$, e.g. $\mathcal{W}^{-1}(\xi_0^{M,\eta}, \eta_t)$ is the prior IW distribution of the observation noise covariance matrix Σ_η^M that is parametrized by $\xi_0^{M,\eta}$, updated with an additional sample of the observation noise, η_t.

To reiterate, samples from the observation noise η, the system noise ϵ, and initial state distribution h_1 can be used to improve the distribution over their covariance and, for the system noise and initial state, their mean. However, it is not possible to sample from these

directly because the true state is not known. Here Gibbs sampling is used to generate the posterior distributions, as is also done in [10]. An overview is given in algorithm 1, which is applied for each type of dynamics M, with its own example tracks.

Algorithm 1 The sampling algorithm.

Require: $\xi_0^{M,\epsilon}$, ξ_0^{M,h_1} and $\xi_0^{M,\eta}$
  Sample initial covariances and means.
  $\Sigma_\epsilon^M, \mu_\epsilon^M \leftarrow \mathcal{NW}^{-1}(\xi_0^{M,\epsilon})$
  $\Sigma_{h_1}^M \mu_{h_1}^M \leftarrow \mathcal{NW}^{-1}(\xi_0^{M,h_1})$
  $\Sigma_\eta^M \leftarrow \mathcal{W}^{-1}(\xi_0^{M,\eta})$
  repeat
    for Each track i **do** $p(h_{1:T})\, y_{1:T}$.
      Sample state from the posterior
      $h_{1:T}^i \leftarrow p(h_{1:T})\, y_{1:T}$
      From eq. (4.19)
      $\epsilon_{1:T-1}^i \leftarrow B^+(h_{2:T} - Ah_{1:T-1})$
      $\eta_{1:T}^i \leftarrow y_{1:T} - Ch_{1:T}$
    end for
    Update the inverse Wishart distributions using their initial distributions.
    $\mathcal{NW}^{-1}(\xi_+^{M,\epsilon}) \leftarrow \mathcal{NW}^{-1}(\xi_0^{M,\epsilon}, \epsilon_{1:T-1}^{1:N_{tracks}})$
    $\mathcal{NW}^{-1}(\xi_+^{M,h_1}) \leftarrow \mathcal{NW}^{-1}(\xi_0^{M,h_1}, h_1^{1:N_{tracks}})$
    $\mathcal{W}^{-1}(\xi_+^{M,\eta}) \leftarrow \mathcal{W}^{-1}(\xi_0^{M,\eta}, \eta_{1:T}^{1:N_{tracks}})$
    Resample the covariances and means
    $\Sigma_\epsilon^M, \mu_\epsilon^M \leftarrow \mathcal{NW}^{-1}(\xi_+^{M,\epsilon})$
    $\Sigma_{h_1}^M \mu_{h_1}^M \leftarrow \mathcal{NW}^{-1}(\xi_+^{M,h_1})$
    $\Sigma_\eta^M \leftarrow \mathcal{W}^{-1}(\xi_+^{M,\eta})$
  until Satisfied

Initially, there is some prior knowledge over the IW and NIW distributions, given by $\xi_0^{M,\epsilon}$, ξ_0^{M,h_1} and $\xi_0^{M,\eta}$. A random sample from the prior distributions is an initial estimate for the entire model. The initial estimate, together with the observations $y_{1:T}$ from each existing track, can give a posterior distribution on the state of the system through Kalman smoothing: $p(h_{1:T})\, y_{1:T}$ is known. If the exact state is known at every time step, it is possible to recover the system noise $\epsilon_{1:T}$ and observation noise $\eta_{1:T}$ by eq. (4.19), a direct result from eqs. (4.1) and (4.2).

$$\epsilon_t = B^+(h_{t+1} - Ah_t), \qquad \eta_t = y_t - Ch_t \tag{4.19}$$

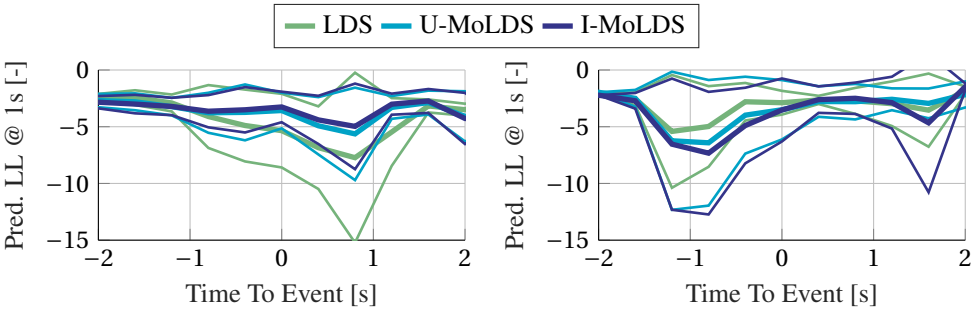
However, as stated, at each time t only the distribution of the state is known, and not the actual state. To circumvent this, Gibbs sampling is used again: sample a random potential state sequence from each track. The sampled state sequence uniquely defines a sampled system noise sequence and an observation noise sequence, which is used to update the distri-

bution of the system noise and observation noise, respectively. Similarly, the initial sample of each sequence is used to update the distribution of the initial state.

A new sample is taken from each distribution to select a system, observation, and initial state covariance as well as a system and initial state mean, and the algorithm is repeated.

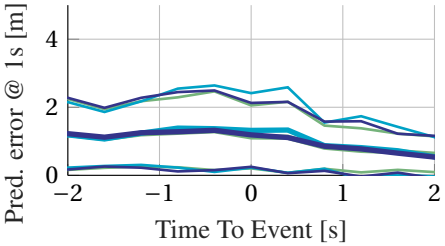
The algorithm is expected to create a random sample of the distribution of the unknown means and covariances, after the algorithm has gone through a certain "burn-in" period. After the burn-in period, a more robust set of means and covariances can be retrieved by averaging the results of multiple iterations.

4.B. ERROR OVER TIME FOR ALL CLASSES

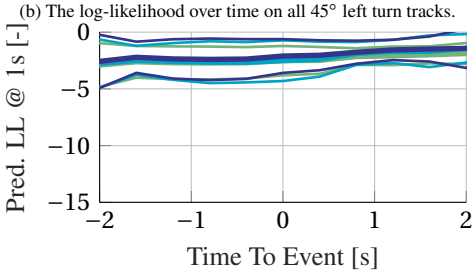


(a) The log-likelihood over time on all 90° left turn tracks.

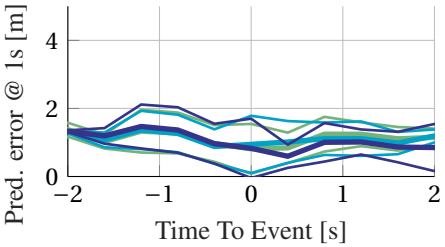
(b) The log-likelihood over time on all 45° left turn tracks.



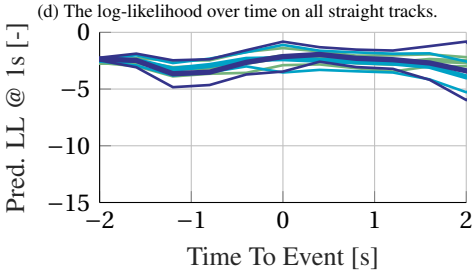
(c) The mean error over time on all straight tracks.



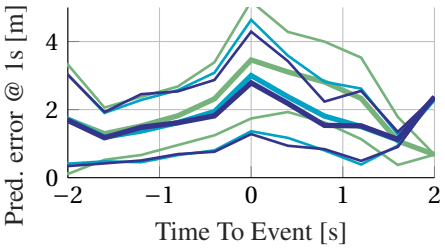
(d) The log-likelihood over time on all straight tracks.



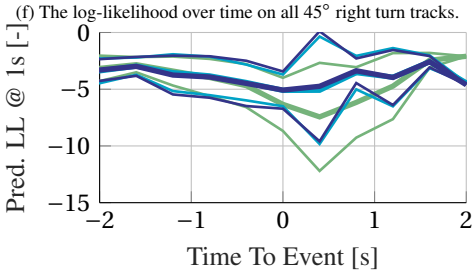
(e) The mean error over time on all 45° right turn tracks.



(f) The log-likelihood over time on all 45° right turn tracks.



(g) The mean error over time on all 90° right turn tracks.



(h) The log-likelihood over time on all 90° right turn tracks.

5

Cyclist Path Prediction Using Context-Based Switching Systems

To know an object is to lead to it through a context which the world provides.

William James

Parts of this chapter have been published in [20] and [99]

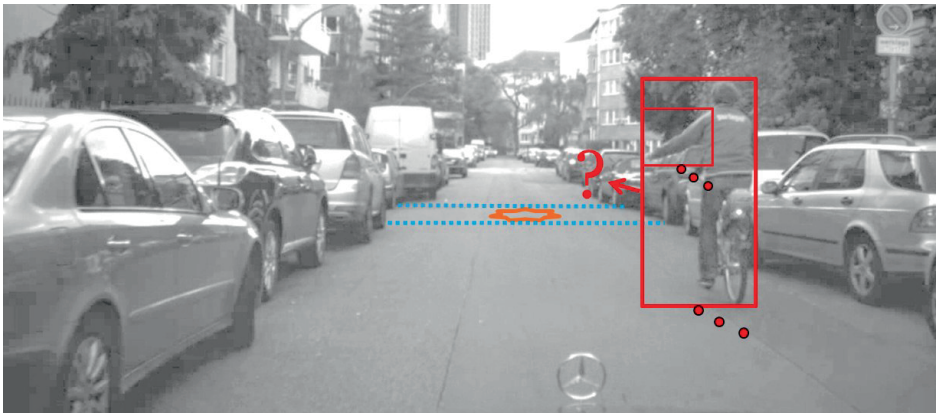


Figure 5.1: Path prediction of a cyclist with switching dynamics. The cyclist approaching an intersection can cycle straight or turn left. Context includes the vehicle trajectory, the cyclist's expressed intent by raising an arm, and distance to the intersection.

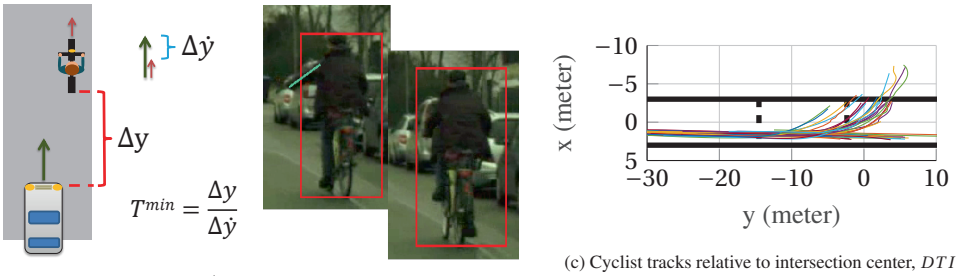
THIS chapter proposes a method for cyclist path prediction that exploits multiple context cues that are extracted from vision data. The contextual cues and dynamics are modeled using the Dynamic Bayesian Network (DBN) from [20], where the latent discrete states control the switching probabilities between the dynamic modes. Existing theory for approximate posterior inference in DBNs [75] allows for efficient computation of predictive distributions on the future state of the target.

The approach is evaluated on the scenario of a cyclist cycling in front of the ego-vehicle, who both approach an intersection where the cyclist may turn left, visualized in fig. 5.1. This scenario has three predictive cues, namely the cyclist raising an arm to indicate the intent to turn at the crossing, the cyclist's proximity to the crossing, and the existence of an approaching vehicle. The approach is general though and can be extended with additional motion types (e.g. cyclist turning right), or to other application domains, such as robot navigation in human-inhabited environments. This method also does not prohibit the use of other sensors or other computer vision methods than the ones considered here.

5.1. DATASET

THE experiments in this chapter use a dataset of cyclist encounters recorded from a moving vehicle, from [20]. Due to the focus on potentially critical situations, both driver and cyclist were instructed during recording sessions. A sufficient safety distance between the vehicle and VRU was applied in all scenarios recorded. In the following sections, 'critical situation' thus refers to a theoretic outcome where the approaching vehicle does not stop when the cyclist turns left, which would result in a collision.

This dataset contains 51 tracks of a cyclist approaching an intersection at a steady pace. These are recorded with a stereo-camera setup at 16 fps from a moving vehicle that drives behind the cyclist, resulting in 5744 frames total. There are no other traffic participants nearby. The cyclist is instructed beforehand to either raise their arm or not, and then either



(a) Time to approach, T^{min} (b) Arm Detection, AD (c) Cyclist tracks relative to intersection center, DTI

Figure 5.2: Context observables used in the cyclist scenario. (a): The extrapolated time until ego-vehicle reaches cyclist. (b): The detection of the cyclist's raised arm. (c): A static environment map is built offline through SLAM. The map's coordinate system is aligned with the intersection center. By projecting tracked cyclist positions to this coordinate system, their longitudinal distance to the intersection is obtained.

Table 5.1: Breakdown of the number of tracks in the cyclist dataset for the sub-scenarios with normal (above the line) and anomalous contextual behavior (below the line).

Sub-scenario			Occurrences
non-critical	arm not raised	straight	6
non-critical	arm not raised	turn	6
non-critical	arm raised	turn	6
critical	arm not raised	straight	10
critical	arm raised	turn	7
non-critical	arm raised	straight	5
critical	arm raised	straight	4
critical	arm not raised	turn	7

turn left or continue straight at the intersection.

The dataset contains the longitudinal and lateral position of each cyclist in a common reference frame, obtained using a stereo camera setup, as well as measurements on the three context cues shown schematically in fig. 5.2. The first, T^{min}, is the time it takes for the vehicle to overtake the cyclist if they would both keep moving with the same velocity. The second, Arm Detector (AD), indicates whether the arm of the cyclist is raised. This is given as a confidence score as computed by a Naive Bayes classifier. The third is Distance To Intersection (DTI), the distance between the cyclist and the intersection along the longitudinal axis of the road.

For each of the three context cues given above, the dataset provides an annotated ground truth value. For example, the arm detector scores come with the annotation on whether the arm of the cyclist is in fact raised or not. Additionally, the dataset provides an estimate of the true position of the cyclist, generated from a Kalman smoother run over each of the trajectories.

The tracks are divided into several sub-scenarios, based on whether the cyclist turned left or went straight, whether the arm was raised or not, and how critical the situation was. These

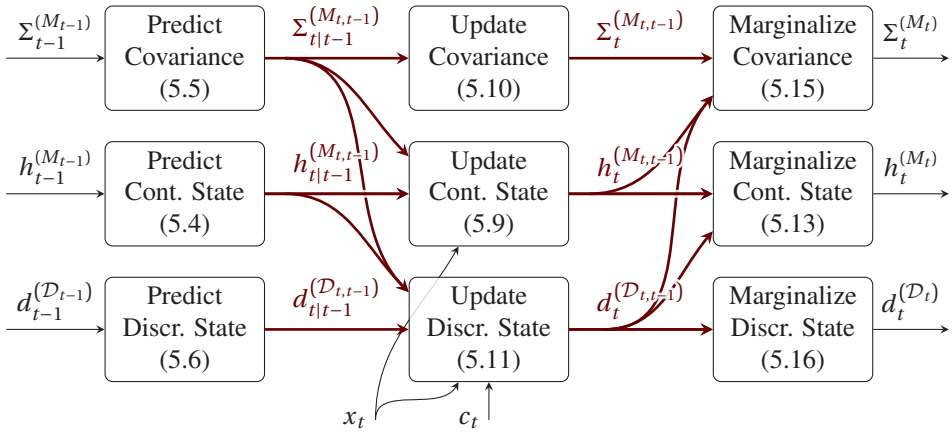


Figure 5.3: A schematic overview of how the DBN incorporates contextual (c_t) and positional (x_t) measurements to infer the state over time. Each block corresponds to an equation, referenced in brackets. The computations are done in joint domains ($(M_{t,t-1})$ and $(\mathcal{D}_{t,t-1})$) for the bold colored lines and in a single domain (e.g. $(\mathcal{D}_{t-1})$) for the thin black lines.

5

sub-scenarios are divided into two categories, based on whether the overall combination of context cues refers to a typical ("normal") scene in real traffic or not. For instance, raising an arm in a critical situation before turning left is considered a typical combination of context cues in such a scenario, whereas not raising an arm in a critical situation is not. The number of tracks per sub-scenario is given in table 5.1.

Each track involving turning has the frame where the cyclist first visibly starts to turn manually labeled as TTE = 0. Frames before and after the labeled frame have negative and positive TTE values, respectively. In the experiments, TTE is used to temporally align tracks in a meaningful way [9]. For the straight tracks, TTE= 0 is defined as the first frame on which the cyclist is past the point on the intersection where 25% of all turning tracks have already started their turn, according to the annotations.

All relevant experimental data (i.e. images, vehicle ego-motion, cyclist detections, extracted cues, and ground truth) is made available to the scientific community for non-commercial benchmarking.[1]

5.2. METHODOLOGY

THIS section discusses the specific version of a DBN as described in [20], although the methodology can be used for alternative scenarios as well, e.g. [34]. At any given time step t, there is a measurement $y_t = [x_t, c_t]$ which contains the position x_t and N_c context cues $c_t = [c_{1_t}, \ldots, c_{N_c t}]$, where $c_t \in \mathbb{R}^{N_c}$. The state of the DBN itself is defined by a partially observable continuous hidden state h_t and discrete hidden state $\mathcal{D}_t$. The discrete hidden state $\mathcal{D}_t = [M_t, z_{1_t}, z_{2_t} \ldots z_{N_z t}]$ specifies the current dynamic mode M_t as well as N_z discrete variables representing the state of the context cues. For a single time step, there are in total $|\mathcal{D}| = |M| \times |z_1| \times \cdots \times |z_{N_z}|$ possible combinations for the discrete state. Note that not each

[1] The dataset is available at http://intelligent-vehicles.org

discrete state necessarily has a corresponding context measurement, i.e. $N_z \neq N_c$.

In the DBN, the discrete state at time $t = 0$ follows a categorical distribution $\mathcal{D}_0 \sim \text{Cat}\left(\mathcal{P}^0\right)$ with parameters $\mathcal{P}^0$, and can stochastically transition at subsequent time steps to a new value:

$$\mathcal{D}_t \sim \text{Cat}\left(\mathcal{P}^{(\mathcal{D}_{t-1})}\right). \tag{5.1}$$

Here, $\mathcal{P}^{(\mathcal{D}_{t-1})}$ is a $|\mathcal{D}|$-dimensional parameter vector conditioned on the past discrete state $\mathcal{D}_{t-1}$, i.e. the row from a $|\mathcal{D}| \times |\mathcal{D}|$ transition table. Of the N_z discrete variables z_{n_t}, N_c have corresponding measurements c_{n_t} and their probability distribution $p\left(c_{n_t}|z_{n_t}\right)$ is specific for that context cue.

The propagation of the continuous state h_t over time and the relation between the measurement x_t and the continuous state h_t are as follows:

$$h_t = A^{(M_t)} h_{t-1} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}\left(\mu_\epsilon^{(M_t)}, \Sigma_\epsilon^{(M_t)}\right) \tag{5.2}$$

$$x_t = C\, h_t + \eta_t, \qquad \eta_t \sim \mathcal{N}\left(0, \Sigma_\eta\right). \tag{5.3}$$

Similar to eq. (5.1), the superscript $^{(M_t)}$ indicates that there is a separate matrix/vector for each of the N_M models M_t. The matrices A and C are model parameters. Both the measurement and the state are perturbed by Gaussian noise that is not directly measurable, η and ϵ, respectively, with parameters μ_ϵ, Σ_ϵ, and Σ_η. Finally, the prior on the continuous state is normally distributed, $p\left(h_0\right) \sim \mathcal{N}\left(h_0, \Sigma_0\right)$ with parameters h_0 and Σ_0.

With this model, inference consists of three main steps: *Predict*, *Update*, and *Marginalize*, see [20]. A schematic overview of the data flow for one time step in the algorithm is shown in fig. 5.3. At each step, the algorithm computes a new distribution over the latent state of the DBN. The probability of a discrete state $\mathcal{D}_t = [M_t, z_{1_t}, \ldots z_{N_z t}]$ is expressed with a scalar $d_t^{(\mathcal{D}_t)}$. The continuous state is represented by N_M means $h_t^{(M_t)}$ and covariances $\Sigma_t^{(M_t)}$, one for each model M_t.

The subsections below list the equations corresponding to each step for the latent states, and fig. 5.3 also refers to these equations per step.

PREDICT

The predict step computes $p(h_t, \mathcal{D}_t, \mathcal{D}_{t-1}|y_{0:t-1})$ given $p(h_{t-1}, \mathcal{D}_{t-1}|y_{0:t-1})$ from the previous iteration, . Prediction of the next continuous state is done using a Kalman filter, for every N_M^2 combination of current and previous model:

$$h_{t|t-1}^{(M_{t,t-1})} = A^{(M_t)} h_{t-1}^{(M_{t-1})} + \mu_\epsilon^{(M_t)} \tag{5.4}$$

$$\Sigma_{t|t-1}^{(M_{t,t-1})} = A^{(M_t)} \Sigma_{t-1}^{(M_{t-1})} A^{\top(M_t)} + \Sigma_\epsilon^{(M_t)} \tag{5.5}$$

$$d_{t|t-1}^{(\mathcal{D}_{t,t-1})} = \mathcal{P}_{\mathcal{D}_t}^{(\mathcal{D}_{t-1})} d_{t-1}^{(\mathcal{D}_{t-1})} \tag{5.6}$$

The superscript $^{(M_{t,t-1})}$ indicates that the predicted state is computed for each possible model combination at time t and $t-1$ (i.e. their joint probability). Accordingly, the distribution of the hidden state is defined as a mixture of N_M^2 Gaussians [20]. Similarly, the discrete state probability $d_{t|t-1}^{(\mathcal{D}_{t,t-1})}$ is computed for the $|\mathcal{D}|^2$ joint discrete state combinations.

UPDATE

The update step obtains the posterior joint probability $p(h_t, \mathcal{D}_t, \mathcal{D}_{t-1}|y_{0:t})$ by incorporating measurement $y_t = [x_t, c_{1_t}, \ldots, c_{N_{ct}}]$, akin to a Kalman update:

$$S_t^{(M_{t,t-1})} = C\Sigma_{t|t-1}C^\top + \Sigma_\eta^{(M_t)} \tag{5.7}$$

$$K_t^{(M_{t,t-1})} = \Sigma_{t|t-1}C^\top S_t^{-1} \tag{5.8}$$

$$h_t^{(M_{t,t-1})} = h_{t|t-1} - K_t(x_t - Ch_{t|t-1}) \tag{5.9}$$

$$\Sigma_t^{(M_{t,t-1})} = (\mathbf{I} - K_tC)\Sigma_{t|t-1} \tag{5.10}$$

$$d_t^{(\mathcal{D}_{t,t-1})} = d_{t|t-1}\, p\left(x_t|h_{t|t-1}, \Sigma_{t|t-1}\right) \prod_{n=1}^{N_c} p\left(c_{n_t}|z_{n_t}\right) \tag{5.11}$$

Where $p\left(x_t|h_{t|t-1}, \Sigma_{t|t-1}\right)$ is the likelihood of the measurement x_t for the state $h_{t|t-1}^{(M_{t,t-1})}$ with covariance $\Sigma_{t|t-1}^{(M_{t,t-1})}$. For readability, the superscript $^{(M_{t,t-1})}$ has been omitted on the right-hand side for the variables $h_{t|t-1}$, $\Sigma_{t|t-1}$, S_t, and K_t, and $d_{t|t-1}^{(\mathcal{D}_{t,t-1})}$ is written as $d_{t|t-1}$. The update step is only applied for the continuous state when integrating past or current measurements, and not for predictions. The same applies for updating the discrete state, unless a reasonable estimate of the future measurement can be made.

MARGINALIZE

Computing the full joint probability by iterating the previous steps would quickly become intractable, as there are $(N_M)^t$ motion model combinations after t steps. To make inference tractable, $p(h_t, \mathcal{D}_t, \mathcal{D}_{t-1}|y_{0:t})$ is marginalized over the past discrete state $\mathcal{D}_{t-1}$ to obtain approximation $p(h_t, \mathcal{D}_t|y_{0:t})$. The mixture of Gaussians over joint models $(M_{t,t-1})$ is therefore collapsed to a mixture over only the current motion models (M_t) through moment matching [20]:

$$d_t^{(M_{t-1})} = \sum_{\mathcal{D}_{t,t-1}/(M_{t-1})} d_t^{(\mathcal{D}_{t,t-1})} \tag{5.12}$$

$$h_t^{(M_t)} = \sum_{M_{t-1}} h_t^{(M_{t,t-1})} d_t^{(M_{t-1})} \tag{5.13}$$

$$e_t^{(M_{t,t-1})} = h_t^{(M_{t,t-1})} - h_t^{(M_t)} \tag{5.14}$$

$$\Sigma_t^{(M_t)} = \sum_{M_{t-1}} \left(\Sigma_t^{(M_{t,t-1})} + e_t^{(M_{t,t-1})} e_t^{\top(M_{t,t-1})}\right) d_t^{(M_{t-1})} \tag{5.15}$$

$$d_t^{(\mathcal{D}_t)} = \sum_{\mathcal{D}_{t-1}} d_t^{(\mathcal{D}_{t,t-1})} \tag{5.16}$$

Here, in eq. (5.12) the notation $\mathcal{D}_{t,t-1}/(M_{t-1})$ refers to all variables in the joint discrete state $\mathcal{D}_{t,t-1}$ except (M_{t-1}).

With the general model structure defined, the following sections specify the dynamics and context used for the VRU scenario of interest, in two parts: first, the sections of the model that are defined a priori, i.e. the model definition, and secondly the sections of the model that are estimated from the available data, i.e. the parameter estimation.

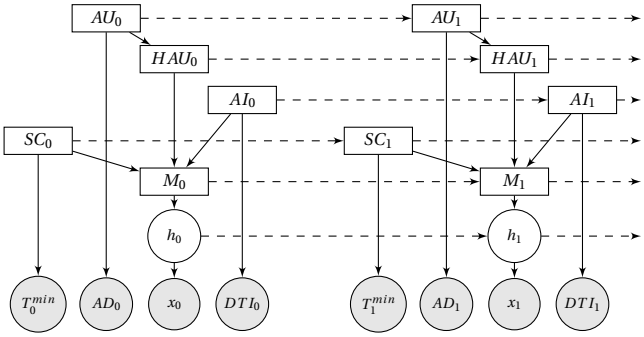


Figure 5.4: The graph representation of the DBN. Rectangular nodes are discrete, round nodes are continuous. Gray nodes indicate measured values.

5.2.1. MODEL DEFINITION

The cyclist can switch between the motion types cycling straight and turning left. Each of these motion types is modeled as a constant velocity linear model, which both have their own latent constant velocity. The elements in the continuous hidden state vector $h_t \in \mathbb{R}^6$ are then the lateral and longitudinal position, the lateral and longitudinal velocity of the cyclist if turning left, and the lateral and longitudinal velocity of the cyclist if moving straight. The noise acting on the latent state is assumed to be zero-mean, only affects the position of the cyclist, and is equal for both dynamic modes. The matrices from eq. (5.2) are then as follows:

$$A^{(1)} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \; A^{(2)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \; \mu_\epsilon^{(M_t)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \; \Sigma_\epsilon^{(M_t)} = \begin{bmatrix} (\cdot) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{5.17}$$

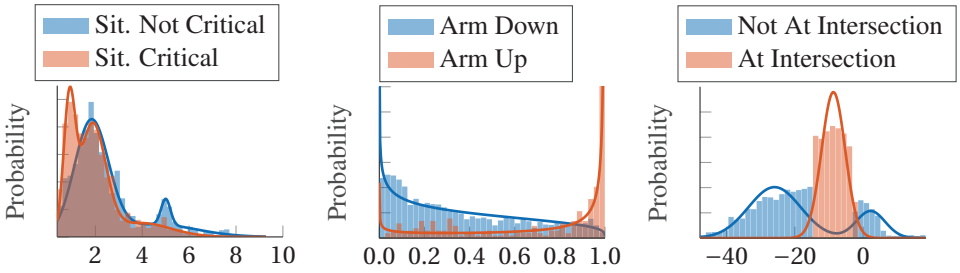
Here, $\mathbf{I}$ defines as a 2×2 identity matrix, $\mathbf{0}$ represents a 2×2 matrix of zeros, and $(\cdot)$ indicates this 2×2 matrix will be estimated from the data. The observations $x_t \in \mathbb{R}^2$ from eq. (5.3) are the observed lateral and longitudinal position with observation matrix $C = [\mathbf{I}\ \mathbf{0}\ \mathbf{0}]$. The three contextual measurements are those shown in fig. 5.2. When predicting, the likelihood of AI_{t+n} is computed using the expected position of the cyclist at that time step.

The discrete hidden state $\mathcal{D}_t = [M_t, AU_t, HAU_t, AI_t, SC_t]$ contains the current model M_t and four context-related binary variables: whether the cyclist's arm is raised, AU_t, whether the cyclist's arm has been raised, HAU_t, whether the cyclist is at the intersection, AI_t, and whether the situation is critical, SC_t. Each variable in the discrete state is assumed to only depend on parts of the previous discrete state, as is shown in fig. 5.4. This leads to five separate transition tables, one for each discrete state: $\mathcal{P}_{AU}$, $\mathcal{P}_{HAU}$, $\mathcal{P}_{AI}$, $\mathcal{P}_{SC}$, and $\mathcal{P}_M$. For $\mathcal{P}_{HAU}$, to represent the notion whether the cyclist has had an arm up, it encodes the following rule:

$$p(HAU_t | HAU_{t-1}, AU_t) = \begin{cases} \text{true} & \text{if } (HAU_{t-1} \vee AU_t) \\ \text{false} & \text{otherwise.} \end{cases} \tag{5.18}$$

5.2.2. PARAMETER ESTIMATION

Parameter estimation for the continuous state priors and their noise distributions is done using maximum likelihood estimates of the ground truth position and velocity. Because



(a) Time to approach, T^{min} (s) (b) Arm Detection confidence, AD (c) Distance To Intersection, DTI (m)

Figure 5.5: Histograms and the fitted distributions of the context observations c_t for the cyclist scenario, conditioned on their ground truth context states z_t. See fig. 5.2 for a visual description of the context observations. (a): The time until the ego-vehicle would approach the cyclist, if both kept moving at the same speed, conditioned on SC (critical vs non-critical). (b): The arm detector's confidence conditioned on AU (cyclists has arm up vs arm down). (c): The cyclists' longitudinal position conditioned on AI (cyclist not at intersection vs at intersection).

of the assumption that noise only affects the position, the ground truth velocities of both motion types of a trajectory are constant. The ground truth velocity for each motion type per trajectory is therefore computed as the average change in position for that motion type.

With all continuous states h_t fully defined for all tracks, ϵ_t is computed using eq. (5.4), i.e. $\epsilon_t = h_t - A^{(M_t)} h_{t-1}$. From these ϵ_t the process noise covariance Σ_ϵ is estimated. Likewise, observation noise covariance Σ_η is estimated from the differences between ground truth and measured positions, since $\eta_t = x_t - C h_t$ from eq. (5.5). Finally, the mean and covariance parameters of the state priors can be estimated from the h_0 of all training tracks.

The prior and transition probability tables for the discrete context states AU, AI are obtained by counting and normalizing the occurrences in the ground truth labels. The same applies to the dynamic switching state M, conditioned on HAU, SC, and AI. The transition probability for HAU is a logical OR, as described in eq. (5.18). Since there is only got one SC label per track, the SC transition probability is fixed to 1/100 for changing state.

Estimating the parameters of the conditional distributions is straightforward, as the values of the latent variables are known. Figure 5.5 shows the empirical distributions of the observables on the cyclist dataset, with the estimated conditional distributions overlaid. These are Mixtures of Gaussians (MoGs) for Situation Critical, Situation Not Critical, and At Intersection. The Expectation-Maximization [100] algorithm is used to fit the Gaussian mixtures. Arm Down and Arm Up are modeled as beta distributions. At Intersection is modeled as a single Gaussian.

5.3. EXPERIMENTS

THE proposed DBN is compared to a variant without cues (a Switching Linear Dynamical System (SLDS)), and a baseline constant-velocity LDS. Leave-one-out cross-validation is used to separate training and test sequences. Sequences from anomalous sub-scenarios are always excluded from the training data.

The metrics used for evaluation are similar to those of the previous chapter. One important difference, however, is that the measurements now encompass more than just the

Table 5.2: Log-likelihood at 16 time steps (~ 1 second) into the future for TTE $\in [-15, 15]$. Below sub-scenarios indicate the expected behavior of a cyclist.

Sub-scenario				Full model	SLDS	LDS
Normal	critical	arm not raised	straight	**0.05**	-0.23	0.00
	non-critical	arm raised	turn	**-2.36**	-3.19	-22.75
	critical	arm raised	turn	**-2.38**	-2.77	-16.66
	non-critical	arm not raised	straight/turn	-2.28	**-2.22**	-14.88
		over all normal sub-scenarios		**-1.93**	-2.22	-14.60
Anomalous	critical	arm not raised	turn	-14.33	-4.62	-31.91

position. Given the measurements up to time step t, each model computes a distribution of the future position x_{t+n} at time step $t+n$: $p\left(x_{t+n}|y_{0:t}\right)$. This predictive distribution is evaluated $n = 16$ steps (one second) into the future, around the point where the cyclist may turn left: the range $TTE \in [-15, 15]$. Let $\widehat{x}_{t+n}$ be the actual future position in the data. Two different performance metrics are used to evaluate the sequences, namely, the log-likelihood of this future position under the predicted distribution (higher is better) and the Euclidean distance between predicted expected position and this actual future position (lower is better).

$$ll\,(t+n|t) = \log p\left(x_{t+n} = \widehat{x}_{t+n}|y_{0:t}\right). \tag{5.19}$$

$$error\,(t+n|t) = \left\|\mathbb{E}_{x_{t+n}}\left[p\left(x_{t+n}|y_{0:t}\right)\right] - \widehat{x}_{t+n}\right\|. \tag{5.20}$$

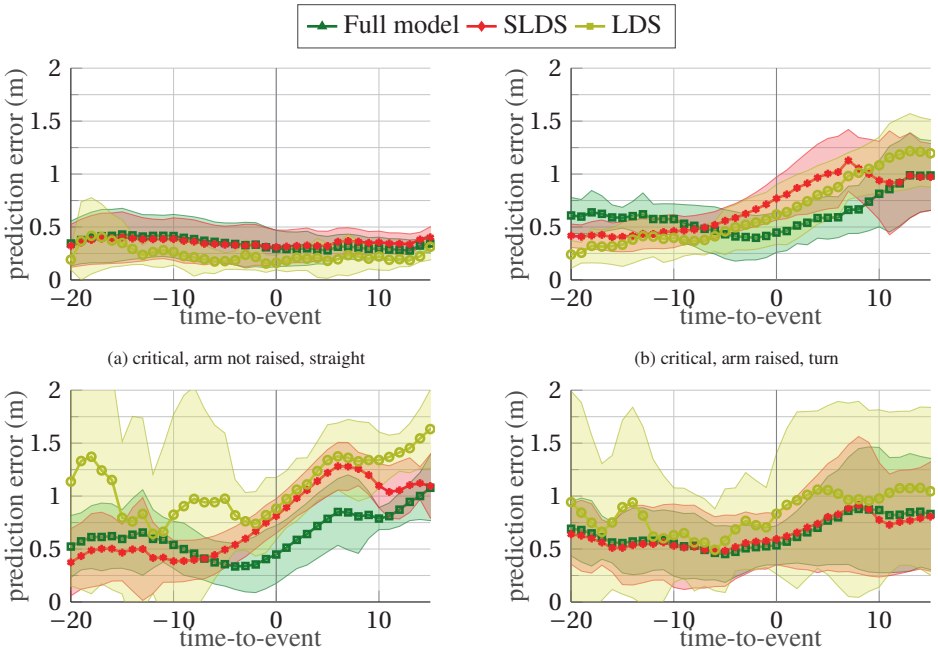
5.3.1. COMPARISON WITH BASELINES

The log-likelihoods of the one-second-ahead prediction (16 time steps) are shown in table 5.2, averaged over TTE $\in [-15, 15]$, i.e. starting with the prediction for the moment when the cyclist either starts to turn or keeps moving straight. Since the cyclist tracks are long and mostly consist of moving straight, the LDS predictions reflect the common behavior of straight motion with little variance. As a result, its predictions are accurate when the cyclist indeed does not turn. As expected, this comes at the cost of inaccurate predictions in normal and anomalous sub-scenarios where the cyclist does turn.

Compared to the LDS, the switching models demonstrate the benefit of having separate dynamics for straight and turning motion, as the predictions for turning sub-scenarios are considerably more accurate. Furthermore, the full model outperforms the SLDS in all but one normal sub-scenario. On the non-critical sub-scenario where there is ambiguity on whether the cyclist will turn or go straight, the SLDS performs best. Still, the proposed method performs best overall on all normal sub-scenarios, demonstrating that context does improve prediction accuracy generally compared to the LDS and SLDS baselines.

Important to note is that all models obtain a lower predictive likelihood for turning than for moving straight. This is likely a result of the large variance in how cyclists execute the turn. The data shows the cyclists vary in when they initiate the turn and in used turning speed and angle. This variance is also reflected in the predictive distributions, which show larger uncertainty for turning than for moving straight.

During the evaluation period of table 5.2 around $TTE = 0$, the cyclist is always near the intersection. When all earlier predictions ($TTE < -15$) are included, the full model outper-



(a) critical, arm not raised, straight

(b) critical, arm raised, turn

(c) non-critical, arm raised, turn

(d) non-critical, arm not raised, straight/turn

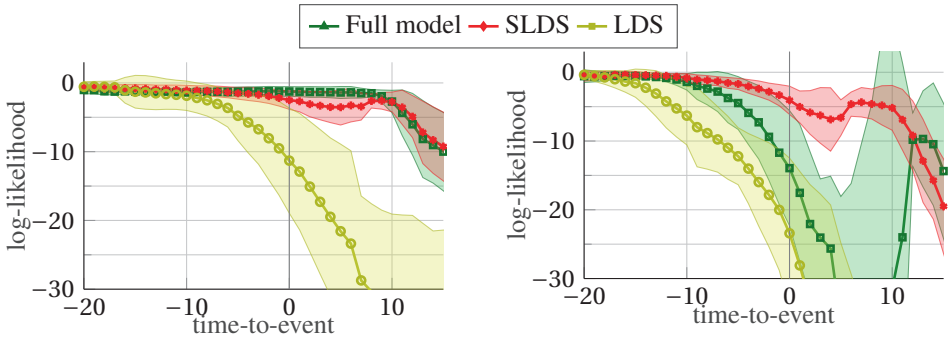
Figure 5.6: Prediction error on the normal sub-scenarios when predicting 16 time steps (~ 1 s) ahead.

forms the baselines in all sub-scenarios. When the cyclists are still far from the intersection, the full model benefits from the static environment context which predicts that turning is not feasible yet.

The final row of the table illustrates the predictive likelihood for the anomalous sub-scenario where the cyclist turns in a critical situation without raising an arm. Here, the full model performs more similar to the LDS, as both expect the cyclist to continue moving straight. The next session will show a more detailed analysis of all sub-scenarios.

COMPARISON OVER TIME

Figure 5.6 compares the prediction error over time around $TTE = 0$ for all normal sub-scenarios. For the critical sub-scenario where the cyclist maintains its straight path, Figure 5.6a, all models demonstrate low prediction errors. The LDS shows the lowest Euclidean error, but it has larger prediction uncertainty as table 5.2 showed. In the critical and non-critical sub-scenarios where the cyclist raises an arm, the full model anticipates the turn and predicts some lateral motion to the left as soon as the cyclist is near the intersection. As a result, the model has a slightly larger prediction error than the baselines before the turn is initiated, but yields lower errors as soon as turnings starts, see figs. 5.6b and 5.6c. It keeps an advantage over SLDS for almost a full second after the turn started, until approximately $TTE = 13$. On the critical sub-scenario, the largest difference in average error of $0.41m$ occurs at $TTE = 5$. On the non-critical sub-scenario without the cyclist raising an arm, fig. 5.6d,



(a) Log-lik. for critical, arm raised, turn (normal sub-scenario)

(b) Log-lik. for critical, arm not raised, turn (anomalous sub-scenario)

Figure 5.7: Prediction likelihood when predicting 16 time steps (~ 1 *s*) ahead on critical sub-scenarios where the cyclist turns. (a): Normal sub-scenario where the arm was raised. (b): Anomalous sub-scenario where the cyclist did not raise an arm to express intent.

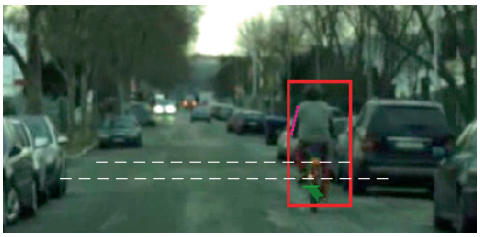
the context cannot uniquely distinguish if the cyclist will turn or continue moving straight. Indeed, here the SLDS and the full model also show similar performance.

Figure 5.7 shows the prediction log-likelihood for two sub-scenarios where the cyclist turns in critical situations, namely the normal sub-scenario where the turn happens after raising an arm (fig. 5.7a), and the anomalous sub-scenario where the turn happens without raising an arm (fig. 5.7b). In both cases, the LDS likelihood drops fast, as it predicts the continuation of the past motion instead of turning. The full model also expects moving straight in case the arm was not raised, therefore its predictive likelihood declines too for the anomaly. Interestingly, the model does adapt after $TTE = 0$ when the turning behavior becomes apparent. Later, at $TTE = 10$, the predictive log-likelihood of all models drops due to the variation in turning behavior.

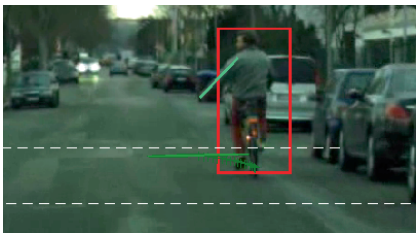
Finally, the LDS shows larger errors for the non-critical sub-scenarios. In these cases, where the vehicle is generally further away, the longitudinal estimates from stereo-vision are more unstable. The LDS is more sensitive to such noise as it adapts to all observed speed changes.

DETAILED ANALYSIS ON A SINGLE TRACK

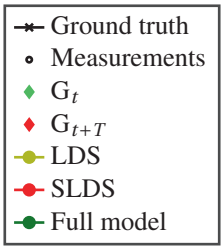
Figure 5.8 shows two snapshots of the prediction over time for a cyclist who is turning at the intersection after holding his arm up while the situation is not critical. Before the cyclist arrives at the intersection, at $t = 67$, the prediction of the full model is very specific. Even though it was already detected that the cyclist raised his arm, he is expected to keep moving straight as he is not yet close enough to the intersection to turn. This prediction is done with less uncertainty than the baseline LDS because the process noise on the LDS must also account for the parts where the cyclist is turning left. At $t = 132$, when the cyclist is at the intersection, there is an increased probability that he could turn left. As a result, the expected future position also shifts left, and the uncertainty region of the prediction increases. The one standard deviation area of the prediction reflects the possibility that the cyclist may still move straight before turning.



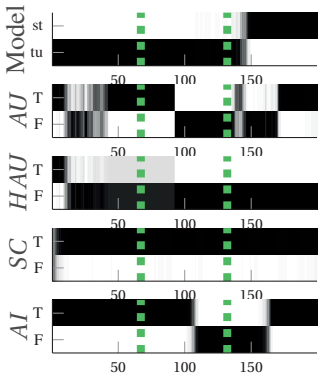
(a) Situation at $t = 67$ (TTE = −68).



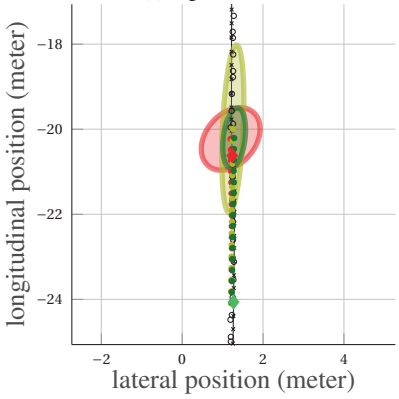
(b) Situation at $t = 132$ (TTE = −3).



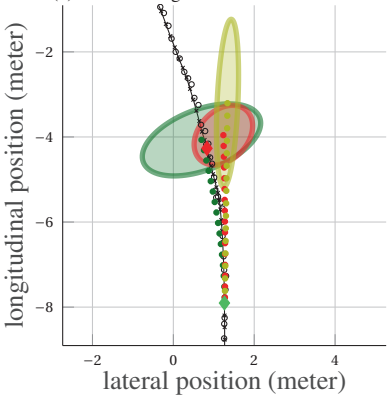
(c) Legend.



(d) Inferred marginal distributions.



(e) Predicted position at $t = 67$ (TTE = −68).



(f) Predicted position at $t = 132$ (TTE = −3).

Figure 5.8: Example of a cyclist turning in after holding his arm up, in a non-critical situation. Predictions are made sixteen time steps (~ 1 *s*) into the future. (a): Cyclist with tracking bounding box (red), arm detection (red line), collapsed predicted distribution of the full model (green ellipsoid shows one standard deviation), and "at intersection" region (white dotted line). (b): The cyclist enters the intersection region, which results in a wider predictive distribution shifted to the left. (d): Marginal posterior distributions of the latent variables in the full model over time. Probabilities range from 0 (black) to 1 (white). Variable labels are **T**rue and **F**alse, and **st**raight and **tu**rning. (e): Predictions (mean, and shaded std.dev.) made at $t = 67$ (green diamond) for the lateral position up to 16 time steps into the future. The red diamond indicates the corresponding future ground truth position. (f): Predictions made at $t = 132$, the moment where the cyclist starts turning.

5.4. DISCUSSION

THIS chapter investigated the use of the DBN from [20] for path prediction of a cyclist near an intersection, who may turn left or continue cycling straight. Measurements of various visual cues inform the model if and when changes in dynamics are likely to occur. An efficient approximate inference method was presented for online path prediction. Parameters are estimated on annotated training data.

The DBN provides predictive distributions of the future position of the tracked objects, reflecting uncertainty on possible trajectories. In the cyclist scenario, there is more variance in turning behavior than in moving straight ahead, making accurate path predictions for the anticipated switch in dynamics challenging. The available context may also be insufficient to unambiguously predict the behavior, as was seen in a cyclist sub-scenario. Here, the performance of the DBN approximated that of the SLDS baseline.

An advantageous property of this model is that it is not a black box, but explicitly defines the relation between context observables, states, motion modes, and their dynamics. This ensures that a designer can inspect the system, investigate how it assesses the context, and determine causes for failure or success. The explicit formulation also facilitates adding additional cues, or improve individual parts (e.g. using different dynamical models) while keeping existing parts unchanged. The generic formulation also ensures that many forms of information can be included, from up-to-date map data to past image classification results.

A downside of the current implementation is that it relies on fully annotated data of all latent factors for parameter estimation. Annotators may not always be able to determine the true labels, and human labeling prevents scaling to a larger dataset. Ideally, the training phase should optimize model parameters on partially or unlabeled data, inferring the latent variables as part of the optimization process. This will be further investigated in the next chapter.

6

Crafted vs. Learned Representations in Predictive Models

'While I'm still confused and uncertain,
it's on a much higher plane, d'you see,
and at least I know I'm bewildered about
the really fundamental and important facts of the universe.'
Treatle nodded. 'I hadn't looked at it like that,'
he said, 'But you're absolutely right.
He's really pushed back the boundaries of ignorance.'
They both savoured the strange warm glow
of being much more ignorant than ordinary people,
who were only ignorant of ordinary things.

from Equal Rites by Terry Pratchett

Parts of this chapter have been published in [99].

The models from the previous chapters can be seen as instances of a larger model category: one where expert knowledge is explicitly crafted into the state representation, and thus it is interpretable. On the other hand, methods that learn a state representations are not easily interpretable, but are more flexible to adapt to the motion and behavioral patterns in the data. While interpretability is a desirable property in an intelligent vehicle, this raises the question how large the performance gap is between methods using a crafted and a learned representation for cyclist path prediction. This chapter therefore compares the performance of the DBN to a Recurrent Neural Network (RNN), of which the state representation is completely learned from data. The comparison is made on the cyclist scenario from the previous chapter.

In order to focus the comparison solely on the state representation, this chapter explains how to ensure that both models are treated similarly with respect to the use of context cues and parameter estimation. First, this chapter provides a RNN which can incorporate the context cues effectively, similar to the DBN. Furthermore, this chapter shows how to employ the same optimization strategy on the DBN as employed for the RNN, namely gradient descent, while ensuring that the meaning of its crafted state representation is not lost.

6.1. Methodology

For the path prediction task, both models predict at every time step t a probability distribution over the top-down 2D position x, n steps into the future, given all previous measurements $y_{0:t}$. A measurement $y_t = [x_t, c_t]$ contains the position x_t as well as multiple context cue measurements $c_t = [c_{1_t}, \ldots, c_{N_{c_t}}]$, where $c_t \in \mathbb{R}^{N_c}$. In general, the prediction task can be written as $p\left(x_{t+n} | y_{0:t}\right)$, and is schematically given in fig. 6.1. This section covers the structure of the two approaches used to determine $p\left(x_{t+n} | y_{0:t}\right)$: one RNN-based (section 6.1.1) and its training scheme (section 6.1.2), and one DBN-based (section 6.1.3) and its training scheme (section 6.1.4).

6.1.1. Recurrent Neural Network Model

For the RNN, the position is supplied as the difference in position between two time steps, $x_t - x_{t-1}$, as is done in [38]. The input for the RNN is then $\tilde{y}_t = [x_t - x_{t-1}, c_{1_t}, \ldots, c_{N_{c_t}}]^\top$. At t_0 the position difference is taken as zero. The architecture of the RNN can be split up into two parts: the first incorporates inputs $\tilde{y}_t$ into the hidden state over time (i.e. inference), and the second predicts a Gaussian distribution as the future trajectory based on the hidden state at a certain time step.

The first part, the inference architecture, is laid out schematically in fig. 6.2. The main component is a Gated Recurrent Unit (GRU), which is used because of its relatively low number of parameters. The hidden layer h_t, a vector with N_h elements, is decoded into an expected input, which is subtracted from the actual input, and the result u_t is fed into the GRU:

$$u_t = W_{enc}\left(\tilde{y}_t - W_{dec}(h_t)\right) \tag{6.1}$$

$$= W_{enc}\left(\begin{bmatrix} x_t - x_{t-1} \\ c_t \end{bmatrix} - \begin{bmatrix} W_{pos}(h_t) \\ W_{cues}(h_t) \end{bmatrix}\right), \tag{6.2}$$

where $W_{enc}(h_t) = w_{enc} h_t + b_{enc}$, a linear layer with w_{enc} and b_{enc} as trainable parameters.

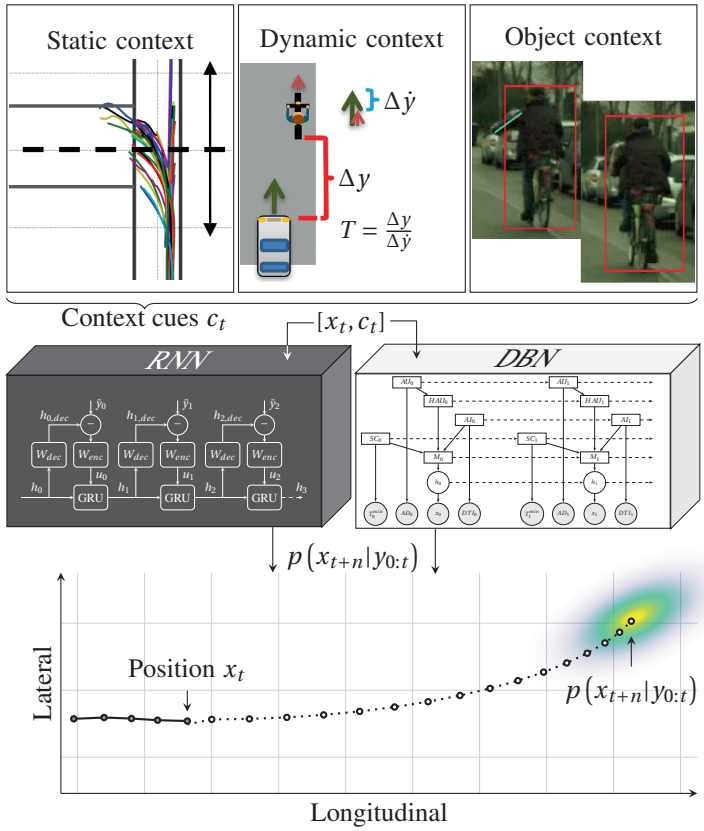


Figure 6.1: Context-based cyclist path prediction with an RNN ("black box", i.e. learned representation) and a DBN ("white box", i.e. crafted representation). The context cues are distance to the intersection (static context), time until the ego-vehicle overtakes (dynamic context), and a possible arm gesture (object context). Predictions involve distributions over future cyclist positions.

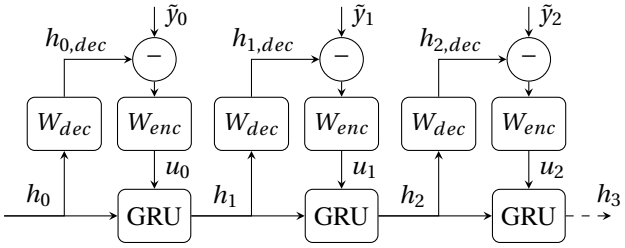


Figure 6.2: The processing of measurements over time by the RNN. This figure shows the incorporation of inputs over three time steps.

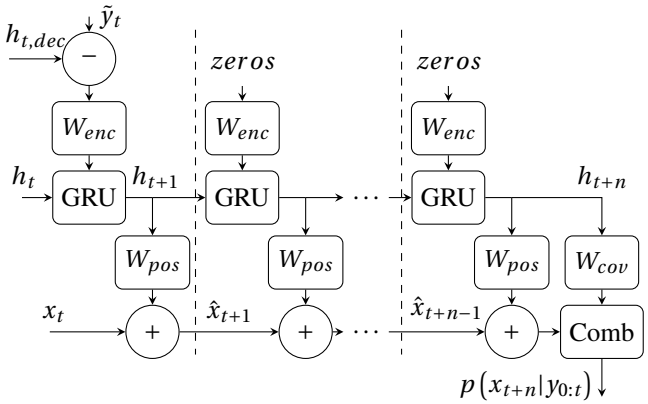


Figure 6.3: The prediction of $p\left(x_{t+n}|y_{0:t}\right)$ at time step t by the RNN. The layers W_{enc} and W_{pos} are shared with the temporal update process (see fig. 6.2). The block labeled Comb is the combination of eqs. (6.4) to (6.9).

All other functions $W_{(\cdot)}()$ are linear layers as well, with parameters $w_{(\cdot)}$ and $b_{(\cdot)}$. The goal of the linear layers is solely to scale the internal representation of the GRU, and as such no nonlinear functions are added.

For prediction, the signal that is fed into the GRU is computed as:

$$u_t = W_{enc}(\mathbf{0}). \tag{6.3}$$

All future hidden states $h_{t+2}, \ldots, h_{t+n}$ are then computed as shown in fig. 6.3. The predicted Gaussian distribution over the future position $\mathcal{N}(\hat{x}_{t+n}, \Sigma_{t+n})$ is computed as in [38]:

$$\hat{x}_{t+n} = x_t + \sum_{i=1}^{n} W_{pos}(h_{t+i}) \tag{6.4}$$

$$\begin{bmatrix} l^{[0]} & l^{[1]} & l^{[2]} \end{bmatrix}^\top = W_{cov}(h_{t+n}) \tag{6.5}$$

$$\sigma_1 = \exp(l^{[0]}) \tag{6.6}$$

$$\sigma_2 = \exp(l^{[1]}) \tag{6.7}$$

$$\rho = \tanh(l^{[2]}) \tag{6.8}$$

$$\Sigma_{t+n} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}. \tag{6.9}$$

6.1.2. RECURRENT NEURAL NETWORK TRAINING

The RNN is trained by minimizing the negative log-likelihood of the predicted Gaussian distribution on the known future position. To ensure that the output of each model is a consistent path the loss is averaged over each time step and the entire range of 1 time step up to and including n time steps ahead. The optimized parameters in the RNN are those of the GRU, the layers W_{enc}, W_{pos}, W_{cues}, W_{cov}, and h_0. Each of these parameters is initialized randomly using the default PyTorch strategy for such layers.

Two training strategies will be considered to reduce overfitting and improve convergence. Firstly, data normalization: The mean and variance of $\tilde{y}_t$ in the training data are computed. The input $\tilde{y}_t$ is scaled and translated accordingly before it is fed to the RNN. The inverse of the scaling and translation is applied to the output of each prediction step in eq. (6.4), i.e. $W_{pos}(h_{t+i})$. The covariance matrix predicted by the RNN is not scaled in any way. Secondly, during training, the hidden state h_t is reset back to the initial hidden state h_0 with a probability of 5% at every time step. This is to prevent the RNN from overfitting by recognizing a specific trajectory from just the first few measurements. Experiments showing the importance of these training strategies are given in section 6.2.1.

6.1.3. Dynamic Bayesian Model

To reiterate section 5.2, the entire state of the DBN is defined by a partially observable continuous hidden state h_t and discrete hidden state $\mathcal{D}_t$ for every time t. The discrete hidden state $\mathcal{D}_t = [M_t, z_{1_t}, z_{2_t} \dots z_{N_z t}]$ specifies the current dynamic mode M_t as well as N_z discrete variables representing the state of the context cues. For a single time step, there are in total $|\mathcal{D}| = |M| \times |z_1| \times \cdots \times |z_{N_z}|$ possible combinations for the discrete state.

In the DBN, the discrete state at time $t = 0$ follows a categorical distribution $\mathcal{D}_0 \sim \text{Cat}\left(\mathcal{P}^0\right)$ with parameters $\mathcal{P}^0$, and can stochastically transition at subsequent time steps to a new value:

$$\mathcal{D}_t \sim \text{Cat}\left(\mathcal{P}^{(\mathcal{D}_{t-1})}\right). \tag{6.10}$$

Here, $\mathcal{P}^{(\mathcal{D}_{t-1})}$ is a $|\mathcal{D}|$-dimensional parameter vector conditioned on the past discrete state $\mathcal{D}_{t-1}$, i.e. the row from a $|\mathcal{D}| \times |\mathcal{D}|$ transition table. Of the N_z discrete variables z_{n_t}, N_c have corresponding measurements c_{n_t} and their probability distribution $p\left(c_{n_t} | z_{n_t}\right)$ is specific for that context cue.

The propagation of the continuous state h_t over time and the relation between the measurement x_t and the continuous state h_t are as follows:

$$h_t = A^{(M_t)} h_{t-1} + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}\left(\mu_\epsilon^{(M_t)}, \Sigma_\epsilon^{(M_t)}\right) \tag{6.11}$$

$$x_t = C\, h_t + \eta_t, \qquad \eta_t \sim \mathcal{N}\left(0, \Sigma_\eta\right). \tag{6.12}$$

Similar to eq. (6.10), the superscript $^{(M_t)}$ indicates that there is a separate matrix/vector for each of the N_M models M_t. The matrices A and C are model parameters. Both the measurement and the state are perturbed by Gaussian noise that is not directly measurable, η and ϵ, respectively, with parameters μ_ϵ, Σ_ϵ, and Σ_η. Finally, the prior on the continuous state is normally distributed, $p\left(h_0\right) \sim \mathcal{N}\left(h_0, \Sigma_0\right)$ with parameters h_0 and Σ_0.

6.1.4. Dynamic Bayesian Network Training

All inference equations of the DBN as given in section 5.2 consist of basic operations such as matrix multiplications or additions. All these steps are differentiable and straightforward to implement in automatic differentiating frameworks, such as Pytorch [69] or TensorFlow [70]. Consequently, the DBN is trained just as the RNN: by minimizing the negative log-likelihood of the predicted Gaussian distribution. The loss is again averaged over the entire range of 1 to n time steps ahead. During optimization, certain parameters are fixed such that the interpretability of the state is guaranteed. For example, assume that the continuous measurements

are the top-down 2D positions. If the first two items in the continuous state vector of length 4 should represent the 2D position, then fixing $C = \left[\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{smallmatrix}\right]$ during optimization ensures a correct state representation. Exact details are given in section 6.1.5.

Additionally, to ensure that the covariance matrices are positive definite, they are reparameterized as upper-triangular matrices U during optimization, e.g. $\Sigma_\eta = U^\top U$ [75, p. 169]. To improve numerical stability, all covariance matrices have a small epsilon 10^{-6} added to their diagonal.

The initial state distribution and the transition matrices for the discrete variables are also re-parameterized, using softmax functions [75, p. 169], since optimizing the values in the probability tables directly could result in invalid values.

For example, a row $\mathcal{P}^{(\mathcal{D}_t)}$ from a probability table is re-parameterized with $|\mathcal{D}|$ learnable parameters $\tilde{\mathcal{P}}^{(\mathcal{D}_t)}$ as follows:

$$\mathcal{P}_i^{(\mathcal{D}_t)} = \frac{\exp\left(\tilde{\mathcal{P}}_i^{(\mathcal{D}_t)}\right)}{\sum_{j=1}^{|\mathcal{D}|} \exp\left(\tilde{\mathcal{P}}_j^{(\mathcal{D}_t)}\right)}. \tag{6.13}$$

Each parameter of the context measurement distributions $p\left(c_{n_t} | z_{n_t}\right)$ that has a limited domain can be reparameterized as well. For example, the variance σ of a Gaussian can be kept positive by reparameterizing it as $\sigma = \exp(\tilde{\sigma})$.

For the initial value of all parameters, one can select a more reasonable initial estimate than random values specifically because each parameter and state variable has a certain interpretation assigned to it, unlike the RNN. For certain parameters this is done by defining them explicitly, such as the motion models $A^{(M_t)}$ and measurement model C. For the noise parameters of eqs. (6.11) and (6.12), the discrete measurement likelihoods $p\left(c_{n_t} | z_{n_t}\right)$, and the discrete state transition probability $\mathcal{P}^{(\mathcal{D}_t)}$, there are two options.

The first option, *annotation-based initialization* as done in chapter 5, is to estimate these using additional ground truth annotations for the discrete variables. Those annotations, together with the context measurements c_{n_t} are used to fit the $p\left(c_{n_t} | z_{n_t}\right)$ distributions. The transition probability $\mathcal{P}^{(\mathcal{D}_t)}$ is estimated from the discrete state annotations as well. This method is explained in detail in section 6.1.5. A downside is that annotating ground truth for these latent variables is laborious and often ambiguous. The second option, *annotation-free initialization*, is to forego the annotations and select initial values for the variables based on expert knowledge. This has become possible thanks to the optimization step afterward. The two options are described later in the next section.

6.1.5. DBN Scenario-specific Crafting

This section first explains what parts of the model are kept fixed during training to keep the model interpretable. What follows is a summary of the annotation-based method from chapter 5 to find the initial estimate for the remaining parameters. Finally, this section explains the annotation-free method for selecting initial parameters. The training of this method is identical to the annotation-based method.

Model definition and training

The model has two constant-velocity models as dynamic modes: one for when the cyclist moves straight and one for when the cyclist turns left. Its graph representation is given

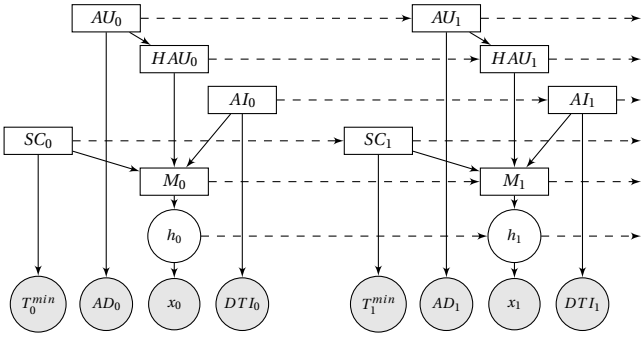


Figure 6.4: The graph representation of the DBN. Rectangular nodes are discrete, round nodes are continuous. Gray nodes indicate measured values.

in fig. 6.4. The elements in the continuous hidden state vector $h_t \in \mathbb{R}^6$ are the lateral and longitudinal position, the lateral and longitudinal velocity of the cyclist if turning left, and the lateral and longitudinal velocity of the cyclist if moving straight. The discrete hidden state $\mathcal{D}_t = [M_t, AU_t, HAU_t, AI_t, SC_t]$ contains the current model M_t and four context-related binary variables: whether the cyclist's arm is raised, AU_t, whether the cyclist's arm has been raised, HAU_t, whether the cyclist is at the intersection, AI_t, and whether the situation is critical, SC_t. As each variable in the discrete state is assumed to only depend on parts of the previous discrete state, there are five separate transition tables, one for each discrete state: $\mathcal{P}_{AU}$, $\mathcal{P}_{HAU}$, $\mathcal{P}_{AI}$, $\mathcal{P}_{SC}$, and $\mathcal{P}_M$. Of these, $\mathcal{P}_{HAU}$ is kept fixed during optimization, the others are optimized.

The optimizable continuous state parameters are shown in table 6.1. When initialized, the A matrices encode two constant-velocity models. During optimization, the A matrices are constrained in such a way that the hidden state keeps the representation of position and velocity, but the constant-velocity assumption is removed. Instead, the velocity at the next time step can be any linear combination of the previous longitudinal and lateral velocity. In the initial parameter estimation, the process noise $\mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ is assumed to only affect the position and is assumed to be zero-mean. During optimization, $\mathcal{N}(\mu_\epsilon, \Sigma_\epsilon)$ can affect both the position and the velocity, and is not assumed to be zero-mean. The measurement noise covariance Σ_η is not constrained. Finally, the continuous initial state distribution $\mathcal{N}(h_0, \Sigma_0)$ is defined with the assumption that the initial dynamic mode of the cyclist is moving straight. Therefore, the position should initially not affect the mean and covariance of the latent turning speed when moving straight. As such, $\mathcal{N}(h_0, \Sigma_0)$ is set up so that the position only correlates with the velocity of the cyclist moving straight. During optimization, the same structure is kept.

ANNOTATION-BASED INITIAL ESTIMATE

This is a summary of the parameter estimation as it was done in chapter 5. The parameters from table 6.1 that require an initial estimate are Σ_ϵ, Σ_η, h_0, and Σ_0. These are found by running a Kalman smoother over the tracks, which gives a ground truth position and velocity at each time step. The transition tables for $\mathcal{P}_{AU}$, $\mathcal{P}_{AI}$, $\mathcal{P}_{SC}$, and $\mathcal{P}_M$ are estimated from the annotations by counting the number of occurrences where the discrete ground truth annotations

Table 6.1: The initial estimation and optimization for all parameters in the DBN that relate to its continuous state. The state vector is, in order: the x and y position, the x and y velocity if turning left, and the x and y velocity if moving straight. $\mathbf{0}$ indicates that part is fixed to be zeros, $\mathbf{I}$ indicates that part is fixed to be identity. In the left column, $(\cdot)$ indicates values retrieved from the initial estimation step [20]. In the right column, it indicates which values are altered during optimization. The size of each $(\cdot)$, $\mathbf{0}$ or $\mathbf{I}$ is 2×2, except for the vectors μ_ϵ and h_0 where it is 2×1.

Initial estimate	Optimization
Kinematic parameters:	
$A^{(1)} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$	$A^{(1)} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & (\cdot) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$
$A^{(2)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}$	$A^{(2)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\cdot) \end{bmatrix}$
$C = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}$	$C = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}$
Noise parameters:	
$\mu_\epsilon = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$	$\mu_\epsilon^{(1)} = \begin{bmatrix} (\cdot) \\ (\cdot) \\ \mathbf{0} \end{bmatrix}, \mu_\epsilon^{(2)} = \begin{bmatrix} (\cdot) \\ \mathbf{0} \\ (\cdot) \end{bmatrix}$
$\Sigma_\epsilon = \begin{bmatrix} (\cdot) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$	$\Sigma_\epsilon^{(1)} = \begin{bmatrix} (\cdot) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\cdot) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \Sigma_\epsilon^{(2)} = \begin{bmatrix} (\cdot) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\cdot) \end{bmatrix}$
$\Sigma_\eta = \begin{bmatrix} (\cdot) \end{bmatrix}$	$\Sigma_\eta = \begin{bmatrix} (\cdot) \end{bmatrix}$
Initial state parameters:	
$h_0 = \begin{bmatrix} (\cdot) \\ (\cdot) \\ (\cdot) \end{bmatrix}$	$h_0 = \begin{bmatrix} (\cdot) \\ (\cdot) \\ (\cdot) \end{bmatrix}$
$\Sigma_0 = \begin{bmatrix} (\cdot) & \mathbf{0} & (\cdot) \\ \mathbf{0} & (\cdot) & \mathbf{0} \\ (\cdot) & \mathbf{0} & (\cdot) \end{bmatrix}$	$\Sigma_0 = \begin{bmatrix} (\cdot) & \mathbf{0} & (\cdot) \\ \mathbf{0} & (\cdot) & \mathbf{0} \\ (\cdot) & \mathbf{0} & (\cdot) \end{bmatrix}$

relevant for that transition table switch from one discrete state to another. Finally, $p\left(c_{n_t}|z_{n_t}\right)$, the distribution for each context feature measurement given their respective discrete variable, is fitted using either a single Gaussian, an MoG, or a beta distribution.

ANNOTATION-FREE INITIAL ESTIMATE

For the initial state h_0, the initial position is taken from the mean position of all initial positions. The velocity when cycling straight is assumed to be 18 km/h. The velocity is assumed to be identical when turning left, albeit at a 45-degree angle. The initial covariance Σ_0 is estimated as a diagonal matrix. The initial position variance is set to the variance of the lateral and longitudinal initial position. The initial covariance of the velocity in both directions and both modes is one-tenth of the initial velocity. The observation noise Σ_η is set to identity in meters. The process noise Σ_ϵ acting on the position is set to one-tenth of the initial velocity.

For the context parameters, the context transition matrices have a 0.01 probability of transitioning to another binary state,

$$\mathcal{P}_{AU} = \mathcal{P}_{AI} = \mathcal{P}_{SC} = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}. \tag{6.14}$$

The model transition matrix $\mathcal{P}_M$ is set to have a transition probability from straight to turning of 0.01 when the conditions of a normal turning subscenario are met, as given in table 5.1, otherwise it is 0. Finally, the parameters for the conditional probabilities $p\left(c_{n_t}|z_{n_t}\right)$ are selected in an intuitive sense: for example, the "at intersection" normal distribution is centered at the intersection, with the "not at intersection" MoG before and after the intersection. The same distribution types from the annotation-based method are used for the annotation-free method.

6.2. EXPERIMENTS

THE predictive accuracy of the RNN and the DBN is evaluated in three separate parts. First, section 6.2.1 evaluates the performance of the RNN and investigates whether incorporating context cues improves its predictive accuracy. Secondly, section 6.2.2 evaluates the performance impact of gradient-based optimization of the DBN parameters in comparison to the previously used annotation-based parameter estimation method. Finally, after having shown that both the RNN and the DBN can be trained on the same context cues with the same optimization strategy, the performance of the RNN is compared to that of the DBN in section 6.2.3.

The RNN and DBN are trained and evaluated on the tracks from the cyclist scenario of the previous chapter. Some of the tracks from the dataset contain frames without position information. Because the proposed RNN has no inherent way to handle missing data, the smoothed tracks as described in section 5.1 are used for both training and evaluation.

Similar to preceding chapters, the predictions are evaluated using the Euclidean distance error and the log-likelihood. Given the measurements up to time step t, each model computes a distribution of the future position x_{t+n} at time step $t+n$: $p\left(x_{t+n}|y_{0:t}\right)$. The predictive distribution is evaluated $n = 16$ steps (one second) into the future, around the point where the cyclist may turn left: the range $TTE \in [-15, 15]$. Let $\hat{x}_{t+n}$ be the actual future position in the data. The evaluation metrics are then as follows:

Table 6.2: The log-likelihood of the predictions 16 steps (one second) in the future, averaged over the period $TTE \in [-15, 15]$. The models are only trained on tracks from the normal sub-scenarios. For the RNNs, the letters indicate which context cues were available to the RNN: **I** if the RNN used the distance to **I**ntersection, a **T** if the RNN used the **T**ime until the vehicle could overtake, and an **A** if it used the probability that the **A**rm was up.

Sub-scenarios	RNN	RNN_I	RNN_T	RNN_A	RNN_IT	RNN_IA	RNN_TA	RNN_ITA
All normal	−0.42	0.68	0.40	0.51	0.47	0.57	0.36	**0.81**
Normal turning	−1.21	0.06	−0.58	−0.32	−0.28	−0.17	−0.70	**0.34**
Normal straight	0.79	1.63	1.90	1.79	1.64	1.73	**2.00**	1.55
All anomalous	−9.47	0.42	−5.54	−5.07	−1.22	−9.72	−8.76	**−11.48**

$$ll(t+n|t) = \log p\left(x_{t+n} = \hat{x}_{t+n} | y_{0:t}\right). \tag{6.15}$$

$$error(t+n|t) = \left\| \mathbb{E}_{x_{t+n}}\left[p\left(x_{t+n}|y_{0:t}\right)\right] - \hat{x}_{t+n} \right\|. \tag{6.16}$$

The predictive distribution for the DBN is a mixture of N_M^2 Gaussians ($N_M = 2$). It is a single Gaussian for the RNN. All models are implemented in PyTorch [69] and evaluated using leave-one-out cross-validation on a Titan X Pascal GPU. For the RNN, after a preliminary hyperparameter search, a hidden layer size of $N_h = 32$ is selected, and trained using the Amsgrad [101] algorithm for 2000 iterations with a learning rate of 0.0015, taking 50 minutes per cross-validation fold. The DBN is trained for 1000 iterations with a learning rate of 0.0001 using the same algorithm, taking 130 minutes per fold. Both models run in real time: 4 ms per frame for the RNN and 10 ms per frame for the DBN.

6.2.1. RNN EVALUATION

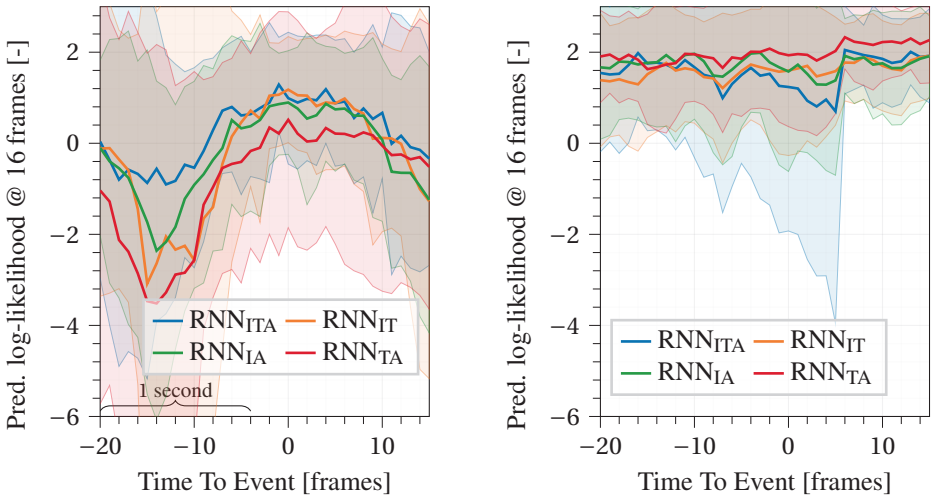
This section evaluates how well the RNN incorporates context cues in its prediction by looking at the performance of the RNN with every combination of context cues as input values. Next, the effectiveness of the training strategies of section 6.1.2 is analyzed through an ablation study.

INCORPORATING CONTEXT CUES IN AN RNN

If the RNN can properly incorporate the meaning of all context cues, the best performing RNN is expected to be the one that includes all context cues.

Table 6.2 shows the one-second ahead predictive log-likelihood of RNNs incorporating different combinations of context cues (the caption defines the naming convention). From left to right, the table shows RNNs with increasingly more information available to them. On the normal sub-scenarios, the addition of one cue (columns RNN_I, RNN_T, and RNN_A) improves the likelihood over the model without any context cues. Using two cues does not improve performance over using a single cue. Apparently, the additional information does not outweigh the disadvantage of increasing the input dimensionality. Utilizing all three context cues (RNN_ITA), however, does result in the best performance.

The full model also attains the lowest log-likelihood of all models on the anomalous data. This further shows that it leverages the context information to inform on its predictions, as the



(a) Normal turning sub-scenarios

(b) Normal straight sub-scenarios

Figure 6.5: One second ahead prediction log-likelihood mean (thick line) and one-sigma standard deviation (shaded area) of RNNs over time. When turning (fig. 6.5a), the RNN with all three context cues (RNN_{ITA}, blue line) performs the best.

only difference between the normal and anomalous sub-scenarios is the validity of the context cues. The full RNN model thus successfully discriminates between such sub-scenarios and has shifted the mass of its predictive distribution away from the anomalous cases.

For a more in-depth analysis, fig. 6.5 shows the log-likelihood over time, using the annotated TTE to temporally align the tracks. These graphs show the log-likelihood of a prediction made at that specific TTE, e.g. the point at TTE = −10 shows the likelihood of the prediction for TTE = 6. Figure 6.5a shows that the RNNs increase in accuracy starting around TTE = −10, a moment where the RNN predicts what happens after the turn. That means that the RNN detects that the cyclist will turn over half a second before the annotated point of turning, TTE = 0.

For the normal sub-scenarios where the cyclist continues straight (fig. 6.5b), all models perform relatively similarly. The performance of the full model does decrease slightly over time. This is in line with the results of table 6.2: the main reason for the overall better performance of the full model is the improved performance on the tracks of the normal turning sub-scenarios, without losing too much performance on the normal straight sub-scenarios.

When comparing the average Euclidean distance error of the predictions, the full model outperforms the other models as well, albeit only slightly: the average Euclidean distance error is 33 cm when evaluated on the tracks from the normal sub-scenarios (34 cm/31 cm on normal turning/straight sub-scenarios, respectively). The other RNNs with one or two context cues have an error between 34 cm and 35 cm, the RNN with no context cues has an error of 49 cm.

Overall, the results suggest that the RNN exploits the additional context cues. This mirrors the results for the DBN found in : both approaches benefit most from combining all distinct types of context in the normal sub-scenarios, while as expected both also assign a

Table 6.3: The categorization of all DBN parameters into distinct groups, to study the effect of optimizing related parameters. The superscript $(\cdot)^{(M_{t-1})}$ indicates that the parameter is distinct for each dynamic mode. The letter in brackets is used to specify what has been optimized in a DBN.

Name of group	Content of group
Context parameters (C)	$\mathcal{P}_{AU}$, $\mathcal{P}_{AI}$, $\mathcal{P}_{SC}$, $\mathcal{P}_{M}$, $\mathcal{P}^0$, $p\left(c_{n_t}\middle\vert z_{n_t}\right)$
Noise parameters (N)	$\mu_{\epsilon}^{(M_t)}$, $\Sigma_{\epsilon}^{(M_t)}$, Σ_{η}, Σ_0, h_0
Kinematic parameters (K)	$A^{(M_t)}$

low probability to the designed anomalous cyclist responses to these context cues. The conclusion is that both models have homogenized context input.

RNN TRAINING STRATEGIES

This section demonstrates the importance of the training strategies discussed in section 6.1.2 through an ablation study. The RNN is trained with all three context cues as additional input and evaluated on all tracks from the normal sub-scenarios. As shown before, the proposed RNN_{ITA} achieves an average prediction log-likelihood of 0.81. Without normalization, the prediction log-likelihood drops to -5.85. Without resetting the hidden layer during training, it drops to -0.61. This shows that both training strategies help improve the accuracy of the RNN.

6.2.2. DBN EVALUATION

The first evaluation of the DBN verifies that the optimization increases the performance compared to the annotation-based initial parameter estimation. The second shows that the optimization improves the alignment of the latent turning probability with the annotated moment of turning. The final evaluation compares the performance of the annotation-based initial estimate with the annotation-free initial estimate.

To better understand the effects of the optimization, the relevant parameters are categorized into three groups, see table 6.3. Various combinations of these groups are either fixed to their initial estimate or optimized. The letter within brackets in the table is used to specify what has been optimized in a DBN. For example, DBN^{CN} has both the **C**ontext and **N**oise parameters optimized. DBN (no superscript) refers to the original, unoptimized DBN from the previous chapter. When optimized, constraints as mentioned in section 6.1.4 apply.

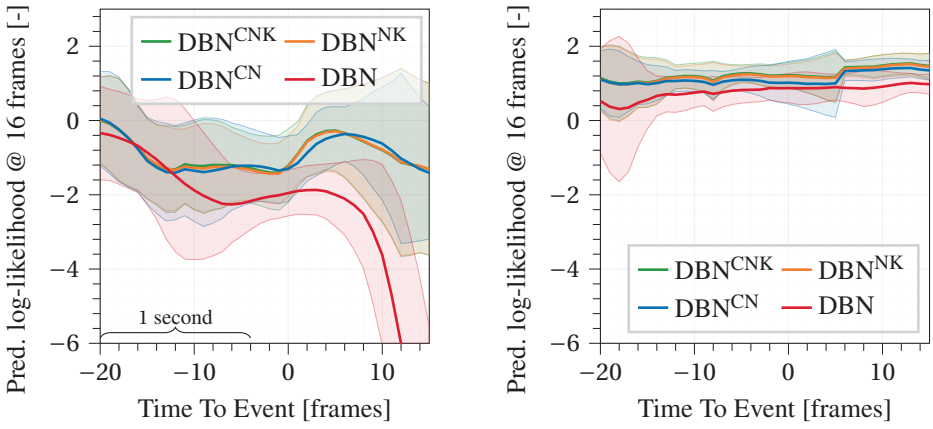
OPTIMIZING THE DBN

Table 6.4 shows the performance of the original and optimized DBNs. Every optimized DBN improves overall performance compared to the original DBN. Optimizing all parameters (DBN^{CNK}) results in the best overall performance.

The Euclidean distance error improves for each optimized model, save one. The unoptimized DBN has an error of 64 cm for the turning sub-scenarios, and 25 cm when moving straight. All optimized DBNs except DBN^{C} attain an error of 39-42 cm when turning, and 22-24 cm when moving straight. For DBN^{C}, the error increases to 67 cm when turning, and decreases to 19 cm when going straight.

Table 6.4: The log-likelihood of the DBNs on their predictions 16 steps (one second) in the future, averaged over the period $TTE \in [-15, 15]$. The models are only trained on tracks from the normal sub-scenarios. The names indicate which parameter groups were further optimized (see table 6.3).

Sub-scenarios	DBN	DBN^{C}	DBN^{N}	DBN^{CN}	DBN^{NK}	DBN^{CNK}
All normal	−1.53	−1.38	−0.22	−0.20	−0.14	**−0.12**
Normal turning	−2.95	−2.63	−1.08	−1.04	−1.02	**−1.00**
Normal straight	0.84	0.69	1.15	1.14	1.25	**1.28**
All anomalous	−2.40	−2.13	**−1.10**	−1.12	−1.11	−1.14



(a) Normal turning sub-scenarios

(b) Normal straight sub-scenarios

Figure 6.6: One second ahead prediction log-likelihood mean (thick line) and one-sigma standard deviation (shaded area) of DBNs over time. In the turning sub-scenarios (fig. 6.6a), the performance of the DBN with no additional parameter optimization (DBN, red line) deteriorates after TTE = 5. The optimized DBNs do not see this deterioration, at the cost of a smaller decrease in performance around TTE = 0. In the straight scenario (fig. 6.6b), optimization steadily improves performance.

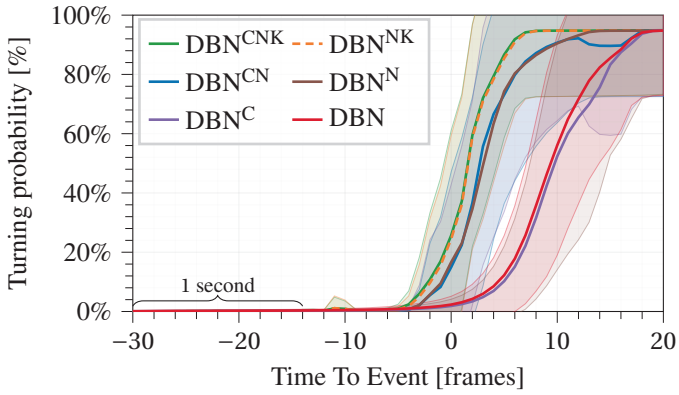


Figure 6.7: The mean (lines) and one-sigma standard deviation (shaded area) of the turning probability for the normal turning tracks. The turning probability is most in line with the annotated moment of turning when all parameters are optimized (DBN^{CNK}, green line, and DBN^{NK}, dashed orange line).

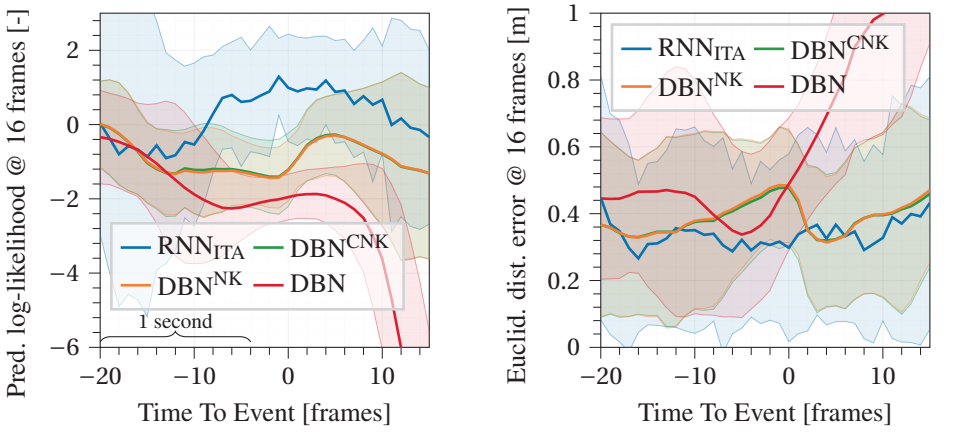
To understand the performance over time, fig. 6.6 shows the prediction log-likelihood of the three best performing optimized DBNs alongside the unoptimized DBN. For the turning case (fig. 6.6a), the main improvement in performance stems from better modeling of the turning dynamics. It is not a constant improvement, however, as the performance of the optimized DBNs dips below the performance of the unoptimized DBN between TTE = −8 and TTE = 2. Because the context cues only inform on the likelihood of *switching* rather than the likelihood of the current dynamic mode, the DBN can only infer the cyclist is turning from position information. For the sub-scenarios where the cyclist continues straight (fig. 6.6b), optimizing consistently improves the performance.

Detection of dynamics change

The probability of being in the turning dynamic mode should remain close to zero when the cyclist moves straight. This is indeed the case: the average probability of turning on straight scenarios is less than 0.5% for all models.

Conversely, the turning probability should go up for the normal turning tracks around TTE = 0, the annotated moment of turning. Figure 6.7 shows how this probability changes over time for the turning scenario. The graph shows that optimizing the context group has no discernible effect on when the model switches to turning: DBN^{CNK} coincides with DBN^{NK}, DBN^{CN} with DBN^{N}, and DBN^{C} with DBN. This is because the context cues inform the model on when the switch from straight to turning is more likely to occur. Whether the cyclist is actually turning is determined by the likelihood of the position measurements and therefore by the dynamics.

The other parameter groups do affect the model's reaction to turning. Optimizing the noise parameter group moves the moment of turning closer to TTE = 0. Optimizing the kinematic parameter group moves it even closer.



(a) Log-likelihood of normal turning sub-scenarios

(b) Euclidean distance error of normal turning sub-cenarios

Figure 6.8: The mean (lines) and one-sigma standard deviation (shaded area) of the one second ahead prediction log-likelihood (fig. 6.8a) and Euclidean distance error (fig. 6.8b) for the normal turning sub-scenarios.

ANNOTATION-FREE INITIAL ESTIMATION

To assess the need for annotations, the annotation-free initial estimation scheme laid out in section 6.1.5 is performed, after which the model is optimized as before, i.e. like DBN^{CNK}. This leads to an average log-likelihood over all scenarios of −0.2, which still outperforms the unoptimized DBN (−1.53, see table 6.4), but it is slightly worse than the log-likelihood of DBN^{CNK} with annotation-based initial estimation (−0.12). At the same time, the average Euclidean distance error over all normal scenarios did improve from 33 cm to 31 cm. This shows one can do without the laborious manual annotation step of the latent variables of the DBN and still obtain a competitive performance.

6.2.3. COMPARISON OF DBN WITH RNN

After having established that both the RNN and the DBN can be trained on the same context cues using the same optimization strategies, both approaches are compared to assess the performance impact for using either a crafted or a learned state representation. When comparing the average log-likelihood, the best RNN in table 6.2 outperforms the best optimized DBN in table 6.4: 0.81 to −0.12. However, at the same time, the average Euclidean distance error over all normal sub-scenarios is 33 cm for both. The source of the Euclidean distance error is not equal for both models, however. The average Euclidean distance error made by the RNN on the turning sub-scenarios and the straight sub-scenarios is almost identical: 34 cm and 31 cm, respectively. Because the RNN is a generic model, it is reasonable that it has no bias towards either type of dynamics. In contrast, the linear models of the DBN can directly encode a cyclist going straight with a constant velocity model, whereas the varying radii of a cyclist turning left cannot be represented as well. The corresponding error values are 39 cm and 23 cm, for the best optimized DBN.

More in-depth, fig. 6.8a shows the predicted likelihood over time for all tracks from the normal turning sub-scenarios, centered around TTE = 0. The results are shown for the best

performing RNN, RNN_{ITA}, as well as the unoptimized DBN and the two best performing optimized DBNs. From TTE = −10, the performance of the RNN (blue line) starts to diverge from the two optimized DBNs (green and orange line). While the gap narrows from around TTE = 0, it never fully closes. When comparing the Euclidean distance errors on the same tracks (fig. 6.8b), one can observe the same divergence between the RNN and the two optimized DBNs starting at TTE = −10 but find that the difference in Euclidean distance error returns to almost zero starting at TTE = 2. It seems that the DBN, when its parameters are optimized, can predict the average position almost as well as the RNN, thus differences in the log-likelihood are mostly due to larger variance in the predictive distribution required to compensate the DBN's linear dynamics.

As a last observation, both the RNN and the DBNs with optimized parameters show a dip in prediction log-likelihood (fig. 6.8a), but the RNN recovers around 10 frames earlier than the DBNs: TTE = −10 versus TTE = 0. This seems to indicate that the current context cues, together with the position information, already contain additional relevant information to predict when a cyclist will turn, but that the DBN is not yet properly capturing this aspect.

6.3. DISCUSSION

THIS chapter examined two models for predicting the distribution over the future position of a cyclist: the RNN and DBN. They use completely different state representations for the dynamic state of the kinematics and context information. When performance is the only goal, the RNN is currently the best choice, as it attained the highest average log-likelihood. By using the right training strategies, the RNN was able to leverage the information present in the context cues (table 6.2). However, because of the "black-box" nature of the RNN, it is difficult to inspect the model and explain how the context cues exactly affect its predictions, other than empirical validation and statistical arguments. On the other hand, the DBN has the benefit that one can ensure that its discrete latent state is interpretable by appropriately specifying the structure of the model (fig. 6.7) and its parameters. These results show that after gradient descent-based optimization similar to the RNN, the performance gap is significantly reduced compared to the unoptimized DBN. The optimized DBN even attains a similar Euclidean distance error (section 6.2.3) as the RNN. Moreover, one can do without the laborious manual annotation step of all latent variables of the DBN (as is the norm in the state-of-the-art experimentation) and still obtain similar performance.

An added value of investigating both an approach with a learned representation such as an RNN and an approach with a crafted representation such as the DBN is that they provide complementary insights into the task: the former shows *if* certain measurements or context cues can help improve prediction, the latter shows *how well* the assumptions on the measurements and context cues hold. In this case, the similar Euclidean distance error but the worse log-likelihood of the DBN compared to the RNN leads to the conclusion that the DBN at times over-estimates the uncertainty in its predictions. This can be attributed to the DBN using only two linear dynamical models for varying turning behaviors. An important direction to improve the DBN is thus to allow for more varied motion dynamics. This could be achieved by loosening existing assumptions, e.g. that noise is constant over time, or by incorporating non-linear motion models with an extended or unscented Kalman filter or particle filter. Another direction is to learn more varied and specialized dynamic modes from the track data itself, e.g. by estimating the number of dynamics and their context with appropriate

priors during model optimization [54, 97].

7

Integrated Path Prediction for Intelligent Vehicles

I couldn't find the sports car of my dreams, so I built it myself.

Ferdinand Porsche

Parts of this chapter have been published in [2].

Figure 7.1: The SafeVRU platform for interaction with VRUs on-board the vehicle demonstrator at the start of the pedestrian scenario. The dummy can be seen in the distance on the right.

THE previous chapters evaluated the performance of path prediction methods in isolation. The effectiveness was asserted by evaluating how well it could predict the future location. This approach makes it possible to compare various prediction methods to each other and ascertain which outperforms the others. However, it does not tell whether any prediction method can benefit an intelligent vehicle, either to assist the driver in emergency situations or to drive autonomously altogether. This is the topic that this chapter concerns itself with: the performance of the DBN in the loop, integrated in the research platform SafeVRU, the converted Toyota Prius depicted in fig. 7.1. Additionally, testing the entire pipeline makes it possible to make some key observations resulting from the interplay of multiple modules, paving the way for future work that bridges the gap between their respective disciplines.

First, this chapter explains how the DBN is used to create an early warning system using the DBN from the previous chapters on the cyclist scenario. For safety, the situation criticality is taken out of the DBN. Secondly, this chapter presents a self-driving setup, where SafeVRU is able to plan collision-free trajectories in the presence of VRUs, using a motion planner based on Model Predictive Contouring Control (MPCC) [102, 103]. This planner allows the self-driving vehicle to adapt its trajectory to the future trajectory of a VRU predicted by the DBN. In this case, a pedestrian (for safety reasons played by an automated dummy) will approach the curbside of a road where the ego-vehicle has right of way, with the intent to cross. Should the pedestrian be unaware of the ego-vehicle because they have not seen it they will cross the road, forcing the self-driving vehicle to plan an evasive maneuver.

7.1. SYSTEM ARCHITECTURE

SAFEVRU relies on the following modules: *(i)* a route planner, *(ii)* a localization module, *(iii)* a perception module, and when driving autonomously *(iv)* a local motion planner and *(v)* real-time PC. Parts *(i)-(iv)* are implemented in the Robot Operating System (ROS) [104]. Figure 7.2 summarizes the overall structure of SafeVRU. The route planner provides the motion planner with the global path to follow and the desired speed of the vehicle. Then, the localization module provides the current position, orientation, and speed of the vehicle to the perception module and the motion planner. Then, the perception module provides the current position of the VRUs and their predicted trajectory. Based on the information provided by *(i)-(iii)*, the motion planner computes a safe (collision-free) trajectory for the vehicle to follow

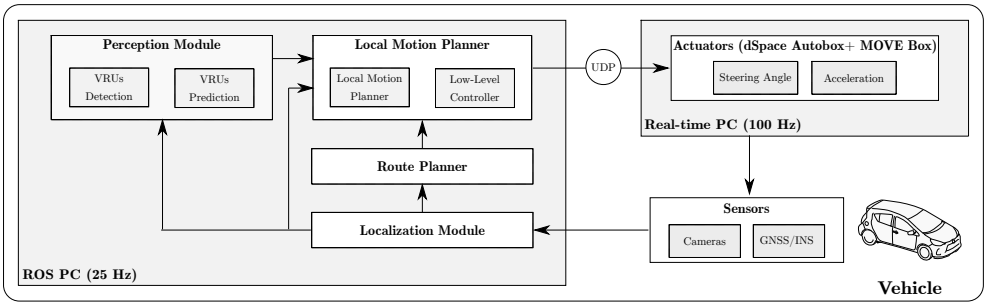


Figure 7.2: Overview of the architecture (from [2]).

using the provided acceleration and steering. The remainder of the section provides more details on the main components of the system architecture.

7.1.1. ROUTE PLANNER

The route planner provides the global path $\mathbf{p}^{\text{path}}(\phi) \in \mathbb{R}^2$ to the local motion planner. The current route planner consists of a set of waypoints selected by the user that connects the current position of the vehicle to the desired destination. These waypoints are then converted by the local motion planner into splines that the vehicle can follow. The route planner provides the desired velocity the vehicle should follow along the path (e.g., according to the rules of the road). The local motion planner has the task of planning collision-free trajectories along the desired path, as detailed in section 7.1.4.

7.1.2. LOCALIZATION

The localization module serves as input to the perception and to the motion planning modules. The perception module needs to correct for the motions of the vehicle to enable tracker-based intent recognition with respect to a world fixed coordinate system. The motion planning module requires the current pose and speed of the vehicle in order to plan the acceleration and steering angle commands.

The ego-vehicle state is estimated using nonlinear state estimation, consisting of an unscented Kalman filter. The state of the ego-vehicle consists of the position, velocity, and angular velocity. The state estimator is implemented using the ROS robot localization package [105]. The localization module works in 2D, projecting all off-plane values to the ground plane. As odometry source, the ROS localization module uses a Global Navigation Satellite System/Inertial Navigation System (GNSS/INS) Combination Advanced Navigation Spatial Dual which provides position, orientation and angular velocity data.

7.1.3. PERCEPTION MODULE

The perception module provides to the local motion planner a probabilistic prediction of the future location of the VRUs in the world-fixed coordinate frame (according to the localization module). This can be separated into the prediction module and the detection module.

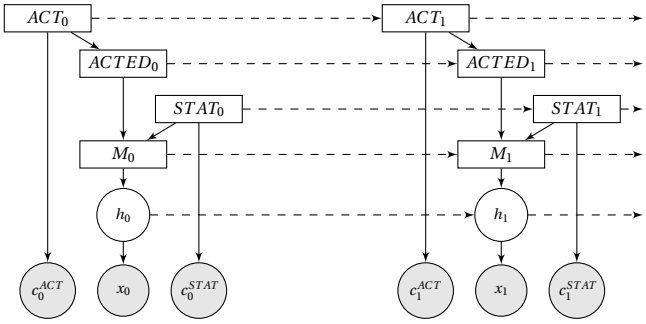


Figure 7.3: The DBN with context cues shown for two consecutive time steps. The discrete, continuous, and observed nodes are rectangular, circular, and shaded, respectively. The binary context nodes represent the relation to the static environment $STAT_t$, and object behavior (i.e. how the VRU acts, ACT_t, or has acted, $ACTED_t$).

PREDICTION MODULE

At time t, the goal is to create a distribution over the VRU position x_{t+n} for all $n \in [1 \dots T]$ time steps into the future. This is done with a modified version of the DBN from the previous chapters. To reiterate, the linear switching model is defined as follows:

$$h_t = A^{(M_t)} h_{t-1} + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(\mu_\epsilon^{(M_t)}, \Sigma_\epsilon^{(M_t)}) \tag{7.1}$$

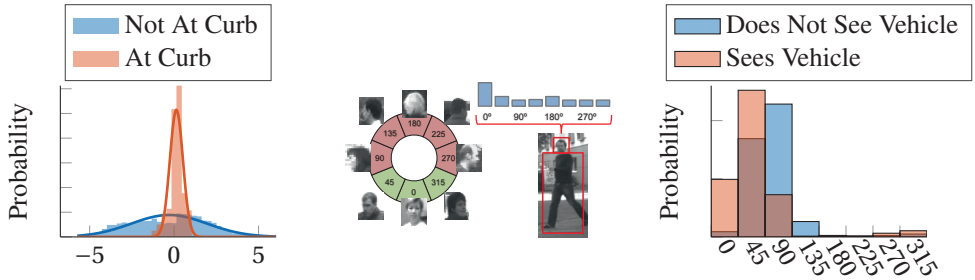
$$x_t = C h_t + \eta_t \qquad \eta_t \sim \mathcal{N}(0, \Sigma_\eta) \tag{7.2}$$

$$\mathcal{D}_t \sim \text{Cat}\big(\mathcal{P}^{(\mathcal{D}_{t-1})}\big), \tag{7.3}$$

where h_t and $\mathcal{D}_t$ are the continuous and discrete state of the model at time t, respectively, and $(\cdot)^{(M_t)}$ indicates that at any time step the state propagation is done with the state matrix A, noise mean μ_ϵ and noise covariance Σ_ϵ of LDS model M. Figure 7.3 depicts the graphical model, which is slightly different from the previous chapters. For one, the context cues in the DBN are described with general terms to fit with both scenarios. Secondly, the discrete latent state pertaining to whether the situation was critical is taken out for safety reasons. Instead, the scenario is assumed to be always critical, no matter the distance of the ego-vehicle.

In general terms, the latent variables are $STAT_t$ and $ACT_t/ACTED_t$. In particular, $STAT_t$ indicates the spatial context, and $ACT_t/ACTED_t$ indicates whether the VRU is acting/has acted, respectively. For the cyclist scenario, $STAT_t$ is whether the cyclist is at a location on the intersection where they might turn left (i.e. AI_t), and $ACT_t/ACTED_t$ is whether the cyclist is raising/has raised their arm to indicate they plan to turn left (i.e. AU_t/HAU_t). The parameters for the cyclist are taken from the experiments of chapter 5.

For the pedestrian scenario, $STAT_t$ is whether the pedestrian is at the location on the curbside where they might stop to wait for the ego-vehicle to pass, and $ACT_t/ACTED_t$ is whether the pedestrian is looking at/has seen the ego-vehicle. The discrete hidded state is comprised of the position and velocity of the pedestrian in longitudinal and lateral direction. The two dynamic modes are defined as “walking” and “stopping”. For the stopping mode,



(a) Distance to curb [m].

(b) Head orientation class confidences.

(c) Head orientation.

Figure 7.4: Pedestrian context. (a): The distributions of how the distance to curb informs $STAT_t$, i.e. At Curb and Not At Curb, are modeled as Gaussian distributions. (b): Eight separate classifiers classify whether the pedestrian is looking in a certain direction, which together make up the measurement c_t^{ACT}. (c): The distributions of how the classifiers inform the two possible states of ACT_t, i.e. Sees Vehicle and Does Not See Vehicle, are modeled as multinomial distributions. Images adapted from [20].

the velocity no longer affects the position of the pedestrian. This is modeled as follows:

$$A^{(walking)} = \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \qquad A^{(stopping)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{7.4}$$

where $\mathbf{0}$ and $\mathbf{I}$ are a 2×2 zero and identity matrix, respectively. The measurements for the continuous state are, as before, the 2D position. The parameters for the pedestrian are set and selected according to [20].

7

DETECTION MODULE

The detection module gets its required inputs from two data sources: the localization module and a stereo camera setup consisting of two uEye cameras (model: UI-3060CP-C-HQ R2). A Single Shot Detector (SSD) [106] trained on the ECP dataset [14] detects the VRU in front of the vehicle for both scenarios. Using the stereo-camera setup and the absolute location of the ego-vehicle, the location of the VRU is transformed into a temporally consistent reference frame.

For the cyclist scenario, the measurement for being at the intersection, c_t^{STAT}, is based on the distance of the cyclist to the intersection. This is computable thanks to the temporally consistent reference frame. The measurement related to whether the arm is raised, c_t^{ACT}, is detected through a pipeline built on top of the SSD. A crop around the bounding box that is found by the SSD is fed to a ROS implementation of OpenPose [107] to retrieve the 2D skeleton of the cyclist. Finally, a Support Vector Machine computes the probability the cyclist has a raised arm, based on the keypoints from the 2D skeleton.

For the pedestrian scenario the, c_t^{STAT} is based on the distance of the pedestrian to the curbside, and modeled Gaussians (fig. 7.4a). For c_t^{ACT}, the setup from [20] is used: the top of the SSD detection is cropped, and fed to eight separate classifiers. Each separate classifier is tasked to classify whether the pedestrian is looking in a certain direction, as outlined in fig. 7.4b. For example, one of the eight classifiers classifies whether the pedestrian is looking at the vehicle: between $-22.5°$ and $22.5°$. The outputs of the eight classifiers to-

gether constitute the measurement c_t^{ACT}, which then informs the discrete state c_t^{STAT} through a multinomial distribution (fig. 7.4c).

7.1.4. LOCAL MOTION PLANNER

The local motion planner, as described in [2], takes the predicted future trajectory from the DBN together with the given trajectory $\mathbf{p}^{\text{path}}(\phi)$, both given in the temporally consistent reference frame from the localization module, and uses this to plan the steering angle and acceleration. The motion planner is based on MPCC for two reasons. The first is that MPCC acts as both a path-following controller as well as a local motion planner. The second is that MPCC is a constrained optimization-based method, meaning that the predicted path of the DBN can be directly integrated.

As mentioned in [2], the planner receives the vehicle's current state $\mathbf{x}_t = [x_t, y_t, \theta_t, v_t]^{\mathrm{T}}$ (i.e. current position, orientation, and forward velocity) at every time step t, from the localization module. The predicted Gaussian distributions are incorporated into the path prediction by using their two-sigma uncertainty region as ellipsoid constraints [103]. An additional constraint incorporates road boundaries into the planner by forcing it to find a solution that does stray further than a fixed distance from the planned trajectory $\mathbf{p}^{\text{path}}(\phi)$.

When the MPCC optimization problem is solved, it results in the acceleration and steering input $\mathbf{u}_t = [a_t, \delta_t]^\top$ for every time step t, up to T steps ahead: $\mathbf{u}_{t:t+T} = [\mathbf{u}_t, ..., \mathbf{u}_{t+T}]$. The cost function of the optimization defines a tradeoff between multiple factors: how far the vehicle strays from the given trajectory, how far it strays from the preferred velocity, how much acceleration and steering input is needed, and finally a penalty factor based on how close the resulting trajectory gets to the road boundaries and predicted path of the VRU. The planner then sends on the current acceleration and steering input $\mathbf{u}_t$ for time t to the real-time PC. The control commands for time steps $t+1$ up to $t+T$ are not used. Instead, the optimization is run again at time step $t+1$ to compute the sequence $\mathbf{u}_{t:t+T}$, and again only the first item $\mathbf{u}_{t+1}$ is used. Exact details are given in [2].

7.1.5. LOW-LEVEL CONTROL SYSTEM

The direct control of the SafeVRU platform is done by a device developed by the Dutch Organization for Applied Scientific Research (TNO). This device, called the MOVE Box, exploits the adaptive cruise control of the Prius for longitudinal control and exploits the electric power steering system for lateral control.

Communication with the MOVE Box is done through CAN. The commands from the local motion planner are sent to a dSPACE Autobox through Ethernet-UDP, which in turn transmits the message on its CAN bus. The dSPACE also implements low-level safety measures, such as limiting the maximum allowed acceleration and steering wheel acceleration. Additionally, in case the ROS PC fails, or for any other reason no longer transmit steering and acceleration control messages, the dSPACE will send out a neutral steering command, along with a braking command to ensure the vehicle comes to a safe standstill.

7.2. EXPERIMENTS

WITH the components as explained above, the experiments are performed with the SafeVRU vehicle. The ROS modules run on Ubuntu 18.04.1 LTS, with an Intel(R) Core(TM)

Figure 7.5: The in-vehicle visualization for the cyclist, intending to turn left. The top left shows the current probability of the three contextual discrete states, as well as the current dynamic mode. The further the bar is filled green, the higher the probability. Drawn on the cyclist itself on the cyclist is the detection done by the SSD [106] (red rectangle), and the estimated skeleton pose from OpenPose [107] (blue lines). Together, these result in the future trajectory, plotted as green and red squares. The color indicates whether the predicted model at that time step is going straight (green), or turning left (red).

i7-6900K CPU at 3.20GHz, and 64GB RAM. Stereo matching, VRU detection, head orientation estimation, and skeleton pose estimation are done using two Titan X (Pascal) GPUs. The DBN predicts the trajectory one second into the future.

7.2.1. SCENARIO DESCRIPTIONS

In the cyclist scenario (fig. 7.5), the ego-vehicle drives manually at a velocity of 10-15 km/h and is overtaken on the right side by the cyclist going 15-18 km/h. Before the cyclist arrives at the intersection, the cyclist either raises their arm if they plan to turn left, or refrains from doing so if they intend to continue straight. If the predicted future location is in the driving corridor of the ego-vehicle (i.e. the expected future position of the cyclist is on the left side of the right wheel), the vehicle emits an audio warning signal to indicate the vehicle and cyclist might collide at the intersection. As a comparison, an LDS predicts the future trajectory alongside the DBN, which emits a warning tone in a similar fashion. This scenario has been showcased at the Forum for Integrated and Sustainable Transportation Systems (FISTS) 2020 in Delft.

In the pedestrian scenario, the autonomous vehicle follows a straight path at 20 km/h. For safety reasons, this scenario is tested with a programmable, actuated dummy from 4Active Systems, shown in fig. 7.6. The dummy stands on top of a base plate, which can be dragged over the ground to move the dummy laterally over the road. Additionally, the arms and legs are actuated to simulate the gait of a normal pedestrian. To simulate whether the dummy sees

(a) The dummy facing forward.

(b) The dummy facing the SafeVRU vehicle.

Figure 7.6: The dummy from 4Active Systems for the pedestrian scenario. The dummy can be moved laterally automatically: it is pulled over the road through straps attached to the base plate, visible in fig. 7.6a. While it moves laterally, the arms and legs move along to simulate realistic behavior (fig. 7.6b). The head can be moved to look straight, left, or right. The exact actions and lateral velocity profile can be programmed. The setup communicates with a GPS attached to the approaching vehicle, which is used to automatically trigger the dummy to move.

the vehicle or not, the head is actuated as well, making it possible to turn it left or right. The setup also contains a GPS that is mounted on the vehicle. This is used to trigger the dummy to start, thereby synchronizing the vehicle and dummy motion to ensure that the scenario is played as it is supposed to. The dummy is programmed to move at 6 km/h, and to start walking at the moment that it will end up in front of the vehicle on the right side, should both the dummy and the vehicle continue moving straight with their set velocity. An example of the response of the DBN to both the dummy looking as well as the dummy not looking is shown in fig. 7.7. The pedestrian scenario has been showcased at the Intelligent Vehicles (IV) conference 2019 in Paris.

7.2.2. RESULTS

In the scenario where the cyclist raises their arm and turns left, the audio signal of the DBN was heard ahead of the audio signal of the LDS, by 100-200 milliseconds. To ensure the system did not give off any false positives, the scenario was repeated with the cyclist continuing straight without raising their arm. In this case, neither the warning signal of the DBN nor that of the LDS was heard.

The scenario in which the pedestrian does not see the vehicle and therefore crosses the road was executed 31 times, of which the DBN was able to correctly predict the outcome 29 times. The scenario in which the pedestrian does see the vehicle and therefore stops at the curbside was played a total of 31 times, of which 27 runs were predicted correctly.

While evaluating the modules in these two scenarios, the following observations are made:

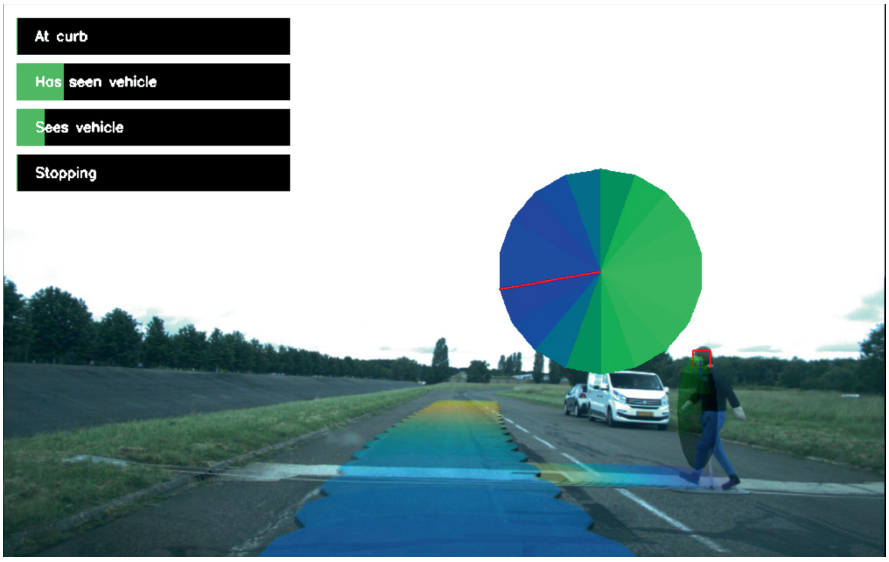
- Both the cyclist skeleton pose and the pedestrian head orientation are estimated after the detections are done, causing additional delay. One solution would be to integrate all measurements module into a single neural network. However, this would make experimentation with additional context measurements, as has been done throughout this thesis, more complex. Instead, the prediction module could be designed with

the ability to process specific measurements asynchronously, incorporating them the moment they are available.

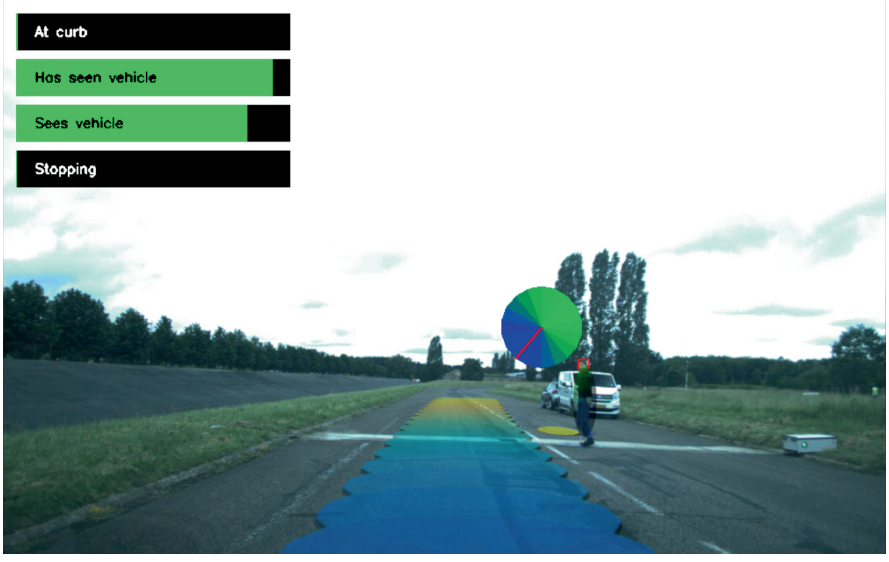
- Both scenarios were designed such that they would show a switch in behavior at a single point in time. However, running the scenarios live indicated that there is another implicit "behavioral switch" in both scenarios: the moment where the first detection is made. At this point the DBN does not yet have an accurate estimate of the velocity of the VRU, making it at this time more difficult to assess whether the dynamics of the VRU will change.
- Three of the pedestrian scenario runs were done while it rained. The detection module performed slightly worse, but it was still able to detect the dummy in a large number of frames. As a result, the whole pipeline was still able to predict the future trajectory of the dummy. In other words, the temporal integration of the DBN improved the effectiveness of the detection module. The most disruptive factor while it was raining was the windshield wiper passing in front of the camera: this caused false positive detections and distorted the depth estimation from the stereo camera setup.
- For one of the failure cases when the dummy was expected to cross, the cause was outside the scope of the scenario: the system tracked a spectator nearby spectator instead of the dummy. However, when such a system would run in real traffic, there is no such thing as out of scope: assessing what scenario is currently relevant for which VRU needs to happen no matter what.
- When the dummy was looking at the vehicle and intended to stop, two failure cases were due to a misaligned person detection, causing the head orientation module to miss the head location altogether. Essentially, the latter module is overly dependent on a correct output from the former. Either the detection module needs to have a more consistent output, or the head orientation module must be designed with the assumption that the person detection can be misaligned, thereby reducing its dependency.
- In the failure cases for the pedestrian scenario where the dummy stopped, the DBN incorrectly expected the dummy to keep moving forward, causing the vehicle to swerve unnecessarily. However, the swerve steered away from the dummy, meaning that the failure response in this scenario was still a safe response. Of course, a swerve to the left is not always a safe response, and this is something that should ideally be assessed online as well.

7.3. DISCUSSION

THIS chapter presented two scenarios in which the DBN was tested to run in real-life applications in real time. In the first scenario, the DBN was used to aid the human driver, by issuing a warning signal if a cyclist were to turn left in front of the vehicle at an intersection. Here, the DBN showed the ability to give a warning 100-200 ms earlier than the LDS baseline. The experiment must be repeated more often to determine the time advantage more robustly. In the second scenario, the DBN was placed in the loop with a motion planner to create a fully autonomous vehicle. Here, the DBN predicted whether a pedestrian – played by an automated dummy for safety reasons– would cross the road, based on whether the pedestrian had seen the oncoming ego-vehicle or not. In this scenario, the DBN succeeded



(a) The dummy is not looking at the vehicle.



(b) The dummy is looking at the vehicle.

Figure 7.7: The in-vehicle visualization for the pedestrian scenario. The top left of each image shows the probability of the three contextual discrete states, as well as the dynamic mode. The further the bar is filled green, the higher the probability. The blue-green circle near the dummy shows the distribution of the head orientation classifiers: blue is more likely, and the red line is the most likely direction. For this demonstration, the dashed centerline of the road acts as the curbside location. (a): Because the DBN currently considers it a low probability that the dummy has seen the vehicle, the DBN's predictions of the future trajectory (blue to yellow line on the ground plane) of the dummy continue past the curbside. In reaction to this, the planner computes a trajectory (blue to yellow trajectory coming from the ego-vehicle's perspective) to the left to evade. (b): The DBN considers it a high probability that the dummy has seen the vehicle, and predicts that the dummy will not move beyond the curbside.

to properly predict the intention of the pedestrian in almost all cases where the pedestrian crossed and succeeded in the stopping case 27 out of 31 times.

Evaluating these scenarios in the loop made it possible to make some observations about the system as a whole. First, when the DBN failed in the stopping scenario, it predicted that the pedestrian would cross. The result is an unnecessary swerve away from the pedestrian. The difficulty lies in the fact that whether such a swerve is safe depends on the scenario definition: if the swerve brings the ego-vehicle onto the lane of oncoming traffic, it would be unsafe. Here, one solution would be to expand the DBN and additionally predict the trajectory of potential oncoming traffic, as the DBN can be extended to predict the action (and interaction) of multiple agents at once [34]. However, this approach does not scale: there are too many scenarios to model every single one separately. Another consequence of a scenario-based solution to autonomous driving is that one needs to somehow decide which traffic participant is in which potential scenario and then predict accordingly. A trivial, but exemplary situation is one of the failure cases when the pedestrian was crossing. In this failure case, the system decided to predict the future trajectory of an audience member instead of the pedestrian dummy. To combat this, future work can investigate methods to automatically learn the different scenarios from the data, through clustering or otherwise.

Furthermore, combining all modules showed a more robust system than expected in some cases, and a less robust system in others. The whole system still worked well when it was raining, as the temporal integration of measurements in the DBN reduced the impact of the lower performance of the detection module. On the other hand, the head orientation module needed the location of 2D bounding boxes to be accurate to assess the position of the head in the image. Additionally, the latency of the whole system was affected by the fact that the head orientation and arm angle measurements were dependent on the detections. In reality, it will always be the case that measurements arrive at different moments in time, however. Future methods should explicitly take this into account, thereby improving the responsiveness of the whole system.

8

CONCLUSIONS

Meaning lies as much
in the mind of the reader
as in the Haiku.

Douglas Hofstadter

THE preceding chapters have investigated cyclist path prediction and its usage in intelligent vehicles. These chapters can be categorized into three subjects. First, chapter 3 investigated the efficacy of 3D person localization for usage in the intelligent vehicle domain. Next, the majority of this thesis (chapters 4 to 6) proposed methods to improve the accuracy of cyclist path prediction. Third and finally, chapter 7 directly studied the applicability of one of the proposed path prediction methods for intelligent vehicles by implementing it in a test vehicle. This chapter will summarize the main findings of those previous chapters. Next, it proposes future work based on the new questions that arose from them, grouped by the three subjects just mentioned.

EVALUATION OF LIDAR-BASED 3D PERSON LOCALIZATION

Chapter 3 presented an experimental study on 3D person localization in traffic scenes using Lidar instead of stereo vision. Experiments on KITTI [18], the de-facto standard for 3D object detection, indicated that the 3D box center localization accuracies were quite high. The biggest contributors to the relatively low AP_{3D} (the performance metric of KITTI) were found to be to the estimates of the bounding box extents, especially width and length. The path prediction methods discussed in this thesis do not utilize these bounding box extents, meaning that the performance metric of KITTI would underestimate the expected performance of the detectors used in conjunction with these path prediction methods.

Of the two evaluated methods, PointPillars [16] and AVOD [17], the former outperformed the latter. Comparing the two datasets, EuroCity Persons 2.5D (ECP2.5D) [19] and KITTI, the performance of both methods was significantly lower on the former. This is attributed to a larger prevalence of distant persons with fewer Lidar points in ECP2.5D, making it not only a larger but also a more challenging dataset. Still, using Lidar-based 3D object detectors is recommended for path prediction.

USING ROAD TOPOLOGY TO IMPROVE CYCLIST PATH PREDICTION

Chapter 4 presented an extension to the Tsinghua-Daimler Cyclist (TDC) benchmark: it extracted trajectories from the TDC benchmark and aligned them with respect to the intersection they were near. For each of these trajectories, the dataset extension also provides information on the road topology, together with what direction the cyclist eventually travels. The chapter provided a prediction method based on a Mixture of Linear Dynamical Systems (MoLDS) that can take the road topology into account and evaluated it on the presented dataset. Compared to the baseline Linear Dynamical System (LDS), it provided an improvement of up to 20% on the Euclidean distance error when predicting one second ahead, showing that such spatial context information can improve path prediction accuracy.

CYCLIST PATH PREDICTION USING CONTEXT-BASED SWITCHING SYSTEMS

Chapter 5 then zoomed in on a specific scenario in order to expand the number and diversity of the considered context cues. It provided and described a dataset that contains the scenario of a cyclist who may cycle straight or turn left on an oncoming intersection, with the ego-vehicle behind said cyclist. The dataset contained annotations and measurements on three contextual cues, namely the interaction of the cyclist with the ego-vehicle as a dynamic environment context, the location of the cyclist with respect to the intersection, and whether the cyclist was raising their arm. The chapter described how to apply the Dynamic Bayesian

Network (DBN) from [20] on this scenario, and evaluated it on the provided dataset. Compared to the Switching Linear Dynamical System (SLDS), when predicting up to $\sim 1\,s$ ahead, the DBN improved up to 0.41 m.

Crafted vs. Learned Representations in Predictive Models

The models used in the previous chapters relied on a manually crafted representation, which provides interpretability and few model parameters. Chapter 6 contrasted this representation to that of a Recurrent Neural Network (RNN) with a learned representation. Both models provide predictions as Gaussian-based distributions over the future position of a cyclist that incorporate various context cues and learn distinct dynamic modes. For the model with a learned state representation, the RNN, the chapter showed that it could leverage the context cues to improve its path prediction. For the model with a crafted representation, the DBN, it explained how to optimize it while keeping its latent state interpretable.

Comparing the two models thus at a level playing field, results indicated that the RNN attained the best predictive performance overall. It significantly outperformed the optimized DBN on the log-likelihood measure and performed similarly on the Euclidean distance error measure, i.e. 31–34 cm vs. 23–39 cm for the DBN. This suggests, more broadly, that if performance is the only relevant metric (and sufficient data is available), a learned state representation is the preferable choice. On the other hand, results showed that optimizing the DBN did partially close the performance gap with the RNN, even without a laborious manual annotation of all latent variables. The conclusion is that crafted state representations remain suitable for safety-critical applications where it is important to understand why a model behaves the way it does, or for those cases where one wishes to further scientific understanding of the underlying causalities.

Integrated Path Prediction for Intelligent Vehicles

Chapter 7 presented two scenarios in which the DBN from chapter 5 was tested to run in real-life applications in real time. It was used for an early warning system in one scenario and for a fully autonomous vehicle in another.

The early warning system was used in conjunction with the cyclist scenario from the previous chapters, warning the driver when the cyclist was predicted to cycle into the driving space of the ego-vehicle. Compared to the baseline LDS, it was able to provide a warning 100-200 milliseconds earlier. Neither the DBN nor the LDS resulted in false positive warnings. For the fully autonomous vehicle, a scenario was played out in which a pedestrian approached the curbside and would cross if it failed to see the approaching ego-vehicle. A set of eight classifiers used visual information of the approaching pedestrian retrieved from a camera to determine in what direction the pedestrian was looking. From this, the system determined whether the pedestrian had seen the ego-vehicle or not, and predicted the future trajectory accordingly. The intent of the pedestrian was properly predicted in almost all cases where the pedestrian crossed and in 27 cases out of 31 total when the pedestrian stopped.

Integrating the DBN in an intelligent vehicle together with the detection and planning modules made it possible to draw additional observations that spanned the entire pipeline. Running the modules together made modules more robust in some cases, and less in others. The head orientation estimation module appeared less robust, as it depended heavily on accurate 2D bounding boxes. A failure to accurately detect the latter caused the failure of the former. On the other hand, the detection module appeared more robust: its relatively

poor performance during rain was remedied by the temporal integration of the detections in the DBN. Furthermore, datasets imply that every measurement type is accessible at the same moment, and as such most prediction methods are design with that assumption built in. Running the entire perception pipeline online made it apparent that each measurement type, contextual and otherwise, had a different processing time. Consequently, the latency of the prediction pipeline was determined by the measurement that was the slowest to process, which does not necessarily need to be the case.

8.1. FUTURE WORK

This section specifies the future work along the three main subjects touched upon in this thesis: 3D person localization, cyclist path prediction, and online path prediction.

3D PERSON LOCALIZATION

Chapter 3 posits that correct bounding box extent estimation is less relevant to current path prediction methods than correct localization and that the metric used to evaluate 3D object detectors should reflect this. Using a position-only metric, future work could compare the Lidar-based detectors to the stereo camera-based 3D position estimation used in this thesis (e.g. section 4.1) to evaluate their respective performance for path prediction.

The cross-dataset evaluation showed a drop in performance when an object detector was evaluated on a different dataset than the one it was trained on. One could then expect a similar degradation in performance if the target Lidar on an intelligent vehicle is different from the one in the dataset used for training. As such, further research is needed on cross-domain adaptation.

Finally, the two detection methods discussed here estimate an orientation alongside the location. While some 2D detectors also directly estimate the orientation (e.g. [108]), it is the standard in 3D object detection. Future work can extend the motion models used in this thesis to include the orientation as an additional measurement, as suggested in [51].

PATH PREDICTION METHODS

Chapter 4 pointed out that using trajectories of a naturalistic dataset gives an accurate assessment of the actual performance, but that some assumptions are still made to facilitate the evaluation of the path prediction method rather than a full prediction pipeline. The three assumptions were the assumption that the road topology is known, that the detected 2D bounding boxes were perfect, and that objects were tracked perfectly. The first was also present in the other two chapters that proposed cyclist path prediction methods (chapters 5 and 6). This is generally a valid assumption, considering road layout datasets exist and some datasets even directly include road layouts [32]. The other two assumptions do affect the performance, however. For example, chapter 6 used the ground truth positions where chapter 5 did not, resulting in a different performance for the unoptimized DBN. Evaluating the performance impact of such assumptions and from that generate rule-of-thumb performance multipliers would be valuable future work, as it could allow for a better comparison between various proposed methods: not all methods are evaluated with the same assumptions in place, as this thesis already shows.

An observation of chapter 5 was that the performance for trajectories of turning cyclists was generally lower than those of cyclists continuing straight. Chapter 6 shows that this is

still the case for the log-likelihood performance of both the optimized DBN and RNN, but not so much for the Euclidean distance error of the RNN. The suggested cause was the higher variance in the motion of the turning trajectories. This raises the question of whether path prediction for cyclists going straight is inherently easier, or whether this is a consequence of the used model. A suggestion for future work is to investigate whether this performance bias is also present in humans, who are at this moment still the most performant path predictors available.

More generally, future work could focus on models that better combine data adaptation and expert knowledge. The DBN and Informed MoLDS (I-MoLDS) could be allowed more flexibility to adapt to the data by means of automatic motion model discovery and to allow more dynamic models [54, 97]. Additionally, they could be extended to higher-order motion models or by introducing non-linear effects, for example by using a non-linear motion model. Another approach is to remove the assumptions that the process and measurement noise have a Gaussian distribution and that they stay constant over time. Introducing non-linear effects would require a different inference approach, however, such as an extended or unscented Kalman filter, or a particle filter. Conversely, the RNN could be more strongly regularized by explicitly encoding physical models or relevant (infrastructure or otherwise) context known to a human expert.

An open challenge is to create predictive methods that scale to a more diverse set of real-life traffic conditions (i.e. multiple scenarios, different road users) while remaining interpretable and incorporating a rich set of context cues. For the DBN, the computational complexity can be partially curbed by limiting the dependencies between discrete states (fig. 6.4), though it may be necessary to learn these dependencies from data instead of designing these relations manually as was done in this study. The interesting alternative is to take a learned representation method and encode expert knowledge in specific areas of the model, thereby making it interpretable and keeping its high performance. Possible directions include combining learned context representations to predict distributions over a fixed set of predefined dynamics [109] and incorporating agent interaction explicitly as a graph structure in the neural networks [44, 110]. In contrast, attentive networks [72] provide interpretability through inspection of node activations for specific inputs, rather than through explicit encoding of expert knowledge.

The advent of large-scale naturalistic datasets such as Argoverse [32] will be important to further these future research directions. Even so, the current findings on the impact of gradient-based optimization are also relevant to other scenarios where DBNs have already been successfully applied without such optimization strategies, such as signalized [35] and non-signalized [20, 42] pedestrian crossing, and in joint pedestrian-driver awareness collision risk estimation [34]. This approach of studying the representation in isolation may be useful for other applications too, such as surveillance with path prediction in crowds, where traditional expert-designed representations [111] have been fully replaced by learned representations [44]. Ideally, expert-knowledge and semantic concepts can be seamlessly incorporated into the learned representation and optimized jointly, potentially resulting in the best of both worlds.

ONLINE PATH PREDICTION

The online path prediction done in this thesis considered a single scenario at a time. Future work could extend the number of scenarios that the DBN can handle at once. This brings

along the complication of how to determine which scenario could potentially play out at a given time. One approach would be to encase all scenarios in the I-MoLDS from chapter 4, in which additional contextual cues inform which scenarios are more likely. Another suggestion is to extend the complexity of the given scenarios, for example jointly predict the future action of multiple agents as done in [34]. One interesting choice for the extended scenario would be to integrate potential oncoming traffic, which would make an evasive maneuver from the motion planner no longer the preferred action in every situation.

Furthermore, because of a difference in processing time, not all measurements and contextual cues arrived at the same moment during online path prediction. The method presented in this thesis assume that they arrive synchronously, so the resulting prediction pipeline was delayed based on the longest measurement processing time. Ideally, prediction methods process every separate measurement whenever it arrives, thereby giving the most accurate prediction it can as quickly as possible.

Finally, cross-domain adaptation was already mentioned in the section on 3D person detection. Evaluating a path prediction model online after optimizing it on an offline dataset is essentially the same cross-domain adaptation problem. There is an interesting difference, however: path prediction attempts to predict the future, and as such can retrieve ground truth labels for its own predictions after some time has passed. Future work on online path prediction could therefore go one step further than only evaluating the performance of a model online, and instead also improve it by adapting to observed behaviors.

ACKNOWLEDGEMENTS

This thesis, just as all others before it, was not possible without the support that I received in the past four years from all the people around me.

In particular, I want to express my gratitude to my advisor and promotor, prof. dr. Dariu M. Gavrila. Dariu, thank you for all your advice on my thesis, the topics therein, and the feedback you have given me over the last four years. Your relentless drive for quality has ensured that I am as proud as I could have been of what we achieved together. You have been my strictest reviewer, and that is all I could have wished for.

Next, I want to thank my copromotor, dr. Julian F.P. Kooij. Julian, thank you for your help, input, and feedback. You have been one of the main driving forces behind my research, and for that, I am very grateful. I look back with fond memories on the many interesting discussions I have had with you, about my research and otherwise. I also have fond memories of the long days we made together to finish that first paper, and I hope one day I will be able to finish one before midnight.

I further want to thank the committee members, prof. dr. Robert Babuska, prof. dr. Miguel Á. Sotelo, prof. dr. Arnaud de La Fortelle, dr. Gijs Dubbelman and prof. dr. ir. Martijn Wisse for their feedback and for joining me on the final part of this journey.

Working on this PhD. has been an amazing journey, thanks to all my colleagues, students, and the staff at the university. András Pálffy and Joris Domhof, both of you have been with me in our office for almost the entire ride, and I have enjoyed every minute of it. Thomas Hehn, Tugrul Irmak, Yanggu Zheng, Jork Stapel, Frank Everdij, Ronald Ensing, Tom Dalhuisen, Zimin Xia, Joram van der Sluis, Abhishek Tomy, Hanneke Hustinx, Karin van Tongeren, dr. Laura Ferranti, dr. Barys Shyrokau, dr. Riender Happee, thank you for all the interesting conversations, especially during lunch and with some of you while bouldering as well. I hope you all have or find someone who looks at you the way András looks at his hamburger. Bruno Brito, Julian Wiederer, Sebastian Krebs, and especially dr. Fabian Flohr, I immensely enjoyed our time working on the intelligent vehicles near Barcelona and Paris. Finishing those projects together with you perfectly on time are memories I will cherish forever. Finally, thanks to all the other colleagues at Daimler for the inspiring discussions during our bi-yearly workshops and the occasional snowboarding advice: Markus Roth, Markus Braun, Christoph Rist, and Christian Münch.

I wish to further thank my closest friend, Floris Aartsen, for being there with me for literally longer than I can remember. I never expected that someday we would both be a doctor. Additionally, I want to thank the collective Ducktape, who helped me to keep everything together. I have enjoyed our trips, holidays, activities, and the amazing friendships that have grown from it. You all made sure I could unwind and let go of my research when I needed it, even on a Tuesday. The same gratitude goes out to all my other friends who have supported me over the past years.

I wish to thank my parents and my brothers for everything they have done for me. More than anyone, you have prepared me to finish this PhD. by being my teachers and by being a

role model for me each in your own way. And of course, I could not have done this without you Thera. You have always encouraged me throughout this journey and supported me when I needed it. I am grateful I had you by my side during my PhD. and I look forward to everything that the future will bring us.

REFERENCES

[1] World Health Organization, "Road traffic injuries," (2018), https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[2] L. Ferranti, B. Brito, E. A. I. Pool, Y. Zheng, R. M. Ensing, R. Happee, B. Shyrokau, J. F. P. Kooij, J. Alonso-Mora, and D. M. Gavrila, "SafeVRU: A research platform for the interaction of self-driving vehicles with vulnerable road users," in *IEEE Intelligent Vehicles Symposium (IV)* (2019) pp. 1660–1666.

[3] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making Bertha Drive–An Autonomous Journey on a Historic Route," in *IEEE Intelligent Transportation Systems Magazine*, 2 (2014) pp. 8–20.

[4] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) pp. 2446–2454.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," in *IEEE Transactions on Intelligent Vehicles*, Vol. 1 (2016) pp. 33–55.

[6] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," in *International Journal of Robotics Research (IJRR)*, 8 (2020) pp. 895–935.

[7] J. F. P. Kooij, *Generative models for pedestrian track analysis*, PhD dissertation, Universiteit van Amsterdam (2015).

[8] N. Schneider and D. M. Gavrila, "Pedestrian path prediction with recursive Bayesian filters: A comparative study," in *Proceedings of the German Conference of Pattern Recognition* (Springer Berlin Heidelberg, 2013) pp. 174–183.

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

[10] J. F. P. Kooij, N. Schneider, and D. M. Gavrila, "Analysis of pedestrian dynamics from a vehicle perspective," in *IEEE Intelligent Vehicles Symposium (IV)* (2014) pp. 1445–1450.

[11] B. Völz, H. Mielenz, R. Siegwart, and J. Nieto, "Predicting pedestrian crossing using quantile regression forests," in *IEEE Intelligent Vehicles Symposium (IV)* (2016) pp. 426–432.

[12] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 4 (2012) pp. 743–761.

[13] X. Li, F. Flohr, Y. Yang, H. Xiong, M. Braun, S. Pan, K. Li, and D. M. Gavrila, "A new benchmark for vision-based cyclist detection," in *IEEE Intelligent Vehicles Symposium (IV)* (2016) pp. 1028–1033.

[14] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "EuroCity Persons: A novel benchmark for person detection in automotive context," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8 (2019) pp. 1844–1861.

[15] C. Banz, H. Blume, and P. Pirsch, "Real-time semi-global matching disparity estimation on the GPU," in *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (2011) pp. 514–521.

[16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019) pp. 12697–12705.

[17] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in *IEEE International Conference on Intelligent Robots and Systems (IROS)* (2018) pp. 5750–5757.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[19] M. Braun, S. Krebs, and D. M. Gavrila, "ECP2.5D - Person localization in traffic scenes," in *IEEE Intelligent Vehicles Symposium (IV)* (2020).

[20] J. F. P. Kooij, F. Flohr, E. A. I. Pool, and D. M. Gavrila, "Context-Based Path Prediction for Targets with Switching Dynamics," in *International Journal of Computer Vision (IJCV)* (2019) pp. 239–262.

[21] K. Saleh, M. Hossny, and S. Nahavandi, "Cyclist trajectory prediction using bidirectional recurrent neural networks," in *Australasian Joint Conference on Artificial Intelligence* (Springer, 2018) pp. 284–295.

[22] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Context-based cyclist path prediction using recurrent neural networks," in *IEEE Intelligent Vehicles Symposium (IV)* (2019) pp. 824–830.

[23] I. Batkovic, M. Zanon, N. Lubbe, and P. Falcone, "A computationally efficient model for pedestrian motion prediction," in *IEEE European Control Conference (ECC)* (2018) pp. 374–379.

[24] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," in *IEEE Transactions on Intelligent Vehicles*, 1 (2016) pp. 90–104.

[25] D. Ridel, E. Rehder, M. Lauer, C. Stiller, and D. Wolf, "A literature review on the prediction of pedestrian behavior in urban scenarios," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2018) pp. 3105–3112.

[26] A. Masalov, J. Ota, H. Corbet, E. Lee, and A. Pelley, "CyDet : Improving camera-based cyclist recognition accuracy with known cycling jersey patterns," in *IEEE Intelligent Vehicles Symposium (IV)* (2018) pp. 2143–2149.

[27] A. Palffy, J. Dong, J. F. P. Kooij, and D. M. Gavrila, "CNN based road user detection using the 3d radar cube," in *IEEE Robotics and Automation Letters (RAL)*, 2 (2020) pp. 1263–1270.

[28] C. G. Keller, T. Dang, H. Fritz, A. Joos, C. Rabe, and D. M. Gavrila, "Active pedestrian safety by automatic braking and evasive steering," in *IEEE Transactions on Intelligent Transportation Systems*, 4 (2011) pp. 1292–1304.

[29] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018) pp. 918–927.

[30] "VRU Trajectory Dataset," `https://www.th-ab.de/vru-trajectory-dataset`.

[31] L. Ballan, F. Castaldo, A. Alahi, F. Palmieri, and S. Savarese, "Knowledge transfer for scene-specific motion prediction," in *European Conference on Computer Vision (ECCV)* (2016) pp. 697–713.

[32] M. F. Chang *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[33] A. T. Schulz and R. Stiefelhagen, "A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2015) pp. 173–178.

[34] M. Roth, F. Flohr, and D. M. Gavrila, "Driver and pedestrian awareness-based collision risk analysis," in *IEEE Intelligent Vehicles Symposium (IV)* (2016) pp. 454–459.

[35] Y. Hashimoto, G. Yanlei, L.-T. Hsu, and K. Shunsuke, "A probabilistic model for the estimation of pedestrian crossing behavior at signalized intersections," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2015).

[36] R. Quintero, I. Parra, D. F. Llorca, and M. Sotelo, "Pedestrian intention and pose prediction through dynamical models and behaviour classification," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2015) pp. 83–88.

[37] B. Volz, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto, "Inferring Pedestrian Motions at Urban Crosswalks," in *IEEE Transactions on Intelligent Transportation Systems* (2018) pp. 544–555.

[38] A. Graves, "Generating sequences with recurrent neural networks," in *arXiv preprint arXiv:1308.0850* (2013).

[39] S. Becker, R. Hug, W. Hübner, and M. Arens, "An RNN-based IMM filter surrogate," in *Scandinavian Conference on Image Analysis* (Springer, 2019) pp. 387–398.

[40] P. Ondrúška and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *AAAI Conference on Artificial Intelligence* (2016).

[41] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Conference on Neural Information Processing Systems (NIPS)* (2017) pp. 3601–3610.

[42] Y. Li, X.-Y. Lu, J. Wang, and K. Li, "Pedestrian trajectory prediction combining probabilistic reasoning and sequence learning," in *IEEE Transactions on Intelligent Vehicles* (2020).

[43] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "DESIRE: Distant future prediction in dynamic scenes with interacting agents," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017) pp. 336–345.

[44] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 961–971.

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Neural Information Processing Systems (NIPS) Workshop on Deep Learning* (2014).

[46] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision (ECCV)* (2012) pp. 201–214.

[47] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *IEEE International Conference on Robotics and Automation (ICRA)* (2016) pp. 2543–2549.

[48] E. Rehder, F. Wirth, M. Lauer, and C. Stiller, "Pedestrian prediction by planning using deep neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)* (2018) pp. 1–5.

[49] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *IEEE International Conference on Computer Vision (ICCV) Workshops* (2015) pp. 50–58.

[50] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *European Conference on Computer Vision (ECCV)* (Springer, 2016) pp. 549–565.

[51] T. Gandhi and M. M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2008) pp. 506–511.

[52] H. Xiong, F. B. Flohr, S. Wang, B. Wang, J. Wang, and K. Li, "Recurrent neural network architectures for vulnerable road user trajectory prediction," in *IEEE Intelligent Vehicles Symposium (IV)* (2019) pp. 171–178.

[53] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 779–788.

[54] J. F. Kooij, G. Englebienne, and D. M. Gavrila, "Mixture of switching linear dynamics to discover behavior patterns in object tracks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2 (2015) pp. 322–334.

[55] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 11 (2011) pp. 2287–2301.

[56] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3d traffic scene understanding from movable platforms," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 5 (2014) pp. 1012–1025.

[57] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2016) pp. 2267–2272.

[58] S. Zernetsch, S. Kohnen, M. Goldhammer, K. Doll, and B. Sick, "Trajectory prediction of cyclists using a physical model and an artificial neural network," in *IEEE Intelligent Vehicles Symposium (IV)* (2016) pp. 833–838.

[59] L. Huang and J. Wu, "Cyclists' path planning behavioral model at unsignalized mixed traffic intersections in china," in *IEEE Transactions on Intelligent Transportation Systems*, 2 (2009) pp. 13–19.

[60] S. Huang, X. Li, Z. Zhang, Z. He, F. Wu, W. Liu, J. Tang, and Y. Zhuang, "Deep learning driven visual path prediction from a single image," in *IEEE Transactions on Image Processing*, 12 (2016) pp. 5892–5904.

[61] K. Saleh, M. Hossny, and S. Nahavandi, "Contextual recurrent predictive model for long-term intent prediction of vulnerable road users," in *IEEE Transactions on Intelligent Transportation Systems* (2019).

[62] G. Raipuria, F. Gaisser, and P. P. Jonker, "Road infrastructure indicators for trajectory prediction," in *IEEE Intelligent Vehicles Symposium (IV)* (2018) pp. 537–543.

[63] S. Neogi, M. Hoy, K. Dang, H. Yu, and J. Dauwels, "Context model for pedestrian intention prediction using factored latent-dynamic conditional random fields," in *IEEE Transactions on Intelligent Transportation Systems* (2020).

[64] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *IEEE International Conference on Computer Vision (ICCV)* (2009) pp. 261–268.

[65] O. Ghori, R. Mackowiak, M. Bautista, N. Beuter, L. Drumond, and F. Diego, "Learning to Forecast Pedestrian Intention from Pose Dynamics," in *IEEE Intelligent Vehicles Symposium (IV)* (2018) pp. 1277–1284.

[66] S. Zernetsch, V. Kress, B. Sick, and K. Doll, "Early start intention detection of cyclists using motion history images and a deep residual network," in *IEEE Intelligent Vehicles Symposium (IV)* (2018) pp. 1–6.

[67] K. Saleh, M. Hossny, and S. Nahavandi, "Real-time intent prediction of pedestrians for autonomous ground vehicles via spatio-temporal densenet," in *IEEE International Conference on Robotics and Automation (ICRA)* (2019) pp. 9704–9710.

[68] I. Cara and E. de Gelder, "Classification for safety-critical car-cyclist scenarios using machine learning," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2015) pp. 1995–2000.

[69] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Conference on Neural Information Processing Systems (NIPS)* (2017).

[70] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," in *USENIX Symp. on Operating Systems Design and Implementation* (2016).

[71] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *International Journal of Computer Vision (IJCV)* (2017) pp. 618–626.

[72] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[73] N. X. Vinh, M. Chetty, R. Coppel, and P. P. Wangikar, "GlobalMIT: learning globally optimal dynamic bayesian network with the mutual information test criterion," in *Bioinformatics*, 19 (Oxford University Press, 2011) pp. 2765–2766.

[74] K. P. Murphy, "Switching Kalman filters," in *Technical report, Department of Computer Science, UC Berkeley* (Citeseer, 1998).

[75] K. P. Murphy, *Dynamic bayesian networks: representation, inference and learning*, PhD. thesis, University of California, Berkeley (2002).

[76] A. L. Rosado, S. Chien, L. Li, Q. Yi, Y. Chen, and R. Sherony, "Certainty and critical speed for decision making in tests of pedestrian automatic emergency braking systems," in *IEEE Transactions on Intelligent Transportation Systems*, 6 (2017) pp. 1358–1370.

[77] T. Dang, J. Desens, U. Franke, D. Gavrila, L. Schäfers, and W. Ziegler, "Steering and evasion assist," in *Handbook of intelligent vehicles* (Springer, 2012) pp. 759–782.

[78] S. Köhler, B. Schreiner, S. Ronalter, K. Doll, U. Brunsmann, and K. Zindler, "Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions," in *IEEE Intelligent Vehicles Symposium (IV)* (2013) pp. 519–526.

[79] J. R. Van der Sluis, E. A. I. Pool, and D. M. Gavrila, "An experimental study on 3D person localization in traffic scenes," in *IEEE Intelligent Vehicles Symposium (IV)* (2020).

[80] A. Naiden, V. Paunescu, G. Kim, B. Jeon, and M. Leordeanu, "Shift R-CNN: Deep monocular 3D object detection with closed-form geometric constraints," in *Proc. of the ICIP* (2019) pp. 61–65.

[81] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proceedings of the Conference on Robot Learning (CoRL)*, Vol. 87 (2018) pp. 146–155.

[82] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[83] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[84] Z. Wang, H. Fu, L. Wang, L. Xiao, and B. Dai, "SCNet: Subdivision coding network for object detection based on 3D point cloud," in *IEEE Access*, Vol. 7 (2019) pp. 120449–120462.

[85] V. A. Sindagi, Y. Zhou, and O. Tuzel, "MVX-Net: Multimodal voxelnet for 3D object detection," in *IEEE International Conference on Robotics and Automation (ICRA)* (2019) pp. 7276–7282.

[86] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[87] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[88] M. Roth, D. Jargot, and D. M. Gavrila, "Deep end-to-end 3D person detection from camera and lidar," in *IEEE International Conference on Intelligent Transport Systems (ITSC)* (2019) pp. 521–527.

[89] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Multimodal vehicle detection: fusing 3D-LIDAR and color camera data," in *Pattern Recognition Letters*, Vol. 115 (2018) pp. 20–29.

[90] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," in *IEEE International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019).

[91] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *IEEE International Conference on Computer Vision (ICCV)* (2019).

[92] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) pp. 4604–4612.

[93] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) pp. 11621–11631.

[94] R. Kesten *et al.*, "Lyft level 5 AV dataset 2019," `https://level5.lyft.com/dataset/` (2019).

[95] D. Zhou *et al.*, "IoU loss for 2D/3D object detection," in *International Conference on 3D Vision* (2019) pp. 85–94.

[96] A. Simonelli, S. R. Bulo, L. Porzi, M. L. Antequera, and P. Kontschieder, "Disentangling monocular 3D object detection: From single to multi-class recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2020).

[97] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Using road topology to improve cyclist path prediction," in *IEEE Intelligent Vehicles Symposium (IV)* (2017) pp. 289–296.

[98] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)* (2011).

[99] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Crafted vs. learned representations in predictive models: A case study on cyclist path prediction," in *IEEE Transactions on Intelligent Vehicles* (2021).

[100] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," in *Journal of the Royal Statistical Society* (1977) pp. 1–38.

[101] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *International Conference on Learning Representations* (2018).

[102] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *IEEE Conference on Decision and Control* (2009) pp. 8642–8647.

[103] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," in *IEEE Transactions on Intelligent Transportation Systems*, 99 (2017) pp. 1–15.

[104] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation (ICRA)* (2009).

[105] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the Robot Operating System," in *Intelligent Autonomous Systems* (2016) pp. 335–348.

[106] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, in *Lecture Notes in Computer Science*, Vol. 9905 LNCS (2016) pp. 21–37.

[107] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017) pp. 1302–1310.

[108] M. Braun, S. Krebs, F. B. Flohr, and D. M. Gavrila, "EuroCity Persons: A novel benchmark for person detection in traffic scenes," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8 (2019) pp. 1844–1861.

[109] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) pp. 14074–14083.

[110] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, "Stgat: Modeling spatial-temporal interactions for human trajectory prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019) pp. 6272–6281.

[111] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *IEEE International Conference on Robotics and Automation (ICRA)* (2010) pp. 464–469.

CURRICULUM VITÆ

Ewoud Alexander Ignacz POOL

SEPT 15 1989	Born in Amsterdam, the Netherlands.

EDUCATION

2016-PRESENT	PhD. at COGNITIVE ROBOTICS, **Delft University of Technology** Thesis: "Context-based Cyclist Path Prediction: Crafted and Learned Models for Intelligent Vehicles".
2012-2015	MSc. in SYSTEMS & CONTROL, **Delft University of Technology** Thesis: "Optimizing the Uncertainty Bounds for a Robust Control Problem using Moving Horizon Estimation" *Graduated Cum Laude*.
2008-2012	BSc. in MECHANICAL ENGINEERING, **Delft University of Technology** Minor: Computer Engineering and Embedded Systems.

PROFESSIONAL EXPERIENCE

2015-2016	Control Engineer at HOUDIJK, Vlaardingen Designed, programmed and commissioned Programmable Logic Controllers (PLCs) for machinery that handles biscuits and similar frail products.
2011-2015	Android programmer at EVECT, Delft Developed and created Android applications for third parties based on their functional requirements and specifications, both individually and in a small team.

List of Publications

7. ***E. A. I. Pool*, J. R. van der Sluis, D. M. Gavrila**, *An Experimental Study on 3D Person Localization in Traffic Scenes*, To be submitted to IEEE Transactions on Intelligent Vehicles (2021).
Author contributions: Equal share of authorship between first two authors. *E. A. I. Pool* and J. R. van der Sluis jointly designed and performed experiments. D. M. Gavrila provided guidance, supervision, and contributed to writing.

6. ***E. A. I. Pool*, J. F. P. Kooij, D. M. Gavrila**, *Crafted vs. Learned Representations in Predictive Models: a Case Study on Cyclist Path Prediction*, IEEE Transactions on Intelligent Vehicles (2021). DOI: 10.1109/TIV.2021.3064253.
Author contributions: E. A. I. Pool created proposed model and performed experiments. J. F. P. Kooij and D. M. Gavrila provided guidance, supervision, and contributed to writing.

5. **J. R. van der Sluis, *E. A. I. Pool*, D. M. Gavrila**, *An Experimental Study on 3D Person Localization in Traffic Scenes*, IEEE Intelligent Vehicles Symposium (2020).
Author contributions: Equal share of authorship between first two authors. J. R. van der Sluis and *E. A. I. Pool* jointly designed and performed experiments. D. M. Gavrila provided guidance, supervision, and contributed to writing.

4. **L. Ferranti, B. Brito, *E. A. I. Pool*, Y. Zheng, R. M. Ensing, R. Happee, B. Shyrokau, J. F. P. Kooij, J. Alonso-Mora, D. M. Gavrila**, *SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users*, IEEE Intelligent Vehicles Symposium (2019).
Author contributions: Equal share of authorship between first three authors. *E. A. I. Pool* implemented prediction module and aided with the implementation of other modules and writing.

3. ***E. A. I. Pool*, J. F. P. Kooij, D. M. Gavrila**, *Context-based Cyclist Path Prediction using Recurrent Neural Networks*, IEEE Intelligent Vehicles Symposium (2019).
Author contributions: E. A. I. Pool created proposed model and performed experiments. J. F. P. Kooij and D. M. Gavrila provided guidance and supervision.

2. **J. F. P. Kooij, F. Flohr, *E. A. I. Pool*, D. M. Gavrila**, *Context-based Path Prediction for Targets with Switching Dynamics*, International Journal of Computer Vision **127.3** (2019).
Author contributions: J. F. P. Kooij created proposed model and performed pedestrian experiments. F. Flohr planned and performed data acquisition and annotation, and worked on visual VRU feature extraction. *E. A. I. Pool* developed cyclist model and performed cyclist experiments. D. M. Gavrila provided guidance and supervision.

1. ***E. A. I. Pool*, J. F. P. Kooij, D. M. Gavrila**, *Using Road Topology to Improve Cyclist Path Prediction*, IEEE Intelligent Vehicles Symposium (2017).
Author contributions: E. A. I. Pool created proposed model, extracted trajectories and performed experiments. J. F. P. Kooij provided guidance, supervision, and contributed to writing. D. M. Gavrila provided guidance and supervision.