

From
CR/AVS3Our Reference
Alessandro MoiaTel
+49 711 811-43672Renningen
25 November 2021
Report Number**R&D Report: Final Report**

Security Class:	Internal	Export control relevant:	No
Title:	ICT-006 - eBike Safe Connect		

1. Issues (situation, motivation and tasks)

The advent of 5G and internet of things creates many new opportunities for the development of innovative safety and comfort functions for eBikes.

There are several traffic scenarios where a collision between an eBike and a motorized vehicle can only be avoided with the implementation of vehicle2vehicle communication.

Additionally, the communication between the eBikes and other communication partners (road side units, other VRUs and vehicles, map providers, Bosch cloud) enables the realization of new functionalities and services with high customer benefit.

2. Results

The most relevant accident scenarios have been analyzed in order to identify the potential of Bike2Vehicle communication in the reduction of collisions. The potential of a bidirectional warning has been derived as well as the impact on uncertainties in the prediction of the future trajectory on the false/positive rate of the warning system.

Additionally, a proof of concept has been realized in form of an eBike and a car with communication capabilities. With help of the developed prototypes, several accident scenarios have been recreated to support the development and implementation of the bidirectional warning system. Particular effort has been invested in the improvement of localization hardware and trajectory prediction.

3. Conclusions and Consequences

The analysis showed the high potential of bidirectional warnings in the relevant scenarios as well as the central role of localization and trajectory prediction accuracy in defining the benefit and feasibility of the system.

In 2022 the activity will be continued with a strong focus on improvement of localization and trajectory prediction accuracy. The scope of the activity will be extended to also comprise earlier warnings for the rider as well as information to increase the rider's awareness of the surroundings, since these functionalities have less strict requirements for the system and enable a shorter time to market.

From
CR/AVS3Our Reference
Alessandro MoiaTel
+49 711 811-43672Renningen
25 November 2021
Report Number**R&D Report: Final Report**

Security Class: Internal Export control relevant: No
 Title: ICT-006 - eBike Safe Connect

Document approval			
	Org.-Unit	Name	Signature
Author(s):	CR/AVS3	Alessandro Moia	electronic release (FEBER)
Responsible person:	CR/AVS3	Alessandro Moia	electronic release (FEBER)
Reviewer:	CR/AEC1 CE-MCS	Hofmann Frank	electronic release (FEBER)
Releasing person:	CR/AEC1 CE-MCS	Hofmann Frank	electronic release (FEBER)
			electronic release (FEBER)
			electronic release (FEBER)

		External release of cover sheet:	No
Enclosure(s):	Yes	External release of enclosure(s):	No

Initial mailing list**Recipients:****Cc:**

Key Words

Enclosures	
Enclosure number	Title

Underlying documents		please link documents	
Document number	Title	Date	Responsible person

This report invalidates			
Document number	Title	Date	Responsible person

DEFINED SCENARIOS

TNO accident scenarios

CATS¹ Accident scenario definition

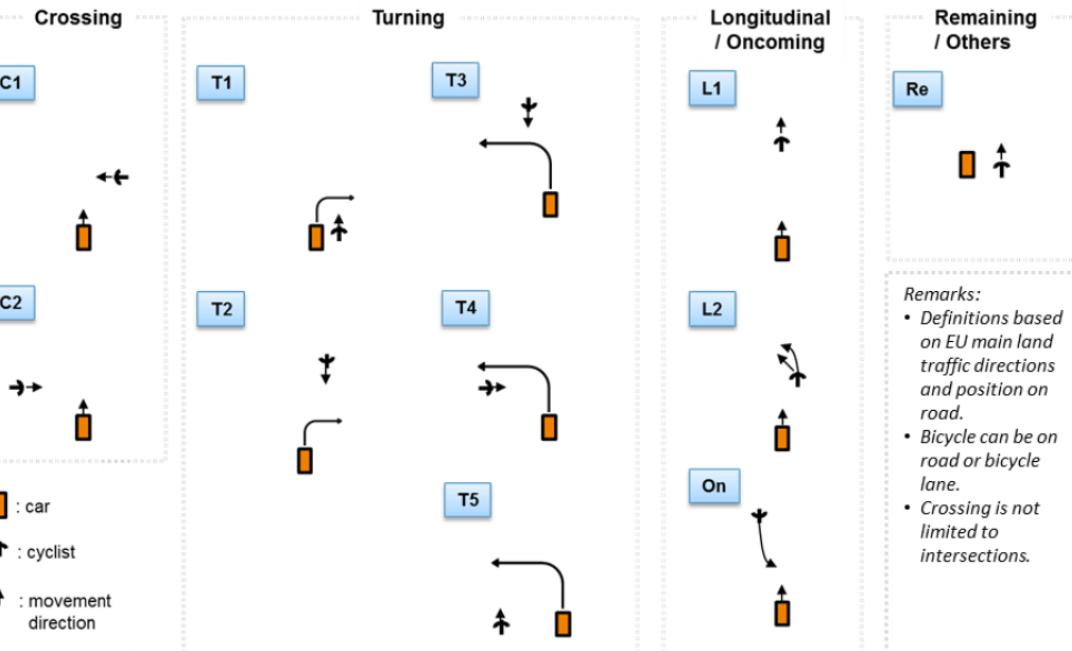


Figure 6: Overview of distinguished car-to-cyclist accident scenarios

Scenario	Description
C1	<ul style="list-style-type: none"> Car driving straight Cyclist crossing the vehicle path from the near side
C2	<ul style="list-style-type: none"> Car driving straight Cyclist crossing the vehicle path from the far side
T1	<ul style="list-style-type: none"> Car turning right Cyclist is riding straight in the same direction as the car before turning Blind spot scenario
T2	<ul style="list-style-type: none"> Car turning right Cyclist is riding straight in the opposite direction of the car before turning
T3	<ul style="list-style-type: none"> Car turning to the left, crossing the (straight) bicycle path Cyclist coming from the opposite direction, riding straight
T4	<ul style="list-style-type: none"> Car turning to the left, crossing the (straight) bicycle path Cyclist is riding straight, coming from the far side of the car. Some similarity with C2
T5	<ul style="list-style-type: none"> Car turning to the left, crossing the (straight) bicycle path Cyclist is riding straight in the same direction as the car before turning
L	<ul style="list-style-type: none"> Car and cyclist driving in the same direction Cyclist is riding straight and hit by the car from the rear
L1	<ul style="list-style-type: none"> Cyclist is swerving to the left in front of the car and hit by the car from the rear
L2	<ul style="list-style-type: none"> Cyclist is swerving to the left in front of the car and hit by the car from the rear
On	<ul style="list-style-type: none"> Car driving straight, driving towards the far road side in a passing manoeuvre Bicyclist coming in the opposite (on-coming) direction riding straight
Re	<ul style="list-style-type: none"> All other scenarios that are not covered by any of the previously described scenarios.

A road traffic accident data analysis has been performed covering 6 European countries: France (LAB), Germany (GIDAS), Italy, Netherlands (BRON), Sweden (STRADA) and the UK (STATS19). Accidents involving one bicycle and one M1 vehicle (passenger car) were selected.

TNO accident scenarios

CATS Fatalities and Injuries Distribution per Nation and Scenario

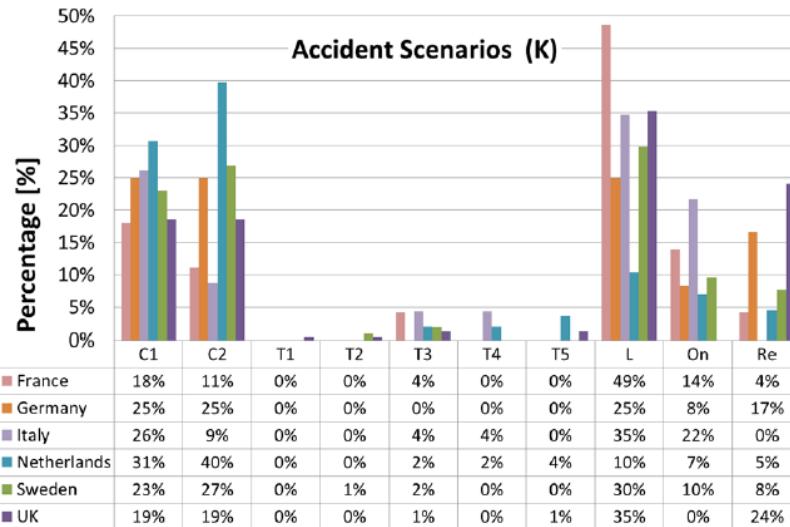


Figure 7: Distribution of fatally injured over the 9 accident scenarios that are distinguished for 6 EU countries.

Table 5: Weighting factors based on the ratio of cyclist fatalities and the total number of road fatalities per one million inhabitants in 2001-2010 [17]

Country	# road fatalities per million inhabitants	# cyclist fatalities per million inhabitants	Weighting [%]
France	62	2,8	11%
Germany	45	6,0	26%
Italy	68	5,4	-
Netherlands	32	9,2	38%
Sweden	28	3,6	15%
UK	30	2,3	10%

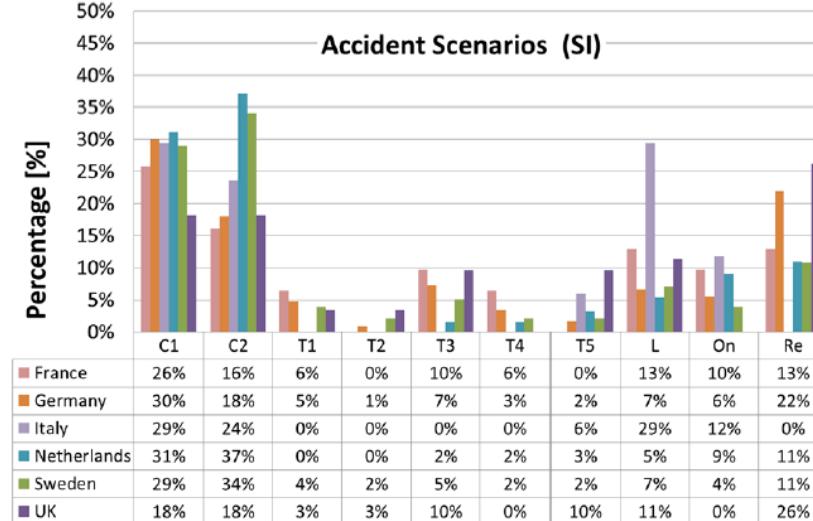


Figure 8: Distribution of seriously injured over the 9 accident scenarios that are distinguished for 6 EU countries.

A single percentage for each scenario resulted as weighting the percentages for the different countries to the number of cyclist fatalities per million inhabitants.

TNO accident scenarios

CATS Weighted Fatalities and Injuries Distribution per Scenario

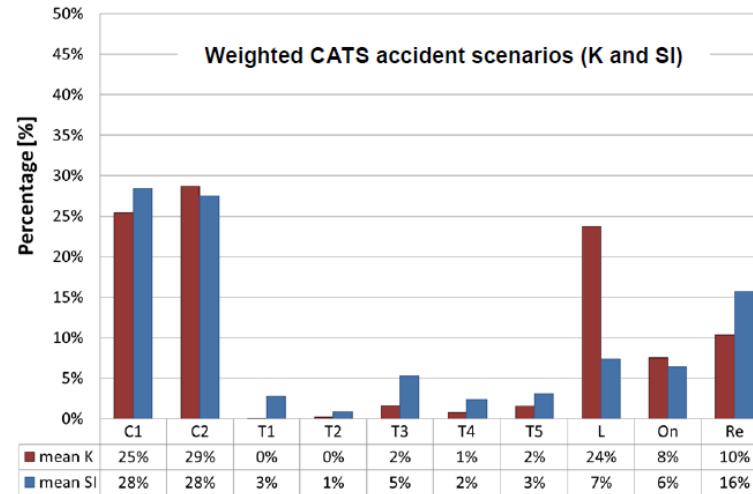


Figure 9: Distribution of fatalities and seriously injured over the 9 accident scenarios, weighted average over 5 countries. The red columns refer to fatalities (K), where the blue columns refer to seriously injured (SI).

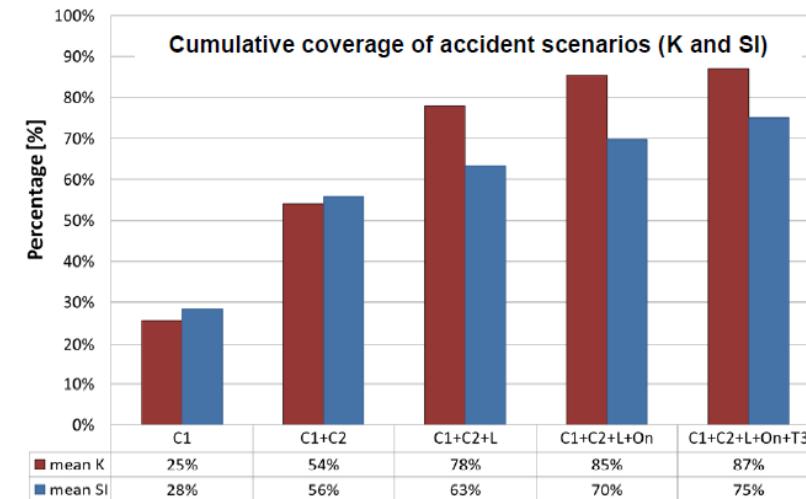


Figure 10: Cumulative coverage of scenarios in the order of importance.

CATS will focus on the top-3 accident scenarios: C1, C2 and L	
C1	<ul style="list-style-type: none"> • Car driving straight • Cyclist crossing the vehicle path from the near side
C2	<ul style="list-style-type: none"> • Car driving straight • Cyclist crossing the vehicle path from the far side
L	<ul style="list-style-type: none"> • Car and cyclist driving in the same direction
L1	<ul style="list-style-type: none"> • Cyclist is riding straight and hit by the car from the rear
L2	<ul style="list-style-type: none"> • Cyclist is swerving to the left in front of the car and hit by the car from the rear

Crossing scenarios and car from behind represent the major percentage of accident scenarios on which the CATS study focuses

Bike2V communication

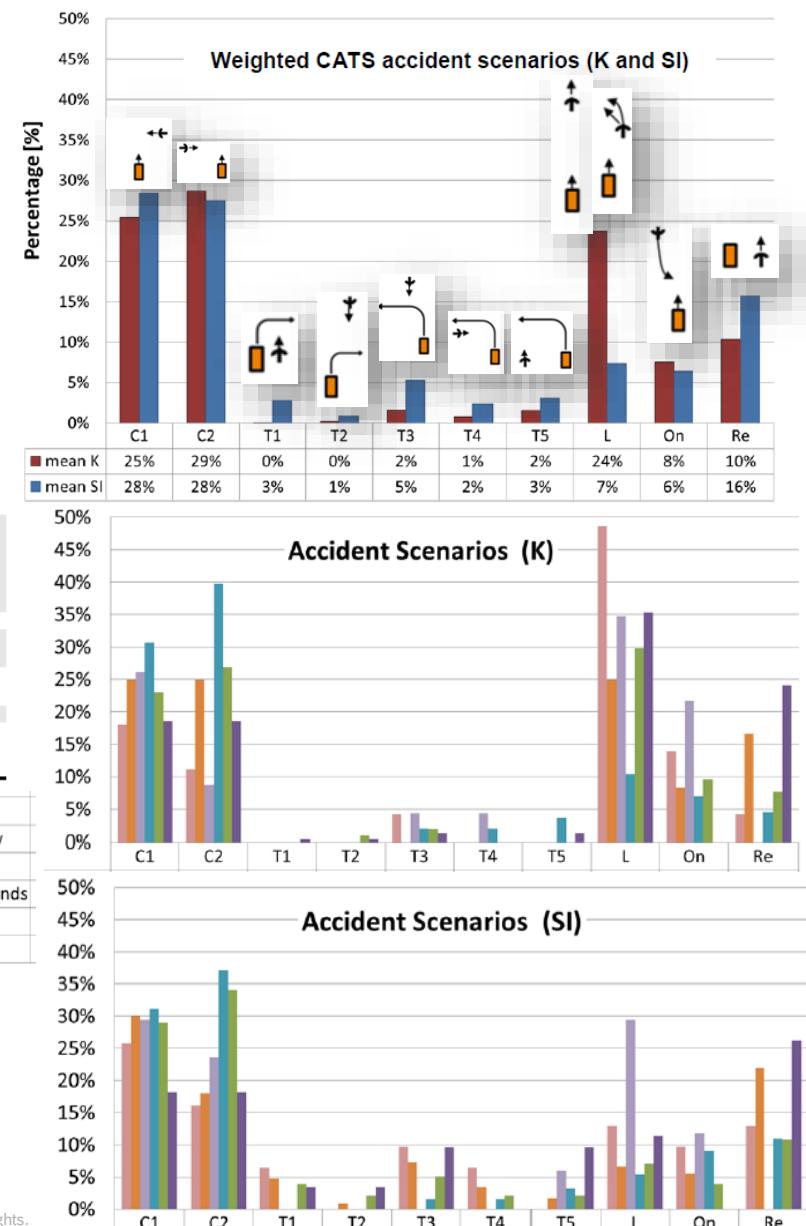
Motivation: accident situation

Name	Crossing cyclist	Oncoming cyclist	Lateral riding cyclist	Upfront riding cyclist
Description				
Actors	<ul style="list-style-type: none"> Cyclist Driver Visual obstructions (e.g. buildings) Other traffic participants 	<ul style="list-style-type: none"> Cyclist Driver Other traffic participants 	<ul style="list-style-type: none"> Cyclist Driver Visual obstructions (e.g. vehicle design) Other traffic participants 	<ul style="list-style-type: none"> Cyclist Driver
Includes	<ul style="list-style-type: none"> Cross road 	<ul style="list-style-type: none"> Turning 	<ul style="list-style-type: none"> Start of cycling 	<ul style="list-style-type: none"> Blind spot for cyclists Lane change Start of cycling Sudden braking by cyclist
Accident frequency*	↑ (36%)	↓ (10%)	↓ (6%)	↓ (1%)
Cyclist fears**	-	Top 1 (88%)	Top 2 (83%)	
Further use-cases (with minor accident frequency)	<ul style="list-style-type: none"> Parking: Vehicle exists parking lot alongside the road Dooring: Vehicle driver opens door after parking alongside the road (Top 3 of cyclist fears rated with 69%) 			

Sources: *Traffic accidents with injured participants weighted for Germany based on GIDAS in 2016 || ** Forsa survey by CosmosDirekt in 2017

Discrepancies → possibly because CATS only considers seriously injured (SI) and killed (K) and M2 vehicles

- C1 + C2 > Crossing cyclist
- T2 + T3 < Oncoming cyclist
- T1 < Lateral riding cyclist
- L >> Upfront riding cyclist



TNO accident scenarios

CATS Speed Distribution

Scenario	50 th percentile [km/h]	90 th percentile [km/h]
C1	~20	~50-55
C2	~20-25	~50-55
L	~40-45	~70-80

C1 C2 predominant in urban area

L predominant outside of urban area

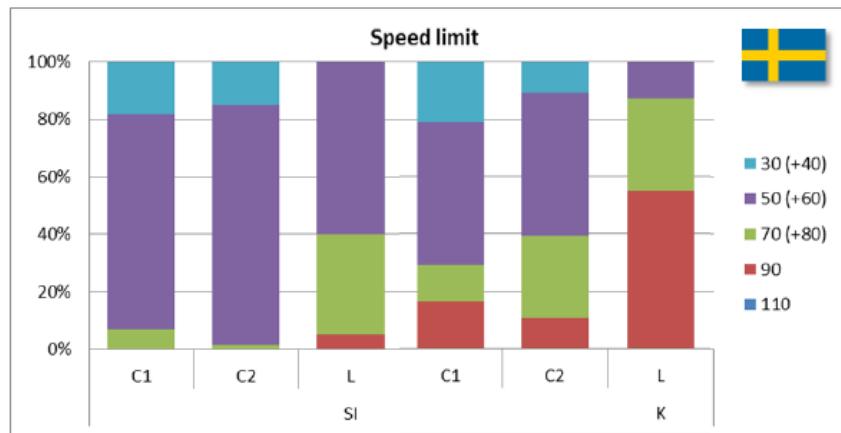


Figure 17: Overview of car-to-cyclist accidents by speed limit in Sweden, distinguished to accident scenario for accidents with seriously injured and fatal accidents.

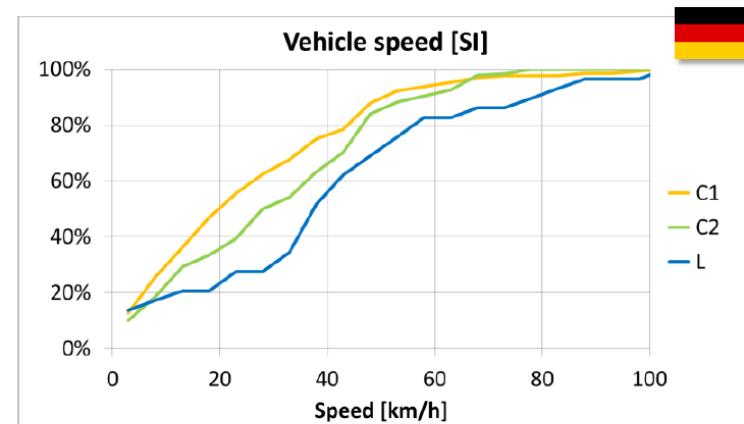


Figure 18: Cumulative vehicle speed distribution distinguished for the 3 accident scenarios in CATS, as found for accidents with seriously injured in Germany.

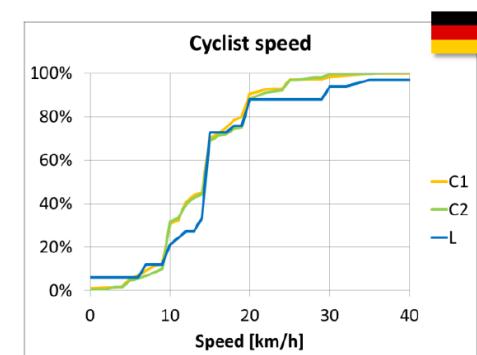
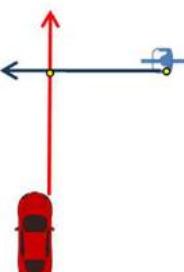
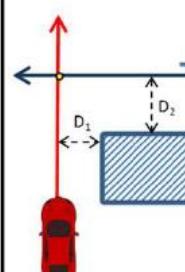
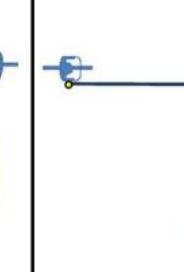
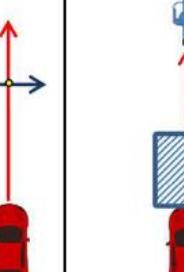
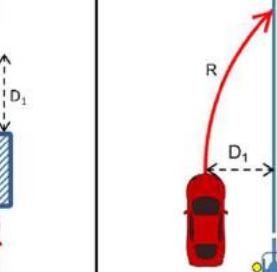


Figure 19: Cumulative bicycle speed distribution distinguished for the 3 accident scenarios in CATS, as found for fatal accidents and accidents with seriously injured in Germany.

The 50th and 90th percentile cyclist speed of both the seriously injured and fatal accidents is 12-15 km/h and 20-25 km/h respectively.

TNO accident scenarios

TNO AEB Study

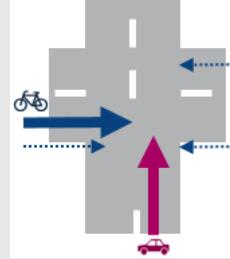
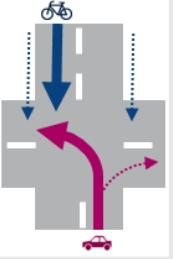
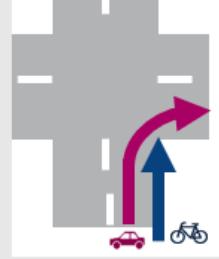
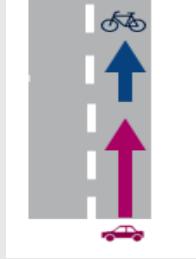
	CVNBU	CVNBO	CVFB	CVLBO	CVBB
	Car-to-VRU Nearside Bicyclist Unobstructed	Car-to-VRU Nearside Bicyclist Obstructed	Car-to-VRU Farside Bicyclist	Car-to-VRU Longitudinal Bicyclist Obstructed	Car-to-VRU Blindspot Bicyclist
Vehicle speed	5 - 60 km/h	5 - 50 km/h	5 - 60 km/h	20 - 80 km/h	5-30 km/h
Cyclist speed	15 km/h	10 km/h	20 km/h	15 km/h	10 km/h
Obstruction	-	D1 = 3.55m, D2 = 4.80m	-	D1 = 10.0m	-
Overlap hitpoint	50%	50%	25%		50% at 45° with D1=5.5m, R=10m
# tests	12	10	12	13	6
Coverage (K)	22%	22%	11%	5%	0%
Coverage (SI)	22%	22%	11%	1%	3%
Layout sketch					

- ▶ The blind spot scenario where the car turns to the nearside hitting a cyclist which is going straight on the nearside of the car do not necessary occur often, but induce the most fear in cyclists
- ▶ For that test scenario the vehicle speed and cyclist speed are taken from other turning scenarios. The layout is based on Dutch guidelines for the width of road layout designs

Figure 2-3. Proposed test scenarios for evaluation. Combined coverage is estimated at 60% for fatal accidents and 59% for accidents with seriously injured cyclists.

Definition of Considered Scenarios

Scenarios and Speed

Name	Crossing cyclist	Oncoming cyclist	Lateral riding cyclist	Upfront riding cyclist
Description				
Bicycle speed	10 - 25km/h	10 - 20km/h	10 - 20km/h	15 - 25km/h
Car speed	5 - 60km/h	5 - 30km/h	5 - 30km/h	20 - 80km/h
Comment	Car and bike speed from TNO study CVNBU, CVNBO CVFB. Bike Speed max increased from 20 to 25km/h	Car and bike speed from TNO study CVBB. Bike Speed max increased from 10 to 20km/h	Car and bike speed and geometric parameters from TNO study CVBB. Bike Speed max increased from 10 to 20km/h	Car and bike speed from TNO study CVLBO. Bike Speed max increased from 15 to 25km/h

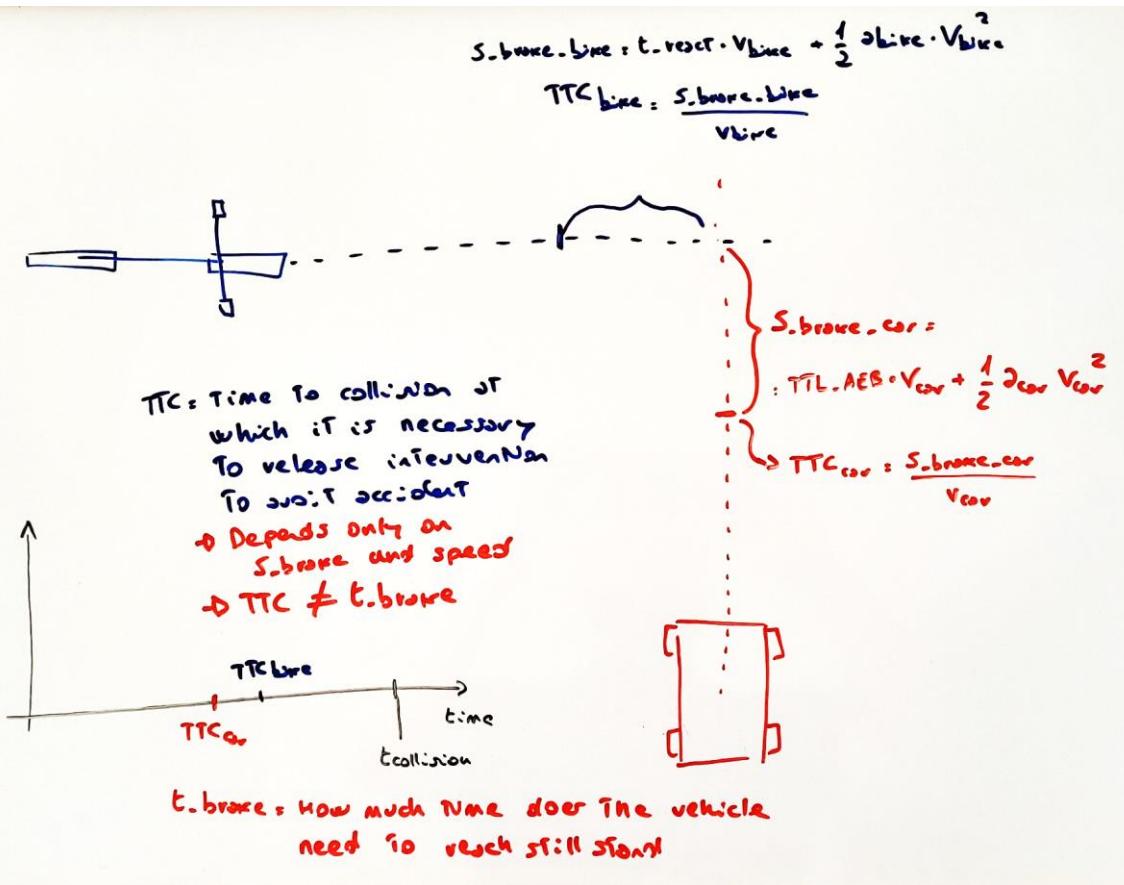
Scenarios defined based on CR accident research,
speed values from TNO CAT study

CROSSING CYCLIST

Potential of bicycle warning vs car warning

Coordination based on TTC

TTC vs t_brake



- Given an initial speed, the braking distance is calculated as:

$$S_{brake_car} = v_{car} \cdot TTL_{AEB} + \frac{1}{2} \frac{v_{car}^2}{a_{car}}$$

$$S_{brake_bike} = v_{bike} \cdot t_{react} + \frac{1}{2} \frac{v_{bike}^2}{a_{bike}}$$

- The current time to collision given the current speed and current position is:

$$t_{coll} = \frac{s}{v}$$

- The time to collision at a distance corresponding to the braking distance is

$$t_{coll_car@S_{brake}} = TTC_{car} = TTL_{AEB} + \frac{1}{2} \frac{v_{car}}{a_{car}}$$

$$t_{coll_bike@S_{brake}} = TTC_{bike} = t_{react} + \frac{1}{2} \frac{v_{bike}}{a_{bike}}$$

- This represents the latest time **before** intervention at which the intervention should be released to avoid the accident

- This is different from t_{brake} which is the time needed to reach still stand **after** the intervention has been activated

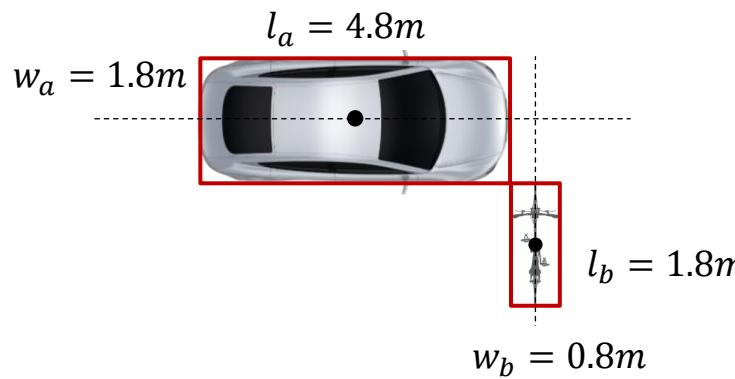
$$t_{brake_car} = TTL_{AEB} + \frac{v_{car}}{a_{car}}$$

$$t_{brake_bike} = t_{react} + \frac{v_{bike}}{a_{bike}}$$

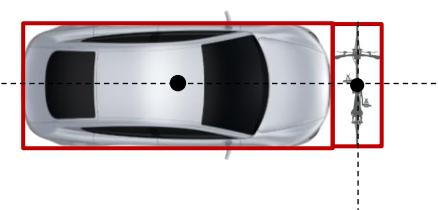
Coordination based on TTC Representation

- ▶ Goal: identify in which situations a warning for the eBike rider can improve his safety
- ▶ Intervention type: AEB on car, rider warning on bicycle
- ▶ Assumptions:
 - ▶ in order to minimize false negatives, the intervention should be activated for the vehicle with the shorter braking time $\min(TTC_{bike}, TTC_{car}) = \text{latest } TTC$ at which the intervention should be activated in order to avoid the accident
 - ▶ car AEB only activated when the accident is unavoidable. For situation where $TTC_{car} < TTC_{bike}$ then the accident is unavoidable by only activating AEB
- ▶ We consider the below three collision point scenarios

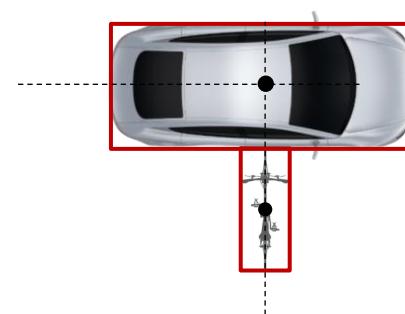
Case 1 - Minimum overlap



Case 2 – Car front



Case 3 – Car side



Coordination based on TTC Representation

- In order to consider the effect of path prediction accuracy and its usage in the coordination of the warnings we perform the simulation with 2 different parameter sets:

Scenario a: aggressive AEB strategy

- the eBike path prediction can predict braking 0.5s in advance and it is used by the AEB in the calculation of the t_{brake_bike} to release an earlier AEB intervention
- assumption: 0.5 is also the reaction time of the rider to a warning. This enables to consider t_{brake_bike} calculated by car AEB and TTC_{bike} at which the warning to the eBike rider is released equal to avoid warning overlap
- more conservative eBike deceleration and aggressive car deceleration

```
% bike_car_warning_coordination_based_on_TTC.m
% Simulation Parameters
t.react_bike = 0.5; %[s] reaction time of rider = prediction time for braking
ttl_AEB_car = 0.2; %[s] car time to lock
a_car = 9.8; %[m/s^2] deceleration of car during AEB braking
a_bike = 6; %[m/s^2] deceleration of bike after warning
```

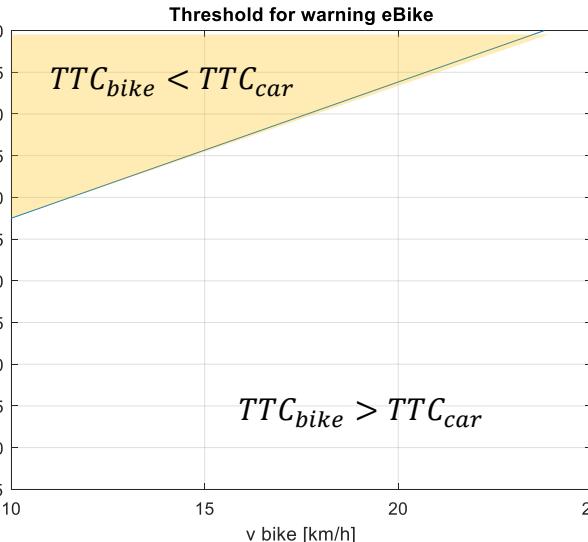
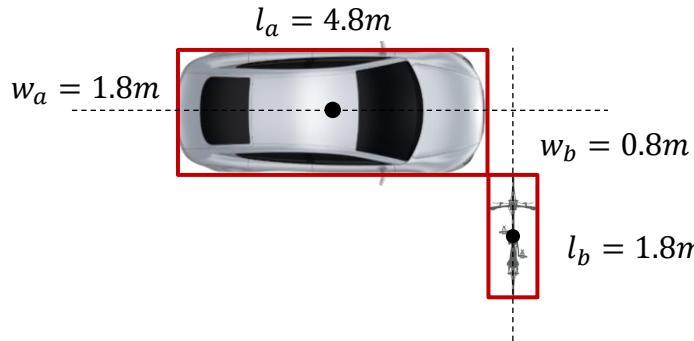
Scenario b: conservative AEB strategy

- the eBike path prediction is not used by the AEB logic in the calculation of t_{brake_bike} , the AEB logic calculates t_{brake_bike} only considering a deceleration of 7m/s^2
- more aggressive eBike deceleration and conservative car deceleration

```
% bike_car_warning_coordination_based_on_TTC.m
% Simulation Parameters
t.react_bike = 0; %[s] reaction time of rider = prediction time for braking
ttl_AEB_car = 0.2; %[s] car time to lock
a_car = 9; %[m/s^2] deceleration of car during AEB braking
a_bike = 7; %[m/s^2] deceleration of bike after warning
```

Coordination based on TTC

Case 1a – Minimum Overlap, aggressive AEB strategy



- The TTC at which the intervention should be activated only depends on the braking distance and current speed of both vehicles

$$s_{brake_car} = v_{car} * t_{tl_AEB_car} + 1/2 * v_{car}^2 / a_{car};$$
$$TTC_{car} = (s_{brake_car}) / v_{car}$$

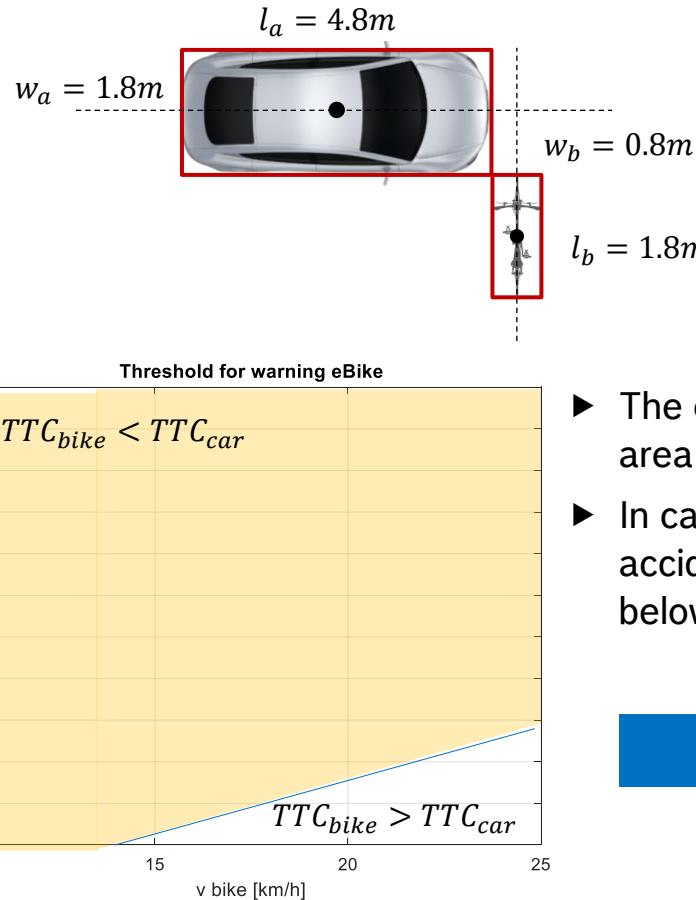
$$s_{brake_bike} = v_{bike} * t_{react_bike} + 1/2 * v_{bike}^2 / a_{bike};$$
$$TTC_{bike} = (s_{brake_bike}) / v_{bike}$$

- The orange area shows the speed range for which the bike has the lower TTC. In this area the accident cannot be avoided w/o rider warning
- In case the car AEB waits until the remaining time has fallen below TTC_{car} , then the accident can only be avoided by warning the eBike rider (until the remaining time falls below TTC_{bike})

Case 1a shows benefit in warning the eBike rider

Coordination based on TTC

Case 1b – Minimum Overlap, conservative AEB strategy



- The TTC at which the intervention should be activated only depends on the braking distance and current speed of both vehicles

$$s_{brake_car} = v_{car} * ttl_AEB_car + 1/2 * v_{car}^2 / a_{car};$$

$$TTC_{car} = (s_{brake_car}) / v_{car}$$

$$s_{brake_bike} = v_{bike} * t_{react_bike} + 1/2 * v_{bike}^2 / a_{bike};$$

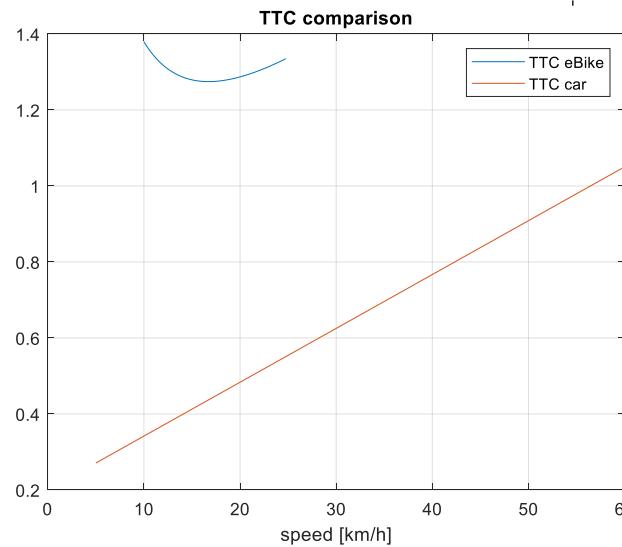
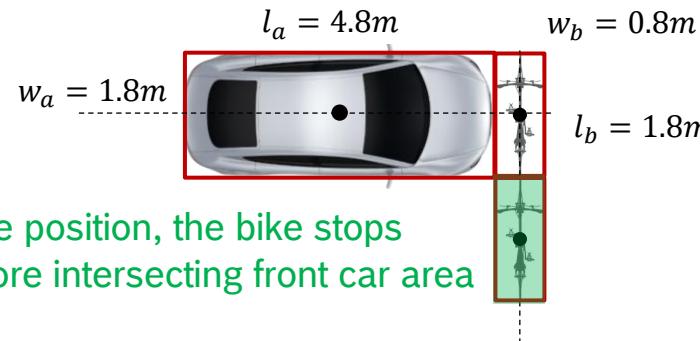
$$TTC_{bike} = (s_{brake_bike}) / v_{bike}$$

- The orange area shows the speed range for which the bike has the lower TTC. In this area the accident cannot be avoided w/o rider warning
- In case the car AEB waits until the remaining time has fallen below TTC_{car} , then the accident can only be avoided by warning the eBike rider (until the remaining time falls below TTC_{bike})

Case 1b shows very high benefit in warning the eBike rider

Coordination based on TTC

Case 2a – Car front middle



- ▶ TTC_{car} only depends on the braking distance and current speed
- ▶ TTC_{bike} depends on the braking distance and current speed, additionally on bike length and car width

$$s_{brake_car} = v_{car} * ttl_AEB_car + 1/2 * v_{car}^2 / a_{car};$$

$$TTC_car = (s_{brake_car}) / v_{car}$$

$$s_{brake_bike} = v_{bike} * t_{react_bike} + 1/2 * v_{bike}^2 / a_{bike};$$

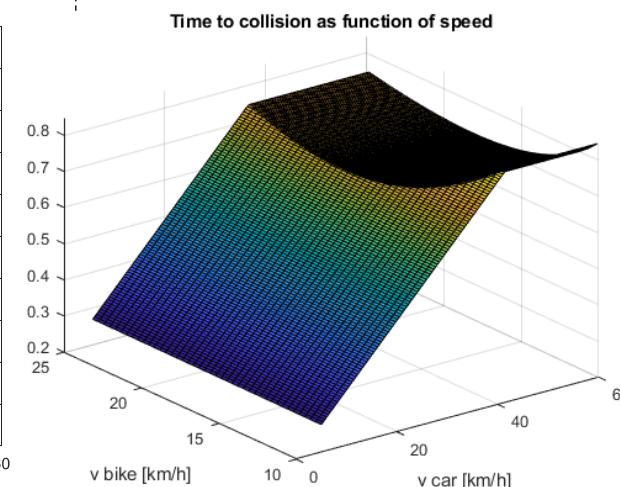
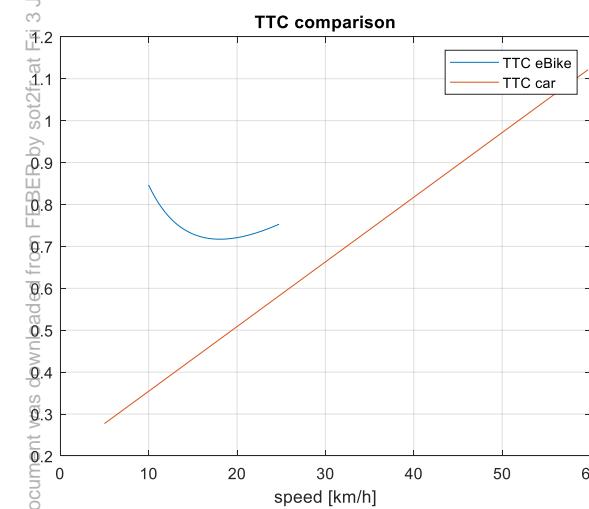
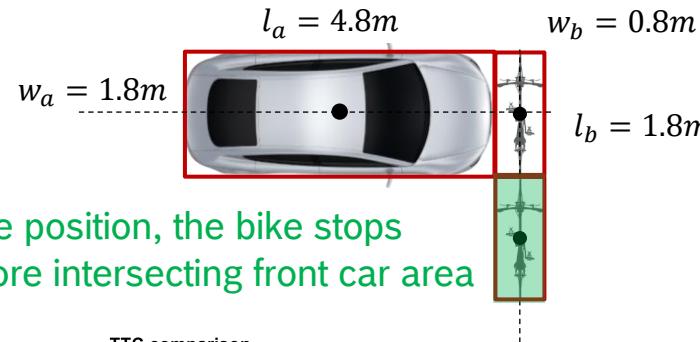
$$TTC_bike = (s_{brake_bike} + \underline{w_a/2 + l_b/2}) / v_{bike}$$

- ▶ In order to enable the bicycle to stop in the safe position, the bike warning must be anticipated for a value depending on vehicle dimensions
- ▶ For the given parameter and in the considered speed ranges $TTC_{car} < TTC_{bike}$

Case 2a shows no benefit in warning the eBike rider

Coordination based on TTC

Case 2b – Car front middle



- ▶ TTC_{car} only depends on the braking distance and current speed
- ▶ TTC_{bike} depends on the braking distance and current speed, additionally on bike length and car width

$$s_{\text{brake_car}} = v_{\text{car}} * \text{ttl_AEB_car} + 1/2 * v_{\text{car}}^2 / a_{\text{car}}$$

$$\text{TTC_car} = (s_{\text{brake_car}}) / v_{\text{car}}$$

$$s_{\text{brake_bike}} = v_{\text{bike}} * t_{\text{react_bike}} + 1/2 * v_{\text{bike}}^2 / a_{\text{bike}}$$

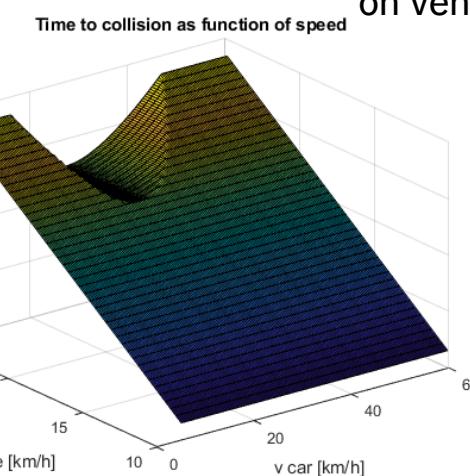
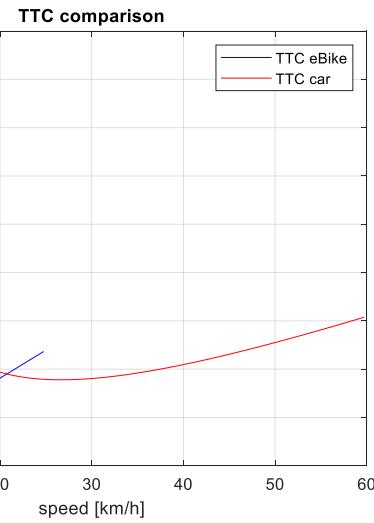
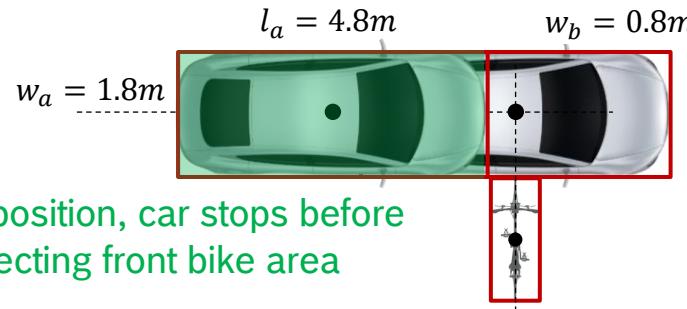
$$\text{TTC_bike} = (s_{\text{brake_bike}} + \frac{w_a}{2} + \frac{l_b}{2}) / v_{\text{bike}}$$

- ▶ In order to enable the bicycle to stop in the safe position, the bike warning must be anticipated for a value depending on vehicle dimensions
- ▶ For the given parameter and in the considered speed ranges $TTC_{car} < TTC_{bike}$

Case 2b shows benefit in warning the eBike rider

Coordination based on TTC

Case 3a – Car side middle



- ▶ TTC_{car} depends on the braking distance and current speed, additionally on bike width and car length
- ▶ TTC_{bike} only depends on braking distance and current speed

$$s_{\text{brake_car}} = v_{\text{car}} * \text{ttl_AEB_car} + 1/2 * v_{\text{car}}^2 / a_{\text{car}};$$

$$\text{TTC_car} = (s_{\text{brake_car}} + l_a/2 + w_b/2) / v_{\text{car}}$$

$$s_{\text{brake_bike}} = v_{\text{bike}} * t_{\text{react_bike}} + 1/2 * v_{\text{bike}}^2 / a_{\text{bike}};$$

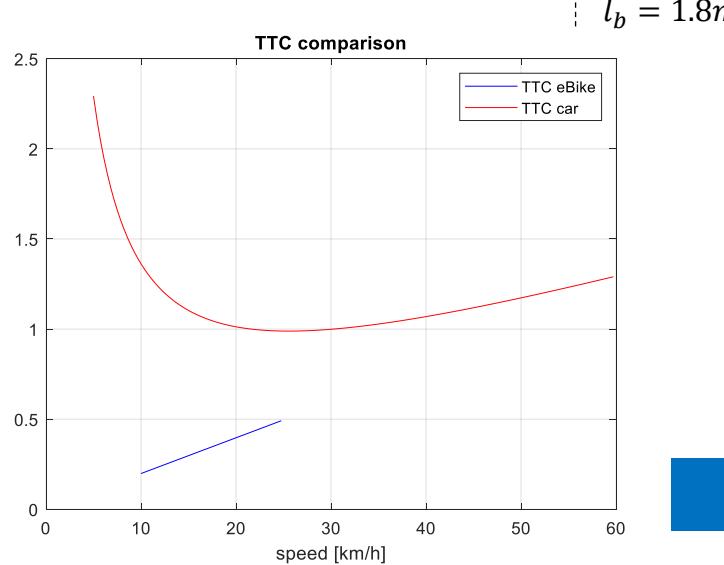
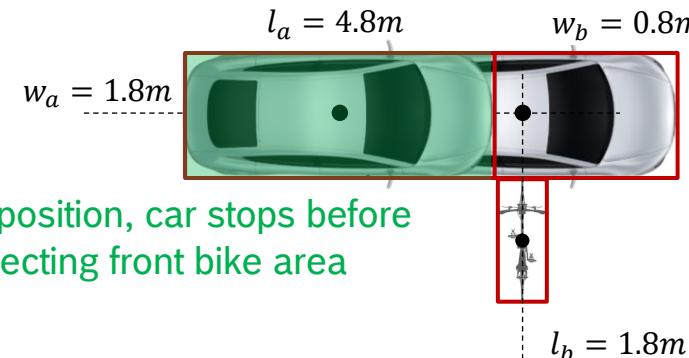
$$\text{TTC_bike} = (s_{\text{brake_bike}}) / v_{\text{bike}}$$

- ▶ In order to enable the car to stop in the safe position, the AEB intervention must be anticipated for a value depending on vehicle dimensions
- ▶ For the given parameter and in the considered speed ranges $TTC_{car} < TTC_{bike}$ for most speed values combinations
- ▶ In this range can only be avoided by warning the eBike rider

Case 3a shows high benefit in warning the eBike rider

Coordination based on TTC

Case 3b – Car side middle



- ▶ TTC_{car} depends on the braking distance and current speed, additionally on bike width and car length
- ▶ TTC_{bike} only depends on braking distance and current speed

$$s_{brake_car} = v_{car} * t_{tl_AEB_car} + 1/2 * v_{car}^2 / a_{car};$$

$$TTC_{car} = (s_{brake_car} + l_a/2 + w_b/2) / v_{car}$$

$$s_{brake_bike} = v_{bike} * t_{react_bike} + 1/2 * v_{bike}^2 / a_{bike};$$

$$TTC_{bike} = (s_{brake_bike}) / v_{bike}$$

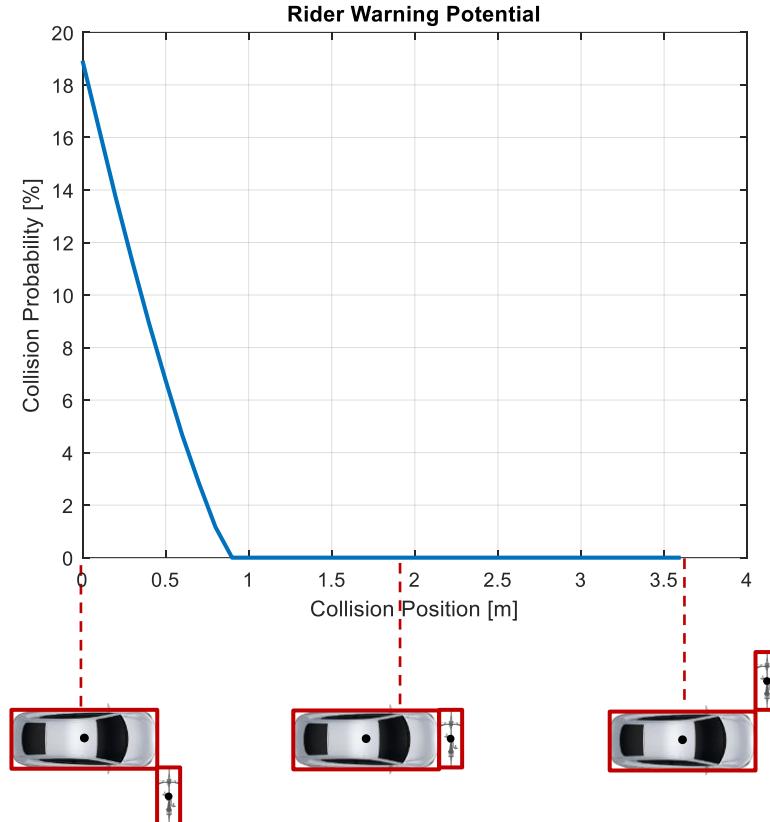
- ▶ In order to enable the car to stop in the safe position, the AEB intervention must be anticipated for a value depending on vehicle dimensions

- ▶ For the given parameter and in the considered speed ranges is always $TTC_{car} > TTC_{bike}$
- ▶ Accidents can only be avoided by warning the eBike rider

Case 3b shows the highest benefit in warning the eBike rider

Coordination based on TTC

Potential of rider warnings considering frontal collision, scenario a

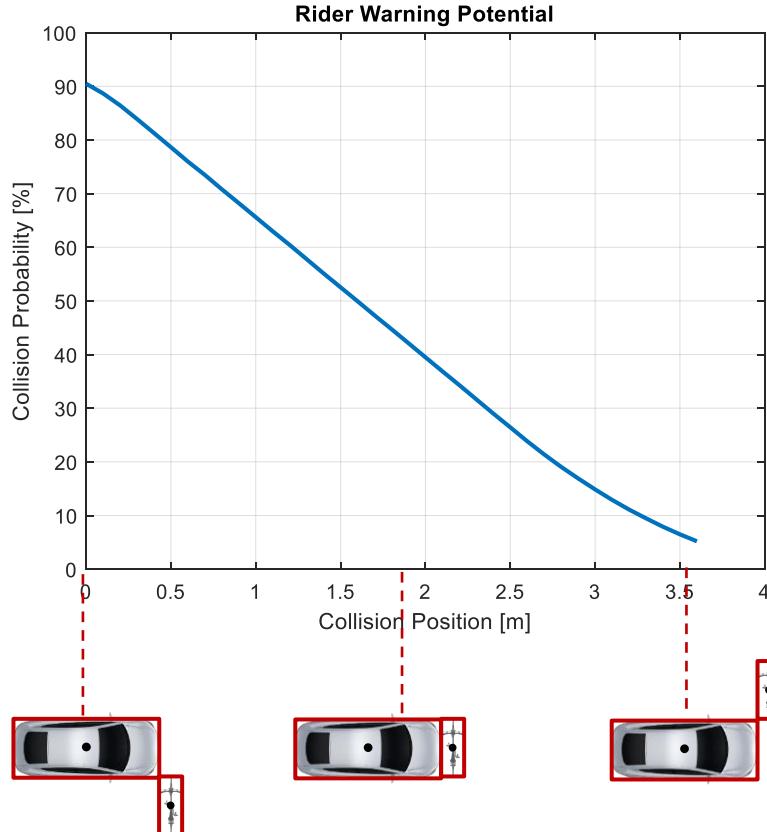


- ▶ The picture shows for a given collision position the percentage of situations calculated over all possible car speeds and bike speeds where the car would hit the bicycle rider
- ▶ In this situations, the only way to avoid an accident it is to release a warning for bicycle rider
- ▶ Assumption: no sight obstruction (AEB can always recognize the rider early enough)
- ▶ Assumption: the AEB system actuates an intervention only when an accident is unavoidable
 - ▶ $t_{intervention} = \min(TTC_{car}, TTC_{bike})$
 - ▶ If $TTC_{car} > TTC_{bike}$ the car hits the bike
- ▶ Assumption: uniform speed distribution for car and bike in the defined ranges
- ▶ The results are strongly dependent on the chosen parameters
 - ▶ car speeds (5-60km/h)
 - ▶ bike speeds (10-25km/h)
 - ▶ car dimensions ($l_a = 4.8$; % car length [m], $w_a = 1.8$; % car width [m])
 - ▶ bike dimensions ($l_b = 1.8$; % bike length [m], $w_b = 0.8$; % bike width [m])
 - ▶ AEB parameters ($ttl_{AEB_car} = 0.2s$; $a_{car} = 9.8m/s^2$)
 - ▶ Bicycle braking parameters ($t_{react_bike} = 0.5s$; $a_{bike} = 6m/s^2$)

For scenario a frontal collisions, bike warnings show potential only for small overlap between car and bicycle

Coordination based on TTC

Potential of rider warnings considering frontal collision, scenario b

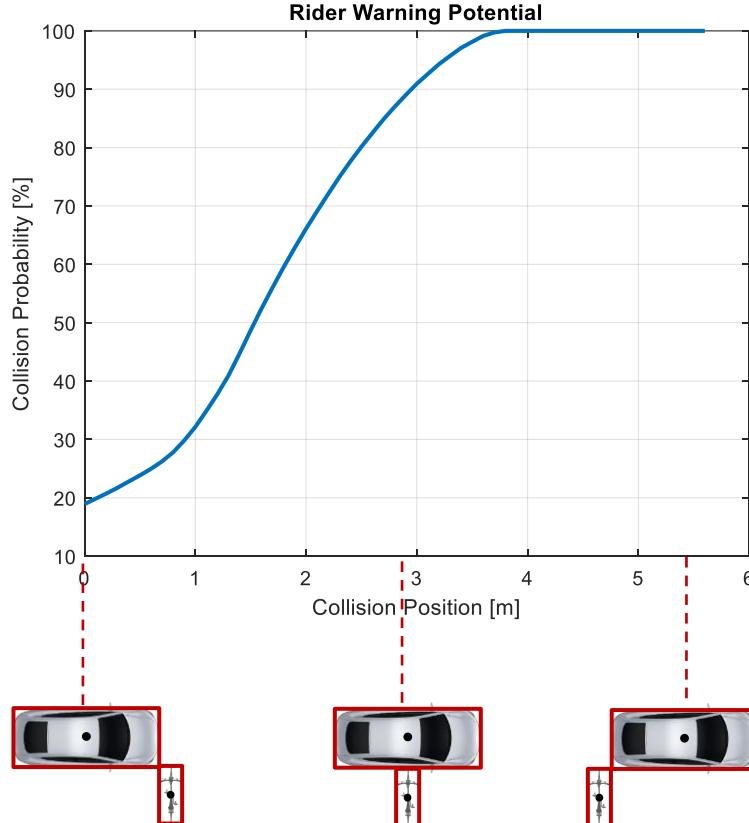


- ▶ The picture shows for a given collision position the percentage of situations calculated over all possible car speeds and bike speeds where the car would hit the bicycle rider
- ▶ In this situations, the only way to avoid an accident it is to release a warning for bicycle rider
- ▶ Assumption: no sight obstruction (AEB can always recognize the rider early enough)
- ▶ Assumption: the AEB system actuates an intervention only when an accident is unavoidable
 - ▶ $t_{intervention} = \min(TTC_{car}, TTC_{bike})$
 - ▶ If $TTC_{car} > TTC_{bike}$ the car hits the bike
- ▶ Assumption: uniform speed distribution for car and bike in the defined ranges
- ▶ The results are strongly dependent on the chosen parameters
 - ▶ car speeds (5-60km/h)
 - ▶ bike speeds (10-25km/h)
 - ▶ car dimensions ($l_a = 4.8$; % car length [m], $w_a = 1.8$; % car width [m])
 - ▶ bike dimensions ($l_b = 1.8$; % bike length [m], $w_b = 0.8$; % bike width [m])
 - ▶ AEB parameters ($ttl_{AEB_car} = 0.2s$; $a_{car} = 9m/s^2$)
 - ▶ Bicycle braking parameters ($t_{react_bike} = 0.0s$; $a_{bike} = 7m/s^2$)

For scenario b frontal collisions, bike warnings show high potential in reducing accident probability

Coordination based on TTC

Potential of rider warnings considering lateral collision, scenario a

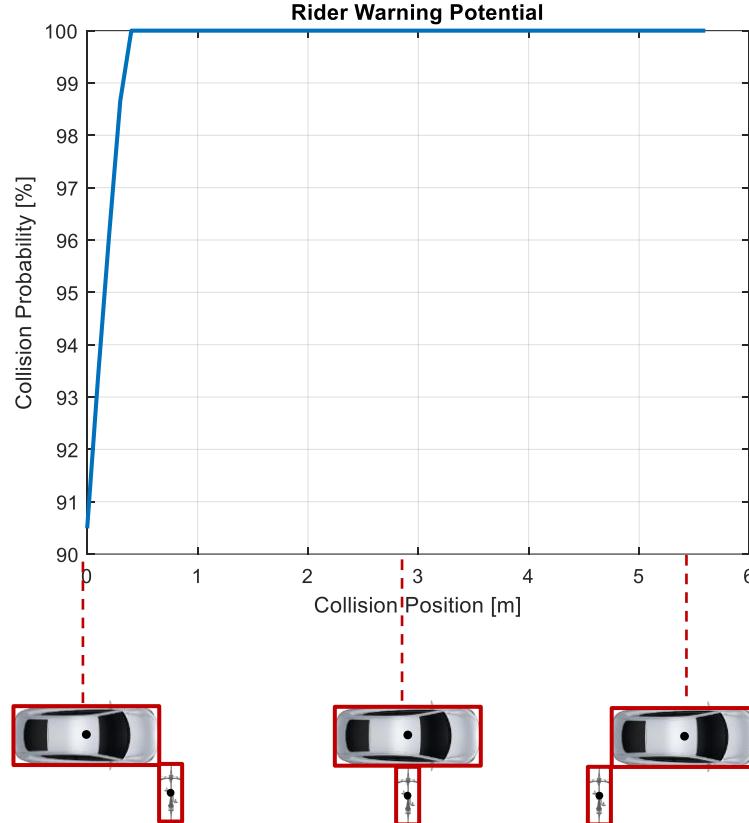


- ▶ The picture shows for a given collision position the percentage of situations calculated over all possible car speeds and bike speeds where the car would hit the bicycle rider
- ▶ In this situations, the only way to avoid an accident it is to release a warning for bicycle rider
- ▶ Assumption: no sight obstruction (AEB can always recognize the rider early enough)
- ▶ Assumption: the AEB system actuates an intervention only when an accident is unavoidable
 - ▶ $t_{intervention} = \min(TTC_{car}, TTC_{bike})$
 - ▶ If $TTC_{car} > TTC_{bike}$ the car hits the bike
- ▶ Assumption: uniform speed distribution for car and bike in the defined ranges
- ▶ The results are strongly dependent on the chosen parameters
 - ▶ car speeds (5-60km/h)
 - ▶ bike speeds (10-25km/h)
 - ▶ car dimensions ($l_a = 4.8$; % car length [m], $w_a = 1.8$; % car width [m])
 - ▶ bike dimensions ($l_b = 1.8$; % bike length [m], $w_b = 0.8$; % bike width [m])
 - ▶ AEB parameters ($ttl_{AEB_car} = 0.2s$; $a_{car} = 9.8m/s^2$)
 - ▶ Bicycle braking parameters ($t_{react_bike} = 0.5s$; $a_{bike} = 6m/s^2$)

For lateral collisions, bike warnings show high potential

Coordination based on TTC

Potential of rider warnings considering lateral collision, scenario a



- ▶ The picture shows for a given collision position the percentage of situations calculated over all possible car speeds and bike speeds where the car would hit the bicycle rider
- ▶ In this situations, the only way to avoid an accident it is to release a warning for bicycle rider
- ▶ Assumption: no sight obstruction (AEB can always recognize the rider early enough)
- ▶ Assumption: the AEB system actuates an intervention only when an accident is unavoidable
 - ▶ $t_{intervention} = \min(TTC_{car}, TTC_{bike})$
 - ▶ If $TTC_{car} > TTC_{bike}$ the car hits the bike
- ▶ Assumption: uniform speed distribution for car and bike in the defined ranges
- ▶ The results are strongly dependent on the chosen parameters
 - ▶ car speeds (5-60km/h)
 - ▶ bike speeds (10-25km/h)
 - ▶ car dimensions ($l_a = 4.8$; % car length [m], $w_a = 1.8$; % car width [m])
 - ▶ bike dimensions ($l_b = 1.8$; % bike length [m], $w_b = 0.8$; % bike width [m])
 - ▶ AEB parameters ($ttl_{AEB_car} = 0.2s$; $a_{car} = 9.8m/s^2$)
 - ▶ Bicycle braking parameters ($t_{react_bike} = 0.5s$; $a_{bike} = 6m/s^2$)

For lateral collisions, bike warnings show high potential every scenario

Coordination based on TTC

Collision Points Statistics from CATS study

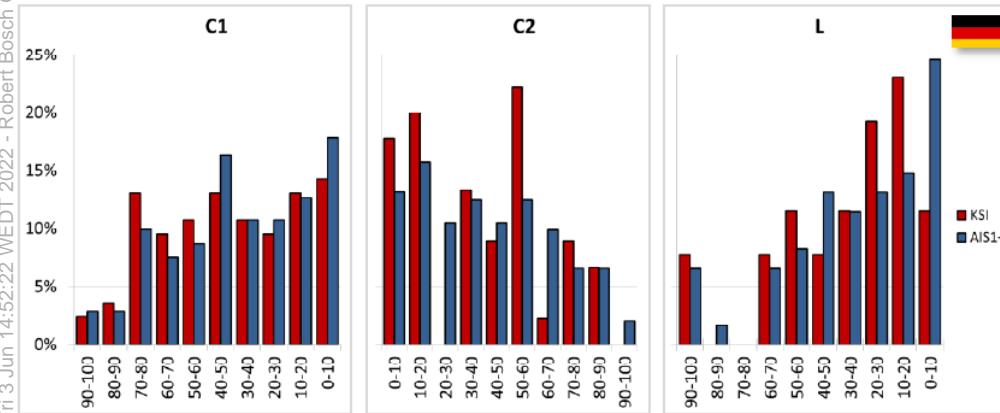


Figure 23: Overview of cyclist accidents in target population by collision point for the crossing and longitudinal accident scenarios for both the low severity injuries (MAIS1+) and the seriously injured and fatalities (KSI)

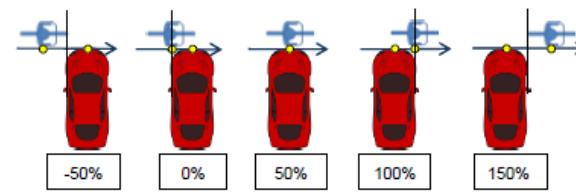


Figure 20: Collision point definition in the C2 scenario

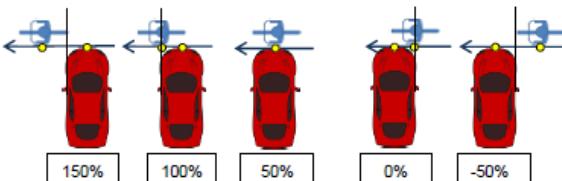
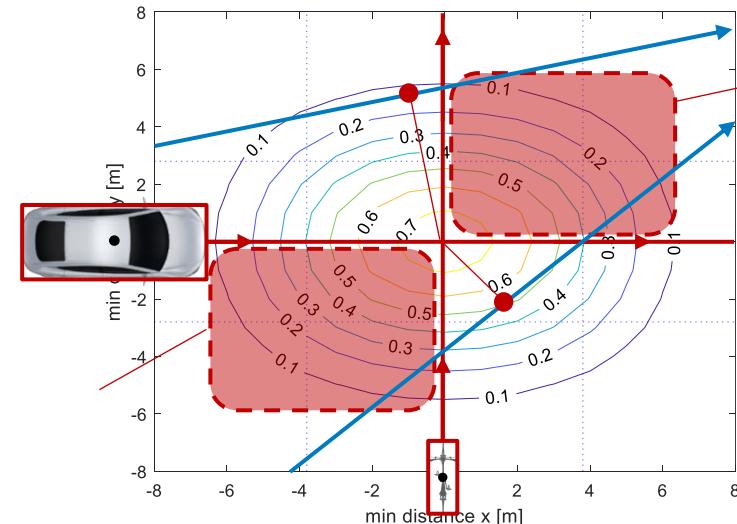


Figure 21: Collision point definition in the C1 scenario

Figure 23: Overview of cyclist accidents in target population by collision point for the crossing and longitudinal accident scenarios for both the low severity injuries (MAIS1+) and the seriously injured and fatalities (KSI)



The statistic only describes accidents in Q2 (combination of case 1 and 2) quadrant, are there any accidents in Q4 (case 3)?

Coordination based on TTC

Collision Points Statistics from CATS study, scenario a

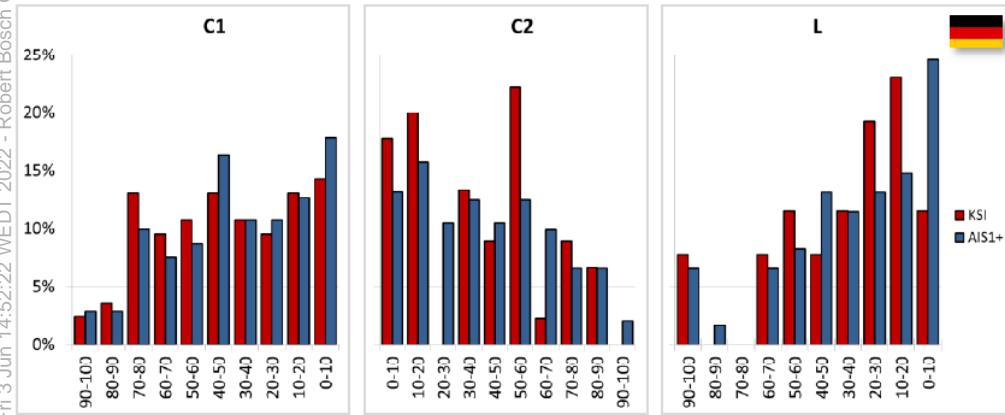


Figure 23: Overview of cyclist accidents in target population by collision point for the crossing and longitudinal accident scenarios for both the low severity injuries (MAIS1+) and the seriously injured and fatalities (KSI)

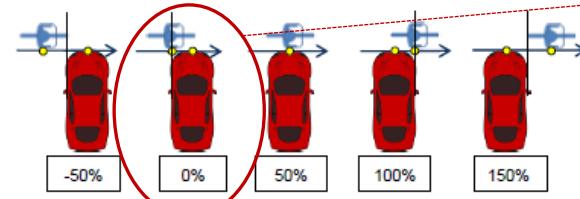


Figure 20: Collision point definition in the C2 scenario

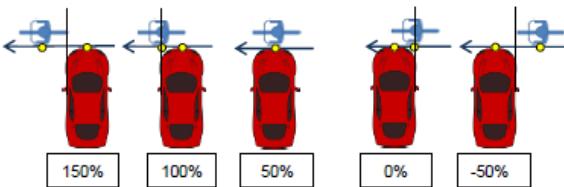
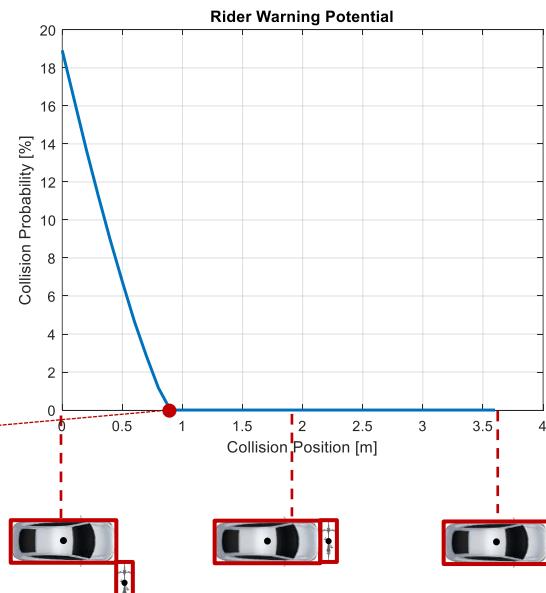


Figure 21: Collision point definition in the C1 scenario



- ▶ 0.9m collision position corresponds to 0% in the CATS study
- ▶ CATS study doesn't show any collisions for positions < 0% and the calculated potential of rider warnings for positions > 0% is zero

Combining CATS statistics with potential of rider warnings shows no potential for rider warnings for scenario a

Coordination based on TTC

Collision Points Statistics from CATS study, scenario b

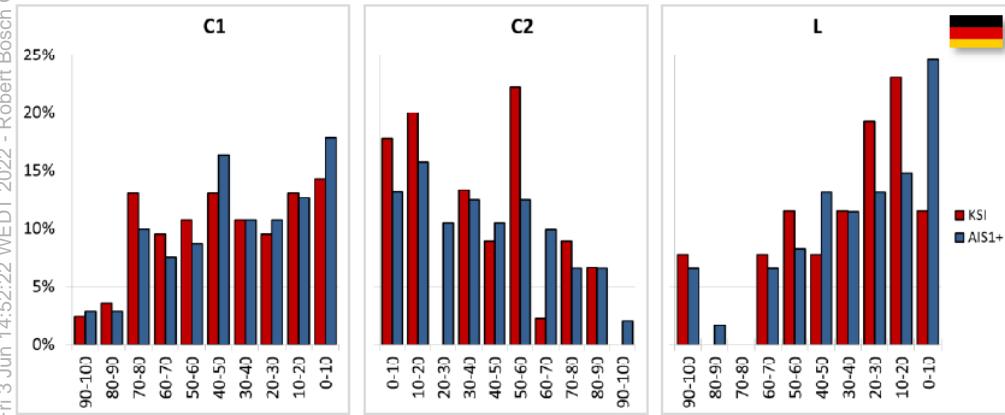


Figure 23: Overview of cyclist accidents in target population by collision point for the crossing and longitudinal accident scenarios for both the low severity injuries (MAIS1+) and the seriously injured and fatalities (KSI)

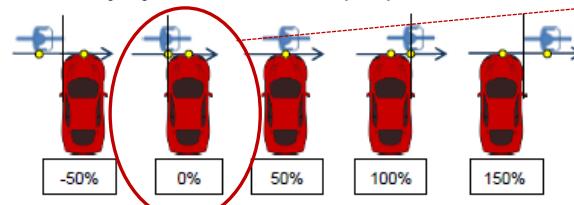


Figure 20: Collision point definition in the C2 scenario

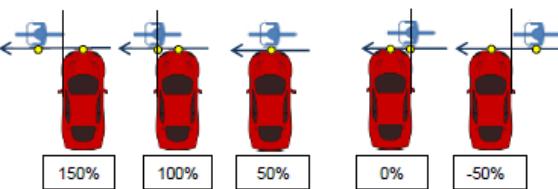
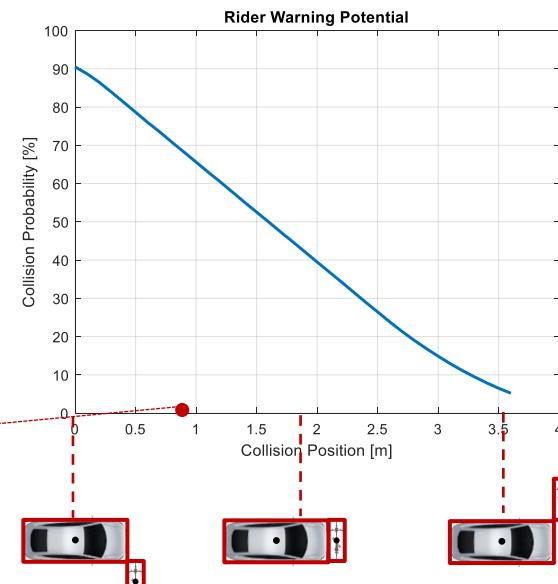


Figure 21: Collision point definition in the C1 scenario



- 0.9m collision position corresponds to 0% in the CATS study
- CATS study doesn't show any collisions between 0% and 100%, for every value in this range, scenario b show potential for rider warnings

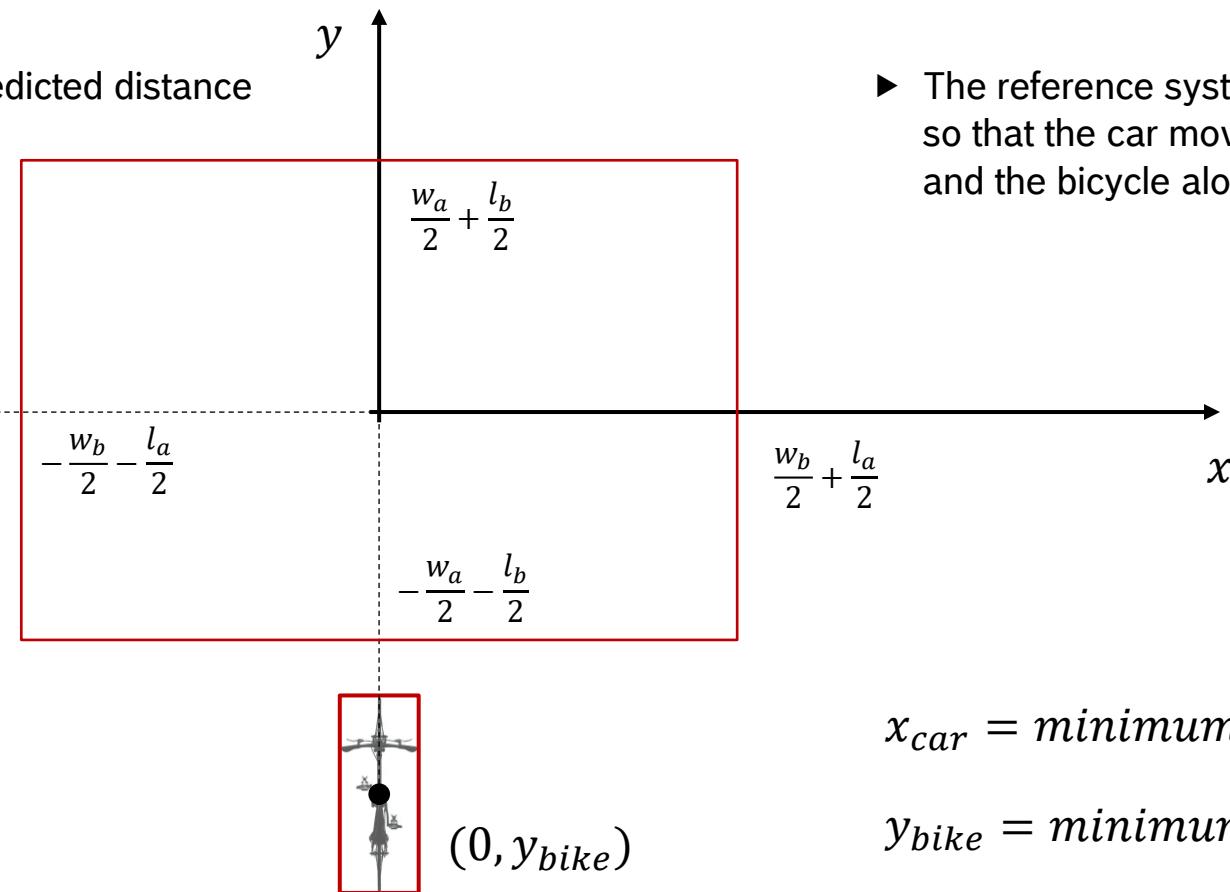
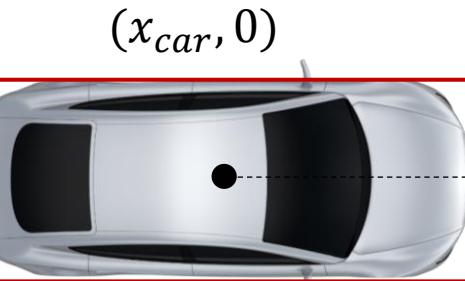
Combining CATS statistics with potential of rider warnings shows significant potential for rider warnings for scenario b

Effect of prediction accuracy on false positives and false negatives

System Architectures

Representation

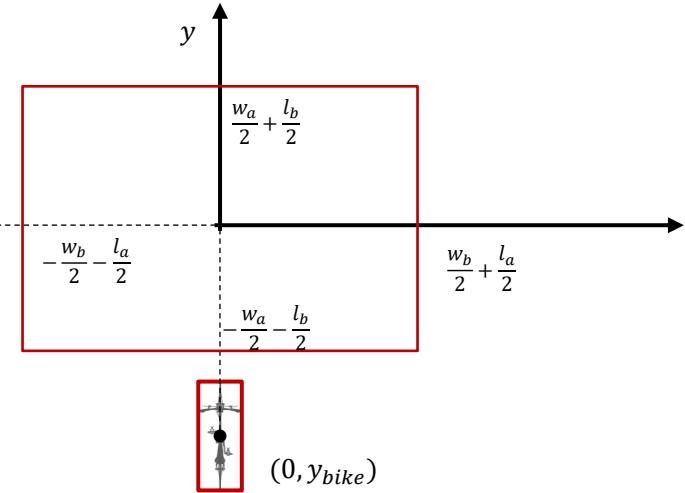
- ▶ Representation of minimum predicted distance



System Architectures

Representation

- ▶ Representation of minimum distance



$$x_{car} = \text{minimum distance } x$$

$$y_{bike} = \text{minimum distance } y$$

% Function to integrate

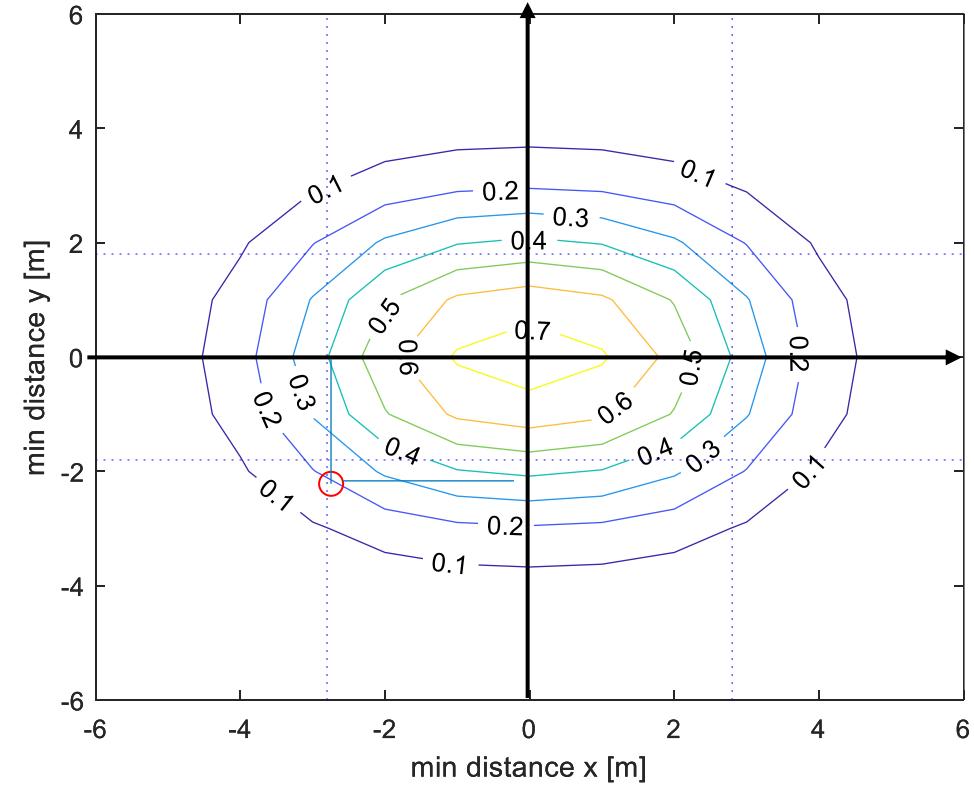
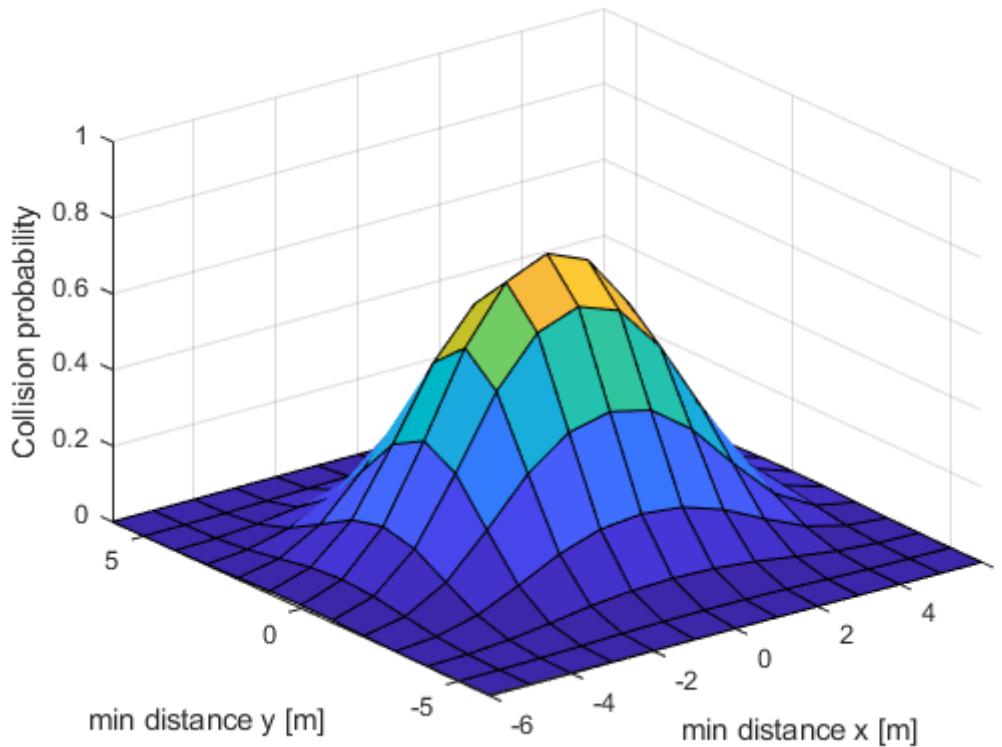
```
p_function_2D_x = @(x) normpdf(x,mu_bike_x, sigma_pred).* (normcdf((x + collision_area_x), mu_car_x, sigma_pred) - normcdf((x - collision_area_x), mu_car_x, sigma_pred));%
p_function_2D_y = @(y) normpdf(y,mu_bike_y, sigma_pred).* (normcdf((y + collision_area_y), mu_car_y, sigma_pred) - normcdf((y - collision_area_y), mu_car_y, sigma_pred));%

p_coll_2D_x = integral(p_function_2D_x, -100, 100);
p_coll_2D_y = integral(p_function_2D_y, -100, 100);
p_coll_2D = p_coll_2D_x * p_coll_2D_y;
```

System Architectures

Probably vs min distance

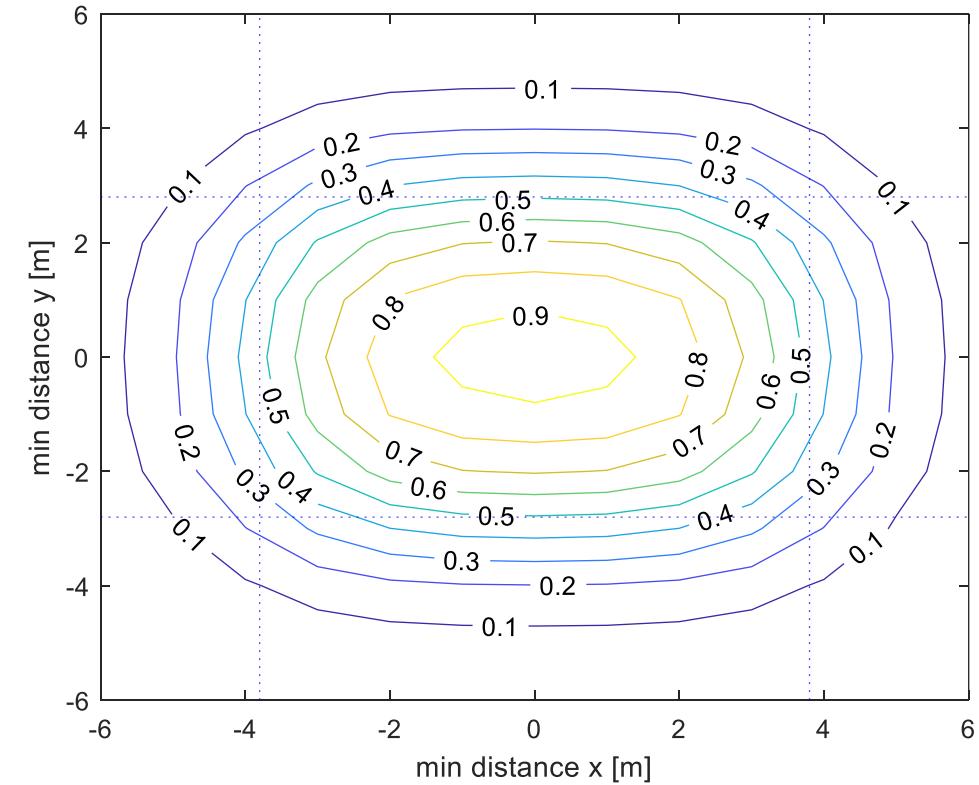
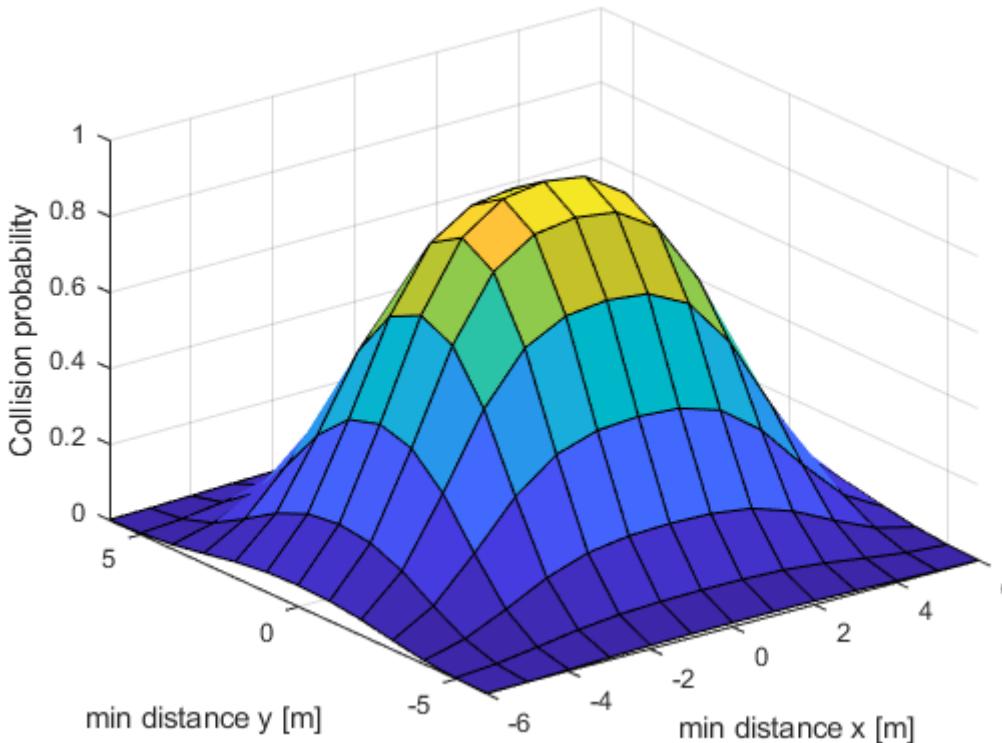
- It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
- No safety buffer around bicycle, sigma pred = 1



System Architectures

Probably vs min distance

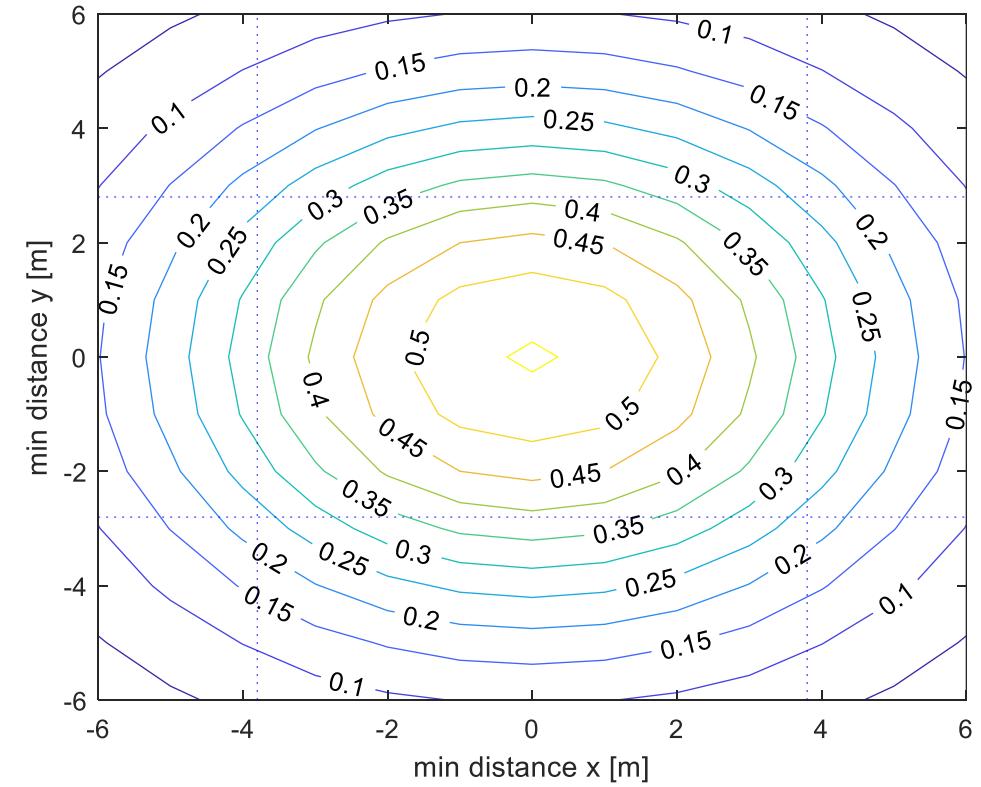
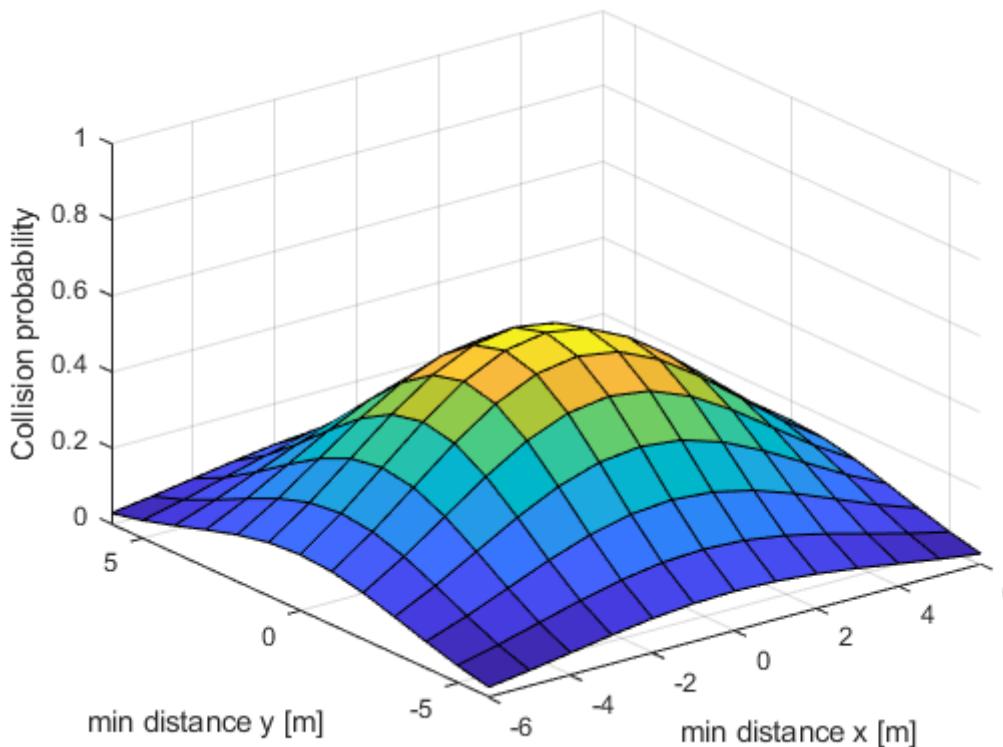
- It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
- Safety buffer = 1m around bicycle, sigma pred = 1



System Architectures

Probably vs min distance

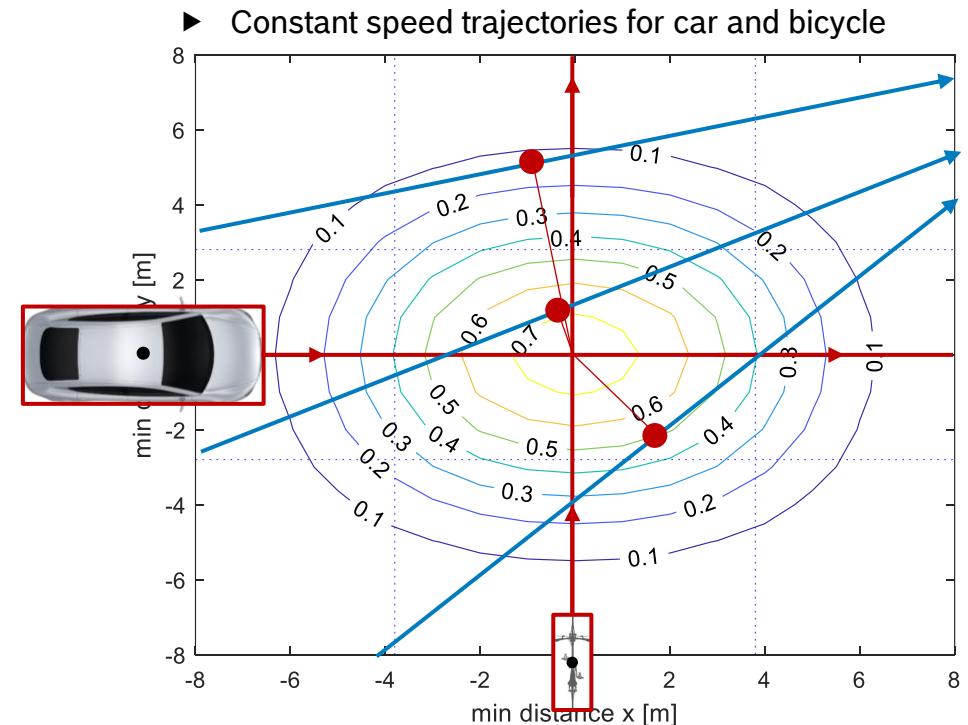
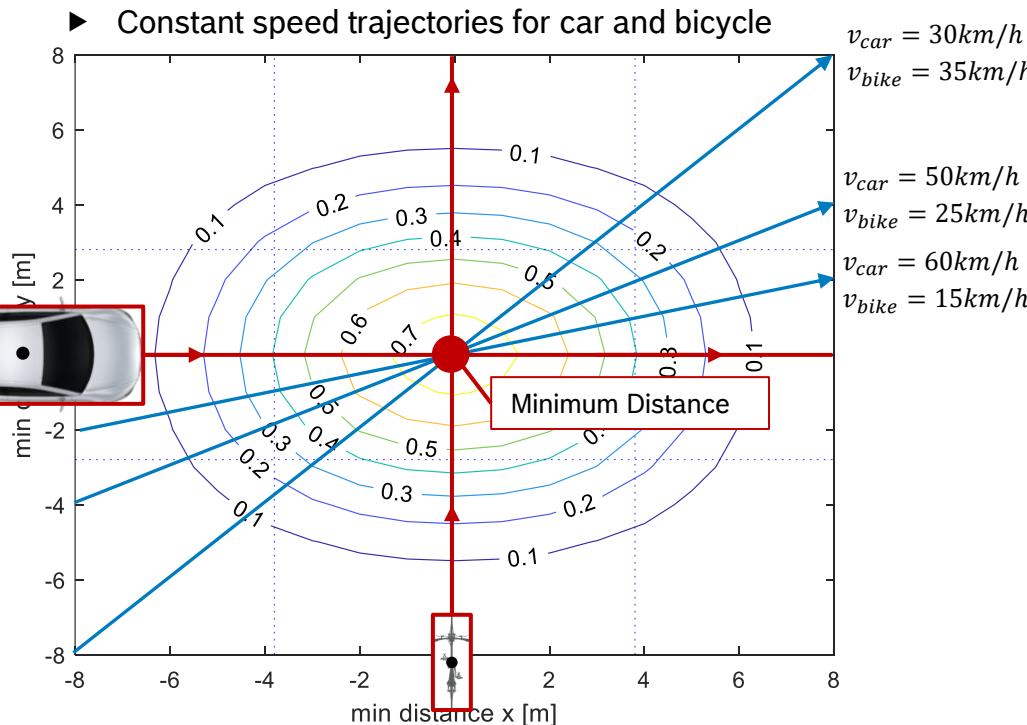
- ▶ It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
 - ▶ Safety buffer = 1m around bicycle, sigma pred = 2



System Architectures

Probably vs min distance

- Safety buffer = 1m around bicycle, sigma pred = 1.5 $\rightarrow \sigma = 2.2$

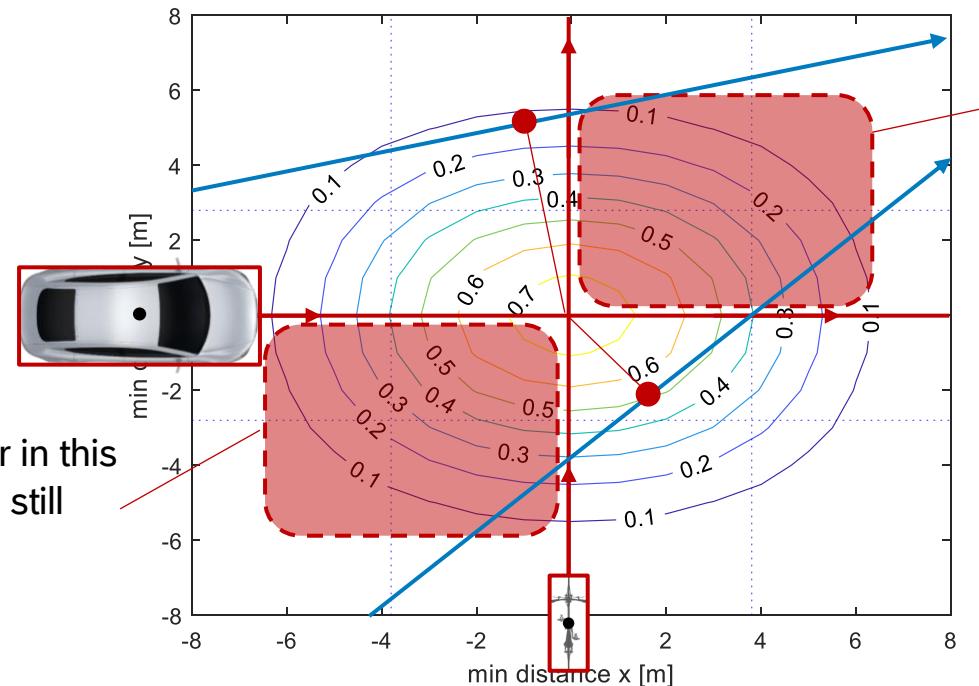


- The point of minimum distance provides approximation of the maximum of the collision probability. It is the intersection of the trajectory and the line orthogonal to the trajectory passing through the center of the graph.

System Architectures

Probably vs min distance

- Safety buffer = 1m around bicycle, sigma pred = 1.5



The minimum distance is never in this area unless both vehicle reach still stand

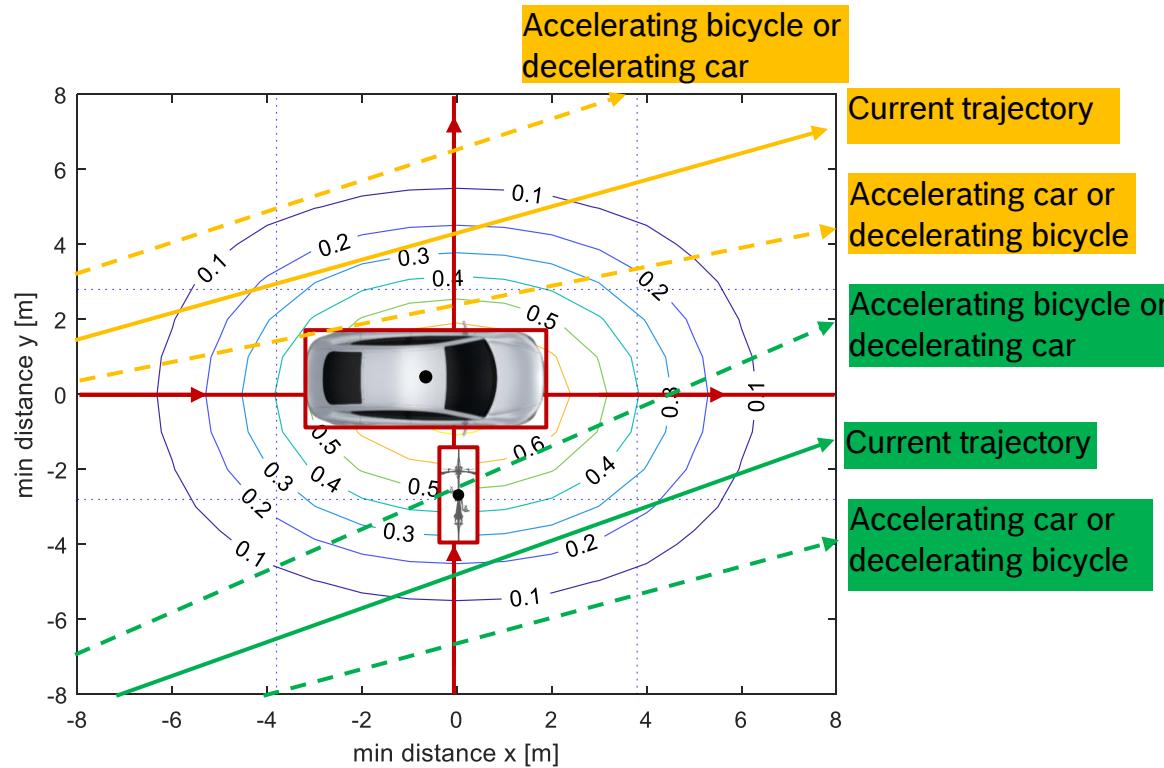
The minimum distance is never in this area unless both vehicle reach still stand

In this area, a braking of the car is initially counterproductive for the accident probability. Further distinction between collision with front or side of car might help predict accident severity

System Architectures

Probably vs min distance

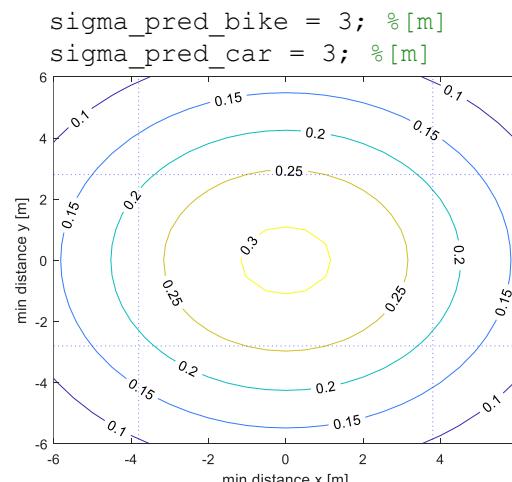
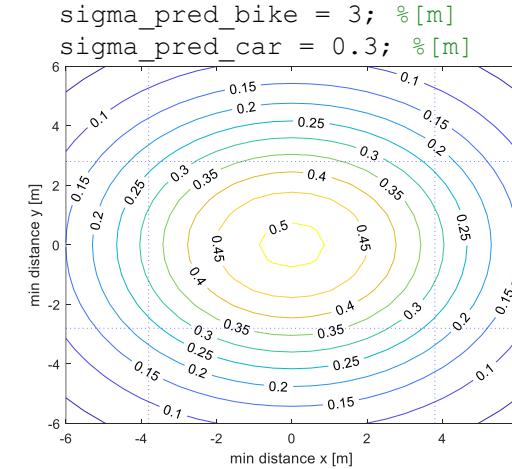
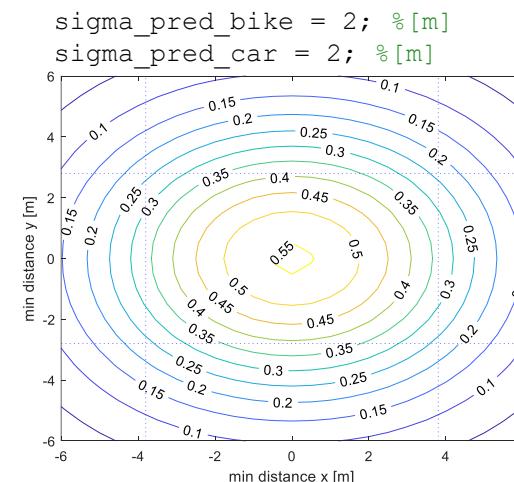
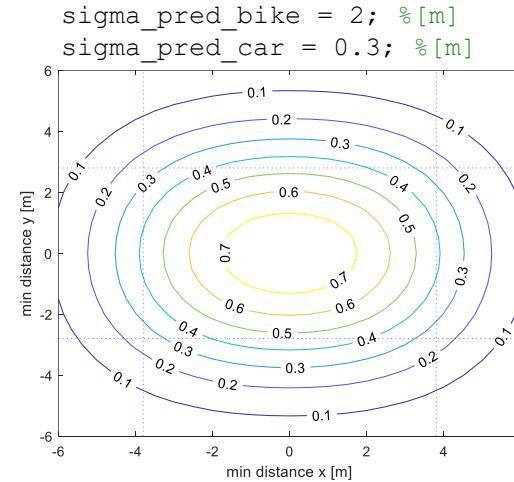
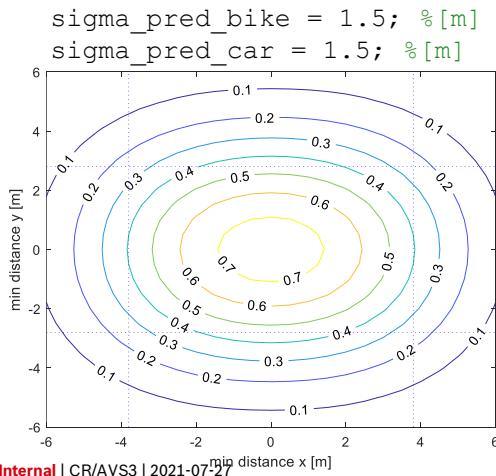
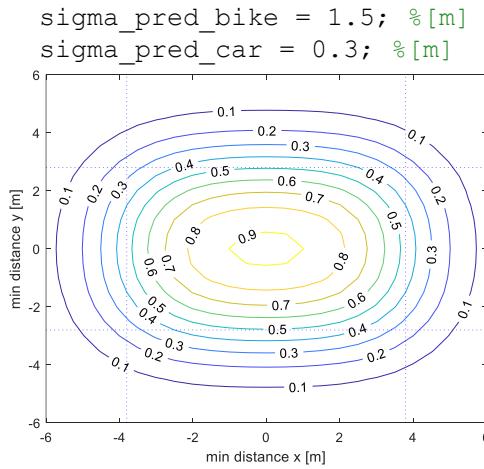
- ▶ Safety buffer = 1m around bicycle, sigma pred = 1.5



- ▶ Depending on the area where the minimum distance between the car and the bicycle happens, a variation in the speed can be helpful or counterproductive
- ▶ Q2 the car has deceleration with high probability
 - ▶ the driver has seen the rider
 - ▶ AEB is active
- ▶ Q4 is where the sight might be obstructed

System Architectures

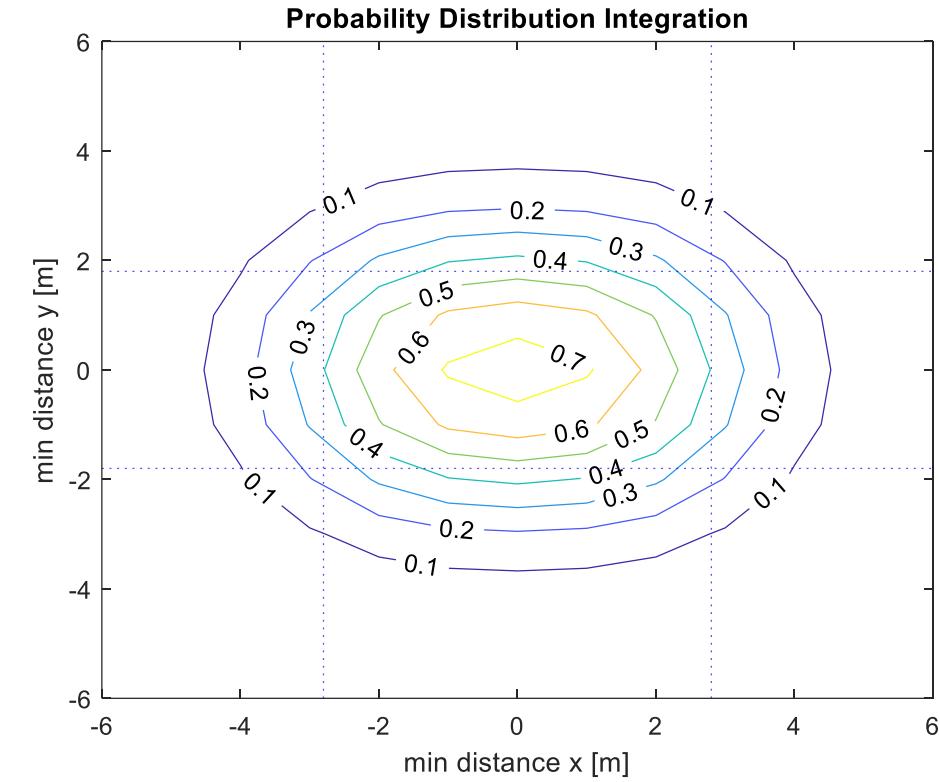
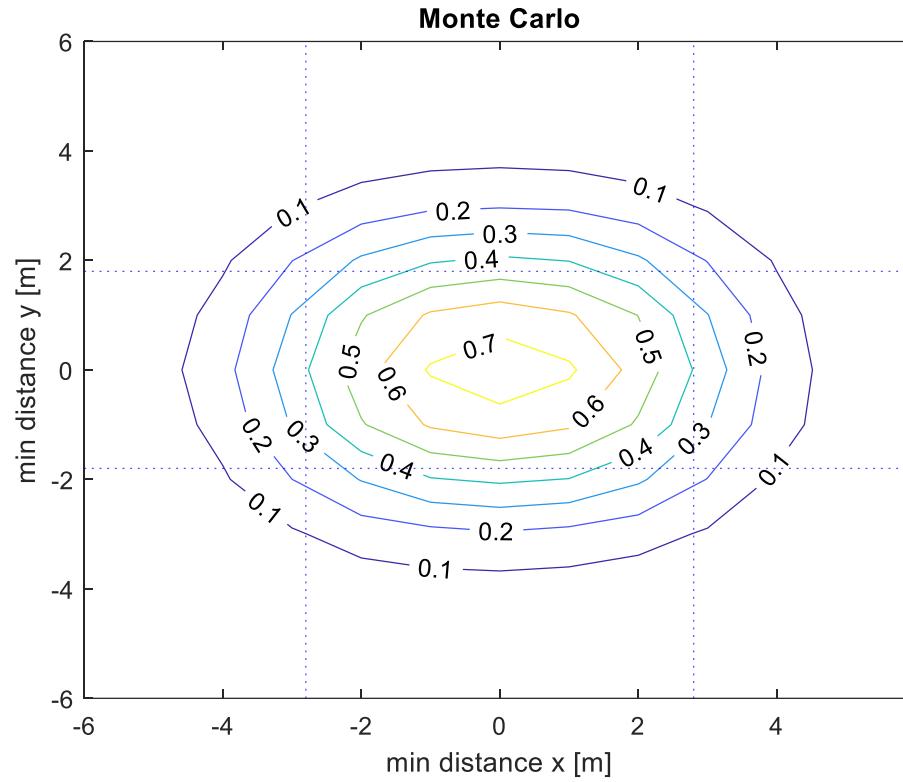
Effect of different sigma for car and ebike



System Architectures

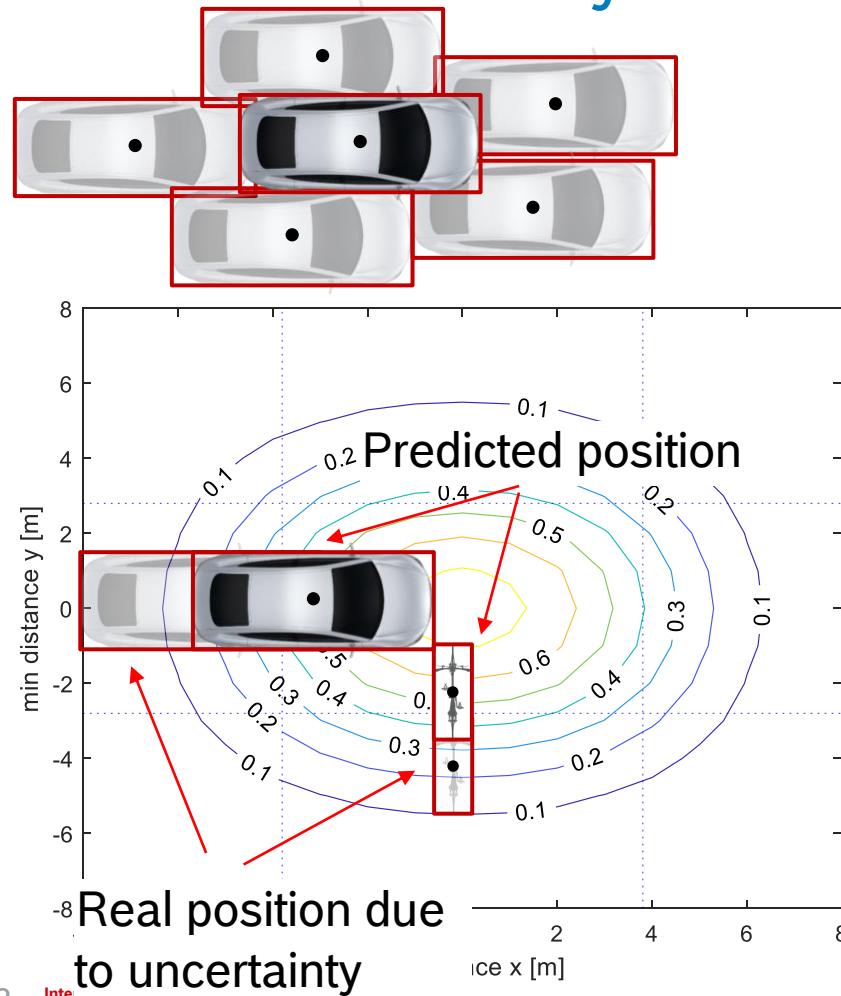
Monte Carlo check

- Monte Carlo simulation with calculation of probability with 10.000 simulations



Crossing Cyclist

Collision Probability Time Independent



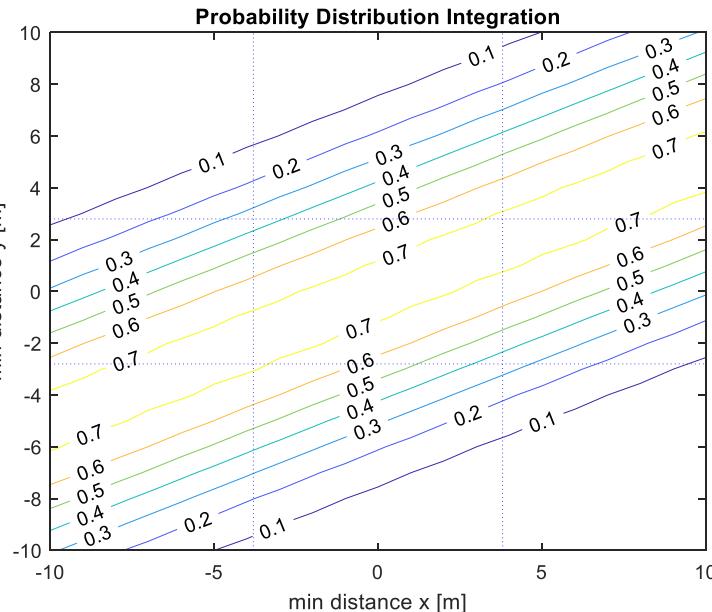
- ▶ Position uncertainty doesn't mean that the real car position is varies continuously, but it is most likely an offset. The gauss distribution is given over several situations.
- ▶ In the current calculation we provide the probability the at certain time, given the predicted position of car and bicycle, a collision occur
- ▶ The situation as shown on the left is in this calculation considered as non collision since the real distance between car an bicycle is greater than the collision area, even if there is still going to be an accident in the future
- ▶ Also with the uncertainty there might be a collision in the future or in the past

Crossing Cyclist

Collision Probability Time Independent

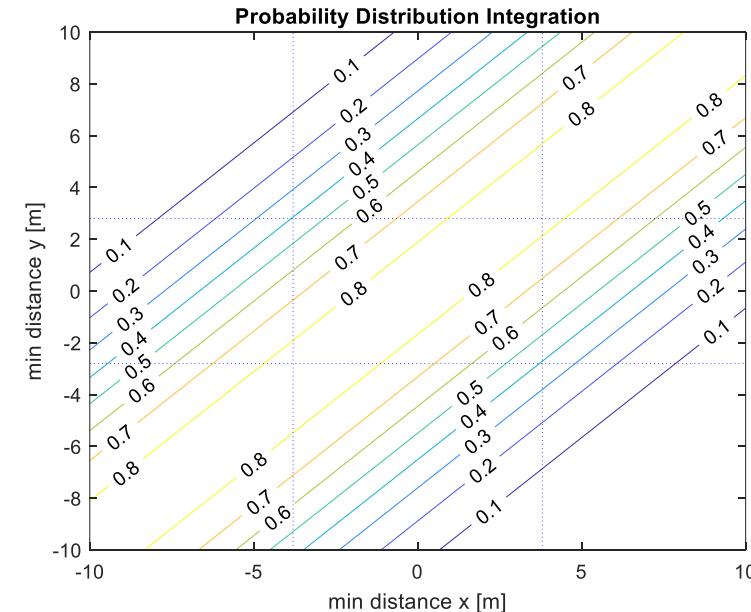
`sigma_pred = 2; % [m]`

`v_car = 50/3.6; % car speed [m/s]`
`v_bike = 25/3.6; % car speed [m/s]`

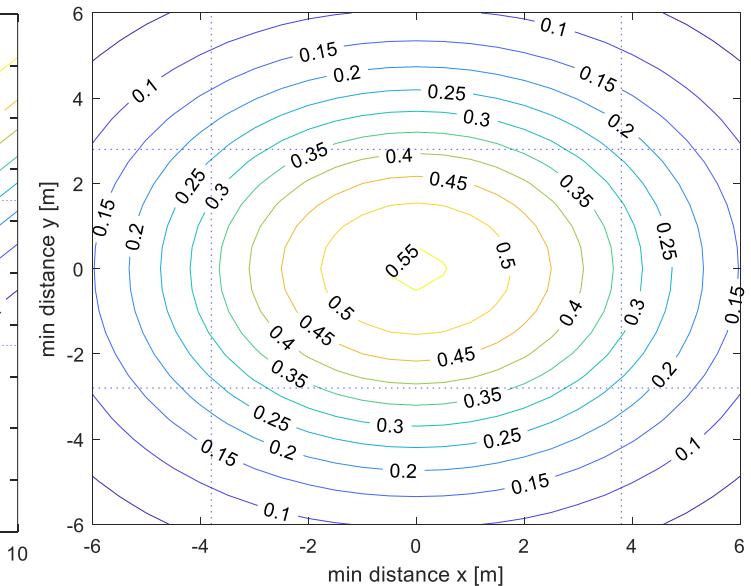


`sigma_pred = 2; % [m]`

`v_car = 15/3.6; % car speed [m/s]`
`v_bike = 15/3.6; % car speed [m/s]`



`sigma_pred_bike = 2; % [m]`
`sigma_pred_car = 2; % [m]`

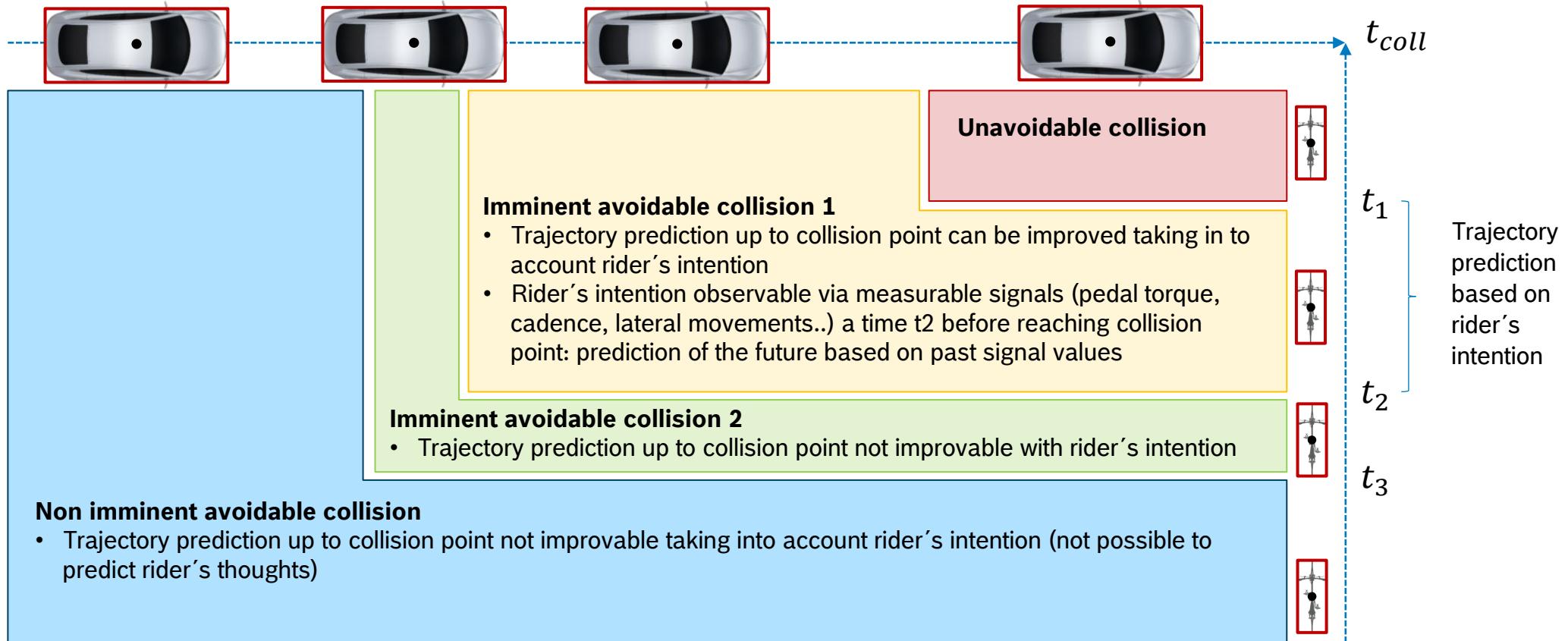


The collision probability becomes speed dependent
 The higher collision probability comes with uncertainty in the time of collision

Intervention and escalation strategies

System Architectures

Situation analysis depending on distance from collision



System Architectures

Unavoidable collision

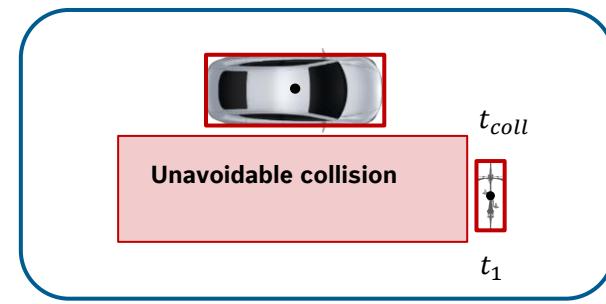
- ▶ For $t < t_1$ the system can detect an unavoidable collision

Goal: reduce severity of accident by reducing speed

Situation recognition

- ▶ Trajectory prediction eBike
 - ▶ ML not helpful:
 - Panic reaction of rider probably not “learnable” with ML (how to gather data?)
 - Since collision is unavoidable ML has no effect on false positive/negative
 - ▶ Mostly based on state estimation via GPS and vehicle sensors
- ▶ Situation detection
 - ▶ Based on minimum distance of predicted trajectories
 - ▶ Based on unavoidability of accident $t < t_1$
 - t_{reac} , a_{max} are unknown parameter

$$t_{brake} = t_{reac} + \frac{v_{init}}{a_{max}}$$



Intervention strategy

- ▶ Release high urgency warning to rider
 - ▶ Cause full braking

Improvement via communication

- ▶ CAM
 - ▶ Car sends GPS and sensor data for trajectory prediction in eBike logic
 - ▶ Car sends to bike measured relative distance and velocity to improve situation detection
- ▶ VAM
 - ▶ Bicycle sends speed and trajectory to improve car accident detection and AEB release
 - ▶ Bicycle sends stability indicator to improve AEB release

System Architectures

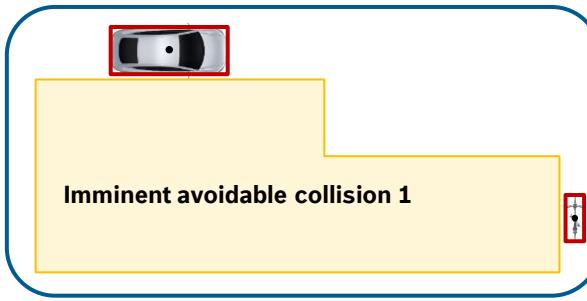
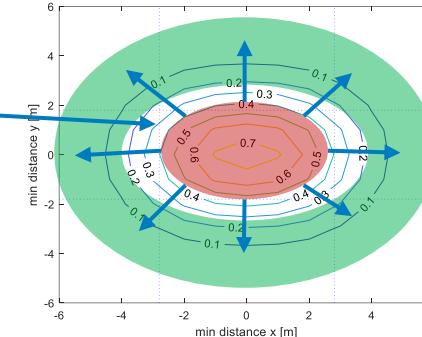
Imminent avoidable collision 1

- ▶ $t_1 < t < t_2$ imminent collision. t_2 is maximum time horizon for trajectory prediction based on rider's intention

Goal: avoid collision

Situation recognition

- ▶ Trajectory prediction eBike
 - ▶ ML helpful:
 - Rider's intention observable in current riding dynamics signals (not available in VAM)
 - ▶ Based on state estimation and trajectory prediction
 - GPS, eBike sensor based state estimation
 - Trajectory prediction based i.e. on ML methods
 - ▶ Situation detection
 - ▶ Based on minimum distance of predicted trajectories and braking distance for eBike and car



Intervention strategy

- ▶ Warning of rider with HMI to release braking intervention
- ▶ AEB release on car
- ▶ Possible speed coordination for collision probability minimization

Improvement via communication

- ▶ CAM
 - ▶ Car sends GPS and sensor data for trajectory prediction in eBike logic
- ▶ VAM
 - ▶ Bicycle sends trajectory based prediction to car which can help with an early AEB release or to avoid false interventions

System Architectures

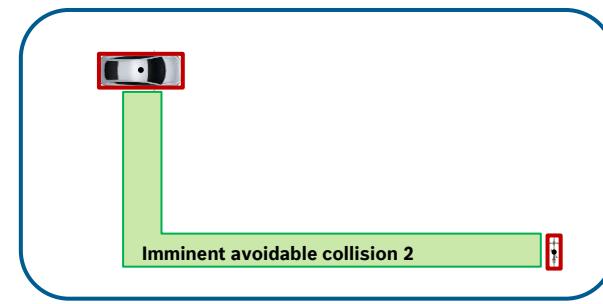
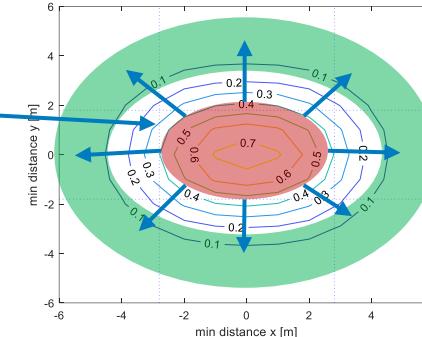
Imminent avoidable collision 2

- ▶ $t_1 < t < t_2$ imminent collision. t_2 is maximum time horizon for trajectory prediction based on rider's intention

Goal: avoid collision

Situation recognition

- ▶ Trajectory prediction eBike
 - ▶ ML less helpful:
 - Trajectory mostly depends on future rider's intention is not observable
 - ▶ Based on state estimation and simpler models
 - GPS, eBike sensor based state estimation
 - Constant speed-heading/acceleration-yaw rate model
- ▶ Situation detection
 - ▶ Based on minimum distance of predicted trajectories and braking distance for eBike and car



Intervention strategy

- ▶ Warning of rider with HMI to release braking intervention
- ▶ AEB release on car
- ▶ Possible speed coordination for collision probability minimization

Improvement via communication

- ▶ CAM
 - ▶ Car sends GPS and sensor data for trajectory prediction in eBike logic
- ▶ VAM
 - ▶ Bicycle sends trajectory based prediction to car which can help with an early AEB release or to avoid false interventions

System Architectures

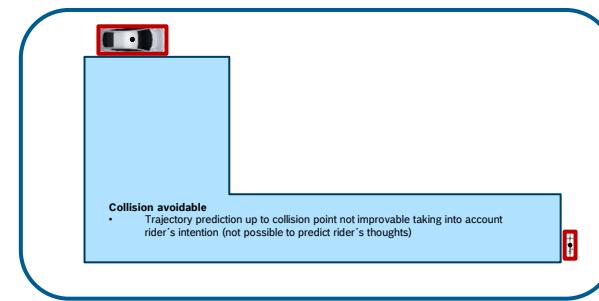
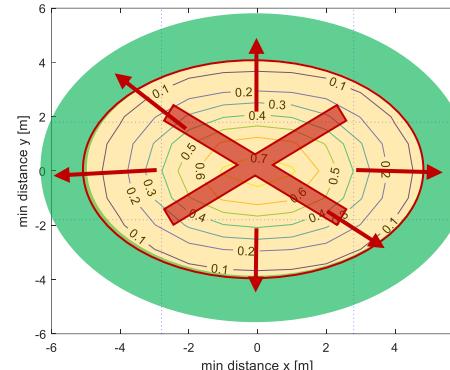
Non imminent avoidable collision

- $t > t_3$ non imminent collision. t_3 : time horizon to avoid a collision with low speed variation

Goal: ensure low collision probability

Situation recognition

- Trajectory prediction eBike
 - ML not helpful:
 - The trajectory mostly depends on future rider's intention is not observable
 - Mostly based on state estimation via **GPS** and **vehicle sensors**
 - Constant speed-heading/acceleration-yaw rate model
 - Trajectory shows where the rider will be w/o changes in the current speed/heading or acceleration/yaw rate
 - Possible improvement with **navigation data** or past user routes
- Situation detection
 - Based on minimum distance of predicted trajectories
 - Information for rider: if you do not change speed/heading your trajectory show a possible collision with a car



Intervention strategy

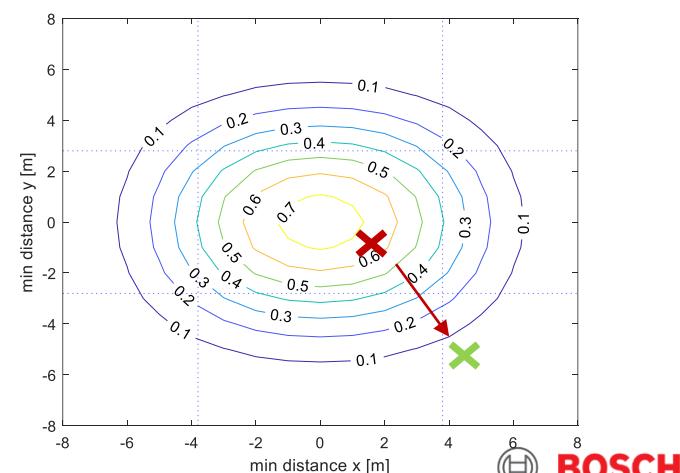
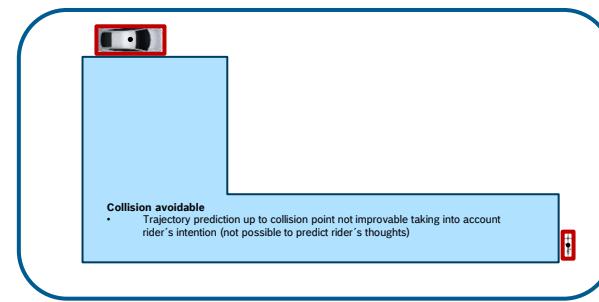
- Information of rider and driver with HMI
- Speed suggestion to eBike and car
- Possible coordination of car and bicycle speed with active speed limitation

Improvement via communication

- CAM
 - Car sends trajectory based on most probable path which considers navigation data and past user routes
 - Exchange of speed suggestions to keep collision probability low
- VAM
 - Bicycle sends trajectory based on most probable path which considers navigation data and past user routes
 - Exchange of speed suggestions to keep collision probability low

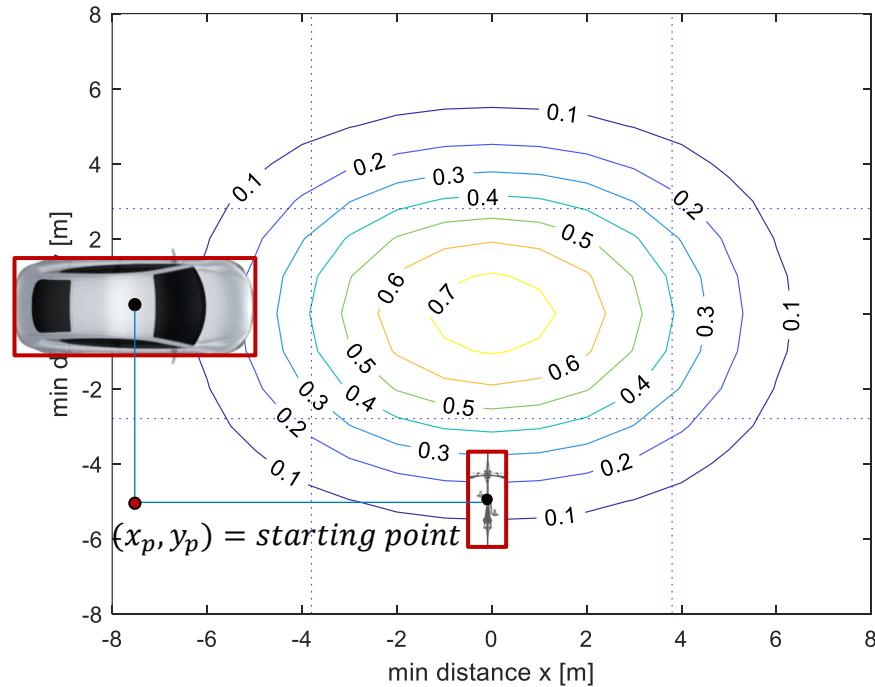
Non imminent avoidable collision Intervention strategy

- ▶ Goals of the function:
 - ▶ Inform rider and auto driver of a possible non imminent collision
 - ▶ Ensure low probability collision with small speed variations
- ▶ Time horizon:
 - ▶ To ensure a small speed variation the time horizon must be long → >2 - 4s (speed dependent)
 - ▶ For the chosen time horizon it is not reasonable to use a complex models: the future trajectory mostly depends on the future rider's intention which is not yet observable
 - A most probable path calculation might be supported with navigation data or with past routes for car and ebike
 - Reasonable model: constant speed → information for the rider/driver: if you continue with current speed there is a certain collision probability
- ▶ Intervention strategy:
 - ▶ Calculate minimum distance between car and ebike with constant speed model
 - ▶ calculate collision probability
 - ▶ if probability > threshold
 - Inform driver and ebike rider
 - calculate speed variation suggestion to reduce collision probability below a given value
 - coordinate speed variation of car and ebike



Non imminent avoidable collision

Calculation of minimum distance with constant v model (Px script)



(x_p, y_p) = starting point

v_x = car speed

v_y = bike speed

$$x_{car}(t) = x_p + v_x \cdot t$$

$$y_{bike}(t) = y_p + v_y \cdot t$$

$$\text{distance}(t)^2 = x_{car}^2(t) + y_{bike}^2(t)$$

$$\text{distance}(t)^2 = x_p^2 + v_x^2 \cdot t^2 + 2x_p \cdot v_x \cdot t + y_p^2 + v_y^2 \cdot t^2 + 2y_p \cdot v_y \cdot t$$

$$\frac{d \text{distance}(t)^2}{dt} = 2 \cdot v_x^2 \cdot t + 2 \cdot x_p \cdot v_x + 2 \cdot v_y^2 \cdot t + 2 \cdot y_p \cdot v_y$$

$$\frac{d \text{distance}(t)^2}{dt} = 0 \rightarrow \text{returns time for minimum distance}$$

$$t_{dmin} = -(x_p \cdot v_x + y_p \cdot v_y) / (v_x^2 + v_y^2)$$

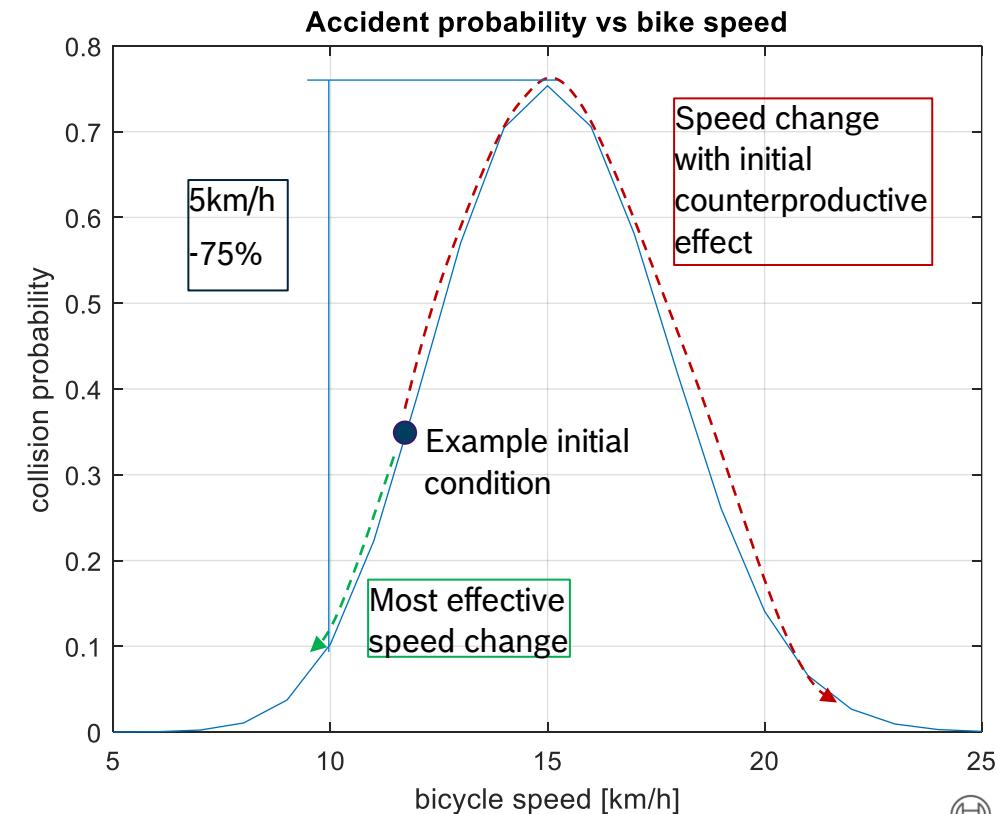
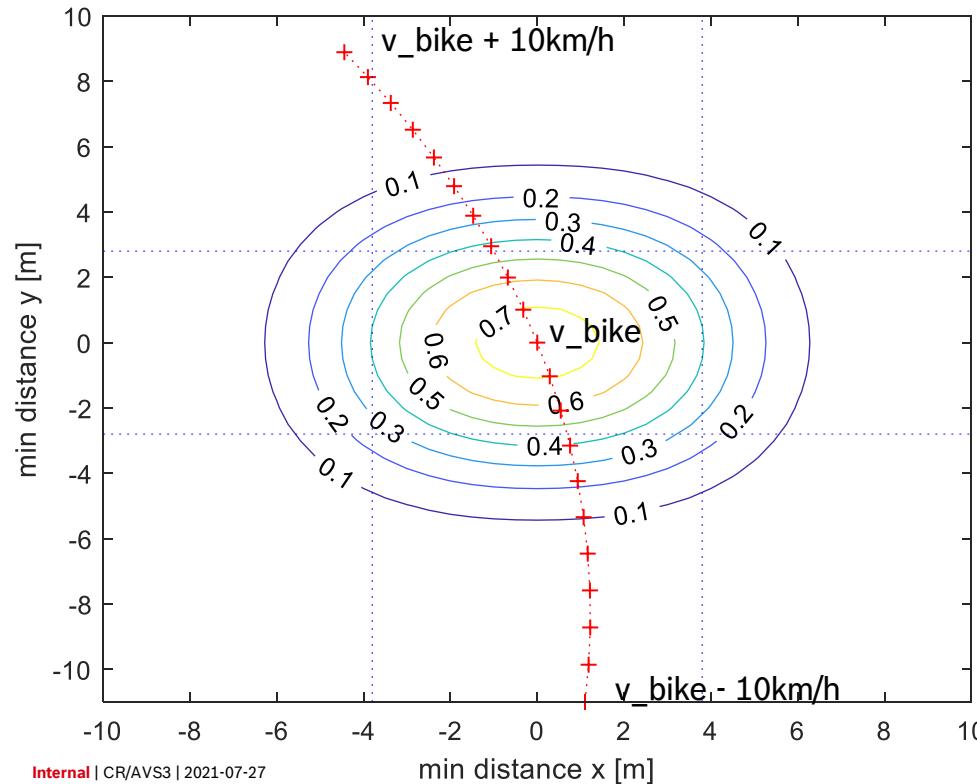
$$x_{car\,dmin} = (x_p \cdot v_y^2 - y_p \cdot v_x v_y) / (v_x^2 + v_y^2)$$

$$y_{car\,dmin} = x_{car\,dmin} \cdot \frac{v_y}{v_x}$$

Non imminent avoidable collision

Variation of collision probability with eBike speed

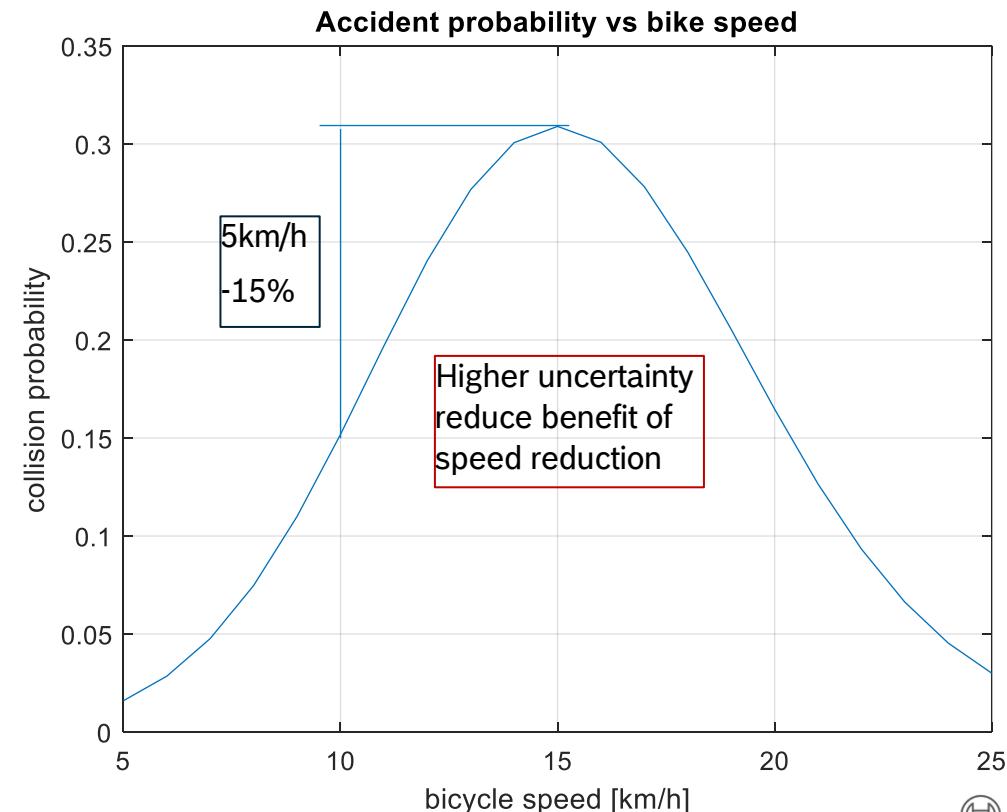
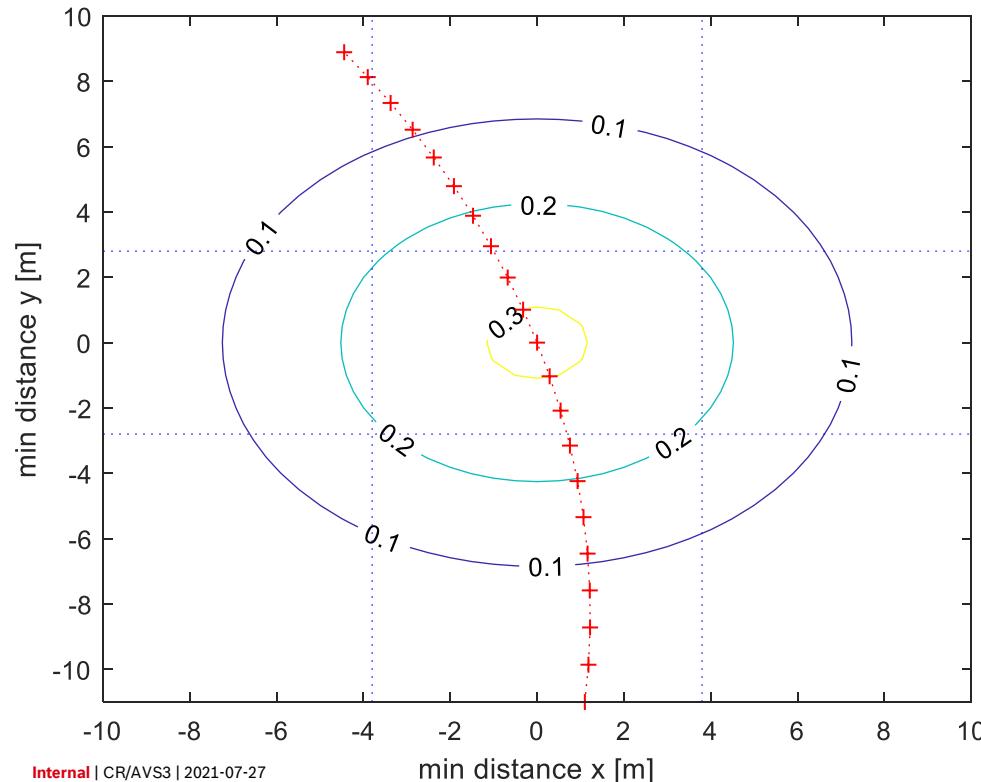
```
v_bike = 15/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 4; % time from minimum distance at which the speed variation is actuated [s]
sigma_pred = 1.5; % [m]
```



Non imminent avoidable collision

Variation of collision probability with eBike speed

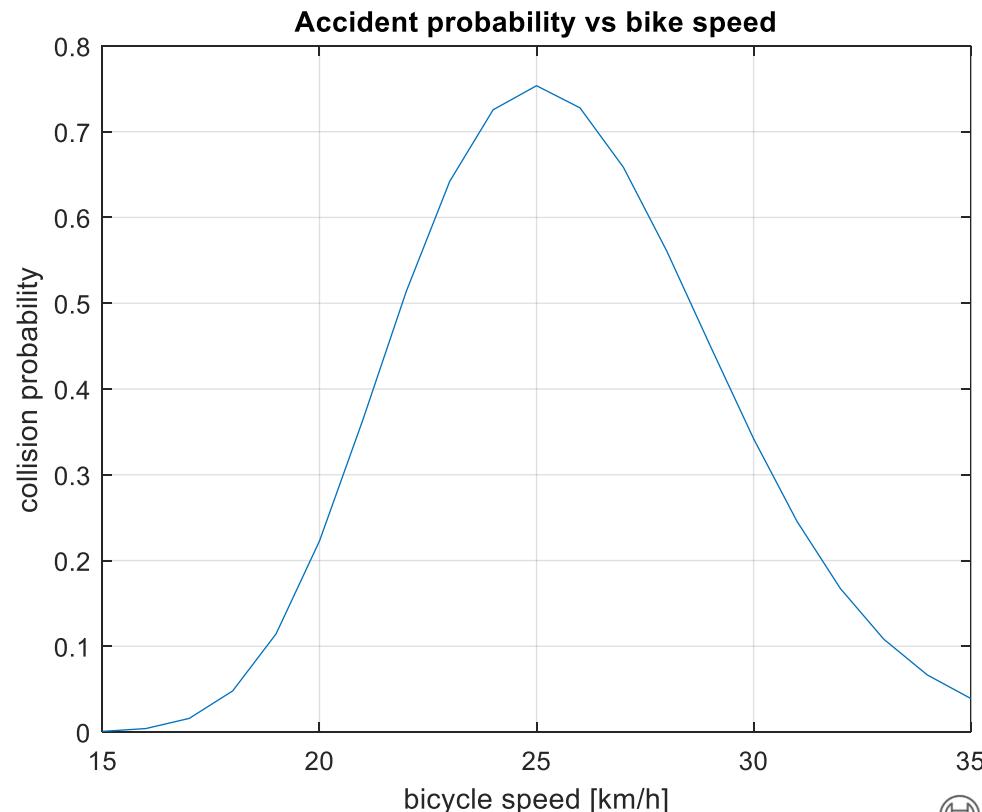
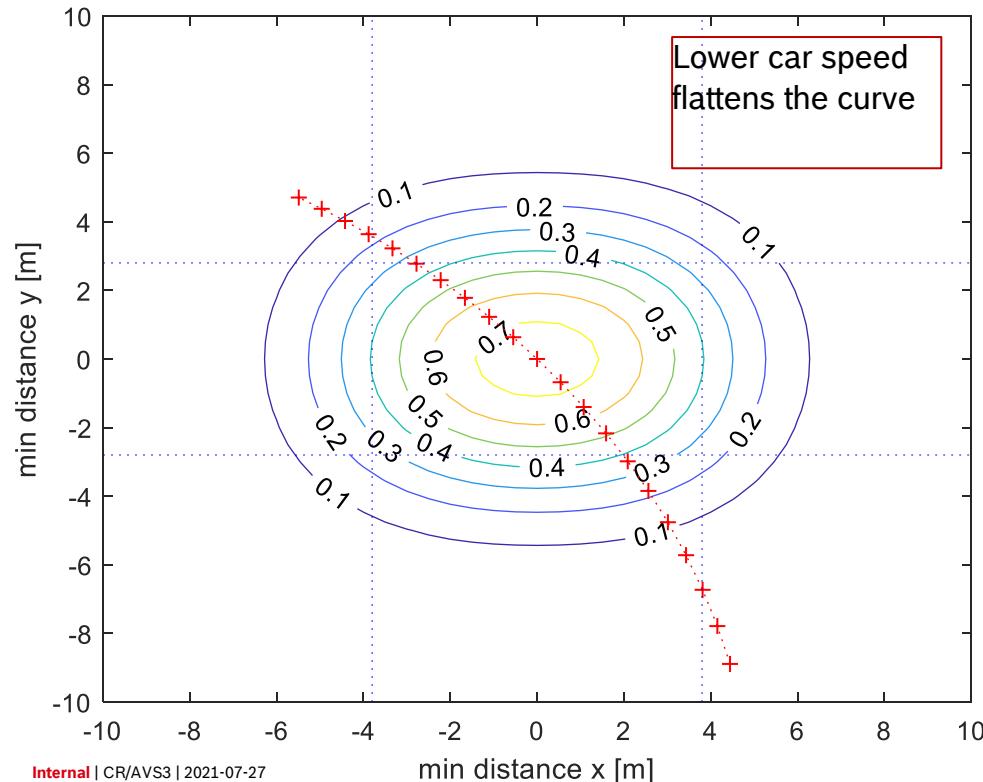
```
v_bike = 15/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 4; % time from minimum distance at which the speed variation is actuated [s]
sigma_pred = 3; % [m]
```



Non imminent avoidable collision

Variation of collision probability with eBike speed

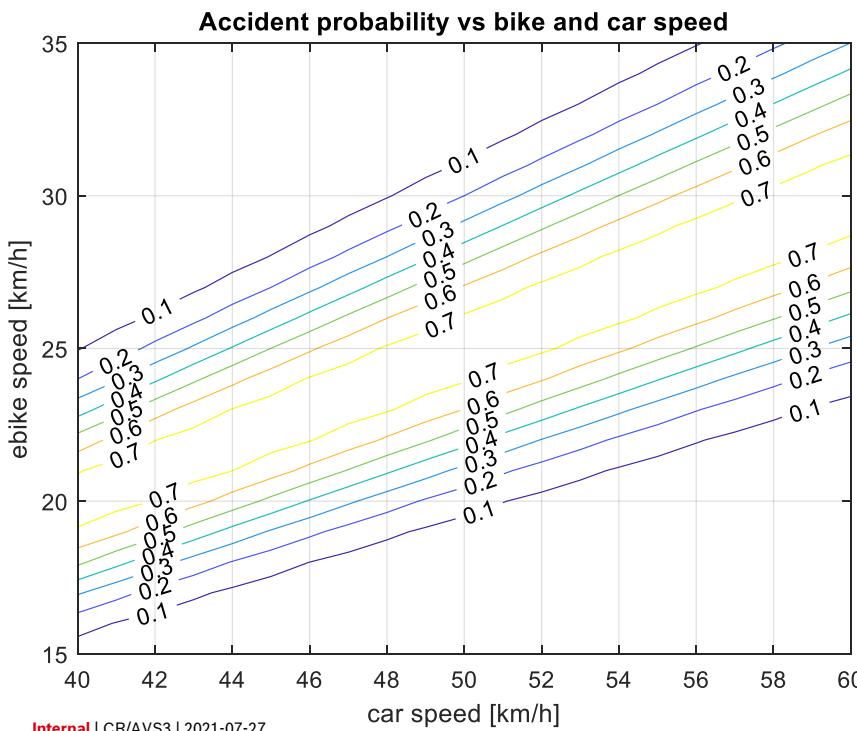
```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 30/3.6; % initial car speed [m/s]
TTC = 4; % time from minimum distance at which the speed variation is actuated [s]
sigma_pred = 1.5; % [m]
```



Non imminent avoidable collision

Collision Probability vs eBike and car speed

```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 4; % time from minimum distance at which the speed variation is actuated [s]
sigma_pred = 1.5; %[m]
```



Coordination of eBike and car speed

- ▶ depending on the quadrant a speed variation might have a negative influence
- ▶ an eBike speed variation can be compensated by a similar speed variation from car side

Improvement via communication

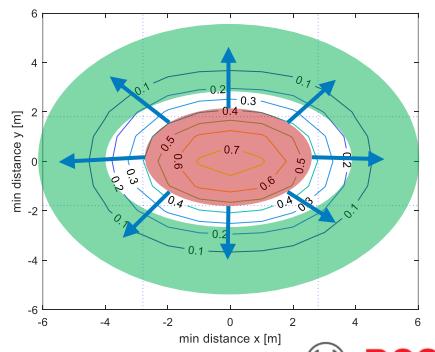
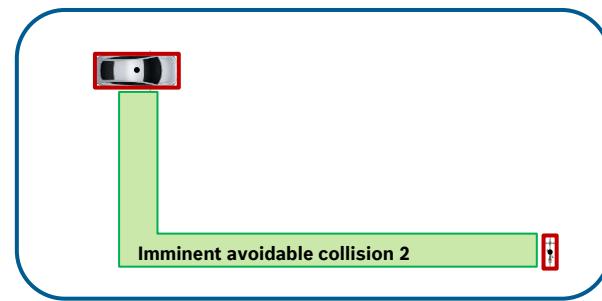
- ▶ CAM
 - ▶ Car sends trajectory based on most probable path which considers navigation data and past user routes
 - ▶ Exchange of speed suggestions to keep collision probability low
- ▶ VAM
 - ▶ Bicycle sends trajectory based on most probable path which considers navigation data and past user routes
 - ▶ Exchange of speed suggestions to keep collision probability low

Imminent avoidable collision 2

Non imminent avoidable collision 2

Intervention strategy

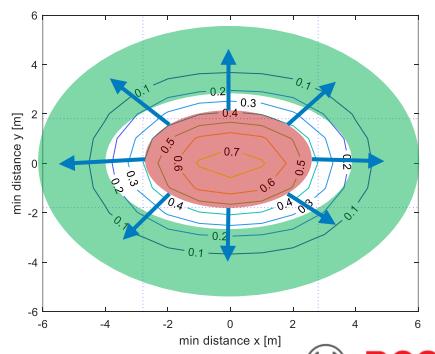
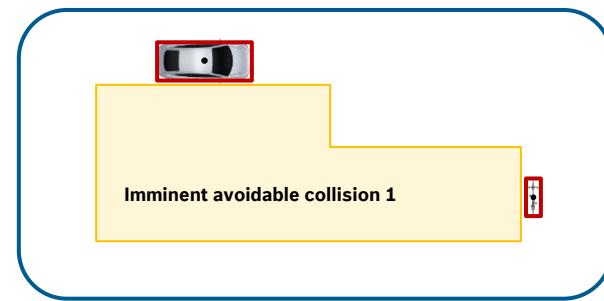
- ▶ Goals of the function:
 - ▶ Prevent imminent collision
 - ▶ Avoid false positive
- ▶ Time horizon:
 - ▶ It needs a stronger reaction from the rider → 1.5 - 2s (speed dependent)
 - ▶ Future trajectory still depends on the future rider's intention which is not yet observable
 - A most probable path calculation might be supported with navigation data or with past routes for car and ebike
 - Reasonable model: constant deceleration → information for the rider/driver: the current deceleration is insufficient to avoid the accident
- ▶ Intervention strategy:
 - ▶ Calculate minimum distance between car and ebike with constant deceleration model
 - The model doesn't know the future intention of the rider, since a turning manoeuvre is possible, only low urgency intervention are an option
 - ▶ calculate collision probability
 - ▶ if probability > threshold
 - calculate deceleration variation suggestion to reduce collision probability below a given value
 - depending on the deceleration value the system can tune urgency of intervention
 - coordinate speed variation of car and ebike



Non imminent avoidable collision 1

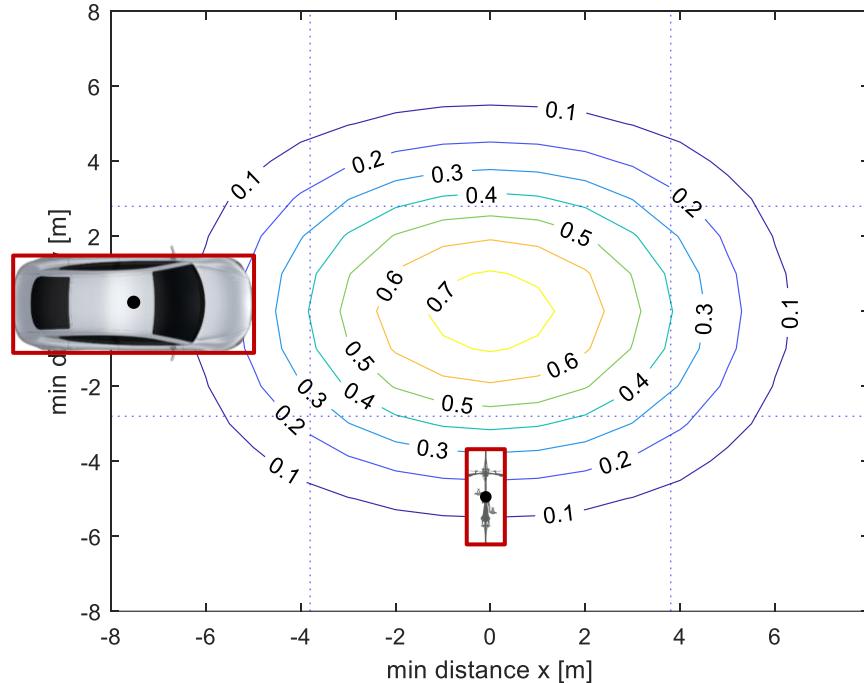
Intervention strategy

- ▶ Goals of the function:
 - ▶ Prevent imminent collision
 - ▶ Avoid false positive
- ▶ Time horizon:
 - ▶ It needs very strong reaction from the rider → 0 – 1.5s (speed dependent)
 - ▶ Future trajectory still depends on the future rider's intention which might be observable (in particular the intention to turn)
 - Path prediction based on physical model supported by ML (possible support with navigation data and previous paths)
 - A constant deceleration model enables to find the time frame at which the intervention should have high urgency given the current speed and decelerations of the vehicles
- ▶ Intervention strategy:
 - ▶ Calculate minimum distance between car and ebike with ML model
 - Due to the higher certainty of the prediction, the system can release high urgency warnings
 - ▶ calculate collision probability
 - ▶ if probability > threshold
 - calculate deceleration variation suggestion to reduce collision probability below a given value
 - depending on the deceleration value the system can tune urgency of intervention
 - coordinate speed variation of car and ebike



Non imminent avoidable collision
Calculation of minimum distance with constant a model

- Deriving distance² between car and bike results in a 3rd order equation



```

a*t^3 + b*t^2 + c*t + d = 1

a = (a_car^2 + a_bike^2);
b = 3*(v_car_init * a_car + v_bike_init * a_bike);
c = 2 * (x_car_init * a_car + y_bike_init * a_bike);
d = 2 * (x_car_init * v_car_init + y_bike_init * v_bike_init);

x1 = (((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3) - b/(3*a) - (-b^2/(9*a^2) + c/(3*a))/(((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (c/(3*a) - b^2/(9*a^2))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3);
x2 = z2^2/(9*a^2)^(1/2)*((2*((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (c/(3*a) - b^2/(9*a^2))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)) - (3^(1/2)*(1 - (-b^2/(9*a^2) + c/(3*a))/(((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (c/(3*a) - b^2/(9*a^2))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2)))^(1/3) + (((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3) + ((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)/2 - b/(3*a) - (((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)/2;
x3 = (-b^2/(9*a^2) + c/(3*a))/((2*((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (c/(3*a) - b^2/(9*a^2))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)) + (3^(1/2)*(1 - (-b^2/(9*a^2) + c/(3*a))/(((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (c/(3*a) - b^2/(9*a^2))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2)))^(1/3) + ((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)*1/2 - b/(3*a) - (((d/(2*a) + b^3/(27*a^3) - (b*c)/(6*a^2))^2 + (-b^2/(9*a^2) + c/(3*a))^3)^(1/2) - b^3/(27*a^3) - d/(2*a) + (b*c)/(6*a^2))^(1/3)/2;

x_car_p_max_1 = x_car_init + v_car_init * x1 + 1/2 * a_car * x1^2;
y_bike_p_max_1 = y_bike_init + v_bike_init * x1 + 1/2 * a_bike * x1^2;

x_car_p_max_2 = x_car_init + v_car_init * x2 + 1/2 * a_car * x2^2;
y_bike_p_max_2 = y_bike_init + v_bike_init * x2 + 1/2 * a_bike * x2^2;

x_car_p_max_3 = x_car_init + v_car_init * x3 + 1/2 * a_car * x3^2;
y_bike_p_max_3 = y_bike_init + v_bike_init * x3 + 1/2 * a_bike * x3^2;

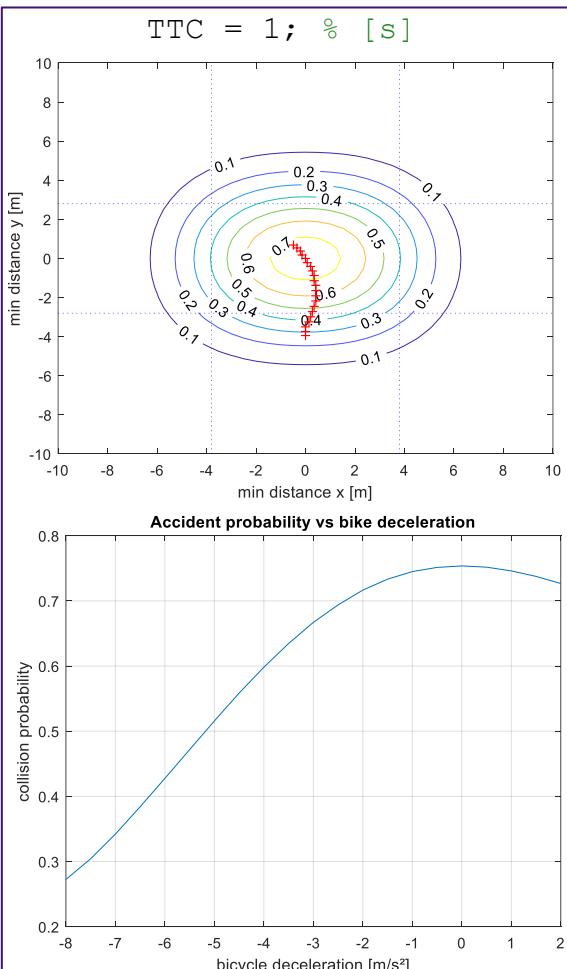
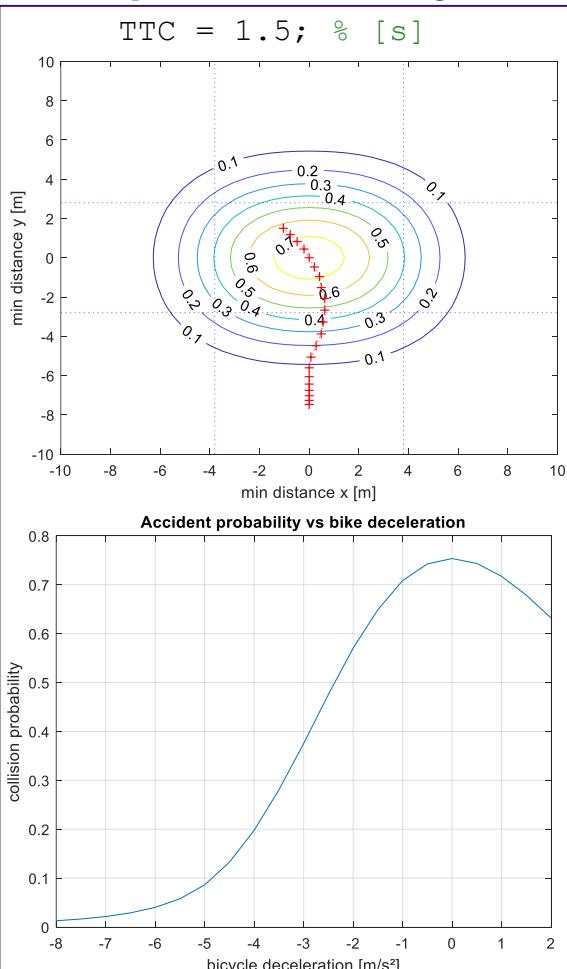
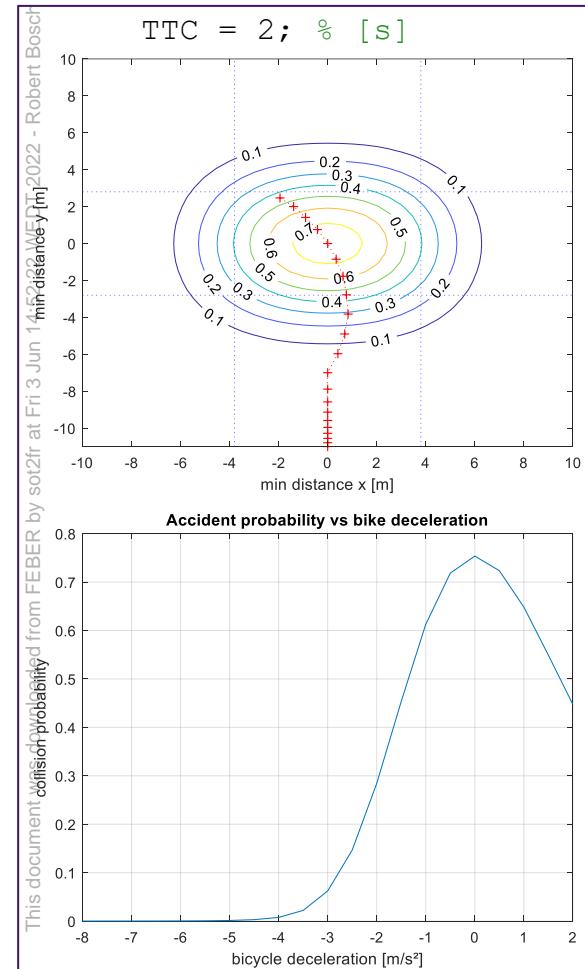
```

- In a constant a model it is not possible to represent in a continuous equation the standing still of the vehicle after reaching 0km/h → the minimum value resulting from the equation is in some situations incorrect → Problem solved numerically

Non imminent avoidable collision

Variation of collision probability with eBike Deceleration

$v_{\text{bike}} = 25/3.6$; % initial bike speed [m/s]
 $v_{\text{car}} = 50/3.6$; % initial car speed [m/s]
 $\sigma_{\text{pred}} = 1.5$; [%]



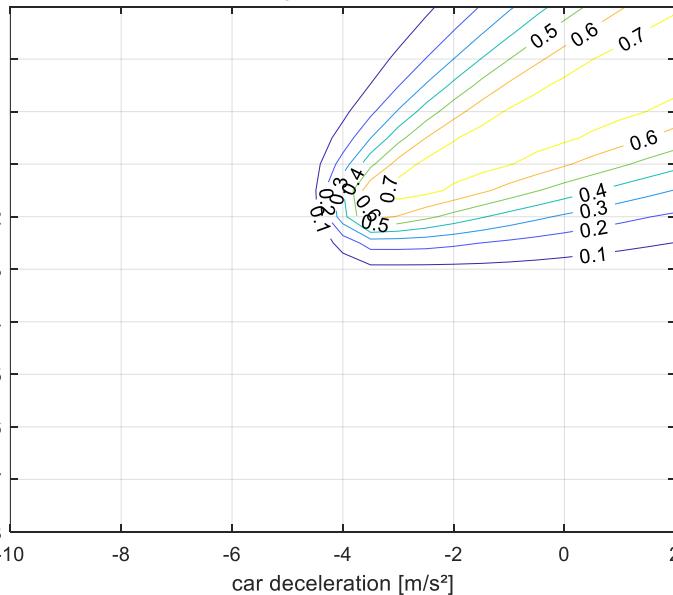
- Timing of “intervention” defines the necessary deceleration to avoid impact
- For a given deceleration the timing impacts the effectiveness of the braking in reducing accident probability

Non imminent avoidable collision

Variation of collision probability with eBike/car Deceleration

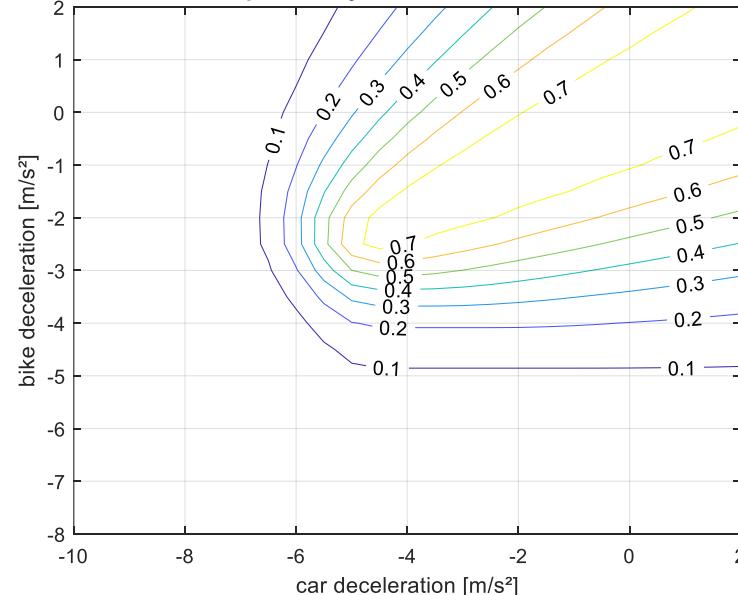
```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 2; % [s]
sigma_pred = 1.5; % [m]
```

Accident probability vs bike and car deceleration



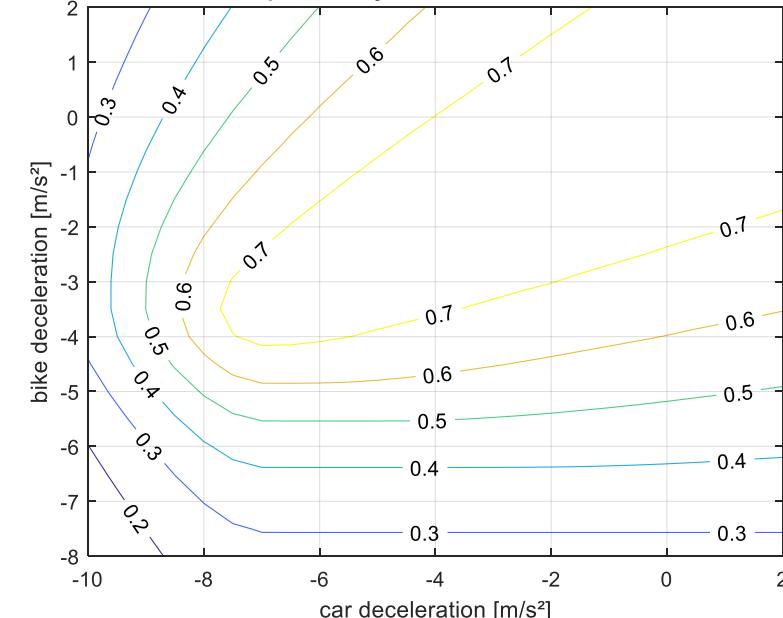
```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 1.5; % [s]
sigma_pred = 1.5; % [m]
```

Accident probability vs bike and car deceleration



```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 1.5; % [s]
sigma_pred = 1; % [m]
```

Accident probability vs bike and car deceleration

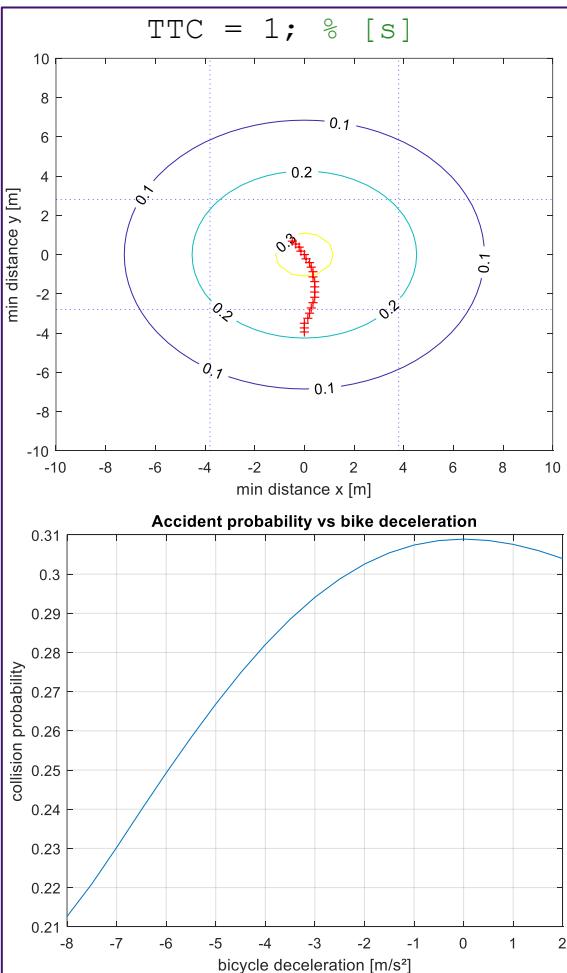
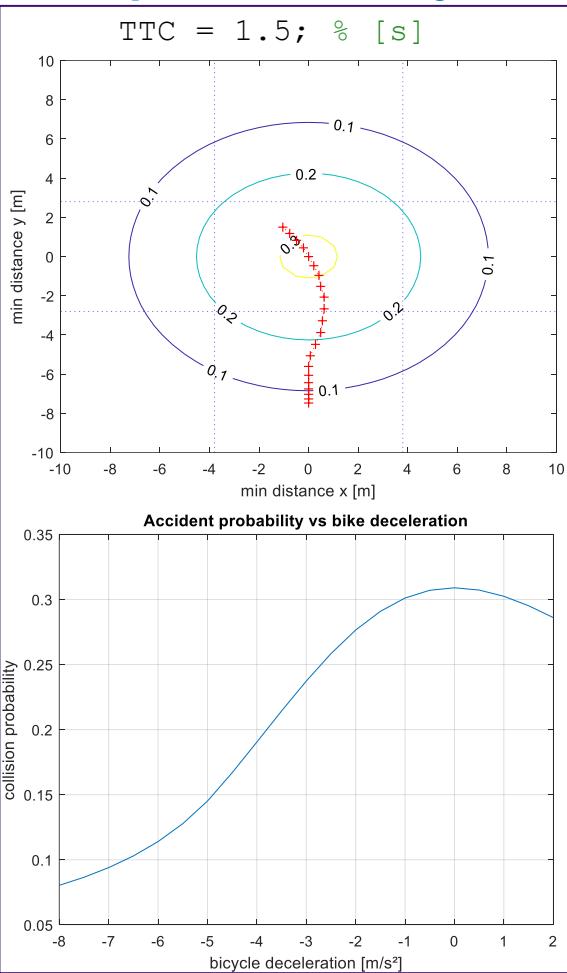
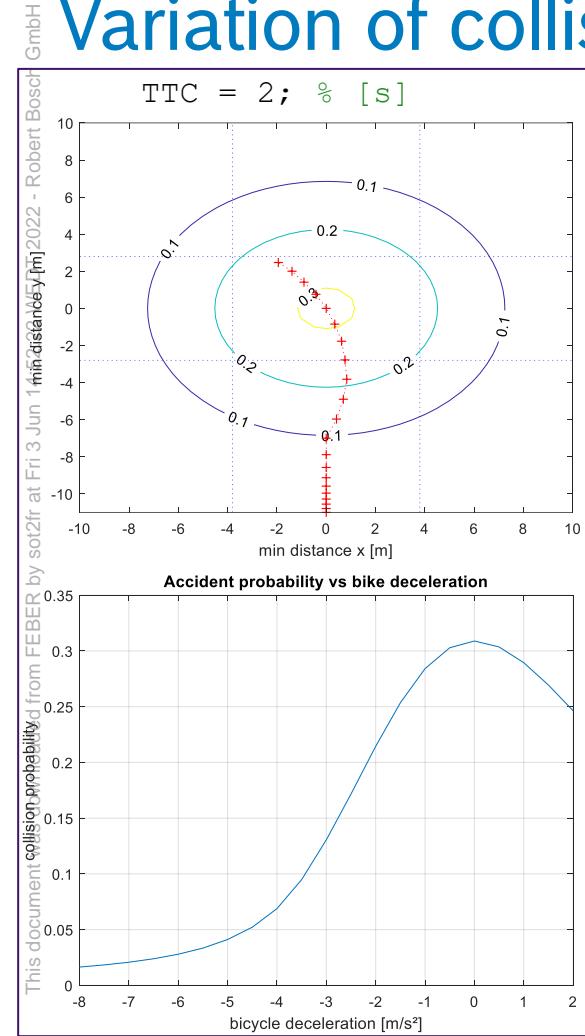


- ▶ For low bike deceleration, low car deceleration has a negative impact on collision probability → bike stops in collision area
- ▶ For high bike deceleration, car deceleration only has a positive impact → bike stops before collision area

Non imminent avoidable collision

Variation of collision probability with eBike Deceleration

$v_{\text{bike}} = 25/3.6; \text{ % initial bike speed [m/s]}$
 $v_{\text{car}} = 50/3.6; \text{ % initial car speed [m/s]}$
 $\sigma_{\text{pred}} = 3; \text{ [%]}$



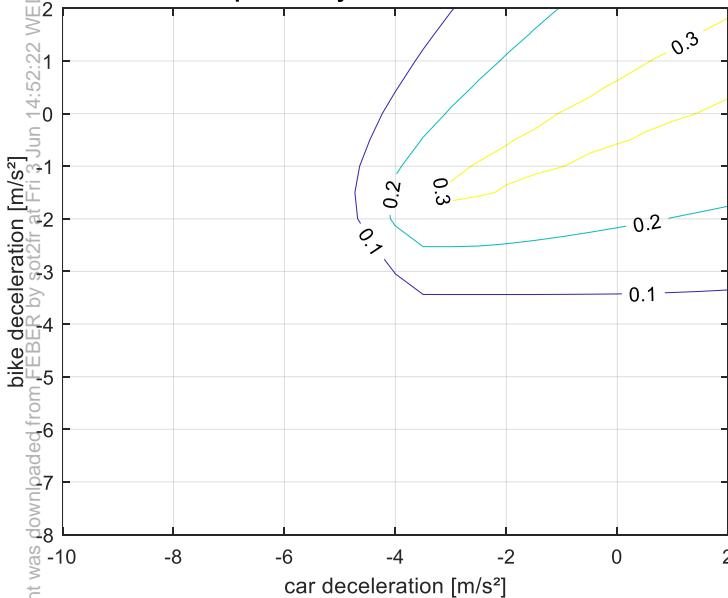
- Timing of “intervention” defines the necessary deceleration to avoid impact
- For a given deceleration the timing impacts the effectiveness of the braking in reducing accident probability

Non imminent avoidable collision

Variation of collision probability with eBike/car Deceleration

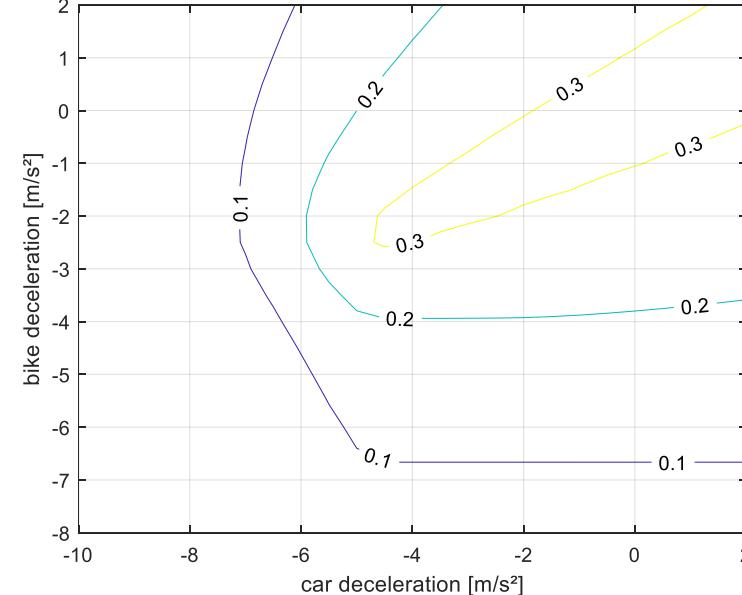
```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 2; % [s]
sigma_pred = 3; % [m]
```

Accident probability vs bike and car deceleration



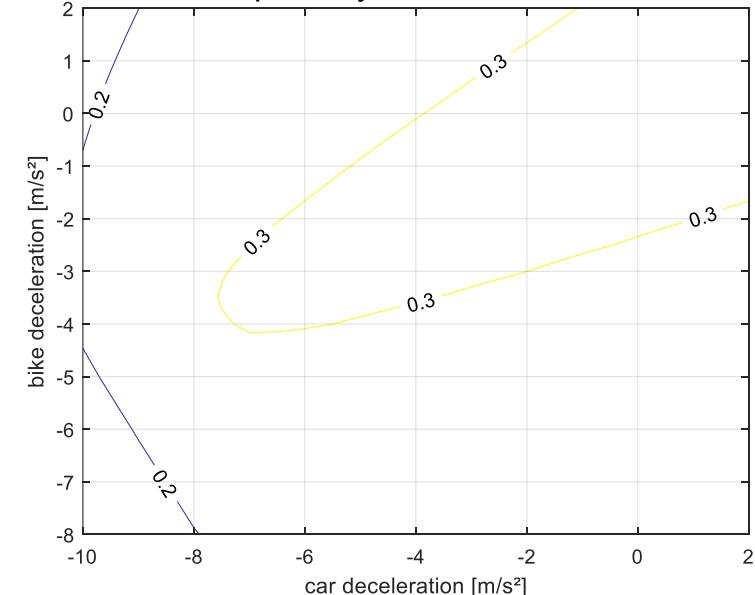
```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 1.5; % [s]
sigma_pred = 3; % [m]
```

Accident probability vs bike and car deceleration



```
v_bike = 25/3.6; % initial bike speed [m/s]
v_car = 50/3.6; % initial car speed [m/s]
TTC = 1.5; % [s]
sigma_pred = 3; % [m]
```

Accident probability vs bike and car deceleration

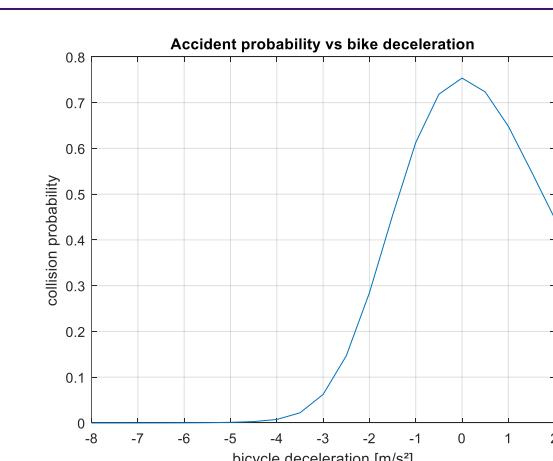


- ▶ For low bike deceleration, low car deceleration has a negative impact on collision probability → bike stops in collision area
- ▶ For high bike deceleration, car deceleration only has a positive impact → bike stops before collision area

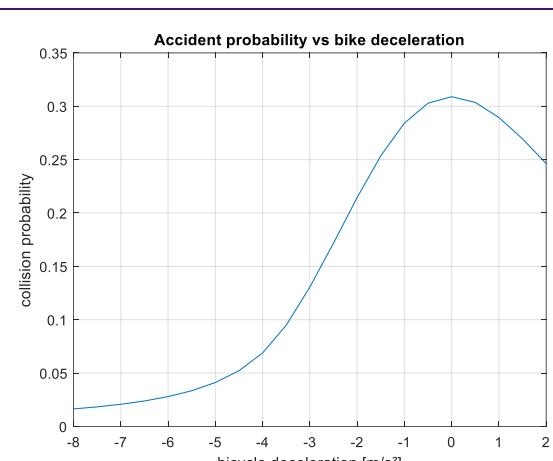
Non imminent avoidable collision

Effect of uncertainty

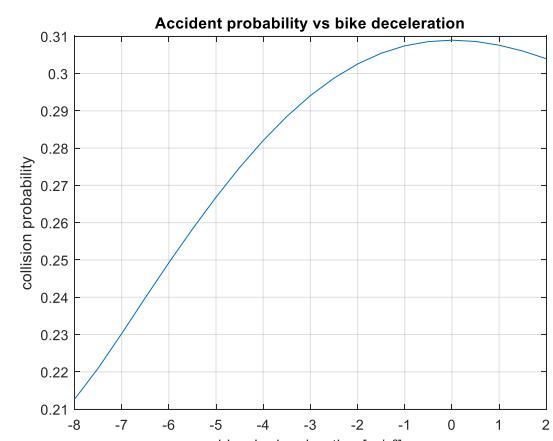
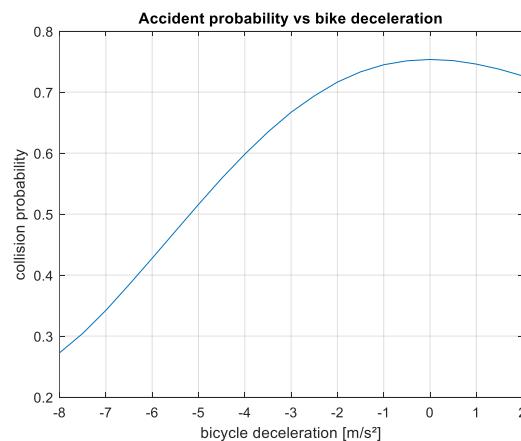
$\Sigma_{pred} = 1.5m$



$\Sigma_{pred} = 3m$



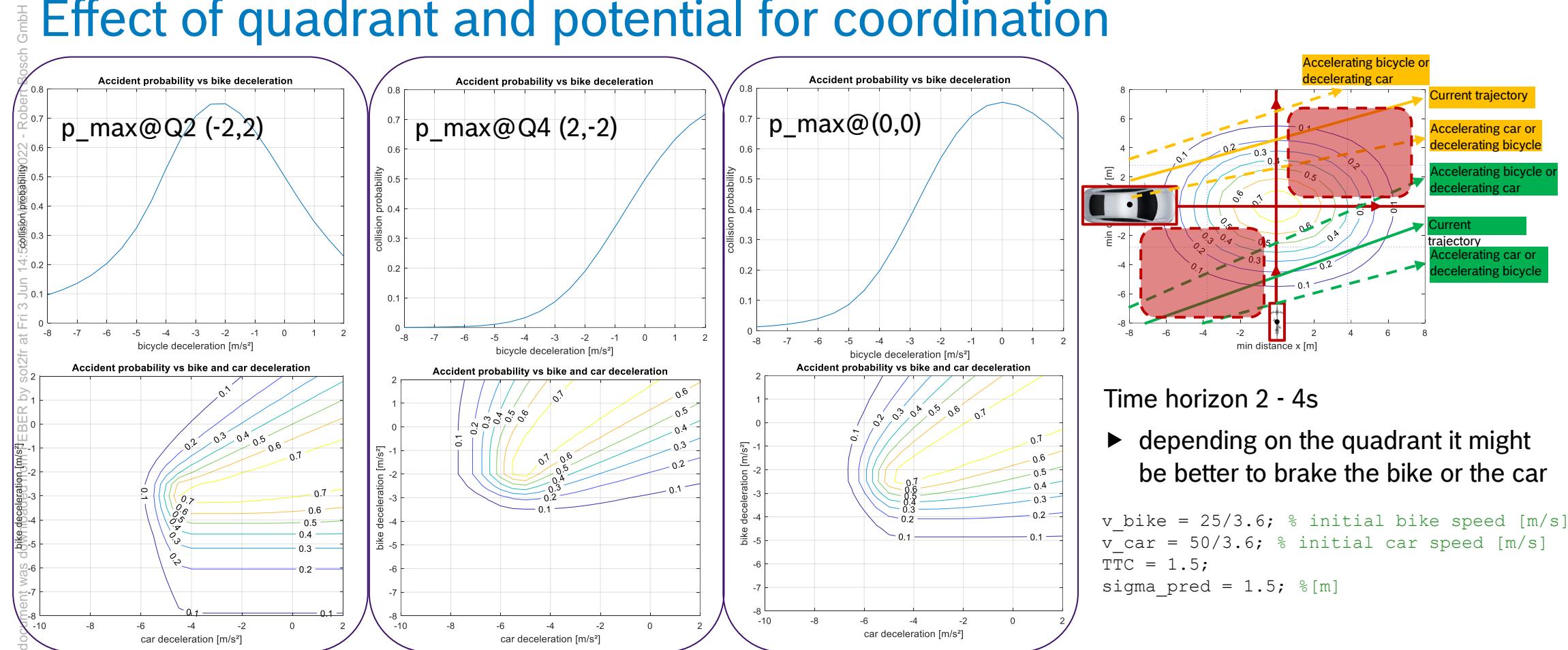
$TTC = 1s$



- ▶ Uncertainty has a very high impact on the false positive of the functionality
- ▶ The minimum probability value is similar as in the case with low sigma for a given deceleration, the overall efficacy of a braking in the reduction of probability reduction is significantly less

Non imminent avoidable collision

Effect of quadrant and potential for coordination



LATERAL RIDING CYCLIST

Potential of bicycle warning vs car warning

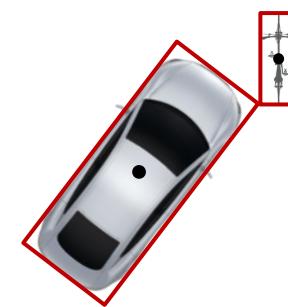
Coordination based on TTC Representation

- ▶ Goal: identify in which situations a warning for the eBike rider can improve his safety
- ▶ Intervention type: AEB on car, rider warning on bicycle
- ▶ Assumptions:
 - ▶ in order to minimize false negatives, the intervention should be activated for the vehicle with the shorter braking time $\min(TTC_{bike}, TTC_{car}) = \text{latest } TTC$ at which the intervention should be activated in order to avoid the accident
 - ▶ car AEB only activated when the accident is unavoidable. For situation where $TTC_{car} < TTC_{bike}$ then the accident is unavoidable by only activating AEB
- ▶ We consider the below three collision point scenarios

Case 1 - Minimum overlap



Case 2 – Car front



Case 3 – Car side



Coordination based on TTC Representation

- In order to consider the effect of path prediction accuracy and its usage in the coordination of the warnings we perform the simulation with 2 different parameter sets:

Scenario a: aggressive AEB strategy

- the eBike path prediction can predict braking 0.5s in advance and it is used by the AEB in the calculation of the TTC_{bike} to release an earlier AEB intervention
- assumption: 0.5 is also the reaction time of the rider to a warning. This enables to consider TTC_{bike} at which the warning to the eBike rider is released equal to avoid warning overlap
- more conservative eBike deceleration and aggressive car deceleration

```
% bike_car_warning_coordination_based_on_TTC.m
% Simulation Parameters
t.react_bike = 0.5; %[s] reaction time of rider = prediction time for braking
ttl_AEB_car = 0.2; %[s] car time to lock
a_car = 9.8; %[m/s^2] deceleration of car during AEB braking
a_bike = 6; %[m/s^2] deceleration of bike after warning
```

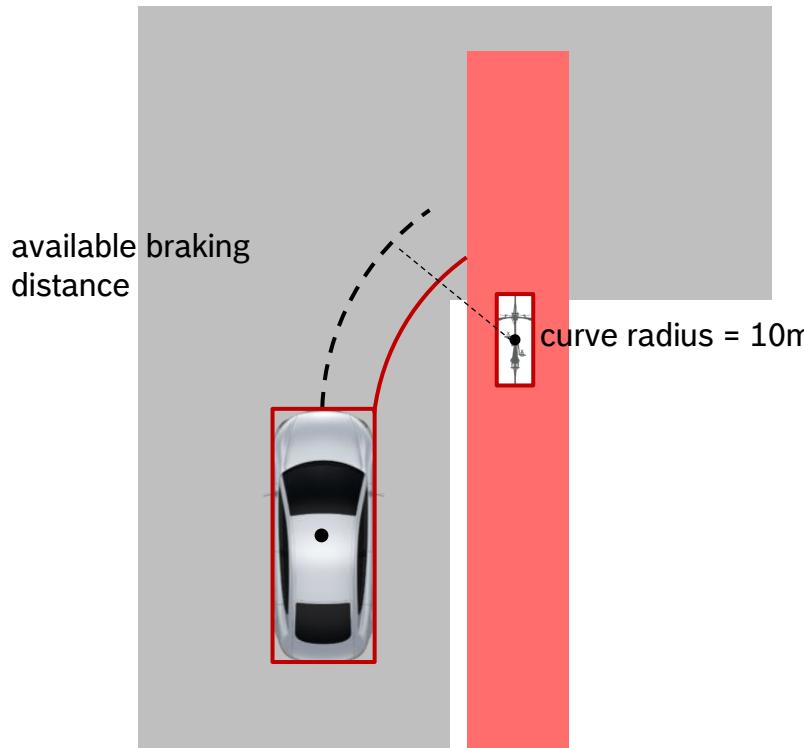
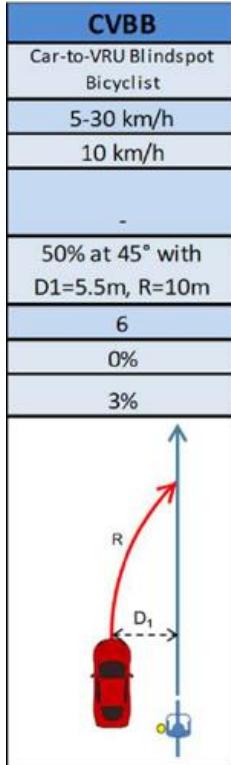
Scenario b: conservative AEB strategy

- the eBike path prediction is not used by the AEB logic in the calculation of TTC_{bike} , the AEB logic calculates TTC_{bike} only considering a deceleration of 7m/s^2
- more aggressive eBike deceleration and conservative car deceleration

```
% bike_car_warning_coordination_based_on_TTC.m
% Simulation Parameters
t.react_bike = 0; %[s] reaction time of rider = prediction time for braking
ttl_AEB_car = 0.2; %[s] car time to lock
a_car = 9; %[m/s^2] deceleration of car during AEB braking
a_bike = 7; %[m/s^2] deceleration of bike after warning
```

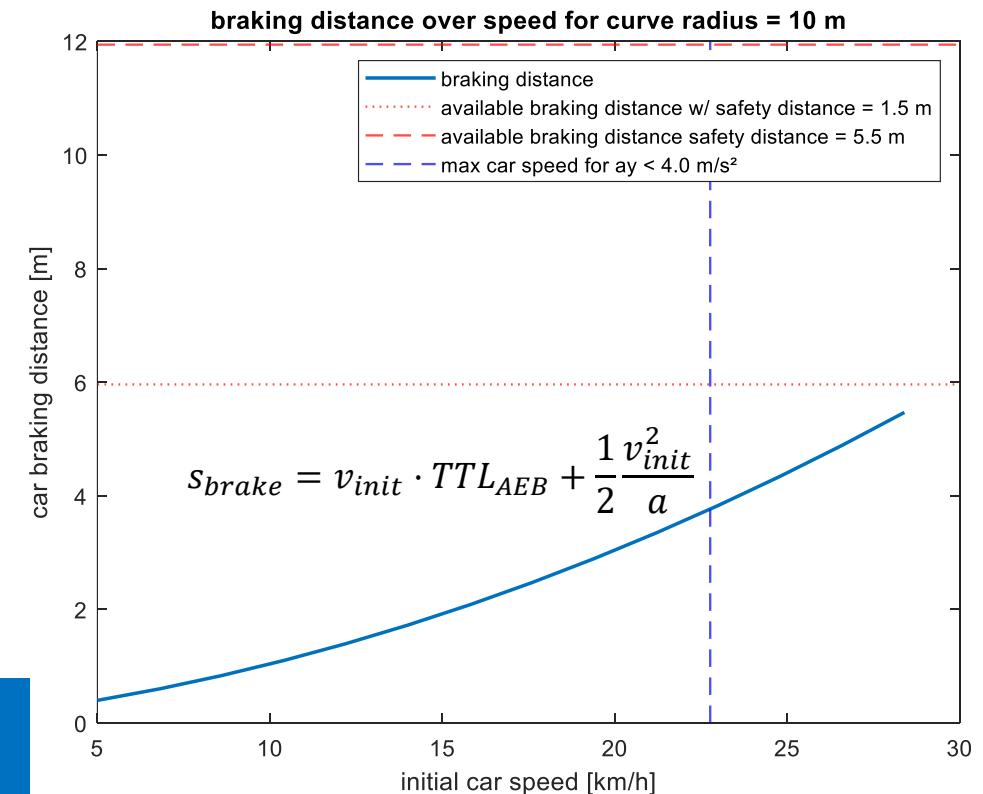
System Architectures

Car Turning braking distance



For the defined parameter, right after turn start, the car can always brake into still stand before crossing eBike trajectory

$ax_{AEB} = 8; \text{ [m/s}^2]$ maximum deceleration with AEB
 $turn_radius = 10; \text{ [m]}$ minimum curve radius from CATS scenarios
 $t_{react_AEB} = 0.2; \text{ [s]}$ AEB TTL



Coordination based on TTC

Case 1 – Minimum Overlap



- The TTC at which the intervention should be activated only depends on the braking distance and current speed of both vehicles

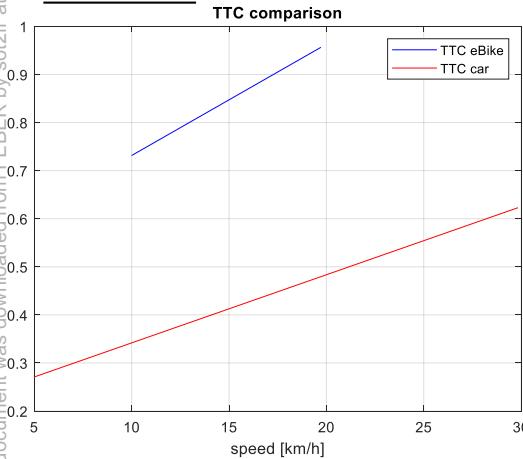
$$s_{\text{brake_car}} = v_{\text{car}} * \text{ttl_AEB_car} + 1/2 * v_{\text{car}}^2 / a_{\text{car}};$$

$$\text{TTC}_{\text{car}} = (s_{\text{brake_car}}) / v_{\text{car}}$$

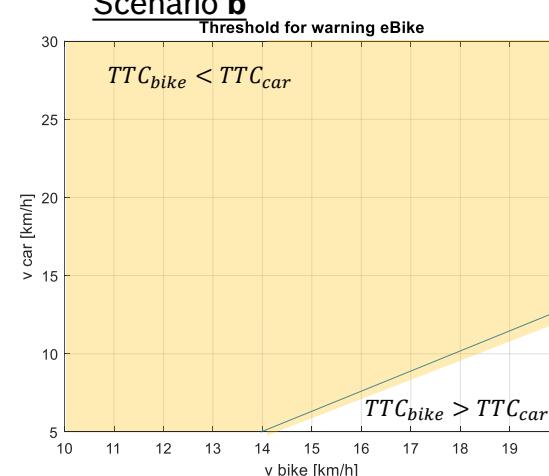
$$s_{\text{brake_bike}} = v_{\text{bike}} * t_{\text{react_bike}} + 1/2 * v_{\text{bike}}^2 / a_{\text{bike}};$$

$$\text{TTC}_{\text{bike}} = (s_{\text{brake_bike}}) / v_{\text{bike}}$$

Scenario a



Scenario b

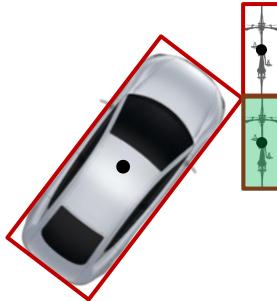


- The orange area shows the speed range for which the bike has the lower TTC. In this area the accident cannot be avoided w/o rider warning
- In case the car AEB waits until the remaining time has fallen below TTC_{car} , then the accident can only be avoided by warning the eBike rider (until the remaining time falls below TTC_{bike})

Case 1 shows benefit in warning the eBike rider only for scenario b

Coordination based on TTC

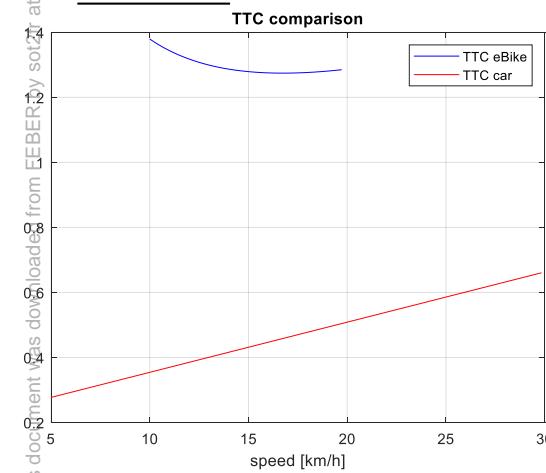
Case 2 – Car Front



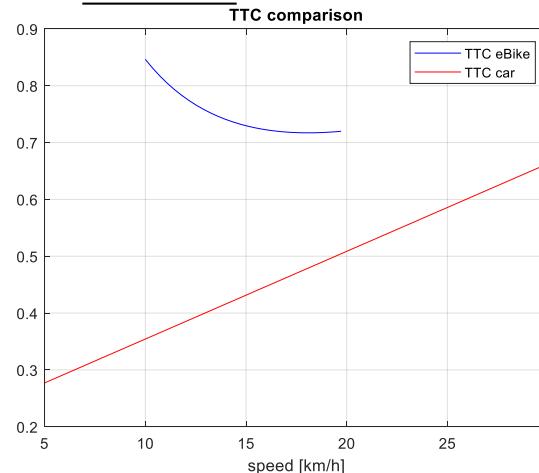
- ▶ The TTC at which the intervention should be activated only depends on the braking distance and current speed of both vehicles
- ▶ For the bicycle it additionally depends on vehicle dimensions

```
s_brake_car = v_car * ttl_AEB_car + 1/2 * v_car^2/ a_car;  
TTC_car = (s_brake_car) / v_car  
  
s_brake_bike = v_bike * t_react_bike + 1/2 * v_bike^2/ a_bike;  
TTC_bike = (s_brake_bike + lb) / v_bike
```

Scenario a



Scenario b

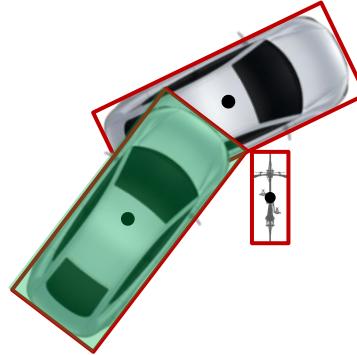


- ▶ The orange area shows the speed range for which the bike has the lower TTC. In this area the accident cannot be avoided w/o rider warning
- ▶ In case the car AEB waits until the remaining time has fallen below TTC_{car} , then the accident can only be avoided by warning the eBike rider (until the remaining time falls below TTC_{bike})

Case 2 shows no benefit in warning the eBike rider

Coordination based on TTC

Case 2 – Car Front



- ▶ The TTC at which the intervention should be activated only depends on the braking distance and current speed of both vehicles
- ▶ For the car it additionally depends on vehicle dimensions

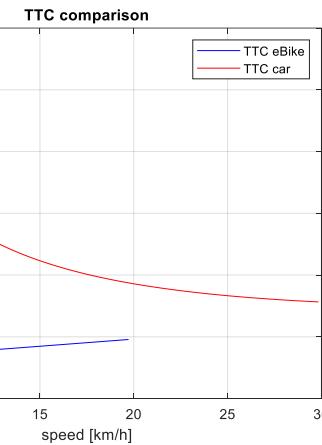
$$s_{\text{brake_car}} = v_{\text{car}} * \text{ttl_AEB_car} + 1/2 * v_{\text{car}}^2 / a_{\text{car}};$$

$$\text{TTC}_{\text{car}} = (s_{\text{brake_car}} + l_a/2) / v_{\text{car}}$$

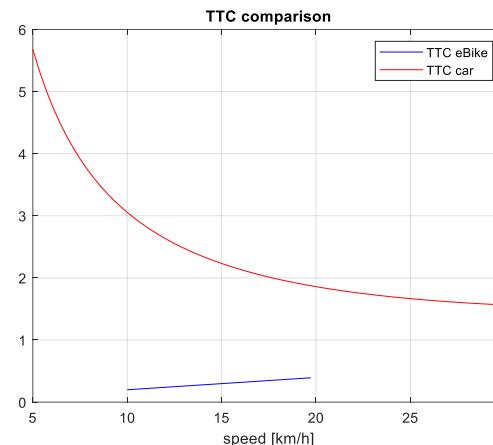
$$s_{\text{brake_bike}} = v_{\text{bike}} * t_{\text{react_bike}} + 1/2 * v_{\text{bike}}^2 / a_{\text{bike}};$$

$$\text{TTC}_{\text{bike}} = (s_{\text{brake_bike}}) / v_{\text{bike}}$$

Scenario a



Scenario b



- ▶ The orange area shows the speed range for which the bike has the lower TTC. In this area the accident cannot be avoided w/o rider warning
- ▶ In case the car AEB waits until the remaining time has fallen below TTC_{car} , then the accident can only be avoided by warning the eBike rider (until the remaining time falls below TTC_{bike})

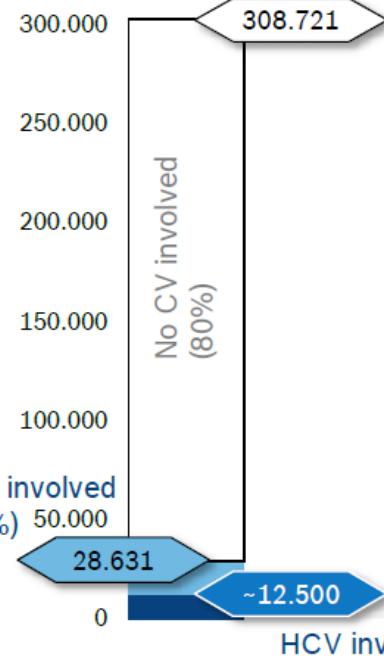
Case 3 shows high benefit in warning the eBike rider

Turning truck use case

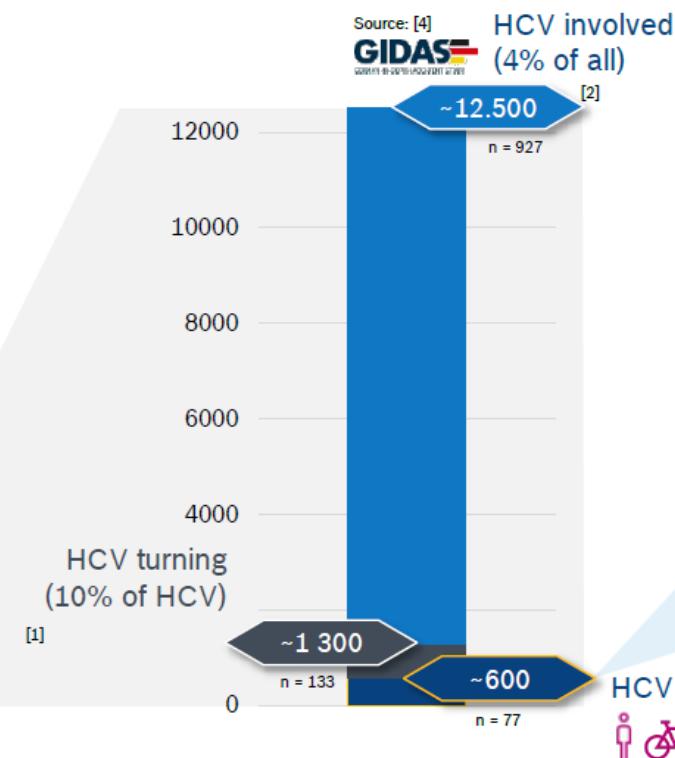
Heavy Commercial Vehicle (HCV >12t)

Turning conflict with VRU involvement

Source: Federal Statistical Office [1,2,3]



Source: [4]



HCV involved
(4% of all)

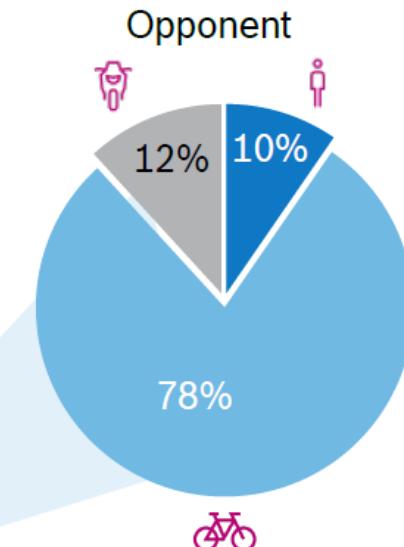
[2]

n = 927

n = 77



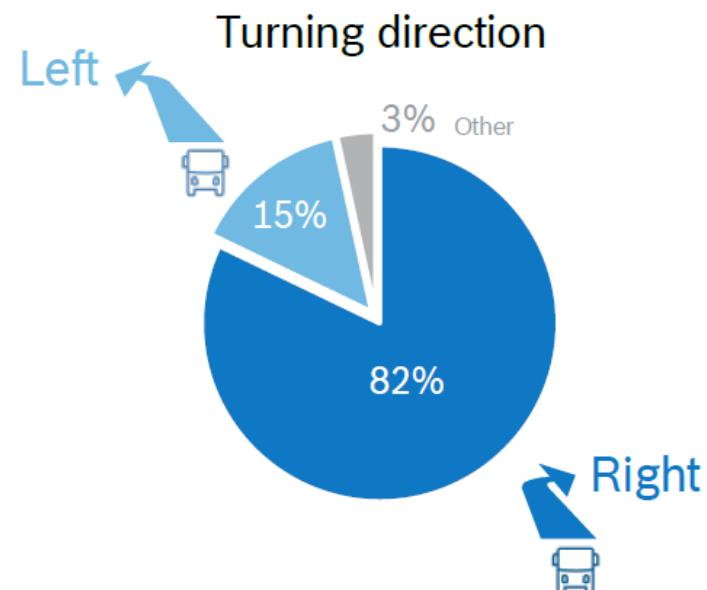
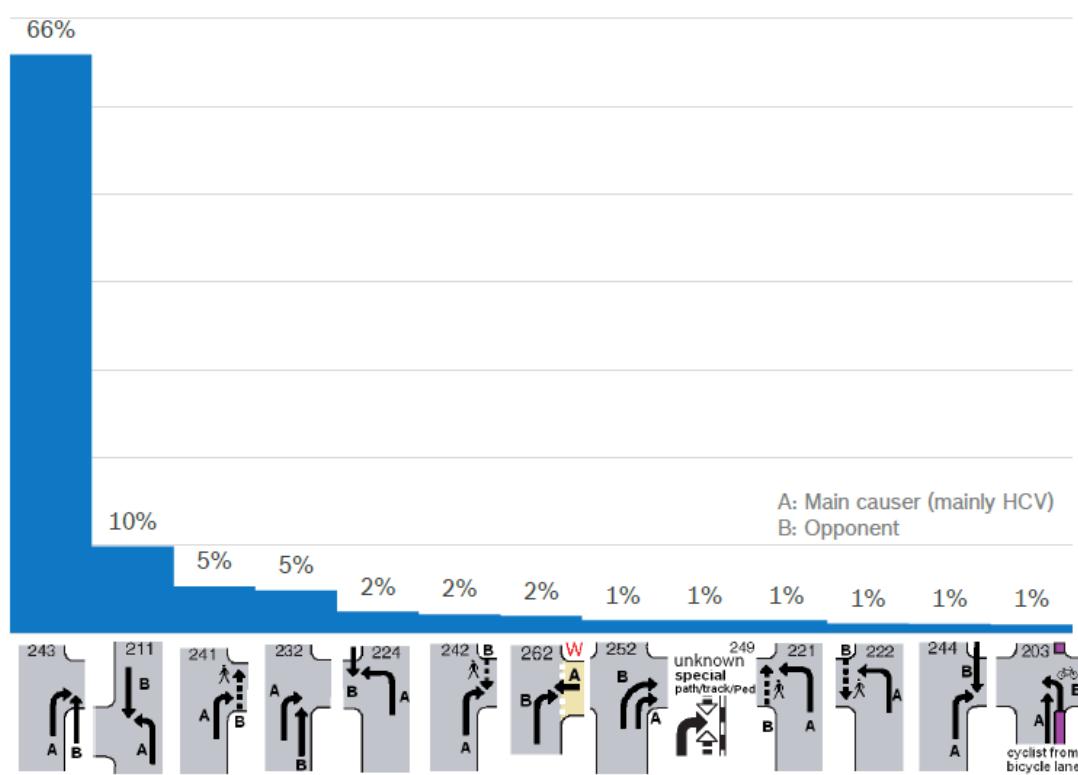
HCV turning conflict with VRU (5% of HCV)



- ▶ Number of HCV turning conflicts with VRU ~600 (5% of all HCV accidents with casualties)
- ▶ Main share are conflicts with bicyclists

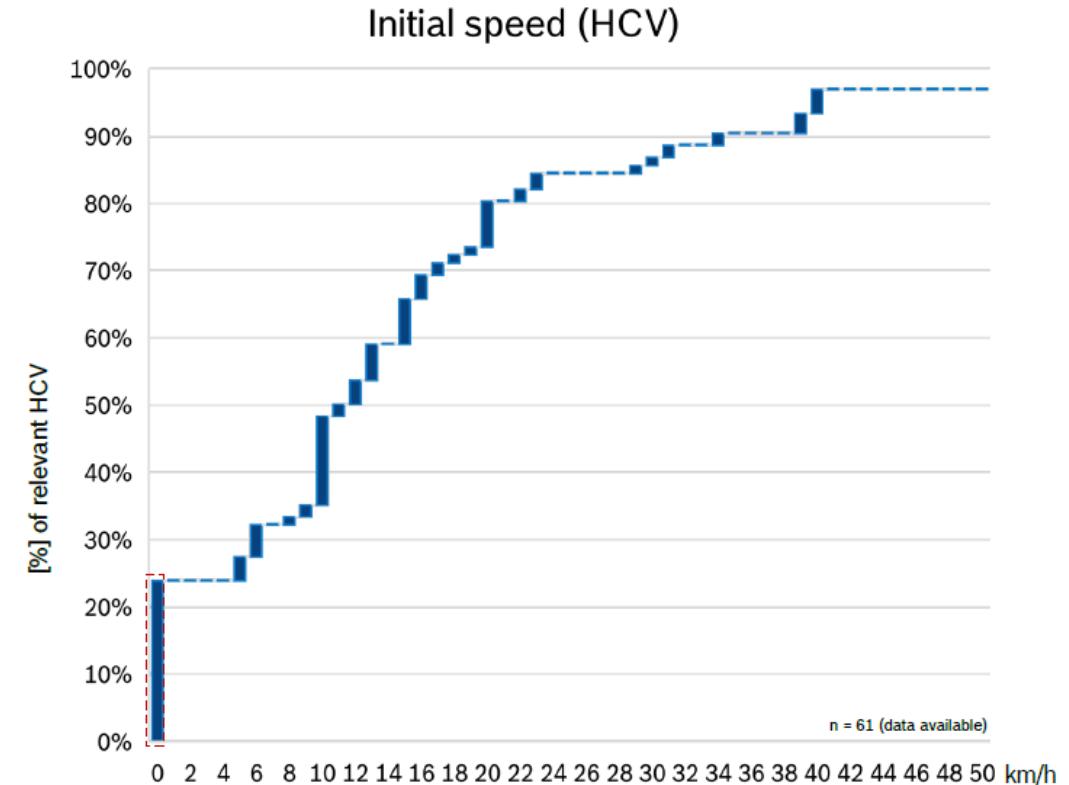
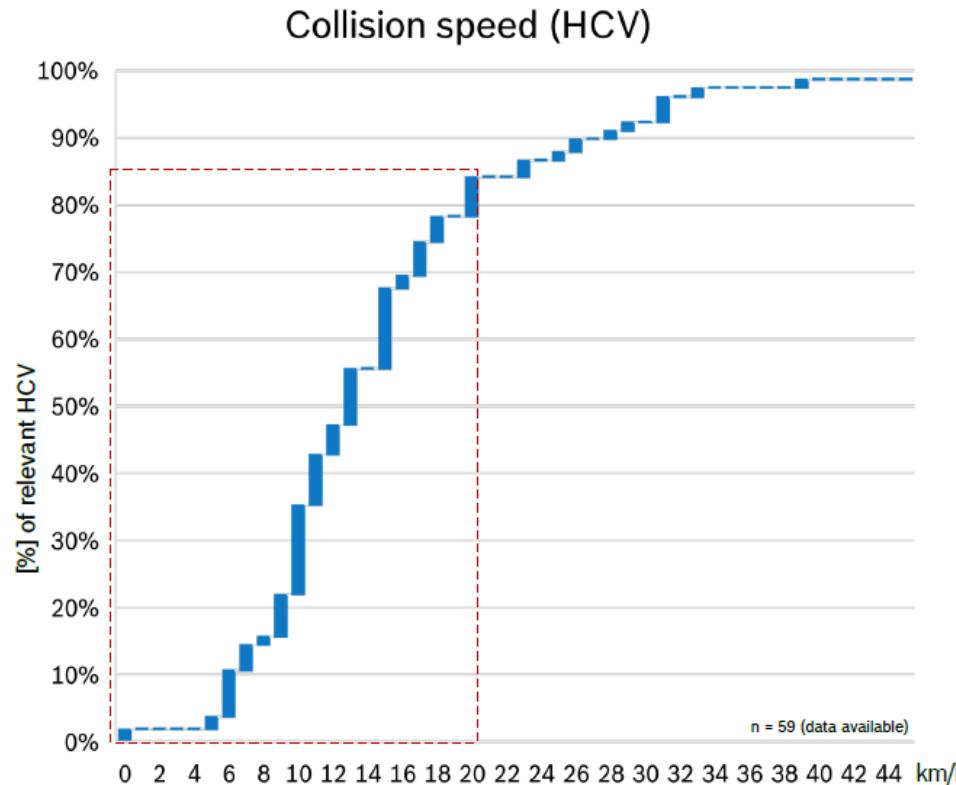
Heavy Commercial Vehicle (HCV >12t)

Turning conflict with VRU involvement – traffic situation



- Main conflicts (84%) while turning to the right (majority of opponents of the conflict are cyclists)

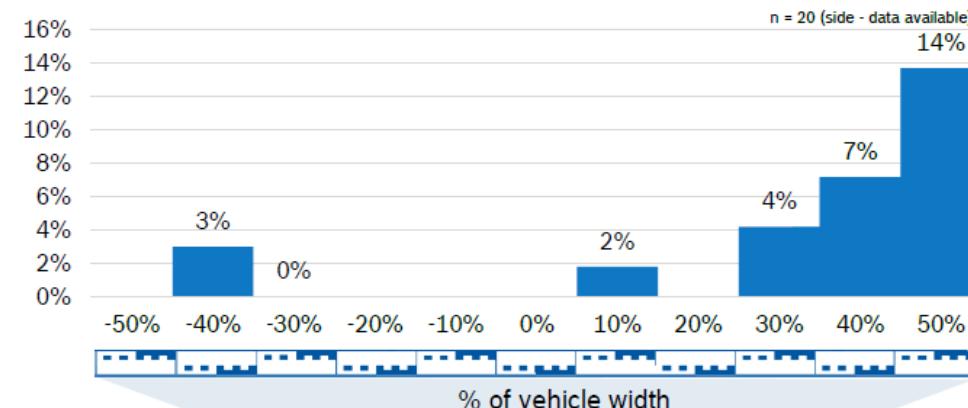
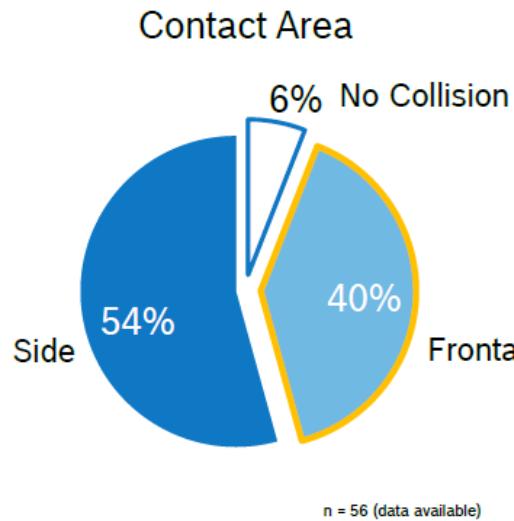
HCV (>12t) right turning conflict with VRU involvement Speed



- ▶ In 85% of relevant accidents collision speed is 20 km/h or below
- ▶ In 1 of 4 critical situation the relevant truck starts from a standstill

HCV (>12t) right turning conflict with VRU involvement

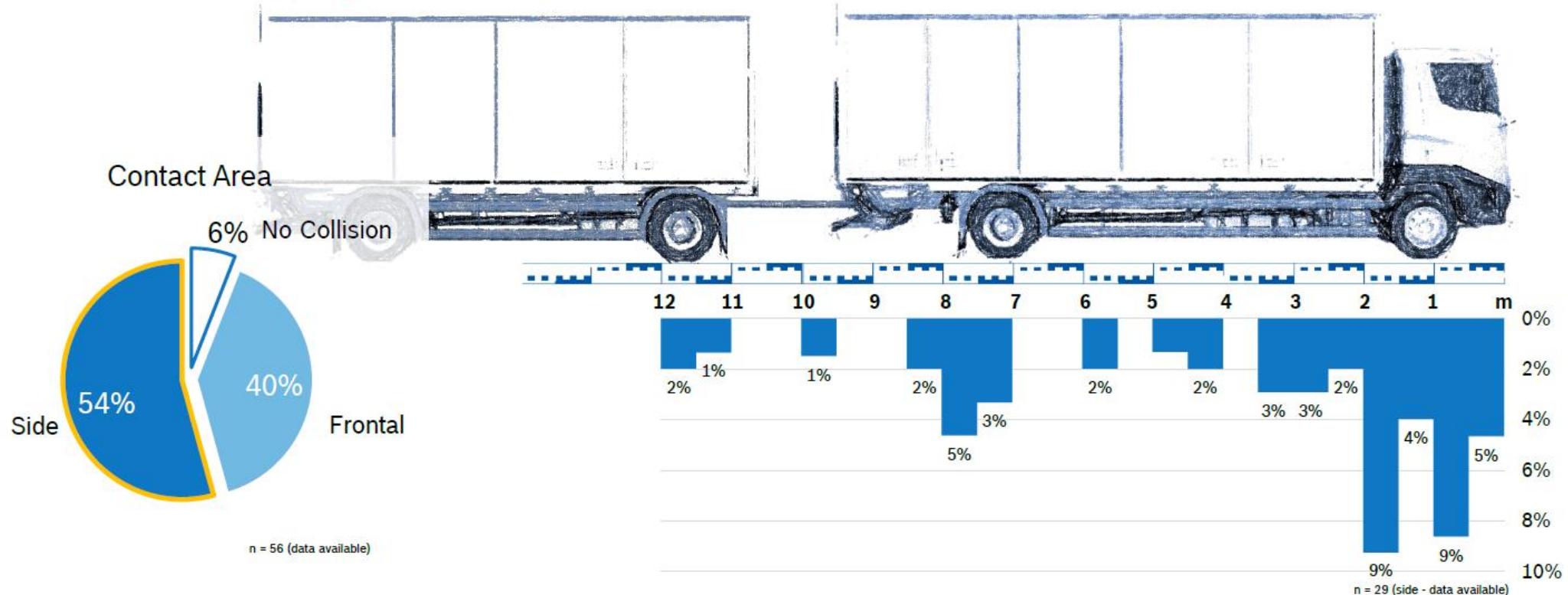
Contact area (y-position) frontal collision



- ▶ Frontal collision in 40% of relevant accidents with VRU
- ▶ Main contact area in frontal collisions: Right side corner area (30%-50% of vehicle width)

HCV (>12t) right turning conflict with VRU involvement

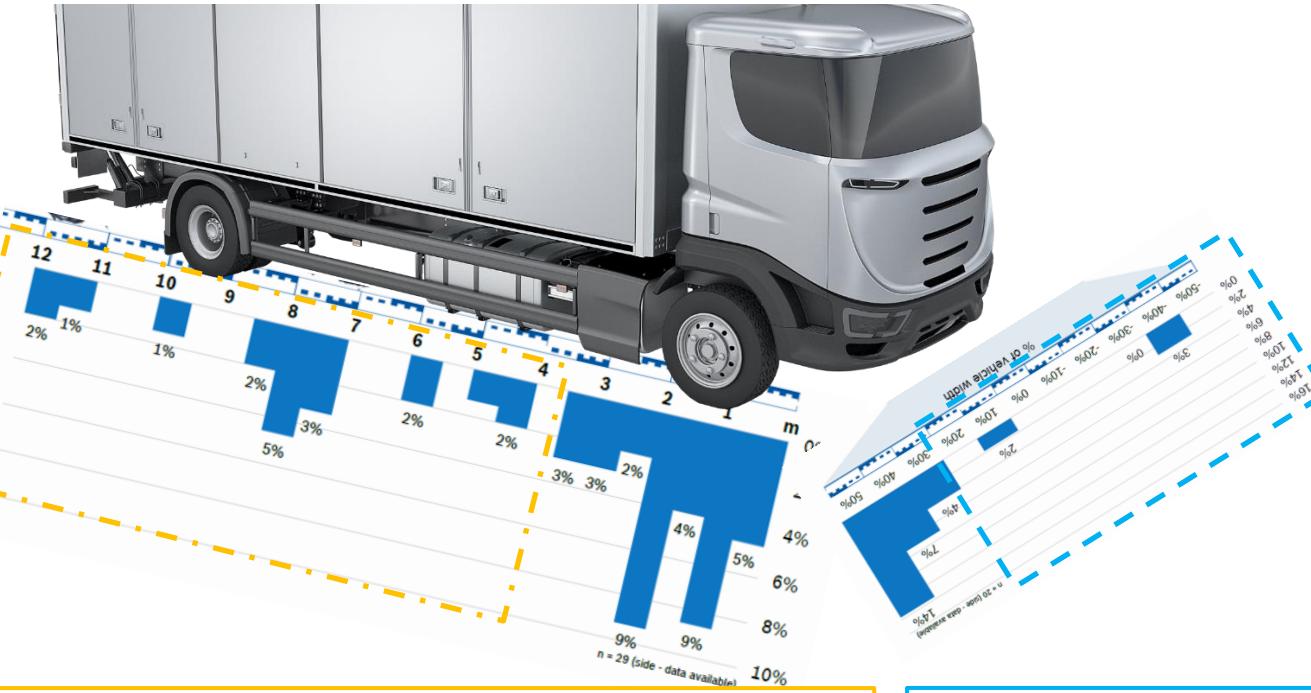
Contact area (x-position) side collision



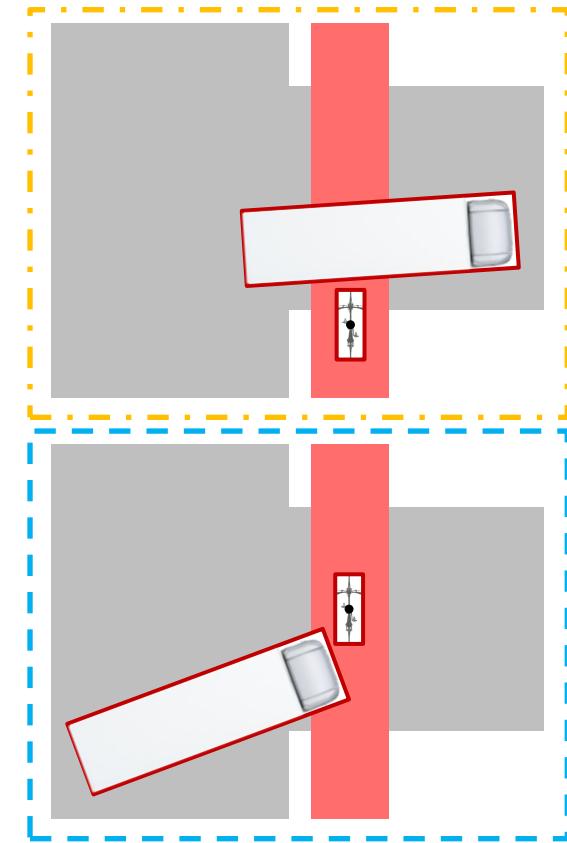
- ▶ Every second collision with VRU occur with a side contact
- ▶ Main contact area in side collisions: $x < 3 \text{ m}$

System Architectures

Probability collision interpretation



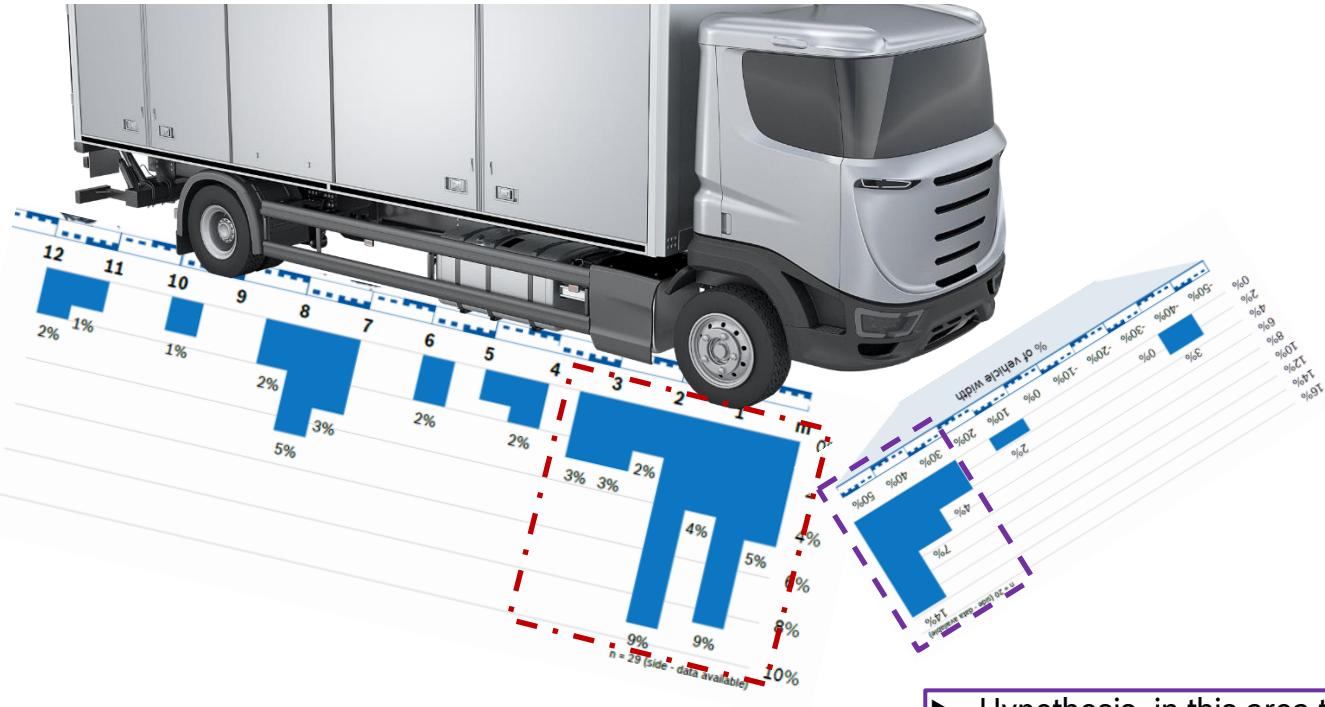
- ▶ Hypothesis: in this area there are less accidents because the truck was for a longer time in the field of view of the cyclist
- ▶ Most likely the accident is avoided by the cyclist through braking
- ▶ Braking the truck might reduce accident severity but not accident probability



- ▶ Hypothesis: in this area there are less accidents because the bicycle was for a longer time in the field of view of the truck driver
- ▶ Most likely the accident is avoided by the truck driver through braking
- ▶ Braking the bicycle might not have any positive impact

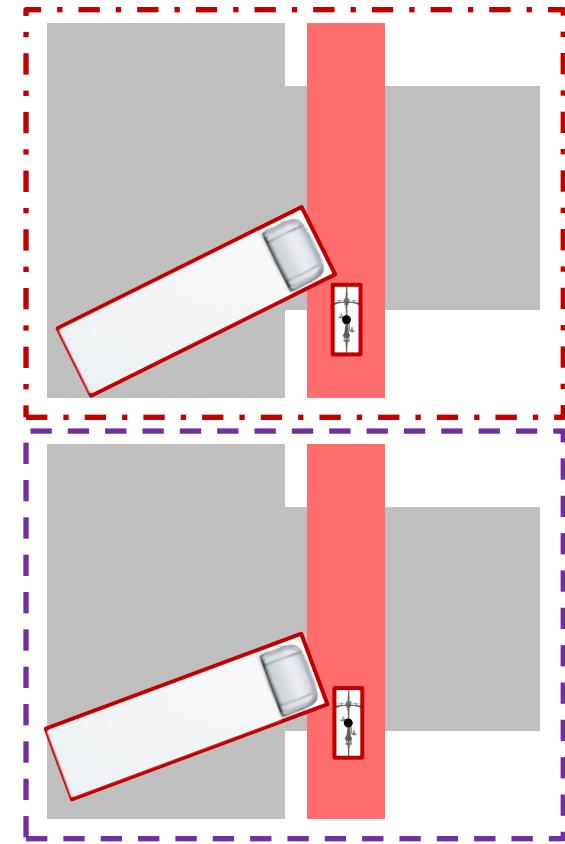
System Architectures

Probability collision interpretation



- ▶ Hypothesis: in this area there are more accidents because the truck was for a too short time in the field of view of the cyclist
- ▶ Braking both vehicles might reduce both accident severity and probability

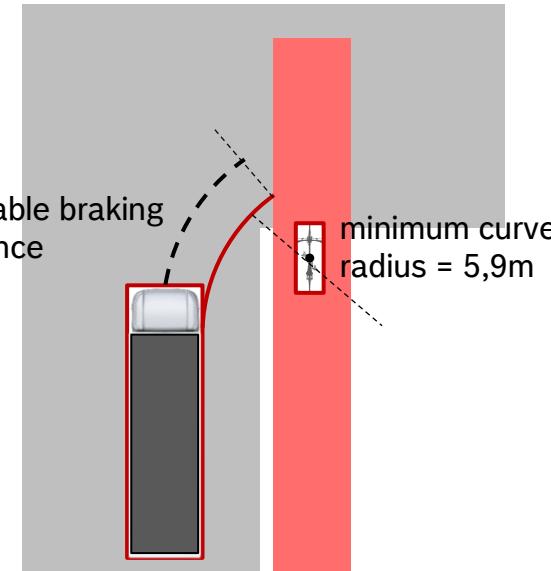
- ▶ Hypothesis: in this area there are more accidents because the bicycle was for a too short time in the field of view of the truck
- ▶ Additionally, through the inclination of the truck cabin, this corner is the first to intersect the rider's trajectory
- ▶ Braking both vehicles might reduce both accident severity and probability



System Architectures

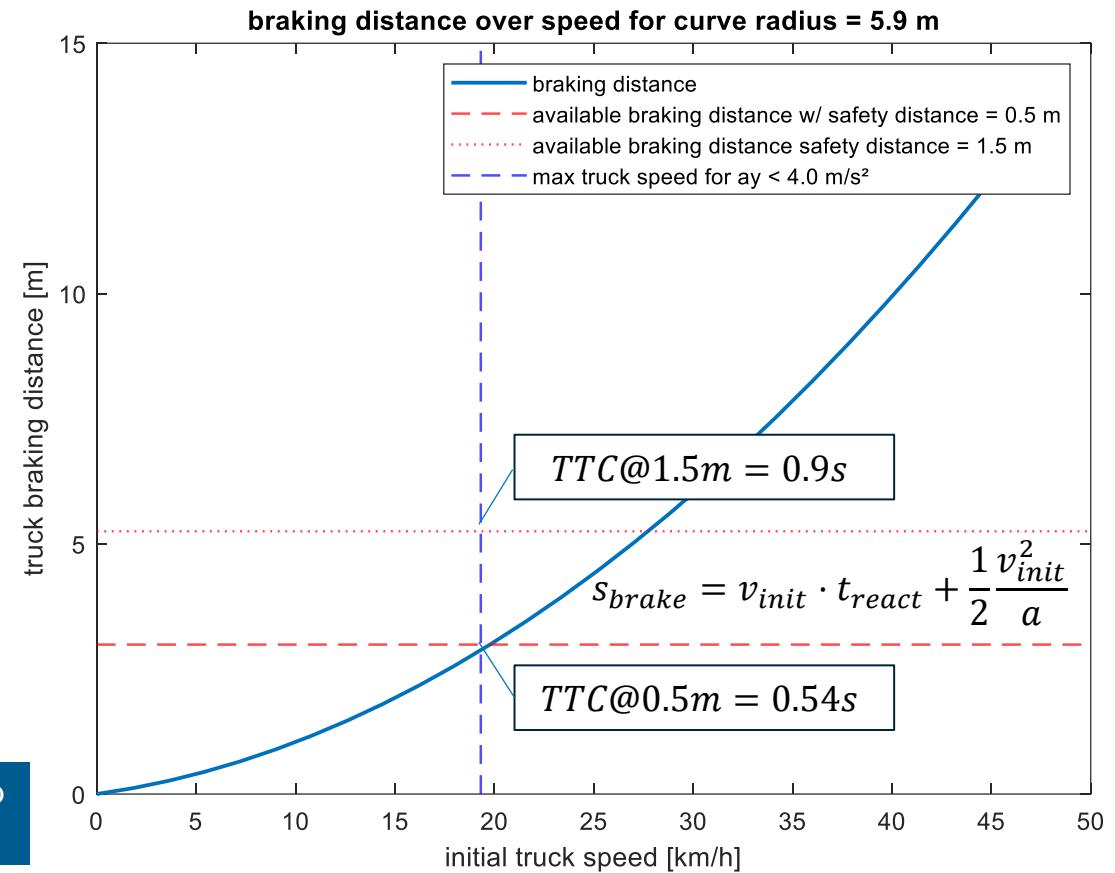
Truck Drive Off/ Turning

- ▶ Truck turning with minimum turning radius = 5,9m ([source](#))
- ▶ Maximum lateral acceleration 4m/s²



At the beginning of the turning maneuver, the truck can brake to prevent the accident in most cases (for $ay < 4 \text{m/s}^2$)

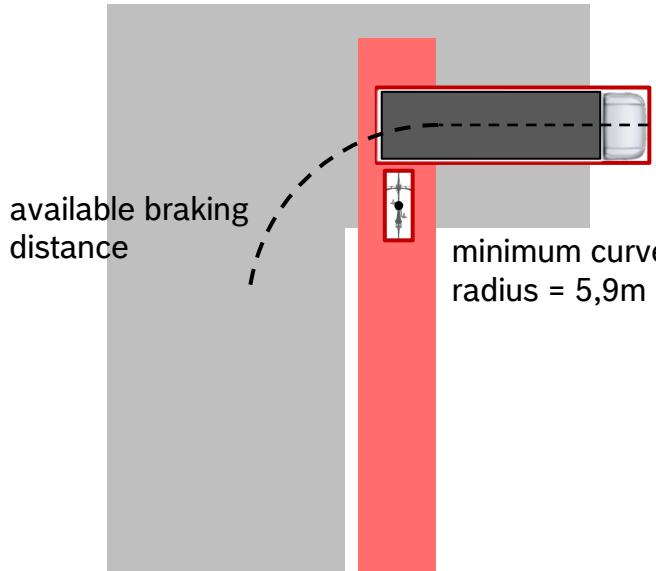
```
lt = 15; % truck length [m]
wt = 2.6; % truck width [m]
ax_truck = 8; %[m/s^2] maximum deceleration with AEB
ay_truck = 4; %[m/s^2] maximum lateral acceleration for curve
min_truck_radius = 5.9; %[m] minimum curve radius for the truck
t_react_AEB = 0.2; %[s] AEB TTL
```



System Architectures

Truck Drive Off/ Turning

- In order to avoid the collision at the back of the truck during turning the truck should actuate a **full braking** several seconds before the collision happens



- Truck length = 12m
- Average speed = 20km/h

$$t_{brake} = \frac{(truck\ length + distance\ to\ crossing)}{average\ speed} = 3s$$

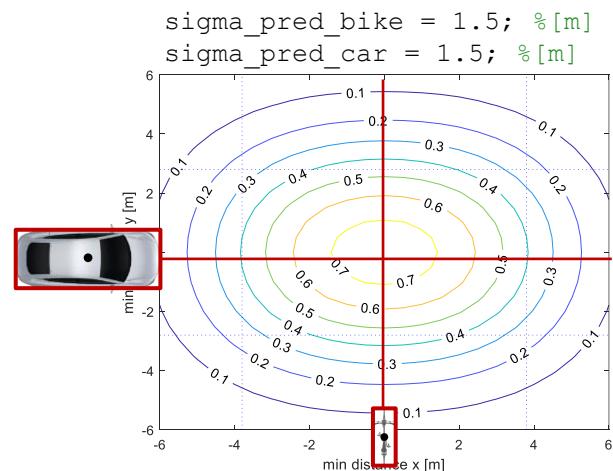
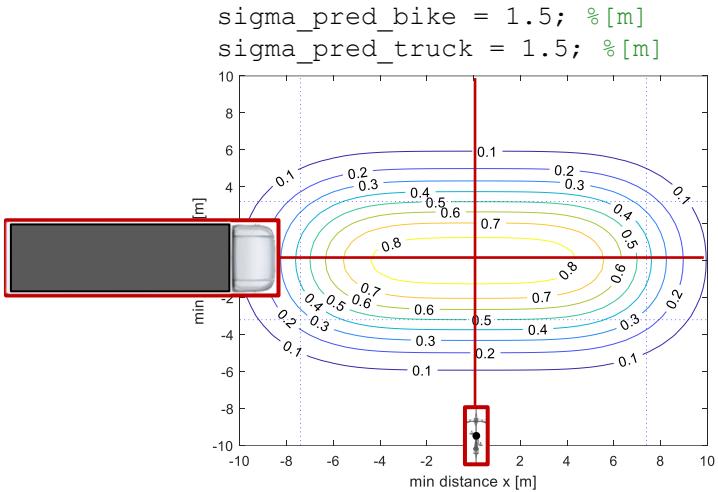
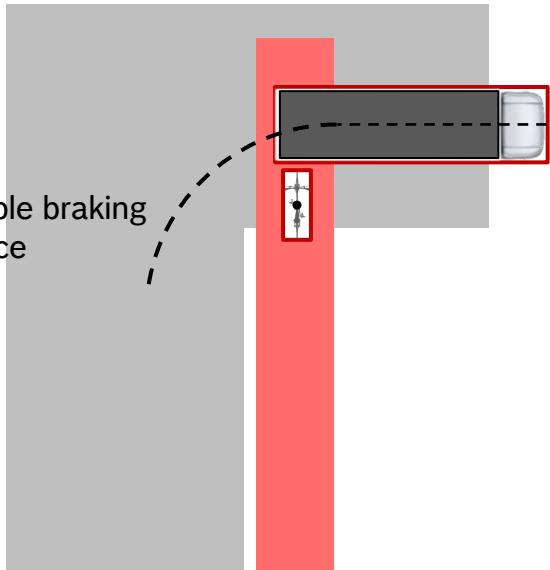


For accidents happening at the back of the truck, AEB should release a full braking at ~ 3s before the accident.
The accident should be avoided by warning the cyclist

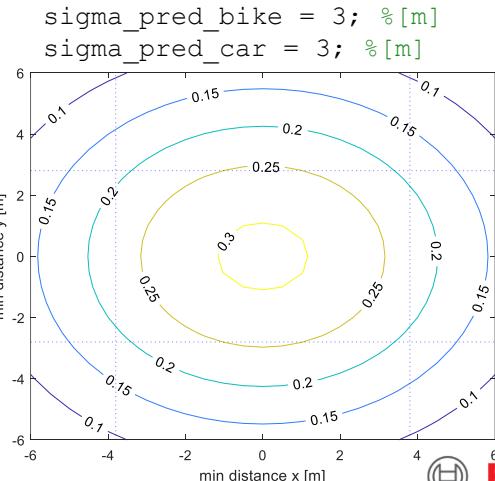
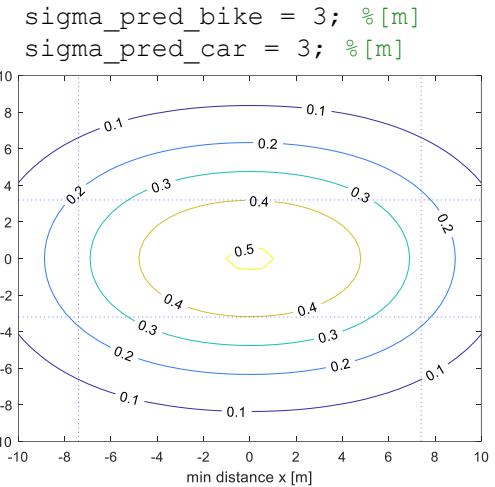
System Architectures

Prediction accuracy

- Due to the significantly bigger size, the prediction accuracy has a lower impact on the false positive ratio of the system compared to a crossroad situation with a car



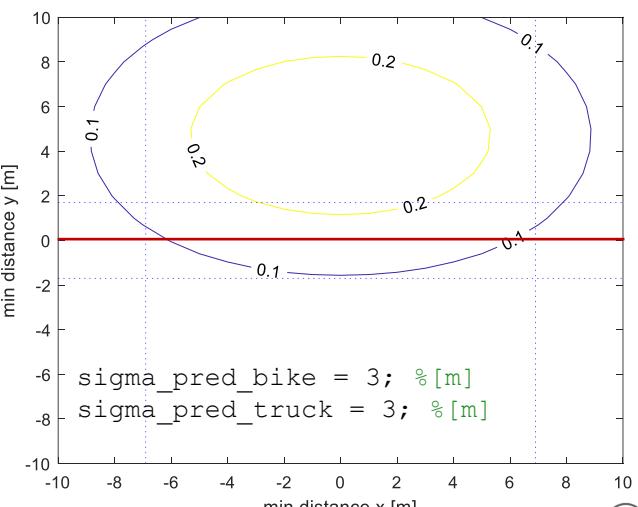
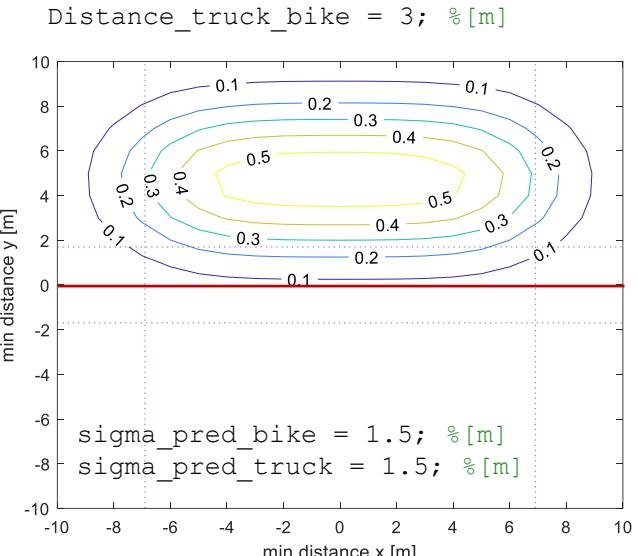
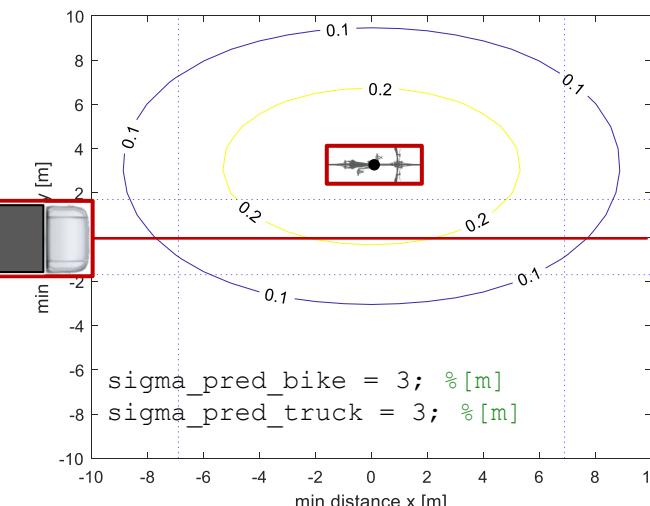
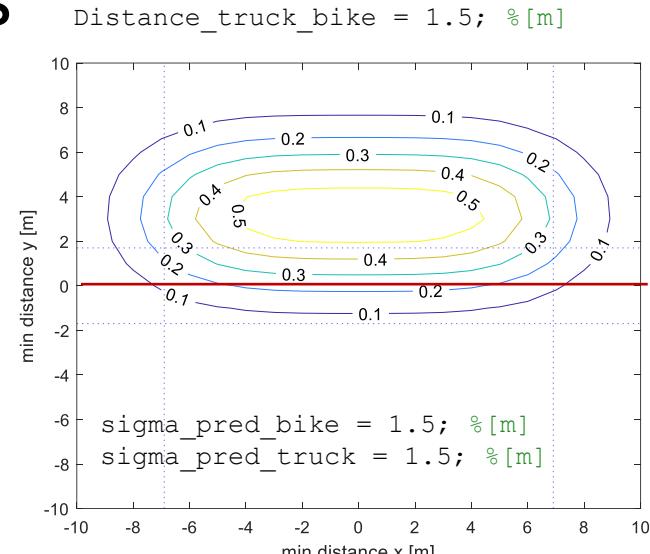
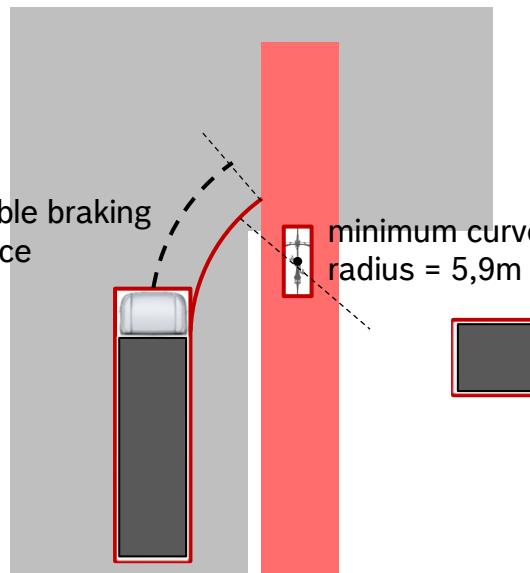
```
lt = 12; % truck length [m]
wt = 2.6; % truck width [m]
lb = 1.8; % bike length [m]
wb = 0.8; % bike width [m]
puffer_acc_detection = 1; % [m]
```



System Architectures

Prediction accuracy

- Even with low prediction confidence it is possible to identify on which side the bike is riding to detect a possible crossing of the trajectories



System Architectures

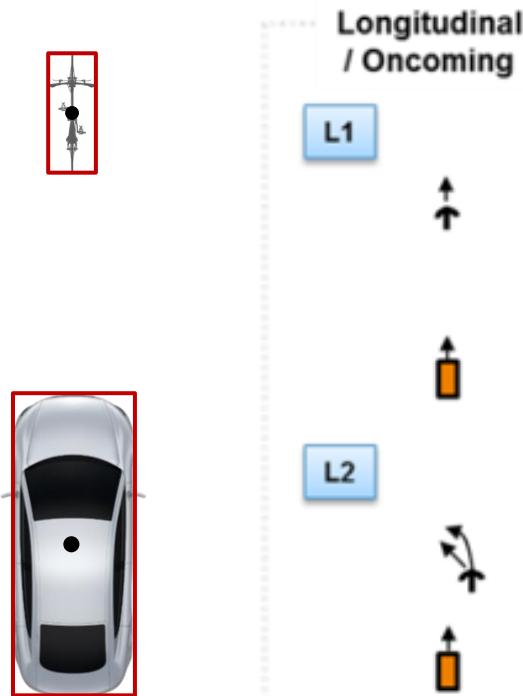
Trucks consideration based on accident analysis

- ▶ Accident mostly on the front part of vehicle
 - ▶ For the accidents on the rear part of the vehicle, the truck was for a certain time on the line of sight of the bicycle giving the rider more time to react
- ▶ 20% accidents with truck starting from stand still
 - ▶ Exploit trucks very low acceleration to use the throttle as driver intention indicator

UPFRONT RIDING CYCLIST

Upfront riding cyclist

Scenario description and strategies



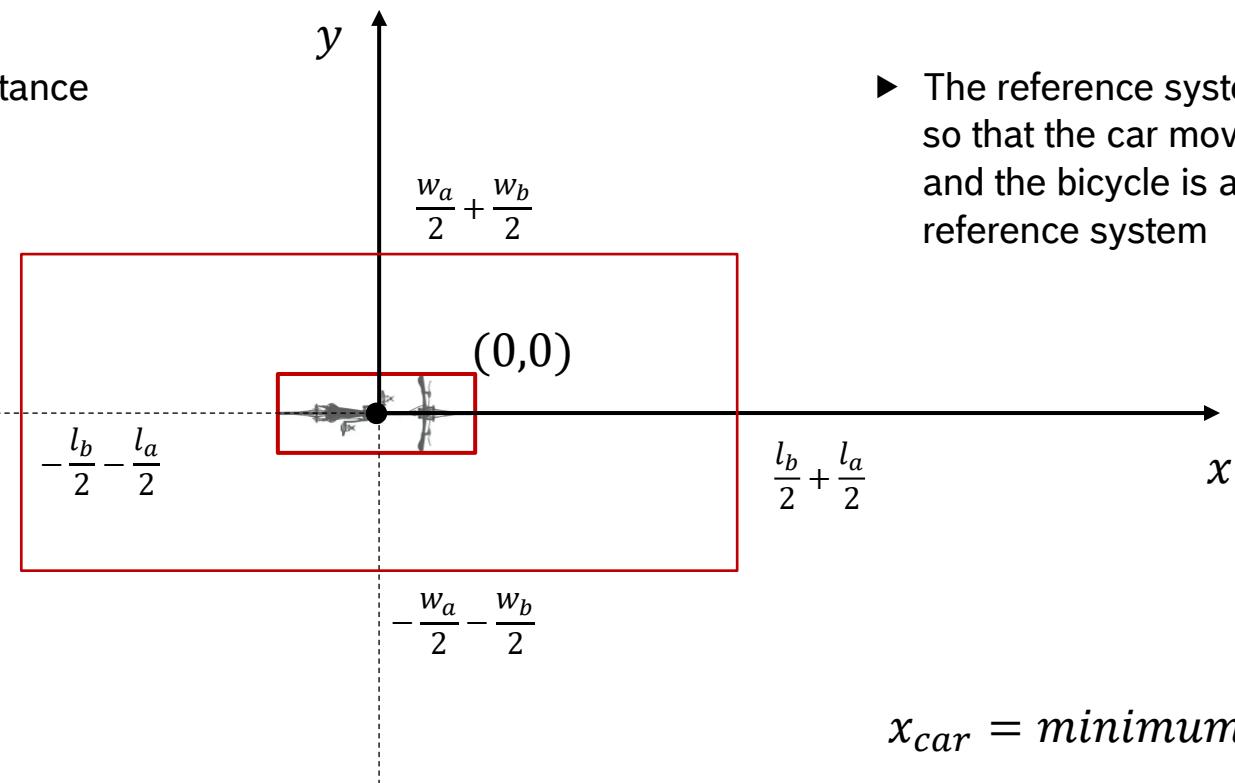
- ▶ This scenario covers the L1 and L2 scenarios from the TNO study
- ▶ In the L1 scenario the cyclist is riding in a straight line and is hit by the car
 - ▶ Benefit of rider warnings limited since the cyclist cannot avoid the accident
- ▶ In the L2 scenario the cyclist crosses the car trajectory causing the accident.
 - ▶ Benefit of rider warning as “blind spot assistant” to inform the rider to ride in a straight line and avoid sudden crossing of the car’s trajectory

Main challenge in the scenario it is to distinguish a normal overtaking from a collision. Even with low prediction accuracy it is possible to implement a blind spot assistant

System Architectures

Representation

- ▶ Representation of minimum distance



- ▶ The reference system can be moved so that the car moves along the x axis and the bicycle is at the origin of the reference system

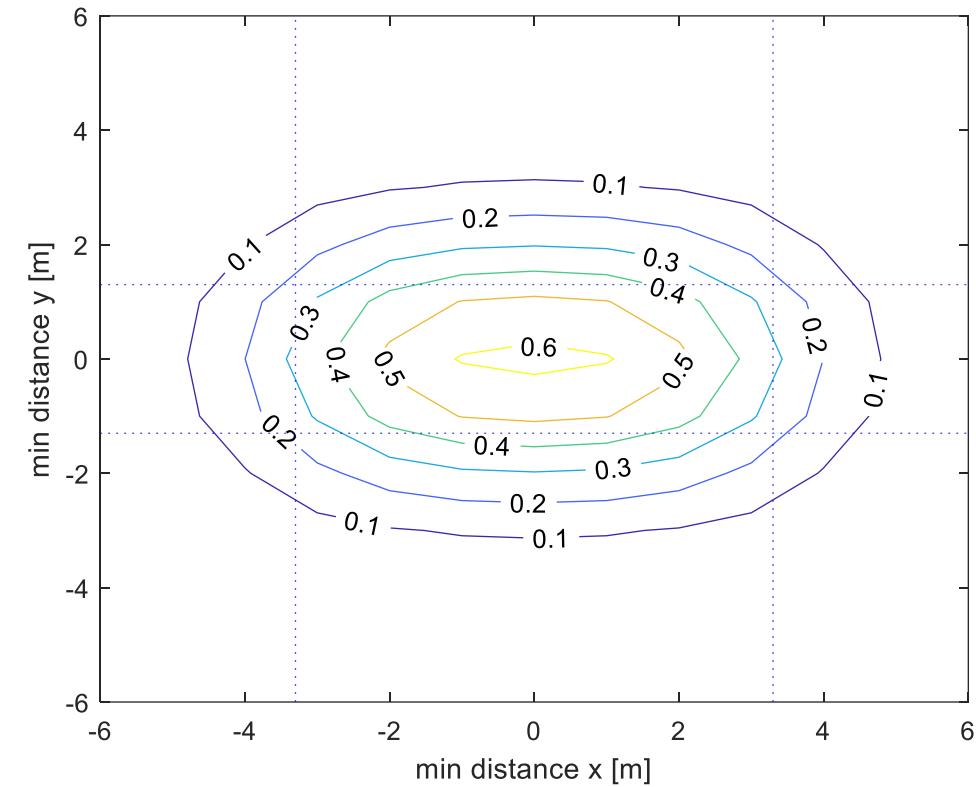
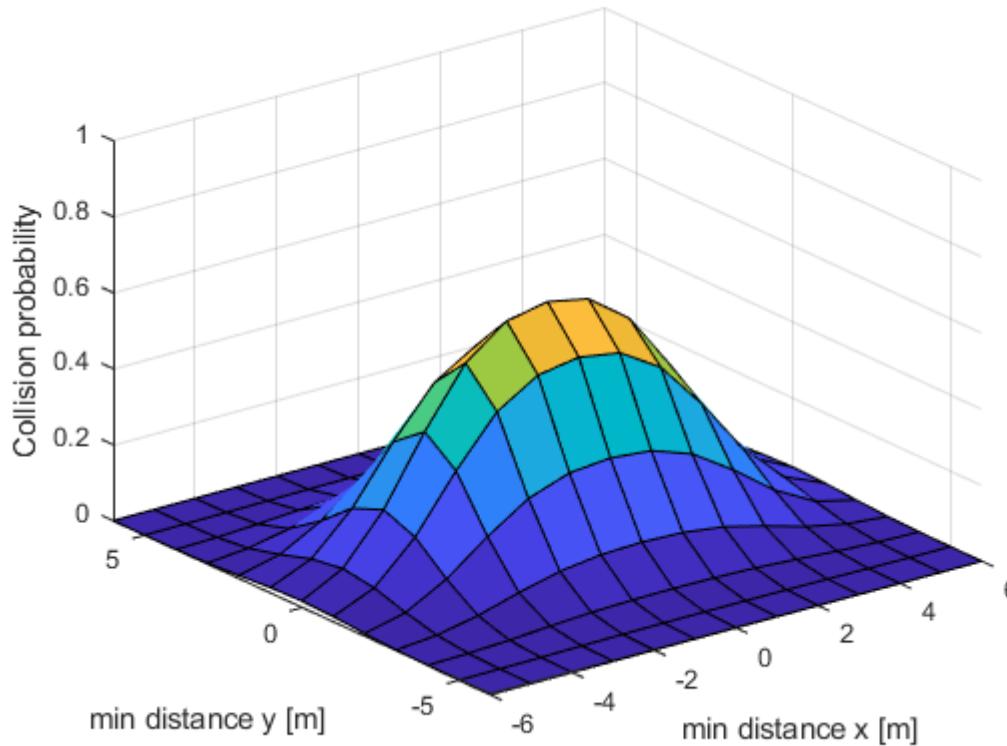
$x_{car} = \text{minimum distance } x$

$y_{car} = \text{minimum distance } y$

System Architectures

Probably vs min distance

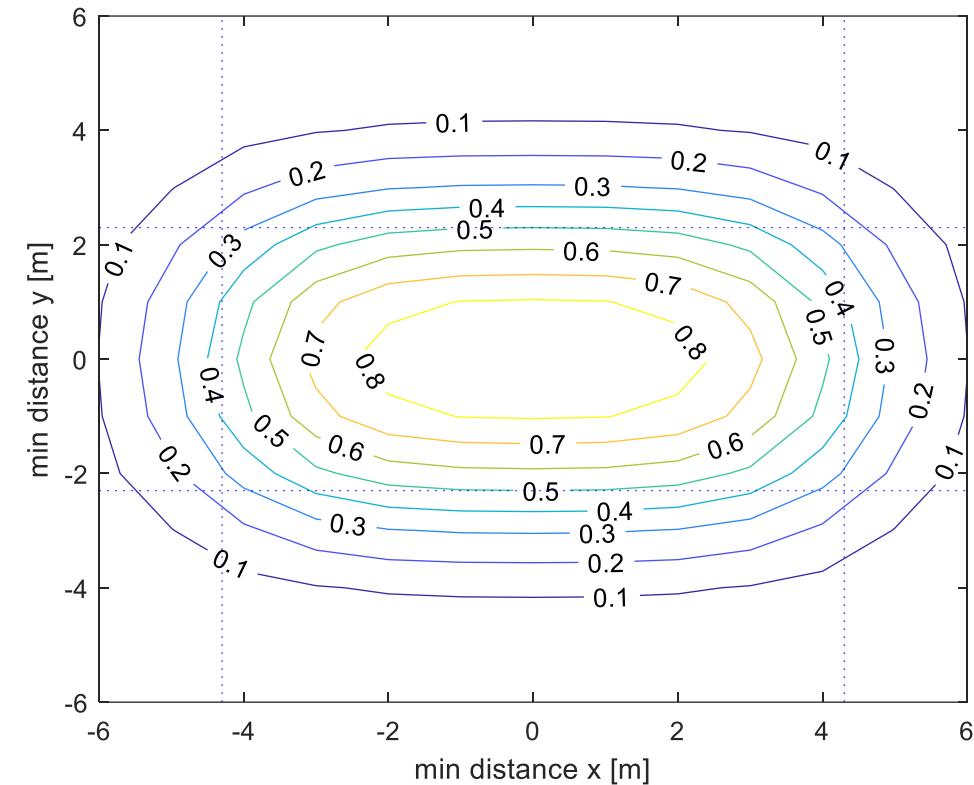
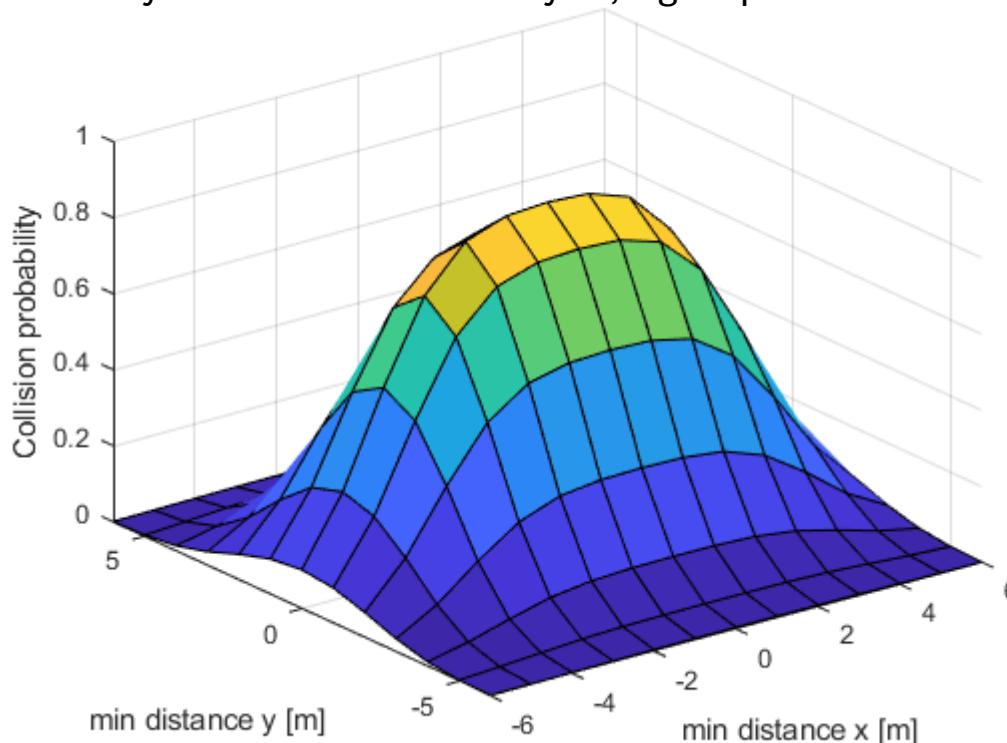
- It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
- No safety buffer around bicycle, sigma pred = 1



System Architectures

Probably vs min distance

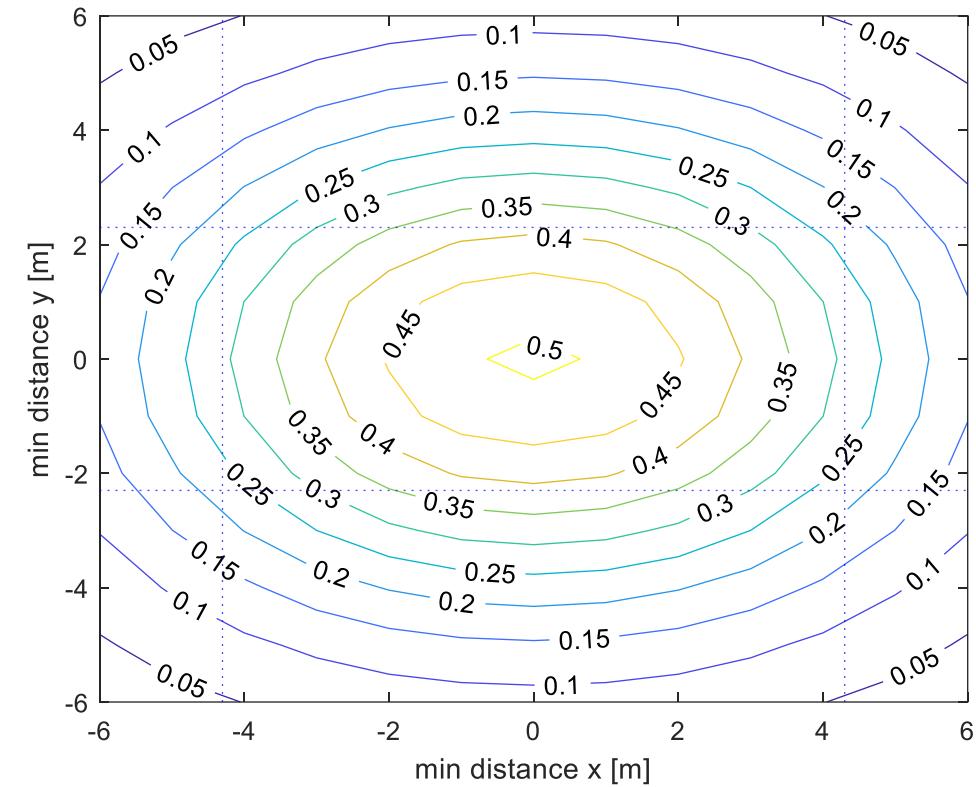
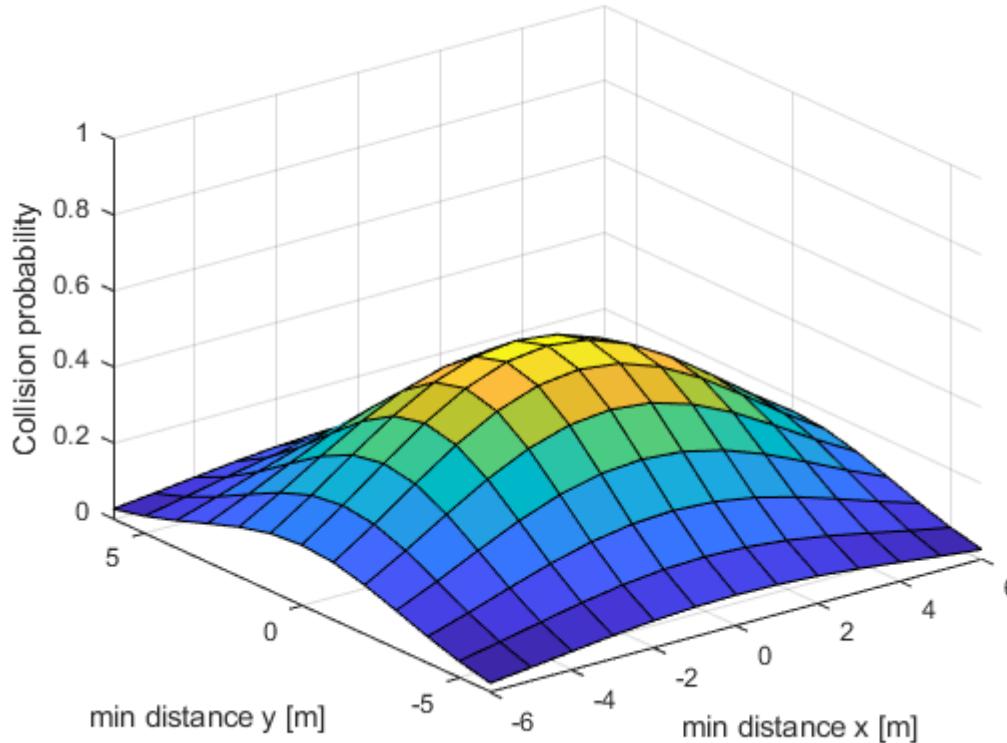
- It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
- Safety buffer = 1m around bicycle, sigma pred = 1



System Architectures

Probably vs min distance

- It is possible to calculate the collision probability as function of the predicted minimum distance in x and y direction
- Safety buffer = 1m around bicycle, sigma pred = 2



ONCOMING CYCLIST

Oncoming Cyclist

Oncoming cyclist

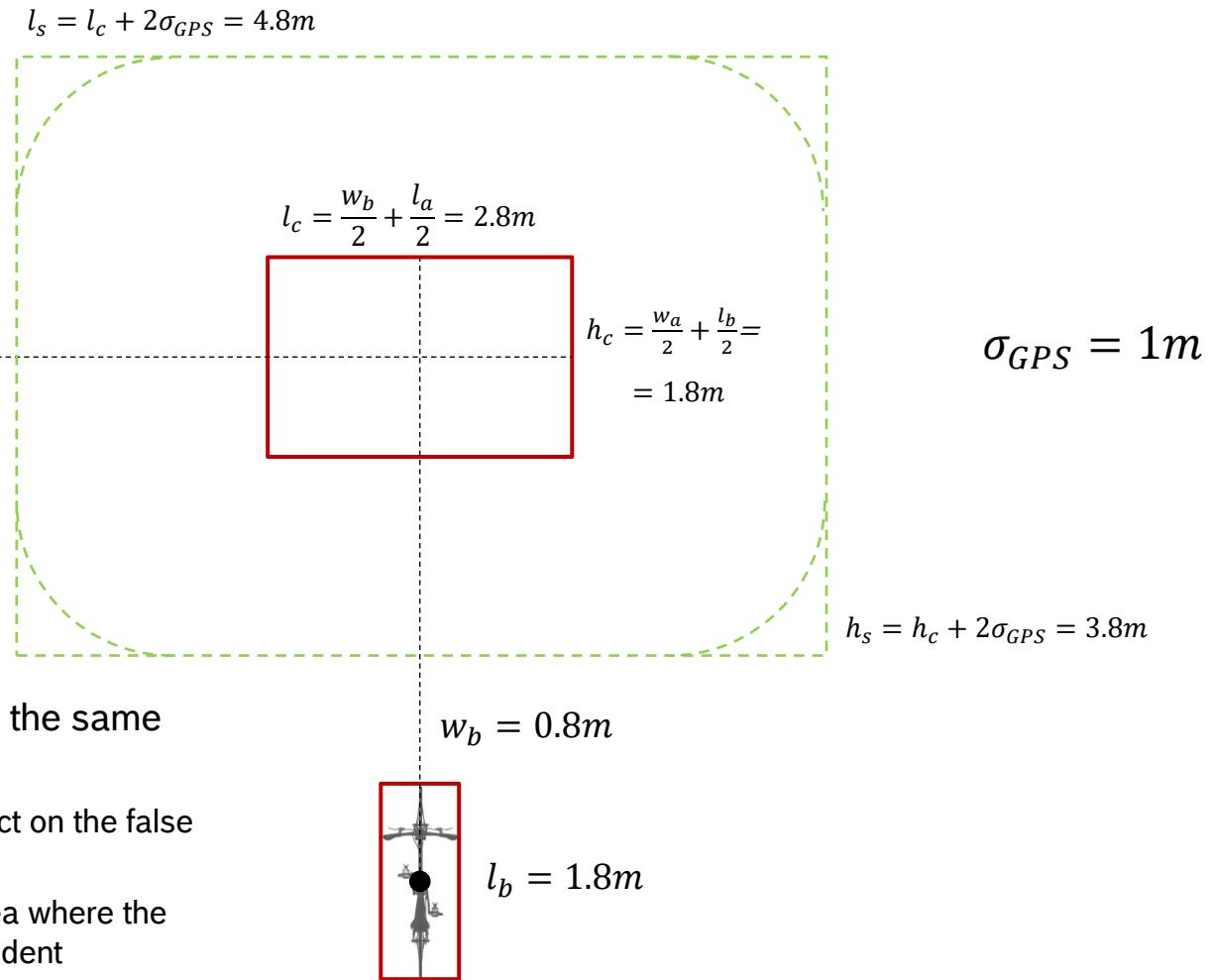
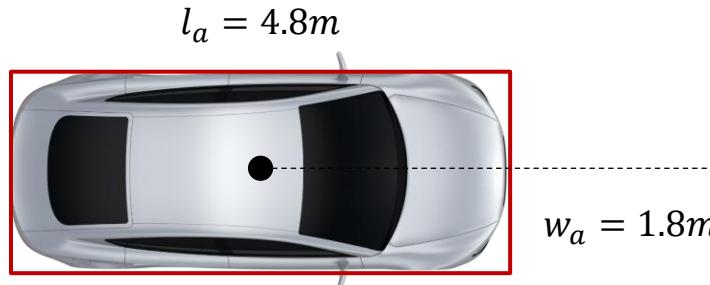
- ▶ Situation similar to the lateral riding cyclist case
- ▶ Missing data in order to quantify benefit

APPENDIX

Collision Probability Calculation

System Architectures

Cross road



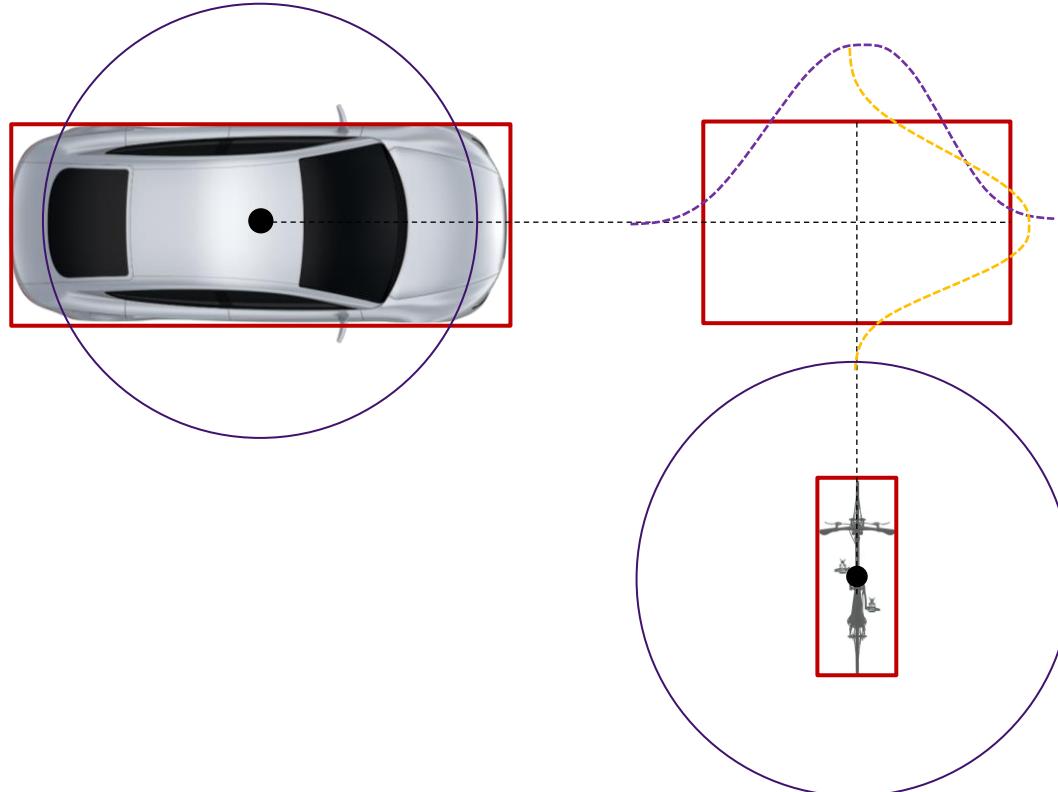
- ▶ If the center of the vehicles are in the impact area at the same time then an accident occurs
 - ▶ given the GPS accuracy it is possible to calculate its effect on the false positive and false negative
 - ▶ given the GPS accuracy it is possible to calculate the area where the vehicle centers should be to be sure not to have any accident
 - It is possible to identify the safe area for bicycle and car

System Architectures

First simplified calculation

► Situation:

- Car and bicycle predict to reach the centers reaches the center of the collision area at the same time after 1sec



► Goal:

- Calculate the real collision probability over time considering GPS and speed measurement inaccuracy

► Procedure

- Calculate the area of the gauss distribution of car and bicycle contained in the collision area over time
 - First step: only in the driving direction
- Multiply the two probabilities to calculate the probability that both vehicles are in the collision area at the same time

System Architectures

Results

```

la = 4.8;      % car length [m]
wa = 1.8;      % car width [m]

safety_buffer = 0;    % safe distance between car and bicycle [m]
lb = 1.8 + safety_buffer;    % bike length [m]
wb = 0.8 + safety_buffer;    % bike width [m]

lc = (la+wb)/2; % length of collision area [m]
wc = (lb+wa)/2; % width of collision area [m]

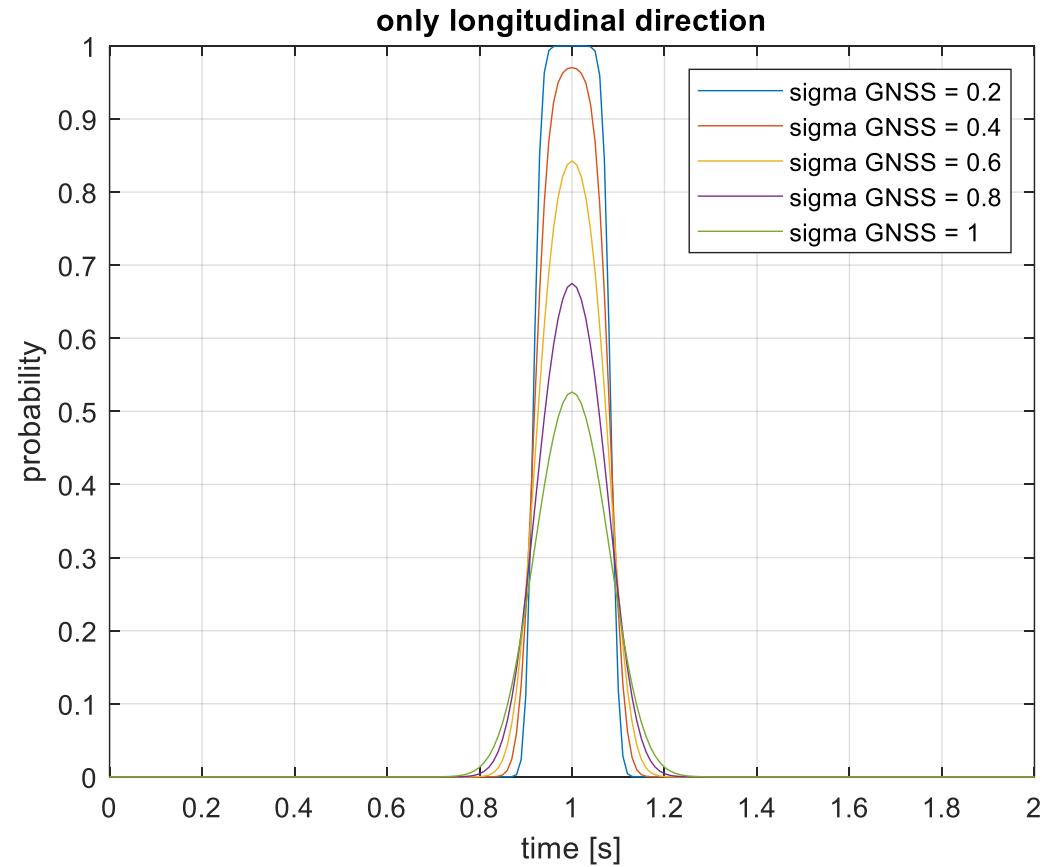
sigma_GNSS = 1; % GPS standard deviation [m]
sigma_v = 0.1; % speed standard deviation [m/s]
pred_t = 1;     % Prediction time [s]

v_car = 60/3.6;      % car speed [m/s]
v_bike = 15/3.6;    % bike speed [m/s]

x0 = v_car * pred_t; % car starting point [m]
y0 = v_bike * pred_t; % bicycle starting point [m]

P_car = normcdf(lc/2, 0, sigma_POS) - normcdf(-lc/2, 0, sigma_POS);
% Probability of car in collision area
P_bike = normcdf(wc/2, 0, sigma_POS) - normcdf(-wc/2, 0, sigma_POS);
% Probability of bike in collision area

```

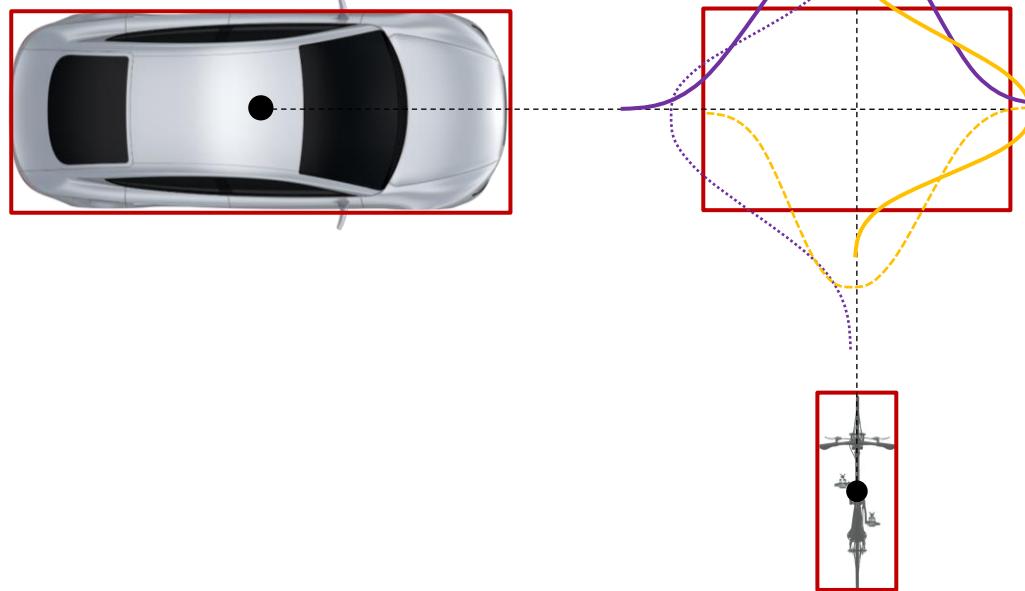


System Architectures

Accuracy on both axes for both vehicles

► Situation:

- Car and bicycle predict to reach the centers reaches the center of the collision area at the same time after 1sec



► Goal:

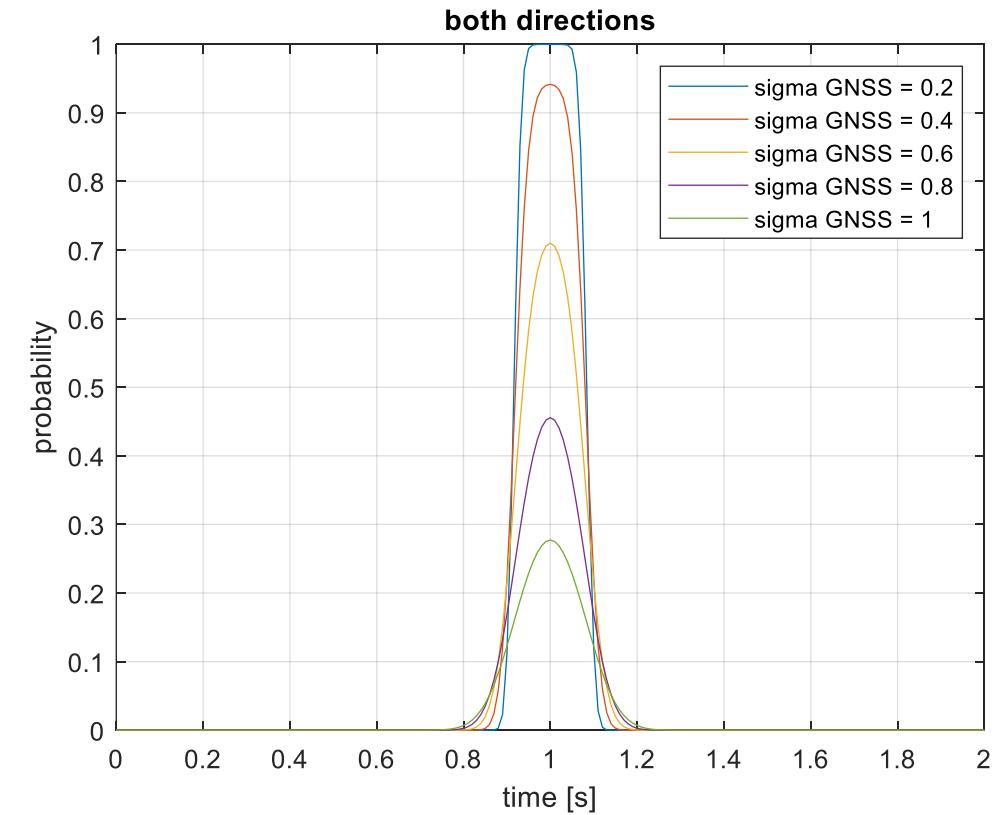
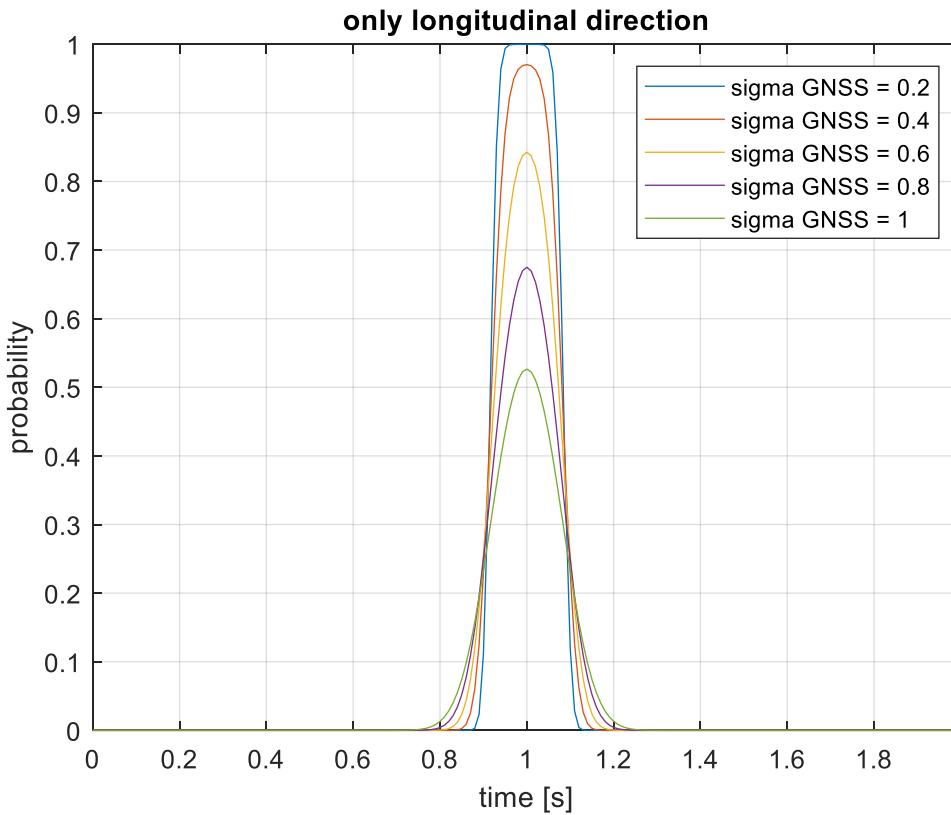
- Calculate the real collision probability over time considering GPS and speed measurement inaccuracy

► Procedure

- Calculate the area of the gauss distribution of car and bicycle contained in the collision area over time
 - **Second step: in both directions for both vehicles**
- Multiply the two probabilities to calculate the probability that both vehicles are in the collision area at the same time

System Architectures

Results

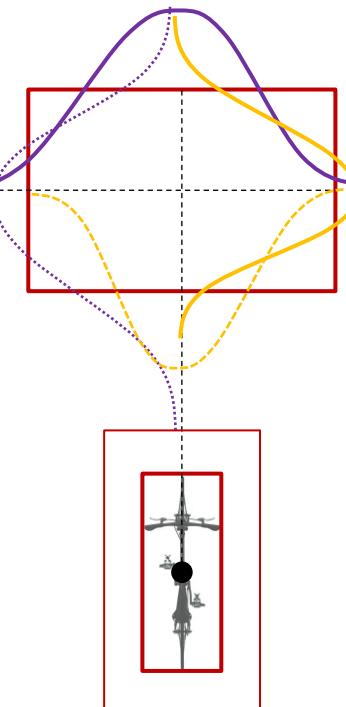
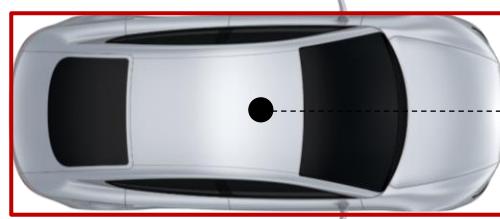


System Architectures

Accuracy on both axes for both vehicles plus safety buffer

► Situation:

- Car and bicycle predict to reach the centers reaches the center of the collision area at the same time after 1sec



$$w_{bSB} = w_b + 2 \cdot SB$$

$$l_{bSB} = l_b + 2 \cdot SB$$

► Goal:

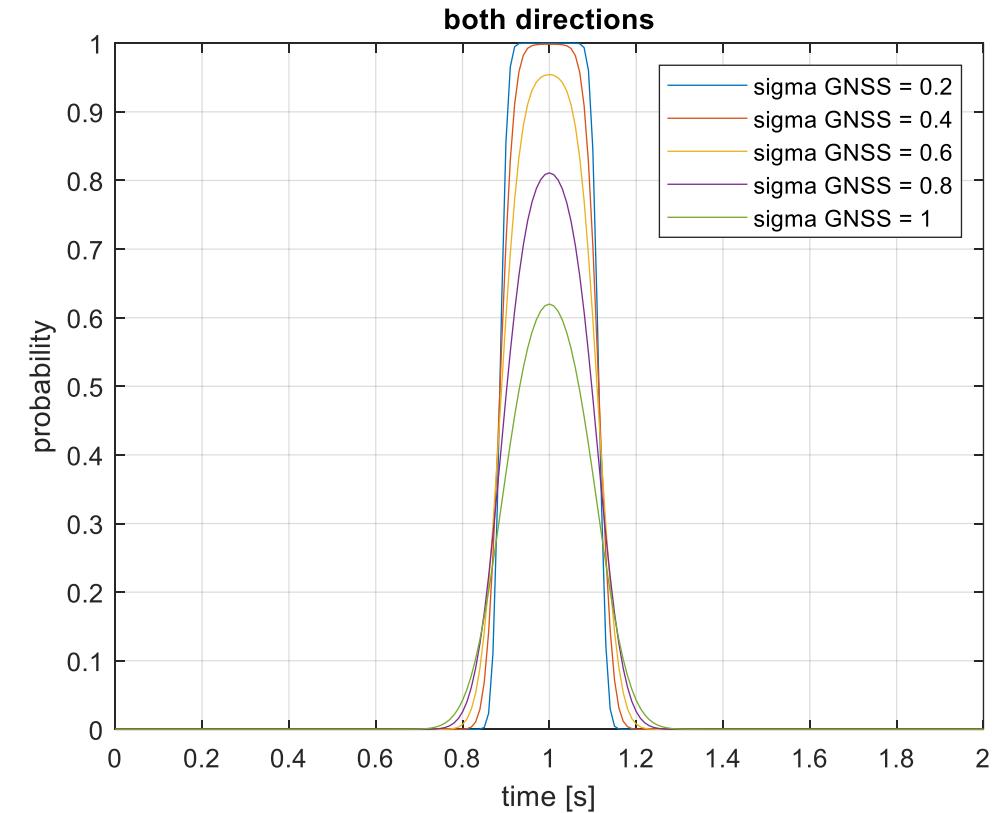
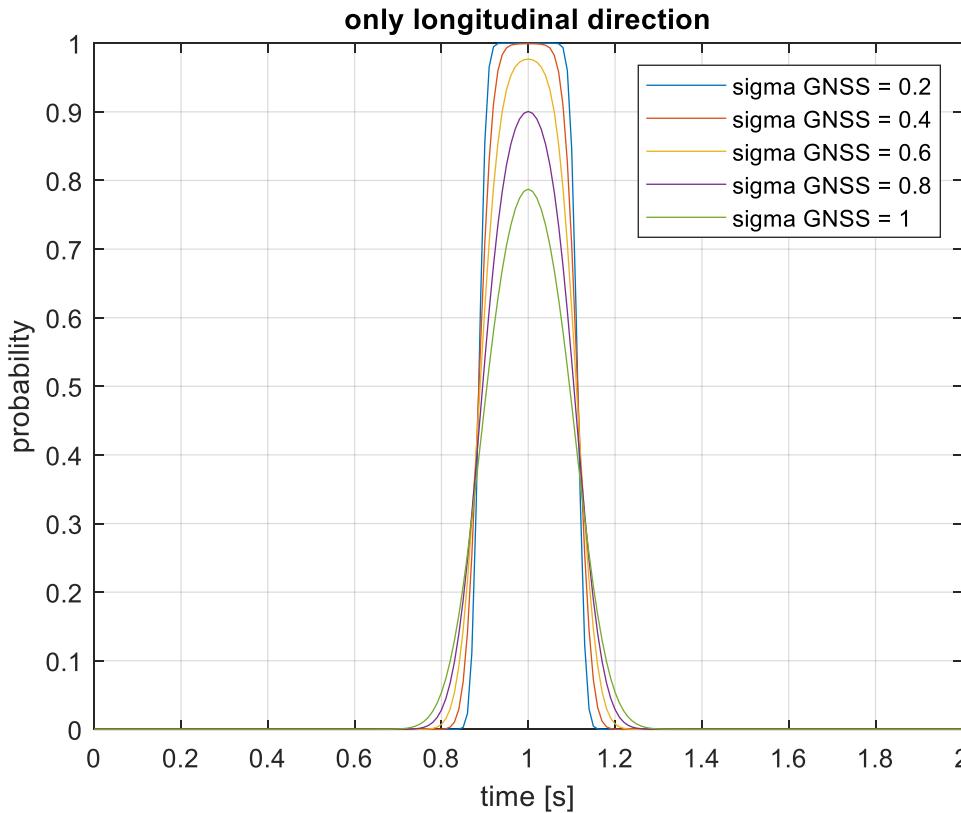
- Calculate the real collision probability over time considering GPS and speed measurement inaccuracy

► Procedure

- Calculate the area of the gauss distribution of car and bicycle contained in the collision area over time
 - Second step: in both directions for both vehicles
 - **Increase bicycle dimensions to consider a safety buffer**
- Multiply the two probabilities to calculate the probability that both vehicles are in the collision area at the same time

System Architectures

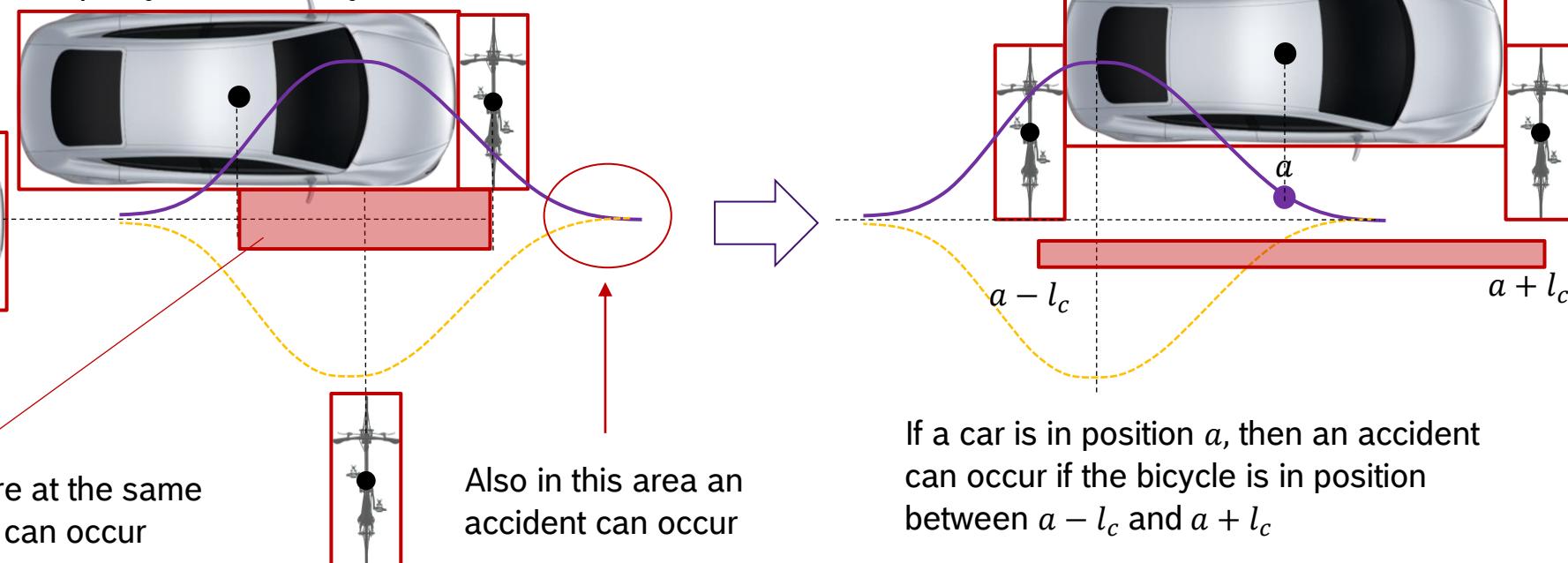
Results safety buffer = 1m



System Architectures

Correct calculation for non fixed collision area

- In the previous calculation only a fixed in space collision area has been considered: this leads to an underestimation of the collision probability → for simplicity consider only x axes



- if both vehicles are here at the same time, then an accident can occur

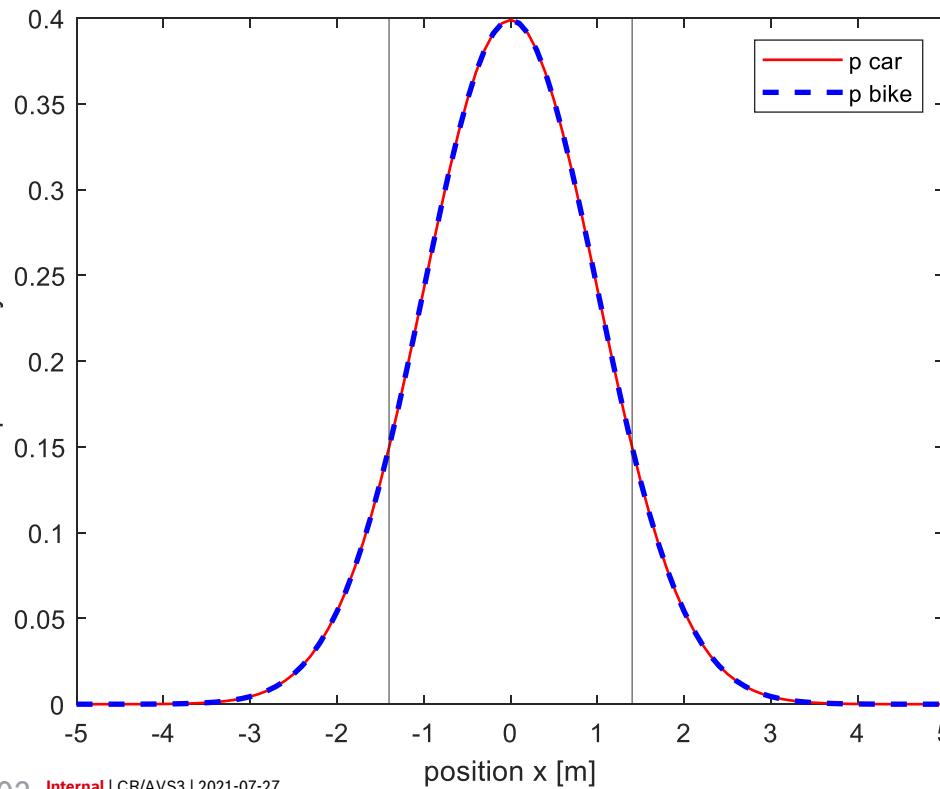
- In order to calculate the collision probability it is necessary to integrate for every a the probability the of the car in a multiplied by the probability of the bike between $a - l_c$ and $a + l_c$

If a car is in position a , then an accident can occur if the bicycle is in position between $a - l_c$ and $a + l_c$

System Architectures

Comparison of collision probability

- First evaluation of the effect on the probability just considering the x axis and prediction of vehicles' centers on the same position

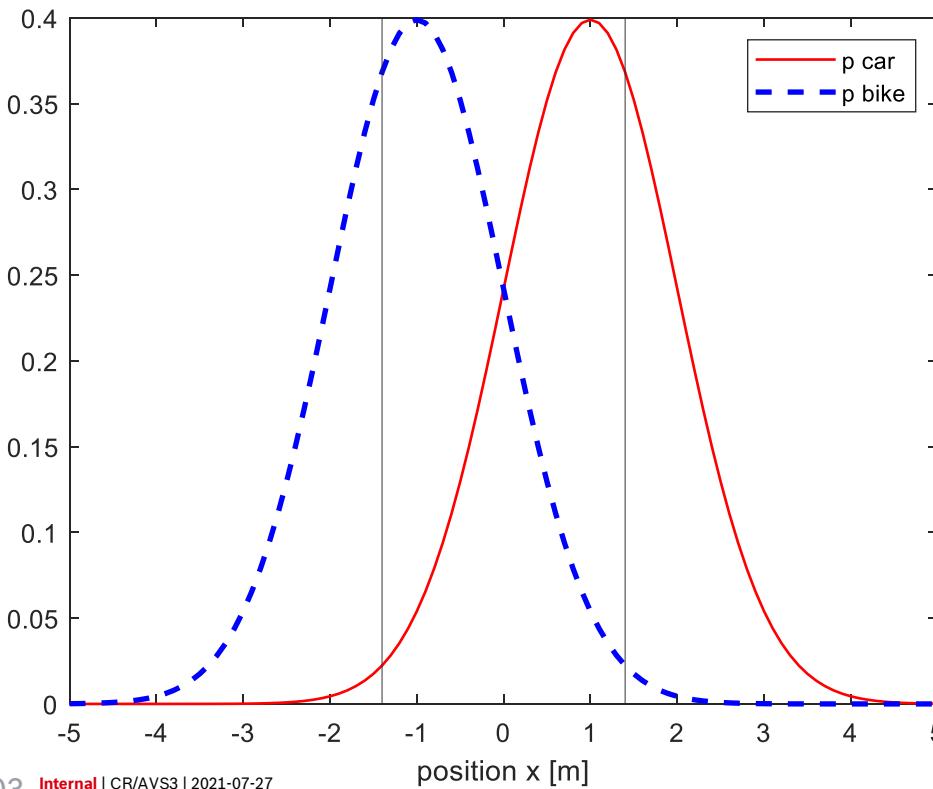


- 1-D collision probability for fixed collision area around the center: **0.70**
- 1-D collision probability considering all points with distance smaller than collision area width: **0.95**

System Architectures

Comparison of collision probability

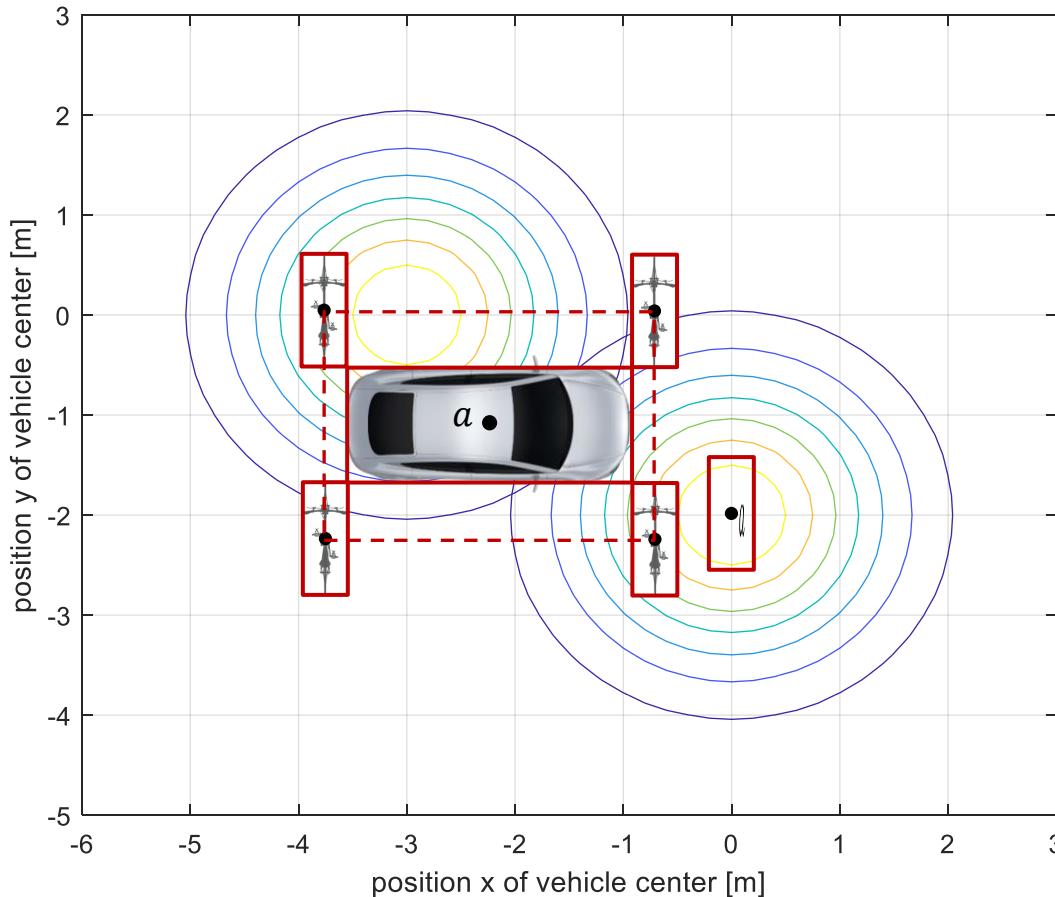
- First evaluation of the effect on the probability just considering the x axis and 2m distance between vehicles' centers



- 1-D collision probability for fixed collision area around the center: **0.41**
- 1-D collision probability considering all points with distance smaller than collision area width: **0.71**

System Architectures

Comparison of collision probability

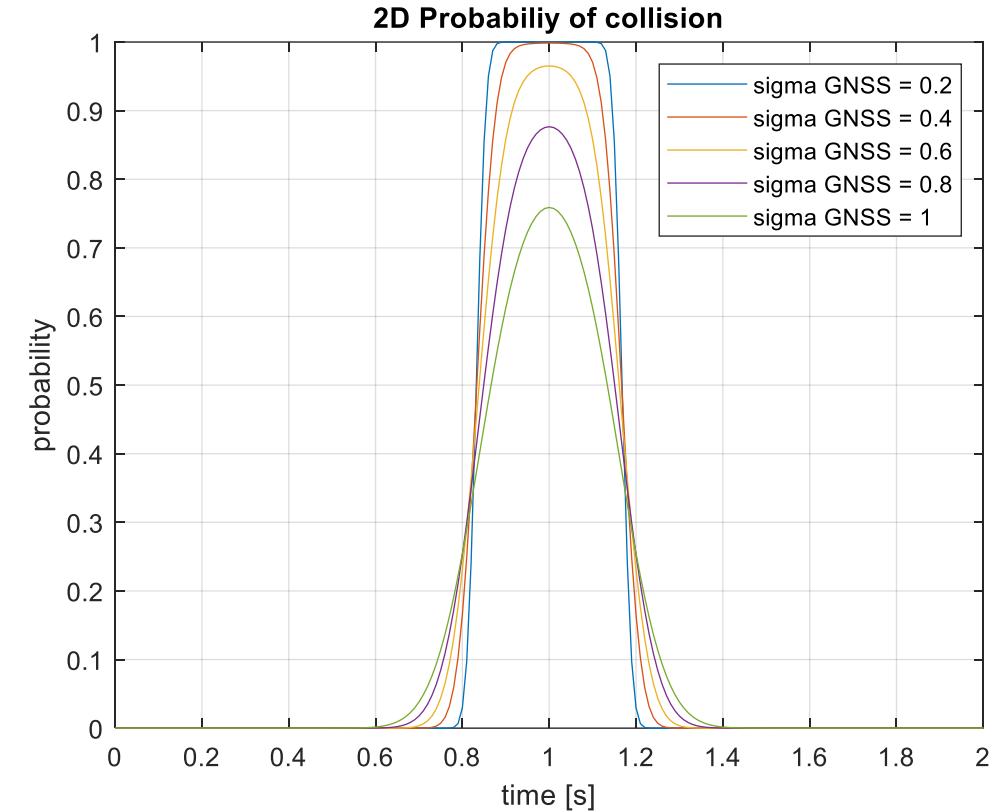
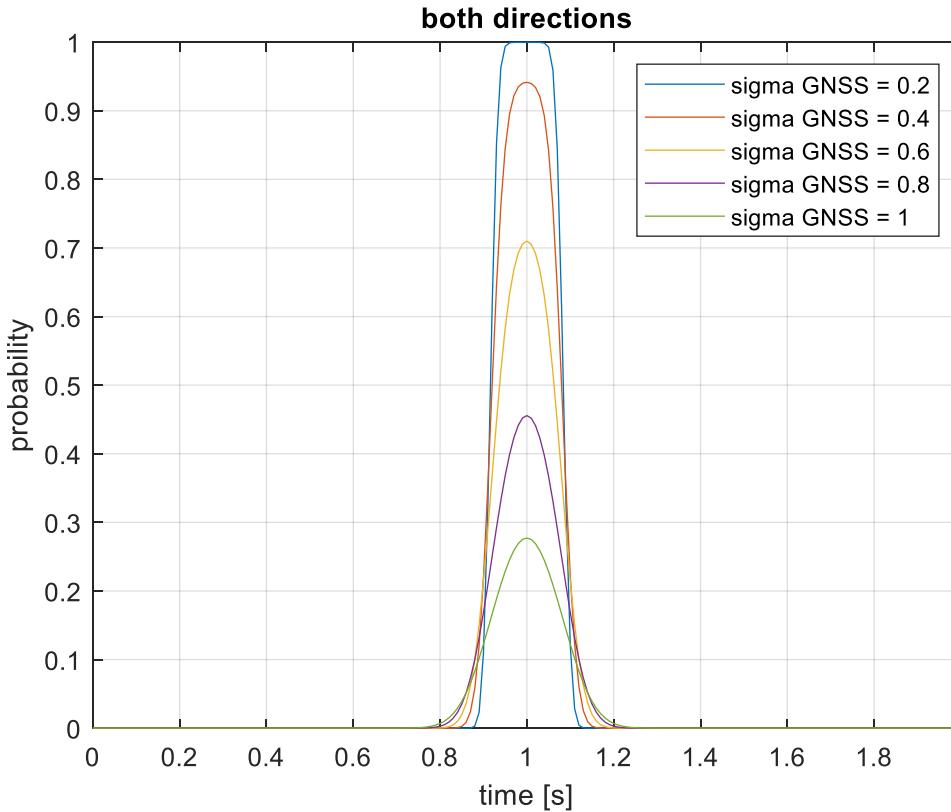


- ▶ In a similar way, for the 2D scenario we need to integrate:
 - ▶ probability that the center of the car is in one point a
 - ▶ multiplied by the probability that the center of the bicycle is between
 - $a_x - l_c < b_x < a_x + l_c$
 - $a_y - w_c < b_y < a_y + w_c$
 - ▶ Considering the covariance matrix of the probability distribution:
- $$\begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$$
- ▶ It is possible to calculate the probability that the distance along the x axes is smaller than l_c and multiply it by the probability that the distance along the y axis is smaller than w_c

System Architectures

Comparison fixed box vs integral of all positions

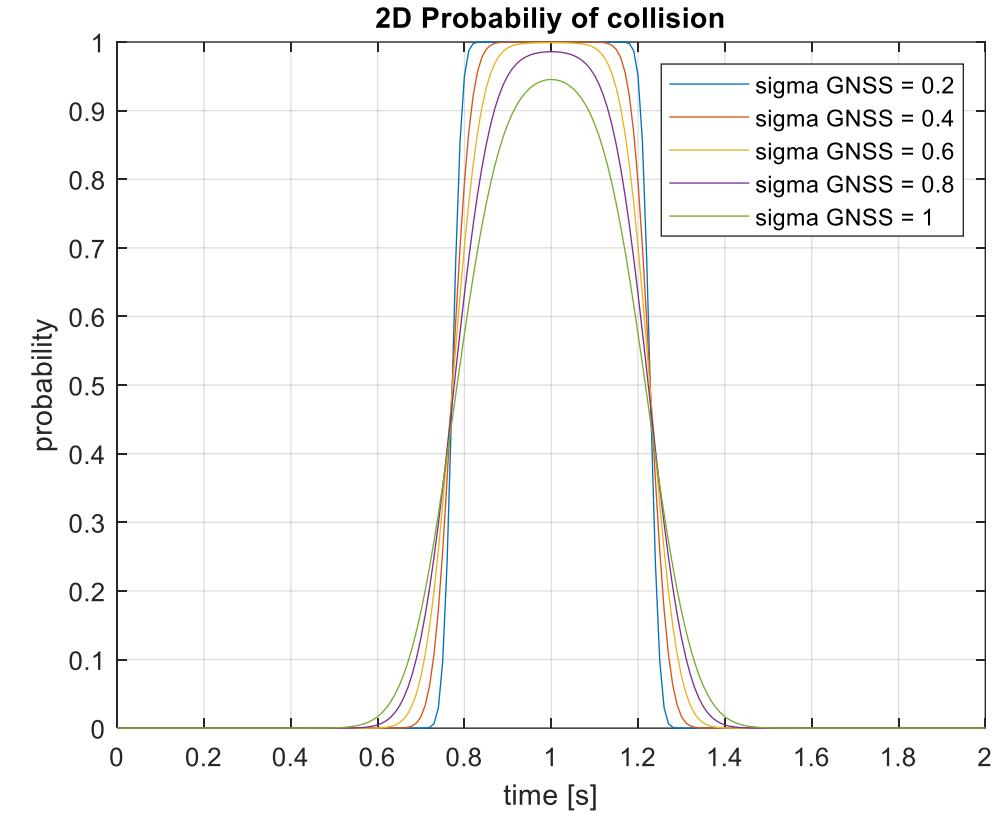
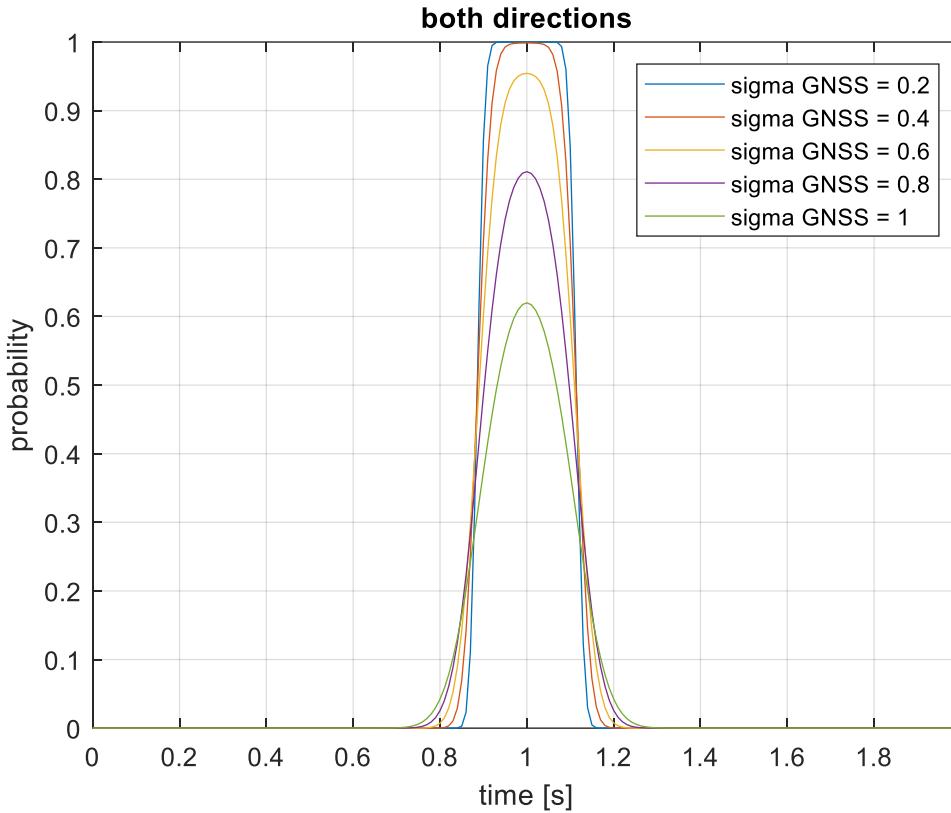
- Car and bicycle centers colliding (best case for prediction)
- Only considering sigma GNSS as uncertainty

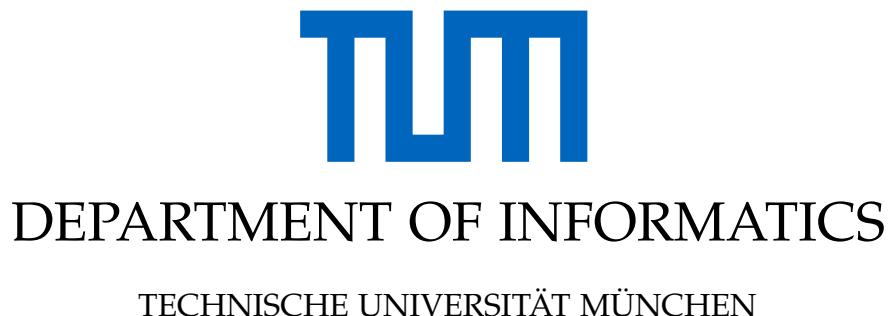


System Architectures

Fixed box vs integral of all positions, safety buffer = 1m

- Car and bicycle centers colliding (best case for prediction)
- Only considering sigma GNSS as uncertainty



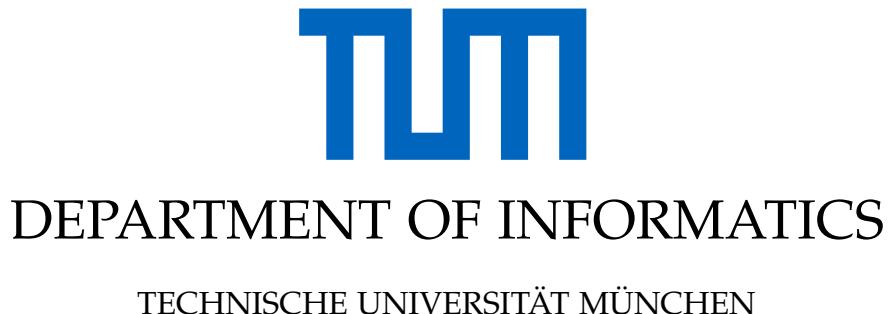


Master's Thesis in Robotics, Cognition, Intelligence

Optimization of a Collision Warning System for Electric Bicycles based on Vehicle-to-Vehicle Communication

Benedikt Kansy





Master's Thesis in Robotics, Cognition, Intelligence

Optimization of a Collision Warning System for Electric Bicycles based on Vehicle-to-Vehicle Communication

Optimierung eines Kollisionswarnsystems für Elektrofahrräder basierend auf Fahrzeug-Fahrzeug-Kommunikation

Author: Benedikt Kansy
Supervisor: Prof. Dr.-Ing. habil. Alois Christian Knoll
Advisor: M.Sc. Alessandro Moia, M.Tech. Venkatnarayanan Lakshminarasimhan
Submission Date: 15th October, 2021



I confirm that this Master's Thesis in Robotics, Cognition, Intelligence is my own work and I have documented all sources and material used.

Stuttgart, 15th October, 2021

Benedikt Kansy

Non-Disclosure

The presented Master's Thesis with the title

Optimization of a Collision Warning System for Electric Bicycles based on Vehicle-to-Vehicle Communication

contains internal or confidential information of the Robert Bosch GmbH company, hence locked with a non-disclosure notice, and it can only be used for examination purposes related to the Robotics, Cognition, Intelligence Master's program at the Technical University of Munich.

The contents of this work must not be accessible as a whole or in parts for any persons outside of the examination and evaluation process, unless explicitly approved by the Robert Bosch GmbH company.

Stuttgart, 15th October, 2021

Ort, Datum

Benedikt Kansy

I would like to thank Prof. Dr. Alois Knoll for supervising this thesis and giving valuable feedback despite of his many tasks and responsibilities as the head of the *Chair of Robotics, Artificial Intelligence and Real-time Systems* at the Technical University of Munich. A big thank you to Venkatnarayanan Lakshminarasimhan as well for additionally advising this thesis. I also want to thank Alessandro Moia, the advisor from Robert Bosch GmbH, for providing this challenging topic, helpful discussions and hints as well as Dr. Andreas Wienss and Florian Wildschütte for their valuable inputs and joint testing. Last but not least, thanks to Moritz Sperrfechter, Michael Giereth and Dr. Silas Klug as well as Johann Skatulla for providing measurements of the steer angle, discussions regarding sensors and test vehicle setup as well as attitude estimation, respectively.

Abstract

Pedelecs have become increasingly popular as a form of personal mobility over recent years and will likely continue to do so. Unfortunately, this also leads to more fatalities involving pedelecs. Accident research shows that the vast majority of fatalities result from collisions with other motor vehicles, often near intersections. Collision avoidance systems in cars provide insufficient protection for cyclists, and no commercial solutions that warn cyclists about oncoming collisions currently exist. Therefore, this thesis aims to improve a prototypical collision warning system for pedelecs based on Vehicle-to-Vehicle (V2V) communication with the main goal of reducing its False Positive (FP) rate. First, the positioning and attitude estimation, that serves as a base for the trajectory prediction, is improved by integrating a new Inertial Navigation System (INS). Moreover, the thesis investigates and corrects a projection error that offsets the Global Positioning System (GPS) coordinates due to high roll angles in curves. The thesis then investigates explainable physics-based approaches to improve the trajectory prediction from only naturalistic data, most importantly the roll angle. Among others, a simplistic multi-model approach that switches between different prediction methods for the lateral dynamics is proposed and analyzed theoretically. This model improves the initial baseline by 38% and 59% in the predicted position accuracy and precision at the end of a two second time horizon respectively. Furthermore, the thesis uses an efficient polygon intersection approach and demonstrates its superior Collision Detection (CD) performance in simulations with perfect positioning and trajectory predictions. Finally, it examines a more sophisticated warning strategy based on the remaining braking distance and introduces a pre-warning functionality aimed to help cyclists avoid the critical zone altogether. Final testing with a car and truck confirm the effectiveness of all improvements in greatly reducing the FP rate on common real-world scenarios.

Kurzfassung

Pedelecs haben in den vergangenen Jahren als persönliches Fortbewegungsmittel immer mehr an Beliebtheit gewonnen und werden dies wahrscheinlich auch in Zukunft tun. Leider führt dies auch zu mehr tödlichen Unfällen mit Pedelecs. Die Unfallforschung zeigt, dass die überwiegende Mehrheit der tödlichen Unfälle auf Kollisionen mit anderen Kraftfahrzeugen zurückzuführen ist, häufig in der Nähe von Kreuzungen. Kollisionsvermeidungssysteme in Autos bieten Radfahrern nur unzureichenden Schutz, und derzeit gibt es keine kommerziellen Lösungen, die Radfahrer vor drohenden Kollisionen warnen. Daher soll diese Masterarbeit ein prototypisches Kollisionswarnsystem für Pedelecs basierend auf Fahrzeug-Fahrzeug-Kommunikation verbessern, mit dem Hauptziel die Falsch-Positiv-Rate zu reduzieren. Zunächst wird die Positions- und Lageschätzung, die als Grundlage für die Trajektorienvorhersage dient, durch die Integration eines neuen Trägheitsnavigationssystems verbessert. Darüber hinaus wird ein Projektionsfehler untersucht und korrigiert, der die Koordinaten des *Global Positioning System* aufgrund von hohen Rollwinkeln in Kurven verfälscht. Die Masterarbeit untersucht dann erklärbare, physikalisch basierte Ansätze, um die Trajektorienvorhersage aus rein naturalistischen Daten zu verbessern, insbesondere der Rollwinkel. Unter anderem wird ein vereinfachter Multi-Modell-Ansatz vorgeschlagen und theoretisch analysiert, der zwischen verschiedenen Vorhersagemethoden der Querdynamik schaltet. Dieses Modell verbessert die vorhergesagten Positionsrichtigkeit und Präzision am Ende eines zweisekündigen Zeithorizonts um 38% und 59% verglichen mit der ursprünglichen Ausgangslage. Darüber hinaus wird in dieser Masterarbeit ein effizientes Verfahren zur Überlappung von Polygonen genutzt und sein positiver Einfluss bei der Kollisionserkennung in Simulationen mit perfekter Positions- und Trajektorienvorhersage demonstriert. Schließlich wird eine ausgereifte Warnstrategie untersucht, welche auf dem verbleibenden Bremsweg basiert. Außerdem wird eine Vorwarnfunktion einführt, welche Radfahrern helfen soll die kritische Zone gänzlich zu vermeiden. Abschließende Tests mit einem Personenkraftwagen und Lastwagen bestätigen die Wirksamkeit aller Verbesserungen durch eine deutliche Verringerung der Falsch-Positiv-Rate in gängigen Szenarien.

Notation

This section briefly describes the notation used throughout the thesis. The mathematical typesetting and syntax rests on [1], whereas the decimal separators and frame attachments are based on [2] and [3], respectively. Table 1 summarizes some of these key notations and provides examples.

Item	Notation	Example
scalar variables	<i>italic</i>	speed: v
column vector variables specification of vector element of vector	<i>italic</i> , boldface and lowercase upright subscript <i>italicized</i> subscript	velocity: v maximum velocity: v_{\max} i-th element of v : v_i
matrices	<i>italic</i> , boldface and UPPERCASE	transformation matrix: T
numbers, functions and units	upright	number five: 5, cosine function: $\cos()$, unit: $\frac{\text{m}}{\text{s}^2}$
decimal separators (English SI)	period to separate decimals	5.500 (five and a half)
software constructs (variables, classes, etc.)	typewriter font	count variable: cnt
frame attachments	in curly braces and typewriter font	bicycle frame: {Bicycle} or {B}

Table 1.: Key Points of Utilized Notation

Some terms in the continuous text may also be italicized as a means of *emphasizing* them. The trajectory prediction models are named according to the prediction modes and names of the chosen signals for their longitudinal and lateral components as described in the following naming convention:

[lon. pred. mode]_[lon. signal]_[lat. pred. mode]_[lat. signal]

If the prediction mode is the same for the longitudinal and lateral component of the model, the model name may be shortened as follows:

[lon. and lat. prediction mode]_[lon. signal]_[lat. signal]

For example, a model that holds both the acceleration as well as the yaw rate signal constant over the prediction horizon is referred to as const_a_const_yawr or const_a_yawr.

Finally, since the juridical distinctions among *Pedelecs*, *S-Pedelecs* and *e-Bikes* do not critically affect the research goals of this master thesis, the terms are used interchangeably and all refer to an electrified bicycle.

Contents

Acknowledgments	iv
Abstract	v
Kurzfassung	vi
Notation	vii
1. Introduction	1
1.1. Motivation	1
1.2. Challenges	3
1.3. Definition of Task	4
1.4. Thesis Outline	5
2. Fundamentals	6
2.1. State-of-the-Art Collision Warning Systems for Bicycles	6
2.2. Overview of Demonstrator (Pedelec)	7
2.3. Reference Coordinate Frames	8
2.3.1. Earth-Centered Inertial Reference Frame	9
2.3.2. Earth-Centered Earth-Fixed Reference Frame	9
2.3.3. Geodetic Reference Frame	9
2.3.4. Street Reference Frame	10
2.3.5. Pedelec Reference Frame	11
2.3.6. Sensor Reference Frames	13
2.4. Inertial Navigation System	15
2.5. Vehicle-to-Vehicle Communication	16
2.6. Separating Axis Theorem for Collision Detection	17
3. Related Work	20
3.1. Previous Works	20
3.2. Related Research	28
3.2.1. Overview of Relevant Research Fields	29
3.2.2. Collision Warning Systems for Bicycles	29
3.2.3. Modeling the Dynamics and Stability of Single Track Two-Wheeled Vehicles	31
3.2.4. Trajectory Prediction for Vulnerable Road Users	34
3.3. Thesis Research Focus	36

4. Solution Approach	38
4.1. Analysis of Initial State	38
4.1.1. Overall Software Framework	38
4.1.2. Visualization and Quality of Sensor Signals	39
4.1.3. Heading and Position Estimation	46
4.1.4. Statistical Evaluations of Trajectory Errors	50
4.1.5. Motion Primitive	51
4.1.6. Trajectory Prediction Models	52
4.1.7. Collision Detection and Warn Strategy	57
4.2. Conceptual Overview	58
4.3. Conducting Automated Scientific Experiments	62
4.4. Adapts and Additions to Visualizations for Analysis Purposes	63
4.4.1. New Visualization Concept	63
4.4.2. Quality of Newly Added Signals	64
4.5. Virtual Vehicle for End-to-End Validation	66
4.6. Integration of Reference Sensors	67
4.7. Offsetting and Correcting Pedelec Position	70
4.8. Development of Physics-based Trajectory Prediction Models	74
4.8.1. Segmenting Trip into Different Riding Situations	74
4.8.2. Multi-Model Approach	77
4.8.3. Longitudinal Component	78
4.8.4. Lateral Component	82
4.9. Collision Detection and Warning Strategy	92
4.9.1. Collision Recognition via Intersecting Rectangles	93
4.9.2. Finding the Last Safe Location	99
4.9.3. Alarm Based on Safe Breaking Distance	101
4.9.4. Multi-Staged Warning Using Pre-Warnings	103
4.10. Adapts to Vulnerable Road User Awareness Message	105
5. Evaluation	106
5.1. Intrinsic Evaluation via Trajectory Prediction Errors	106
5.1.1. Influence of Stationary Periods	106
5.1.2. Influence of State Estimation and Inertial Measurements	108
5.1.3. Influence of Route	110
5.1.4. Influence of Selected Sensor Signal for Longitudinal Component	113
5.1.5. Influence of Selected Sensor Signal for Lateral Component	121
5.1.6. Comparison of Trajectory Prediction Models	126
5.2. Extrinsic Evaluation via Collision Warning Function	137
5.2.1. Influence of State Estimation and Inertial Measurements	137
5.2.2. Influence of New Trajectory Prediction Models	138
5.2.3. Influence of New Collision Detection and Warning Strategy	139
5.2.4. Qualitative Comparison	141
5.3. Discussion of Results	146

6. Future Work	153
7. Conclusion	157
A. Appendix	160
A.1. Review of Homogeneous Transformation	160
A.2. Full Derivations	161
A.3. Supplementary Tables	163
A.4. Supplementary Figures	163
Acronyms	170
List of Figures	173
List of Tables	176
Bibliography	178

1. Introduction

The following chapter quickly outlines the relevance, underlying motivation and challenges of the topic, the specific problem definition as well as a rough outline of this thesis.

1.1. Motivation

E-bikes have been gaining strong popularity in the last few years and e-bike sales will likely continue to rise. For example, over the last decade, e-bike sales in Germany have continually risen every year from two hundred thousand in 2010 to almost two million units sold yearly in 2020. Especially the last three years showed a strong trend in growing sales with 2020 achieving the biggest increase in sales per year ever [4]. E-bikes already make up 38.7% of the bicycle market in 2020, and the German association for Two-Wheelers forecasts a short term market share of 40% and medium to long term market share of 50% [5]. However, there might exist a negative trend when it comes to the fatal accidents involving e-bikes. Despite the fact that the number of fatalities has decreased gradually in Germany over the last few decades, the Federal Statistical Office of Germany reported an increase in fatally injured e-bike riders of 19.1% in 2020 compared to 2019 while conventional cyclist fatalities decreased [6]. Accident researcher Brockmann believes that (especially older) customer struggle to handle the faster and heavier e-bikes in critical situations [7]. Pedelecs thus make up an extremely exposed share of traffic participants: They are much quieter than motorcycles and faster than pedestrians which makes them harder to judge but one of the most vulnerable road users when involved in a collision.

While cars have benefited from multiple Advanced Driver Assistance Systems (ADAS) functions like Adaptive Cruise Control (ACC) or Autonomous Emergency Braking (AEB), bicycles have barely improved their safety in the last decades. According to the European Transport Safety Council (ETSC), bicycles showed no reduction in road deaths in 2018 compared to 2010, opposed to a 19% decrease for pedestrians and 24% decrease for cars [8]. The electrification of the bicycle drive train now offers the possibility of conveniently equipping bicycles with similar ADAS functions as enough electricity is available to power these systems. The ETSC also reports that, in the EU, 83% of all cyclist deaths results from a collision with another motor vehicle [8]. About 28% of these fatal accidents occur at intersections in EU [9]. Common reasons for accidents at intersections include a lacking line of sight as depicted in Figure 1.1 or overlooking of another road user because of temporary occlusion e.g. by the A-pillar of the vehicle [10]. The blind spot of vehicles marks another dangerous collision situation for cyclists.

1. Introduction

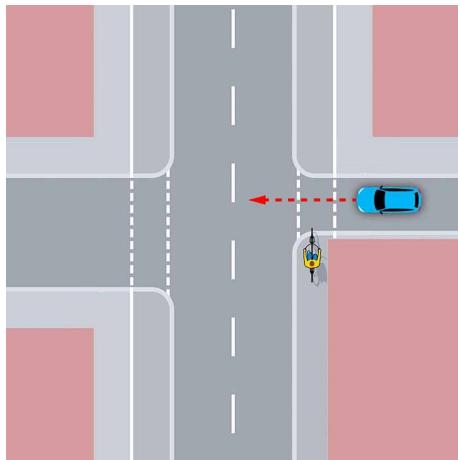


Figure 1.1.: Intersection Without Line of Sight

Collisions at intersections without line of sight often can not be prevented by state-of-the-art safety system of cars as their camera, Radio Detecting and Ranging (Radar) or even Light Detection and Ranging (Lidar) sensors cannot detect objects occluded by infrastructure such as buildings. In addition, AEB for cars have extremely low FP rate requirements. A FP event represents the case when the system falsely detects a "pseudo" collision which does not actually occur. This means, the AEB of a car will only get triggered if a crash is unavoidable which in return leads to an extremely high False Negative (FN)-rate where the system fails to detect lots of actual crashes. Moreover, a test in the European New Car Assessment Programme (NCAP) awards the full crash safety rating to cars travelling over 40 km/h if they manage to reduce the speed of a collision by at least 20 km/h [11]. For example, a car traveling at 50 km/h may hit a cyclist at 29 km/h and receive full points, despite the impact speed potentially killing the cyclist.

Knowing that a car cannot physically sense some collision with its current sensors and will only brake automatically if it is nearly 100% certain that a crash is unavoidable, it may be beneficial to warn the cyclist about potential oncoming collisions with a car instead. Warning the cyclists may not require those extremely low FP rates and thus enable a lower FN-rate compared to cars. Another advantage in warning the cyclist lies in the fact that bicycles generally move slower than cars. This means that warning the cyclist can happen at a later point in time compared to the car to achieve a complete stop. A later warning point means a shorter time span in the future needs to be predicted.

Some of these scenarios could be prevented through V2V communication which allows road users to share their headings, positions, speeds, acceleration, turning rates etc. with each other to predict their future trajectories and detect potential collisions. Accident research shows that V2V may potentially prevent between 41.0-57.8% of accidents occurring at intersections compared to only 22.9-35.7% with traditional camera and Radar based systems [10] [12].

To summarize, pedelec sales have been exploding recently and will likely continue to do so. However, bicycle safety has barely improved in the last decade and most fatal collisions occur with cars, often at intersections. Accident research claims that V2V might generally be more efficient than conventional intersection assistants and some collisions at occluded intersections may only be prevented through communication between the cyclist and the other road user. Warning the cyclist instead of the car may enable more crashes being detected while having to predict shorter time horizons. These arguments show that research in V2V-based collision warning systems specifically designed to warn e-bike riders is highly relevant and may

improve the overall safety on the roads.

1.2. Challenges

This section introduces some of the unique challenges in developing a collision warning system for pedelecs based on V2V compared to other vehicles. Compared to cars, the following factors may complicate the prediction of pedelec trajectories: First, pedelecs generally possess more complex vehicle dynamics that may not be simplified as easily as in cars. For example, pedelecs may experience a significant roll motion when leaning into turns while this can often be neglected in cars. In addition, bicycles are inherently unstable and generally require the cyclist to perform many corrections to follow a certain path. For example, pedalling causes lateral movements of the bicycle which may have to be counteracted by small steering inputs of the cyclist. Also, cyclists tend to evade small bumps or rocks on the roads more frequently than car drivers to avoid falls. Moreover, cyclists usually have more path choices to begin with given a certain route. They can alternate between riding on a sidewalk, the right-hand side of the lane or near the center markings (e.g. when turning left), whereas cars typically are confined to stay centered in their own lane. Car drivers also seem to comply with the traffic rules better than cyclists, who for example often change directions on a short notice or without signaling at all, leading to less predictable trajectories. Furthermore, a pedelec generally has far inferior sensing capabilities compared to cars because of the lower price point. Pedelecs typically do not feature cameras, Radar or even Lidar sensor to perceive the environment they are in, which could provide useful indications about the intended trajectory. For instance, a perceived stop sign or intersection might indicate the cyclist will slow down or turn. To make matters worse, the bicycle typically faces more vibrations due to less sophisticated suspension (e.g. hard-tail) and very thin tire walls for absorbing bumps compared to cars. This may result in noisy sensor signals that are harder to process. Besides, the pedalling may cause undesired oscillations as well. All of these reasons may result in more complex paths that are less smooth and harder to predict than for cars.

As a result of these complications, simple trajectory prediction models used for cars may only work to a certain extend for bicycles even for short-term prediction horizons. Figure 1.2 visualizes this by deploying the same trajectory prediction model to a car (green) and a pedelec (red). A new trajectory is computed every 50 milliseconds (20 Hz) and the prediction horizon of each trajectory is two seconds into the future in this example. After some initial turns, the pedelec and the car drive parallel to each other roughly into the East direction. Note how the cyclist does not accomplish to ride a fully straight path despite intending to go straight as seen by the orange dots. This figure also shows how the highly complex dynamics of the pedelec cause the red trajectories to swing significantly to the left and right of the actual path, even when going straight. On the other hand, the car can drive in a perfectly straight line and its corresponding trajectories do not swing at all when going straight. This shows that predicting trajectories of a pedelec is significantly harder than for a car even for the supposedly easiest scenario, which is driving in a straight line.

1. Introduction

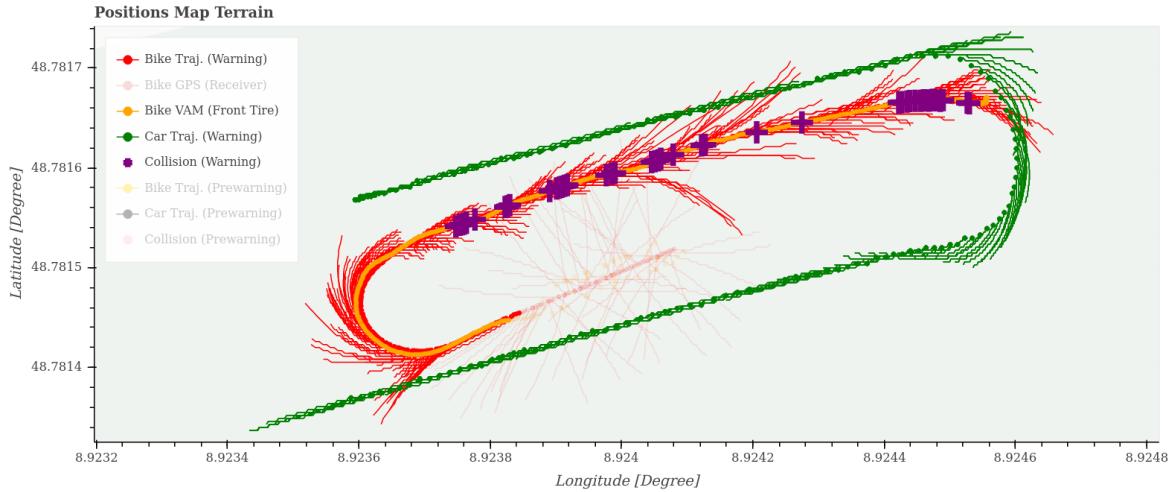


Figure 1.2.: Visualized Trajectories for Pedelec and Car

Compared to conventional bicycles, pedelecs typically accelerate faster and reach higher speeds due to the assistance by the electric motor. At the same time, the additional weight of the battery and motor typically amount to longer braking distances. Both of these factors generally require pedelecs to predict further into the future compared to conventional bicycles in order to prevent accidents.

Of course, there exist many further general issues applying to all road users that are not further discussed such as accurate positioning in dense urban areas and achieving a cooperative, reliable and efficient communication between them.

1.3. Definition of Task

Previous works build up a basic demonstrator that laid a foundation for the development a collision warning system for pedelecs via V2V communication. While the demonstrator is able to detect collisions, it does so at the cost of an extremely high FP rate as shown in Figure 1.2. This represents the common scenario of a cyclist riding on a sidewalk or bicycle lane and temporarily being in the rear-quarter view (blind spot) of the car which is driving on the road. The driver of the car oversees the pedelec rider when turning right and hits the cyclist who intends to keep going straight. In this case, only the last purple crosses towards the right-hand side of the figure represent True Positive (TP) alarms. All other warnings before that are considered FPs as the pedelec and car were driving parallel to each other with a safety margin of around 2-3 meters which did not result in any collisions or critical situations.

The aim of this thesis is to gain insights on what factors contribute to these FP warnings

and improve the ADAS function with the main goal of reducing the number of FP collision warnings without sacrificing the other metrics. Some research question include: How can the trajectory predictions for the pedelec be improved in an explainable way? Since the ADAS function is safety critical for the cyclist and other road users, being able to retrace and understand how and why a collision was detected or missed is crucial. What are the expected prediction accuracies and precisions for different horizons in the future? How could physics-based approaches be combined with Machine Learning in a promising way? When do we need to incorporate more information or constraints to the naturalistic data? How can the collision detection and warning strategy be improved to detect less cases where the road users may come close to each other but will not collide? What is a useful time horizon to warn the cyclist? Warning too soon may confuse the rider as to which situation the warning is referring to or lead to high FP rate which may both cause the rider to lose trust in the ADAS function. Warning too late would mean the rider does not have enough time to react and brake to actually prevent a crash.

1.4. Thesis Outline

The thesis first introduces some topic specific fundamentals in chapter 2 e.g. an overview of the existing demonstrator, a definition of the used reference coordinate frames, a clarification on a Inertial Navigation System, a summary of very basic V2V terminologies and relevant message types as well as an explanation of an algorithm used for detecting polygon intersections efficiently. Then, chapter 3 reviews the contributions of previous works to the demonstrator and ADAS function as well as providing an overview of relevant research works before clarifying the research scope of this thesis. At that point, chapter 4 describes the solution approach in detail from critically analyzing the initial state, providing an overview of the overall software framework before diving into the specific improvements and novel concepts contributed in the course of this thesis. Subsequently, chapter 5 first evaluates the influence of these improvements intrinsically e.g. on the trajectory prediction errors but also compares the errors of these new prediction models with the ones from previous works. Furthermore, this chapter extrinsically evaluates the improvements on the overall collision warning system and compares it with the state at the beginning of this thesis before discussing all of the results. Finally, chapter 6 states some further improvements and ideas for future works and chapter 7 concludes the thesis.

2. Fundamentals

This chapter covers the fundamentals specific to understanding the topic of this thesis. A general robotics background of the reader is assumed and common concepts such as rotation matrices, Kalman Filters, the publisher-subscriber model of the Robot Operating System, convex sets, etc. are not explicitly repeated.

In the beginning, a forecast of current state-of-the art approaches for warning Vulnerable Road User (VRU) about potential collisions. While there does not yet exist a commercially available collision warning system for cyclists, the first section discusses current efforts by companies to introduce such a product into the market. Following that, the hardware setup of the test vehicle (pedelec demonstrator) used in the course of this thesis is described briefly. Afterwards, the different type of reference coordinate systems for describing and transforming data from the various sensors, the pedelec, the street and world frames are defined. Then, a section quickly introduces the technologies inside an INS and its differences to an Inertial Measurement Unit (IMU) and Attitude and Heading Reference System (AHRS). Towards the end, some basic V2V terms and two prevalent message formats used in this thesis are briefly covered. Finally, a simple but efficient algorithm used in this thesis to detect whether two convex polygons collide in two-dimensional (2D) space is explained.

2.1. State-of-the-Art Collision Warning Systems for Bicycles

While most car manufacturers offer collision warning systems that warn drivers about VRUs such as pedestrians and cyclists, no commercially available collision warning systems exist yet that actively warn bicycle riders, especially in scenarios without line of sight. Furthermore, the collision warning system for cars often do not reliably detect and predict the trajectories of pedelecs yet. For example, Volvo, who introduced the world's first commercial auto brake system reacting to cyclists in 2013, states that their automated braking assistant in their latest V40 model only detects cyclists from behind which are stationary or moving in the same direction as the vehicle [13]. They further state that partially occluded cyclists, cyclists wearing body contour obscuring clothing, bicycles without a rearward-facing red reflector or bicycles loaded with large objects cannot be detected [14].

However, Continental and Deutsche Telekom are currently developing a collision warning system to protect VRUs such as cyclists, pedelecs, scooter riders or pedestrians by using GPS and acceleration sensors as well as the mobile communications of the users' smartphones to transmit this data to a cloud where the predictions and warnings are computed. The companies strive to present their results at the Intelligent Transport Systems (ITS) World Congress in middle of October, 2021 [15] [16]. Published research attempts that have not yet

been introduced to the market are summarized in subsection 3.2.2

2.2. Overview of Demonstrator (Pedelec)

This section briefly presents the hardware setup of the test vehicle demonstrator as found at the start of this thesis. It merely summarizes what type of hardware components are used and how they generally interact, whereas the software structure for the collision warning function is described further in section 3.1. For explanations why certain components are chosen and a more detailed description of the hardware setup, the reader is referred to [17] that built up this demonstrator.

Figure 2.1 depicts the *Centurion E-Fire Tour R2600i ABS* pedelec that serves as the base of the test vehicle [18]. To avoid cluttering, only the relevant sensors and actuators for the ADAS function are highlighted in blue and labelled with a number. Note that some sensors were replaced over the course of this thesis as they provided data of insufficient signal quality as explained in subsection 4.1.2 and subsection 4.4.2.

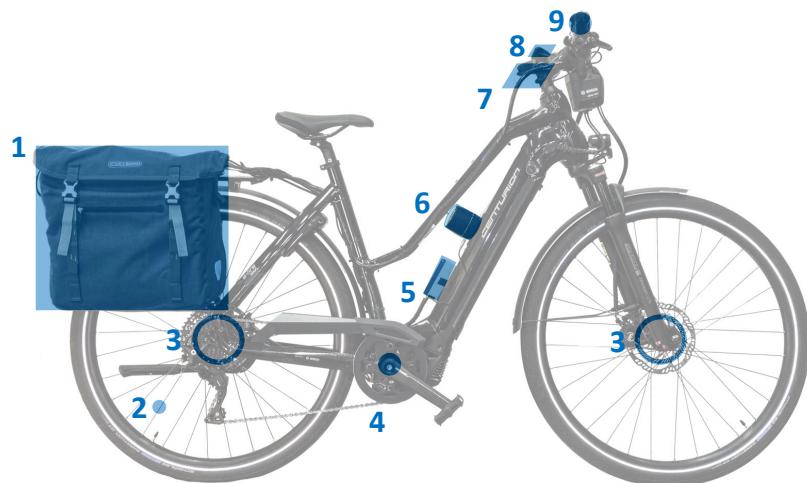


Figure 2.1.: Test Vehicle with Mounted Processing and Measurement Hardware

The factory configuration of this specific pedelec features a complete drive train and Anti-lock Braking System (ABS) by Bosch. Potentially relevant sensors for the collision warning function are a standard wheel speed sensor capturing one pulse per revolution (2) and a high-resolution 56-toothed wheel speed sensor for its ABS (3) as well as a motor torque and cadence sensor from the drive unit (4). The pedelec is powered on by the Kiox board computer (8) that also displays the current speed, riding mode, state-of-charge of the accumulators etc. All of these sensors are connected to the Controller Area Network (CAN)-based communication bus of the pedelec¹

¹The accumulator charging system, the ABS actuators as well as an internal IMU within the drive unit are also

Furthermore, a consumer grade AHRS (5), namely the *Bosch Sensortec BNO055 USB-Stick Eval Kit (BNO055)*, is mounted additionally to the pedelec frame to measure accelerations, angular velocities and estimate the absolute orientation such that the yaw rates can be transformed from the sensor frame into the street frame where the predicted trajectory is projected to [19]. The rear pannier (1) mounted to the luggage rack contains a Direct Current-to-Direct Current (DC-DC), a *Intel® Next Unit of Computing (NUC)*, a *Ixxat® USB-to-CAN V2* USB interface, a Bosch developed prototype *Communication Control Unit (CCU)*, a *Navilock NL-8012U* Global Navigation Satellite System (GNSS) USB-dongle as well as a custom-designed signal amplifier and a *Sinus Live® GL-205* ground-loop-isolator [20] [21] [22] [23]. The DC-DC converts the 36V of the pedelec's accumulator to appropriate voltages to power all of the added components. The NUC acts as the vehicle computer for processing all of the sensor signals and computing the ADAS function. The sensors are connected to the NUC either directly via USB or indirectly through the USB-to-CAN V2 USB interface. The CCU handles the bidirectional V2V communication with other road users and the GNSS dongle obtains absolute positions of the pedelec.

The hardware used to alert cyclists consists of a portable *JBL GO 2* speaker (6) and custom-designed vibrating handlebars (9) which are both triggered via the audio output of the NUC. The audio signal is split, where one string directly attaches to the speaker and the other one is passed through the amplifier and ground-loop isolator. The amplifier is necessary for the analog signal to power the small electric motors inside of the handle bar that produce the vibration and the isolator galvanically decouples the grounds to provide a clear signal transmission without constant low-frequency vibrations. For development purposes, an additional display (7) allows to control the Linux operating system of the NUC and display arbitrary information like connection statuses of the GNSS, AHRS and CCU components and further visual warnings.

2.3. Reference Coordinate Frames

This thesis deals with description of objects using different representations. To avoid confusion throughout the thesis and to uniquely describe the position the attitude of the sensors and pedelec in relation to each other and to the street coordinates, this section provides an overview of the involved coordinate frames in a top-down manner. First, generic reference systems for describing the earth in the universe are introduced. Then, the geodetic reference system for describing GPS locations using longitudes and latitudes on earth are described and how they can be converted to a local street frame. Towards the end, the position and orientation of the pedelec within this street frame where the collisions are computed is introduced. Last but not least, this section describes the various frames in which the sensor data is actually collected in relation to the pedelec frame, based on their mounting alignments.

connected to the bus but not further highlighted in Figure 2.1 as they are not directly used in the targeted ADAS function.

2.3.1. Earth-Centered Inertial Reference Frame

In order to describe motions of celestial objects such as satellites or spacecrafts, a non-rotating coordinate frame is useful. Thus, there exists a stationary Earth-centered inertial (ECI) coordinate frame $\{i\}$ whose origin lies at the center of mass of the Earth and its axes are fixed with respect to the stars [24]. The plane spanned by the x and y axes cut through the equator, and the z -axis extends through the Earth's instantaneous rotational axis at a certain terrestrial time e.g. 12:00 on January, 1st 2000 for the commonly used J2000 ECI frame [25]. An IMU measures angular velocities and linear accelerations with respect to this frame.

2.3.2. Earth-Centered Earth-Fixed Reference Frame

Describing the motion of objects on the Earth's surface is more convenient using an Earth-centered - Earth-fixed (ECEF) coordinate frame, denoted with $\{e\}$. Hereby, the origin and axes of the generally coincides to $\{i\}$, but the frame rotates with the earth, opposed to staying fixed with the stars [26]. More precisely, the z -axis this time extends through the true North, which may not coincide with the instantaneous rotational axis [27]. The x -axis extends through the intersection of the equator and prime meridian (Greenwich). Since the frame rotates with the earth, any point fixed to the Earth's surface can be described uniquely in this Cartesian coordinate system by using a three-dimensional vectors representing the x -, y - and z -coordinates, where $(0, 0, 0)^T$ represents the earth's center of mass. Alternatively, the horizontal location on the earth's surface can be expressed a geocentric longitudes and latitudes. Lines of longitudes are called meridians. Generally speaking, a *geocentric* longitude represents a signed angle of a meridian in reference to a reference meridian, usually at or near the Greenwich Zero Meridian. A longitude of 0° is thus defined at or near the Greenwich Zero Meridian, positive for the Western Hemisphere and negative for the Eastern Hemisphere. The *geocentric* latitude of a point on the earth's surface represents the signed angle between the equatorial plane and the point measured from the earth's center of mass. A latitude of 0 thus represents the equator, while the North pole is at $+90^\circ$ and the south pole at -90° [28]. Figure 2.3 shows how the ECI and ECEF frames relate. A special type of ECEF is described in the next section.

2.3.3. Geodetic Reference Frame

A geodetic reference frame is a special type of ECEF for representing a location on or near the Earth's surface in latitude, longitude and optionally altitude relative to a reference point. There exist various geodetic system but World Geodetic System 1984 (WGS 84), the reference frame for the American GNSS standard GPS, is used most commonly for cartography, geodesy and satellite navigation². WGS 84 takes the earth's eccentricity, gravitational field and rotation rate into account and thus models it as a flattened ellipsoid rather than a

²There exist other variants of GNSS such as GloNASS (Russian), Galileo (European) and BeiDou (Chinese) that use different geodetic reference frames, namely Parametry Zemli 1990, Galileo Terrestrial Reference Frame and BeiDou Coordinate System [29]

sphere [30]. While the geodetic longitude is defined similar the geocentric longitude³, as explained in subsection 2.3.2, the geodetic latitude differs from the geocentric one. In this case, the angle is measured between the surface normal at the point and the equatorial plane, opposed to the radius originating from its center of mass. Figure 2.2 illustrates the differences in the geodetic and geocentric latitudes for the same point on the earths surface.

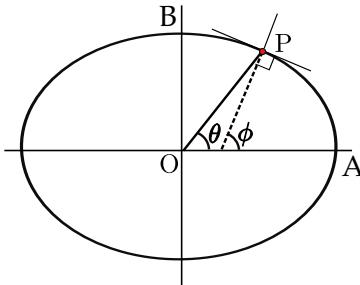


Figure 2.2.: Geodetic Latitude ϕ and Geocentric Latitude θ [32]

Expressing a location on the earth's curved surface onto a plane e.g. the street frame as described in subsection 2.3.4 requires a map projection. Converting from WGS 84 is achieved using the also known as EPSG:4326 projection method. Note that Google Maps uses a simpler, more efficient but slightly inaccurate Spherical Mercator projection method called EPSG:3857 instead [33] [34].

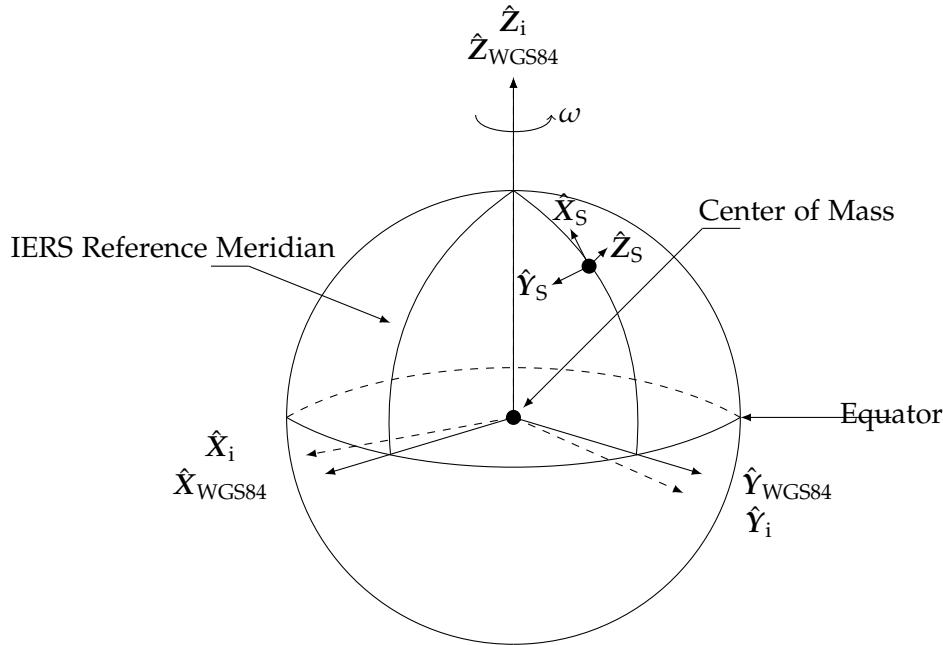
2.3.4. Street Reference Frame

The Street frame denotes a local geographic frame where the trajectory prediction takes place. In this thesis, the Street frame is expected to stay fixed with respect to the earth. If the sensor travel large distances, it may make sense to translate and rotate the Street frame along the surface of the earth.

While positions from the GNSS may be measured and reported in the WGS84 frame (see) and the accelerations and angular speeds in the AHRS frame, this data need to be transformed into the street plane for comparing it with the car (whose data also needs to be transformed to this frame). For example, we care about ${}^S\omega_z$, i.e. how fast the pedelec changes the heading direction relative to Street. Since the cyclist may lean into the curve, the AHRS would sense angular speeds around multiple axes of Bicycle simultaneously and needs to be transformed such that ${}^S\omega_z$ contains most of the information regarding the cyclist's change of direction. This is achieved by minimizing the pitch and roll rate of the pedelec in Street.

Analogous to DIN ISO 8855, the Street frame consists of an axis system whose \hat{X}_S - and \hat{Y}_S axes lay in in the horizontal plane and its \hat{Z}_S axis points up to the sky [35]. Figure 2.3 visualizes how the i, WGS85 and Street frames relate.

³In WGS 84, the International Earth Rotation Service (IERS) Reference Meridian is actually around 102 m east of the Greenwich meridian at the latitude of the Royal Observatory [31]


 Figure 2.3.: Reference Frames $\{i\}$, $\{WGS84\}$ and $\{\text{Street}\}$

2.3.5. Pedelec Reference Frame

To uniquely identify the attitude of the pedelec with respect to the $\{\text{Street}\}$ frame, a right-handed Cartesian $\{\text{Bicycle}\}$ coordinate system is constructed based on DIN ISO 8855 [35]. Note that the term *bicycle frame* can denote either the pedelec's physical metal frame or the coordinate system. In this section, the term frame always refers to the coordinate system. Figure 2.4 shows the assignment of this coordinate frame from the side and top view of the pedelec.

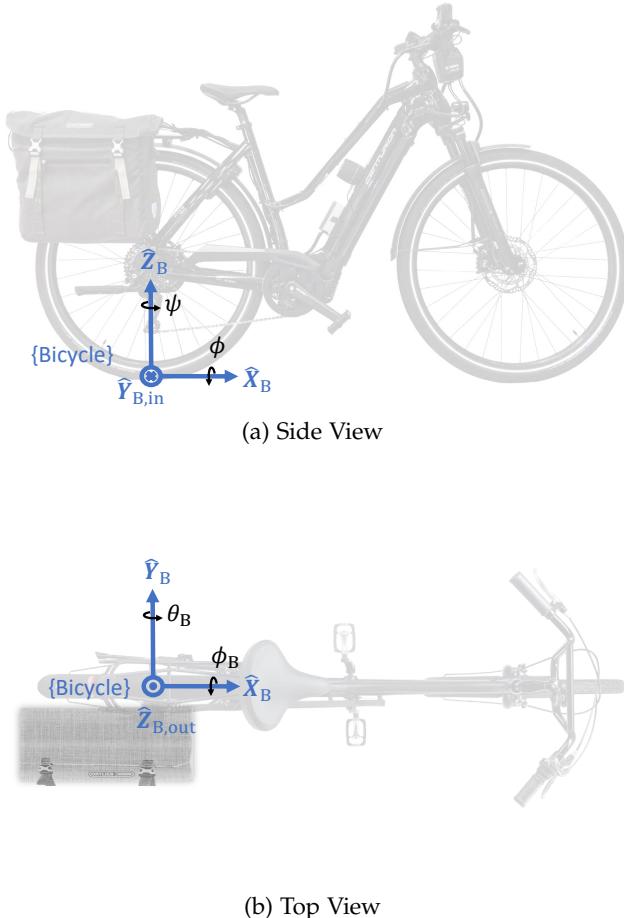


Figure 2.4.: Pedelec Reference Frames

The contact point of the rear wheel with the street forms the vehicle reference point, i.e. the origin of the bicycle frame. The x-axis extends from the origin along the pedelec's longitudinal axis into its driving direction. As marked with a cross, the y-axis extends from the origin along the pedelec's lateral axis into the sketch when seen from the side view. The z-axis extends from the origin up towards the sky, orthogonal to the xy-plane.

Rotations about the x-axis, y-axis and z-axis of the pedelec are referred as roll, pitch and yaw rates and denoted with ω_x , ω_y and ω_z respectively. The roll, pitch and yaw angles are Euler-angles relative to a user-defined earth-fixed coordinate system, in this case the street coordinate frame as described in subsection 2.3.4. For this thesis only the roll angle, denoted with ϕ , and yaw angle, denoted with ψ , will be of significance. The signs of the angles are determined by the direction of rotation. A positive angle corresponds to a motion in the positive direction of rotation as defined by the right hand rule. For example, a right turn with leaning into the curve from the point of view of the rider would correspond to a negative yaw rate and positive roll rate. Note that in this case, the yaw rate in bicycle frame is not the same as the yaw rate in the street frame as explained already in subsection 2.3.4. Based on

the definition of the street coordinate system, $\phi = 0$ corresponds to the bike in its upright riding position on a flat road and $\psi = 0$ denotes an eastbound driving direction.

2.3.6. Sensor Reference Frames

The hardware setup of the test vehicle consists of several sensors. However, each sensor typically reports its measurements with respect to its own body coordinate frame. Since the sensors might be mounted different angles, their measurements may not correspond with each other or the vehicle they are actually trying to capture. In this case, it can be useful to transform all of the measurements into the same unified coordinate frame of the vehicle, e.g. the {Bicycle} frame as defined in subsection 2.3.5.

The coordinate of the moving AHRS, denoted with {AHRS} originates from the center of the accelerometer triad and its axes are aligned with the case as shown in Figure 2.5 for the *BNO055 USB-Stick*.

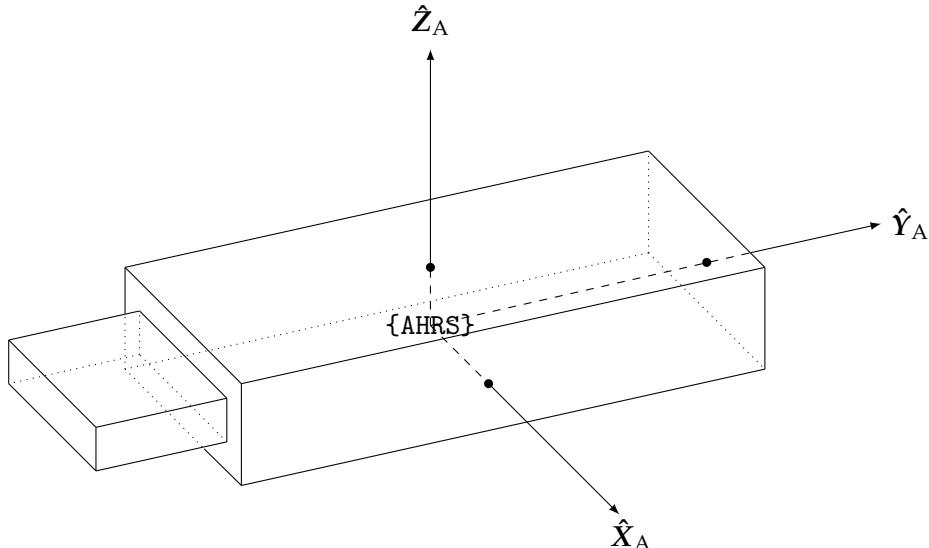


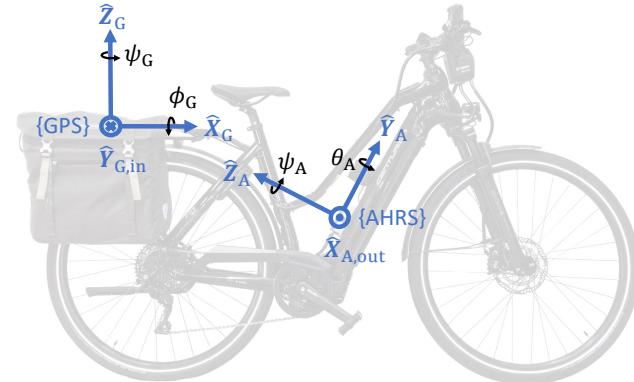
Figure 2.5.: Coordinate Frame of the *BNO055 USB-Stick* AHRS. Adapted from [17]

Note that the axes of the sensors frame generally are specific to the individual sensor and sometimes can be re-configured to the liking of the user. All of the reported inertial and gyroscopic as well as attitude solutions from the fusion algorithm are resolved with respect to this {AHRS} frame. In our case, placing the *BNO055 USB-Stick* on a flat surface, similar to Figure 2.5, and pointed into the magnetic North results in all measured angles equalling zero (or 360) [36].

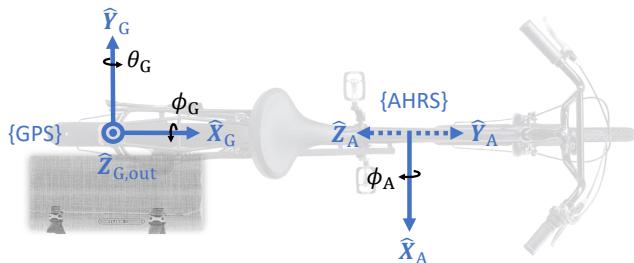
The coordinate frame of the GNSS, denoted with {GPS}, is placed at the center of the antenna with its axes set arbitrarily. In contrast to the AHRS, the positions of the GNSS are reported at the origin of {GPS} but with respect to the {WGS84}frame.

2. Fundamentals

Figure 2.6 shows $\{\text{AHRS}\}$ and $\{\text{GPS}\}$ frames at their actual mounting position in the side and top view of the test vehicle introduced in section 2.2.



(a) Side View



(b) Top View

Figure 2.6.: Sensor Reference Frames

The necessity of transforming the sensor coordinate frame of the AHRS to the bicycle frame becomes evident. The axes of $\{\text{AHRS}\}$ are flipped and additionally rotated by the mounting angle with respect to $\{\text{Bicycle}\}$. For example, if the bicycle rode straight, the AHRS reports both \hat{Y}_A and \hat{Z}_A accelerations while the bicycle only experiences an acceleration in its longitudinal \hat{X}_A -component. Similarly, the the GPS position is reported at the origin of $\{\text{GPS}\}$, which is offset from the origin of $\{\text{Bicycle}\}$ that corresponds to the location of the contact point of the pedelec's rear wheel with the street⁴.

⁴The $\{\text{Bicycle}\}$ frame is already depicted in Figure 2.4 and hence not drawn in again to avoid cluttering

2.4. Inertial Navigation System

This section first briefly describe how inertial sensors can be used to estimate position and orientation, sometimes referred to *pose estimation*. Then, it explains the key differences of INS to the closely related terms IMU and AHRS as well as explaining some other common technologies of modern INS.

A Micro Electro Mechanical System (MEMS) accelerometer generally measures the specific inertial force f_s in its AHRS body frame $\{A\}$ expressed in Equation 2.1, opposed to the actual acceleration [26].

$${}^A f_s = {}^A R ({}^S a_{ii} - {}^S g) \quad (2.1)$$

Hereby, g denotes the gravity vector and ${}^S a_{ii}$ the sensor's linear acceleration in the Street frame $\{S\}$ used for navigation. The term ${}^S a_{ii}$ can be further expressed in terms of the acceleration required for navigation purposes ${}^S a_{nn}$ as shown in Equation 2.2⁵.

$${}^S a_{ii} = {}^S a_{nn} + 2^n \omega_{ie} \times {}^S v_S + {}^n \omega_{ie} \times {}^n \omega_{ie} \times {}^S p \quad (2.2)$$

The term $2^n \omega_{ie} \times {}^S v_S$ represents the represents the Coriolis acceleration and ${}^n \omega_{ie} \times {}^n \omega_{ie} \times {}^S p$ represents the centrifugal acceleration [26].

The gyroscopes measure the angular velocity of the AHRS body frame $\{A\}$ with respect to the inertial frame $\{i\}$. Similarly to the acceleration, this measurement thus includes other factors such as the angular velocity of the earth around the inertial frame $\{i\}$ or the transport rate in case the street frame moves with respect to the earth.

Velocities can generally be obtained by integrating the gravity and Coriolis corrected accelerations over time in each direction. A position relative to an initial location can be obtained by integrating the velocities again. However, integrating inherently noisy accelerations causes the computed positions to drift away from the actual path. The longer the integration is, the greater the drift accumulates. Similarly, relative orientations can be computed by integrating the angular speeds from the gyroscopes.

An IMU is usually referred to a combination of MEMS accelerometer and gyroscope sensors [26]. For example, three orthogonally placed accelerometers and three orthogonally placed gyroscopes can capture all 6 Degree of Freedom (DoF) of a freely moving rigid body (translation in x-,y- and z-axis and rotation about x-,y- and z-axis). Sometimes, three-axis magnetometers and temperature sensors are additionally included to extend to a 9 DoF IMU for reducing drift [37]. Classical IMUs do not feature a smart system and just output the raw sensor values.

An AHRS extends the IMU with a magnetometer and Microcontroller Unit (MCU) that

⁵This relationship is derived by rewriting the velocity and acceleration using its differential expressions and frame transformations while assuming that the navigation frame is fixed to the earth frame. The full derivation, in a slightly different notation, can be found in [26] from page 10-11.

fuses its sensors to provide more accurate and drift reduced three-dimensional (3D) orientations than if purely integrated (full attitude including the heading relative to North) [38]. An AHRS can thus also be seen as a *smart* IMU as the case with the *BNO055*, introduced in section 2.2.

An INS consists of an IMU, a GNSS and a MCU that fuses all sensors in order to obtain the full navigation information such as absolute positioning, full attitude and speeds [39]. Often-times, these system provide a highly accurate overall pose estimation because the sensors can "correct" each other. For example, the GNSS position and velocity measurement can correct the IMU's integration drift, while the IMU can compute the position, velocity and attitude via integration for short GNSS outages (dead reckoning). The integrated GNSS often use Real Time Kinematics (RTK) positioning, which takes known positions from a highly-accurate base station nearby, obtains phase measurements of the signal's carrier wave and sends the same common corrections to the rover in real-time [40]. Via triangulation between base and a rover up to centimeter-level accurate positions can be computed [41] [42]. The fusion algorithms which are usually based on Extended Kalman Filter (EKF)s will not be discussed further as position and attitude estimation are not the focus of this thesis.

2.5. Vehicle-to-Vehicle Communication

This section mainly serves to introduce some V2V terms that are used throughout this thesis instead of providing an exhaustive overview of existing V2V technologies.

V2V communication generally refers to the wireless exchange of motion-related data among vehicles. The software in these vehicles can then use the newly gained information about other road users to better interact with each other such as triggering AEB or cooperative driving maneuvers like warning other cars ahead about slippery road conditions or a sudden traffic jam. In order for vehicles to actually understand the newly received data from other road users, there needs to exist standardized communication messages. The following subsections briefly introduce two message types used throughout this thesis.

The Cooperative Awareness Message (CAM) is sent from large road users like cars and trucks to all other road users. The structure of the message and parameter interfaces of the CAM are defined in the norm ETSI EN 302 637-2 [43]. For example, the PoseWGS-message defines the positions of the vehicle. Other vehicle sensor data like speed, acceleration, heading etc. are part of this message as well. Moreover, the CAM contains meta data like the weight class and dimensions of the vehicle.

On the other hand, VRU such as pedestrians and cyclist broadcast a Vulnerable Road User Awareness Message (VAM) to other road users. At the time of writing this thesis, the specification of the VAM is not yet uniformly defined. While the SAE J2945/9_201703 standard defines some communication recommendations for VRUs, it is heavily geared towards pedestrians [44]. Therefore, along with this thesis, a VAM is developed with a greater focus on

pedelecs by extending the CAM with pedelec-specific information. However, the actual definition of the VAM is out of scope for this thesis.

In our use case, the car shares its speeds, positions, headings etc. with the pedelec using the CAM. The pedelec only receives and processes the CAM but sends its information to the car by using the VAM.

2.6. Separating Axis Theorem for Collision Detection

Separating Axis Theorem (SAT) is a popular CD algorithm for rigid bodies that finds frequent use in physics engines e.g. of simulations or video games but can also be applied in other domains such as robot motion planning. The main advantage lies in the fact that SAT also handles collisions of convex Oriented Bounding-Boxes (OBB), which are rotated against the world coordinate axes, unlike simpler methods such as Axis Aligned Bounding Box (AABB) where the axes of the objects must be aligned with the axes of the world coordinate system. Generally speaking, the SAT marks a special case of the Separating Hyperplane Theorem (SHT), described in Theorem 2.6.1.

Theorem 2.6.1 „Suppose A and B are nonempty disjoint convex sets, i.e., $A \cap B = \emptyset$. Then there exist a $\mathbf{a} = 0$ and \mathbf{b} such that $\mathbf{a}^T x \leq \mathbf{b}$ for all $x \in A$ and $\mathbf{a}^T x \geq \mathbf{b}$ for all $x \in B$.” [45]

Put in simple terms, the affine function $\mathbf{a}^T x - \mathbf{b}$, called *separating hyperplane*, is nonpositive on C and nonnegative on D and thus separates the two convex sets. The SAT is a method to find whether convex polygons intersect and relies on the fact that convex polygons do not collide if a line can be drawn between them as shown in Figure 2.7.

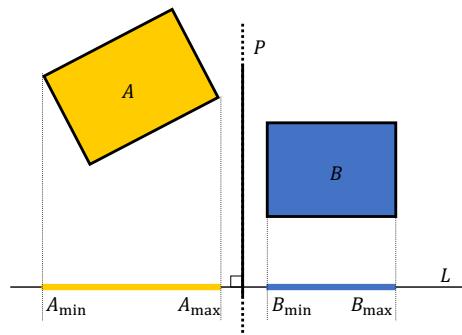


Figure 2.7.: Separating Hyperplane and Separating Axis

This can equivalently be formulated by Theorem 2.6.2, which states the following:

Theorem 2.6.2 „Two convex polygons do not intersect if and only if there exists a line such that the projections of the two polygons onto the line do not intersect.” [46]

2. Fundamentals

The line mentioned in Theorem 2.6.2 is called *separating axis* and denoted with L in Figure 2.7, while P denotes the *separating hyperplane*. The theorem does not work for concave shapes. However, these shapes could be approximated with their convex hull or composed of smaller convex shapes. Since the proofs of these theorems are not necessary to further understand how SAT can be applied in CD, they are omitted.

There exist many ways to apply the SAT in CD, but the algorithms usually consist of four main steps: First, obtain the vertices of the polygons e.g. by transforming from the relative polygon frames to the world frame. Second, compute the normal vectors for each edge of the polygons. They are usually normalized between 0 and 1. Third, project each vertex onto the normal vectors using the dot product and overwrite the constantly overwrite new minimum and maximum projections for each axis. Fourth, check whether the minimum and maximum projections of the polygons intersect for each normal axis. Stop early if an axis is found that has no intersection of the polygon's projections, indicating the that the polygons are not colliding. Figure 2.8 visualizes the projections onto the normal vectors for a non-colliding and colliding scenario.

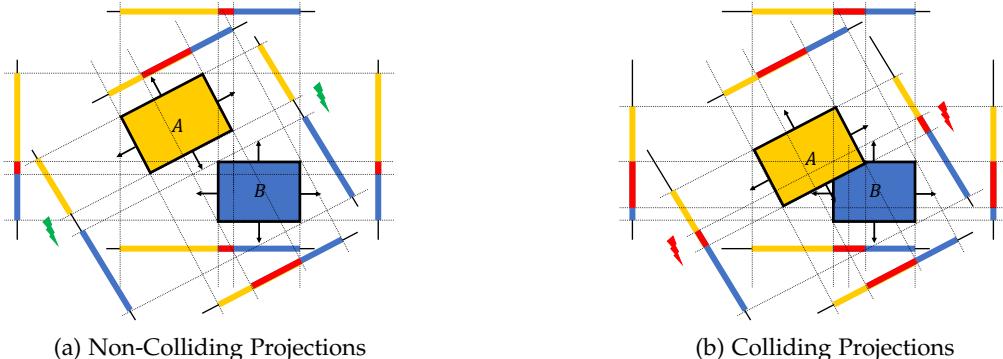


Figure 2.8.: Minimum and Maximum Projections of Edges onto the Normal Vectors

The non-colliding case on the left-hand side has two separating axes, marked with green flash symbols next to it. On contrast, in colliding case on the right-hand side, all axes show intersections of their corresponding minimum and maximum projections.

Note that the SAT computation is computationally quite expensive, especially when it has to be performed multiple times a second. This is especially important to prevent the tunneling effect, where objects seem to pass through each other without collision, simply because the CD is not performed often enough. Generally speaking, nonlinear Continuous Collision Detection (CCD) aims at solving the tunnelling effect by providing highly efficient CD algorithms for moving objects while accounting for rotations [47]. In contrast, linear CCD does not account for rotations. Therefore, CD is often split up into a *broad phase* and *narrow phase* [48]. The *broad phase* uses cheaper CD algorithms when the objects are far apart to determine which objects might be critically close to each other e.g. by approximating them

with a simpler convex hull. There usually exists a trade-off between picking a computationally efficient geometric shape and the tightness of the convex hull around the actual object, i.e. how well the hull approximates the object. The *narrow phase* then computes the exact intersection between nearby objects e.g. via SAT. Furthermore, the SAT itself can be optimized for moving 2D rectangle collisions. Some popular options include the following: The algorithm only needs to test two of the four edges for each rectangle because of its two parallel sides resulting in the same normal axes for projection [49]. Even more efficient, since they are centrally symmetric objects, a comparison of their projected radii from their geometric center to the vertex is sufficient: Two OBBs do not collide if the sum of their projected radii is less than the distance between their projected geometric centers [50] [49]. For moving objects, once a separating axis is found it can be stored and checked first in the next frame. If the objects did not rotate significantly, the separating axis is likely to remain the same and the algorithm can exit early [49] [51].

A penetration vector (or sometimes separation vector) of how deep one shape intersects with the other (or how far apart they are) and the extension of SAT to 3D space are not discussed further as in this thesis it suffices whether moving 2D polygons collide as explained further in subsection 4.9.1.

3. Related Work

This chapter first summarizes previous works conducted at *Robert Bosch GmbH* related to the development of the collision warning system via V2V. The next section then gives a general overview of the related research fields before focusing on existing collision warning systems for bicycles and trajectory prediction of VRUs. Besides, popular models to describe the unique dynamics and stability of bicycles are reviewed. Finally, the last section explains what research field this thesis focuses on and which contributions are made.

3.1. Previous Works

This section shortly summarizes the relevant contributions of each of the previous works in a subsection that leads up to the current state of the V2V-based collision warning system.

Comparing perception of signals in different modalities during the cycling task: A field study

A study by Erdei et al. compared the perception of acoustic, tactile and visual signals in various cycling settings across 52 participants. They showed a low response rate of only 65.1% to visual signals, while tactile and acoustic signals achieved around 87.2% and 97.4%, respectively. The environment, which was selected according to its distraction type (visual, acoustic and haptic), played a negligible role on the response rate of acoustic signals, a noticeable role on visual signals and a crucial role on tactile signals [52].

Entwicklung und prototypischer Aufbau von Fahrerwarnsystemen für Pedelecs

Consequently, Zeradjanin equipped the pedelec with portable speakers and custom-designed vibrating handlebar grips to warn the rider about potential collisions. Hereby, he chose a vibration concept that involves a small electric motor with an imbalance mass and a plug that is inserted inside of the handlebars. Finally, Zeradjanin compared different sound and vibration profiles to obtain the most effective warning signals [53].

Entwicklung einer Fahrerassistenzfunktion für Zweiräder zur Früherkennung nicht sichtbarer Gefahrensituationen mittels 5G-Mobilfunktechnologie

Piscol then assembled the first fully-functional pedelec demonstrator that warns about collisions with other road users in scenarios without line of sight via V2V. The involved

3. Related Work

power supplies, sensors, actuators and processing unit of the test pedelec were already described in section 2.2. Due to the numerous sensors that have to be processed in parallel, a modular software architecture was implemented in Robot Operating System (ROS). The core architecture of the ADAS, depicted in Figure 3.1, abstracts sub-functions into corresponding nodes (represented as boxes) and shows the interactions among them. By reference to Figure 3.1, the next paragraphs briefly explain the overall ADAS function. The linear

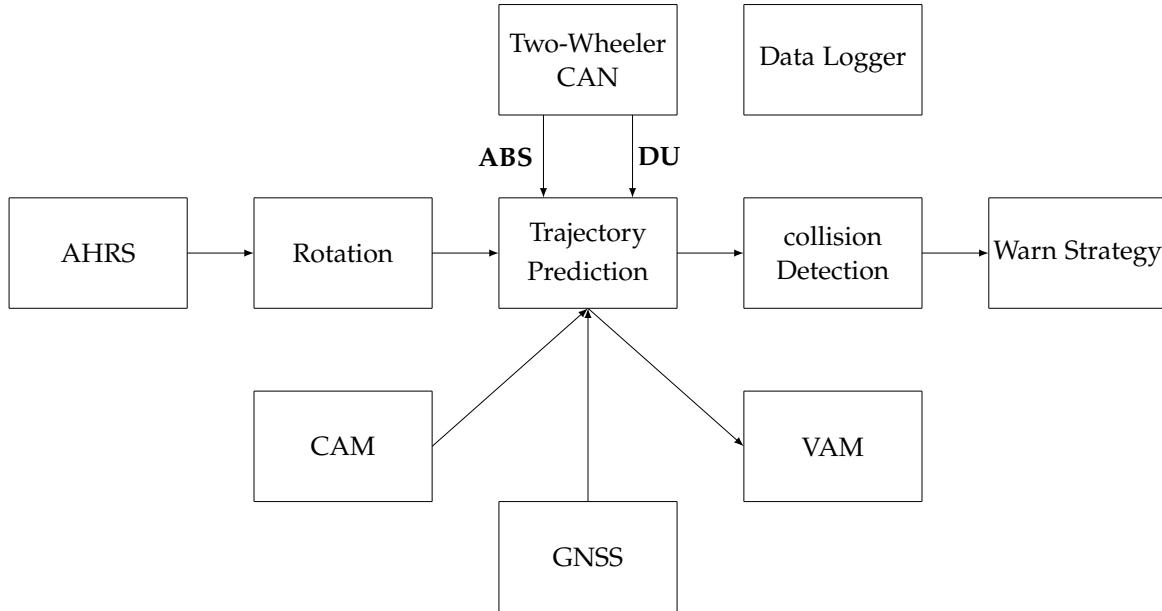


Figure 3.1.: Software Architecture of the Collision Warning. Adapted from [17]

accelerations and angular speeds of the pedelec are continuously read and transformed from the sensor coordinate frame to the bicycle frame and then to the world street coordinate system via rotation matrices by using the attitude solution of the IMU¹. In parallel, further sensor data from the ABS and Drive Unit (DU) systems such as the wheel speed, rider's torque or cadence, motor power output, state of charge, etc. are read from the the two-wheeler CAN. Simultaneously, the GNSS provides the pedelecs absolute position. Besides this naturalistic ego-bicycle motion data, the demonstrator receives data describing the position and motion of larger road users from the CAM via V2V.

All of this information is then processed in the central trajectory prediction block in order to estimate the pedelec's and other road user's future positions. The trajectory prediction is divided as a two step process. First, a node computes the headings of the pedelec by harnessing trigonometric relations between neighboring GPS points and/or integrating the yaw rates of the IMU: Below a threshold speed of $5 \frac{\text{km}}{\text{h}}$, the algorithm only uses the IMU heading while above a threshold speed of $24 \frac{\text{km}}{\text{h}}$ it resorts to only using the GNSS heading. Inbetween these thresholds, the headings consists of a linear combination of the IMU and

¹The coordinate frames were defined in section 2.3

3. Related Work

GNSS headings. Then, another node predicts the pedelec's and other road users' trajectories by using a `const_a_yawr_model` for both participants. This physics-based prediction model assumes a constant acceleration a_x and constant yaw rate ω_z for the entire prediction horizon and integrates these constant values into corresponding position and heading deltas for each time step in the future. These deltas are then recursively added to the initial pose consisting of position x_i, y_i , speed v_i and heading ψ_i which results in a trajectory consisting of many small piece-wise linear segments as defined mathematically in Equation 3.1 and shown in Figure 3.2.

$$x_{i+1} = \sin \psi_i \cdot v_i \cdot dt + x_i \quad (3.1a)$$

$$y_{i+1} = \cos \psi_i \cdot v_i \cdot dt + y_i \quad (3.1b)$$

$$\psi_{i+1} = \omega_z \cdot dt + \psi_i \quad (3.1c)$$

$$v_{i+1} = a_x \cdot dt + v_i \quad (3.1d)$$

Note that v_i represents the longitudinal speed in driving direction at time step i , which implies that the speed perpendicular to the driving direction has to be zero (nonholonomic constraint). Then, the software checks for possible future collisions by computing the L2-

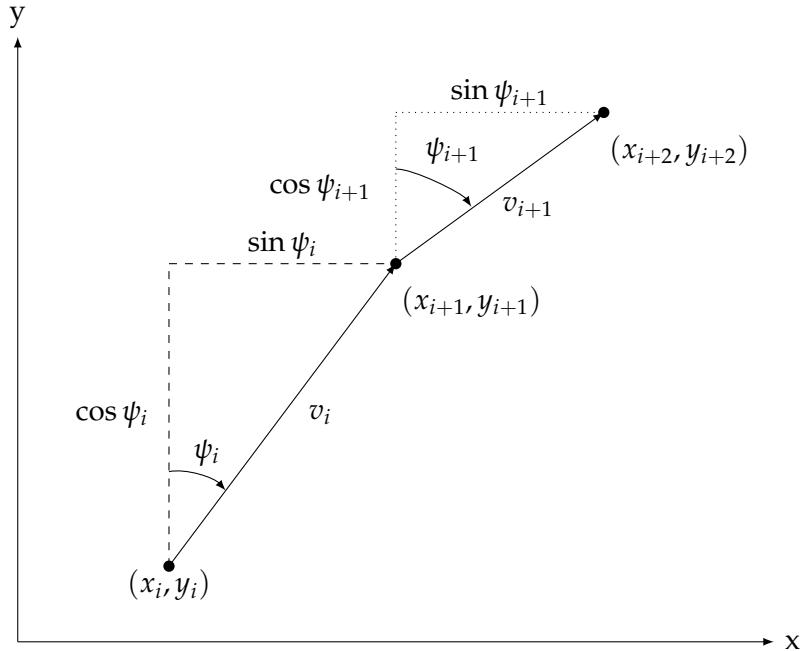


Figure 3.2.: Trajectory Computed from Piece-Wise Linear Segments

distances between the pedelec's and road user's position at every point in time as shown in Figure 3.3. A collision is detected if the distance between two corresponding positions in time is less than 6 m as visualized with red circles at time step t=2 of Figure 3.3. This threshold distance represents the virtual intersection of the two vehicles' critical collision zones. The critical collision zones are defined by their maximum radii around their GPS

3. Related Work

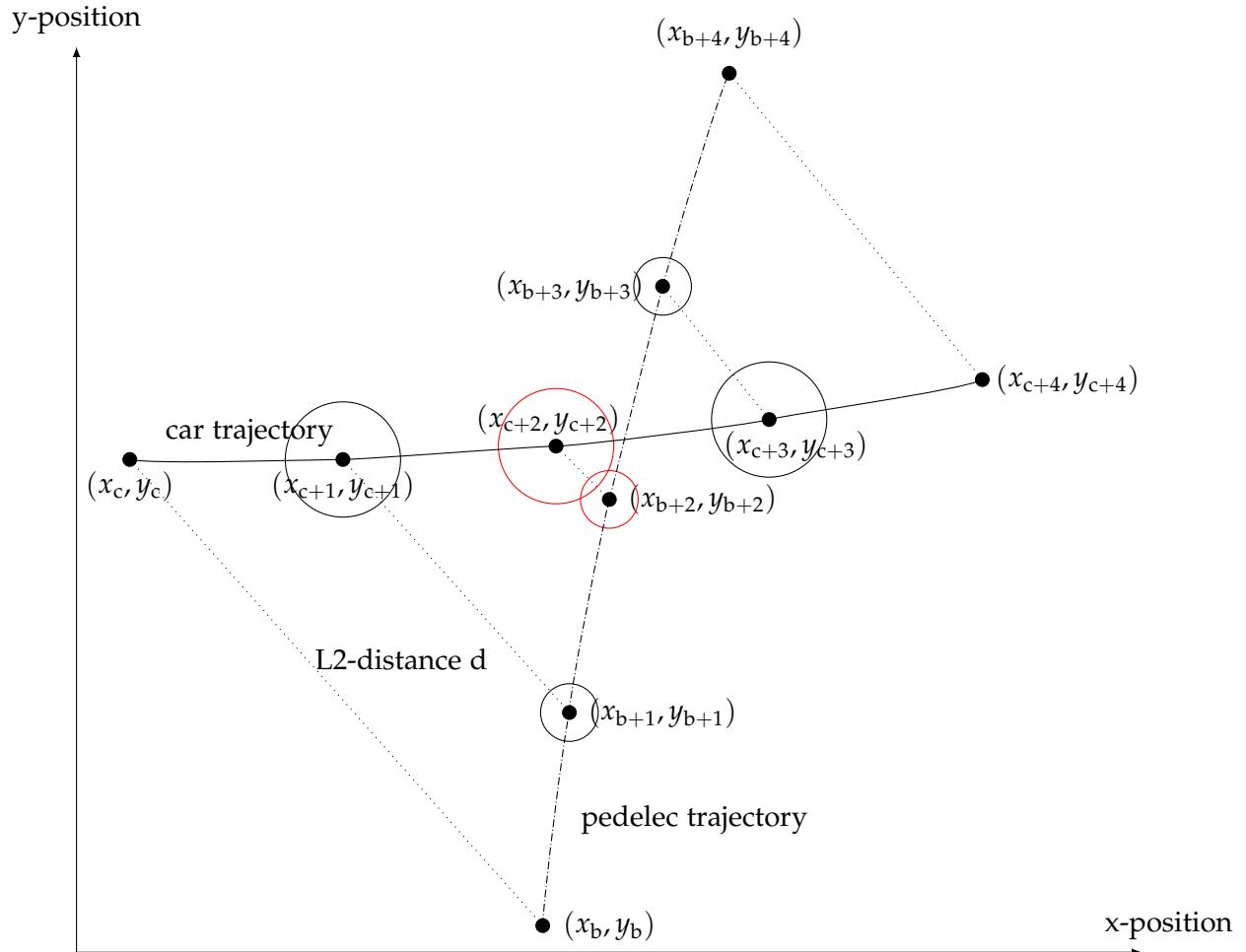


Figure 3.3.: Distance between Corresponding Trajectory Points and CD Radius. Adapted from [17]

3. Related Work

Antennas. Detected collisions are then outputted in the warn strategy module that plays the sounds and vibration profiles as determined by [53] to warn the rider of an incoming collision. Conceptually, the pedelec would send its position, motion and even trajectory data to other V2V participants using the Vulnerable Road User Awareness Message (VAM) [17] to trigger an appropriate reaction of other road users (e.g. an automated emergency brake of a car). At the time of the hand-in of this thesis, the pedelec could only receive and process information from other road users but not share its own data to others yet.

Multivariate Forecasting of E-Bike Sensor Data using Machine Learning

Kailasam investigated whether Machine Learning (ML) techniques can improve the trajectory prediction of the pedelec by forecasting its speed and yaw rate signals rather than GPS positions which jumped and drifted off of the presumable ground truth path. He implemented, trained and compared Random Forest Regressor (RFR), 1D Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models for both single point and multi-step predictions against a linear extrapolation baseline. The goal of the single point prediction is to predict only the final point of a given signal at the end of the prediction horizon, while the multi-step prediction forecasts consecutive points of the signal within the prediction horizon. Predicting only the final point of the speed and yaw rate signal does not provide enough information about the shape of the signals to be integrated to obtain future positions. Therefore, only the relevant multi-point approaches (CNN and LSTM) will be shortly summarized in the following.

The data pre-processing for the time series forecasting models works as follows: First, the signals are cut into small equally-sized input samples using a sliding-window approach with a configurable stride. Kailasam uses a rather large striding length, producing fewer samples, which results in speed benefits during hyper-parameter tuning. He claims that down sampling preserves the underlying pattern and structure of the signals better than deleting samples [54]. The first part of each sample represents the inputs and the last part as the ground truth outputs compared against the model's predictions. Kailasam notes that straight cruising sections (yaw rates near zero and roughly constant speed) heavily outweigh other samples during a trip, and the imbalance could lead to undesired bias in the model's predictions towards straights at constant speed [54]. Therefore, the data is balanced by first binning all samples based on headings deltas from integrated yaw rates within the prediction horizon and speed deltas between the first and last point of the prediction horizon. Then, only a certain number of randomly drawn samples from the biggest bin are kept while removing the others. The number of kept samples equals the mean value of samples present in the remaining four bins. Furthermore, the data samples are normalized to speed ranges from $0 \frac{\text{km}}{\text{h}}$ to $40 \frac{\text{km}}{\text{h}}$ and acceleration ranges from -4 g to $+4 \text{ g}$. Finally, the samples are stored as separate training, validation and test data sets using a 70/15/14 split for supervised learning. Note that Kailasam also constructed a so-called "Overview Data Frame" used to track which samples originates from what measurements to simplify offline analysis of cases with bad predictions.

The overall architecture of the multi-step approach is shown in Figure 3.4, which only differs

3. Related Work

in type of networks used to extract features from the signals (either CNN or LSTM). The

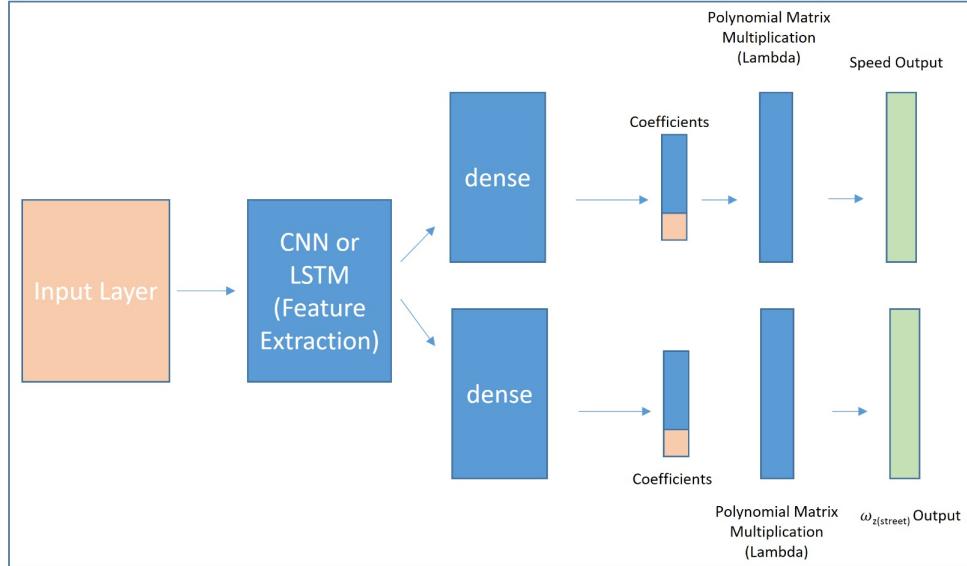


Figure 3.4.: Simplified Architecture of the ML-Based Multi-Step Prediction Models [54]

CNN feature extraction model follows a generic structure consisting of three sets of CNN layers, each consisting of a convolution and max pooling layer, that end in a flattened layer which is split into two separate dense layers. The LSTM feature extraction model consists of a single LSTM layer, followed by a dropout layers. Note that after the shared feature extraction, the speed and yaw rate signals are split into two separate heads using two separate fully-connected layers in both models. To maintain continuity and achieve smoothness for the predicted signals, which are more in line with their underlying physics, the models do not predict succeeding signal points directly. Instead, each head predicts the coefficients of a polynomial function with a predetermined degree from which the future points are obtained. For multi-step prediction of up to 20 future points (equivalent to 4 s), Kailasam concludes that all of his trained ML models outperform the linear extrapolation baseline with the LSTM achieving the best results this time [54]. He also claims that the LSTM model has the ability to predict changes of trend in the signals. Kailasm leaves the extension of the RFR for multi-step prediction and optimization of all models to run on embedded systems as future work.

Entwicklung eines Kollisionswarnsystems basierend auf Fahrzeug zu FahrzeugKommunikation und Optimierung der Performance durch Machine Learning

Muresan improved the ADAS function by Piscol in the areas of heading estimation, trajectory prediction and warning system. He also enabled sharing the pedelec's data to other road users. In addition to the core ADAS function architecture as depicted in Figure 3.1, Muresan

3. Related Work

expanded the overall software framework with some supporting nodes and visualization scrip to help analyze and evaluate the performance of the ADAS. Muresan also split the trajectory prediction block as shown in Figure 3.1 into two nodes: A *Heading* node that estimates the current heading (and positions) of the pedelec and a *trajectory prediction node* that uses this initial state for predicting the actual future positions [55]. Muresan compared the headings from the linear combination of IMU and GNSS algorithm with purely GPS-based headings and headings estimated from a one-dimensional (1D) EKF and a 3D EKF. Muresan concluded that the 3D EKF resulted generally in the most accurate and most stable headings, especially during phases of bad GPS reception. This 3D EKF essentially uses Equation 3.1 from [17] to fuse the GNSS, IMU and Wheel Speed Sensor (WSS) sensors. Muresan then compared Piscol's linear `const_a_yawr_model` trajectory prediction model with two of Kailasam's nonlinear ML-based prediction models (LeNet CNN and LSTM) and newly developed hybrid models that use the linear model for computing either the lateral or longitudinal component and the LSTM model for the remaining component². For example, an LSTM estimates the longitudinal speed components while a linear model estimating the lateral yaw rate component. In addition, Muresan increased the capacity of the LSTM model by doubling the number of cells in the hidden layers and experimented with a few different loss functions. An inference time analysis showed that the linear model clearly outperforms the hybrid model and LSTM-based models by a factor of more than 97 times. Muresan notes that the inference times of the full LSTM model for a short prediction horizon of 0.5 s already clearly violate the real-time capability of the system running at 20 Hz. Therefore, he does not include the full LSTM model in further comparisons. However, the large standard deviation in inference time leads to some missing trajectories even for the faster hybrid models as a result of not finishing the computations in time. Interestingly, these missing trajectories seem to occur at the same locations, independent of whether the longitudinal or lateral component is predicted using the LSTM. Moreover, Muresan defines a range of metrics $\mu_{\Delta x}$, $\sigma_{\Delta x}$, $\mu_{\Delta y}$, $\sigma_{\Delta y}$, $\mu_{\Delta l}$, $\sigma_{\Delta l}$, $\mu_{\Delta \sigma}$ and $\sigma_{\Delta \sigma}$ to quantify the accuracy and precision of the trajectory prediction models. These metrics are obtained by computing the mean and standard deviation over all absolute errors in x-direction Δx , y-direction Δy , length Δl and angle $\Delta \sigma$ of the predicted endpoints compared to the actual path points at the end of the prediction horizon for every time step of a ride. These absolute errors are always defined as $pred - real$. Figure 3.5 visualizes these absolute errors for a single time step, where the red circle represents the initial point, the blue the predicted trajectory endpoint and the green point the actual path point at the end of the prediction horizon.

²Muresan refers to the `const_a_yawr_model` simply as *linear* trajectory prediction model without any explanation as to what the linear part refers to.

3. Related Work

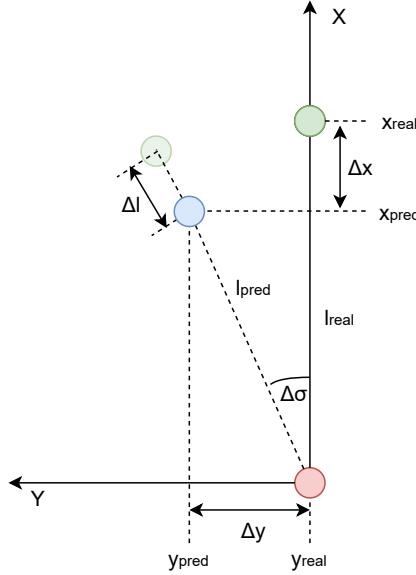


Figure 3.5.: Trajectory Prediction Error Metrics. Adapted from [55]

In this figure, Δx , Δy , Δl and $\Delta \sigma$ are all negative. Such a statistical evaluation over the absolute errors showed that the `const_a_yawr_model` predicted most accurate trajectories for one second while the hybrid models outperformed this simple model for a prediction horizon of two seconds. The comparatively large validation loss and small trajectory accuracy and precision indicate that the LSTM struggles to predict future yaw rate values. Furthermore, Muresan states that the predictions of the yaw rate sometimes even noticeable influence the errors in the longitudinal direction due to 'rolling' effects of the resulting trajectories. Muresan hypothesizes that inappropriately high thresholds in the data balancing step accidentally discard the majority of curve samples from training because they mistakenly end up in the heavily over-represented bin for straight sections. Muresan also concludes that the weighted mean average error loss function did not increase the performance noticeably compared to the default mean square error loss. To improve the CD, Muresan replaced the static 6 m static range indicating the collision-warning-zone with a dynamic range, shown in Figure 3.6, which uses cosine and sine transformations of the heading difference between the two vehicles to construct a more sophisticated warning buffer. This solution accounts the shapes and headings of the vehicles better, given that the GPS antennas might not be mounted at the geometric centers of the vehicles.

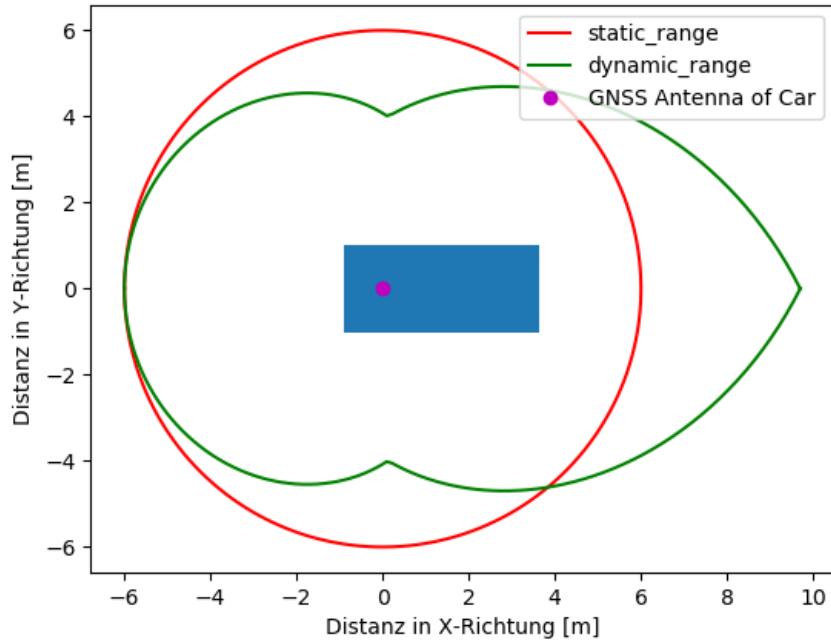


Figure 3.6.: Comparison of *static_range*- and *dynamic_range*-Methods from an Aerial Perspective [55]

Besides computing the Time-To-Collision (TTC), the crash direction is computed and displayed to the rider adapted for the peripheral vision of humans. For example, a collision that is predicted to occur at an angle of more than +110 or -110 degrees from the current heading is display as 'behind'. Also, the warning strategy was slightly improved by oppressing the acoustic warning if the pedelec stays within the vicinity of the car. A counter in seconds simply defines how long the warning signal should sleep after the first detected collision. Finally, Muresan integrated the VAM message which defines the type and frequency of data to be sent by the pedelec to other road users and thus enabled bidirectional V2V communication [55].

3.2. Related Research

This section first provides an overview of potentially relevant research fields to the problems of this thesis. Then, it summarizes research attempts to develop a collision warning system specifically for bicycles. Afterwards, some popular models to describe the special dynamics and stability of bicycles are introduced. Then, it describes research dealing specifically with trajectory prediction for bicycles as this is one of the main focuses of this thesis. Finally, gaps in current research are outlined and the targeted research is clarified.

3.2.1. Overview of Relevant Research Fields

As depicted in Figure 3.1, the to-be-optimized collision warning system consists of numerous components that combine multiple technologies, which means many research fields could potentially be relevant. Obviously, V2V communication is a research field of its own concerning itself with the full communication stack from wireless technology to protocol efficiency and networking. Achieving accurate and reliable positioning in urban areas using GNSS is another research area of its own. Similarly, fusing IMU sensors to estimate the full attitude in dynamic environments is quite encompassing as well. Another research community investigates modelling the unique and complex dynamics and stability phenomena of bicycles, which may be useful to incorporate into the trajectory prediction models. Concerning the trajectory prediction itself, there exist a large number of research communities trying to improve the trajectories of all kinds of road users from highly automated or autonomous cars to VRUs such as motorcycles and pedestrians. Some may concern themselves with short-term prediction horizons of a single vehicle e.g. predicting a curve in rural conditions, whereas others try to model long-term interactions of many participants in dense urban highways or intersections. Generally, there exist many prediction approaches that may draw knowledge from more general research fields like time series prediction using deep learning. Finally, detecting collisions can have relevant overlap with research fields such as robot motion planning or physics engines used in games or simulations. One such classical algorithm from the latter domain was introduced in section 2.6.

3.2.2. Collision Warning Systems for Bicycles

Even though CD, warning and avoidance are well-established in automotive research, only little research is available regarding approaches that specifically warn cyclists about possible collisions with other road users. This subsection summarizes a few of those.

Van Brummelen et al. propose a reliable and low-cost cyclist collision warning system that consists of a single-beam Lidar, two ultrasonic sensors, a MCU and a taillight mounted to the rear of the bicycle. Additionally, two eccentric mass vibrators are installed in the handlebars. The system detects rear-on collisions by abstracting a light-weight mathematical model of a rule-based Fuzzy Inference System (FIS) or alternatively lookup tables for deployment on the MCU. The FIS receives the distance and velocity information as inputs and outputs a Pulse-Width Modulation (PWM) signal to provide haptic feedback to the rider via the vibrators and warns oncoming vehicles using the taillight. Hence, the Lidar makes up the main sensor while the ultrasonic sensors are mostly used for distinguishing whether a vehicle approaches from left or right and detect close range blind spots which fall out of the Lidar's range. The authors merely conduct a perception study regarding the levels of the haptic feedback but do not validate the collision warning system itself [56].

Similar rear-end collision warning systems for bicycles based on ultrasonic sensors to reduce blind spot accidents were developed by Li et al. [57] and Gooi [58].

3. Related Work

Iwamoto and Otake propose a warning system that computes time deltas between road-side distance sensors to avoid collisions among cyclists and pedestrians at intersections [59]. They only tested the perpendicular crossing scenario where only one road is equipped with sensor while the other road is not: The subject approaches the intersection on the normal road and gets signalized by a flashing light near the intersection about a possible collision with the other cyclist that passes the distance sensors several meters before the intersection. They also successfully conducted experiments at night. Similar to [52], they found that visual warnings are not sufficient and need to be aided with additional sounds.

Jeon and Rajamani developed a collision avoidance system for bicycles aimed to prevent two common crash scenarios between cars and bicycles at intersections: rear and side crashes (blind spot). They custom developed low-cost sonar and laser sensors alongside with their estimation algorithms. The sonar is used to estimate the distance as well as the angular orientation of vehicles to the side of the bicycle. The single-target rotating laser tracks vehicles in longer distances. A clustering technique, based on Density Based Spatial Clustering of Application with Noise, first detects the approaching target from behind with the rotating laser. Then, the target gets tracked by using a Kalman Filter (KF) that estimates the relative longitudinal distance between them which is used to control the laser sensor orientation to follow the target via Model Predictive Control. The authors only presented simulation results for the rear tracking and did not evaluate the collision avoidance as a whole [60].

In follow-up publications, the target tracking was extended to the two-dimensional vehicle motion, and the rear collision avoidance system was evaluated on real world roads. A receding horizon controller is used for the active sensor control and an Interacting Multiple Model (IMM) framework for the vehicle motion estimation [61] [62]. The system thus still does not predict the trajectory of the bicycle but rather the trajectory of the cars relative to the bicycle by using the same planar motion primitive as explained in section 3.1. They showed that the custom-developed active sensing system can successfully track vehicles of different types and sizes to the side and rear of the bicycle also in situations involving a series of passing cars.

As of June 2021, the bicycle was equipped with an additional low-density Lidar and a side laser sensor at the front as well as two speakers facing to the front and rear. These additions enable the collision avoidance system to handle two more use-cases: left-turn and sideswipe collisions. Since all of the sensors, speakers and processors required for the collision avoidance system are on-bicycle, the bicycle essentially protects itself. As it does not rely on any additional technology on cars such as V2V communication or object detection from windshield cameras, this system could be deployed immediately on today's roads. However, the idea is to alert other road users of the bike's presence instead of only alerting the bicycle rider. The warning strategy is purely based on a TTC of about 2 s. The conducted field operational test showed a large number of FP either due to the bicycles rolling and turning motion, stationary objects like a sequence of trees near roads or parked cars or slightly turning to maneuver around parked parks [63]. The false alarms at high bicycle dynamics are "solved" by using an additional IMU to estimate the bicycles attitude in conjunction with a magnet and reed

3. Related Work

switch to estimate wheel speed. Essentially, if the roll angle or yaw rate surpass a certain threshold the alarms are discarded. Using the measured speed of the bicycle, the target's speed can be estimated and used suppress alarms if the target is doomed stationary.

3.2.3. Modeling the Dynamics and Stability of Single Track Two-Wheeled Vehicles

Literature concerning about the dynamics of bicycle models³ are focused mostly around controlling bicycles (or motorcycles) by simulating rider inputs such as pedalling or steering torques (e.g. autonomous riderless bicycle robot) or stability analysis. The subsection introduces two popular bicycle models of vastly different complexities and overviews their underlying assumptions and what they model. It finishes with some research about bicycle stability that makes use of these models.

3.2.3.1. Getz and Zhang Models

The first type of bicycle model is based on Getz, which models the rider and bike as a point mass m at their combined Center of Gravity (CoG) between the contact points of the wheels W_R and W_F at a vertical height h [65]. This means, the wheels are not modelled as bodies of a multi-body system and only serve as a nonholonomic constraint given by the wheel-ground interaction: A contact point between wheel and ground can only move along their velocity vector v_x , opposed to perpendicular (no lateral slip). Furthermore, the normal vectors of the moving wheel-ground contact points W_R and W_F intersect at the instant center of rotation as long as the roll movement is not considered. Figure 3.7 depicts a sketch of the Getz model along with its parameters.

³Bicycle models refers to dynamics models for bicycles that include the roll angle, opposed to the single-track vehicle dynamics model by [64] mostly used to simplify car dynamics.

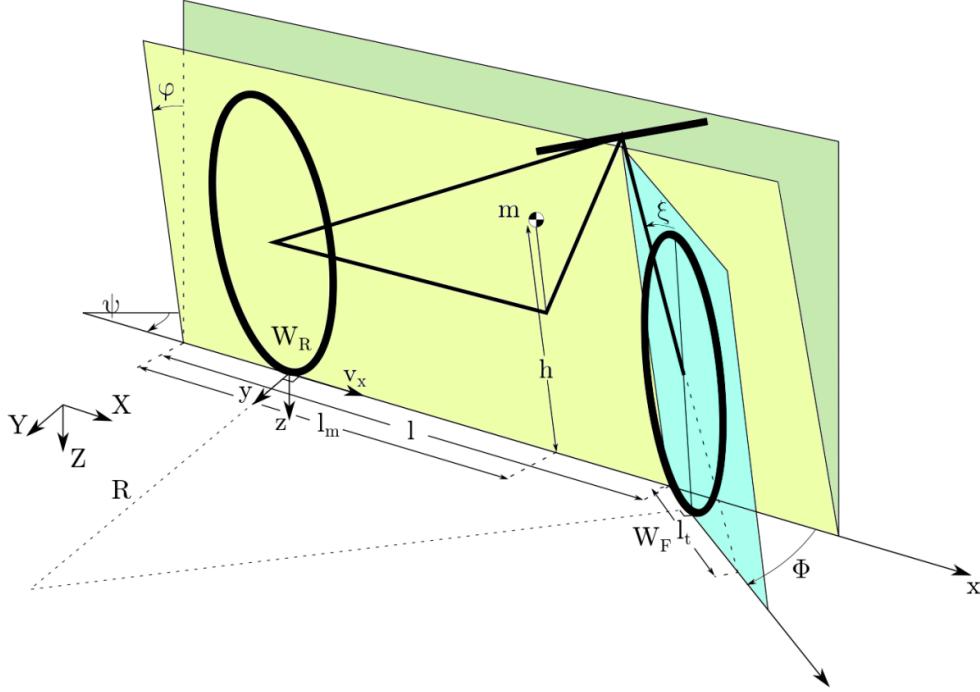


Figure 3.7.: Sketch With Parameters of the Mathematical Getz and Zhang Bicycle Model [66]

Note that the {Street} and {Bicycle} frames in Figure 3.7 are rotated by 180 degrees compared to the definitions of this thesis from subsection 2.3.4 and subsection 2.3.5, and the roll angle is denoted with φ opposed to ϕ . Getz derives the following differential equation for the kinematics of his model (see Equation 3.2) [65]:

$$h\ddot{\phi} = g \sin \phi + ((1 + h\psi \frac{\sin \phi}{v_x}) v_x \dot{\psi} + l_m \ddot{\psi}) \cos \phi \quad (3.2)$$

As explained in subsection 2.3.5, ψ denotes the heading of the bicycle relative to the {Street} frame. The variable l_m defines the distance between W_R and the CoG.

Zhang extends the Getz model with a caster angle η which lays between the axis of the steering joint and and the z-axis of the {Bicycle} and resulting trail l_t as shown in Figure 3.7 [67]. The kinematic equations of the Zhang model are captured in Equation 3.4:

$$\frac{l_m h \sigma}{l} \dot{v}_x \cos \phi + h \dot{v}_y \cos \phi + h^2 \ddot{\phi} + \left(1 - \frac{h \sigma}{l} \sin \phi\right) + \frac{h \sigma \cos \phi}{l} v_x^2 \quad (3.3)$$

$$-g(h \sin \phi + \frac{l_t l_m \cos \eta}{l} \sigma \cos \phi) = -\frac{l_m h}{l} v_x \dot{\sigma} \cos \phi \quad (3.4)$$

Hereby, l denotes the distance between the wheel-ground contact points W_R and W_F as depicted in Figure 3.7 and $\sigma = l/R$ the kinematic steering variable, where R is the instantaneous turning radius related to the rear wheel. Oftentimes, the slide slip velocity $v_y = 0$ similar to the Getz model, which cancels out the second term of Equation 3.4.

3.2.3.2. Whipple-Cavallo

In contrast to the previous bicycle models, the Whipple-Cavallo model is much more sophisticated and consists of four hinged rigid bodies, namely the front wheel F , the front bicycle frame H consisting of a fork and handlebar assembly, the rear frame B with a rigidly attached rider and the rear wheel R [68] [69]. The axis-symmetric wheels represents ideal knife-edge rolling points without any lateral or longitudinal slip. In total, this model is described by 25 parameters (e.g. trail, steer axis tilt, mass moments of inertia for each body etc.) that represent a seven-dimensional configuration space, which is reduced to three velocity DoF because of its non-holonomic rolling constraints. The three velocities are the roll rate of the rear frame B , the steering rate and the rotation rate of the rear wheel R relative to the rear frame B . A popular set of linearized differential motion equations in canonical form for this Whipple-Cavallo model are derived in [70].

Motorcycle models for example by Cossalter are even more complex, often additionally using sophisticated suspensions and tire models and up to 11 DoF [71].

3.2.3.3. Bicycle Stability

There exist a fair amount of papers on the stability of bicycles starting with the analysis by Whipple in 1899 [68]. For this thesis, findings about the speed ranges of stability for straight ride driving are highlighted. Note that there exist many other stability analysis e.g. for steady state turning.

According to [70] [72], which used the linearized Whipple-Cavallo Model introduced in subsubsection 3.2.3.2, the self-stabilizing window for an unmanned bicycle lies between the weave and capsize speeds which vary on the geometry of the bicycle as well as the mass and riding position of the cyclist. For most practical applications, we could claim that the bicycles becomes easily balanced roughly at speeds greater than $2 \frac{\text{m}}{\text{s}}$ due to the gyroscopic effect or the caster trail of the front wheel or an appropriate mass distribution⁴. Below its geometry specific *weave speed*, a riderless bicycle is unstable and can easily fall over because of the sinus like wiggling of the front wheel about the headed direction. Similarly, a moving bicycle above its *capsize speed* becomes mildly unstable again. In this case, an unstable bicycle would progressively move into a tightening spiral with both lean and steer angle increasing as the bicycle falls over.

While all of the bicycle models make a fair amount of assumptions on the bicycle geometry, they make even more assumptions on the rider and his/her impact on the stability by moving the CoG of the system e.g. via small displacements, tilts and rotations in the hip or dipping the knees or shoulders into turns. The impact on stability of these rider-specific effects are still open research questions.

⁴They found that the gyroscopic and caster effect are not the only parameters that determine stability as previously thought. They showed that a bicycle with a correct mass distribution but with counter-rotating small wheels, that cancel out the gyroscopic effect, and with negative trail was able to self-stabilize at certain speeds [73].

3.2.4. Trajectory Prediction for Vulnerable Road Users

Most literature about predicting trajectories for road vehicles focuses on cars rather than VRU⁵. These approaches are not discussed further and rather focused on trajectory prediction for VRUs, which can generally be grouped into three main approaches: physics-based, statistical models and ML-based. The following sub-subsections summarize previous attempts for each of these approaches.

3.2.4.1. Physics-based Approaches

Yuan et al. investigate the trajectory planning and tracking control by using the Getz's Model, as introduced in subsubsection 3.2.3.1, for the purpose of self-balancing a autonomous bicycle robot [74]. Rather than predicting the intended path of a cyclist, they plan an optimal trajectory to move from an initial pose to a pre-defined end pose (position and heading) such that the maximum roll angle at a rotational equilibrium is minimized, and then focuses on designing a controller to follow this ideal trajectory while keeping the bike balanced. The underlying kinematics remain modeled by a planar motion with a nonholonomic constraint similar to Equation 3.1 with the exception that the yaw rates are computed from the speed and curvature of the path. The roll dynamics of the Getz Model, that also contain the speed, roll angle and curvature, are used to constrain the min max optimization problem to a rotational equilibrium.

Zernetsch et al. use a physical model for only predicting the longitudinal component of a cyclist's starting motion (speed over time). The model is based on an equilibrium of the resisting forces (acceleration, inclination, rolling and air resistance) and driving force and uses polynomial fitting to obtain its two parameters (acceleration and steady state speed) from recorded data [75].

A paper by [66] proposes a method to predict a reachability set of Single-Track Two-Wheelers (1T2W) by simulating various combination of a parameterized dynamics model of a two-wheeler. Essentially, they predict the spanned area in which a 1T2W can physically end up for a given prediction horizon under worst-case speed and roll angle assumptions. These assumptions are pre-defined speed and roll angle profiles. The speed profile consists either of a constant deceleration phase limited by the maximum deceleration rate or an acceleration phase limited by a maximum acceleration and target steady-state-speed. The roll angle profile consists of a rising section, a constant section and another falling section. The rising and falling sections are modelled using shifted cosines and its slopes are limited by the magnitude of the maximum roll rate. The constant section is a line with a slope of zero at the magnitude of the maximum roll angle. The reachability set is computed by inputting these time dependent speed and roll angle profiles (alongside the constant bicycle geometry) into the dynamics equation of the Zhang Model, as explained in subsubsection 3.2.3.1, where the

⁵Trajectory prediction for air-crafts and vessels also exists.

kinematic steer angle σ is replaced by a yaw rate $\dot{\theta}$ relation described in Equation 3.5.

$$\sigma = l \cdot \dot{\theta} \cdot \frac{1}{v_x} \quad (3.5)$$

In a real world validation, the spatial reachability sets held for most of the extreme maneuvers, but the authors recommend an additional safety margin. The paper claims that "this is the first work known to the authors which makes use of the unique kinematics of a 1T2W" [66].

3.2.4.2. Probabilistic Approaches

Similarly to [75], Luo et al. predict only longitudinal acceleration/deceleration profiles using naturalistic data for the purpose of a traffic simulation. However, they make use of a Most Likelihood estimator to guess the parameters of a polynomial curve [76] [77].

Pool et al. compare three probabilistic motion models for cyclist path prediction based on linear dynamics: Linear Dynamical System (LDS), Uninformed Mixture of Linear Dynamical Systems (U-MoLDS) and Informed Mixture of Linear Dynamical Systems (I-MoLDS) [78]. While the source of data in this paper are images from vehicle dash cameras, the ground truth paths of the bicycles were reconstructed carefully and transformed into the ego perspective of the bicycles. The LDS model consists of a constant velocity model with white noise acceleration (Gaussian). In this case, a recursive Bayesian filter is just a common KF, where the predictive distribution is Gaussian as well. The U-MoLDS model combines five LDS with specific underlying dynamics corresponding to each of the five directions that occur in the road topology according to the hidden direction intent of the cyclist. Because the intent is not observed, its distribution needs to be estimated online. Selecting a uniform prior distribution over all directions makes the model uninformed about the hidden intent. The model then predicts the prior distributions of both the continuous motion state and hidden direction intent based on past observations. The I-MoLDS model works similar to the U-MoLDS but incorporates useful prior information about the frequency of directions in the track's topological layout (e.g. from a map). For example, if a certain road direction is not present in the track, its intent prior is set to zero. The parameters of the initial state, processes noise and observations noise distributions are learned offline from a training set from a fully Bayesian approximate inference with conjugate prior distributions. To summarize, they showed by exploiting prior knowledge about the road topology, the trajectory prediction can be improved.

3.2.4.3. Machine Learning-based Approaches

Machine or Deep Learning-based approaches for trajectory prediction are most heavily researched in long-term behavioral predictions of cars, oftentimes modeling the interaction among multiple vehicles via LSTMs and attention mechanism (e.g. lane switching on highways). Intention and trajectory prediction of pedestrians from the third perspective of a car is also a fairly well researched topic, while the amount of research geared towards bicycles is

3. Related Work

comparatively small. The following paragraphs focus on the ML-based trajectory prediction specifically for bicycles.

In [75], Zernetsch et al. also attempted to predict a cyclist's future positions from past positions by training a Multilayer Perceptron (MLP). Similarly to Kailasam's polynomial approach as summarized in section 3.1, the network actually predicts parameters of a polynomial. However, Zernetsch et al. also input polynomials to the network, which differs significantly in its architecture and apply the prediction to the positions, whereas Kailasam predicts the speed and yaw rate signals and passes them into a motion model, where they are integrated into positions.

Saleh et al. use two stacked Bi-LSTMs to predict the future positions (latitude, longitudes) of bicycles directly from previous positions without any underlying vehicle model [79]. These ML-based models clearly outperform the MLP polynomial approach from the previous paragraph as well as the three probabilistic LDS models, described in subsubsection 3.2.4.2 by more than 50% in mean error over all predicted trajectories of the cyclists for a prediction horizons of one second⁶.

Gao et al. predict cyclist trajectories at unsignalized intersections by using a Bayesian network that predicts an intention probability which is fed into an encoder-decoder LSTM architecture that incorporates the road environment (e.g. road lines) as well besides the ego vehicle's and bicycle's position, speed and direction [80]. Their models achieves an average position error (euclidean distance) of 1.11 m, 0.91 m and 0.80 m for going straight, turning left and turning right, respectively.

Cycling-Net by Ding et al. predicts cyclist behaviors from geo-referenced egocentric video data using a parallel LSTM architecture, where one string is fed from a Mask-RCNN object detector and the other one with the heading and distances from the GPS. Instead of images from a third-person observer, they use images from the point of view of the bike and supplement data from a GPS receiver [81]. However, the predicted behavior classifications are not used to compute future trajectories but rather to understand their road behaviors.

3.3. Thesis Research Focus

As explained in section 3.1, two major components influence the accuracy of the trajectory prediction: the state estimation of position, heading, etc. and the actual prediction of future positions. However, an accurate state estimation using low-cost sensors in urban areas will be a thesis of its own. While the CD and warning strategy is also improved, it is not a research focus of this thesis.

Instead, this research focuses on the *physics-based* trajectory prediction of a *single cyclist*

⁶The same image based data set and processing as in [78] is used.

3. Related Work

at the *operational physical layer* using *naturalistic* data only that harnesses the *unique dynamics* of bicycles.

Investigating complex and long-term interactions beyond more than two road users (car and pedelec) is not a research goal.

The operational physical layer, as grouped by [82], focuses on short-term time horizons of only a few seconds, and the rider's pedalling and steering decisions within a given route are relevant opposed to incorporating high-level tasks such as path choices, activity scheduling, route choices, departure times, etc.

There do not seem to exist any promising approaches that predict cyclist trajectories only from its naturalistic ego-data in a V2V context. Naturalistic data refers to e.g. IMU, WSS and GNSS measurements collected from the ego vehicle. The trajectory prediction of the cyclist does not rely on any external data from a third-person perspective e.g. through expensive Lidar, Radar or image sensors.

There does not yet seem to exist a research attempt that predicts distinct cyclist trajectories by making use of the unique physical characteristics of bicycles that can be used effectively to warn about potential collisions⁷. This thesis aims to fill this research gap by investigating ways to harness e.g. the roll angle.

Due to the limited research effort so far in some of these domains, this thesis conducts basic research with the aim of gaining a deeper understanding of the challenges in designing a collision warning system for pedelecs rather than blindly applying and adapting state-of-the-art ML approaches to a new domain.

⁷Zernetsch et al. [75] only predict the longitudinal speed, which makes this approach useless for turning scenarios. Wirth et al. [66] generate a fixed worst-case reachability set without a probability surface to indicate any sense of direction.

4. Solution Approach

This chapter covers the solution approach to improving the collision warning system. The first section briefly analyzes the initial state of the components that may have an effect on the FP rate of the collision warnings or evaluation thereof. The following section provides a general overview of the improved software framework and explains the interactions of its components. After that, all succeeding sections describe the contributions made over the course of this thesis more detailed, split into a section for each major component. One section describes the empowerment of the software to conduct and track pre-configured scientific experiments in an automated fashion. Then, a section quickly covers some crucial settings for evaluating the error statistics as well as adaption of the visualization scripts used for further analysis. Then, a section describes the use of a virtual car for an independent end-to-end validation of the improved collision warning system as a whole. After that, a section covers the integration of new reference sensors for the positioning and attitude estimation into the existing hard- and software of the test vehicle (pedelec) in order to facilitate the real-world validation of the ADAS function. The following section introduces a way of potentially decreasing the influence of the state estimation errors on the trajectory prediction errors. Then, a section derives a transformation for virtually offsetting the pedelec's position and correcting for the projection error due to high roll angles. Towards the end, a large section covers the development of novel physics-based trajectory prediction models to reduce the prediction errors. Subsequently, a section introduces a new approach to detecting collisions and warning appropriately to reduce the number of FP. Finally, minor changes in the VAM are summarized.

4.1. Analysis of Initial State

This section investigates the initial state of components that may have an effect on the FP rate of the collision warnings or evaluation thereof. After a general analysis of the overall software framework, the sensor signal quality, current algorithms for estimating the pedelec's initial state (heading and position), the statistical evaluation of trajectory errors, the motion primitive and trajectory prediction models as explained in section 3.1 are analyzed in more detail. Finally, the last section dissects the CD and warning strategy for possible false detections.

4.1.1. Overall Software Framework

As mentioned in section 3.1, the core ADAS function of the pedelec is implemented in ROS and abstracted well into nodes that encapsulate specific sub-functions. Additionally, the overall software framework of the test vehicle consists of visualization nodes for analysis

4. Solution Approach

and evaluation purposes. Due to its modularity and useful structuring, the overall software framework is comparatively simple to understand, maintain and expand.

However, the existing software is ill-suited for automatically conducting and tracking scientific experiments. For example, changing a parameter of the software often requires setting the same variable in multiple files. If the user accidentally forgets to change one of these duplicated parameters, the results may be inconsistent. In addition to that, neither the involved parameters nor the results are stored automatically and have to be saved and managed manually by the user. This may cause results to get mixed up or even lost. Furthermore, preparing the test vehicle (pedelec) for simulations or demonstrations involves a list of cumbersome steps that have to be executed manually on up to four terminal windows in parallel. While this slows down rapid prototyping, these manual steps also prevent conducting experiments sequentially in an automated way, e.g. for parameter tuning. Section 4.3 describes the empowerment of the software to conduct predefined experiments automatically while storing all of the involved parameters, source code and results.

4.1.2. Visualization and Quality of Sensor Signals

The existing software architecture uses the *bokeh* visualization library to plot trajectories and signals [83]. *Bokeh* allows for highly interactive plots with e.g. automatically highlighting corresponding points in all plots or synchronized scrolling, making it very suitable for offline analysis. The next few paragraphs first discuss some general visualization issues and then analyzes the signal quality of the acceleration, yaw rate and speed signals in more detail. Hereby, it also explains why signal filtering is decided against in the context of trajectory prediction despite of extremely noisy signals.

According to [84] a sampling frequency of 10 Hz captures most of bicycle kinematics, while higher sampling frequencies seem to be rather useful for analyzing impacts and road conditions from e.g. vertical accelerations. Therefore, a 20 Hz sampling frequency that was used to capture all of our data seems to be high enough for our use-case. However, the visualization scripts are parameterized such that only the data corresponding to the end points of the predicted trajectories are plotted. This means at a prediction horizon of $t_{\text{pred}} = 2 \text{ s}$, recorded at $f = 20 \text{ Hz}$, only every 40th point is visible. While this allows to easily visually compare the endpoints of a predicted trajectory with the actual end points, it leads to an extremely undesired aliasing effect in the remaining signals. The aliasing and a small jitter when replaying the recorded rosbags may cause the signals to look completely different every time, which makes a reliable analysis of the signals nearly impossible. This especially the case for high frequency signals such as the acceleration of the pedelec as shown in Figure 4.1.

4. Solution Approach

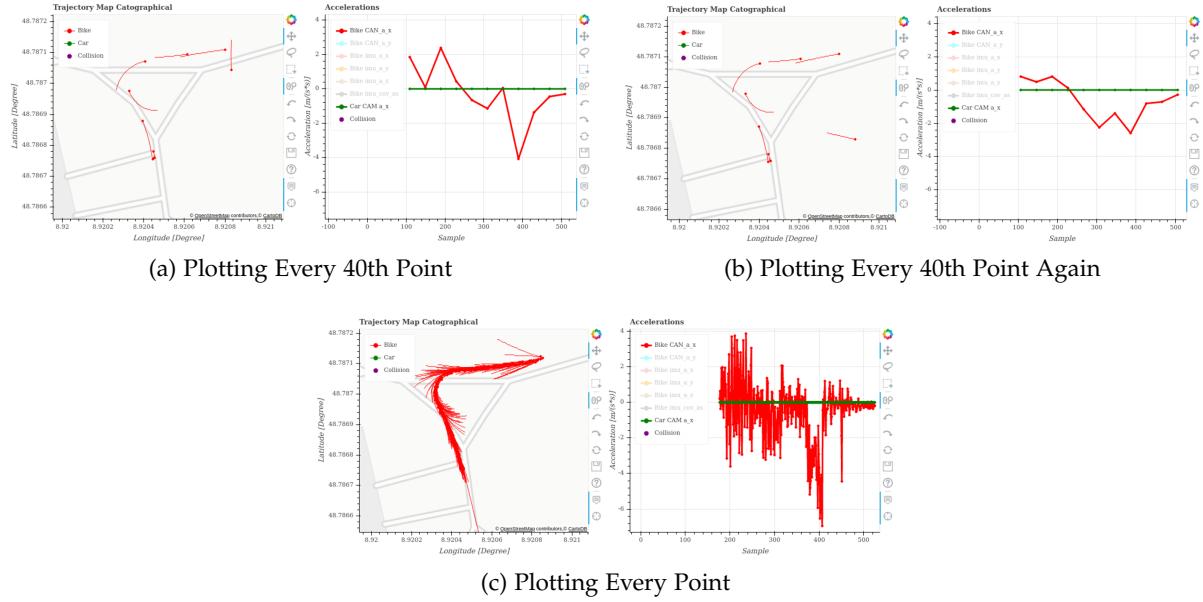


Figure 4.1.: Aliasing Effect When Visualizing Only Trajectory Endpoints

4.1c shows a plot where every recorded data sample of the acceleration signal is visualized, while the remaining sub-figures show plots where only visualize every 40th data point. Note how different the plots of the acceleration signal in 4.1a and 4.1b look despite resulting from the same rosbag and triggered with same bash script. The greater t_{pred} is, the worse the shown effect is. Therefore, it is crucial that every point is plotted when analyzing high frequency signals if possible. However, plotting every point of all of the signals leads to a blank visualization in *bokeh* for long measurements.

Furthermore, changing any of the visualization-specific parameters such as the aforementioned skip parameter or other plot settings requires the ROS bags to be replayed again in real-time. The user has to wait until the entire measurement has finished replaying before seeing the slightly changed plots. Since some of the measurements are up to an hour, this is highly impracticable for analysis and further development of the visualization scripts. Hence, section 4.4 describes how the visualization concept is changed to allow for faster analysis and development of the visualization.

In addition to the predicted trajectories of the pedelec and car, only the bare minimum signals used in the heading computation and motion primitive are plotted, which are further analyzed in subsection 4.1.3 and subsection 4.1.5, respectively. This includes the velocities, yaw rates, longitudinal accelerations and headings of the pedelec and car. In fact, three types of headings of the pedelec are visualized: The heading from triangulation of GPS positions, integration of IMU and the fused heading. The initial visualization of the these signals is depicted in Figure A.1. However, the sensors of the test vehicle (pedelec) transmit a lot more

4. Solution Approach

potentially useful signals such as the roll angle to describe the state or motion of the pedelec and should thus be visualized for further analysis. As a result, section 4.4 adds plenty of additional signals from the INS and DU CAN to the visualization, which are then analyzed for their signal quality similarly as described here.

Generally speaking, most of the signals from the pedelec show noticeable artifacts, especially when viewing every data point, compared to the corresponding signals from the car. Figure 4.2 compares the acceleration and yaw rate signals from a car (green) and a pedelec (red)¹. The highlighted section shows them accelerating next to each other in parallel towards the same direction at similar rates and steady state speeds.

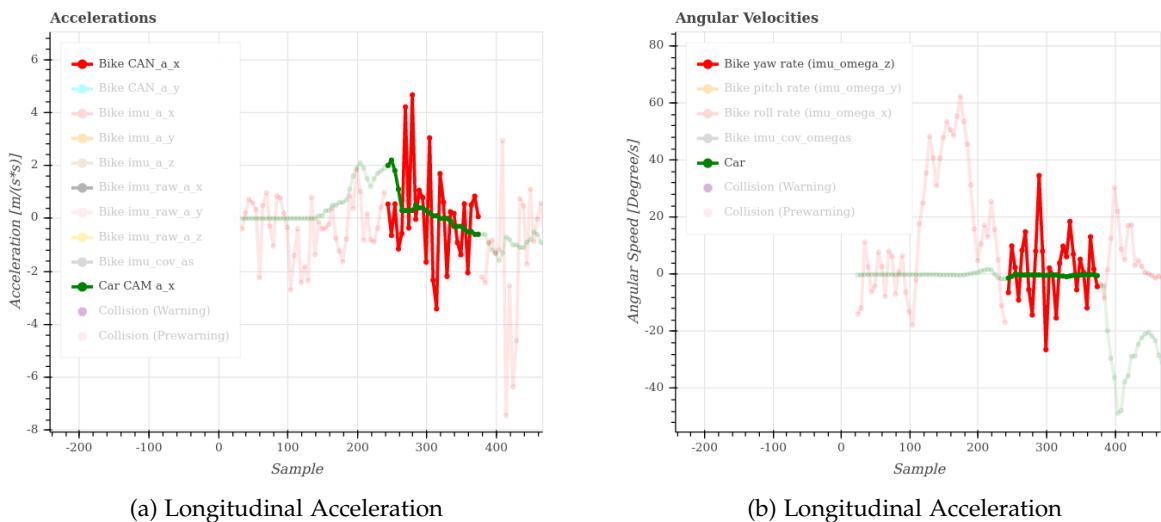


Figure 4.2.: Comparison of Signal Quality between Car and Pedelec

The signals from the pedelec are generally noisier and show trends less obviously. For example, the acceleration phase of the pedelec from sample 200 to 340 is barely visible compared to the one of the car from sample 140 to 240 (Figure 4.2a). In addition, the yaw rate signal of the pedelec shows wild oscillations even when performing the same driving maneuver as shown in the highlighted section (Figure 4.2b).

The acceleration signal, as depicted in Figure 4.1c, shows arguably the lowest Signal-to-Noise Ratio (SNR). However, this noise does not come from the IMU itself, which shows spikes of negligible amplitude when at rest as depicted in 4.3b, but rather from all of the vibrations or jerks resulting from rough road surface and small obstacles like curbs, potholes, etc.

¹Only every fifth point is plotted deliberately to ensure a fair comparison because of different sampling speeds of the sensors.

4. Solution Approach

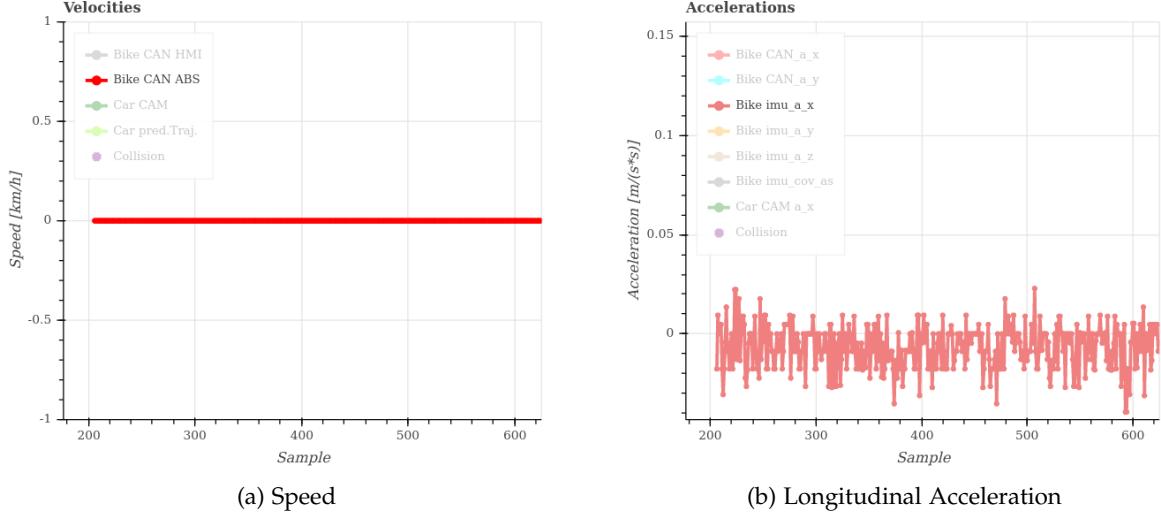


Figure 4.3.: BNO055-USB-Stick: Pedelec at Rest Showing Negligible Sensor Noise from IMU

The absence of a rear suspension and a thin tire sidewall of the pedelec do not seem to compensate for this enough. As shown on the right-hand side of 4.1c, the acceleration may easily fluctuate by a magnitude of $4 \frac{\text{m}}{\text{s}^2}$ from one data point to the next during a steady-state curve riding on concrete at nearly constant speed without pedalling. So, it seems obvious to low-pass filter the signals. While forward-backward filtering, a.k.a. *smoothing*, could potentially remove data artifacts such as outliers, noise and oscillations reliably from the signals without lag (*zero-phase*) in an offline manner, they cannot be applied online in real-time: They usually require a memory containing future data points or run the filter again backwards (non-causal).

In contrast, causal filters can smooth signals online in real-time but come with significant trade-offs. They always introduce some lag between the filtered and original signal and may damp the useful part of the signal too much, which is undesired when detecting certain situations as early and reliably as possible based on these filtered signals. For example, a simple running mean filter, which marks the most common Finite Impulse Response (FIR) filter in digital signal processing, with a window size of the most recent $N - 1$ samples has an *expected value* of

$$\tau_g = \frac{N - 1}{2} \quad (4.1)$$

samples on average for the group delay τ_g (see Equation 4.1) [85]. This delay is undesired since relatively large windows sizes of $N \approx 20$ are necessary to achieve sufficient filtering, which would result in a lag of half a second at a node frequency of $t_{\text{node}} = 20 \frac{1}{\text{s}}$.

Low order, flat response Infinite Impulse Response (IIR) filters such as Butterworth or Bessel filter generally have a smaller group delay, which depends on the cut-off frequency f_c as shown in Equation 4.2 [85].

$$\tau_g \frac{1}{f_c} \quad (4.2)$$

4. Solution Approach

However, since the noise frequencies in the relevant sensor signals are ever-changing and unpredictable as described above, finding and adapting appropriate FIR or IIR filter coefficients to produce the correct filter responses for every scenario in real-time becomes challenging. A predictive KF could potentially be useful to filter the acceleration and yaw rate in real-time. However, they often require sophisticated modelling of the motion states and sensor transformations. Ideally, some of the differential equations of the bicycle models described in subsection 3.2.3 can be used. Since the heading and position estimation that currently uses a KF needs to be reworked anyways as explained in subsection 4.1.3, it makes sense to extend the KF to additionally estimate the essential signals for trajectory prediction such as the acceleration, yaw rate and roll angle when improving the KF in the course of a future work.

Since the ever-changing noise and oscillations in the sensor signals would make finding appropriate IIR and FIR coefficients tricky and the fact that the existing KF will be reworked anyway in another thesis, this thesis does not further pursue signal filtering. Instead, the next section investigates whether alternative signal provide better signal characteristics and could be used instead.

While the yaw rate does not suffer as much from such high-frequency noise, there exist irregular oscillations that do not seem to depend only on the rider's torque and cadence applied to the pedals but also on other factors. The pedelec speed, road conditions and rider style could potentially attribute to these oscillations as well. The dependence on speed may partially be explained by the inherent lateral instability of a bicycle as explained in section 3.2. For slow (and sometimes very fast speeds), this may require the rider to execute more steering corrections in order to prevent the pedelec from falling over or becoming unstable. For example, the amplitudes of the yaw rate signal during the highlighted coasting section in Figure A.3c rise to a magnitude of 8 deg/s at low speed compared to only 4 deg/s at stable speeds in Figure 4.4d. Also, note how in Figure A.4d, the yaw rate amplitudes slightly decrease as the coasting speed approaches a stable speed range and then rise again at higher coasting speeds² Furthermore, many cyclist struggle to cycle along a perfectly straight line even when intending to go straight for a variety of reasons besides pedalling such as evading surface irregularities and small obstacles on the street, or for a lack of coordinative and motoric capabilities or even just for the fun of riding a mini-slalom. Figure 4.4 shows a pedelec cyclist, instructed to drive off as straight as possible and then coast along a straight road segment to eliminate the influence from pedalling. The roughly 20 second long coasting section is shown in 4.4c and highlighted in all other sub-figures as well.

²While these plots are from different IMUs, the trends in the acceleration and yaw rate signals show across all sensors as described in section 4.6.

4. Solution Approach

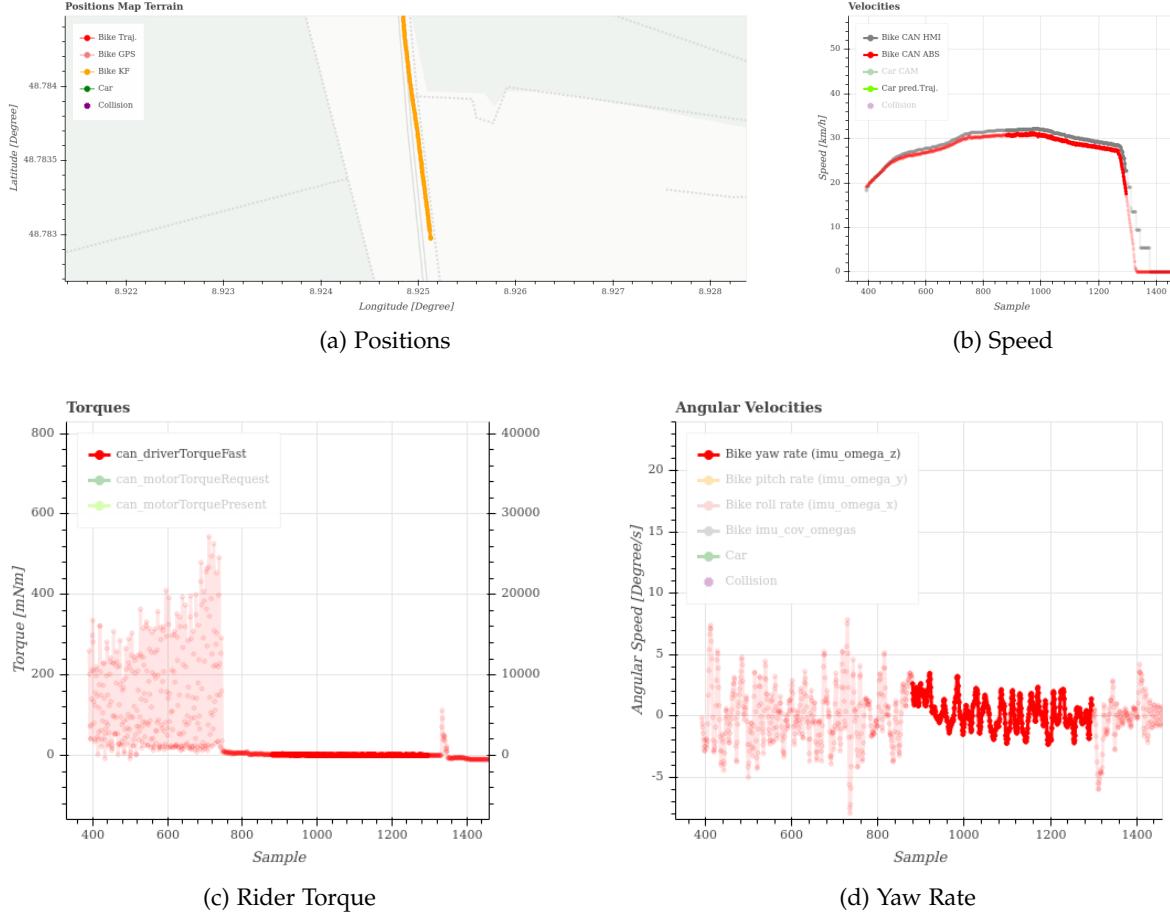


Figure 4.4.: BNO055-USB-Stick: Oscillating Yaw Rates when Coasting for around 20 Seconds

Note how the yaw rate signal in the highlighted section of Figure 4.4d still oscillates noticeably the entire time, even under these "ideal" coasting-in-a-straight-line circumstances. In some cases, as shown in Figure A.3d of the appendix, the yaw rate signal of the highlighted coasting phase can barely be distinguished between the driving-off phase from sample 90 to 240. In addition, the oscillations in the yaw rate during the driving-off phase, when zooming in to e.g. sample 530 to 590 as shown in 4.5c, do not seem to correlate with the nearly constant frequency as well as maximum/minimum amplitudes of the rider torque as shown in 4.5b³.

³(The nearly constant torque frequency can be verified with the cadence signal in Figure 4.5a

4. Solution Approach

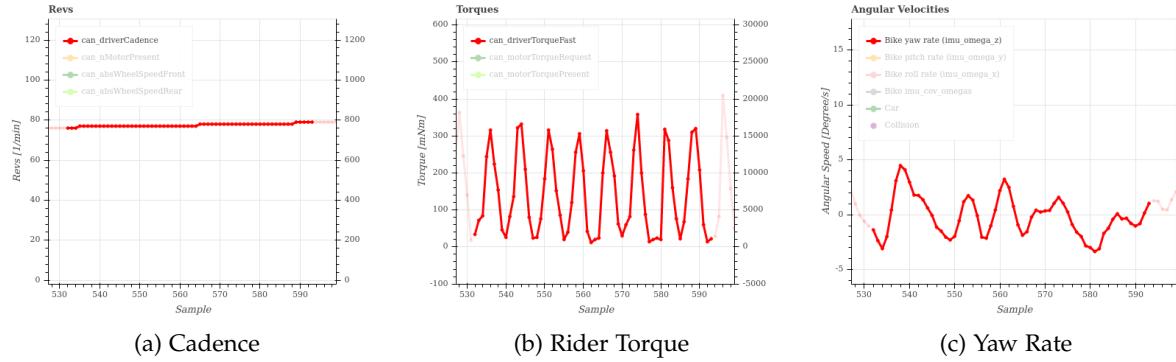


Figure 4.5.: BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence

For further offline analysis of the frequencies amounting to the oscillating yaw rate signal, a Fast Fourier Transform (FFT) is performed over a section of constant pedalling and riding straight, similar to Figure 4.5. Figure 4.6b compares the rider torque signal with the yaw rate signal in both time and frequency domain.

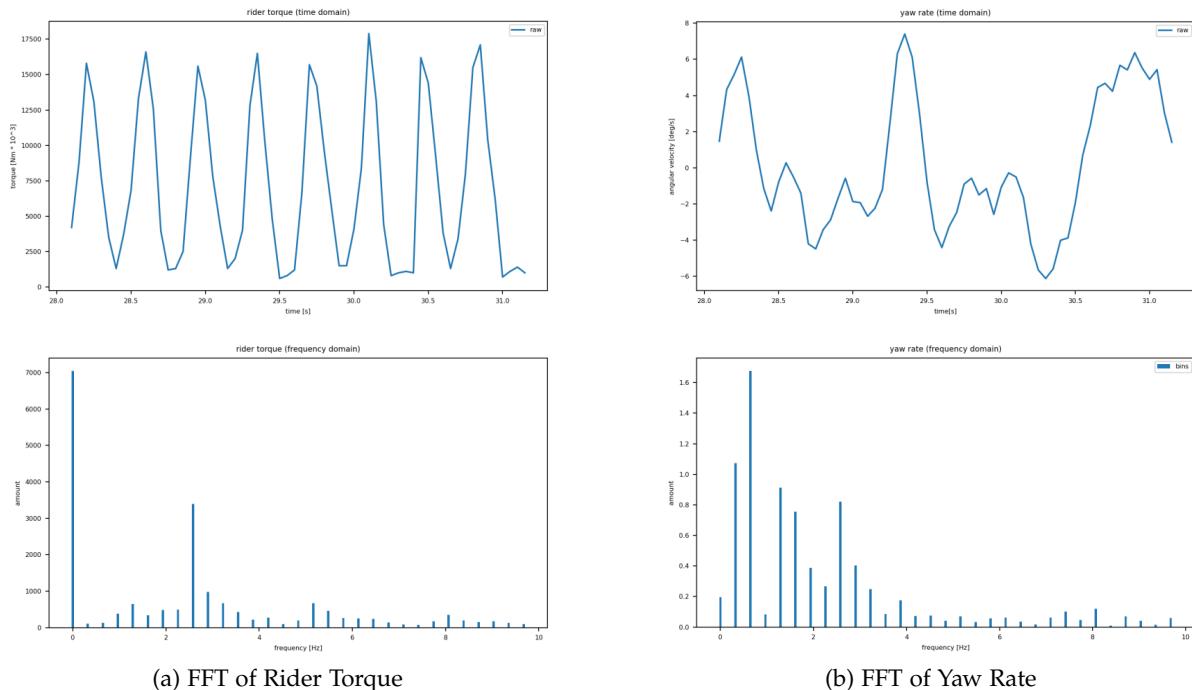


Figure 4.6.: BNO055-USB-Stick: FFT of Rider Torque Compared to Yaw Rates at Constant Cadence

While the yaw rate signal in Figure 4.6b contains the frequency peak around 2.5 Hz from the rider torque in Figure 4.6a, which corresponds to double of the measured cadence of 75

4. Solution Approach

rpm, it also contains many other frequencies that do not show up at all in the rider torque. In fact, in many cases it becomes hard to tell which of these frequencies correspond to the actual signal and which ones to undesired noise/oscillations.

These graphs hence indicate that the oscillations in the yaw rate signal cannot trivially be accounted for by only filtering out the peaks from the rider torque at the corresponding cadence using conventional filters as other factors seem to contribute to the oscillations as well whose oscillations frequencies do not seem to be measured, known or changing constantly. The described trends in the acceleration and yaw rate signals also show across more expensive sensors as described in section 4.6.

As expected, the speed signal from the default 1-pulse-WSS (gray signal in A.3b) is highly non-differentiable due to the step-like behavior resulting from only one update per revolution which takes increasingly longer the slower the wheel turns. The 56-toothed-WSS from the ABS (red signal in A.3b) minimizes this effect, leading to a more usable signal.

Finally, the signals may also show data outliers. This is especially the case for the acceleration signal but also observable in various bicycle CAN signals such as WSS as depicted in A.3b between sample 240 and 260. It seems the impulse noise could stem from both shocks of the road and electromagnetic inference of the DU.

Subsections 4.8.3 and 4.8.4 cover how trajectory predictions errors arising from poor signal quality are circumvented.

4.1.3. Heading and Position Estimation

The *ublox MAX 8* receiver of the utilized *Navilock NL-8012U* GNSS dongle has an accuracy of 2.0 - 4.0 m (at one standard deviation) [86]. Merely internal filtering of the GNSS assures that the positions look fairly consistent to each other and do not jump several meters from one time step to another. However, the resulting path might still be offset by a few meters as shown in Figure 4.7 where the pedelec was actually ridden strictly on the approximately 3.5 m wide paved roads.

4. Solution Approach

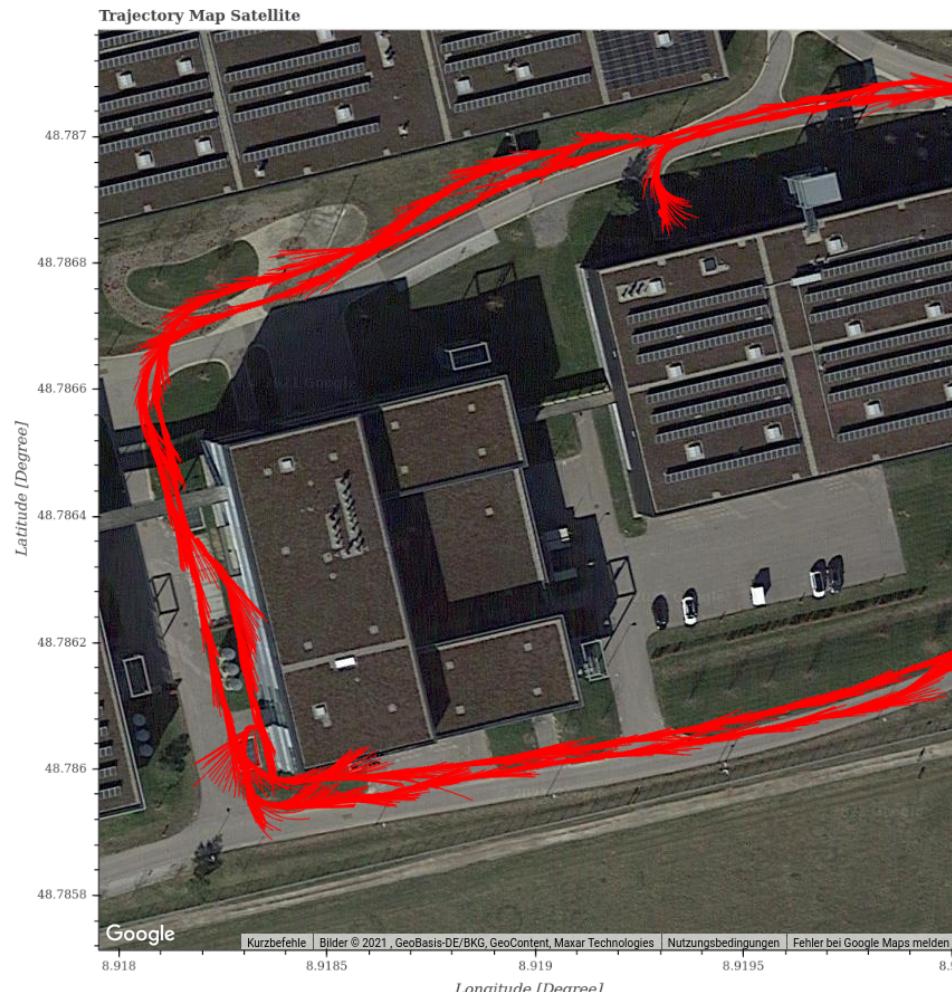


Figure 4.7.: Navilock NL-8012U: Inaccurate Positioning

In the top left section of Figure 4.7, note how the trajectory predictions originating from the highly inaccurate positioning falsely indicate the pedelec to always ride on the grass off to the side of the paved road. If another vehicle drove on the same paved segment and was about to collide with the pedelec head-on, the ADAS may not recognize a collision at all because the vehicle's trajectories might end up on the other side of the road and never intersect with the pedelec's. Alternatively, if the CD was parameterized very generously to compensate for these inaccuracies, it would lead to a high FP rate any time the two vehicles are close to each other. For example, it may detect a head-on collision even if the pedelec and vehicle drove on separate lanes.

First evaluations within the Robert Bosch GmbH show that if two road users approach each other at 90 degrees for a perfect dead centered collision but have a circular covariance in their predicted positions of 2.0 m, the crash probability of an actual collision occurring reduces up to only 50%. Since the accuracies of the initial positions from the GNSS already exceed this

4. Solution Approach

target covariance, the overall system stands no chance of achieving this target. After further analysis and discussion with positioning experts for mobile robots at Robert Bosch GmbH, it becomes clear that achieving a target positioning accuracy of below roughly 1.0 m using the current low-cost GNSS would mean extraordinary effort and require redoing the position estimation completely.

Furthermore, the current 3D EKF shows undesired behavior, indicating it may be tuned inappropriately, conflict with ROS or even buggy.

KFs take time to converge until they provide reasonable values. However, it seems strange that the filtered positions approach from Algeria as shown in Figure 4.8, which is hundreds of kilometers away from the initialized and actual location in Renningen, Germany.

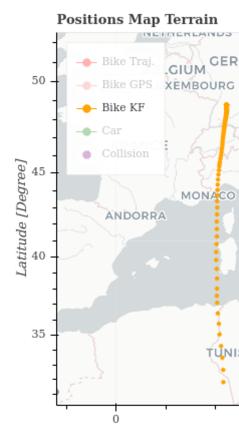


Figure 4.8.: Converging Positions

In addition, the filtered positions oscillate around the GPS position and may add a positioning offset as shown in Figure 4.9, where the pedelec is at rest and the filtered positions oscillate around the reported GPS position with a radius around a meter.

4. Solution Approach

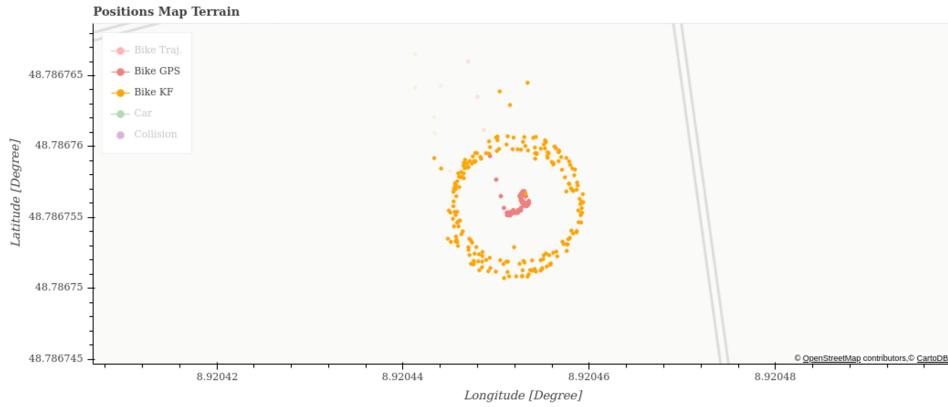


Figure 4.9.: Filtered Positions Oscillating around Stationary Pedelec

As a result, the headings oscillate from 0 to 360 and back in this case.

Finally, the EKF does not compensate the position and heading well when GPS signal is lost despite having valid information from the WSS and IMU at hand. The reported positions of the GNSS in Figure 4.10a show a pedelec riding from West to East while facing a short GPS outage in the middle.

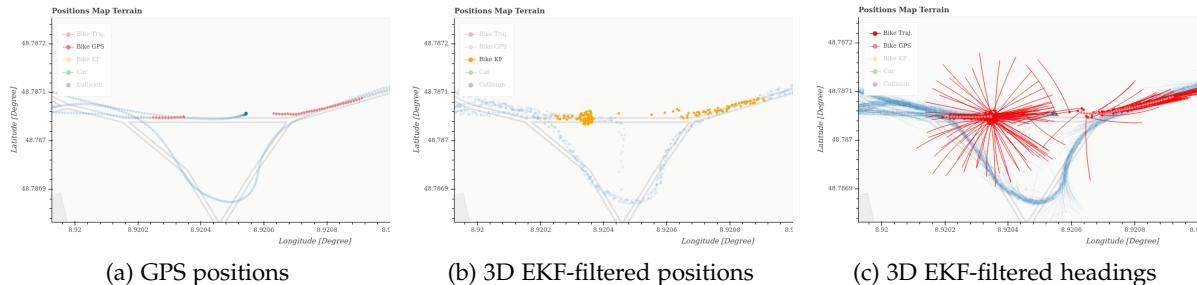


Figure 4.10.: BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence

As shown in Figure 4.10b and 4.10c, the filtered positions and estimated headings oscillate around the last known GPS location and then jump to the new GPS location where the GNSS receives a valid signal again. The headings are visualized by their resulting trajectories which obviously originate from the heading.

Since improving the positioning using a low-cost IMU would require replacing the 3D EKF from [55] completely anyway, the described problems of the state estimation will be tackled as part of a separated thesis by another student. However, section 4.6 describes the integration of an INS to obtain a more accurate and precise positioning and attitude estimation.

4.1.4. Statistical Evaluations of Trajectory Errors

The idea behind the metrics from [55] explained in section 3.1 is to allow a separate evaluation of the impact from the longitudinal and lateral components of the trajectory prediction model. For example, in the case of the `const_a_yawr` model, Δx and Δl represent the errors from the predicted accelerations while Δy and $\Delta \sigma$ represents the errors from the yaw rate. However, Muresan already noted that there could still be a noticeable influence from one component to the other e.g. a large $\Delta \sigma$ will result in a shorter projection on the x-axis compared to a small $\Delta \sigma$.

Furthermore, there technically exists an ambiguity in these metrics as shown in Figure 4.11.

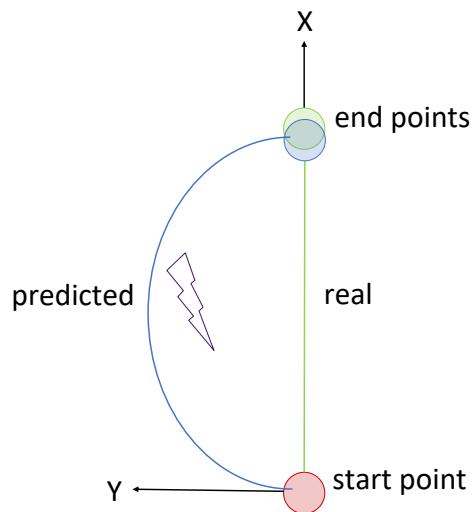


Figure 4.11.: Ambiguity in Metric

The real path in green takes place straight along the x-axis while the predicted trajectory in blue takes an arced path. Since they both end up in the same endpoint, all of the metrics would equate to zero, thus falsely indicating a perfect prediction while the path might differ significantly. Similarly, this could lead to multiple distinct trajectories resulting in the same errors. This ambiguity can be circumvented by running the statistical evaluation multiple times, ideally as often such that the end point of the predicted trajectories will represent every time step of the to be investigated prediction horizon.

The L2-distance between the predicted and actual endpoint could be used to obtain a combined positional accuracy (Equation 4.3) and precision (Equation 4.4).

$$ACC_{\text{pos}} = \sqrt{\Delta \mu_x^2 + \Delta \mu_y^2} \quad (4.3)$$

$$PREC_{\text{pos}} = \sqrt{\Delta \sigma_x^2 + \Delta \sigma_y^2} \quad (4.4)$$

4. Solution Approach

However, this would suffer from the same ambiguity as mentioned above.

Since the statistics are averaged over the entire trip, there might be a considerable imbalance in e.g. straight sections vs. curves for different trips. So a better overall score for a certain trip does not guarantee a better performance in another trip as well. Also, a bad performance in underrepresented segments may barely influence the overall score.

Because the starting and stopping of a recording on the vehicle computer using ROS requires typing on a keyboard, all measurements include periods at the beginning and end where the pedelec is at rest. These stationary periods are problematic for the position and heading estimation as the EKF needs movement to converge and deliver useful values. As depicted in Figure 4.8, enormous positioning errors result while the EKF is converging which carry over to enormous Δx , Δy and Δl errors in the trajectory prediction as these initial positions and headings mark the start of each trajectory. Similarly, the EKF positions and headings circulate around the true location before stopping the measurements as shown in Figure 4.9. While this circling motion causes relatively low positioning errors, it causes significant errors in $\Delta \sigma$. Of course, the longer the riding time is compared to these stationary periods, the smaller the impact of these falsifications will become. Specific values on how much falsification is added by these stationary phases are presented in subsection 5.1.1.

As explained above, in order to obtain uncompromised absolute error values from the statistical evaluation it is crucial to exclude the beginning and end of every measurement from the statistical evaluation if the EKF is used. The visualization scripts allows to specify the number of samples from the beginning to be omitted but has to be adapted to discard a specified number of samples at the end as well. Since the described ambiguity in the metrics is unlikely to have a significant effect and can be resolved, the same metrics from [55] are used and not adapted further. This also eases the comparisons of newly developed trajectory prediction models with models from previous works. Moreover, segmenting the track into different sections might be useful to pinpoint the strengths and weaknesses of different trajectory prediction models more easily (see subsection 4.8.1).

4.1.5. Motion Primitive

As introduced in section 3.1, the motion primitive from [17] uses piece-wise linear segments obtained from integration to compute future positions. A nearly identical motion primitive is also used in the state equations of the EKF for the heading and position estimation. This subsection analyzes possible downsides and improvements of the current motion primitive.

The motion primitive essentially models a point movement restricted to a 2D plane with a nonholonomic constraint that prevents side slip of the rear wheels. While this model simplifies a car's motion appropriately, it does not use any of the unique physical characteristics of a bicycle. For example, this model does not allow for a roll motion of the pedelec. It may make sense to extend this motion primitive to a 3D model that either uses kinematic bicycle

4. Solution Approach

models e.g. similar to subsubsection 3.2.3.1 or constrains the movement of a rigid body in 3D more appropriately.

Furthermore, the piece-wise linear segments lead to step-like trajectories for sharp turns with a small radii of curvature as shown in Figure 4.12a when using `const_a_yawr` for $t_{\text{pred}} = 4 \text{ s}$.

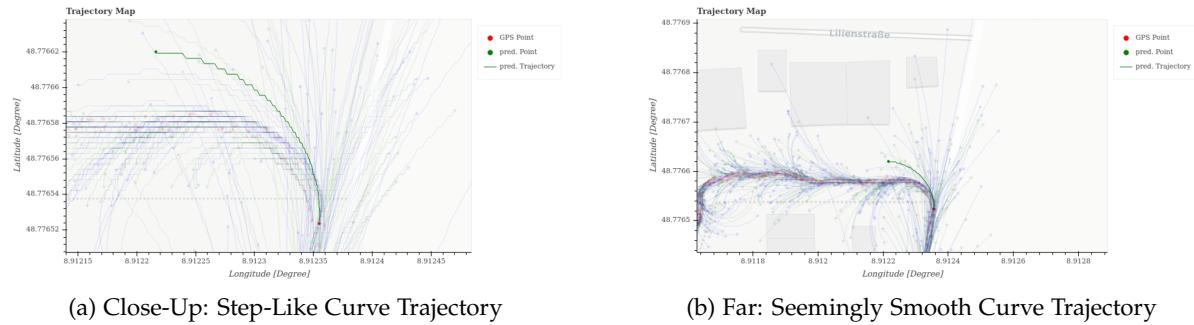


Figure 4.12.: BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence

However, since $dt = 0.05 \text{ s}$, a trajectory for $t_{\text{pred}} = 4 \text{ s}$ consists of 80 of these linear segments, leading to a seemingly smooth curve trajectory in the end as depicted in Figure 4.12b. The error when approximating an actual curve trajectory with the piece-wise linear segments thus seems to be negligible compared to the errors made in predicting the longitudinal and lateral signals which are fed into the motion primitive when predicting a future trajectory as further analyzed in subsection 4.1.6.

Since, the EKF for estimating the pedelec's initial state likely needs to be reworked completely to achieve better positioning, it makes sense to investigate a more sophisticated motion primitive in this follow-up master thesis. Then, the trajectory prediction may just reuse this same motion primitive. Moreover, the linearization likely does not impact the final trajectory prediction errors significantly. As a result, this thesis keeps the motion primitive unchanged.

4.1.6. Trajectory Prediction Models

The trajectory prediction models from the previous works [17], [54] and [55] as introduced in section 3.1 can be formalized such that they essentially all use the same basic motion primitive from [17] to compute future positions and only differ in how their input signals (heading, yaw rate, speed and acceleration) are predicted/modelled in the future. Consequently, the following two sub-subsections analyze the resulting trajectories when predicting these inputs signals using either by holding the signal values constant or linearly extrapolating them. As mentioned in section 3.3, improving the ML-based trajectory prediction models is not part of the research focus in this thesis. Hence, the ML approaches are not analyzed in detail.

4. Solution Approach

4.1.6.1. Constant Signal Trajectory Prediction Model

The `const_a_yawr` model from [17] marks the simplest trajectory model by simply predicting the same instantaneous longitudinal acceleration and yaw rate for all future time steps of the prediction horizon. This model essentially keeps both the longitudinal and lateral signals constant over time and feeds the values step by step into the motion primitive where they are further integrated to future positions. While this is computationally efficient and explainable, the constant acceleration and yaw rate assumptions do not apply in all situations where they may lead to inaccurate trajectories. For example, Figure 4.13 shows the trajectories generated from the `const_a_yawr` model for $t_{\text{pred}} = 2$ s during a long and smooth turn, divided into curve entry, middle of curve and curve exit and their corresponding yaw rate signals.

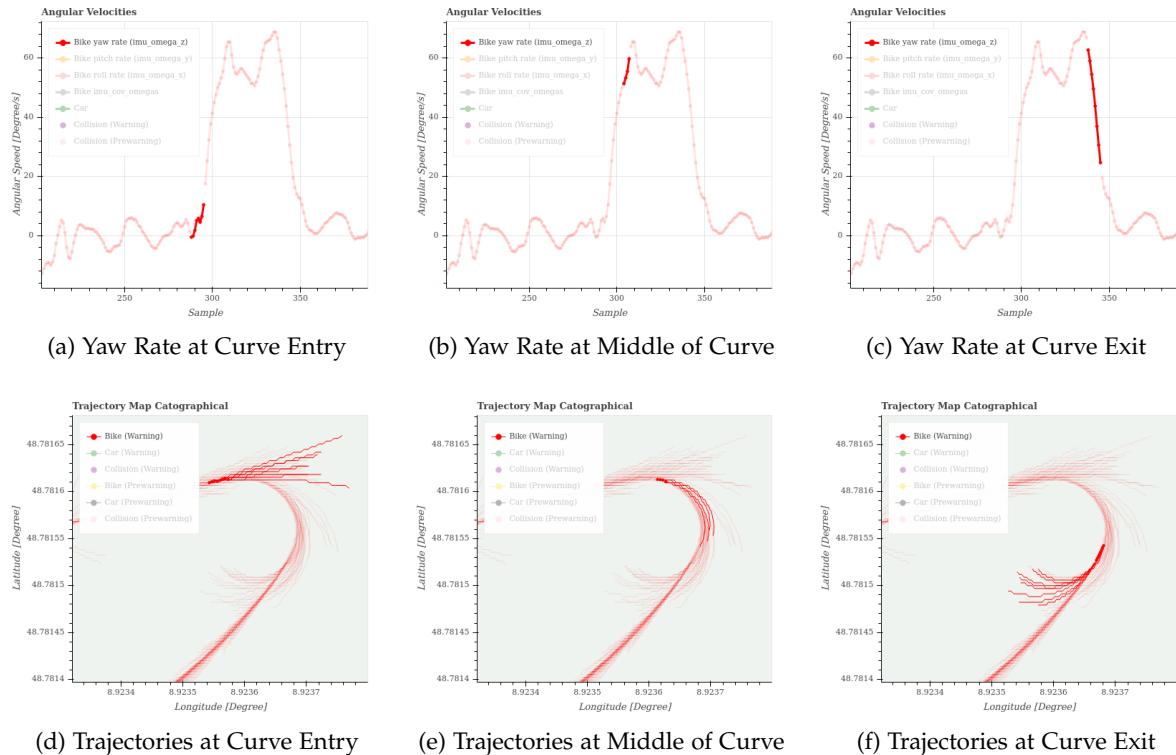


Figure 4.13.: `const_a_yawr`: Trajectories and Yaw Rates During a Curve

Generally, the trajectories at the curve entry undershoot the turn compared to the actual path (Figure 4.13d). Since the constant yaw rate assumption of the models holds the initial yaw rate at its same value for the entire prediction horizon while the actual yaw rates keep rising (Figure 4.13a), the predicted Area Under the Curve (AUC) underestimates the actual AUC for every future time step⁴. Consequently, the underestimated AUC of the predicted yaw rate signal leads to an underestimation of the heading deltas, which manifest themselves as

⁴Unless the prediction horizon is long enough for the actual yaw rates to decrease again. Then, an instantaneous value held constant could result in the same AUC as the actual signal by chance.

4. Solution Approach

trajectories that point too straight. In the middle of the curve, the actual yaw rates sometimes oscillate roughly around their maximum magnitude (Figure 4.13b), which means holding an instantaneous yaw rate approximates the actual AUC well, if the prediction horizon roughly matches the duration of steady state curve riding. Since these conditions are met in Figure 4.13b, the predicted trajectories (Figure 4.13e) originating near the apex curve correspond well to the actual path in this case. However, if the prediction horizon is shorter than this duration, the trajectories would undershoot similar to the curve entry case. If the prediction horizon is too long, the predicted trajectories will overshoot, similarly to the curve exit case. As already mentioned, the predicted trajectories generally overshoot the curve exit (Figure 4.13f), since the AUC resulting from the constant yaw rate predictions repeatedly overestimate the actual AUC from the actually decreasing yaw rates (Figure 4.13c).

Similarly, just because the rider is momentarily decelerating at a point in time does not mean he/she will continue to do for the entire prediction horizon and may result in under- or overestimated AUC of the actual acceleration signal. However, in this case, the impact of a prediction error in the acceleration signal compared to the true acceleration signal results in quadratic errors in trajectory length with respect to time. This can be explained by the quadratic term $\frac{1}{2}at^2$ that results from double integration of a constant acceleration with respect to time in order to obtain the position over time as shown in Equation 4.5:

$$x(t) = \frac{1}{2}at^2 + v_0t + x_0 \quad (4.5)$$

The notations v_0 and x_0 refer to the initial speed and position at time $t = 0$. The full derivation can be found in Equation A.1 of the appendix.

On the other hand, a prediction error in the yaw rate results only in a linear relationship of the heading angle with respect to time $\omega_z t$ as shown in Equation 4.6 (full derivation in Equation A.10 of appendix).

$$\psi(t) = \omega_z t + \psi_0 \quad (4.6)$$

Here, the notation ψ_0 refers to the initial heading at time $t = 0$. Figure 4.14 visualizes the impact of quadratic relationship of the position and linear relationship of heading with respect to time when doubling the prediction horizon from $t_{\text{pred}} = 2\text{s}$ to $t_{\text{pred}} = 4\text{s}$.

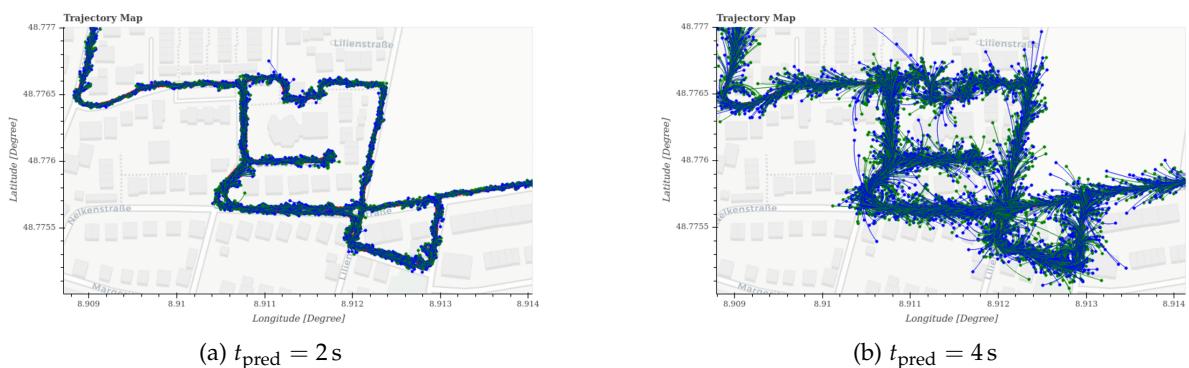


Figure 4.14.: Comparison of Trajectory Lengths for $t_{\text{pred}} = 2\text{s}$ and $t_{\text{pred}} = 4\text{s}$

4. Solution Approach

Note how the length of the predicted trajectories multiplies by a factor of four while the direction only by a factor of two when doubling the prediction horizon.

Furthermore, subsection 4.1.2 showed that the acceleration signal is extremely noisy while the yaw rate signal suffers from oscillations. The noisy acceleration signal, where jumps of $4 \frac{\text{m}}{\text{s}^2}$ are easily possible from one sample point to the next, result in greatly over- or underestimated AUC compared to the actual signal. Figure 4.15 highlights two succeeding trajectory predictions that jump in length as a result of these huge over-and underestimations of the AUC.

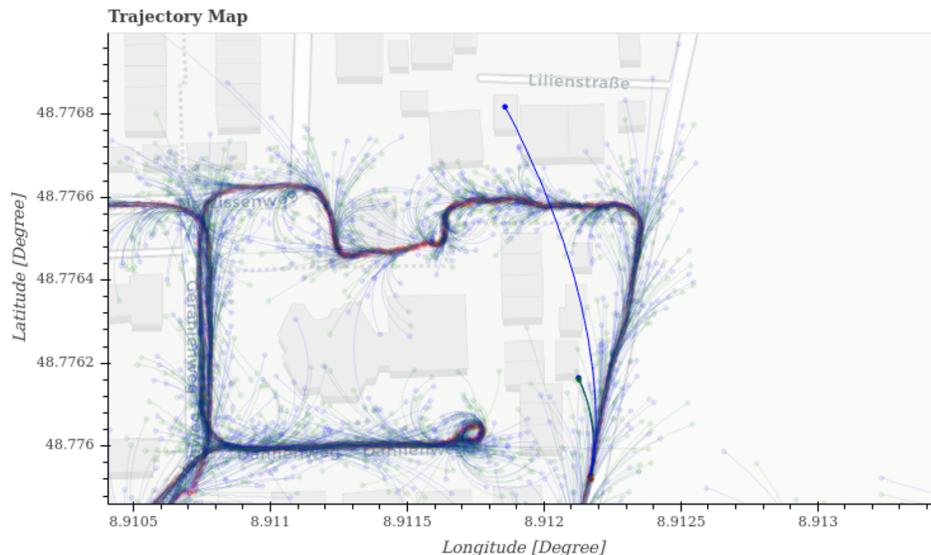


Figure 4.15.: Two Succeding Trajectory Predictions Jumping in Length

Anytime an extreme instantaneous acceleration value is held constant and integrated over the entire prediction horizon thus leads to a big error in the length of the predicted trajectory. The oscillating yaw rates result in trajectories swinging from left-to-right despite riding straight as already shown in Figure 1.2 during the introduction of this thesis.

4.1.6.2. Extrapolated Signal Trajectory Prediction Model

The `extrapol_speed_yawr` model from Kailasam fits a line through recent data points of each the speed and yaw rate signals using least squares function approximation and extrapolates future values along these fitted lines until the end of the prediction horizon [54]. It thus does not rely on the noisy acceleration signal and uses the more robust speed signal instead. The impact of the linearly extrapolated speed signal on the position remains quadratic, despite only integrating once, as shown in Equation 4.7.

$$\frac{1}{2} p_1 t^2 + p_0 t + x_0 = x(t) \quad (4.7)$$

4. Solution Approach

The notation p_0 and p_1 refers to the zero-th and first degree polynomial coefficients that linearly fit the most recent *speed* values. This stems from the fact that integrating a linear polynomial function results in a quadratic polynomial function (see full derivation in Equation A.15 of appendix).

However, the impact of the linearly extrapolated yaw rate on the heading becomes quadratic as shown in Equation 4.8, opposed to linear when keeping the yaw rate constant.

$$\frac{1}{2}p_1 t^2 + p_0 t + \psi_0 = \psi(t) \quad (4.8)$$

In this case, p_0 and p_1 refer to the zero-th and first degree polynomial coefficients that linearly fit the most recent *yaw rate* values. As explained before, this stems from the fact that integrating a linear polynomial results in a quadratic polynomial function (see full derivation in Equation A.21 of appendix).

Compared to the previous methods, the extrapolation incorporates data from the past instead of relying solely on a single instantaneous value. The fitting of a line through recent data point has a similar effect to filtering in the sense that single noisy outliers do not have such an extreme impact anymore. This theoretically makes the trajectory prediction more robust against noise and improves the behavior at the curve entry and exit compared to holding the instantaneous values constant. However, not in all cases does it make sense to project the past to the future, nor does it improve the signal prediction in all cases compared to holding the acceleration and yaw rate constant. When extrapolating near changes of trend in the signal, e.g. when braking-turning-accelerating, this method may introduce larger errors compared to holding a value constant. Similarly, some sections of the signals may show a nearly constant trend despite oscillations. These effects are discussed more detailed in subsubsection 5.1.4.2.

Note that Kailasam chose to use five times as many points for fitting the line compared to the number of extrapolated points from that line [54]. This can also be stated in terms of the duration of the prediction horizon t_{pred} as follows:

$$t_{\text{in}} = 5 \cdot t_{\text{pred}} \quad (4.9)$$

It seems, that the input duration t_{in} in Figure 3.4 may be too long to incorporate useful information from the past. Using this parameterization, when predicting two seconds into the future, essentially the 10 most recent seconds worth of data points get averaged. This could make the prediction worse in some cases compared to `const_a_yawr`. For instance, if the pedelec has been "seeing" a curve entry for the last 10 data points with monotonically and sharply rising yaw rates, similar as highlighted in 4.13a, and the previous 9.5 seconds mostly contained low yaw rates oscillating around zero in the case of riding straight, the slope of the fitted line would be close to zero as well. The impact of the most recent 10 data samples, which are arguably most important, would be negligible small. The resulting trajectories would indicate the pedelec is still going straight. It would take quite some time to collect enough high yaw rates, perhaps until the curve exit, for slope of the fitted line to change significantly. However, in this case, the pedelec might ride straight again while the trajectories

4. Solution Approach

would now indicate a curve. Essentially, the predicted trajectories constantly lag behind in curves. In return, the predicted trajectories are more robust against the oscillations in the yaw rate and are thus better for long and straight segments.

4.1.7. Collision Detection and Warn Strategy

The current implementation is able to detect and warn the cyclist in a computationally efficient one-shot dynamic range approach by using the heading difference between the car and pedelec as explained in section 3.1. However, there may be unintended implementation errors and general drawbacks from this method, which are further analyzed now.

Some implementations details may have been unintended:

The dynamic range is constructed by assuming that the GPS positions of both vehicles correspond to a location at the middle of the rear axes of each vehicle. However, the CAM actually reports the car's positions at the middle of the front bumper while the pedelec reports the positions at its GNSS antenna [43]. This means the actual range needs to be larger to fully enclose the vehicle. Muresan noted that in real-world tests the collision range had to be increased from 4 m to 11 m to ensure CDs [55]. To be fair, this increase was mostly necessary due to the large combined errors from positioning and trajectory prediction as described in subsection 4.1.3 and subsection 4.1.6.

Also, when computing the crash direction, the headings at the location with the minimal distance between the pedelec's and car's GPS points was used instead of the headings at the location where the vehicles actually start colliding.

In addition, the dynamic range method itself has several drawbacks:

First, the dynamic collision range does not account well for the reported positions of the GNSS which are mounted with an offset or transformed to another virtual location. If the pedelec is within the vicinity of the dynamic range of the car e.g. when both participants are waiting at a red light, the CD keeps triggering alarms even though there is no risk of collision because both participants are at rest⁵.

Furthermore, the collision alarms triggers at the same location irrelevant of the speed at which the pedelec rider approaches the car.

Moreover, there exist multiple configurations with the same heading difference that are modelled with the same dynamic range but would result in different outcomes (collision or no collision). Some of these ambiguous configurations are shown in subsection 4.9.1.

Finally, the location where the pedelec and car collide might not always be appropriate to base the TTC on. For example, if a car just barely clips the bicycle at its rear tire in a 90 degree intersection, the pedelec should either accelerate slightly or brake way earlier in order to come to a stop before the driving tube of the car.

From these points it becomes clear that the CD also needs to be improved to decrease

⁵To avoid a continuous beeping and handle bar vibrations in this scenario, alarms were suppressed by imposing a minimum time of a few seconds between succeeding alarms.

the FP rate. It seems that the CD should not just be a simple function based on the heading difference of the two road users. Furthermore, it may make sense to split this one-shot approach up into multiple stages. These improvements are discussed more detailed in section 4.9.

4.2. Conceptual Overview

This section provides a general overview of the involved software components and how they interact. First, it explains the two main operating modes of the pedelec's software and then presents a simplified ROS architecture to briefly explain the interaction between the nodes.

The software has two main operating modes that are coupled to different use-cases. On one hand, the software can run online in real-time to execute the ADAS e.g. for demonstration purposes. On the other hand, the software may also be used offline e.g. for simulation purposes to improve the system. Figure 4.16 shows the overall software framework used in this thesis with its two operating modes.

The left-hand side shows the online mode of the software used to test and demo the software in the real-world or collect data. An desktop launcher can be configured to call the `ride_measure_demo_ebike.bash` script with pre-defined flags when double-clicked that represents a unique use-case e.g. starting all ROS nodes for testing or a live-demo with the car and recording a rosbags. Of course, the bash script can also be called directly from the terminal without using the desktop launcher. The `ride_measure_demo_ebike.bash` script automatically parses a global configuration file, checks whether the devices are connected as defined in this configuration file, rotates the screen output for the upside-down display of the pedelec, sources the catkin workspace, starts the ROS master, then calls the appropriate ROS launcher which starts only the relevant nodes (all expect visualization) for the online operation and optionally triggers the recording of a rosbag in a directory defined in the configuration file.

The right-hand side shows the offline mode used to simulate the software from previously recorded rosbags. The user just needs to define the path of the rosbag in the global configuration file as well as configure a set of parameters to be iterated through and the desired visualization type in `automate_experiments.bash` before calling the script from the terminal without any parameters. Among other things, it calls another script `simulate_visualize.bash` with appropriate flags which then plays the pre-recorded rosbag and triggers a simulation by launching `sim.launch`. These two scripts are explained more precisely in section 4.3. The simulations then starts all relevant nodes (all expect sensor drivers and Human Machine Interface (HMI)). Starting the sensor drivers is not desired as these topics should from the rosbag, and only the higher-levels signals are recomputed when simulating. After the rosbag has finished playing and the visualization is generated, all of the generated artifacts, which are explained in more detail in section 4.3, are stored before the `automate_experiments.bash` script automatically executes the next experiment until all parameter sets are iterated through (symbolized with the bold arrow).

4. Solution Approach

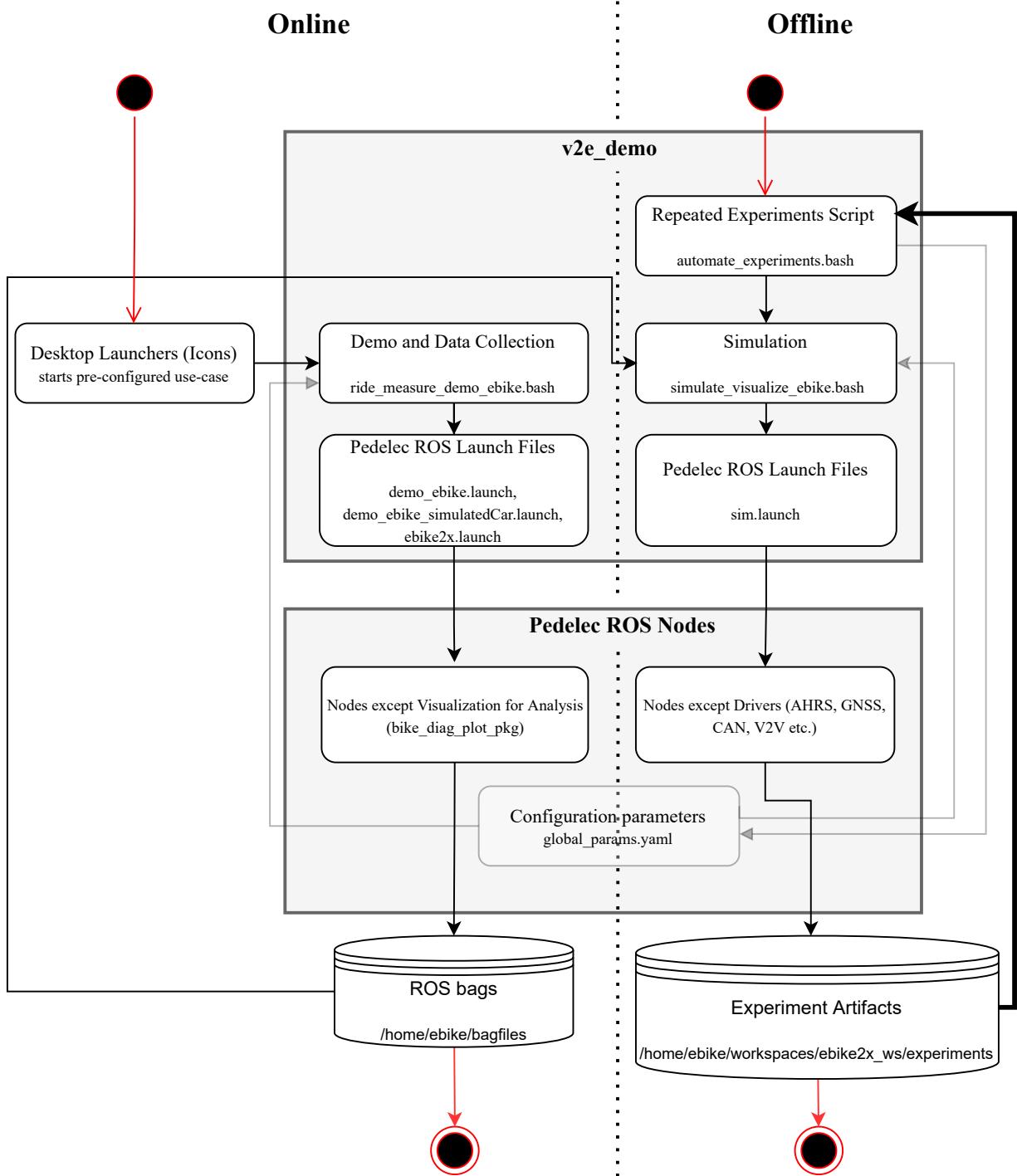


Figure 4.16.: Overview of the Entire Software Framework

4. Solution Approach

The software architecture in ROS structurally remains similar to [17] and [55] but additions and improvements are made. While all nodes run in parallel, the logic of the software still works sequentially. Figure 4.17 shows a simplified overview of the new software architecture in ROS⁶. Every rectangle represents a catkin package which may contain multiple nodes. The packages in dark grey represents hardware drivers for the sensors and actuators, which were written by other contributors. However, some of these were newly integrated into the system as described later on in section 4.6. The main nodes inside of the packages are represented with rounded rectangles and can have one of three colors: red means the node only publishes topics, which is the case for the sensor drivers. Blue means the nodes subscribes to previously generated topics but also publishes new topics, which is the case for core logic of the ADAS. Green means the node only subscribes but does not publish new topics, which is the case for the output drivers controlling the actuators or generating visualizations. Helper functions in separate files are not explicitly shown to avoid clutter. Some nodes are transparent which means they are not launched when using the default `demo_ebike.launch`. Which nodes get started depends on the launch file as well as the parameters configurations in `global_params.yaml`. When using `demo_ebike_simulatedCar.launch` the `v2x_cam_simulated_deserializer_etsi.cpp` node instead of the `v2x_cam_deserializer_etsi.cpp` node is launched. This serves to virtually place a car at a pre-defined location for independent end-to-end validation as explained in section 4.5. `ebike2x.launch` only launches the nodes directly relevant for the pedelec and does not start any V2V or HMI at all. The simulation `sim.launch` starts the least amount of nodes and further omits all the driver nodes because the topics are fed from a pre-recorded rosbag instead. Furthermore, parameters determine which nodes get launched. Note that the figure does not show the influence of every "switching" parameter as this would clutter the diagram. The names of the main topics used to communicate between the nodes are represented with yellow ellipses. The message type is also stated. Arrows represent the flow of topics among the nodes with dots indicating a split. Topics originating from or supplying currently inactive nodes are left out to avoid cluttering the figure further.

The overall signal flow of the ADAS function is as follows: The `rotate_coordinates_node.py` node transforms the inertial data of the INS such as accelerations and angular speeds from the `{INS}` frame to the `{Bicycle}` and finally `{Street}` frame using rotation matrices. In addition, the `{offset_position_node.py}` node virtually offsets and corrects the GNSS positions based on its mounting position and the pedelec's roll angle.

The transformed signals are then passed to the `{heading_position_calculation_node.py}` alongside with the signals from the CAM and CAN. An EKF fuses these sensors to estimate the heading of the pedelec in `{Street}` frame.

The fused heading, the transformed INS data, the car's data from the CAM and the pedelec's remaining sensors on its CAN are then passed to the `{trajectory_prediction_node.py}`

⁶Especially the CAN and HMI software components are heavily simplified and actually consist of more nodes and topics than depicted to avoid cluttering this figure further.

4. Solution Approach

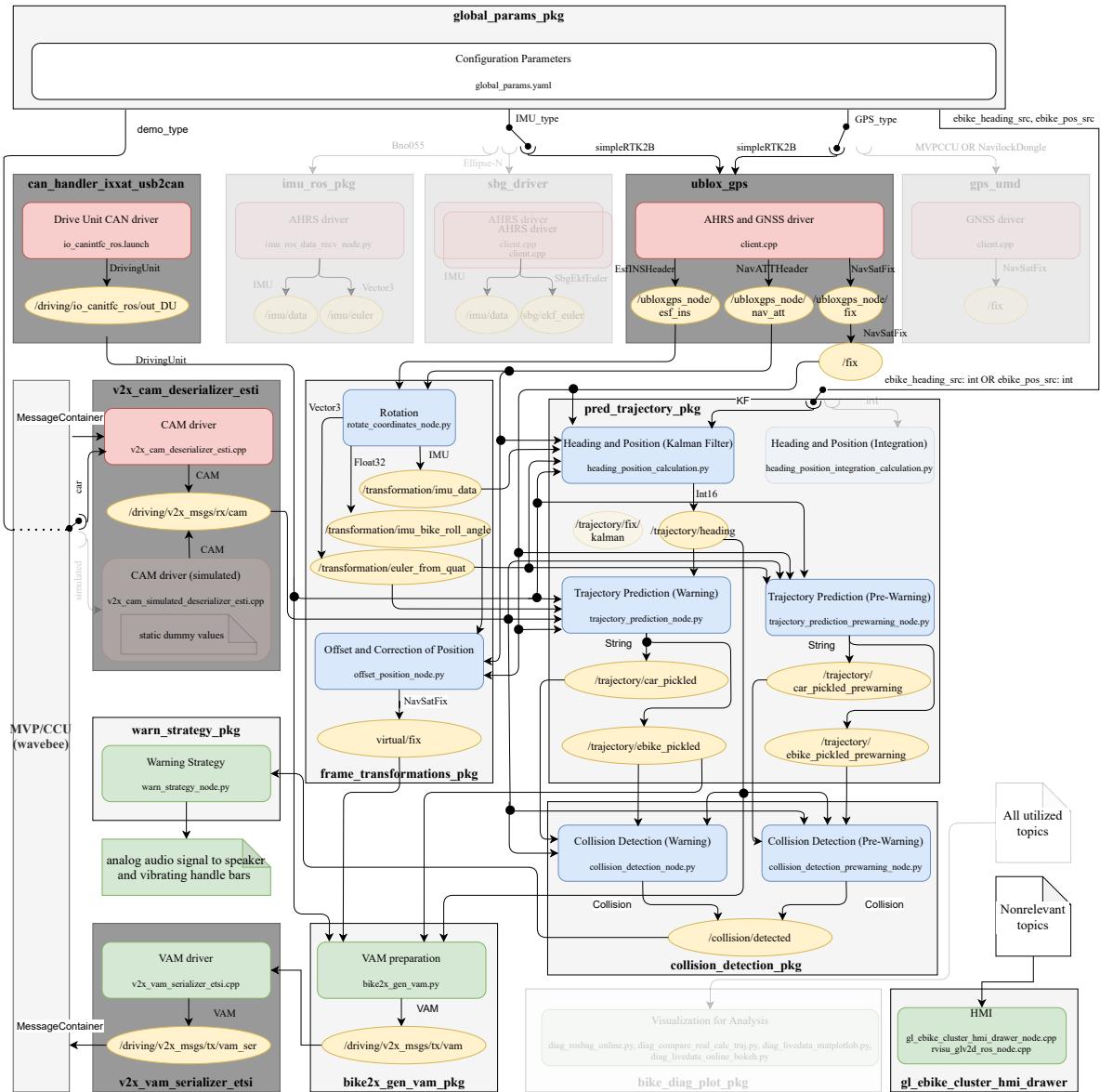


Figure 4.17.: Overview of the Software Architecture in ROS

node, which is responsible for actually predicting the future positions of both the pedelec and the car. The investigations and novel trajectory prediction models developed in this thesis will be covered in section 4.8. These high-accuracy trajectories look around 2 s into the future and are used to trigger urgent warnings e.g. an emergency brake. In parallel, longer trajectories, usually around 4 s, are predicted using different models in order to gently warn the cyclist about potentially dangerous positions. The ideas and execution behind this two-staged warning approach are discussed more detailed in subsection 4.9.4.

The predicted trajectories of the pedelec and car then get processed by the `collision-detection_node.py` node with the goal of finding out whether they will intersect at the same point in time, i.e. whether the vehicles will crash and when to warn the cyclist. To be exact, this node finds out when and where they collide using circle and polygon intersections and warns the cyclist based on the remaining distance distance to this location compared with the anticipated braking distance. section 4.9 deals with the algorithms of this split detection and warning approach. Moreover, the predicted trajectories, along with the corrected and virtually translated pedelec positions and transformed INS data, get shared with other road users through the `{bik2x_gen_vam.py}` and `{v2x_vam_serializer_etsi.cpp}` nodes.

Finally, the `{warn_strategy_node.py}` node outputs the vibration profile and beep sound to the handlebar and speaker in order to gently and urgently alert the cyclist about potentially dangerous situations and oncoming collisions.

4.3. Conducting Automated Scientific Experiments

This section tackles the downsides of the initial situation as analysed in subsection 4.1.1 with the focus of empowering the software towards the automated conduction of scientific experiments. This generally corresponds to the offline mode depicted in Figure 4.16.

Opposed to defining parameters over multiple files, all parameters are now centrally defined in a single configuration file. If attached to ROS, the native parameter server loads the parameters. Other Bash or Python scripts can simply parse the configuration file, which is in YAML. This improves the consistency and stability of the system by ensuring that related nodes which share the same parameters will actually use the same parameter values. A centralized configuration file also allows to automate experiments by iterating through pre-configured sets of parameters as done in the `automate_experiments.bash` script.

In the following, the rough steps automated by the supporting scripts `automate_experiments.bash` and `simulate_experiments.bash` are briefly explained.

The `automate_experiments.bash` writes the current parameter combination to the global configuration file and then calls the `simulate_experiments.bash` script with the corresponding flags. The `simulate_experiments.bash` script can thus also be used stand-alone by calling it with appropriate flags from the terminal.

Subsequently, the `simulate_experiments.bash` script reads the new parameters from the global configurations file, plays the pre-recorded rosbag and triggers a simulation by launch-

ing `sim.launch`. Furthermore, this script automatically stores the used parameters, a backup of the entire source code and visualization results such as plots and tables in a combined folder. Consequently, all of the results are reproducible even if the software evolved in the meantime.

Since these steps usually involve opening multiple terminals in parallel, some of the commands need to be executed as background tasks within the shell while suppressing their outputs in order to achieve parallel tasks in a single terminal. In this case, remembering the process IDs of the commands and killing them at the end is crucial to prevent "zombie" child processes. These child processes may not linked to the operating system and thus continue to run as empty shells in the background despite terminating the parent process. This is problematic as the operating system still reserves the computational resources for them. As a result, many of these "zombie" processes will either massively slow down the system or even cause it to crash.

4.4. Adoptions and Additions to Visualizations for Analysis Purposes

Subsection 4.1.2 mentions possible improvements and additions to the visualization of the signals which are explained further in the following subsections. First, a new visualization concept is introduced to save time when intending to only change the visualization itself. Then, lots of previously unused signals are visualized and analyzed regarding their usability in the trajectory prediction.

4.4.1. New Visualization Concept

As mentioned in subsection 4.1.2, the visualization concept needs to be changed such that it allows to re-parameterize or improve the actual plot script without having to re-play the entire ROS bag every time.

This is achieved by dumping the pickle file with a unique name consisting of a timestamp, trajectory prediction model and prediction horizon instead of a fixed name that gets overwritten every time. Furthermore, this pickle file along with some useful visualization options are now passed as parameters when calling the actual plotting/analysis scripts (e.g. `outputBokeh_rosbag.py`, `outputBokeh_acc_rosbag.py`, etc.), which run independently from the ROS nodes that only collects and dumps the data (e.g. `diag_rosbag_online.py`, `diag_compare_real_calc_traj.py` etc.). This allows the plotting/analysis scripts to be called stand-alone by passing the paths of pre-generated pickle files. So, if a change needs to be made to the actual visualization like displaying new signals, adding new plots etc., the user does not need to wait until a rosbag has finished replaying every time he/she wants to test the changes.

Obviously, after changing any of the actual logic of the ADAS function such as the state

4. Solution Approach

estimation, prediction models, CD etc. the entire rosbag has to be replayed to generate a new pickle file that stores all of the adapted signals in order to (visually) analyze its impacts.

In addition, the visualization script now warn the user of too many data points in order to prevent blank visualizations or extremely long loading times and recommends the user plotting parameters for reducing the number of points without having to replay the entire rosbag (e.g. `process_plot_after`, `cut_last`, `plot_every_xth`). The maximum number was determined experimentally roughly between $14.26 * 10^6$ and $15.57 * 10^6$ data points⁷.

4.4.2. Quality of Newly Added Signals

As subsection 4.1.2 notes, many potentially useful signals are not yet visualized. This section analyzes the quality of the newly visualized signals in order to validate some of the frame transformations and evaluate their usefulness for the trajectory prediction task. It also unravels a previously unknown problem with the roll angle estimation of the BNO055 and shows the necessity of either upgrading to a reference INS or estimating the roll angles from the raw signals with a more sophisticated fusion algorithm.

Figure A.2 in the appendix shows the visualization with the new signals and plots, which include angular speeds, angles and linear accelerations for all axes, wheel speeds, cadence, motor speeds, odometer, rider and motor torques as well as the motor's current and peak power output.

Most notably, the additional roll and pitch rates can be used to validate the transformation from {AHRS} into the {Street} frame. Figure 4.18 shows the yaw rates in conjunction with the latter signals for a dynamic left turn at roughly constant speed.

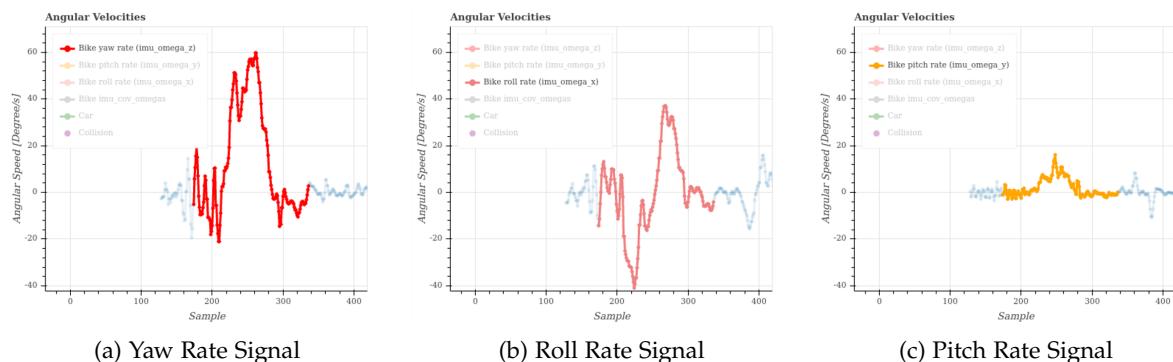


Figure 4.18.: Angular Velocities for Dynamic Left Turn Transformed to {Street} Frame

Keep in mind that the BNO055 that measured this data is mounted at an angle as shown in Figure 2.6. For this curve, the sensor experiences angular velocities about all axes in the

⁷The $14.26 * 10^6$ data points amounted to a 490.0 MB html file that took around 20 minutes to load in the Firefox browser of the NUC.

4. Solution Approach

{AHRS} frame. However, the {Street} frame only shows clear yaw rate and roll rate peaks, whereas the pitch rate remains nearly constant near zero. This makes sense as the bicycle turns about the z-axis and rolls about the x-axis of the {Street} frame but only pitches slightly about its y-axes due to suspension travel.

When analyzing the roll angle, it turns out that attitude estimation of the BNO055 does not work as expected in our pedelec use-case. Figure 4.19 shows how the the roll angle is offset significantly after a turn.

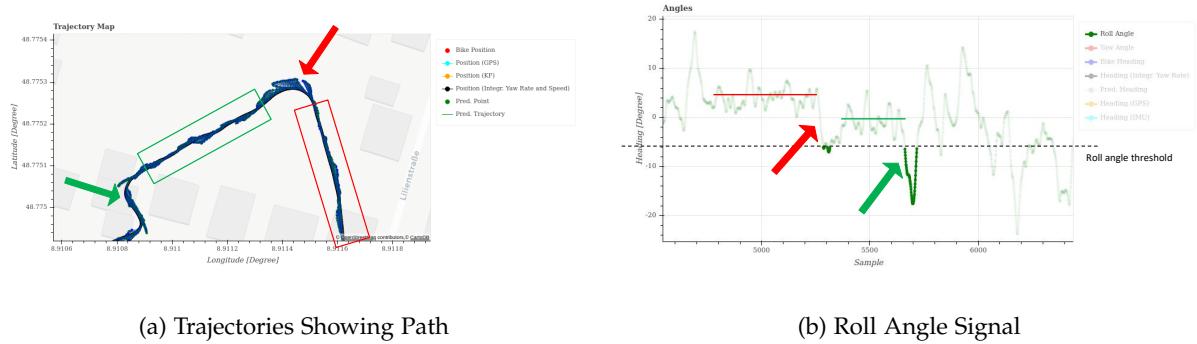


Figure 4.19.: Roll Angle Offset after Turn

Note how for the first section, marked in red on Figure 4.19a, the roll angles oscillate about 5 degrees despite the pedelec riding straight because of a previous turn. After another left turn at the red arrow, the roll angles are offset back again to around 0 degrees as seen in the green section. Note, that the roll angle returning to zero was rather a coincidence as these angle offsets are of different magnitude for nearly every curve.

The utilized BNO055 is a low-cost consumer grade AHRS which seems to be mainly designed for gaming, toys, augmented and virtual reality, robotics or industrial applications [87]. All of these applications seem to have comparatively low speeds. Perhaps, the influence of the Coriolis force as explained in Equation 2.2 is not modelled sufficiently. Potentially, either the speed signal from the integrated acceleration is too imprecise or the AHRS fusion does not even account for the influence of the Coriolis force at all. In contrast to these low dynamic applications, the pedelec can achieve high linear and angular speeds, e.g. in fast but tight curves, which may lead to a noticeable Coriolis acceleration. If the accelerometer measures a specific force f_s but the signal processing of the AHRS does not account for the Coriolis force, it would obtain a deflected value for the gravity vector (see Equation 2.1), leading to an incorrect roll angle estimation. This would explain why the accelerations and attitude look fine when only rotating the sensor at rest, as ${}^S v_S \approx 0$ leads to a negligible impact of the Coriolis acceleration on ${}^S a_{ii}$ and thus barely affecting the measured f_s . However, this hypothesis is not yet confirmed. Either way, all fusion modes of the BNO055 lead to this offset in the roll angle estimation after turning.

Consequently, a more sophisticated INS was integrated in section 4.6 to circumvent the insufficient roll angle estimation and provide a reference system for future evaluations of custom-designed fusion algorithms using the raw inertial data of a low-cost IMU to estimate the roll angle from scratch. Section 4.6 describes the integration of the reference INS further.

4.5. Virtual Vehicle for End-to-End Validation

The colleagues that work on the vehicle side of the V2V only show up in Renningen, the location of this thesis, every few months for joint testing. Furthermore, the test vehicle (pedelec) is not yet fully waterproof, which causes testing to be additionally dependent on dry weather conditions. In order to validate independently whether changes in the collision warning system of the pedelec work properly, a new node was created that simulates a virtual car. Essentially, the actual C++ CAM driver is replaced by a dummy driver, that will repeatedly transmit the same pre-configured data. This allows to configure a *virtual* car by setting the latitude, longitude, heading and dimensions in the CAM appropriately. If these parameters are set to a fixed value and the remaining ones to useful default data, it will look like a stationary car to the pedelec system, despite no car actually existing at the pre-defined location.

The virtual car allows the validate the collision warning whenever a new feature exists at an arbitrary location without having to coordinate and wait joint tests with the colleagues from Hildesheim. Because the CAM driver was set to sent static data, the pedelec still uses all of the involved ROS nodes, thus guaranteeing end-to-end validation just like the joint tests with the actual car. Figure 4.20 shows a simulated collision with a stationary virtual car conveniently placed on-site and nearby to the office.

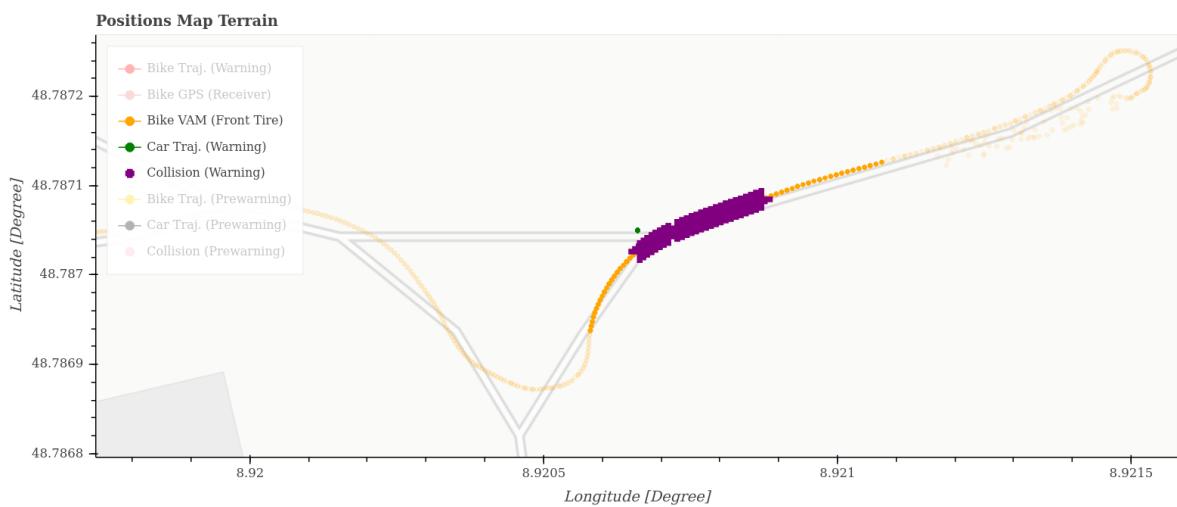


Figure 4.20.: Collision Test with Virtual Car

4. Solution Approach

Note how the even the system reaction of the pedelec gets triggered by sending the cyclist the collision warnings at the location marked with a purple cross. Of course, the signal tone and vibration profile in the handlebars is also played but not visible in this measurement.

Furthermore, the virtual car could enable testing of actual "crash" events without any participants injuring themselves. For example, a rectangle with the corresponding dimensions from the CAM could be drawn around the virtual GPS location with chalk. Scenarios could then be recorded with an additional action-camera mounted to e.g. the handle bar to analyze offline whether a "crash" occurred or not for a certain rosbag recording.

As explained in section 4.2, a parameter when calling `ride_measure_demo_ebike.bash` controls the use-case which calls a different ROS launch file. Specifically, the use-case parameter `-u` needs to have a value of either `v` (virtual car) or `c` (actual car).

4.6. Integration of Reference Sensors

As described in subsection 4.1.2, the currently integrated sensors did not suffice to properly predict or evaluate trajectories. Therefore, a high-precision reference INS was commissioned, mounted to the test vehicle and fully integrated into the ROS software.

The INS consists of a *simpleRTK2B-F9R* development board that is connected to a multi-band GNSS antenna and a shield for the Networked Transport of RTCM via Internet Protocol (NTRIP) client which is further connected to its two 4G antennas. The board itself carries the u-blox u-blox F9 high precision dead reckoning module (ZED-F9R) GNSS receiver chip. The NTRIP client holds a SIM card with a yearly subscription that accesses and downloads the RTK correction data from a server such that the ZED-F9R receiver can theoretically compute highly accurate positions with errors of a few centimeters. The necessary configuration of the SIM card and NTRIP client was performed as outlined in [88]. Furthermore, the ZED-F9R receiver features a built-in IMU to obtain positions and attitude without satellite reception via integration and Kalman-filtering (Dead-Reckoning Mode). Due to inherently noisy signals from the IMU, the integrated position will drift more from the actual position the longer the board does not receive a GPS signal.

Figure 4.21 and 4.21b shows the mounting positions and frame alignments with the new INS from the side and top view, respectively.

4. Solution Approach

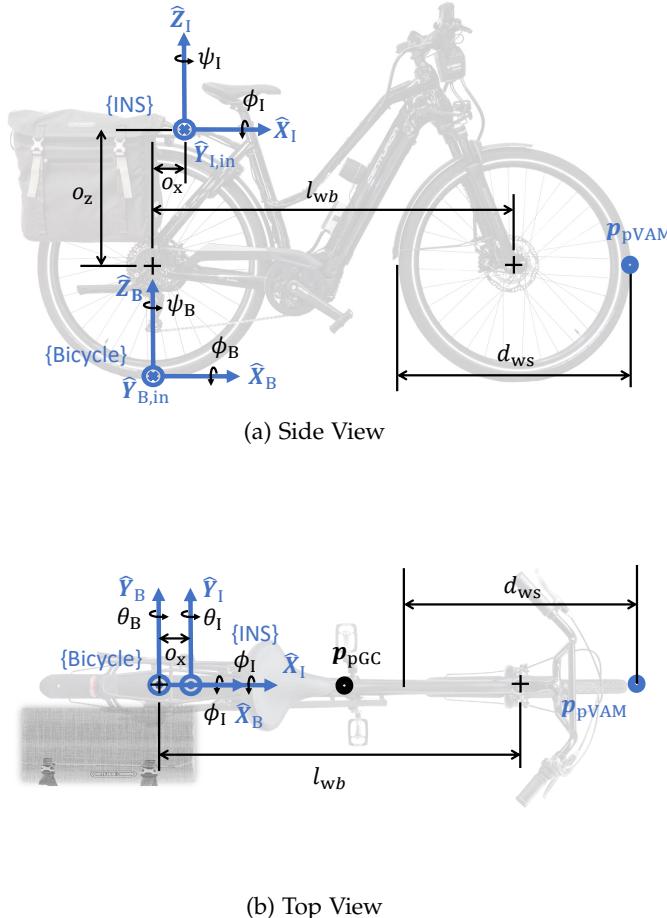


Figure 4.21.: Reference Coordinate Frames With New Sensors

Since the IMU is part of the GNSS receiver with the antenna close-by, a single frame $\{\text{INS}\}$ suffices. Note that the board is mounted horizontally on top of the luggage carrier such that the axes of the sensors and bike frame align and there only is an offset in the z-direction. This simplifies the transformation of the GNSS positions as described in section 4.7.

A Bosch internal u-blox F9R driver takes care of reading out the raw signals from the receiver chip and porting them to the ROS environment. However, this driver uses non-standard ROS message types which are not compatible with the software on the pedelec which resulted in notable integration efforts. Since the project might want to go back to experimenting with e.g. the BNO055 and estimate the attitude inside of ROS from its raw values, the users can easily switch between different IMUAHRS/INS setups in the software by setting the `IMU_type` parameter in the `global_params.yaml` file appropriately⁸.

The trends in the acceleration and yaw rate signals described in subsection 4.1.2 unfor-

⁸As long as the mounting position remains as initially configured in the software.

4. Solution Approach

tunately remain across all three integrated IMUs (Bosch BNO055 USB Stick, SGB Ellipse-N and ZED-F9R) and do not improve significantly when utilizing a more expensive system as depicted in Figure A.3 and Figure A.4. On the other hand, the similarity of the raw signals indicates that achieving better roll angles through a more sophisticated fusion algorithm seems possible. This is good news as the goal eventually is to use a low-cost IMU. In contrast, the new INS solves the roll angle issues from subsection 4.4.2. Figure 4.22 shows how the roll angle returns and oscillates about zero after riding two tight right turns, opposed to being offset by 5 degrees.



Figure 4.22.: Roll Angle Offset after Turn

The data sheet of the ZED-F9R rates its performance as shown in Table 4.1 [42]:

Parameter	Specification	Value
Max navigation update rate (RTK)	Priority navigation mode	30 Hz
Dynamic attitude accuracy	Heading	0.2 deg
	Pitch	0.3 deg
	Roll	0.5 deg
Horizontal pos. accuracy	RTK	0.01 m + 1 ppm CEP

Table 4.1.: Performance Rating of ZED-F9R

Since a high-precision WSS is not interfaced with the ZED-F9R, the *simpleRTK2B-F9R* does not quite reach these data sheets specs in our configuration. However, the positioning accuracy is still reported to be around 10-20 cm for cars when receiving correction data according to [89]. Of course, when dead-reckoning, the position accuracy decreases over time due to the sensor drift. Subsection 5.1.2 and subsection 5.2.1 qualitatively evaluate the impact of the new INS on the trajectory predictions and the overall ADAS function compared to the old setup. However, evaluating the true position accuracy of the pedelec test vehicle (INS mounted on ground plate at luggage carrier) during its dynamic operation is not a focus of this thesis. This will be covered in a follow-up thesis that deals only with accurate and reliable positioning of the pedelec in urban environments.

4.7. Offsetting and Correcting Pedelec Position

The position of the car is reported at the middle of its front bumper as defined in the CAM. For better standardization, it makes sense to transform the positions of the pedelec, shared via the VAM, from the location of its GNSS antenna to a virtual location corresponding to the first collision point of the pedelec (front tire). Furthermore, the reported position of the GNSS antenna get incorrectly projected to the street if the pedelec rolls. Not only is correcting this projection error important for CD but also for evaluating the positioning as discussed in chapter 6. This section derives a transformation that can solve both of these problems.

Figure 4.23 shows a rolling pedelec and the resulting projection error Δy_{proj} of the virtual VAM position compared to the mounting position of the GNSS.

The error in the projection of the actual GPS antenna location onto the street compared to the VAM location of front tire due to the cyclist leaning into the curve can be computed by Equation 4.10:

$$\Delta y_{\text{proj}} = (o_z - \frac{d_{\text{ws}}}{2}) \cdot \sin \phi(t) - o_y \quad (4.10)$$

Hereby, o_z defines vertical mounting offset from the origin of {Bicycle} and o_x the horizontal offset. Technically, there is an additional steering joint which amounts to another projection. However, this would complicate the fitting of a bounding box around the pedelec for CD: Only the VAM point of the front tire would see an additional curve-inward projection, while the rest of the pedelec and rider would not. So in this case, the bounding box could not use symmetry around the VAM point and would have to be dynamically shifted based on the current roll angle. Furthermore, the steer angle is currently not measured and would need to be estimated. Most importantly, the virtual location of the VAM at the first collision point is merely a proposal and not yet standardized. For example, the contact point of the pedelec's rear wheel with the street could also be useful as the virtual VAM position. In this case, the steer angle would not have a direct influence on the projection onto the street⁹ As a result, the

⁹Technically, steering ultimately always affects the path of the bike and thus also where the rear wheel is projected onto the street.

4. Solution Approach

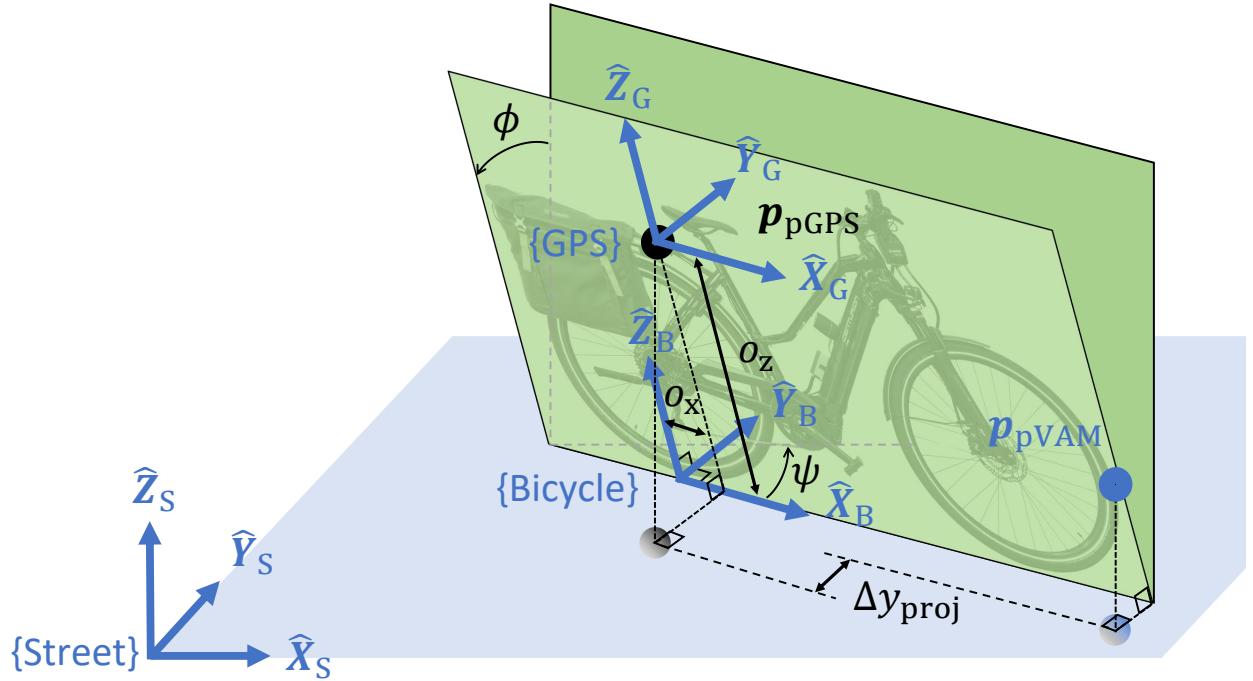


Figure 4.23.: Coordinate Frames with Bike Roll (3D view)

projection due to the steer angle is ignored for now. In the local bicycle frame, the projection error can thus be seen as a dynamic offset in y-direction that depends on time. The projection due to the rolling motion can be assumed to not contribute to any error the x-direction of the bicycle frame. In our case, the GPS antenna is mounted such that $o_z = 0.77 \text{ m}$ and $o_y = 0.00 \text{ m}$. Hence, the projection error at $\phi_{\max} = 40 \text{ deg}$ amounts up to 0.27 m. If the contact point of the rear wheel was chosen instead, this error would be up to 0.49 m.

The projected antenna GPS location always lies on the inside of a curve compared to the actually travelled path because of the roll angle of the pedelec. In some crossing scenarios, accounting for this false projection might be crucial. Figure 4.24 shows an exemplary simulation of a scenario where the system would miss a collision if the pedelec's reported positions at the location of its GPS antenna were considered. The blue rectangle which symbolizes the car does not collide with the orange rectangle which represents the projection of the pedelec onto the street centered around the GPS antenna location. However, the actual positions of the pedelec with respect to its front tire are offset by Δy_{proj} towards the outside of the curve as shown by the green path (compared to the orange path). When transforming the pedelec's reported position from the GPS antenna to a virtual positions at its front tire and thereby accounting for the projection error, the car would actually collide with the pedelec (green

4. Solution Approach

rectangle) in this situation¹⁰.

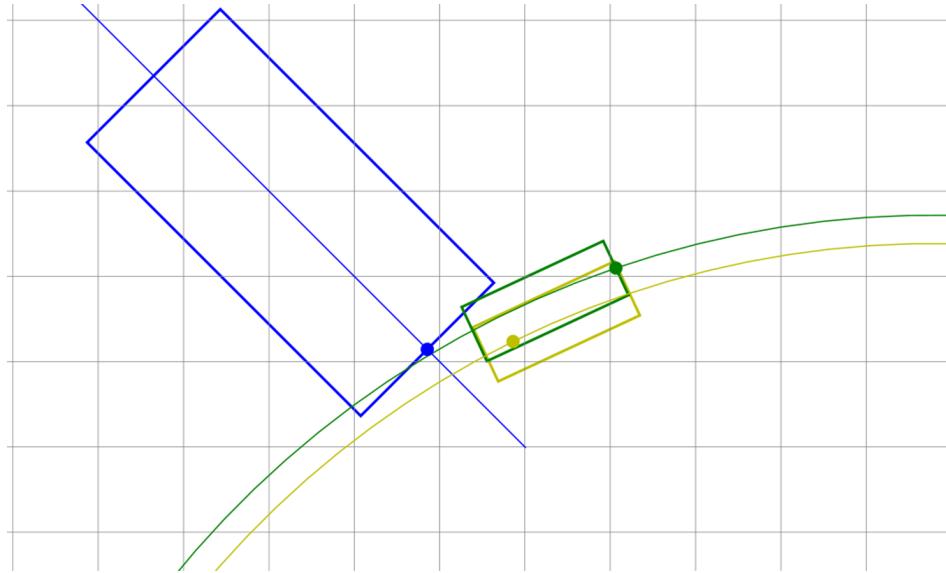


Figure 4.24.: Ignoring Roll Angle Correction leading to FN

The following paragraphs derive the transformation for virtually offsetting the pedelec position.

The virtual VAM position at the first collision point (front tire) can easily be expressed in $\{\text{Bicycle}\}$, as denoted by ${}^B\mathbf{p}_{\text{VAM}}$ in Equation 4.11

$${}^B\mathbf{p}_{\text{VAM}} = \begin{bmatrix} l_{\text{wb}} + \frac{d_{\text{ws}}}{2} \\ 0 \\ \frac{d_{\text{ws}}}{2} \\ 1 \end{bmatrix} \quad (4.11)$$

where l_{wb} denotes the length of the pedelec's wheel base, and d_{ws} the diameter of the pedelec's wheel (wheel size). Note that this position vector ${}^B\mathbf{p}_{\text{VAM}}$ is actually four-dimensional (4D): three coordinates describe the position in space from the origin of $\{\text{Bicycle}\}$ while the last coordinate (one) extends the vector to the fourth dimension to mathematically allow for a translation followed by a rotation of the vector with a single homogeneous transformation matrix. However, a coordinate frame transformation is necessary to obtain the absolute coordinates in $\{\text{Street}\}$ from the given coordinates relative to $\{\text{Bicycle}\}$ while accounting for the mounting offset of the GNSS and the yaw and roll motion of the pedelec as described

¹⁰In this case, transforming to the contact point of the pedelec's rear wheel with the street would show the collision even sooner.

4. Solution Approach

mathematically in Equation 4.12 by using a homogeneous transformation.

$${}^S \mathbf{p}_{VAM} = {}^S_B \mathbf{T} \cdot {}^B \mathbf{p}_{VAM} \quad (4.12)$$

$${}^S \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{VAM} = {}^S_B \mathbf{T} \cdot {}^B \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{VAM} \quad (4.13)$$

The vector ${}^B \mathbf{p}_{VAM}$ denotes the given position of the front tire for the VAM relative to {Bicycle} while ${}^S \mathbf{p}_{VAM}$ denotes the corresponding target position in {Street}. As described in Equation 4.11, both position vectors ${}^B \mathbf{p}_{VAM}$ and ${}^S \mathbf{p}_{VAM}$ are in homogeneous coordinates. The transformation from {Bicycle} to {Street} can be further broken down into transforming first from {Bicycle} to {GPS} and then from {GPS} to {Street} as shown in Equation 4.14.

$${}^S_B \mathbf{T} = {}^S_G \mathbf{T} \cdot {}^G_B \mathbf{T} \quad (4.14)$$

The first transformation ${}^G_B \mathbf{T}$ is simple and achieved via translation of the negated GNSS mounting offsets, which are originally defined in {Bicycle}. As shown in Figure 4.23, the origin of {Bicycle} can be described as $[-o_x, 0, -o_z]^T$ in {GPS} coordinates. Since the axes of {Bicycle} and {GPS} are aligned as shown in Figure 4.23, no rotation is needed in this case¹¹. Equation 4.15 shows the final homogeneous transformation matrix from {Bicycle} to {Street}.

$${}^G_B \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -o_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.15)$$

Transforming from {GPS} to {Street} is more complicated as shown in Figure 4.23. In addition to translating by the GPS coordinates, the frames get rotated by ψ about the z-axis, then additionally rotated by ϕ about the x-axis because of the bicycles yaw and roll motions, respectively, when cornering. This can be achieved by matrix multiplication of \mathbf{R}_z with \mathbf{R}_x or using the generic rotation matrix shown in section A.1 and setting $\theta = 0$. The homogeneous transformation matrix from {GPS} to {Street} thus results in Equation 4.16.

$${}^G_S \mathbf{T} = \begin{bmatrix} \cos \psi & -\sin \psi \cos \phi & \sin \psi \sin \phi & x_{GPS} \\ \sin \psi & \cos \psi \cos \phi & -\cos \psi \sin \phi & y_{GPS} \\ 0 & \sin \phi & \cos \phi & z_{GPS} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.16)$$

After plugging in Equation 4.11, Equation 4.15 and Equation 4.16 into Equation 4.12, the formula for transforming the virtual VAM position defined in {Bicycle} to the global

¹¹If the GNSS was mounted at an angle, the identity matrix would need to be replaced with the corresponding rotation matrix about the mount angle

position in {Street} by accounting for the GNSS mounting offset and the pedelec's yaw and roll motion results in Equation 4.17:

$${}^S \mathbf{P}_{\text{VAM}} = \begin{bmatrix} \cos \psi & -\sin \psi \cos \phi & \sin \psi \sin \phi & x_{\text{GPS}} \\ \sin \psi & \cos \psi \cos \phi & -\cos \psi \sin \phi & y_{\text{GPS}} \\ 0 & \sin \phi & \cos \phi & z_{\text{GPS}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -o_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_{\text{wb}} + \frac{d_{\text{ws}}}{2} \\ 0 \\ \frac{d_{\text{ws}}}{2} \\ 1 \end{bmatrix} \quad (4.17)$$

Since the utilized *NumPy* package multiplies matrices computationally efficiently, Equation 4.17 is not further derived analytically (even though it could be) [90]. Obviously, the formula is abstracted into a function that accepts any {Bicycle} coordinates rather than the static vector on the right-hand side describing the virtual VAM location.

Despite reporting the corrected virtual location via the VAM to other road users as described in section 4.10, the software uses the GNSS antenna position internally for the following trajectory prediction and CD. That way the same transformation rule as derived in Equation 4.17 can be used to obtain the positions of the rectangle's vertices that represent the pedelec. Section 4.9 describes the polygon overlap approach for detecting collisions in more detail.

4.8. Development of Physics-based Trajectory Prediction Models

This section concerns with the development of new physics-based trajectory prediction models in order to improve the prediction accuracy as one way of achieving a lower FP rate. First, subsection 4.8.1 explains why segmenting a trip into different riding situations may be useful for evaluating the models better. Then, subsection 4.8.2 introduces the intuition behind using a multi-model approach. subsection 4.8.3 analyzed the longitudinal component regarding what sensor signals may be useful for the prediction and what prediction method should be used. Afterwards, subsection 4.8.4 answers the same questions for the lateral component with a focus on investigating how curves entries are initiated on pedelecs, how they can be detected reliably and what signals may be useful to use for predicting an entire curve. Hereby, a novel trapezoidal extrapolation logic is presented.

The implementation of the physics-based trajectory prediction models discussed in the next subsections is based on an abstract class that provides a blueprint for a longitudinal and lateral component, that can be combined with different signals and prediction methods. In addition, models can easily be inherited and slightly adapted. The models follow a naming convention based on the signal and prediction method of the longitudinal and lateral components as introduced in chapter .

4.8.1. Segmenting Trip into Different Riding Situations

As noted in subsection 4.1.4, evaluating the errors of the trajectory prediction models averaged over an entire trip might favor models whose assumptions fit best to the largest percentage of

4. Solution Approach

the trip. This subsection thus investigates how a trip can be segmented into useful riding sections to compare the performance of the trajectory prediction models more truthfully.

First of all, a cyclist generally rides straight most of the time. Straight segments seem to outweigh curve segments even in urban trips e.g. through a tightly spaced neighborhood in Malsmehim as depicted in Figure 4.25. From the depicted section of the trip it seems



Figure 4.25.: Comparison of straight and curved segments for exemplary urban riding

like urban rides could approximated roughly with predominantly straights and some turns, whereas long and smooth curves are comparatively rare.

As a result of this, it seems plausible that a primitive model, e.g. holding speed and heading constant for the entire prediction horizon, might outperform more complex models simply because it does a great job at predicting straights, which generally make up the largest section of a trip. However, the trajectory prediction in curves would be significantly worse and may not be usable for the ADAS function. Therefore, it makes sense to segment the trip at least into straight segments and curves and evaluate the models over these sections additionally. The next paragraph explains how the track is segmented into curves and straights.

There does not exist a clear definition of what is counted as a curve or when a curve begins and ends. In the context of this thesis, a curve is defined through the average rate of change in the heading signal as it provides a large SNR where curves are clearly visible in the signal. As described in subsection 4.1.2, other signal like the yaw rate or lateral acceleration oscillate or contain noise that are non-trivial to filter out without which makes them hard to apply any logic to reliably such as thresh-holding or peak detection. The algorithm, which only works offline, segments the ride into straights and curves as follows:

First, the raw heading signal is split at its periodicities, where the heading crosses 360 degree and starts at 0 degree, or vice versa. Then, in between these segments, the heading is smoothed using a Savitzky-Golay-Filter. Filtering is necessary to remove mini-peaks due to

4. Solution Approach

noise in the heading signal. If the entire heading was smoothed instead, the periodicity jumps would be overdamped and change the entire heading signal. Then, a peak detector finds the local minimizers and maximizers within each segment that represent a change of direction. Finally, the algorithms computes the heading delta between every extremizers pair. If the heading delta between these direction changes $\Delta\psi_{\min/\max}(s)$ is larger than an arbitrary threshold, e.g. 25 degrees, it counts as a curve. If it is below that threshold, it counts as a straight segment. For every segment s , the condition in Equation 4.18 is thus checked:

$$\text{segment}(s) = \begin{cases} \text{straight}, & \text{if } \Delta\psi_{\min/\max}(s) < 25 \text{ deg.} \\ \text{curve}, & \text{if } \Delta\psi_{\min/\max}(s) \geq 25 \text{ deg.} \end{cases} \quad (4.18)$$

The threshold of 25 degrees was chosen after obtaining a heading delta statistic over all segments as shown in Figure 4.26: Figure 4.26

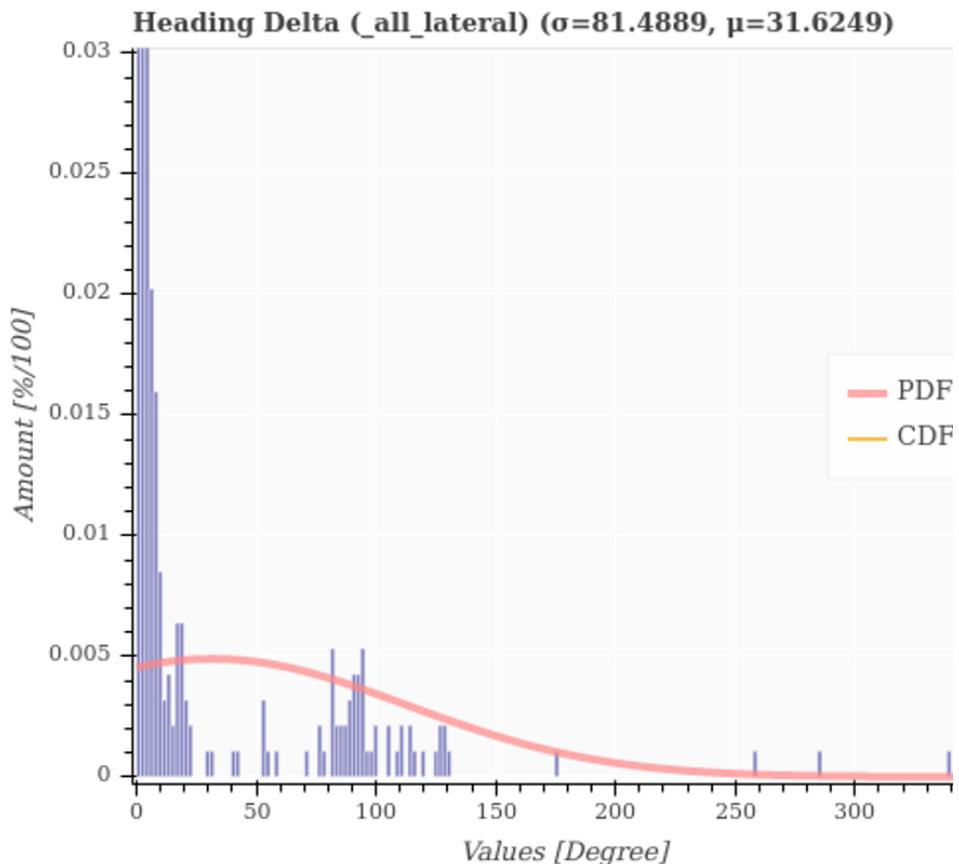


Figure 4.26.: Comparison of heading difference between various riding sections in an urban environment

The histogram clearly shows that the largest number of segments contain heading deltas of below approximately 25 deg. It also shows that the majority of curve segments exhibit a

4. Solution Approach

heading delta of about 90 degrees. As depicted in Figure 4.25, the segmentation algorithm works fairly well.

Additionally, in subsection 4.1.2, the impact of the pedalling was discussed and subsubsection 3.2.3.3 investigated the stability of a bicycle in certain speed ranges. Therefore, it makes sense to also segment the ride into speed ranges and whether the rider was pedaling or not at every time step t as described in Equation 4.19:

$$segment(t) = \begin{cases} unstablePedalling, & \text{if } 1 < v(t) \leq 17.4 \frac{\text{km}}{\text{h}} \wedge \tau(t) > 0.2 \text{ N m.} \\ stablePedalling, & \text{if } v(t) > 25.0 \frac{\text{km}}{\text{h}} \wedge \tau(t) > 0.2 \text{ N m.} \\ slightlyUnstablePedalling, & \text{if } 1 < v(t) \leq 17.4 \frac{\text{km}}{\text{h}} \wedge \tau(t) < 0.2 \text{ N m.} \\ unstableCoasting, & \text{if } 1 < v(t) \leq 17.4 \frac{\text{km}}{\text{h}} \wedge \tau(t) < 0.2 \text{ N m.} \\ stableCoasting, & \text{if } 17.4 \frac{\text{km}}{\text{h}} < v(t) \leq 25.0 \frac{\text{km}}{\text{h}} \wedge \tau(t) < 0.2 \text{ N m.} \\ slightlyUnstableCoasting, & \text{if } v(t) > 25.0 \frac{\text{km}}{\text{h}} \wedge \tau(t) < 0.2 \text{ N m.} \end{cases} \quad (4.19)$$

The variable $v(t)$ represents the speed at time step t and $\tau(t)$ the rider's torque at that time step. As indicated by the research in subsubsection 3.2.3.3, there exists a calculator that computes the weave and capsize speeds of a specific bicycle geometry. However, since these stability analysis do not account for mass of the rider, they are chosen rather arbitrarily to $v_{\text{weave}} = 17.4 \frac{\text{km}}{\text{h}}$ and $v_{\text{capsize}} = 25.0 \frac{\text{km}}{\text{h}}$. These values roughly align with the bicycles used in the research of [72]. Additionally, $25.0 \frac{\text{km}}{\text{h}}$ marks the cut-off speed at which the electric assist is faded out, and the cyclist has to pedal harder again.

Essentially, the program takes the speed and torque signals stored in NumPy arrays and determines the indeces at which a single threshold condition, e.g. $v > 25.0 \frac{\text{km}}{\text{h}}$ is met. Then, the indeces corresponding to another single conditions is evaluated, e.g. $\tau(t) < 0.2 \text{ N m}$. The riding situation *slightlyUnstableCoasting* is then obtained via the intersection \wedge of those indeces. This allows to construct arbitrary complex scenarios from basic tests that do not have to be recomputed.

While there exist many more riding situations that may show some distinct signal characteristics, the complexity in detecting all of these conditions reliably online and switching between dozens of models quickly becomes overwhelming and computationally expensive. Therefore, this thesis mostly focuses on the discussed straight and curved segments as well as the riding speed ranges during coasting and pedalling.

4.8.2. Multi-Model Approach

The trajectory prediction based on a single physics-based model is greatly limited to its underlying assumptions about the riding situation. As seen in subsection 4.1.6, the `const_a_yawr` model only performs well near the apex of curves where the yaw rate can be momentarily approximated fairly well by a constant yaw rate. On straights, curve entries and curve exits, the constant yaw rate assumptions are violated too much, thus leading to swinging,

overshooting, undershooting trajectories, respectively. Therefore, Hypothesis 4.8.2.1 arises:

Hypothesis 4.8.2.1 *Different riding situations favor different types of physics-based prediction models with different underlying assumptions.*

This hypothesis indicates that a multi-model approach that recognized or predicts certain riding situations, e.g. those introduced in subsection 4.8.1, and intelligently switches between specialized models for them should outperform a single model. The trapezoidal extrapolation logic explained in subsubsection 4.8.4.5 implements a very simplistic switching mechanism between different models for predicting the lateral component during curves or turns.

4.8.3. Longitudinal Component

This subsection concerns about the longitudinal component of physics-based trajectory prediction models. First, subsubsection 4.8.3.1 introduces a polynomial extrapolation method to predict future values based on previous ones. Then, subsubsection 4.8.3.2 investigates what signals and prediction methods are available and useful for the prediction of the longitudinal component. The results and analysis of the corresponding experiments are covered in subsection 5.1.4.

4.8.3.1. Polynomial Extrapolation

This sub-subsection describes an extrapolation method used to predict future values from multiple past values, opposed to considering only a single instantaneous value that might be affected severely by noise. It also shows why limiting the extrapolation may be necessary.

The extrapolation method is explained along with a visualization in Figure 4.27: First, a rolling buffer collects and holds a number of recent samples n_{in} (marked in blue), defined by the input duration t_{in_long} as described in Equation 4.20

$$n_{in} = t_{in_long} \cdot f_{node} \quad (4.20)$$

where f_{node} denotes the node frequency at which the data is sampled. In Equation 4.20, $t_{in_long} = 0.5\text{ s}$ and $f_{node} = 20\text{ Hz}$ which means the last 10 samples are always kept. Then, a polynomial of a certain polynomial degree n_{pg_long} is fitted through these points using Least-Squares (marked in orange). In Equation 4.20, $n_{pg_long} = 1$ and thus fits a line. The algorithm then extrapolates future values along this fitted polynomial (marked in green) until the end of the prediction horizon t_{pred} as defined in Equation 4.21:

$$n_{pred} = t_{pred_long} \cdot f_{node} \quad (4.21)$$

where n_{pred} denotes the number of predicted samples. In Equation 4.20, $t_{pred} = 4.0\text{ s}$ and $f_{node} = 20\text{ Hz}$ which means the next 80 samples are extrapolated.

For longer prediction horizons or high polynomial degrees and short input durations, it

4. Solution Approach

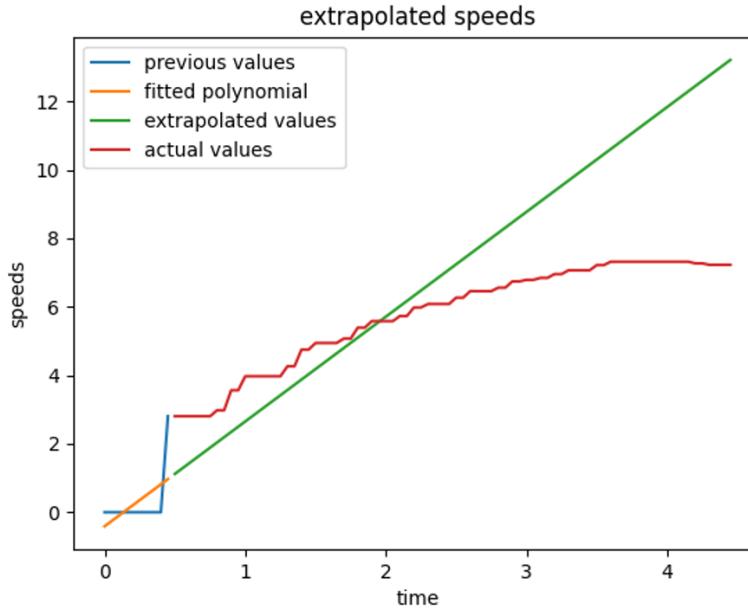


Figure 4.27.: Linear extrapolation of the speed signal

becomes crucial to limit the extrapolated signals as they may reach physically impossible values. Figure 4.28a shows an unlimited speed extrapolation for $t_{\text{pred}} = 2 \text{ s}$ with $t_{\text{in}} = 0.25 \text{ s}$, $n_{\text{deg}} = 3$ reaching almost $10.000 \frac{\text{km}}{\text{h}}$ in magnitude.

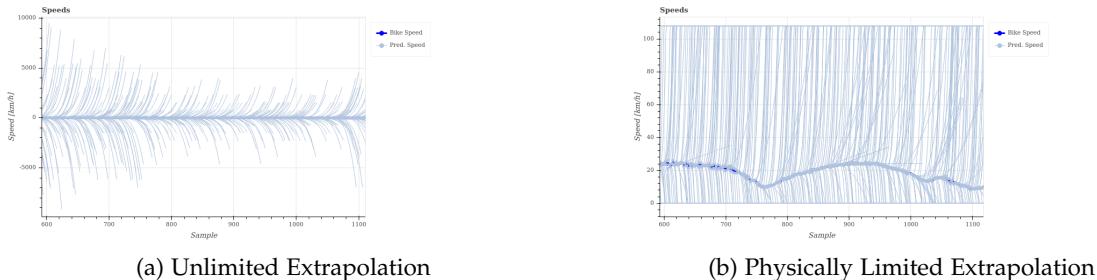


Figure 4.28.: Overview of Two Routes through Malmsheim

While it may be hard to limit the maximum speed, the minimum speed should not become less than zero. This is because the WSS only measures the speed that consists of only a magnitude, compared to the velocity that is a vector consisting of magnitude and direction. These negative speeds would lead to physically illogical trajectories as shown exemplary in Figure 4.29. Figure 4.29a shows all trajectories from an unlimited extrapolated speed model where the rider brakes deep into a slow turn. While most trajectories follow this curve, there exist some physically illogical ones that abruptly go back into the opposite direction as shown in Figure 4.29b, where the red dot represents the initial position of the pedelec, and the green

4. Solution Approach

dot represents the predicted final position.

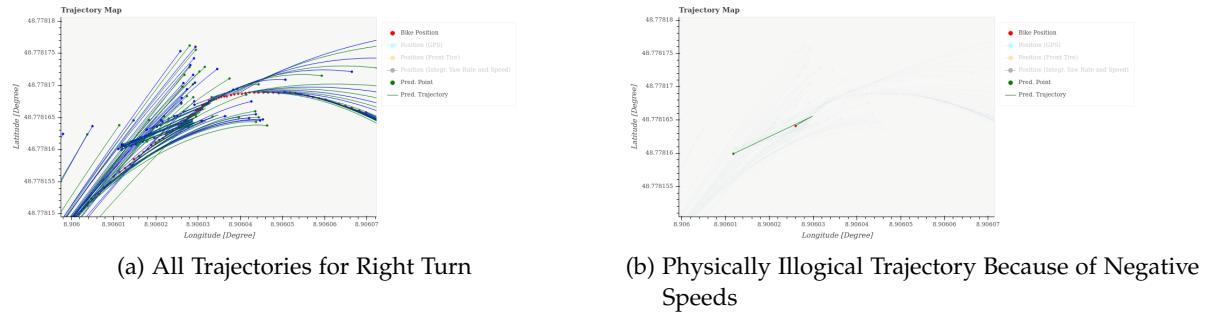


Figure 4.29.: Physically Illogical Trajectories

Such illogical trajectories are safety critical as the car would think the pedelec moves "back out of the way" while in reality it would approach the car further. Figure 4.28b shows the same speed signal from Figure 4.28a but limited between $0.0 \frac{\text{m}}{\text{s}} \leq v \leq 30.0 \frac{\text{m}}{\text{s}}$. These boundaries represent the physical limits of the factory pedelec. The upper bound of $30.0 \frac{\text{m}}{\text{s}}$ is set arbitrarily and assumes the pedelec is riding on paved streets without being towed or riding directly behind a motorized vehicle¹².

The extrapolation method of the longitudinal component allows to freely set the parameters $t_{\text{in}_{\text{long}}}$, $n_{\text{pg}_{\text{long}}}$ and t_{pred} as well as replacing the shown speed signal from the HMI WSS with other longitudinal signals such as the acceleration or even lateral signals. Furthermore, the extrapolation of the speed signal is physically limited.

Subsection 5.1.4.2 investigates the influence of these parameters as well as the limitation of the extrapolated speed on the trajectory prediction errors of the models in detail.

4.8.3.2. Selection of Sensor Signal for Prediction of Longitudinal Component

This sub-subsection concerns about which combination of sensor signal and prediction method may be most suitable for the longitudinal component of the trajectory prediction models.

Generally speaking, an algorithm can barely distinguish whether the rider only harshly breaks into a curve or comes to completely stop only on the basis of naturalistic data. In other words, the algorithm does not know when to stop extrapolation or integrating. The characteristics of the brake pressure, acceleration, speed, etc. signals might look very similar in the first few moments for both of the cases and because of the lacking perception or mapping the bike can hardly infer anything. To make the prediction of the longitudinal signal even harder, emergency brakes may occur "out of the blue" at any given time during the

¹²Race prepped bicycles with different gear ratios may physically exceed this speed if ridden e.g. down a very steep snow mountain/volcano or in the absence of drag as done in speed records.

4. Solution Approach

prediction horizon. Figure 4.30 shows how brake pressure builds up in just a few milliseconds (lower graph) and the pedelec decelerates in roughly 60 ms after brake pressure build-up (upper graph). These reasons make it very challenging to build a useful physics-based brake

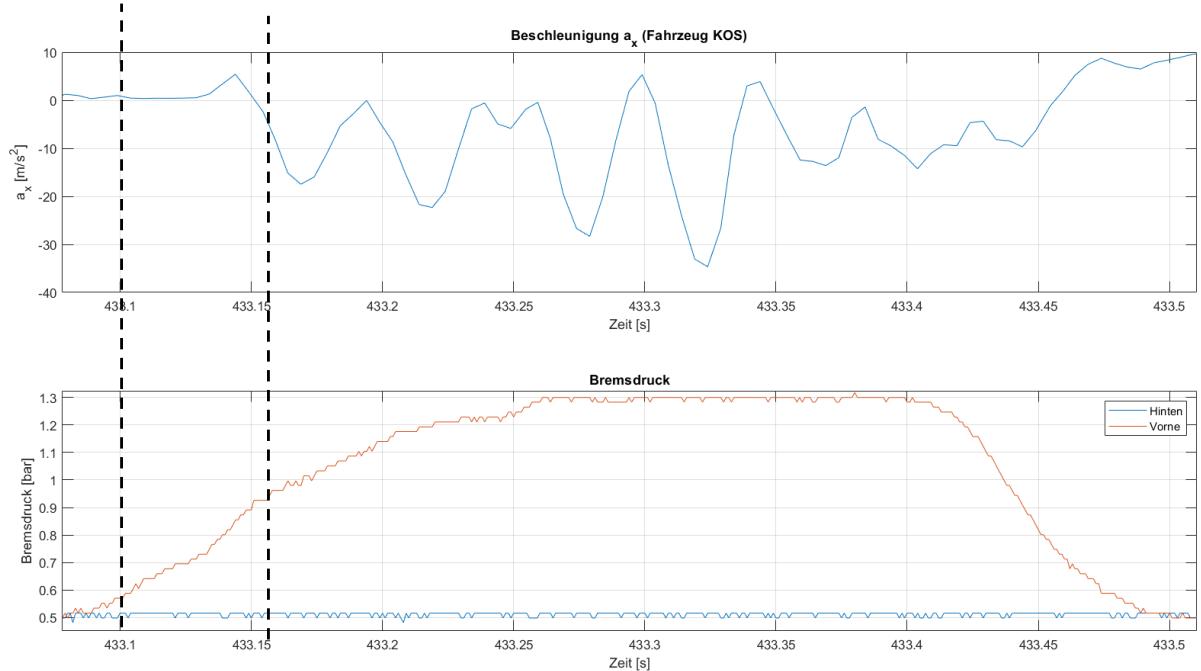


Figure 4.30.: Pedelec's response compared to brake pressure

detection given only naturalistic data.

Out of the available naturalistic data, two types of signals are useful to predict the longitudinal component: the speed v either from the ABS or HMI WSS and the longitudinal acceleration a_x from the additional AHRS or build-in IMU. Similarly, two prediction methods are considered for each of these signals: constant assumption or polynomial extrapolation as explained in subsubsection 4.8.3.1. This results in four possible combinations, which are now discussed regarding the usability.

As analyzed in subsection 4.1.2, the a_x is extremely noisy and filtering it would be non-trivial and likely introduce an undesired delay and overdamping. Since filtering is not attempted, these noisy acceleration do not provide a good basis for holding it constant over the prediction horizon. Extrapolating the acceleration would likely amplify this effect, especially for short $t_{in, long}$ and high $n_{pg, long}$ as the positional error with respect to time would increase further from quadratic to cubic, quartic, etc. In subsection 4.1.2 it was also noted that the speed from the standard HMI WSS results in undesired steps at low speeds. This would be problematic for both holding an instantaneous value constant as well as extrapolation when using short $t_{in, long}$. Using the high-resolution ABS WSS that shows a much smoother signal, especially

at low speeds, should deliver better results for both methods. The evaluations in subsubsection 5.1.4.1 demonstrate the importance of a smooth speed signal. A linear extrapolation of the speed is expected to show superior performance compared to higher orders as the positional error will grow exponentially with $n_{pg, long}$ as explained in subsubsection 4.1.6.2. For these reasons, a linearly extrapolated speed signal of the ABS WSS is expected to provide the best overall prediction performance for the longitudinal component. However, holding the ABS speed constant might be interesting for long time horizons as the positional error with respect to time stays linear in the case. The influence of the selected longitudinal signal and prediction method on the trajectory prediction accuracy is evaluated detailed in subsection 5.1.4.

4.8.4. Lateral Component

This subsection concerns about the lateral component of physics-based trajectory prediction models. First, subsubsection 4.8.4.1 examines the counter-steer and curve initiation behavior of cyclists. Then, subsubsection 4.8.4.2 investigates the usability of signals for the prediction of the lateral component with the focus of predicting curves. It answers question such as what signals can be used to detect curve entries and exists early and reliably and how usable are their signal characteristics for predictions. Afterwards, subsubsection 4.8.4.3 derives a relationship to compute the yaw rate from the roll angle. Consequently, subsubsection 4.8.4.4 validates this relationship against the measured yaw rates. Finally, subsubsection 4.8.4.5 explains a novel prediction method that extrapolates the roll angle along a trapezoid in order to predict curves. The results and analysis of the corresponding experiments are covered in subsection 5.1.5.

4.8.4.1. Initiating Curve Entries Through Weight Transfer

This sub-subsection examines the counter-steer and curve initiation behavior of cyclists to understand whether there exist any early indicators in the signals that can be used to recognize a curve in advance.

Figure 4.31 compares the measured steering angle against the yaw rate for an exemplary curve. Note that the steering angle is measured explicitly using an encoder sensor mounted to another pedelec. This pedelec also uses a high quality AHRS to obtain the yaw rates. The top graph, depicting the steering angle, does not show an opposite steering input before entering a curve. The counter-steer before curve entry is thus not visible at all in this measurement. Similar behavior was observed on other measurements as well.

Furthermore, the steering angle does not lead the yaw rate signal and both signals rise simultaneously when entering the curve. This indicates that pedelec riders tend to initiate the roll motion into curves mostly by weight-transfer to shift the center of gravity inward the curve, opposed to twisting the handle bar into the opposite direction to tilt the pedelec into the curve. This finding matches up with research results, which report that a curve entry can be initiated by weight transfer only for bicycles, opposed to motorcycles, where

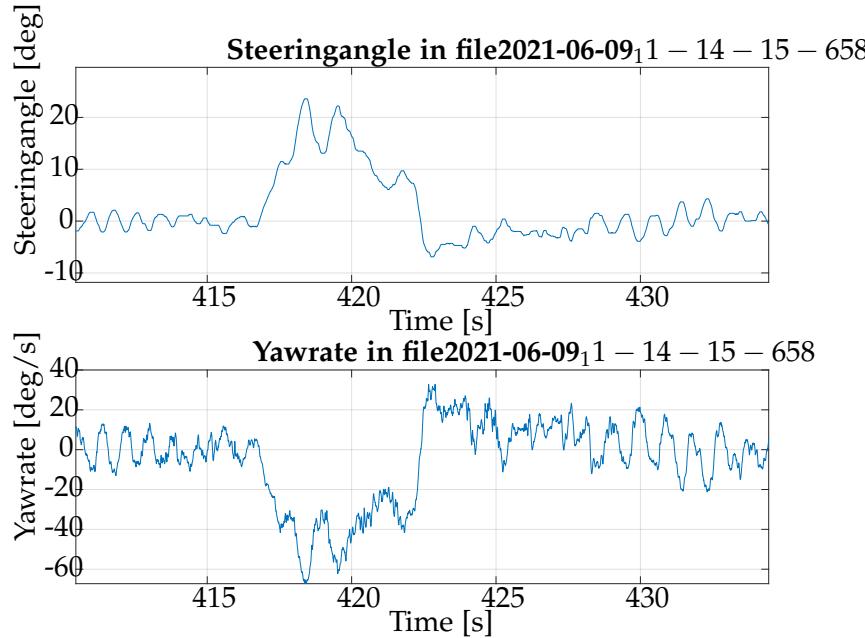


Figure 4.31.: Steer angles and yaw rates for a curve

counter-steer is almost always necessary.

To summarize, pedelec riders seem initiate a curve entry through weight transfer rather than counter-steering which means there does not seem to exist a clear early indicator in the naturalistic data to mark the beginning of a curve which was expected previously.

4.8.4.2. Appropriate Signal for Detecting and Predicting Curves

This sub-subsection investigates which signals are appropriate for detecting curve entries and exits early and reliably as well as their potential for predicting curves with the two prediction methods constant and extrapolation. Moreover, this section also briefly covers what lateral signals are useful for predicting straight segments.

Out of the available naturalistic data, five types of signals are generally considerable for predicting the lateral component: the yaw rate $\dot{\omega}_z$ from the INS, the heading ψ from the INS or EKF-based heading estimation, the lateral acceleration a_y from the INS or build-in IMU, the roll rate $\dot{\omega}_x$ from the INS and the roll angle ϕ from the INS. Similarly, two prediction methods are considered for each of these signals: the constant assumption or polynomial extrapolation as explained in subsubsection 4.8.3.1. This results in ten possible combinations, which are now discussed regarding their usability for predicting the lateral component in curves and straights.

Similar to the longitudinal accelerations, the lateral accelerations of the pedelec are ex-

4. Solution Approach

tremely noisy. As already explained in subsection 4.1.2, filtering out the noise would make a detection more reliable but may over-damp the usable signal or introduce an undesired delay when processes online. Figure 4.32 depicts a low-pass filtered a_y using a running mean filter with a window size of 31 samples.

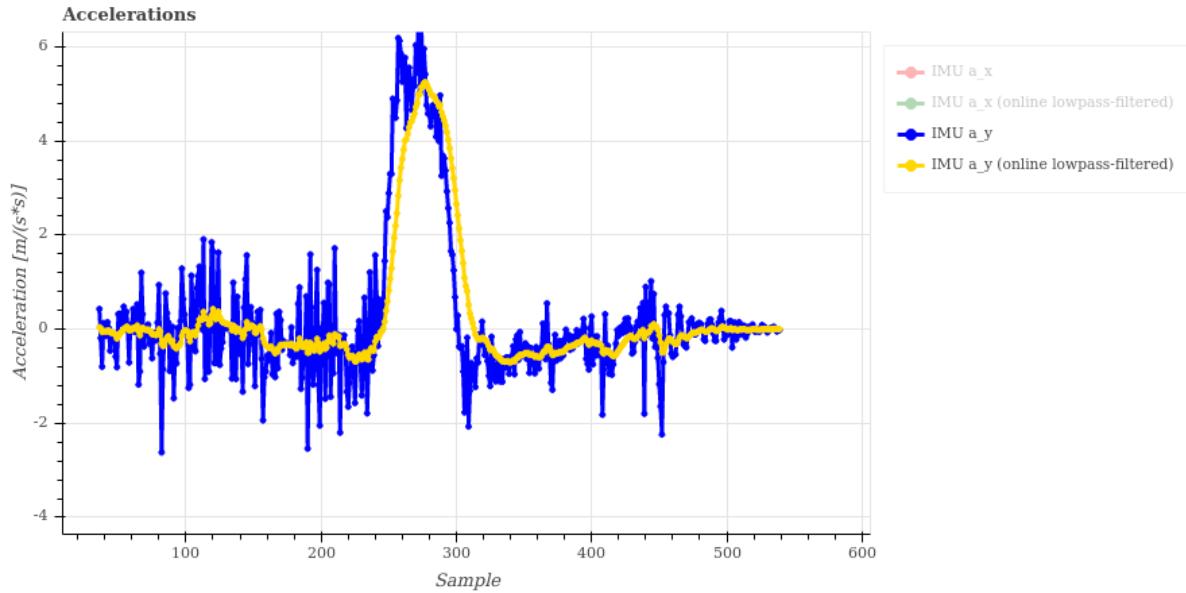


Figure 4.32.: Running Mean-Filtered Lateral Acceleration

Note how the signal is delayed in the curve by up to 15 samples as given by Equation 4.1. This is undesirable as it would mean the curve is essentially detected 0.75 s later. Using a shorter window size would decrease the group delay but may not provide significant smoothing anymore and thus lead to an unreliable detection. Because of its extreme noise and non-trivial filtering, the lateral acceleration a_y is therefore generally an unsuitable signal for both curves and straights using either the constant or extrapolation method.

Similarly, the yaw rates $\dot{\omega}_z$ oscillate in an "unpredictable" way, especially at low speeds, as explained in subsection 4.1.2 which cannot be filtered out in a trivial way. Analogous to a_y , $\dot{\omega}_z$ is not ideally suited for detecting or predicting curves with either the constant or extrapolation method. Moreover, the heavy oscillations in $\dot{\omega}_z$ make it a poor signal choice for predicting the lateral component of straight road segments. The same applies for the steering angle which can be derived from the yaw rate as shown in Equation 3.5 and thus shares very similar signal characteristics with the yaw rate as shown in Figure 4.31. Moreover, the steering angle is not measured in our test pedelec or any factory pedelecs.

The roll rate $\dot{\omega}_x$ and roll angle ϕ might be better suited for detecting curves and predicting them. Figure 4.33 shows the roll rate, yaw rate, roll angle and heading signals for a left turn, starting from East as shown in Figure 4.33c.

4. Solution Approach

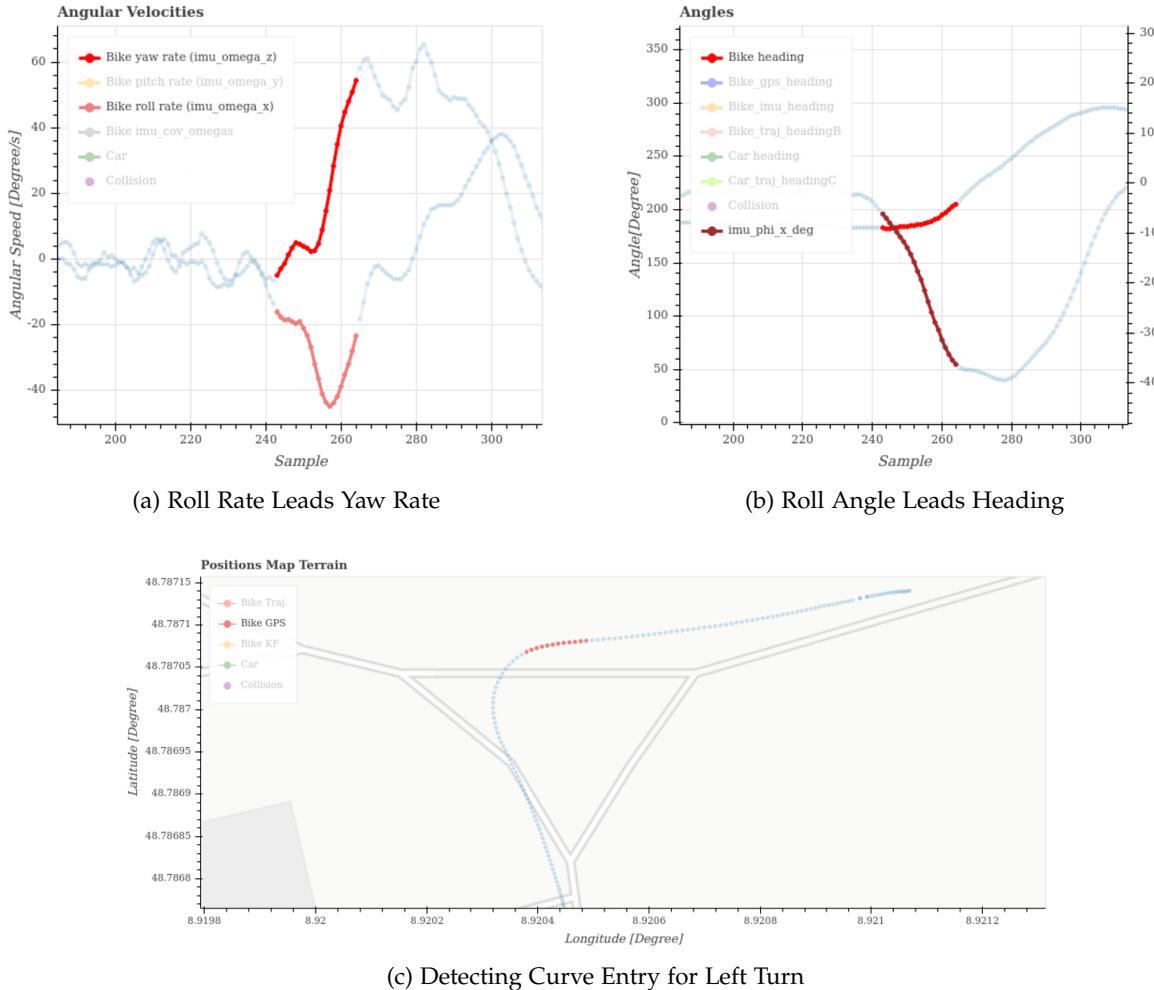


Figure 4.33.: Lateral Signals for Left Turn (highlighted curve entry)

The roll rate builds up first and leads the yaw rate as shown in Figure 4.33a. In fact, it seems that at the peak of the roll rate, the yaw rate increases noticeably. The roll rate is thus the first lateral signal to notice at a curve entry. However, its signal characteristics are comparatively complex to the yaw rate and therefore not well suited for prediction. The roll angle leads the heading in Figure 4.33b. The heading rises comparatively slow until the maximum roll angle is reached, where it increases faster linearly. In fact, an appropriate roll angle threshold may detect a curve entry before actually turning as depicted in Figure 4.33c. Note how smooth the roll angle signal is, making it well suited for prediction techniques by holding an instantaneous value constant or extrapolation. Moreover, the roll angle is much smoother than the yaw rate (or lateral acceleration)

This sub-subsection concludes that when entering a turn, the pedelec rolls first and then turns. The roll angle allows for an early and reliable detection of curves even by just a simple

4. Solution Approach

threshold. Such a roll angle threshold would even reliably separate the curve from its exit. Because of its smoothness, the roll angle is also well suited for prediction. For all of these reasons, the roll angle may be the most usable signal for predictions and should at least be better suited than using the yaw rate directly. Sub-subsection 5.1.5.1 evaluates whether replacing the yaw rates (indirectly) with the roll angles indeed boost the trajectory prediction performance.

4.8.4.3. Derivation of Yaw Rate as a Function of Roll Angle

This sub-subsection derives the yaw rate as a function of the more suitable roll angle. This transformation is necessary as the planar motion primitive, introduced in [17], expects a yaw rate in Street frame.

The system is simplified to a single mass body at the combined center of mass of the pedelec and rider and can therefore be modelled as an inverted pendulum as depicted in Figure 4.34, where the mass of the string is negligible.

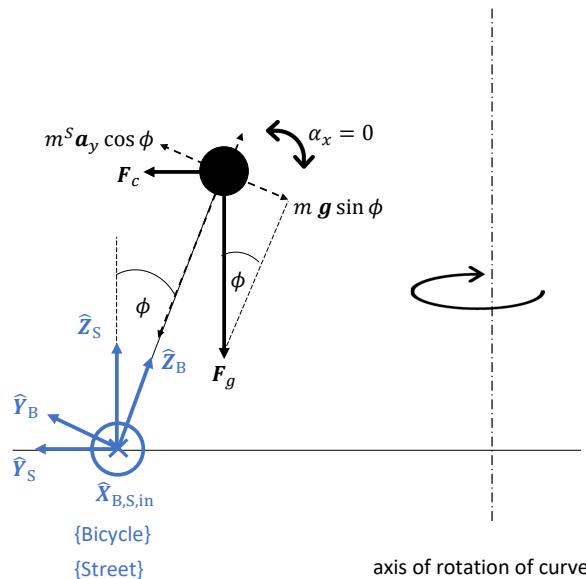


Figure 4.34.: Bicycle Simplified as Inverted Pendulum

Assuming a quasi-static scenario, where the angular acceleration about the x-axes of the frames $\alpha_x = 0$ for indefinitely small time deltas i.e. the bike and rider are assumed to be perfectly balanced while cornering at every time step. The torques exerted due to the

4. Solution Approach

centrifugal force τ_c and the gravitational force τ_g must therefore equal each other.

$$\begin{aligned}\sum \tau &= \tau_c - \tau_g = I \cdot \alpha_y = 0 \\ \tau_c &= \tau_g\end{aligned}$$

Only the forces perpendicular to the mass-less string can exert a torque based on their lever arm l , which is the distance from the center of rotation to the mass. From Figure 4.34 it becomes apparent that

$$F_c \cdot l \cdot \cos \phi = F_g \cdot l \cdot \sin \phi$$

Since both forces act upon the same center of gravity, their l are the same and cancel each other out which leads to:

$$m \cdot {}^B a_y \cdot \cos \phi = -m \cdot g \cdot \sin \phi \quad (4.22)$$

with g being a scalar (hence the minus). Cancelling out the masses and rearranging for the lateral acceleration in Bicycle frame results in:

$${}^S a_y = -g \cdot \tan \phi \quad (4.23)$$

Plugging in the relationship Equation 4.24

$${}^S a_y = {}^S \omega_z \cdot {}^S v_x \quad (4.24)$$

into Equation 4.23 and solving for the yaw rate results in:

$${}^S \omega_z \cdot {}^S v_x = -g \cdot \tan \phi \quad (4.25)$$

$${}^S \omega_z = -\frac{g}{{}^S v_x} \cdot \tan \phi \quad (4.26)$$

The maximum yaw rate ${}^S \omega_{z_{\max}}$ can be further determined by defining a maximum lateral acceleration through curves ${}^S a_{y_{\max}}$ by Equation 4.27:

$${}^S \omega_{z_{\max}} = \frac{{}^S a_{y_{\max}}}{{}^S v_x} \quad (4.27)$$

This way a useful physical limit can be set e.g. when extrapolating the roll angle signal.

This sub-subsection derived a yaw rate from the roll angle, pedelec speed and gravitational acceleration by modelling the system as a point mass at the combined CoG attached to an inverted pendulum with a mass-less rod. The next sub-subsection tests whether these assumptions hold in real-life.

4. Solution Approach

4.8.4.4. Validating Yaw Rate as a Function of Roll Angle

This sub-subsection validates the derived formula Equation 4.26 against the measured yaw rates in an effort to find out whether this relationship can be used to replace the yaw rates with the roll angles. Moreover, the yaw rates are integrated into headings which are compared with the measured heading from the INS.

Figure 4.35 shows a variety of signal comparison in order to validate Equation 4.26.

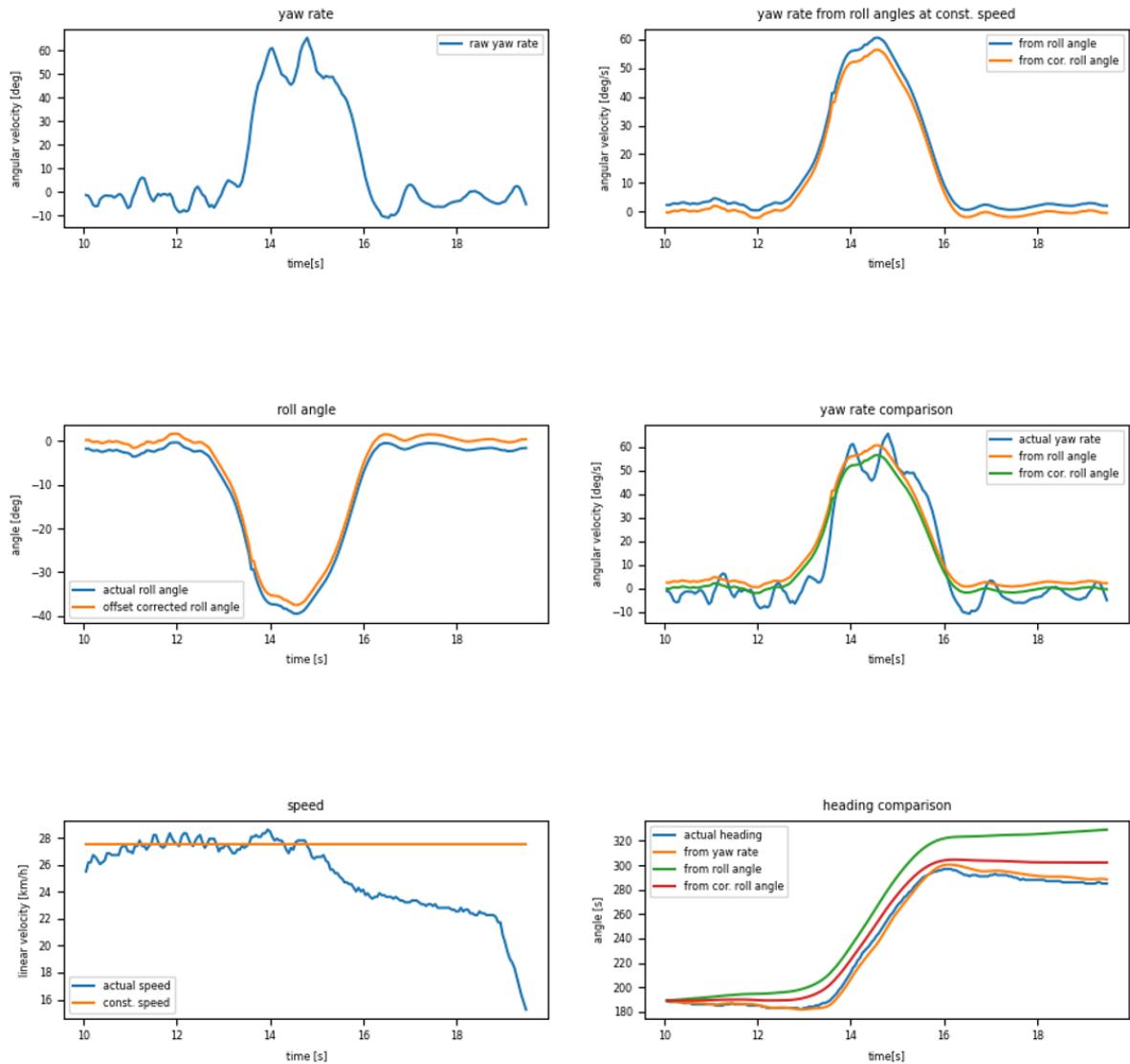


Figure 4.35.: Validating Yaw Rate as a Function of Roll Angle

The first row compares the raw yaw rate measured from the INS on the left with the yaw rates computed from the roll angle signals depicted in the first column of the second row. The

4. Solution Approach

plot in the second column of the second row lays all yaw rates on top of each other. Finally, the plot in the bottom right corner compares the measured headings with the integrated headings from the measured yaw rates and the computed yaw rates from the roll angles.

There are a few important things to point out:

The measured roll angle (blue signal in the second row, first column) has a slight offset which is corrected in the orange signal. Note how big of an impact this small roll angle offset has when comparing the headings in the bottom right plot: The green headings (integration of the yaw rates computed from the raw roll angles) deviates noticeably from the measured headings in blue. The heading error is much smaller when integrating the yaw rates from the offset-corrected roll angles (in red). In fact, this signal comes very close to the integrated headings from the measured yaw rates (in orange).

As depicted in the second row and second column, the yaw rates computed from the roll angles match the measured yaw rates (in blue) well. Actually, the resulting signal is much smoother and even leads the measured yaw rates. Smoothness should mean more promising results for both the constant and extrapolation prediction methods. A leading signal is obviously also desired for the prediction as the trajectories for the curve entries are likely improved.

All in all, the derived formula Equation 4.26 holds really in real-life. The resulting yaw rates match the actual signal well but are much smoother and even slightly lead the original signal which is both desired for prediction purposes. However, the formula is quite sensitive to errors in the roll angles such as offsets. Therefore, it is crucial to estimate the roll angle accurately and precisely. Nevertheless, yaw rates computed from roll angles seem promising. Sub-subsection 5.1.5.1 evaluates whether replacing the yaw rates from the INS with yaw rates computed from roll angles indeed boost the trajectory prediction performance.

4.8.4.5. Trapezoidal Extrapolation Logic

This sub-subsection describes a novel trapezoidal extrapolation aimed to improve the trajectory prediction for high dynamic curves. First, the general steps of this approach are listed, then the extrapolation logic is explained. Finally, some optional corrections are discussed.

It is generally observed that area under the roll angle signal in high dynamic curves can be approximated with a trapezoid that consists of four main sections as depicted with the yellow, green, purple and orange segments in Figure 4.36. A novel trajectory prediction model is thus developed that tries to harness this trapezoidal shape of the roll angle. It generally works as follows: First, after detecting a curve entry a roll angle profile is generated via extrapolation and some model parameters and followed along until the curve exit is detected. Then, these predicted roll angles need to get converted to yaw rates using the relationship Equation 4.26 derived in subsubsection 4.8.4.3. Optionally, the total heading difference from curve entry to exit is limited. Finally, the resulting yaw rate predictions are fed into the motion primitive which generates the future positions (and headings) via integration.

4. Solution Approach

The extrapolation logic that implements the multi-model Hypothesis 4.8.2.1 is explained with the help of Figure 4.36, which visualizes the generation of the roll angle profile.

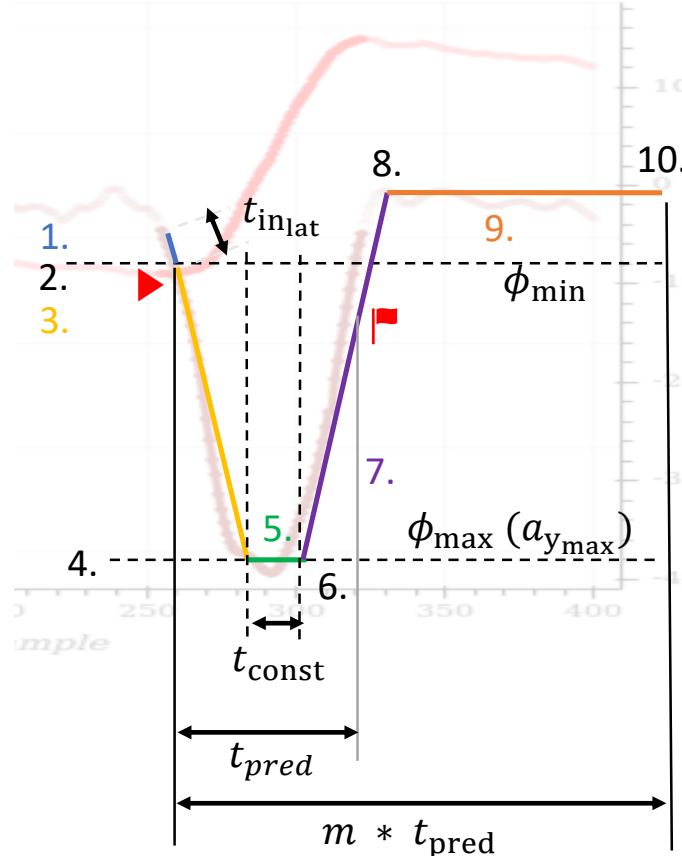


Figure 4.36.: Trapezoidal Extrapolation Logic

The generation of the roll angle profile consists of the following 10 steps: First, fit a line ($n_{pg_{lat}} = 1$) through the most recent points, defined via $t_{in_{lat}}$ analogous to Equation 4.20. Second, the current roll angle reaches above the magnitude of the curve threshold $|\phi_{min}|$ (detect curve entry). Third, linearly extrapolate the fitted line from 1. until the extrapolated roll angle reaches the magnitude of the maximum roll angle $|\phi_{max}|$, pre-defined via $|a_{y_{max}}|$ (see 4.). Fifth, hold the extrapolated roll angle constant at $|\phi_{max}|$ until the extrapolated roll angle hits a pre-defined constant duration t_{const} (see 6.). Seventh, linearly extrapolate with the opposite slope from 1. until the extrapolated roll angle hits zero (see 8.). Ninth, hold the extrapolated roll angle at zero until $m \cdot t_{pred} \cdot \Delta t$ samples are reached (see 10.).

Then, this generated roll angle profile is followed until the current roll angle falls below $|\phi_{min}|$ again, which marks the curve exit. At this point, the lateral signal is predicted by a simpler model e.g. constant heading or constant roll angle. Note that there were also other variants of this model implemented which extrapolate at every time step in the first and third section of the roll angle profile. However, it crucially does not extrapolate in the

4. Solution Approach

second section as roll angles tend to swing there which would lead to gradients that are too steep. Extrapolating more often than just at the beginning may make the model more robust against an inappropriate parameter selection for a specific curve but leads to a broader band of trajectories.

Since the model heavily depends on how well its pre-defined roll angle profile parameters reflect the actual curve, an online adaption is developed in an effort to correct bad parameter guesses. Figure 4.37 visualizes how this adaption works. The plots on the left show an initial overestimate of both the $a_{y_{\max}}$ and t_{const} parameters which leads to a wrong roll angle profile that integrates to an overestimated AUC. The plots on the right show the adapted roll angle profile.

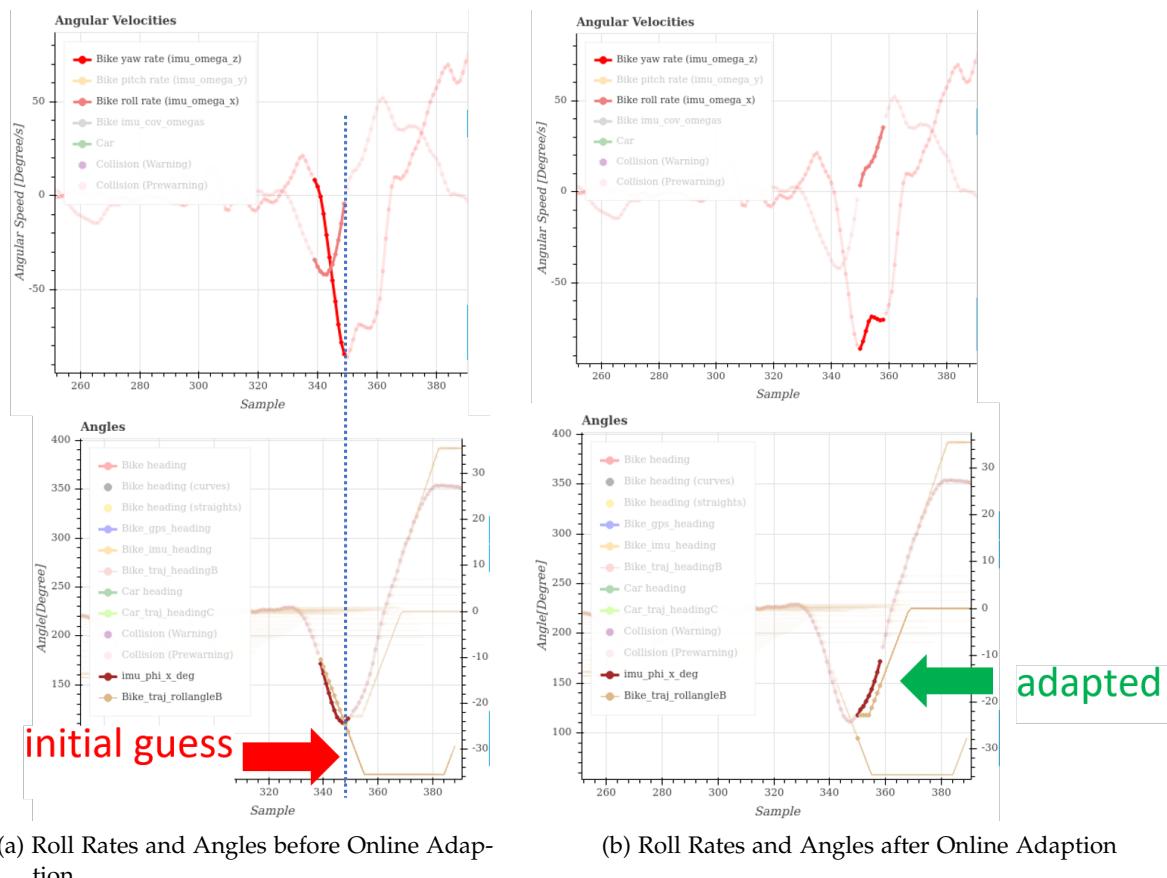


Figure 4.37.: Adaption of Overestimated $a_{y_{\max}}$ and t_{const} Parameters Based on Roll Rate

Essentially, after a curve entry has been detected, the algorithm waits until the roll rate crosses zero, which is equivalent to finding the local minimum/maximum of the roll angle signal. The algorithm stores the corresponding roll angle value. Subsequently, the algorithm determines the number of samples in the constant section of the trapezoid at this point,

4. Solution Approach

doubles the value and stores it as well. This assumes a mirror symmetry in the roll angle profile with the maximum occurring at its axis of reflection. Then, it re-generates the roll angle profile using the new constant roll angle and constant duration parameters. Note how after the online adaption, the new roll angle profile pointed by the green arrow predicts the actual roll angles much better. Figure 4.38 depicts the possible benefit on the predicted trajectories from correcting these parameters online.

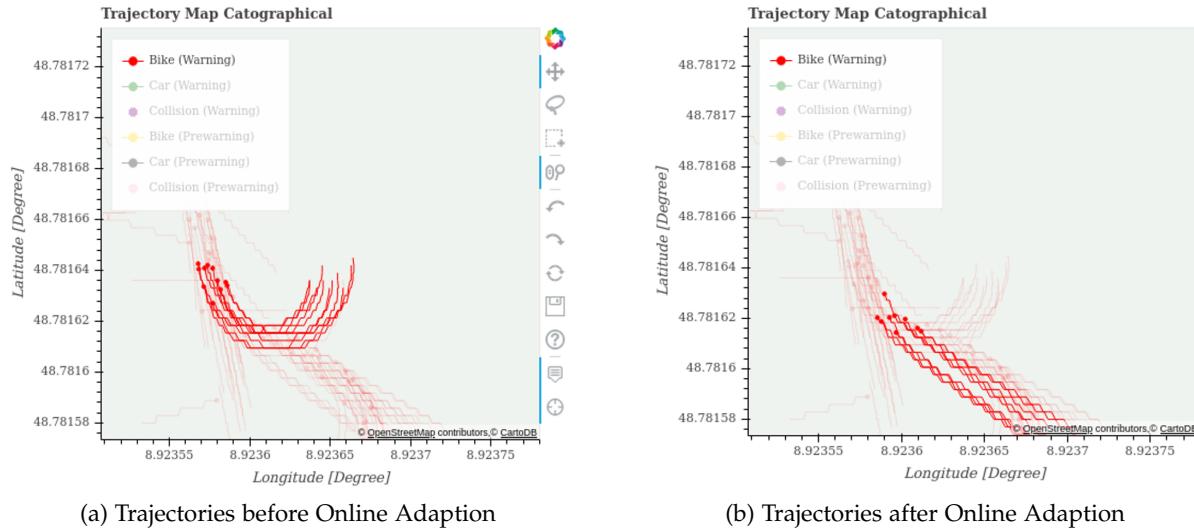


Figure 4.38.: Effect of Useful Online Parameter Adaption on Predicted Trajectories

Note how an initial overestimation of the curve parameters leads to trajectories that curl too much. After the online adaption, the trajectories point towards the actual path.

This sub-subsection a new trapezoidal extrapolation method that essentially switches between the linear extrapolation and constant prediction methods and thus implements the multi-model hypothesis discussed in subsection 4.8.2. The expectation of this approach is, that if its parameters are guessed correctly the curve entry and exit will be improved greatly due to the limited extrapolation and well-timed switches between specialized trajectory prediction models. However, if the parameters are guessed far off of the actual curve, the predicted trajectories may be worse than for e.g. a constant yaw rate model. Sub-subsection 5.1.5.2 evaluates the sensitivity of the parameters in detail.

4.9. Collision Detection and Warning Strategy

So far, the CD approximated crashes using a dynamic range dependent on the heading difference between the two vehicles and rather large buffers as explained in section 3.1 and analysed in subsection 4.1.7. This sufficed for simple demonstrations since the positioning and trajectories were too inaccurate for a sophisticated CD anyway. However, it resulted in

large FP rates.

To solve the discussed issues it may be useful to split the CD into three major parts: First, a collision is recognized via collision tests of circles in the broad phase and polygons in the narrow phase for every future point in time as described in subsection 4.9.1. Second, if a collision is detected, the location at which the pedelec should have come to a stop opposed to the actual collision location to prevent an accident is obtained as described in subsection 4.9.2. Third, based on the current speed of the pedelec, a safe distance to come to a complete stop. If the actual distance to the collision position from the current location is less than the assumed safe braking distance to come to a stop, a collision alarm is triggered. This is explained further in subsection 4.9.3.

4.9.1. Collision Recognition via Intersecting Rectangles

After integrating the more accurate GNSS (section 4.6) and virtually offsetting and correcting the pedelec's positioning (section 4.7), the collision recognition can be fine-tuned for a lower FP rate by solving the issues noted in subsection 4.1.7. Consequently, this subsection derives a custom algorithm that uses SAT-based CD in order to find the last safe point of a pedelec, which may be sooner than the actual collision location.

One improvement could be to virtually translate the heading dependent dynamic range by [55] around the geometric center of the vehicles to ensure symmetry. For some crash configurations e.g. perfect rear-end collisions as presented in Figure 4.39, this would solve the issue of the same heading difference leading to different distances.

4. Solution Approach

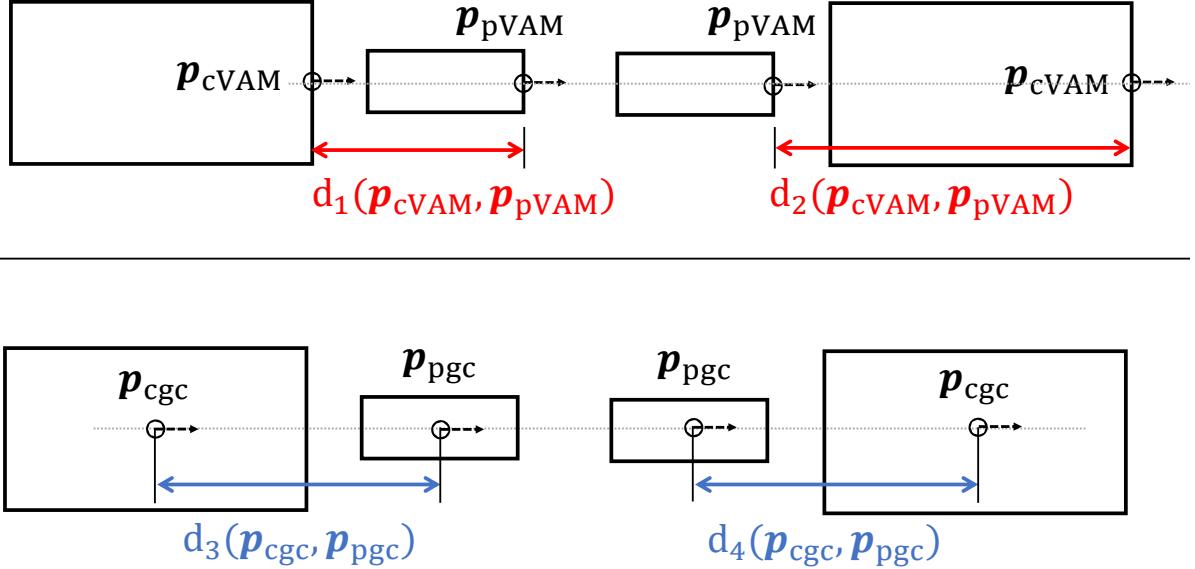


Figure 4.39.: Using Geometric Centers Can Solve Certain Crash Configurations

The figure shows two scenarios. In each scenario both vehicles travel from West to East as indicated by the dashed arrows, which results in a heading difference of zero between them. However, in the top scenario, $d_1(p_{cVAM}, p_{pVAM})$ is smaller than $d_2(p_{cVAM}, p_{pVAM})$ due to the vehicles' different lengths and virtual VAM locations at their fronts. Taking the geometric centers of the vehicles instead leads to the same distance for both cases, as symbolized by $d_3(p_{cgc}, p_{pgc})$ and $d_4(p_{cgc}, p_{pgc})$ in the lower scenario. However, even when comparing the geometric centers, there still exist too many possible configurations with the same heading difference but different L2-distances between the virtual GPS positions as visualized in Figure 4.40.

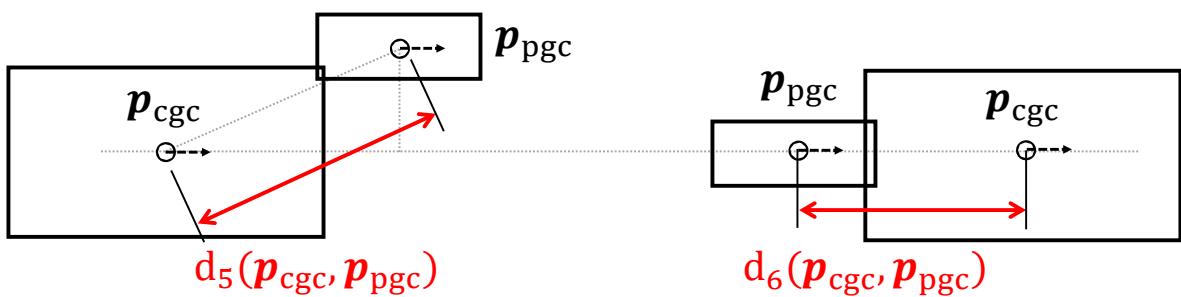


Figure 4.40.: Positions at the Geometric Centers Still Lead to Different Distances

The vehicles approach each other with a heading difference of 0 deg as before: In the upper scenario their corners hit while the bottom scenario shows a perfect rear-end collision.

4. Solution Approach

Note how the L2 distances $d_5(p_{cgc}, p_{pgc})$ and $d_6(p_{cgc}, p_{pgc})$ differ because of the Pythagoras's Theorem despite the crash configurations having the same heading delta. Thus, a different approach is chosen that relies on the intersection of polygons instead of heading difference. The polygon approach is explained in the next few paragraphs.

For computational efficiency, the car and pedelec are approximated using tight rectangle bounding boxes as shown in Figure 4.41 and assumed to be rigid bodies without deformation. The approach can easily be extend to more complex polygons as long as they stay convex.

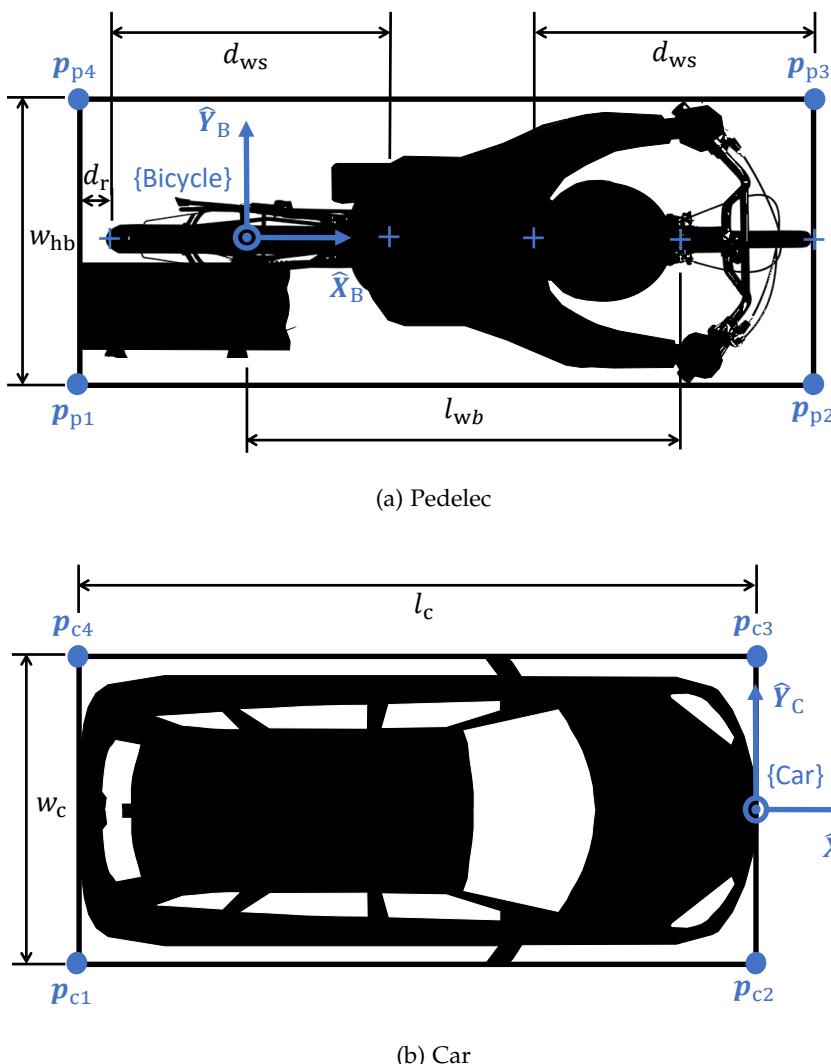


Figure 4.41.: Rectangular Convex Hulls around Road Users

Many CD algorithms for polygons require the vertices of the involved objects in world coordinates. In order to obtain the vertices for the pedelec from the reported GPS positions, the transformation function resulting from Equation 4.17 of section 4.7 can be reused by using

4. Solution Approach

an arbitrary vertex location ${}^B\mathbf{p}_{ci}$, instead of a static VAM location (see Equation 4.28):

$${}^S\mathbf{p}_{pi} = \begin{bmatrix} \cos \psi & -\sin \psi \cos \phi & \sin \psi \sin \phi & x_{GPS} \\ \sin \psi & \cos \psi \cos \phi & -\cos \psi \sin \phi & y_{GPS} \\ 0 & \sin \phi & \cos \phi & z_{GPS} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -o_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^B\mathbf{p}_{pi} \quad (4.28)$$

The transformation from a point in $\{\text{Car}\}$ to $\{\text{Street}\}$ is simple in comparison as the positions do not need to be translated virtually and the roll motion is not considered. The positions only need to be translated by the reported GPS coordinates of the car and rotated by ψ about its z-axis of the car (see Equation 4.29):

$${}^S\mathbf{p}_{ci} = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^C\mathbf{p}_{ci} \quad (4.29)$$

Car Original Equipment Manufacturer (OEM)s typically provide the length l_c and width (including side mirrors) w_c of their cars in millimeters.

In contrast, bicycles are more configurable and often come in different frame and handle bar sizes, so bicycle OEMs often provide the wheelbase length l_{wb} and handle bar width w_{hb} alongside the diameter of the wheels d_{ws} (often in inches). These dimensions can also conveniently be measured or found out if not provided by the OEM. All of the aforementioned dimensions are drawn into Figure 4.41 alongside their reference coordinate systems. The nomenclature of the vertices for each rectangle are defined as follows: The position of the first point, e.g. \mathbf{p}_{p1} or \mathbf{p}_{c1} lies in the third quadrant of the corresponding reference coordinate system and gets incremented counterclockwise. From Figure 4.41 it becomes evident what the homogeneous coordinates of the vertices for the pedelec with respect to $\{\text{Bicycle}\}$ are as described in Equation 4.31:

$${}^B\mathbf{p}_{p1} = \begin{bmatrix} -(d_{ws}/2 + d_r) \\ -(w_{hb}/2) \\ 0 \\ 1 \end{bmatrix}, {}^B\mathbf{p}_{p2} = \begin{bmatrix} l_{wb} + d_{ws}/2 \\ -(w_{hb}/2) \\ 0 \\ 1 \end{bmatrix}, {}^B\mathbf{p}_{p3} = \begin{bmatrix} l_{wb} + d_{ws}/2 \\ w_{hb}/2 \\ 0 \\ 1 \end{bmatrix}, {}^B\mathbf{p}_{p4} = \begin{bmatrix} -(d_{ws}/2 + d_r) \\ w_{hb}/2 \\ 0 \\ 1 \end{bmatrix} \quad (4.30)$$

Note that the z-components are set to an arbitrary value of zero as this thesis only cares about the projection into 2D space anyways. Similarly, the homogeneous vertex coordinates of the car in $\{\text{Car}\}$ are defined by Equation 4.31 as depicted in Figure 4.41.

$${}^C\mathbf{p}_{p1} = \begin{bmatrix} -l_c \\ -w_c/2 \\ 0 \\ 1 \end{bmatrix}, {}^C\mathbf{p}_{p2} = \begin{bmatrix} 0 \\ -w_c/2 \\ 0 \\ 1 \end{bmatrix}, {}^C\mathbf{p}_{p3} = \begin{bmatrix} 0 \\ w_c/2 \\ 0 \\ 1 \end{bmatrix}, {}^C\mathbf{p}_{p4} = \begin{bmatrix} l_c \\ w_c/2 \\ 0 \\ 1 \end{bmatrix} \quad (4.31)$$

4. Solution Approach

There exist many methods of detecting collisions between polygons and many corresponding Python implementations can be used. For example, the `intersection(*)` method of *Sympy*, a Python library for symbolic mathematics, computes the intersection points of a polygon with another geometric instances such as polygons, circles, lines etc. [91] [92]. However, it seems to iterate through the points of the geometric instances in a brute-force-like manner and hence is painfully slow. This becomes problematic because predicted trajectories are computed at 20 Hz in the ROS architecture. At the worst-case, where no collision occurs, a collision check has to be performed for each future steps. The number of steps k depends on the sampling resolution of the motion primitive Δt and the prediction horizon t_{pred} in seconds as shown in Equation 4.32:

$$k = \frac{t_{\text{pred}}}{\Delta t} \quad (4.32)$$

Up to k intersections have to be computed within the node's cycle, defined via its frequency f_{node} which results in the computation time for a single intersection Δt_{ints} as described in Equation 4.33:

$$\Delta t_{\text{ints}} < \frac{1}{f_{\text{node}} k} \quad (4.33)$$

Given $t_{\text{pred}} = 2$ s and $\Delta t = 0.05$ s at $f_{\text{node}} = 20$ Hz, this would result in $\Delta t_{\text{ints}} < 1.25$ ms on average to ensure real-time capability. For $t_{\text{pred}} = 4$ s, this would decrease to $\Delta t_{\text{ints}} < 0.625$ ms. Obviously, this does not yet account for all of the other computations that may also take place when computing the TTC and whether a crash is likely to occur. However, when using *Sympy*, even computing only a single intersection already takes considerably longer f_{node} as shown in Table 4.2, which means more efficient CD methods are necessary. Furthermore, finding the intersection point of the rectangles equates to location at which the crash occurs, which does not necessarily represent the location where the pedelec should come to a stop to avoid the accident. For example, in Figure 4.24 the pedelec would need to come to a stop significantly sooner than the intersection point of the rectangles in order to avoid the accident. Alternatively, the pedelec accelerates and passes the car completely. For this reason, most CCD algorithms used in physics engines, which find the collision point, cannot be used out of the box to solve this problem. Therefore, a custom approach with a run-time efficient CD backbone is necessary as explained in subsection 4.9.3.

A considerably faster method for computing whether two polygons collide is the SAT, as introduced in section 2.6. Note, that the utilized vanilla SAT code, implements just the basic theorem without any of the discussed optimizations in section 2.6 for moving 2D rectangles and uses plain Python without performance enhancements via *NumPy* [93]. However, it still greatly outperforms the *Sympy* intersection method up to a factor of over 800 times. Table 4.2 compares the mean, standard deviation, minimum and maximum run-times in milliseconds of the *Sympy* and SAT approaches for computing a single intersection between two rectangles averaged over 5000 iterations. The table clearly shows that the *Sympy* function takes on average significantly longer than the available maximum budget of $\Delta t_{\text{ints}} < 1.25$ ms. In fact, not even the fastest iteration could achieve the time budget, while the worst-case execution time exceeded it by a factor of over 100. In contrast, all iterations of the SAT manage to

4. Solution Approach

Algorithm	μ [ms]	σ [ms]	Minimum [ms]	Maximum [ms]
SymPy	86.1631	5.5864	80.0201	140.5232
SAT	0.1062	0.0201	0.0974	0.2803

Table 4.2.: Comparison of Computation Times for Single Intersection

execute within the time budget, while the worst-case is still almost 4.5 times faster.

As mentioned in section 2.6, the run-time efficiency of a CD is further improved by computing the polygon collision only if the rectangles are critically close to each other and otherwise using encapsulating circles for the collision checks which is computationally much less expensive. The encapsulating circles are constructed from the geometric centers of the rectangles as shown in Figure 4.42. The critical distance d_{critical} is defined in Equation 4.34 as the sum of

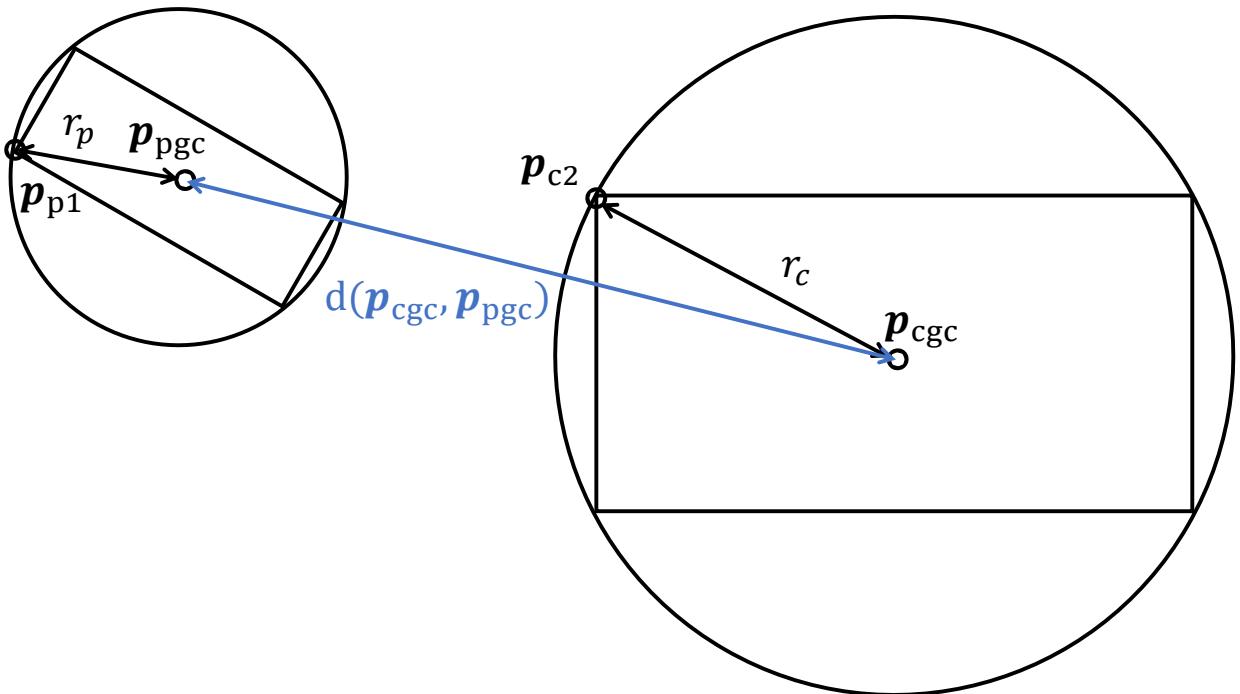


Figure 4.42.: Non-Colliding Circles

the radii of circles that encapsulates each of the rectangles

$$d_{\text{critical}} = r_c + r_p \quad (4.34)$$

where r_c is the L2-distance between the geometric center of the car rectangle and any of its vertices p_{c1} through p_{c4} and r_p is the L2-distance between the geometric center of the pedelec rectangle and any of its vertices p_{p1} through p_{p4} . Similarly, $d(p_{cgc}, p_{pgc})$ defines the distance

between the geometric centers of the two objects as shown in Equation 4.35

$$d(\mathbf{p}_{\text{cgc}}, \mathbf{p}_{\text{pgc}}) = \|\mathbf{p}_{\text{cgc}} - \mathbf{p}_{\text{pgc}}\|_2 = \sum_{i=1}^S |(p_{\text{cgc}_i} - p_{\text{pgc}_i})^2| \quad (4.35)$$

where n defines the number of spatial dimensions. So besides computing the radii r_c and r_p once, the CD in the broad phase only requires the computation of the distance between the geometric centers and a single collision check for every time step t . The collision check for two circles $\text{cir}(t)$ becomes trivial as shown in Equation 4.36:

$$\text{cir}(t) = \begin{cases} 1, & \text{if } d(\mathbf{p}_{\text{cgc}}, \mathbf{p}_{\text{pgc}}) \leq d_{\text{critical}} \\ 0, & \text{otherwise.} \end{cases} \quad (4.36)$$

If the distance between the geometric centers of the rectangles is less than or equal to their critical distance, then the circles must collide.

Putting all things together, the CD of the ADAS $cd(t)$ uses the circle intersection $c(t)$ in the broad range and SAT-based rectangle intersection $sat(t)$ in the near phase as mathematically described in Equation 4.37:

$$cd(t) = \begin{cases} sat(t), & \text{if } cir(t) = 1 \\ cir(t), & \text{otherwise.} \end{cases} \quad (4.37)$$

So, the algorithm always checks the cheap circle intersection first and only if it returns a collision, the rectangle collision is run. Then, in case the rectangle intersection returns a collision as well, the algorithm concludes that an accident is bound to happen. The next subsection describes how backtracking is started in this case to find the actual last safe location for the pedelec and when the cyclist is warned about this collision.

4.9.2. Finding the Last Safe Location

As noted in subsection 4.1.7, the collision location may not be a good indicator where the pedelec should actually come to a stop in order to prevent a crash. This section investigates where this stopping location should be and derives an algorithm to obtain this spot.

Consider Figure 4.43 where the car (blue rectangle) crosses the path of the pedelec (green rectangle) at roughly a 90 degree angle. In this case, the collision location occurs when the car barely clips the rear wheel of the pedelec. If the pedelec came to a complete stop one time step before this collision point, the situation would actually be worse for the cyclist as the car would hit a larger portion of the pedelec. The pedelec actually needs to come to a complete stop at the stopping location marked with a red rectangle that does not cross the car's driving tube. Alternatively, the pedelec could accelerate such that the car would arrive too late to clip the cyclist.

4. Solution Approach

A simple way to obtain the stopping location of the pedelec from the trajectories of the vehicles, rather than the collision location, uses SAT and backtracking as visualized in Figure 4.43.

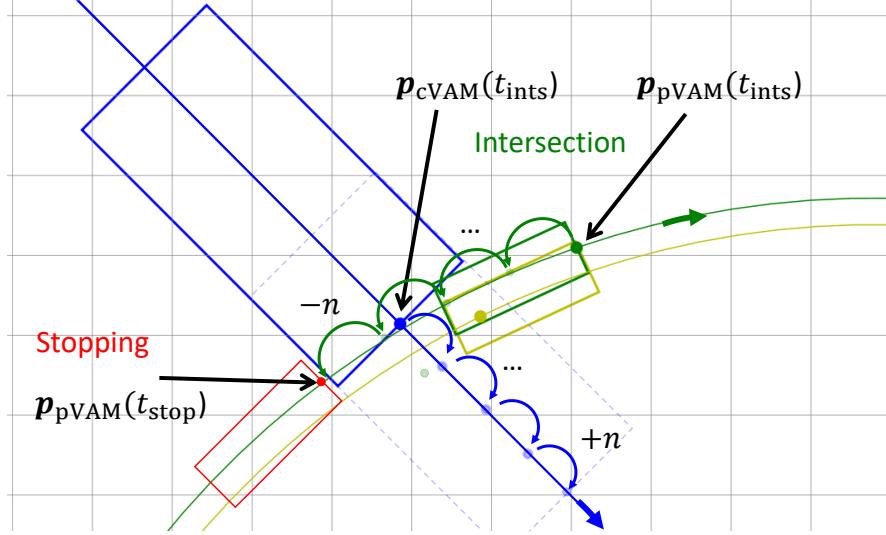


Figure 4.43.: Finding Stopping Location for Pedelec

First, the algorithm finds the first point of actual intersection (if any) of the rectangles via SAT and stores this timestamp t_{ints} corresponding to the pedelec's position at the moment of intersection $p_{\text{pVAM}}(t_{\text{ints}})$. The algorithm then repeatedly sets the pedelec's rectangle one step back in time and checks whether it still intersects with the car's rectangle which is propagated one time step into the future. The backtracking continues until the pedelec does not collide with the car anymore, say after $-n$ and $+n$ steps. If the start or end of the prediction horizon for the pedelec or car is reached, respectively, the time steps are held constant to avoid array-index-out-of-bounds errors. If no solution is found after k steps, as defined in Equation 4.32, the initial collision location is used as the stopping location to prevent endless loops. The stopping location of the pedelec $p_{\text{pVAM}}(t_{\text{stop}})$ as symbolized by the red dot of the red rectangle can thus be defined in Equation 4.38

$$p_{\text{pVAM}}(t_{\text{stop}}) = p_{\text{pVAM}}(t_{\text{ints}} - n\Delta t) \quad (4.38)$$

where t_{ints} denotes the time step at the rectangle intersection, n the number of backtracking steps, Δt the time delta between each trajectory step and t_{stop} the time step at which the pedelec should actually come to a complete stop. Note that this is actually computed recursively, by repeatedly calling the same rectangle intersection function with a decremented time index.

4.9.3. Alarm Based on Safe Breaking Distance

As discussed in subsection 4.1.7 the collision trigger should be a function of the pedelec's initial speed to ensure that warnings arrive sooner the faster the pedelec is traveling and avoiding constant alarms when the pedelec is nearby the car but moving at slow speeds or even is at rest. Therefore, this section explores an alarm based on the total distance required for the pedelec to come to a complete stop before hitting another vehicle is developed.

subsection 4.9.2 noted that the stopping location might differ from the collision location. This stopping distance should thus be based on the stopping location, rather than the collision location. Furthermore, the stopping distance should also account for the cyclist's reaction time after the software triggers a warning signal. The new alarm works as follows: After an upcoming collision is recognized via rectangle intersection and the stopping location is found via backtracking as described in the previous subsections, the trajectory length from the stopping location to the pedelec's position at the current time step is computed. Note that the distance along the pedelec's trajectory is taken rather than shortest straight line distance from collision position to current pedelec position. If this distance is shorter than the conservatively computed total stopping distance for the pedelec, then the alarm is triggered. The following paragraphs describe the formulas used to compute the total stopping distance of the pedelec. More detailed derivations of Equation 4.39 and Equation 4.41 can be found in section A.2.

The total stopping distance can be modelled as the sum of two parts. The first part models the travelled distance during the time interval from the system's collision alarm to the rider fully engaging the brake levers. By assuming a constant speed throughout this duration, the distance can be obtained via integration of the initial speed v_i over a time duration t_{react} as follows:

$$d_{react} = v_i \cdot t_{react} \quad (4.39)$$

The second part models the travelled distance of the pedelec during the emergency brake by assuming a constant braking deceleration a_{brake} . The derivation is based on rewriting $a = \frac{dv}{dt}$ using the chain rule of differentiation and integrating the acceleration part with respect to position and the velocity part with respect to velocity:

$$d_{brake} = \frac{v_i^2}{2 \cdot a_{brake}} \quad (4.40)$$

The total stopping distance then results in Equation 4.41:

$$d_{stop} = d_{react} + d_{brake} \quad (4.41)$$

$$d_{stop} = v_i \cdot t_{react} + \frac{v_i^2}{2 \cdot a_{brake}} \quad (4.42)$$

The combined mass of the pedelec and rider, coefficient of friction between the road and tires, the efficiency of the brakes, the inclination etc. thus are only indirectly accounted for by assuming a conservative average deceleration. Similarly, the reaction time needs to be

4. Solution Approach

chosen on the conservative side as well. It is crucial to underestimate the a_{brake} and t_{react} parameters based on the specific braking setup of the pedelec. This ensures that the rider can achieve a smaller stopping distance in reality than d_{stop} used to call out the alarm, even under worse-case conditions (e.g. downhill and slippery). Otherwise, the alarm could arrive too late in some cases. According to a literature review on the braking performance of motorcycles, decelerations with ABS usually lay between $0.642g - 0.842g$, where $g = 9.81 \frac{\text{m}}{\text{s}^2}$ [94] [66]. A study by [95] reports average decelerations of $3.5 \frac{\text{m}}{\text{s}^2} - 4.6 \frac{\text{m}}{\text{s}^2}$ for bicycles with caliper brakes and no ABS under normal road conditions. A recent study by Society of Automotive Engineers (SAE) state decelerations between $0.40g - 0.71g$ for bicycles with caliper and disc setups when both rear and front bakes are applied [96]. Similar to [95], all test were conducted on dry asphalt without ABS. According to a recent sports study, reacting to a single visual target while riding a bicycle ranges between $0.30 \text{ s} - 0.52 \text{ s}$ with a median of roughly 0.4 s for female and male cyclists of varying ages [97]. Our constellation consists of disc brakes for both front and rear wheels, front wheel ABS and alert cyclists. For the aforementioned constellation, conservative estimates of $a_{\text{brake}} = 3.0 \frac{\text{m}}{\text{s}^2}$ and $t_{\text{react}} = 0.75 \text{ s}$ are thus assumed. Additionally, a fixed safety buffer between the car and pedelec rectangles can be parameterized. In this case, the rectangle of the car is padded by this buffer on all sides (`warn_range_buffer`= 0.25 m). Note that picking this value too large will increase the FP rate in cases where the pedelec is close to the car but not actually colliding, e.g. when standing next to or behind a car at a traffic light.

The TTC can be seen as a Time-To-Stop (TTS) as it denotes the remaining time until the location where the pedelec should come to a complete stop rather than the location at which the collision occurs. Furthermore, the distance between the pedelec's current position in time and the stopping location is computed along the pedelec's trajectory opposed to the straight line distance. For example, if the pedelec is riding a curve, the distance along its trajectory will be significantly larger than the straight line distance. However, from a safety point of view, taking the straight line distance might be more desirable as it will always return an underestimation of the actual distance as it will always return the shortest distance. The distance along the pedelec's trajectory until the stopping location can mathematically defined as in Equation 4.43

$$d_{\text{traj}}(t) = \sum_{i=t+1}^{TTS} \|\mathbf{p}_{\text{pVAM}i-1} - \mathbf{p}_{\text{pVAM}i}\|_2 \quad (4.43)$$

where $\mathbf{p}_{\text{pVAM}i}$ denotes the pedelec's position at its VAM location (front tire) at time step i . Essentially, this sums the norm of the difference over all adjacent position pairs until the TTS. Note that this formula is not implemented as a loop in the software and rather by duplicating the arrays, padding and shifting them by one before subtracting them and computing the vectorized sum of the norm of this difference vector.

The collision warning $\text{warning}(t)$ is triggered at time step t if the remaining distance of the pedelec until the stopping location $d_{\text{traj}}(t)$ is less than the conservatively expected total

4. Solution Approach

stopping distance $d_{\text{stop}}(t)$ as defined in Equation 4.44.

$$\text{warning}(t) = \begin{cases} 1, & \text{if } d_{\text{traj}}(t) \leq d_{\text{stop}}(t). \\ 0, & \text{otherwise.} \end{cases} \quad (4.44)$$

To prevent FP when the cyclist is pushing the bike or jerking it back and forth at very low speeds, v_i is clipped to zero if below a certain threshold:

$$v_i = \begin{cases} v_i, & \text{if } v_i > v_{\min}. \\ 0, & \text{otherwise.} \end{cases} \quad (4.45)$$

This way the breaking distance $d_{\text{stop}}(t)$ is assured to be zero for these cases, whereas the distance along the pedelec's trajectory $d_{\text{traj}}(t)$ is greater than zero. As a result, the alarm will not trigger.

4.9.4. Multi-Staged Warning Using Pre-Warnings

As discussed already, if the system triggers too many FP emergency warnings the cyclist and surrounding people might be put in danger because of unnecessary and unexpected emergency brakes that could lead to additional rear-end collisions. In addition, they may get annoyed by the frequent beeps and may even lose trust in the ADAS completely because it "does not work" and deactivate it. On the other hand, an extremely low FP rate similar as in cars that only triggers a warning extremely late if a collision is unpreventable may not protect the cyclist enough. This subsection describes how a multi-staged and split warning strategy may solve the trade-off between not warning enough and warning too often.

One possible way is to use two sets of trajectory prediction models that generate urgent warnings and pre-warnings. One model predicts high-accuracy short-term trajectories, e.g. $t_{\text{pred}} = 2\text{s}$, that are used to trigger an urgent and aggressive reaction by the cyclist using the beep sounds e.g. an emergency brake. In parallel, longer but more inaccurate trajectories, e.g. $t_{\text{pred}} = 4\text{s}$, are predicted using a different model in order to gently warn the cyclist about potentially dangerous positions using only the vibration. The intuition is that the sooner the system warns, a gentle speed change by the cyclist might already be sufficient in order to avoid the critical situation completely, e.g. in a 90 degree intersection the car would have already passed by the time the cyclist arrives there. A gentle speed adaption, opposed to an emergency brake, should result in less rear-end collisions between road users behind the cyclist and the pedelec. Moreover, the vibration only alerts the cyclist and does not confuse or annoy other road users.

Furthermore, it may make sense to additionally group these warnings based on their TTS. If a pre-warning occurs way too early, e.g. the TTS is above a threshold t_3 , the cyclist may get confused what situation this may refer to and might change his/her intention multiple times in this time frame. On the other hand, triggering an auditory urgent-warning right before a crash, e.g. the TTS is below a threshold t_1 , does not benefit the cyclist anymore and may even

4. Solution Approach

disturb their panic reaction. In this case, it may make more sense to trigger special safety devices that additionally protect the VRUs e.g. a cyclist airbags that fully encloses the cyclist shortly before impact. The final grouped warning strategy $strategy(t)$ based on TTS can be formalized in Equation 4.46

$$strategy(t) = \begin{cases} nowarn(t), & \text{if } TTS(t) > t_3. \\ prewarning(t), & \text{if } t_2 < TTS(t) < t_3. \\ warning(t), & \text{if } t_1 < TTS(t) < t_2. \\ airbag(t), & \text{if } TTS(t) < t_1. \end{cases} \quad (4.46)$$

where $nowarn(t)$ suppresses all warning modalities (vibration and sound), $prewarning(t)$ only vibrates, $warning(t)$ beeps and $airbag(t)$ fires an active safety device but suppresses auditory and tactile warnings. The thresholds t_1 through t_3 are picked in alignment with t_{pred} for the warning and pre-warning models.

Subsection 4.1.6 showed that the positional error increases quadratically with respect to the prediction horizon. Hence, a Hypothesis 4.9.4.1 arises:

Hypothesis 4.9.4.1 *Different time horizons favor different type of physics-based prediction models.*

Since a `const_speed_const_heading` model only introduces a linear error for the longitudinal and lateral components with respect to time, this simple model may perform best on average for very long t_{pred} despite being the simplest model. Figure 4.44 compares the trajectories of the `const_speed_const_heading` model against the base `const_a_yawr` model for $t_{pred}=4\text{s}$.

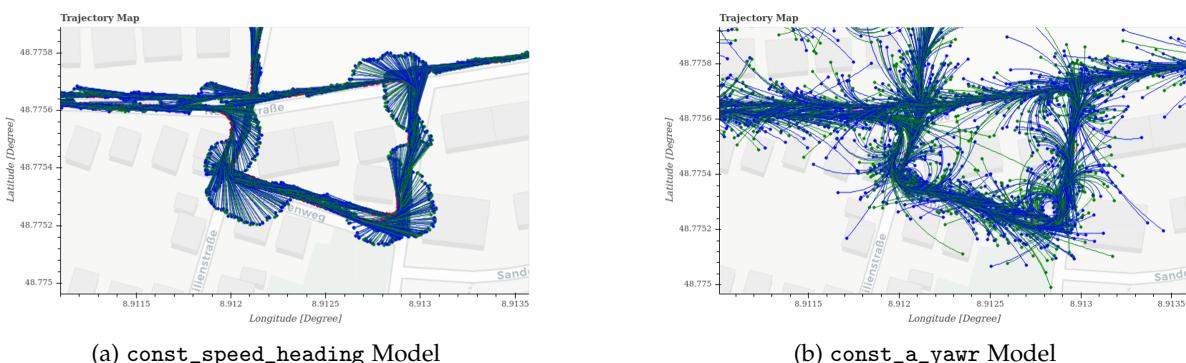


Figure 4.44.: Comparison of Pre-Warning Trajectories for $t_{pred} = 4\text{s}$

Note how the `const_a_yawr` model leads to wild trajectories while the `const_speed_const_heading` model trajectory represents the intuition of a cyclist that keeps riding exactly the same way, so not changing the speed or heading, which might result in more reliable pre-warnings.

4. Solution Approach

Finally, a bug is fixed in the initial `warn_strategy_node.py` at the multiprocessing section, where the replay of the beep sound and vibration profile is started in parallel. Since the child processes are forked before they have finished, the operating system will not find them again. This results in "zombie" processes as explained in section 4.3 that pile up the NUCs ressources with empty shells.

4.10. Adaptons to Vulnerable Road User Awareness Message

The pedelec now sends the virtual positions at the front tire instead of the GNSS antenna, as described in section 4.7. Ergo, the VAM communicates the `/virtual/fix` instead of the `/fix` topic to other road users. Furthermore, the `/imu/data` topic containing the AHRS data from the BNO055 does not exists anymore and is replaced with the `/transformation imu_data` topic that contains the inertial measurements from the ZED-F9R driver in the `{Street}` frame.

5. Evaluation

This chapter compares and evaluates the improvements of the sub-components developed in the course of this thesis. First, section 5.1 investigates the impacts of various factors intrinsically in terms of the trajectory prediction error metrics introduced in section 3.1. Such intrinsic evaluations provide a solid understanding of the trajectory prediction component but not necessarily how much the actual target ADAS function improves or what the impact of the other improved components such as positioning, CD and warning strategy is¹. Therefore, section 5.2 further evaluates the impacts of the improved components extrinsically on the collision warning system as a whole. Finally, section 5.3 discusses these quantitative and qualitative results.

5.1. Intrinsic Evaluation via Trajectory Prediction Errors

This section evaluates the impacts of certain factors intrinsically by computing the x- and y-position, length and angle errors of the predicted endpoints of a trajectory with the corresponding points of the ground truth positions (actual path driven) as defined in [55]. More specifically, it examines the impact of the stationary periods in every measurement, the route choice as well as the signal and extrapolation methods used for the longitudinal and lateral components of the trajectory prediction models on these metrics. In addition, the last subsection benchmarks promising models from this thesis against those from previous works for a variety of prediction horizons. Hereby, this subsection does not only quantitatively compare their trajectory prediction error metrics but also their qualitative performance on selected road segments. This section starts out with the base `const_a_yawr` model, investigates the impact of a single factor separately, picks the best option before sequentially moving on with this setting for all subsequent experiments where other single factors are evaluated separately. For example, once a subsection concludes that trajectory prediction models using the ABS WSS indeed outperform those that use the HMI WSS, the superior ABS WSS will be used for every future experiments involving the speed signal. These factors evolve from simple to complex.

5.1.1. Influence of Stationary Periods

As observed in subsection 4.1.4, the stationary phase in the beginning of every measurement until a recording has been started in ROS and the EKF has converged may falsify the metrics

¹As mentioned in section 4.6, evaluating the actual positioning error of the *simpleRTK2B-F9R board* for this specific pedelec setup is not scope of this thesis

5. Evaluation

of the trajectory prediction models and should thus be excluded in the evaluations. This subsection investigates the magnitude of these falsifications in more detail.

Table 5.1 compares the accuracy metrics from a `const_a_yawr` model with $t_{\text{pred}} = 2 \text{ s}$ for a short trip including these stationary phases with the same trip where the phase in the beginning is removed from the statistical evaluation. In both cases, the stationary phase at the end was omitted to highlight the impact of only the stationary phase at the beginning.

const_a_yawr	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta\sigma} [\text{deg}]$	$\sigma_{\Delta\sigma} [\text{deg}]$
Stationary period <i>only</i> at beginning	34510.85	206349.47	107254.90	424908.74	137277.04	465807.57	9.80	39.95
No stationary period	0.262	2.25	0.23	1.51	0.27	2.27	1.00	6.91

Table 5.1.: Impact of stationary phase of 19 s at beginning on trajectory prediction errors for $t_{\text{pred}} = 2 \text{ s}$ (riding time 65 s)

The stationary phase in the beginning is roughly 19 s, compared to an actual riding time of 65 s, resulting in a high ratio of $\frac{19}{65} * 100 = 29.23\%$ that leads to a significant falsification of the prediction errors. As reported in Table 5.1, the accuracy and precision of the positional metrics are falsified by multiple kilometers. As expected, the impact on the mean and standard deviation of the angle is comparatively small but still noticeable.

Similarly, Table 5.2 compares the accuracy metrics from a `const_a_yawr` model with $t_{\text{pred}} = 2 \text{ s}$ for a longer trip with and without the stationary phase at the end. In both cases, the stationary phase at the beginning was omitted.

const_a_yawr	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta\sigma} [\text{deg}]$	$\sigma_{\Delta\sigma} [\text{deg}]$
Stationary period <i>only</i> at end	0.16	2.62	0.13	1.78	0.10	2.63	0.71	20.47
No stationary period	0.16	2.65	0.13	1.80	0.10	2.66	0.57	11.52

Table 5.2.: Impact of stationary phase of 49 s at end on trajectory prediction errors for $t_{\text{pred}} = 2 \text{ s}$ (riding time 1795 s)

Since the ratio of stationary duration to riding duration is much lower at only $\frac{49}{1795} * 100 = 2.73\%$ compared to the previous comparison and the positions circulate around the resting location with a radius of a meter compared to offsets of multiple kilometers, the falsification is much smaller but still relevant. As expected, the standard deviation of the angle is worsened dramatically by almost a factor of two due to the oscillating headings. On the other hand, the standard deviations of the positional errors are actually falsely improved by up to 6 cm in this case. This results from the fact that the trajectory prediction errors for this model and prediction horizon are often of multiple meters, whereas the circulating motion at the end is around one meter, thus resulting in comparatively accurate predictions. These comparatively accurate trajectory predictions falsely shift the mean and standard deviations from the true distribution towards the normal distribution of the end phase with a zero-mean and standard

deviation of 1 m.

The falsification from the start and end of each recording is independent of the trajectory prediction model. This means when comparing the accuracy and precision of different models relative to each other this falsification does not matter as the same errors are present for all models as long as the EKF is used for the heading (and position) estimation in all models². However, as shown in Table 5.1 and Table 5.2 the magnitudes deviate significantly compared to the actual ones. Since the magnitudes are of importance to indicate how likely an actual collision will occur under a certain prediction variance, it is crucial to omit the stationary phases at the beginning and end of every measurement, especially for short recording that usually have a high stationary-to-riding ratio.

As a result, all of the following trajectory prediction error statistics omit the stationary start and end phases of the recordings (usually two minutes each).

5.1.2. Influence of State Estimation and Inertial Measurements

This subsection evaluates the impact of the enhanced state estimation (position and heading) as well as inertial motion data as a result of integrating a high-precision INS in section 4.6 on the trajectory prediction. Remember from section 3.1 that the trajectory prediction metrics take the positions of the "actually" ridden path as reference for computed the prediction errors of the predicted positions. So, the reference positions include the offsets and other errors from the corresponding sensor setup that recorded the ridden path, opposed to ground truth data that truthfully represent the ridden trip. Comparing the trajectory prediction metrics thus will not necessarily tell us which sensor setup actually works better. Therefore, this comparison is of qualitative nature.

For comparison, a cyclist rides roughly the same path about the office and records the state estimations two times: For the first ride, the ROS software uses the initial *BNO055-USB-Stick* and *Navilock NL-8012U* setup. For the second ride, it uses the new *SimpleRTK2B-F9R board*³. To enable a fair comparison, the same base trajectory prediction model (`ebike_model_type:const_a_yawr`) and prediction horizon (`t_pred: 2s`) was used. The EKF is tuned for the respective setups and only fuses the heading. Subsection 4.1.3 already showed that the current EKF does not improve the positioning and rather oscillates about the positions from the GNSS receiver. Hence, the GNSS positions are directly used as the initial positions for the trajectory prediction.

The visualized trajectories of the first ride have already been described in subsection 4.1.3 (see Figure 4.7).

Figure 5.1 depicts the trajectories with the new INS. The top left part shows a segment

²If the EKF is not used at all these falsifications should not appear. However, it might still take a while until the INS is warmed up and returns correct positions and attitudes from its internal sensor fusion.

³Ideally, these setups would have been measured in parallel to guarantee that the exact same path was ridden.

5. Evaluation

where the pedelec was ridden near the outer edge of the paved road, exactly as in the first experiment. Generally speaking, the start points of the trajectories originating from the

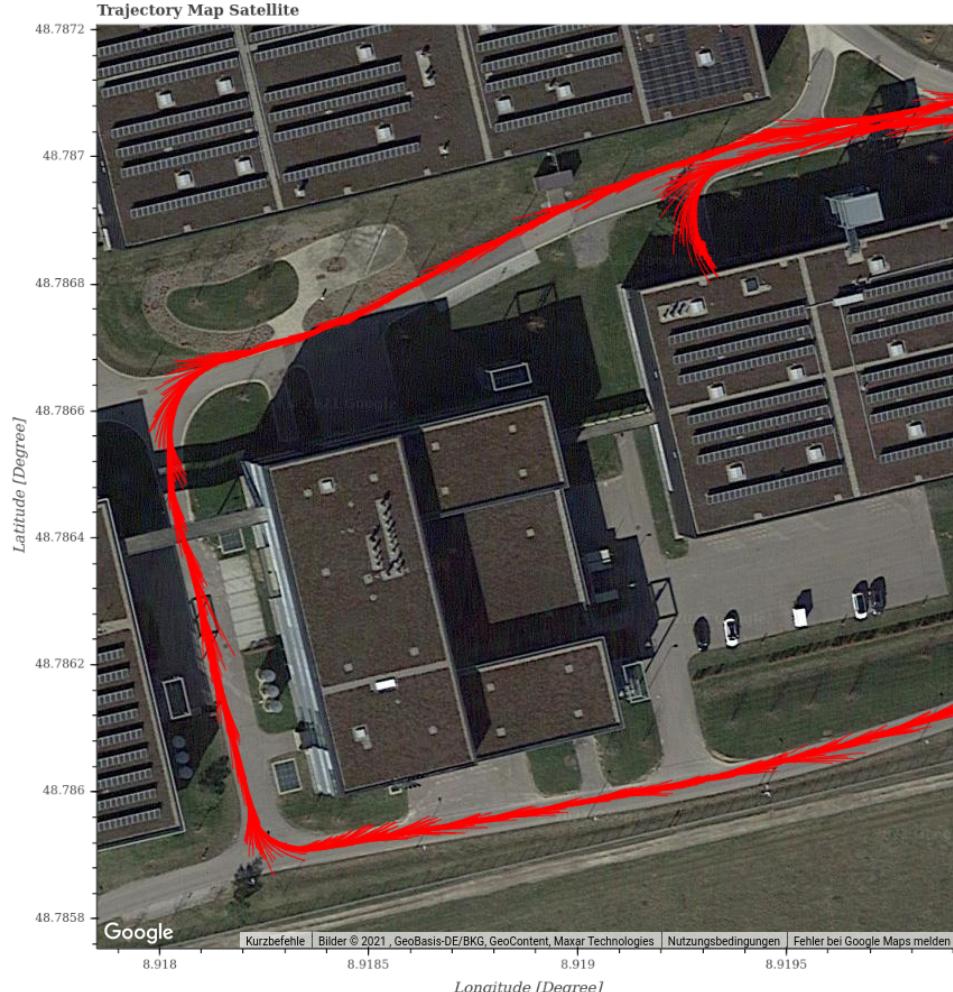


Figure 5.1.: SimpleRTK2B-F9R Noard: Accurate Positioning positioning

new INS are on the paved roads at all times. This is expected since the rider did not leave the paved segments at any time. With the old sensor setup, the trajectories would often incorrectly start somewhere in the grass. More specifically, the positioning offset is gone almost completely with new INS, and the start points of the trajectories are located near the outer edge of the paved road accordingly to the actually travelled path.

However, the swinging trajectories of different length resulting from the noisy accelerations and the oscillating yaw rates are still present even with a more sophisticated INS compared to the low-cost IMU.

section 4.6 already showed that the new INS solves the roll angle offset issue after sharp

curves. Highly accurate roll angles without offsets are especially important for the conversion from the roll to yaw rate as validated in subsubsection 4.8.4.4, where even slight offsets lead to rather significant deviations in the integrated heading changes. The roll angle offsets may cause a misinterpretation for any trajectory models relying on this information. For example, a constant roll angle of 5 degrees would physically mean the cyclist is leaning slightly to the right, e.g. for a long curve with a large radius. Furthermore, the novel trapezoidal extrapolation logic, discussed in subsubsection 4.8.4.5, relies on truthful roll angles to detect a curve and switch from the low-dynamic model to the high dynamic model. With this model, the turn marked with the red arrow in Figure 4.19a might almost be missed completely. Note how the trajectories of the low-dynamic model (here `const_speed_heading` point outward of the curve, resulting in large prediction errors. Compare this to the next curve in Figure 4.19a, marked with the green arrow, where the roll angle threshold works and the high-dynamic model using the trapezoidal extrapolation greatly improves the curve prediction. Despite a relative roll angle difference of up to 12 degrees from the curve entry (red line) to its minimum roll angle, the absolute roll angle of only -7 degrees barely crosses the roll angle threshold at -6 degrees.

For these reasons, the new INS (*SimpleRTK2B-F9R board*) replaces the initial GNSS (*Navilock NL-8012U*) and IMU (*BNO055-USB-Stick*) for all of the following evaluations. Furthermore, only the fused heading (and not also the position) of the tuned EKF by [55] is used. These settings correspond to the following parameters of the global configuration file `global_params.yaml`:

- `IMU_type: simpleRTK2B`
- `GPS_type: simpleRTK2B`
- `ebike_heading_src: kf`
- `ebike_pos_src: gps`

These parameters then only launch the relevant ROS nodes and topics as already depicted in Figure 4.17.

5.1.3. Influence of Route

This subsection investigates the influence of different types of routes on the accuracy of the same trajectory prediction model.

Figure 5.2 compares two possible routes to travel from one location to another. The final destination of the first trip ends at the start point of the second trip. Similarly, the end point of the second trip marks the starting location of the first trip. The first route in Figure 5.2a mainly traverses modern neighborhoods and a small city center with many sharp and blind turns as well that required braking maneuvers to check and wait for oncoming traffic or evade pedestrians. The second route contains less urban riding and instead more country

5. Evaluation

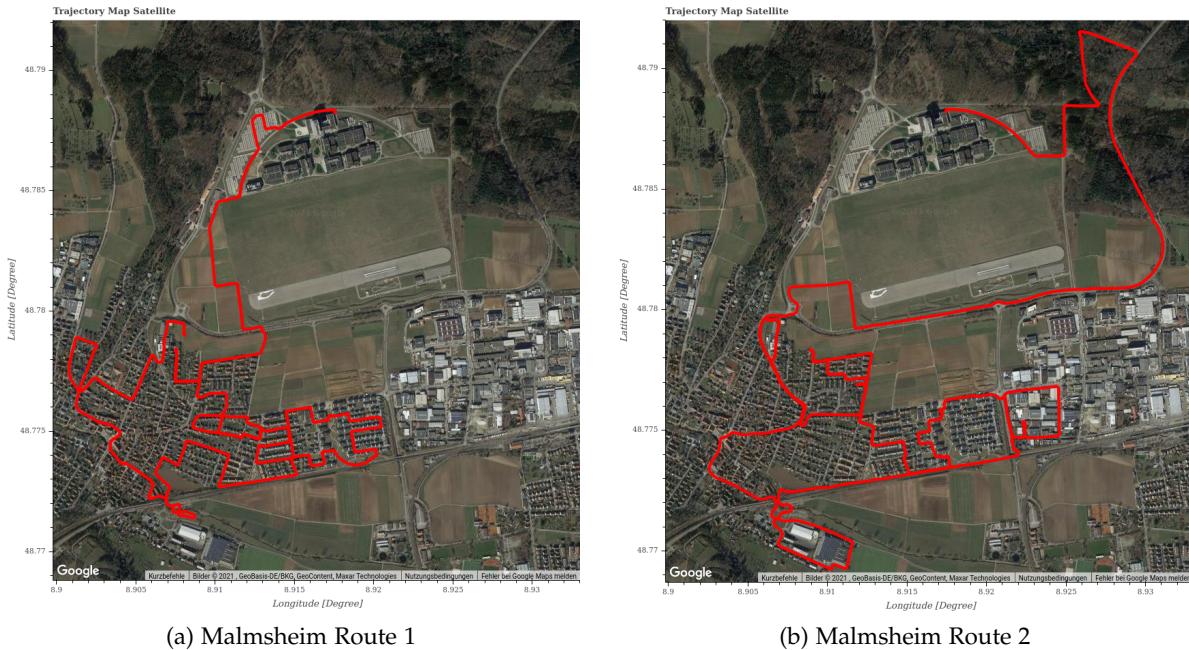


Figure 5.2.: Overview of Two Routes through Malmsheim

roads and forest sections with less curves and braking maneuvers. However, the forest section features gravel instead of pavement, and the trees may cause more GNSS outages. Table 5.3 compares the percentage at which specific road segments occur in relation to each total trip as explained in subsection 4.8.1.

Riding Section	Conditions	Malmshaus Route 1	Malmshaus Route 2
All	None	100.00%	100.00%
Straights	$\Delta\psi_{\min/\max} < 25 \text{ deg}$ AND $v > 1 \frac{\text{km}}{\text{h}}$	76.88%	77.48%
Curves	$\Delta\psi_{\min/\max} > 25 \text{ deg}$ AND $v > 1 \frac{\text{km}}{\text{h}}$	23.04%	21.72%
Unstable Pedaling	$1 < v \leq 17.4 \frac{\text{km}}{\text{h}}$ AND $\tau > 0.2 \text{ N m}$	12.96%	8.05%
Stable Pedaling	$17.4 \frac{\text{km}}{\text{h}} < v \leq 25.0 \frac{\text{km}}{\text{h}}$ AND $\tau > 0.2 \text{ N m}$	40.09%	36.44%
Slightly Unstable Pedaling	$v > 25.0 \frac{\text{km}}{\text{h}}$ AND $\tau > 0.2 \text{ N m}$	9.18%	23.07%
Unstable Coasting	$1 < v \leq 17.4 \frac{\text{km}}{\text{h}}$ AND $\tau < 0.2 \text{ N m}$	11.51%	5.02%
Stable Coasting	$17.4 \frac{\text{km}}{\text{h}} < v \leq 25.0 \frac{\text{km}}{\text{h}}$ AND $\tau < 0.2 \text{ N m}$	18.46%	9.74%
Slightly Unstable Coasting	$v > 25.0 \frac{\text{km}}{\text{h}}$ AND $\tau < 0.2 \text{ N m}$	6.19%	9.14%

Table 5.3.: Comparison of Riding Situations for Malmshaus Route 1 and 2

For the Malmshaus 1 trip, the straight and curve sections add up to 99.94% which means the stationary beginning and end phases were omitted successfully and the pedelec hit $v \leq 1.0 \frac{\text{km}}{\text{h}}$ for only $(1 - 0.9994) * 1724 \text{ s} \approx 1 \text{ s}$ during this trip. A similar result can be observed for the Malmshaus 2 trip.

These statistics demonstrate that for both routes the riding sections are composed mostly of (nearly) straight segments with a frequency of more than 75%. This means, only every fourth

5. Evaluation

segment is considered a curve, and that the averaged statistics over the whole trip indeed seem to favor simplistic models that are only good at predicting straights. Surprisingly, the route choice does not play a huge role in this comparison when it comes to the frequency of straights and curves: They only differ by roughly a single percentage point.

As expected, the Malmsheim route 1 that involves more urban riding shows a larger percentage in low-speed pedaling and coasting sections. In general, the Malmsheim 1 route contains more coasting sections. A reasonable explanation might be that the densely populated neighborhoods and city center offers many situations where the cyclist cannot see the course of the road. As a result, the cyclist generally rides slower by coasting more often or stops pedalling to prepare for tight turns or to react faster to unforeseen events.

The most notable difference lies in the *Slightly Unstable Pedaling* sections. The Malmsheim 2 route generally involves a better visibility, less traffic and slightly fewer turns which means the cyclist can continuously pedal at high speeds more often.

The Malmsheim 1 and Malmsheim 2 routes are evaluated over selected riding sections from Table 5.3 using the same base trajectory prediction model (`const_a_yawr`) for $t_{\text{pred}} = 2 \text{ s}$. The trajectory prediction errors are reported in Table 5.4 and Table 5.5, respectively.

Riding Sections Malmsheim 1	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
All	0.16	2.66	0.13	1.79	0.10	2.67	0.59	11.43
Straights	0.07	2.7	0.09	1.26	0.06	2.72	0.43	7.01
Curves	0.46	2.46	0.28	2.95	0.23	2.47	1.19	20.06
Unstable Pedaling	0.52	2.56	0.36	2.72	0.18	2.55	1.97	20.76
Stable Coasting	0.10	2.88	0.01	1.81	0.06	2.88	-0.18	9.78

Table 5.4.: Trajectory Prediction Accuracies for Different Sections of Malmsheim 1 Trip with `const_a_yawr` Model and $t_{\text{pred}} = 2 \text{ s}$

Riding Sections Malmsheim 2	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
All	0.15	2.97	0.15	1.74	0.12	2.98	0.85	12.08
Straights	0.13	3.08	0.12	1.35	0.11	3.10	0.50	6.55
Curves	0.22	2.59	0.27	2.71	0.13	2.60	1.13	16.64
Unstable Pedaling	0.28	2.70	0.35	2.52	0.07	2.78	1.82	18.07
Stable Coasting	0.15	4.01	0.14	2.21	0.18	3.97	0.61	12.71

Table 5.5.: Trajectory Prediction Accuracies for Different Sections of Malmsheim 2 Trip with `const_a_yawr` Model and $t_{\text{pred}} = 2 \text{ s}$

Since the tracks were ridden by the same rider on the same day with stable weather directly after each other, cyclist and environmental factors such as riding style, tire pressure, temperature, humidity etc. can be assumed to stay constant and thus not affect the comparison.

On a general note, the error deviations for the lateral metrics $\sigma_{\Delta y}$ and $\mu_{\Delta \sigma}$ double for both

models when comparing straights the curves. These lateral metrics are comparatively high in the sections of unstable pedalling, which represents nearly the worst-case for the yaw rate signal: the rider typically has to pedal hard to start accelerating and needs to counter-balance a lot. This amounts to large oscillations in the yaw rate signal, as discussed in subsection 4.1.2, that is used for predicting the lateral component. The stable coasting sections depict better lateral metrics $\sigma_{\Delta y}$ and $\mu_{\Delta \sigma}$ as the pedelec is self-stable which requires less balancing maneuvers and additionally does not swing because of pedaling.

The prediction models perform quite similar across both routes. This makes sense as Table 5.3 determined a similar distribution of riding sections for both routes. Interestingly, the model generally performs worse on the second route. Especially, deterioration for the Malmsheim 2 route in all riding sections, ranging from 0.13 m up to 1.13 m. This may be explained by the gravel surface in the forest section that adds even more noise to the signals, especially the acceleration signal used for the longitudinal prediction. Judging by the 0.24 cm gap in $\sigma_{\Delta y}$, the more frequent tight turns in Malmsheim 1 route seem to challenge the prediction model more.

This section confirms that straight road segments are greatly over-represented as assumed in subsection 4.1.4, and the ratio is nearly independent of the route choice. This showed the usefulness of segmenting the trip into sections for a more detailed analysis. As a result, the evaluation concludes that urban routes such as Malmsheim 1 route may challenge the models more regarding the lateral prediction with "tougher" curves and more low-speed unstable speed ranges. On the other hand, countryside routes such as the Malmsheim 2 route may challenge the longitudinal component more because of worse road conditions. Since the collision warning system is mostly designed for paved surfaces, the Malmsheim 1 route is used in all subsequent experiments (as a whole or in parts).

5.1.4. Influence of Selected Sensor Signal for Longitudinal Component

This subsection evaluates the selection of the signal and prediction method for the longitudinal component. In all experiments, the lateral component models remains constant and same as the base `const_a_yawr` model. The first sub-subsection analyzes the magnitude of improvement in the trajectory prediction errors when using the high-resolution WSS of the ABS compared to the base WSS of the HMI. Then, a sub-subsection investigates the impact of the extrapolation parameters on the trajectory prediction performance and tries to find an appropriate combination that works on most riding sections. Finally, this "optimal" extrapolation method is compared over the Malmsheim 1 route to the base and other promising longitudinal components for $t_{\text{pred}} = 2\text{ s}$.

5.1.4.1. Influence of High-Resolution Wheel Speed Sensor

This sub-subsection evaluates how beneficial the usage of a high-resolution speed signal really is compared to a standard 1-impulse WSS. The evaluations are performed for both prediction methods and divided into different speed ranges.

5. Evaluation

Table 5.6 compares the trajectory prediction errors of models that use different WSS and prediction methods (constant and extrapolation) for $t_{\text{pred}} = 2 \text{ s}$, averaged over the entire Malmsheim 1 trip.

Model	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
const_speed_const_yawr HMI WSS	-0.32	1.77	0.15	1.79	-0.31	1.58	0.79	14.21
const_speed_const_yawr ABS WSS	0.18	1.22	0.10	1.60	0.14	1.02	0.54	11.55
extrapol_speed_const_yawr HMI WSS	-0.31	1.60	0.12	2.02	-0.32	1.37	0.69	15.23
extrapol_speed_const_yawr ABS WSS	0.21	0.99	0.08	1.69	0.14	0.84	0.47	11.75

Table 5.6.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip

For both prediction methods, all longitudinal metrics are clearly improved when switching from the HMI to ABS WSS. The precision $\sigma_{\Delta x}$ clearly improves by 0.55 cm when holding the instantaneous speed value constant over the prediction horizon or even up to 0.61 m when extrapolating the most recent speed values. Similarly large gaps occur in the length difference $\sigma_{\Delta l}$. When replacing the HMI with the ABS WSS, the sign of all longitudinal accuracies changes from negative to positive, and the magnitude shrinks. This means the predicted trajectories shift from "way too short" to "too long". In terms of safety, a slight overestimation of the trajectories are preferred for the collision warning system. Otherwise, the system may wrongfully conclude a crash will not occur because it underestimates the trajectory length. Moreover, the EKF for the heading and position estimation takes longer to converge using the HMI WSS. However, a sufficiently large section at the start is omitted to prevent this from affecting the results.

Table 5.7 and Table 5.8 investigate the performance difference of using HMI and ABS WSS in the const_speed_const_yawr model more detailed for different speed ranges and $t_{\text{pred}} = 2 \text{ s}$.

HMI WSS: Speed Ranges	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
All	-0.32	1.77	0.15	1.79	-0.31	1.58	0.79	14.21
$1 < v \leq 7.2 \frac{\text{km}}{\text{h}}$	1.08	2.65	-0.28	3.1	2.24	2.11	2.25	40.65
$7.2 \frac{\text{km}}{\text{h}} < v \leq 17.4 \frac{\text{km}}{\text{h}}$	0.25	2.56	0.42	2.6	0.35	2.11	2.59	25.29
$17.4 \frac{\text{km}}{\text{h}} < v \leq 25 \frac{\text{km}}{\text{h}}$	-0.45	1.56	0.07	1.48	-0.48	1.42	0.24	8.77
$v > 25 \frac{\text{km}}{\text{h}}$	-0.56	1.08	0.1	1.45	-0.59	1.01	0.39	6.36

Table 5.7.: Trajectory Prediction Accuracies for Different Speed Ranges of Malmsheim 1 Trip with HMI WSS, const_speed_const_yawr Model and $t_{\text{pred}} = 2 \text{ s}$

5. Evaluation

ABS WSS: Speed Ranges	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
All	0.18	1.22	0.1	1.6	0.14	1.02	0.54	11.55
$1 < v \leq 7.2 \frac{\text{km}}{\text{h}}$	0.67	1.08	-0.34	1.72	1.05	1.13	-3.53	21.09
$7.2 \frac{\text{km}}{\text{h}} < v \leq 17.4 \frac{\text{km}}{\text{h}}$	0.39	1.73	0.24	1.99	0.33	1.36	1.81	18.52
$17.4 \frac{\text{km}}{\text{h}} < v \leq 25 \frac{\text{km}}{\text{h}}$	0.11	1.07	0.07	1.49	0.05	0.9	0.24	8.44
$v > 25 \frac{\text{km}}{\text{h}}$	0.11	0.78	0.11	1.3	0.07	0.72	0.41	5.33

Table 5.8.: Trajectory Prediction Accuracies for Different Speed Ranges of Malsheim 1 Trip with ABS WSS, const_speed_const_yawr Model and $t_{\text{pred}} = 2 \text{ s}$

As expected, the ABS WSS clearly outperforms the standard WSS in every speed range with the most notable gap in the very low-speed range $1 < v \leq 7.2 \frac{\text{km}}{\text{h}}$: The HMI WSS only receives one impulse per revolution at extremely slow wheel speeds which leads to a slow update of the speed signal and results in a pronounced step-like behavior of the signal as already depicted in A.3b. The higher the speeds get, the better the HMI WSS performs relative to the ABS WSS, which performs roughly similar for all speed ranges.

The evaluations clearly confirmed that a high-resolution WSS from the ABS significantly improves the longitudinal performance of the trajectory prediction models for both prediction methods as hypothesized in subsubsection 4.8.3.2. As expected, the biggest improvement occurs in low speed ranges. Hence, all further models that involve the pedelec speed in any form will use the ABS WSS.

5.1.4.2. Influence of Extrapolation Parameters

This subsection investigates the influence of the extrapolation parameters on the errors of the trajectory prediction models. It evaluates the limitation on the extrapolated speeds, discussed in subsubsection 4.8.3.1, before attempting to find an appropriate combination of $t_{\text{in}_{\text{long}}}$ and $n_{\text{pg}_{\text{long}}}$. Once again, only the ABS speed signals of the longitudinal component are extrapolated while the lateral component is kept constant and same to the base const_a_yawr model.

Table 5.9 compares the trajectory prediction errors when extrapolating speed with a limit of $v_{\min} = 0 \frac{\text{m}}{\text{s}}$ and $v_{\max} = 30 \frac{\text{m}}{\text{s}}$ for $t_{\text{pred}} = 2 \text{ s}$ averaged over a 372 s portion of the Malsheim 1 trip (Figure 4.28 shows a small section of this exact comparison). This experiment deliberately chooses a "bad" parameter combination of $t_{\text{in}_{\text{long}}}$ and $n_{\text{pg}_{\text{long}}}$ to show the effect of the extrapolation limit more clearly. The first half of the table represents the errors without any extrapolation limits, whereas the second half shows the improvement by limiting the extrapolation from below and above.

extrapol_speed_const_yawr	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
$-\infty < v < \infty$, $t_{\text{in}_{\text{long}}} = 0.25 \text{ s}$, $n_{\text{pg}_{\text{long}}} = 3$	11.91	522.39	2.89	125.83	-363.92 385.06	-1.56	120.56	
$0.0 \frac{\text{m}}{\text{s}} \leq v \leq 30.0 \frac{\text{m}}{\text{s}}$, $t_{\text{in}_{\text{long}}} = 0.25 \text{ s}$, $n_{\text{pg}_{\text{long}}} = 3$	-11.37	21.85	0.17	6.94	-11.93	22.41	0.27	12.58

Table 5.9.: Trajectory Prediction Errors when Extrapolating Speed with a Limit of $v_{\min} = 0 \frac{\text{m}}{\text{s}}$ and $v_{\max} = 30 \frac{\text{m}}{\text{s}}$ for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malsheim 1 Trip

5. Evaluation

All metrics show a clear improvement: As expected, $\mu_{\Delta l}$ gets shifted closer to zero, because the backward trajectories do not exist anymore that results in negative length differences. The precision $\sigma_{\Delta x}$ and $\sigma_{\Delta l}$ also improves notably by a factor of over 65. The longer t_{pred} , the larger the improvements become. On the other hand, the lower $n_{\text{pg, long}}$ and longer $t_{\text{in, long}}$ are set the improvements become less noticeable because the extrapolated signals are generally less steep and hence do not result in the physically illogical magnitudes very often after just a few seconds.

Table 5.10 compares the prediction accuracies and precisions of the `|extrapol_speed_const_yawr|` model for $t_{\text{pred}} = 2 \text{ s}$ with different combinations of input durations $t_{\text{in, long}}$ and polynomial degree $n_{\text{pg, long}}$ averaged over a 372 s long section of the Malmsheim 1 trip. As demonstrated to be effective above, the extrapolation is bound between $0.0 \frac{\text{m}}{\text{s}} \leq v \leq 30.0 \frac{\text{m}}{\text{s}}$ for all of the models in this comparison to clip physically illogical speeds.

<code>extrapol_speed_const_yawr</code>	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
$t_{\text{in, long}} = 0.25 \text{ s}, n_{\text{pg, long}} = 1$	0.17	1.53	0.10	1.66	0.10	1.49	0.43	11.77
$t_{\text{in, long}} = 0.50 \text{ s}, n_{\text{pg, long}} = 1$	0.16	0.97	0.10	1.70	0.11	0.86	0.42	12.36
$t_{\text{in, long}} = 0.75 \text{ s}, n_{\text{pg, long}} = 1$	0.19	0.89	0.10	1.71	0.11	0.77	0.4	12.37
$t_{\text{in, long}} = 1.00 \text{ s}, n_{\text{pg, long}} = 1$	0.17	0.89	0.09	1.76	0.10	0.80	0.39	12.64
$t_{\text{in, long}} = 2.00 \text{ s}, n_{\text{pg, long}} = 1$	0.15	1.08	0.09	1.78	0.09	1.02	0.38	12.94
$t_{\text{in, long}} = 4.00 \text{ s}, n_{\text{pg, long}} = 1$	0.10	1.51	0.09	1.73	0.07	1.49	0.42	12.65
$t_{\text{in, long}} = 8.00 \text{ s}, n_{\text{pg, long}} = 1$	0.08	2.40	0.05	1.60	0.07	2.37	0.27	11.99
$t_{\text{in, long}} = 0.25 \text{ s}, n_{\text{pg, long}} = 2$	-3.15	11.65	0.17	3.93	-3.41	11.99	0.48	12.12
$t_{\text{in, long}} = 0.50 \text{ s}, n_{\text{pg, long}} = 2$	-0.37	5.58	0.12	2.37	-0.48	5.74	0.49	11.71
$t_{\text{in, long}} = 0.75 \text{ s}, n_{\text{pg, long}} = 2$	0.10	2.69	0.11	1.91	0.03	2.77	0.55	11.74
$t_{\text{in, long}} = 1.00 \text{ s}, n_{\text{pg, long}} = 2$	0.17	1.83	0.12	1.85	0.08	1.85	0.57	12.28
$t_{\text{in, long}} = 2.00 \text{ s}, n_{\text{pg, long}} = 2$	0.23	1.17	0.10	1.84	0.10	1.20	0.43	12.47
$t_{\text{in, long}} = 4.00 \text{ s}, n_{\text{pg, long}} = 2$	0.18	1.32	0.09	1.96	0.08	1.40	0.30	13.31
$t_{\text{in, long}} = 8.00 \text{ s}, n_{\text{pg, long}} = 2$	0.06	1.93	0.10	1.79	0.05	1.96	0.49	12.63
$t_{\text{in, long}} = 0.25 \text{ s}, n_{\text{pg, long}} = 3$	-11.32	21.86	0.17	6.87	-11.87	22.42	0.27	12.5
$t_{\text{in, long}} = 0.50 \text{ s}, n_{\text{pg, long}} = 3$	-7.53	17.14	0.22	5.58	-7.98	17.59	0.44	12.86
$t_{\text{in, long}} = 0.75 \text{ s}, n_{\text{pg, long}} = 3$	-3.75	11.84	0.14	4.44	-4.10	12.27	0.33	12.66
$t_{\text{in, long}} = 1.00 \text{ s}, n_{\text{pg, long}} = 3$	-1.41	7.89	0.11	3.27	-1.62	8.22	0.52	12.08
$t_{\text{in, long}} = 2.00 \text{ s}, n_{\text{pg, long}} = 3$	0.12	2.45	0.14	2.05	0.00	2.56	0.55	12.36
$t_{\text{in, long}} = 4.00 \text{ s}, n_{\text{pg, long}} = 3$	0.24	1.73	0.08	2.02	0.08	1.91	0.33	12.42
$t_{\text{in, long}} = 8.00 \text{ s}, n_{\text{pg, long}} = 3$	0.14	2.03	0.14	1.98	0.06	2.19	0.45	12.41

Table 5.10.: Trajectory Prediction Errors for Speed Extrapolation Parameter Combinations with $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip

Note that changing the parameters of the longitudinal extrapolation has as expected large effect on its corresponding longitudinal metrics $\mu_{\Delta x}$, $\sigma_{\Delta x}$, $\mu_{\Delta l}$ and $\sigma_{\Delta l}$. While a longitudinal extrapolation generally effects the lateral metrics $\mu_{\Delta y}$, $\sigma_{\Delta y}$, $\mu_{\Delta \sigma}$ and $\sigma_{\Delta \sigma}$ much less, these errors become noticeable the larger the error of the longitudinal component and angle. This makes

5. Evaluation

sense as the metrics are not fully decoupled due to their tangent relationship. Table 5.10 shows that higher degree polynomials generally have a negative effect on the relevant longitudinal metrics $\mu_{\Delta x}$, $\sigma_{\Delta x}$, $\mu_{\Delta l}$ and $\sigma_{\Delta l}$ despite potentially fitting recent points better. This is especially the case for short input durations where the most recent points might be a bad indicator for the future signal, and the positional error with respect to time rises from linear a (constant speed) to $n_{pg_{long}} + 1$ (extrapolated speed with $n_{pg_{long}}$) relationship (refer to Figure 4.28 again). This higher order dependence of the positional error on time can be counteracted with longer $t_{in_{long}}$ that fit more data points, thus averaging over a bigger window and leading to less extreme slopes. However, this causes a slow response to changing trends in the signal causes a significant lag that may result in large errors. Figure 5.3 shows a third order extrapolation with $t_{in_{long}} = 4$ s for $t_{pred} = 2$ s.

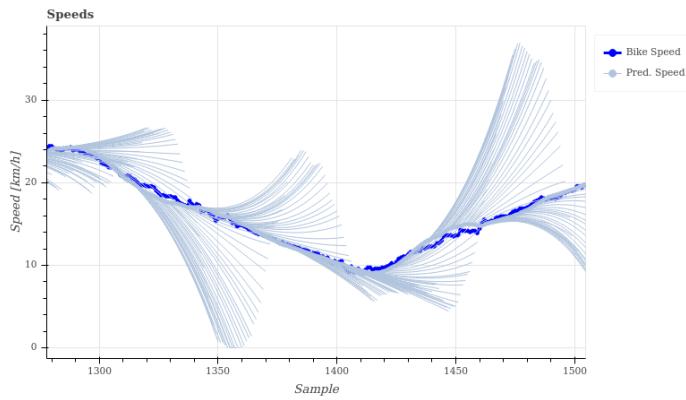


Figure 5.3.: Maximum Roll Angles of All Curves in Malmsheim 1 Trip

On the other hand, reducing the polynomial to a line and setting the shortest possible input duration does not result in the best performance despite being able to quickly respond to a changing signal. For example, Figure shows how $n_{pg_{long}} = 1$ and $t_{in_{long}} = 0.25$ s overemphasize a speed ripple resulting from pedalling when extrapolated for $t_{pred} = 2$ s.

5. Evaluation

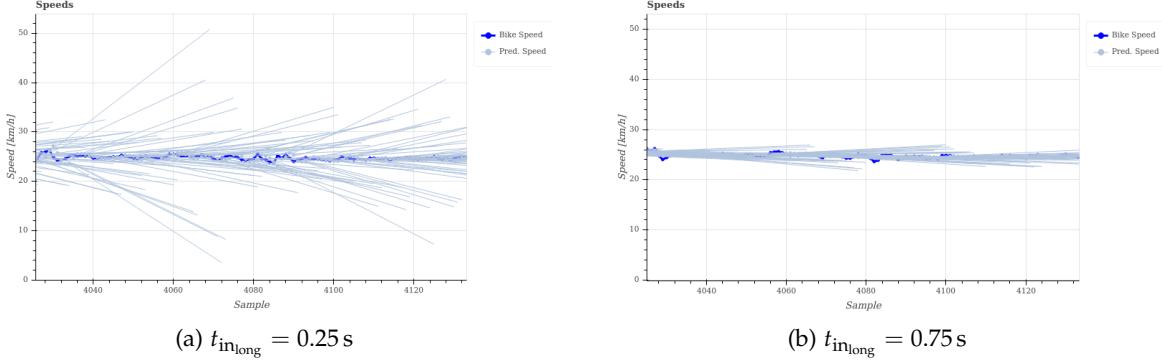


Figure 5.4.: Speed Ripple Linearly Extrapolated with Different t_{pred} Parameters

Note how the extrapolated speeds (light blue) are off up to 25 km/h at the end of the two second prediction horizon, given a nearly constant speed (dark blue). Figure 5.4b uses $t_{inlong} = 0.75\text{ s}$ which smooth out those ripples and lead to far better speed predictions. Therefore, it makes sense that the best combination consists of the lowest polynomial degree $n_{pg_{long}} = 1$ and moderately short $t_{inlong} = 0.75\text{ s}$ opposed to extremely short or long input durations. The best and worst parameter combinations of Table 5.10 are shown in Table 5.12 again.

Table 5.11 and Table 5.13 investigate the errors of the best and worst parameter combinations of Table 5.10 for $t_{pred} = 2\text{ s}$ (see Table 5.12) in the context of $t_{pred} = 1\text{ s}$ and $t_{pred} = 4\text{ s}$, respectively.

<code>extrapol_speed_const_yaw</code>	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
$t_{inlong} = 0.75\text{ s}, n_{pg_{long}} = 1$	0.19	0.22	0.05	0.45	0.18	0.22	0.47	6.08
$t_{inlong} = 0.25\text{ s}, n_{pg_{long}} = 3$	-3.40	8.18	0.09	1.51	-3.46	8.27	0.49	6.53

Table 5.11.: Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{pred} = 1\text{ s}$ Averaged over 372 s of Malmseim 1 Trip

<code>extrapol_speed_const_yaw</code>	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
$t_{inlong} = 0.75\text{ s}, n_{pg_{long}} = 1$	0.19	0.89	0.10	1.71	0.11	0.77	0.40	12.37
$t_{inlong} = 0.25\text{ s}, n_{pg_{long}} = 3$	-11.32	21.86	0.17	6.87	-11.87	22.42	0.27	12.5

Table 5.12.: Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{pred} = 2\text{ s}$ Averaged over 372 s of Malmseim 1 Trip

5. Evaluation

extrapol_speed_const_yaw	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
$t_{in_{long}} = 0.75$ s, $n_{pg_{long}} = 1$	0.73	4.71	0.29	6.30	0.12	3.41	0.19	27.85
$t_{in_{long}} = 0.25$ s, $n_{pg_{long}} = 3$	-25.36	48.91	1.47	23.17	-29.11	50.74	0.28	24.57

Table 5.13.: Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{pred} = 4$ s Averaged over 372 s of Malsheim 1 Trip

Note how for all prediction horizons the first parameter combination resulted in significantly smaller trajectory prediction errors than the second parameter combination. Obviously, the longer the prediction horizon is, the bigger the gap gets between these parameter combinations. The trends that low $n_{pg_{long}}$ and short $t_{in_{long}}$ generally lead to good results while large $n_{pg_{long}}$ and short $t_{in_{long}}$ lead to bad results thus remain also for other prediction horizons.

The aforementioned effects show even more on the acceleration signals because the positional error with respect to time rises from quadratic to $n_{pg_{long}} + 2$. For comparison with extrapolated speed, Table 5.14 shows the magnitudes of extrapolated accelerations for the best and worst parameter combinations from Table 5.10 for $t_{pred} = 2$ s.

extrapol_a_const_yawr	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
$t_{in_{long}} = 0.75$ s, $n_{pg_{long}} = 1$	1.48	3.47	0.21	3.96	0.68	3.43	0.84	24.46
$t_{in_{long}} = 0.25$ s, $n_{pg_{long}} = 3$	-1276.18	2935.31	55.97	2472.8	-1959.85	3536.69	0.97	38.17

Table 5.14.: Trajectory Prediction Errors for Extreme Acceleration Extrapolation Parameter Combinations with $t_{pred} = 2$ s Averaged over 372 s of Malsheim 1 Trip

The magnitudes of errors from an extrapolated acceleration are significantly larger compared to the extrapolation of the speed with the same corresponding parameters. As extrapolating speed instead of acceleration generally makes more sense and yields to better results, an extrapolation of the longitudinal component will only be done on the speed signal from now on.

This sub-subsection confirms the hypotheses from subsubsection 4.8.3.1 that physically illogical speeds arising from extrapolation and short input durations coupled with high polynomial degrees may lead to terrible trajectory prediction errors. Moreover, the evaluations show that these negative effects grow with larger prediction horizons or when using the noisier acceleration signal instead. As a result, further model comparisons that involve an extrapolated longitudinal component will only linearly extrapolate the speed signal (not acceleration) with short input durations and speed limits of $v_{min} = 0$ $\frac{m}{s}$ and $v_{max} = 30$ $\frac{m}{s}$.

5.1.4.3. Comparison of Selected Longitudinal Signals

This subsection compares the influence of the signal selection and processing of the longitudinal component on the trajectory prediction errors. For this purpose, the lateral component is

5. Evaluation

kept constant over all experiments while only the longitudinal one is changed. The extrapolation parameters are set as described at the end of the previous section () .

As noted in subsubsection 5.1.4.2, an extrapolation of the longitudinal component is most useful when linearly extrapolating the speed signal. Furthermore, as noted in subsubsection 4.8.3.2, a constant speed decreases the relationship of the predicted positions with respect to time to linear, suggesting it may outperform the linearly extrapolated speed and constant acceleration model for large t_{pred} . Therefore, Table 5.21, Table 5.16 and Table 5.17 compare a linearly extrapolated and constant speed component with the constant acceleration base for $t_{\text{pred}} = 1 \text{ s}$, $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$, respectively.

Model	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
const_a_const_yawr	0.20	0.66	0.05	0.40	0.20	0.67	0.53	5.16
extrapol_speed_const_yawr	0.20	0.23	0.05	0.41	0.20	0.23	0.53	5.38
const_speed_const_yawr	0.20	0.29	0.05	0.42	0.20	0.28	0.54	5.58

Table 5.15.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 1 \text{ s}$ Averaged over 1724 s Malmsheim Trip

Model	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
const_a_const_yawr	0.16	2.65	0.13	1.80	0.10	2.66	0.57	11.52
extrapol_speed_const_yawr	0.20	0.99	0.08	1.68	0.14	0.84	0.48	11.66
const_speed_const_yawr	0.18	1.18	0.10	1.56	0.14	1.00	0.54	11.17

Table 5.16.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim Trip

Model	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
const_a_const_yawr	0.23	10.6	0.43	7.11	-0.43	10.14	0.72	26.71
extrapol_speed_const_yawr	0.72	5.61	0.21	6.08	0.17	3.87	0.33	27.17
const_speed_const_yawr	0.41	5.79	0.29	5.63	0.20	4.14	0.67	25.89

Table 5.17.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 4 \text{ s}$ Averaged over 1724 s Malmsheim Trip

Both of the new trajectory prediction models that utilize the ABS WSS show a significant improvement in all of the longitudinal precision metrics $\sigma_{\Delta x}$ and $\sigma_{\Delta l}$ over all prediction horizons compared to the base const_a_yawr model. These precisions show roughly half the magnitude compared to the baseline or even less. The improvements in the accuracies $\mu_{\Delta x}$ and $\mu_{\Delta l}$ generally remain small, and the case of $\mu_{\Delta x}$ for $t_{\text{pred}} = 4 \text{ s}$ even become worse. The longitudinal precisions of the linearly extrapolated speeds increase slightly more than quadratically with respect to the prediction horizon as expected. For example, doubling the prediction horizon from $t_{\text{pred}} = 2 \text{ s}$ to $t_{\text{pred}} = 4 \text{ s}$ more than quadruples $\sigma_{\Delta x}$ from 0.99 m to 5.61 m. Similarly, the length increases from 0.84 m to 3.87 m.

Contrary to the expectation, the constant speed does not behave anywhere near linear. In fact, it behaves more exponentially than the linearly extrapolated speed model.

Note that for longer t_{pred} the cross influence to the lateral metrics becomes noticeable. For example, the precisions $\sigma_{\Delta y}$ and $\sigma_{\Delta \sigma}$ are nearly identical for $t_{\text{pred}} = 1$ s among all models. However, for $t_{\text{pred}} = 4$ s, the `const_speed_const_yawr` model shows an improvement of 1.48 m in the lateral position precision $\sigma_{\Delta y}$. Generally speaking, the `const_speed_const_yawr` model

Judging only by the longitudinal metrics, the linearly extrapolated ABS speed signal is most suitable for the prediction of the longitudinal component. Unfortunately, the constant speed assumption statistically does not behave as well for long time horizons as expected. For this reason, the linearly extrapolated ABS speed is selected as the longitudinal component for all further model comparisons that will focus on the lateral prediction.

5.1.5. Influence of Selected Sensor Signal for Lateral Component

This subsection investigates the selection of the signal and prediction method for the lateral component further while keeping the longitudinal component constant. First, it evaluates whether the roll angle signal really is superior than the yaw rate for trajectory prediction. Afterwards, it carries out simple statistical evaluations of curve sections averaged over an entire trip. These statistical values are then used to set the parameters of the novel trapezoidal extrapolation, as introduced in subsubsection 4.8.4.5. Finally, the trapezoidal extrapolation method is compared to the base and other promising lateral components for $t_{\text{pred}} = 2$ s.

5.1.5.1. Influence of Roll Angle Instead of Yaw Rate

This sub-subsection examines whether the roll angle, due to its smooth signal characteristics, is superior to the yaw rate as the lateral trajectory prediction signal of a pedelec.

Table 5.21 compares the prediction errors from different lateral components while keeping the longitudinal components the same. The first model holds the instantaneous yaw rate constant over the prediction horizon which is then integrated in the motion primitive to heading deltas. The second model holds instantaneous roll angles constant and converts them to yaw rates by using Equation 4.26 before integrating it further into a heading deltas.

Model	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
extrapol_speed_const_yawr	0.20	0.99	0.08	1.68	0.14	0.84	0.48	11.66
extrapol_speed_const_rollangle	0.14	0.85	0.07	1.15	0.12	0.84	0.40	7.83

Table 5.18.: Trajectory prediction Accuracies for $t_{\text{pred}} = 2$ s averaged over 1724 s Malmsheim Trip

The results confirm that substituting the measured yaw rates with yaw rates computed from roll angles improves all of the lateral metrics. The most significant improvement occurs

5. Evaluation

in the precision of the y-error $\sigma_{\Delta y}$ by more than half a meter which amounts to a relative improvement of around 31% compared to the base. In fact, the roll angle approach even slightly improves all of the longitudinal metrics as well.

Figure 5.5 visually compares the predicted trajectories of the models from Table 5.21. The comparison shows that replacing the yaw rate with the roll angle improve the performance on both curves and straights significantly.

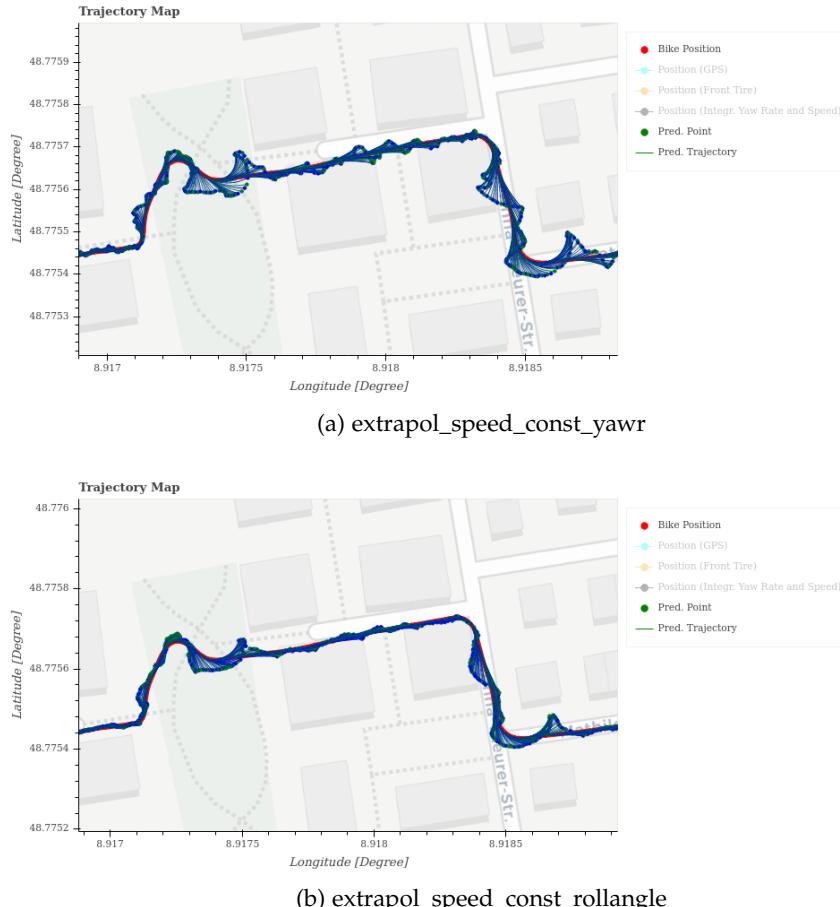


Figure 5.5.: Trajectory Comparison When Replacing Yaw Rate with Roll Angle for $t_{\text{pred}} = 2 \text{ s}$

Since the yaw rate signal computed from roll angles is generally smoother than the measured yaw rate, as shown in Figure 4.35, the constant roll angle model performs better than the constant yaw rate model as expected. A smoother signal is also better suited for extrapolation, as confirmed in the comparison of ABS and HMI wheel speed in subsubsection 5.1.4.1. Therefore, extrapolating the roll angles instead of the yaw rates will lead to a better performance as well.

This sub-subsection confirmed the hypothesis from subsubsection 4.8.4.2 that the roll angle sig-

5. Evaluation

nal is superior to the yaw rate for both prediction methods. Consequently, the roll angle signal will be used to evaluate the trapezoidal extrapolation method in the next sub-subsection.

5.1.5.2. Statistical Evaluation of Curves for Trapezoidal Extrapolation Parameters

This sub-subsection computes statistics of curves to help pick the best average values for the static parameters of the trapezoidal extrapolation. It also investigates how sensitive the trapezoidal model reacts to an inappropriate parameterization. As explained in the previous section, the roll rate is used as the lateral signal.

Figure 5.6 shows a histogram of all maximum roll angles for all curves in the entire Malsheim 1 trip, alongside with the mean and standard deviation.

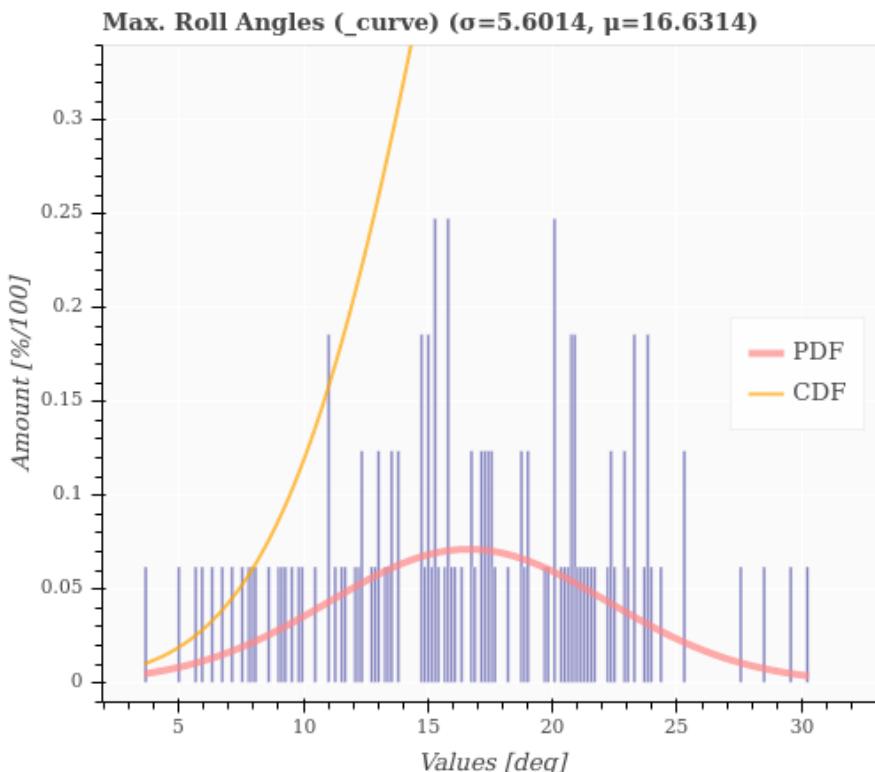


Figure 5.6.: Maximum Roll Angles of All Curves in Malsheim 1 Trip

It becomes clear that many of the curves are of low dynamics (low roll angles) which likely can be handled fine by a constant roll angle assumption. The trajectory prediction model thus uses a constant roll angle for the non-dynamic part (before the roll angle crosses threshold). Therefore, the switch to the trapezoidal extrapolation is done fairly late at around the mean value of $|\phi_{\min}| = 16.00$ deg in order to not overemphasize low dynamic curves too often

5. Evaluation

and be able parameterize the roll angle profile parameters a_y and t_{const} [s] more aggressively towards high dynamic curves.

Another statistical curve evaluation is run over a 372 s segment of the Malmsheim 1 trip. However, this time only those high dynamic curves ($|\phi_{\min}| > 16.00 \text{ deg}$) are evaluated to better capture the statistics of the high dynamic curves. Figure 5.7 shows the results of this evaluation.

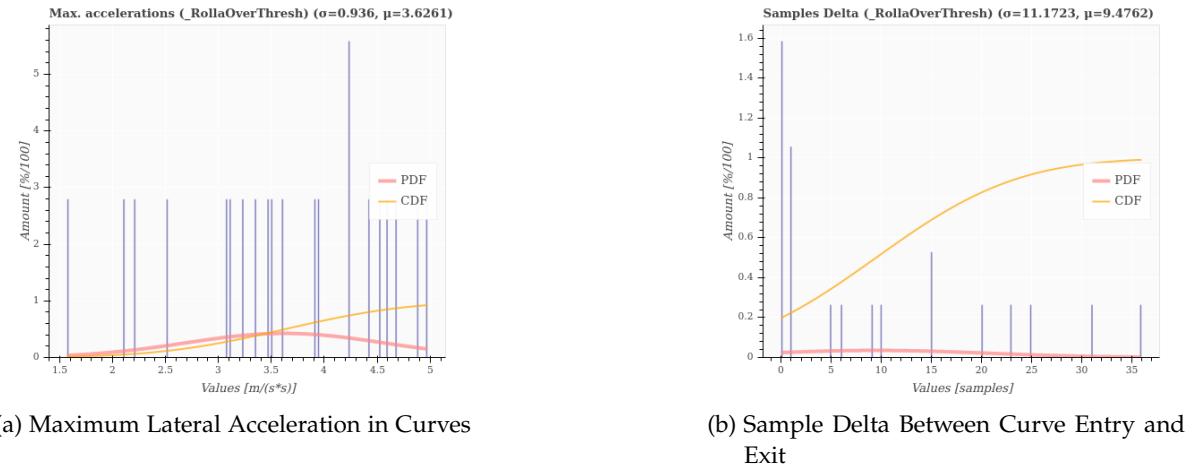


Figure 5.7.: Statistics of Curves Above $|\phi_{\min}| = 14.0 \text{ deg}$ for 372 s of Malmsheim 1 Trip

The remaining parameters of the roll angle profile are set close to the mean values of this statistical evaluation: $a_{y_{\max}} = 4.0 \frac{\text{m}}{\text{s}^2}$ and $t_{\text{const}} = 10 \text{ Samples} = 0.5 \text{ s}$.

The parameters concerning the extrapolation are set to $n_{pg_{\text{lat}}} = 1$, analogous to the extrapolation of the longitudinal component. However, the input duration is set to a shorter $t_{in_{\text{lat}}} = 0.25 \text{ s}$ as the roll angle signal is smoother than the speed signal and a slow response to changes in trend is undesired for high dynamic curves.

The following paragraphs investigate the sensitivity of the $a_{y_{\max}}$ and t_{const} parameters, while keeping the remaining ones constant. The average trajectory prediction errors of the good parameter combination from above is compared with a slight over- and underestimation of the trapezoidal roll angle profile. Table 5.19 summarizes the parameters for these experiments while Table 5.20 lists the average prediction errors over the Malmsheim 1 trip for $t_{\text{pred}} = 2 \text{ s}$.

extrapol_lim_speed_trap_roll	$n_{pg_{\text{long}}}$	$t_{in_{\text{long}}} [\text{s}]$	$n_{pg_{\text{lat}}}$	$t_{in_{\text{lat}}} [\text{s}]$	$\phi_{\min} [\text{deg}]$	$a_y [\frac{\text{m}}{\text{s}^2}]$	$t_{\text{const}} [\text{s}]$	$\Delta\psi_{\max} [\text{deg}]$
Roughly Average	1	0.75	trap.	0.25	16.00	4.00	0.5	N/A
Slightly overestimated	1	0.75	trap.	0.25	16.00	5.00	0.75	N/A
Slightly Underestimated	1	0.75	trap.	0.25	16.00	3.00	0.25	N/A

Table 5.19.: Average, Over- and Underestimated Parameters for extrapol_lim_speed_trap_roll Trajectory Prediction Model

5. Evaluation

extrapol_lim_speed_trap_roll	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
Roughly Average	0.06	0.86	0.10	0.99	0.07	0.86	0.50	7.62
Slightly overestimated	0.09	0.87	0.10	1.00	0.08	0.84	0.35	7.90
Slightly underestimated	0.14	0.88	0.16	1.10	0.06	0.88	0.64	8.37

Table 5.20.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 2$ s Averaged over 372 s of Malmsheim 1 Trip

In general, if the parameters are slightly off, the results only worsen a bit, which is probably explained in the fact that curves are generally underrepresented in real life trips compared to straights. In this case, overestimating $a_{y_{\max}}$ by 25% and t_{const} by 50% worsens the standard deviation of the lateral position error only by around 1% while the mean stays the same. However, underestimating these parameters by the same amount amounts to a roughly 11% error increase in the standard deviation and 60% increase in the mean of the lateral position error.

To summarize, experiments show that the trapezoidal extrapolation is fairly robust to slightly incorrectly guessed parameters if the roll angle threshold is picked rather high (more robust than originally anticipated in subsubsection 4.8.4.5). This could also indicate that the online adaptions effectively increase the robustness of the model in many cases. Moreover, a slight overestimation of the trapezoidal shape does not hurt the performance as much as a slight underestimation. Finally, the trapezoidal extrapolation parameters are set to $|\phi_{\min}| = 16.00$ deg, $a_{y_{\max}} = 4.0 \frac{\text{m}}{\text{s}^2}$ and $t_{\text{const}} = 10$ Samples = 0.5 s based on the results of a statistical evaluation over the curves. This set of parameters will be used for further comparisons.

5.1.5.3. Comparison of Selected Lateral Signals

This subsection compares the influence of the signal selection and processing of the lateral component on the trajectory prediction errors. For this purpose, the longitudinal component is kept constant over all experiments while only the lateral one is changed. As confirmed in subsubsection 5.1.5.1, the roll angle is more suitable than the yaw rate and thus the signal choice of the lateral component. The extrapolation parameters are set as described at the end of the previous section () .

Analogous to the speed limitation from subsubsection 4.8.3.1, the linear extrapolation of the absolute roll angle ϕ is clipped to $-|\phi_{\max}| < \phi < +|\phi_{\max}|$, where $|\phi_{\max}| = 40$ deg was obtained experimentally by riding extreme tight and fast curves on dry asphalt without falling. Similarly, the extrapolation of the heading ψ is limited to a maximum difference as follows $-|{}^S\omega_{z_{\max}} * t_{\text{pred}}| < \Delta\psi < +|{}^S\omega_{z_{\max}} * t_{\text{pred}}|$, where $\Delta\psi = \psi_{\text{final}} - \psi_{\text{initial}}$ and $|{}^S\omega_{z_{\max}}| = 95$ deg / s was the highest achievable instantaneous yaw rate. In both cases, these limits prevent the trajectories from spiralling extensively, which would be the case for steep signal gradients resulting from high instantaneously roll or yaw rates.

The extrapolation parameters of the lateral component are set as determined for the longitudinal component in subsubsection 5.1.4.2 to $t_{\text{in}_{\text{lat}}} = 1$ and $t_{\text{in}_{\text{lat}}} = 0.75$ s. However, the parameters

5. Evaluation

for trapezoidal extrapolation of the roll angle are set analogous to subsubsection 5.1.5.1, except the statistics were run over the entire Malmsheim 1 trip.

Table 5.21 summarizes the results from comparing these different lateral components average over the entire Malmsheim 1 route for $t_{\text{pred}} = 2 \text{ s}$.

Model	$\mu_{\Delta x} [\text{m}]$	$\sigma_{\Delta x} [\text{m}]$	$\mu_{\Delta y} [\text{m}]$	$\sigma_{\Delta y} [\text{m}]$	$\mu_{\Delta l} [\text{m}]$	$\sigma_{\Delta l} [\text{m}]$	$\mu_{\Delta \sigma} [\text{deg}]$	$\sigma_{\Delta \sigma} [\text{deg}]$
extrapol_speed_const_yawr	0.20	0.99	0.08	1.68	0.14	0.84	0.48	11.66
extrapol_speed_const_rollangle	0.14	0.85	0.07	1.15	0.12	0.84	0.40	7.83
extrapol_speed_const_heading	-0.13	0.99	0.14	1.70	0.05	0.87	0.64	12.48
extrapol_speed_extrapol_heading	0.56	2.30	0.10	2.26	0.23	1.03	0.57	23.04
extrapol_speed_extrapol_rollangle	0.22	0.89	0.08	0.94	0.16	0.87	0.35	6.69
extrapol_speed_trap_rollangle	0.14	0.92	0.08	1.04	0.12	0.89	0.45	7.48

Table 5.21.: Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip

The results confirm that holding or linearly extrapolating the heading worsens all of the lateral metrics, even when limiting it to physically logical values. Therefore, this signal generally does not seem to be useful for predicting the lateral component.

On the other hand, the models using the roll angle all improve all of the lateral metrics or stay at the baseline value as already discussed in subsubsection 5.1.5.1.

Interestingly, the `extrapol_speed_extrapol_roll` model with limitation to 40 degree outperforms even the `extrapol_speed_trap_rollangle` model, which is the second best model. This was not expected originally, and the `extrapol_speed_extrapol_roll` model will thus be compared for shorter and longer t_{pred} to find out whether this is somewhat of a coincidence for this specific prediction horizon.

This sub-subsection concludes that all of the available signal processing methods applied to the roll angle signal improve the trajectory predictions, confirming the effectiveness of using this signal once again. As a result, all three of these options will be included for further comparisons against the baseline and previous work in the next subsection.

5.1.6. Comparison of Trajectory Prediction Models

This subsection compares the novel trajectory prediction models of this thesis with the best-performing physics-based and ML-based models from the previous works by [17], [54] and [55]⁴. The first part compares the models quantitatively regarding their trajectory prediction errors for $t_{\text{pred}} = 1 \text{ s}$, $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$, followed by a qualitative contrast of their trajectories for selected road segments.

⁴Technically, a fully LSTM prediction for both speed and yaw rate produced the best results in [54] but [55] showed this model is not real-time capable at 20 Hz and therefore not included

5.1.6.1. Quantitative Comparison of Trajectory Prediction Errors

This sub-sub-section compares the trajectory prediction errors of various models for prediction horizons of $t_{\text{pred}} = 1 \text{ s}$, $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$ over the Malsheim Route 1 as depicted in Figure 5.2a. The best-performing models from previous works are shown in the top halves of the tables, while the bottom halves contain the novel models from this thesis.

Table 5.22 summarizes the parameters of the various models, which were selected either as determined in previous works or as discussed in the previous subsections.

Model	$n_{\text{pg}_{\text{long}}}$	$t_{\text{in}_{\text{long}}} [\text{s}]$	$n_{\text{pg}_{\text{lat}}}$	$t_{\text{in}_{\text{lat}}} [\text{s}]$	$\phi_{\min} [\text{deg}]$	$a_y [\frac{\text{m}}{\text{s}^2}]$	$t_{\text{const}} [\text{s}]$	$\Delta\psi_{\max} [\text{deg}]$
const_a_const_yawr (as in [17])	const.	N/A	const.	N/A	N/A	N/A	N/A	N/A
extrapol_speed_extrapol_yawr (as in [54])	1	$5 * t_{\text{pred}}$	1	$5 * t_{\text{pred}}$	N/A	N/A	N/A	N/A
const_a_LSTM_yawr (as in [55])	const.	N/A	1	$5 * t_{\text{pred}}$	N/A	N/A	N/A	N/A
LSTM_speed_const_yawr (as in [55])	5	$5 * t_{\text{pred}}$	const.	N/A	N/A	N/A	N/A	N/A
const_speed_const_heading	const.	N/A	const.	N/A	N/A	N/A	N/A	N/A
extrapol_speed_const_heading	1	0.75	const.	N/A	N/A	N/A	N/A	N/A
extrapol_speed_const_roll	1	0.75	const.	N/A	N/A	N/A	N/A	N/A
extrapol_speed_extrapol_rollangle	1	0.75	1.	0.75	N/A	N/A	N/A	N/A
extrapol_speed_trapezoid_roll	1	0.75	trap.	1	0.25	4.5	0.85	N/A

Table 5.22.: Selected Parameters of the Trajectory Prediction Models for Comparisons

The models are all evaluated over the Malsheim 1 rosbag, which was recorded with the new INS introduced in section 4.6 and same EKF settings. This way the relative comparisons of the trajectory predictions are neither affected by slightly different route choice or different state estimations (positioning and heading).

The only and thus best performing physics-based model from [17] is the `const_a_const_yawr` model. Because of its reliability, this model actually remained the number one choice in the trajectory prediction module of the collision warning system before starting this thesis.

The only and thus best performing physics-based model from [54] is the `extrapol_speed_extrapol_yawr` model. This comparison uses a ABS WSS rather than the originally used HMI WSS for a fair comparison with the other models. The true model performance thus would likely be worse than reported in Table 5.22 as examined in subsubsection 5.1.4.1. The extrapolation parameters $n_{\text{pg}_{\text{long}}}$, $t_{\text{in}_{\text{long}}}$, $n_{\text{pg}_{\text{lat}}}$ and $t_{\text{in}_{\text{lat}}}$ are set as described as described in his thesis, though.

The best performing ML-based model from [54] technically is a fully LSTM. However, it shares the same general architecture as in the respective LSTM components of `const_a_LSTM_yawr` and `LSTM\speed\const\yawr` models from [55] used in this comparison. Some slight differences include: First, the model from [54] uses LSTMs to predict both the longitudinal (speed) and lateral (yaw rate) components, whereas the models from [55] only use the LSTM to predict only one of the components while keeping the other one constant. Second, [54] uses a 5th order polynomial for the yaw rate prediction, while [55] uses a 1st order polynomial. Third, the architecture in [55] uses twice as many nodes in the LSTM and fully-connected

5. Evaluation

layers. Finally, the models were trained on different data bases where [54] seemed to have trained on more data. Since the fully LSTM model does not manage to perform inference at 20 Hz on the NUC as shown by [55], it is not used in this comparison. Since the fully LSTM model by [54] was evaluated on different metrics, the performance of this model can only be roughly estimated. It can be expected that fully LSTM model would be near the $\mu_{\Delta x}$, $\sigma_{\Delta x}$, $\mu_{\Delta l}$ and $\sigma_{\Delta l}$ metrics of LSTM_speed_const_yawr as they use the same architecture and polynomial order for the longitudinal component. However, one model is deeper and the other one trained on more data, which makes forecasting the expected performance gap hard. Similarly, the performance of the model is expected to be near to the $\mu_{\Delta y}$, $\sigma_{\Delta y}$, $\mu_{\Delta \sigma}$ and $\sigma_{\Delta \sigma}$ metrics of const_a_LSTM_yawr. However, the performance gaps between the lateral components are likely bigger as they additionally use different polynomial degrees.

Muresan [55] used the same physics-based model from [17] without any adaptions. The best-performing ML-based model from [55] is the LSTM_speed_const_yawr model. For comparison purposes, the const_a_LSTM_yawr model is also benchmarked. Even though these models were originally trained on the HMI WSS, inference is performed using the ABS WSS sensor. Table A.1 and Table A.2 in the appendix generally show worse errors if these models are inferred with the HMI WSS for $t_{\text{pred}} = 1$ s and $t_{\text{pred}} = 2$ s, respectively. This comparison directly uses the trained models from [55]. Since he did not train these models for $t_{\text{pred}} = 4$ s the corresponding cells in Table 5.25 are marked with *not applicable* (N/A).

Table 5.23, Table 5.24 and Table 5.25 compare the trajectory prediction errors for all of the aforementioned models averaged over the entire Malsheim 1 route for $t_{\text{pred}} = 1$ s, $t_{\text{pred}} = 2$ s and $t_{\text{pred}} = 4$ s, respectively.

Model	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
const_a_const_yawr (as from [17])	0.20	0.66	0.05	0.40	0.20	0.67	0.53	5.16
extrapol_speed_extrapol_yawr (as from [54])	0.22	0.28	0.05	0.64	0.21	0.26	0.44	9.35
const_a_LSTM_yawr (as in [55])	0.20	2.21	0.06	0.44	0.20	2.22	2.00	16.68
LSTM_speed_const_yawr (as from [55])	0.25	1.80	0.09	0.56	0.21	1.77	2.30	14.71
const_speed_const_heading	0.17	0.29	0.06	0.46	0.19	0.28	0.59	6.39
extrapol_speed_const_heading	0.27	0.28	0.06	0.43	0.19	0.24	0.60	6.75
extrapol_speed_const_roll	0.20	0.21	0.04	0.26	0.20	0.22	0.49	3.51
extrapol_speed_extrapol_rollangle	0.20	0.22	0.04	0.23	0.20	0.23	0.47	3.42
extrapol_speed_trapezoid_roll	0.20	0.22	0.04	0.25	0.20	0.22	0.49	4.28

Table 5.23.: Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 1$ s averaged over 1724 s Malsheim 1 Trip

5. Evaluation

Model	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
const_a_const_yawr (as from [17])	0.16	2.65	0.13	1.80	0.10	2.66	0.57	11.52
extrapol_speed_extrapol_yawr (as from [54])	0.21	1.08	0.12	2.53	0.14	0.87	0.62	17.42
const_a_LSTM_yawr (as in [55])	0.30	3.58	0.83	1.77	0.15	3.58	4.28	14.17
LSTM_speed_const_yawr (as from [55])	0.55	2.76	0.07	1.63	0.49	2.61	0.33	13.65
const_speed_const_heading	-0.10	1.05	0.14	1.69	0.05	0.97	0.66	12.29
extrapol_speed_const_heading	-0.13	0.99	0.14	1.7	0.05	0.87	0.64	12.48
extrapol_speed_const_roll	0.14	0.85	0.07	1.15	0.12	0.84	0.4	7.83
extrapol_speed_extrapol_rollangle	0.22	0.89	0.08	0.94	0.16	0.87	0.35	6.69
extrapol_speed_trapezoid_roll	0.14	0.92	0.08	1.04	0.12	0.89	0.45	7.48

Table 5.24.: Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 2$ s averaged over 1724 s Malmsheim 1 Trip

Model	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
const_a_const_yawr (as from [17])	0.23	10.60	0.43	7.11	-0.43	10.14	0.72	26.71
extrapol_speed_extrapol_yawr (as from [54])	0.14	4.40	0.15	8.49	-0.11	3.65	0.85	31.49
const_a_LSTM_yawr (as in [55])	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LSTM_speed_const_yawr (as from [55])	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
const_speed_const_heading	-1.40	4.03	0.30	5.34	-0.44	3.49	0.80	21.34
extrapol_speed_const_heading	-1.55	3.84	0.31	5.43	-0.56	3.67	0.81	21.61
extrapol_speed_const_roll	0.32	4.91	0.09	4.82	0.04	3.83	0.11	21.00
extrapol_speed_extrapol_rollangle	1.15	5.12	0.24	5.55	0.77	4.59	0.46	22.95
extrapol_speed_trapezoid_roll	-0.08	4.54	0.15	4.73	-0.19	3.84	0.74	19.87

Table 5.25.: Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 4$ s averaged over 1724 s Malmsheim 1 Trip

The magnitudes of the errors for the base `const_a_const_yawr` model are generally smaller than in the comparable *Renningen-Malmsheim Testroute* trip from [55]. For example, the positional precisions $\sigma_{\Delta x}$ and $\sigma_{\Delta y}$ in this comparison are lower by roughly 0.50 m for $t_{\text{pred}} = 1$ s and $t_{\text{pred}} = 2$ s. However, since the routes, rider and reference sensors differ the deviations seems reasonable.

As suspected in subsubsection 4.1.6.2, the very long input duration of $5 * t_{\text{pred}}$ chosen in [54] for the linear extrapolation of the lateral component from `extrapol_speed_extrapol_yawr` model actually worsens the lateral predictions compared to the constant yaw rate baseline. In fact, this parameterization leads to the worst $\sigma_{\Delta y}$ out of all of the compared models over all prediction horizons. Such long input durations for the speed extrapolation seem to have a surprisingly positive effect on the longitudinal metrics, though.

The ML-based models generally show very similar magnitudes for these error statistics as in the comparable *Renningen-Malmsheim Testroute* trip from [55]. The models perform worse throughout all metrics compared to the `const_a_const_yawr` baseline for $t_{\text{pred}} = 1$ s. A similar trend was also shown in [55]. However, in this comparison, both ML-models perform slightly worse in the longitudinal component and slightly better in the lateral component than the `const_a_const_yawr` baseline even for $t_{\text{pred}} = 2$ s. This does not match the observations in [55], where the the `LSTM_speed_const_yawr` model barely outperformed the baseline in nearly all metrics. This may be explained by the fact, that the baseline achieves significantly better results in this comparison compared to [55], while the performance of the ML models

5. Evaluation

shows similar magnitudes.

The `const_speed_const_heading` model, chosen mostly because of its expected linear behavior in the longitudinal and lateral components with respect to time, does not perform that well on long prediction horizons as expected. Even for $t_{\text{pred}} = 4\text{ s}$, this models generally falls behind or is roughly similar in performance compared to all other newly models developed in this thesis. Problematic for this prediction horizon may the fact that predicted trajectories are generally too short as indicated by the negative mean values for $\mu_{\Delta l}$ and especially $\mu_{\Delta x}$. However, it is believed that this mostly occurs because of the terrible performance in curves. Nevertheless, this model generally still outperforms the `const_a_const_yawr` baseline rather significantly for all prediction horizons.

The `extrapol_speed_const_heading` model slightly outperforms the `const_speed_const_heading` model in terms of longitudinal precision $\sigma_{\Delta x}$ as expected. However, it generally performs slightly worse for all other metrics in this comparison. Moreover, it faces the same issues regarding a strong offset in the longitudinal predictions for $t_{\text{pred}} = 4\text{ s}$.

The `extrapol_speed_const_roll` model indicates promising results for the accuracies and precision for both the longitudinal and lateral components over all prediction horizons. In fact, only $\mu_{\Delta l}$ for $t_{\text{pred}} = 2\text{ s}$ is worse by 0.02 m and $\mu_{\Delta x}$ for $t_{\text{pred}} = 4\text{ s}$ is worse by 0.10 m compared to the `const_a_const_yawr` baseline. Besides these two minor setbacks, all metrics for all prediction horizons are at least retained and often significantly outperformed. For example, the precision of the longitudinal and lateral positions are improved by 1.80 m and 0.65 m, respectively, for $t_{\text{pred}} = 2\text{ s}$. The relative improvement compared to the baseline thus amounts to around 68% and 36% in these metrics. Moreover, the mean values are near zero for all metrics over all prediction horizons, so only few systematic errors are made and the predictions can generally be trusted.

The `extrapol_speed_extrapol_roll` model surprisingly shows the best overall performance for $t_{\text{pred}} = 1\text{ s}$ and $t_{\text{pred}} = 2\text{ s}$. However, its performance degrades a lot for $t_{\text{pred}} = 4\text{ s}$, where especially $\mu_{\Delta x}$ shows a big offset.

The `extrapol_speed_trapezoid_roll` model shows the best overall performance for $t_{\text{pred}} = 4\text{ s}$ and falls only slightly behind the `extrapol_speed_extrapol_roll` model for the other prediction horizons. This indicates that the parameters set through the statistics did not achieve an optimal result, but the model still performs well.

The following paragraph visualizes all predictions over the Malmsheim 1 and compares the baseline against two of the new models. The scatter plots in Figure 5.11, Figure 5.12 and Figure 5.13 compare the positional errors of the three models `const_a_const_yawr`, `extrapol_speed_const_roll` and `extrapol_speed_trapezoid_roll` in that order for $t_{\text{pred}} = 1\text{ s}$, $t_{\text{pred}} = 2\text{ s}$ and $t_{\text{pred}} = 4\text{ s}$, respectively. For better comparison over the prediction horizon, the plots are all fixed from -30 m to + 30 m for the errors in both x- and y-direction. The scatter plots Figure 5.11, Figure 5.12 and Figure 5.13 compare the length and angle difference of these models for $t_{\text{pred}} = 1\text{ s}$, $t_{\text{pred}} = 2\text{ s}$ and $t_{\text{pred}} = 4\text{ s}$, respectively. The length difference (x-axis) ranges from -30 m to +30 m, and the angle difference (y-axis) ranges from -120 degrees

5. Evaluation

to +120 degrees in all of the plots.

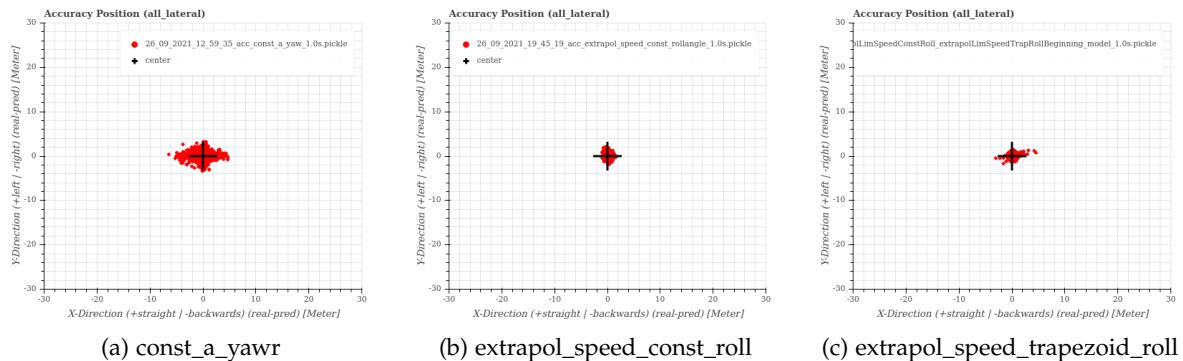


Figure 5.8.: Comparison of Positional Errors for $t_{\text{pred}} = 1 \text{ s}$

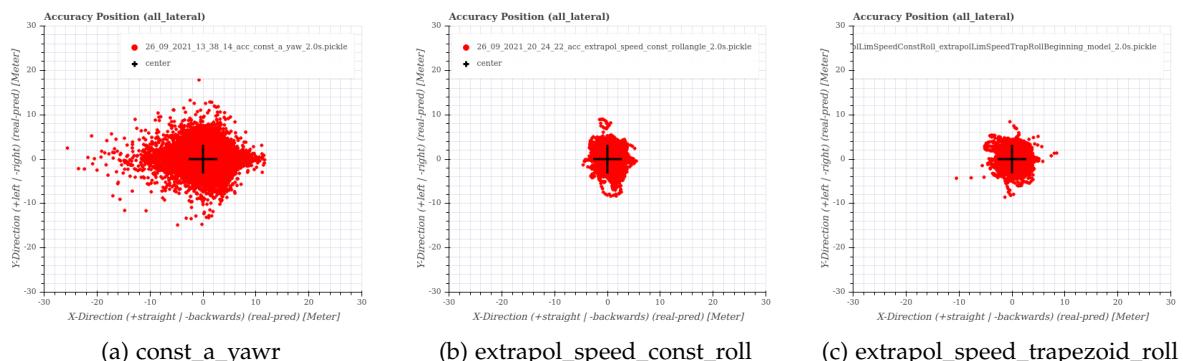


Figure 5.9.: Comparison of Positional Errors for $t_{\text{pred}} = 2 \text{ s}$

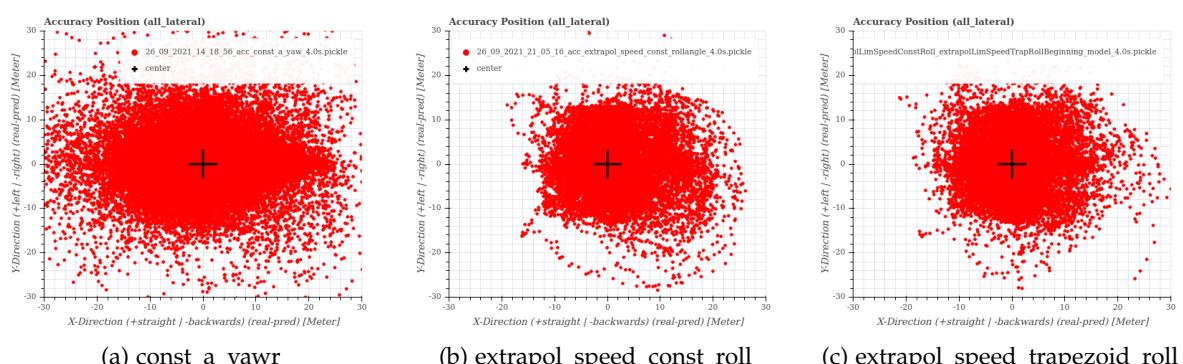


Figure 5.10.: Comparison of Positional Errors for $t_{\text{pred}} = 4 \text{ s}$

5. Evaluation

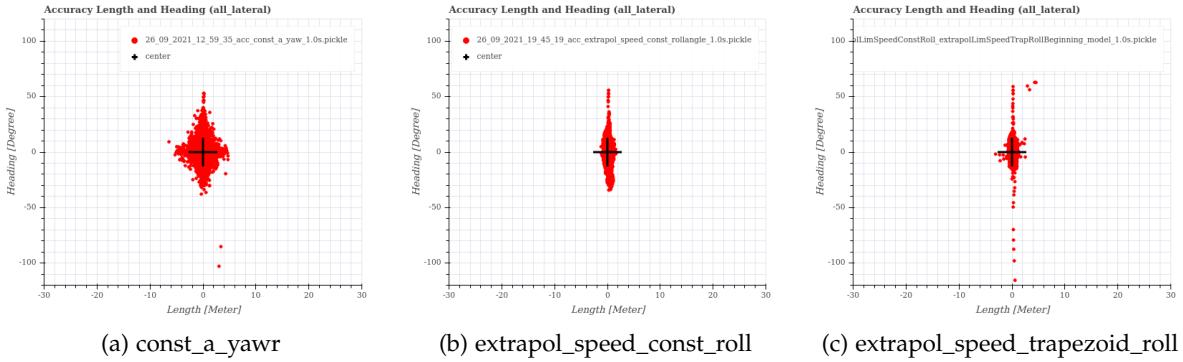


Figure 5.11.: Comparison of Length and Angle Errors for $t_{\text{pred}} = 1 \text{ s}$

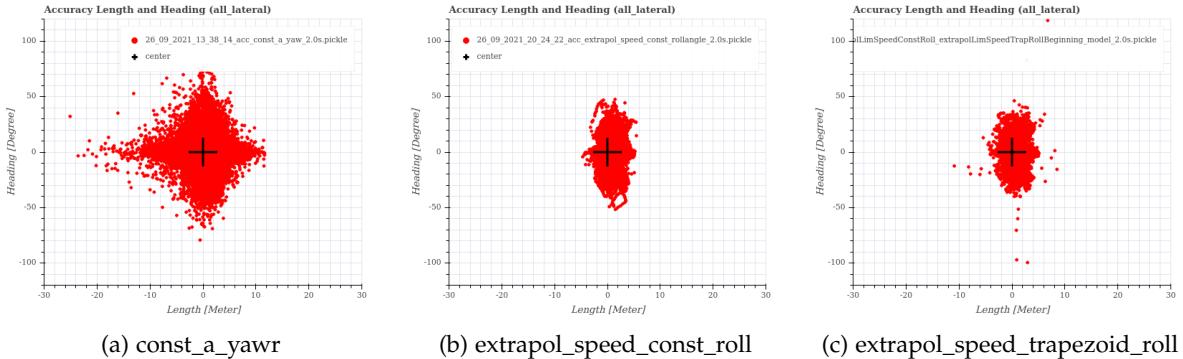


Figure 5.12.: Comparison of Length and Angle Errors for $t_{\text{pred}} = 2 \text{ s}$

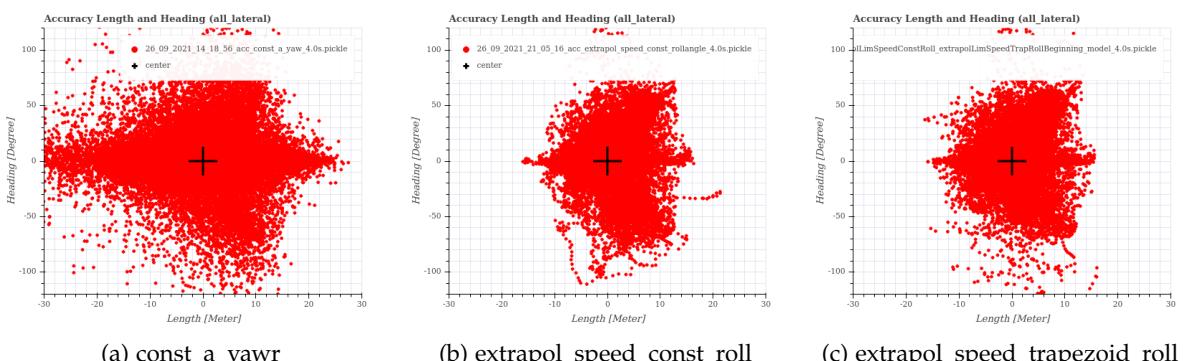


Figure 5.13.: Comparison of Length and Angle Errors for $t_{\text{pred}} = 4 \text{ s}$

The plots clearly visualize how the errors increase exponentially with respect to the prediction horizon (go vertically through the plots). They also clearly show the improvements of

5. Evaluation

both new models compared to the baseline (go horizontally though the plots). Furthermore, the scatter plots let us further compare the maximum magnitudes of the prediction errors, which were not captured in the statistics above.

For example, the `extrapol_speed_const_roll` shows the smallest maximum deviation in x-position by far, whereas `extrapol_speed_trapezoid_roll` has the smallest maximum deviation in y-position by a small margin over `extrapol_speed_const_roll` for $t_{\text{pred}} = 1 \text{ s}$. For $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$ similar trends show.

The `extrapol_speed_const_roll` also shows less outliers for the length and angle difference than `extrapol_speed_trapezoid_roll` over all prediction horizons.

From a safety perspective, the `extrapol_speed_const_roll` may be more desired as it "guarantees" that not a single trajectory endpoint is off by more than 3 m in the x- and y-direction for $t_{\text{pred}} = 1 \text{ s}$ and no more than 10 m for $t_{\text{pred}} = 2 \text{ s}$ for the Malsheim 1 route.

Table 5.26, Table 5.27 and Table 5.28 discuss the relative improvements on the combined positional accuracy and precision, as introduced in Equation 4.3 and Equation 4.4, of the `extrapol_speed_extrapol_roll` model compared to the `const_a_const_yawr` baseline for $t_{\text{pred}} = 1 \text{ s}$, $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$, respectively.

Model	ACC_{pos} [m]	PREC_{pos} [m]
<code>const_a_const_yawr</code> (as from [17])	0.21	0.77
<code>extrapol_speed_trapezoid_roll</code>	0.20	0.33
relative improvement	5%	57%

Table 5.26.: Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 1 \text{ s}$

Model	ACC_{pos} [m]	PREC_{pos} [m]
<code>const_a_const_yawr</code> (as from [17])	0.37	3.20
<code>extrapol_speed_trapezoid_roll</code>	0.23	1.30
relative improvement	38%	59%

Table 5.27.: Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 2 \text{ s}$

Model	ACC_{pos} [m]	PREC_{pos} [m]
<code>const_a_const_yawr</code> (as from [17])	0.49	12.76
<code>extrapol_speed_trapezoid_roll</code>	0.17	6.56
relative improvement	65%	49%

Table 5.28.: Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 4 \text{ s}$

5. Evaluation

These tables show that the novel `extrapol_speed_trapezoid_roll` model developed in this thesis significantly outperforms the baseline that it started with when evaluated over an entire trip. The relative improvement on the combined positional accuracy increases with longer prediction times, and the improvements in the positional precision stay consistently high between 49 - 59% for all prediction horizons.

To summarize, all of novel trajectory prediction models developed in this thesis outperform the `const_a_const_yawr` baseline model from [17]. In fact, the alternative physics-based model from [54] is also surpassed by most models. More so, all new models even outperform the ML-based approaches from previous works. Despite being very simple, the performance of the `extrapol_speed_const_roll` model comes close to the `extrapol_speed_extrapol_roll` and `extrapol_speed_trapezoid_roll` models. In comparison to the baseline, the `extrapol_speed_trapezoid_roll` model cuts down the combined positional accuracy and precision roughly in half for the targeted prediction horizon of 2 - 4 s.

5.1.6.2. Qualitative Comparison on Specific Segments

This subsection rates the performance of various models on specific segments in an effort to pinpoint their strengths and weaknesses.

Figure 5.14, Figure 5.15, Figure 5.16 and Figure 5.17 show some trajectories of the `const_a_const_yaw`, `extrapol_speed_const_roll`, `extrapol_speed_extrapol_roll` and `extrapol_speed_trapezoid_roll` models for $t_{\text{pred}} = 2 \text{ s}$, respectively. The riding direction is from West to East.

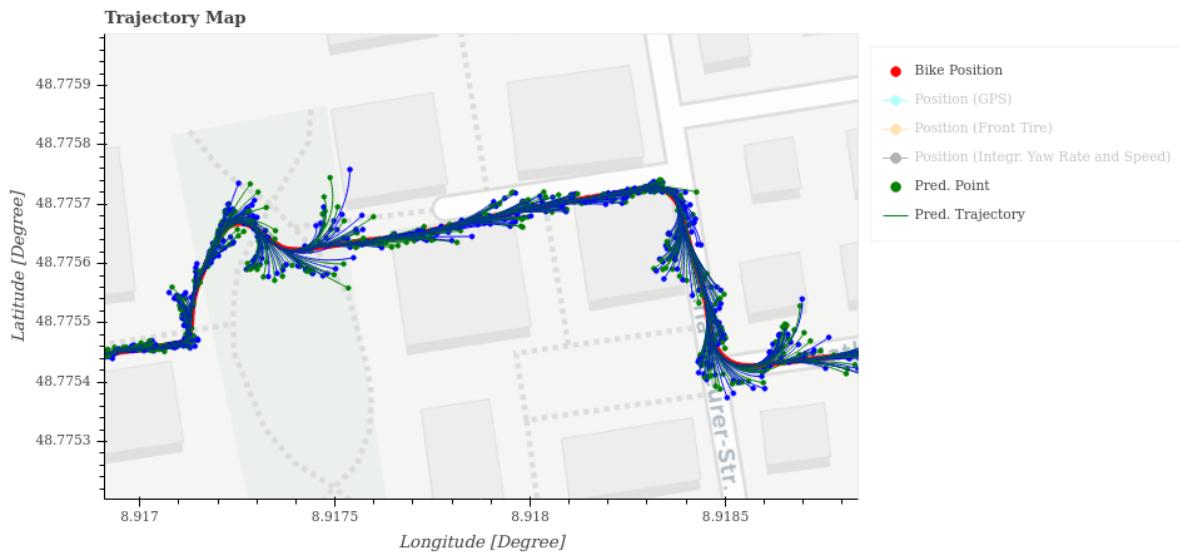


Figure 5.14.: Trajectories of `const_a_const_yaw` for $t_{\text{pred}} = 2 \text{ s}$

5. Evaluation

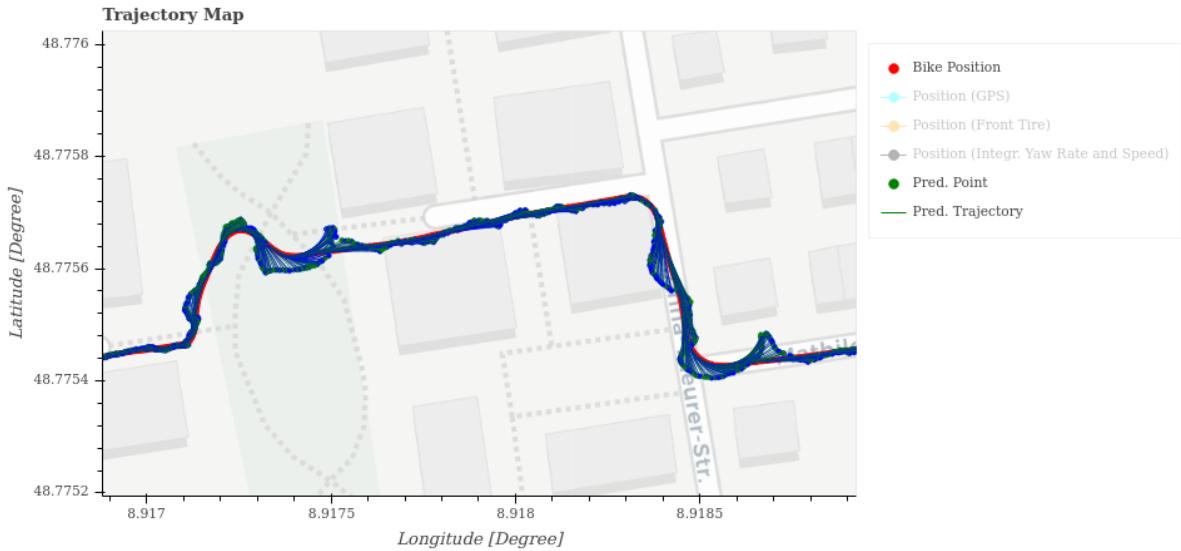


Figure 5.15.: Trajectories of `extrapol_speed_const_roll` for $t_{\text{pred}} = 2 \text{ s}$

All of the new novel using the roll angle significantly improve the trajectories compared to the `const_a_const_yaw` baseline in both curves and straight sections. Obviously, the `extrapol_speed_extrapol_roll` model shares the same issues regarding undershooting trajectories at the curve entry and overshooting trajectories at curve exits as the `const_a_const_yaw` like analyzed in subsubsection 4.1.6.1. In alignment with the results from Table 5.24, the `extrapol_speed_extrapol_roll` shows trajectories matching most closely with the actually ridden path in red. Note that the `extrapol_speed_trapezoid_roll` in Figure 5.17 only triggers the high-dynamic trapezoidal extrapolation model in the last curve (compare to last curve in Figure 5.15). This is because the minimum roll angle threshold is set really high to $|\phi_{\min}| > 16.00 \text{ deg}$ as discussed in subsubsection 5.1.5.2. The bundled trajectories in the middle of this curve show the trapezoidal extrapolation at work. However, as these trajectories point too far out of the curve, the parameters of this model seems to underestimate the this curve.

All in all, this shows that setting the parameters of the `extrapol_speed_trapezoid_roll` as the mean values of statistical evaluation over all curves does not work well in practice.

Moreover, the online adaptions of the roll angle profile parameters explained at the end of subsubsection 4.8.4.5 in some cases may lead to worse roll angle profiles. For example, Figure 5.18 shows the roll rate and roll angle signals for a very long 180 degree curve ridden at high speeds. This was recorded separately from the Malmsheim evaluation and $|\phi_{\min}| > 7.50 \text{ deg}$ in this case.

5. Evaluation

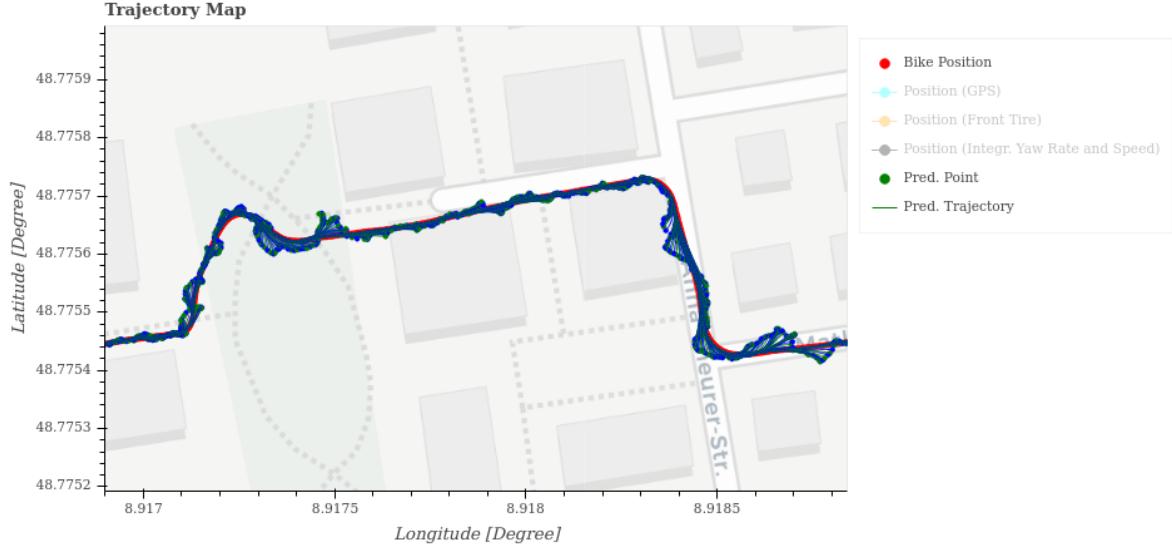


Figure 5.16.: Trajectories of extrapol_speed_extrapol_roll for $t_{\text{pred}} = 2 \text{ s}$

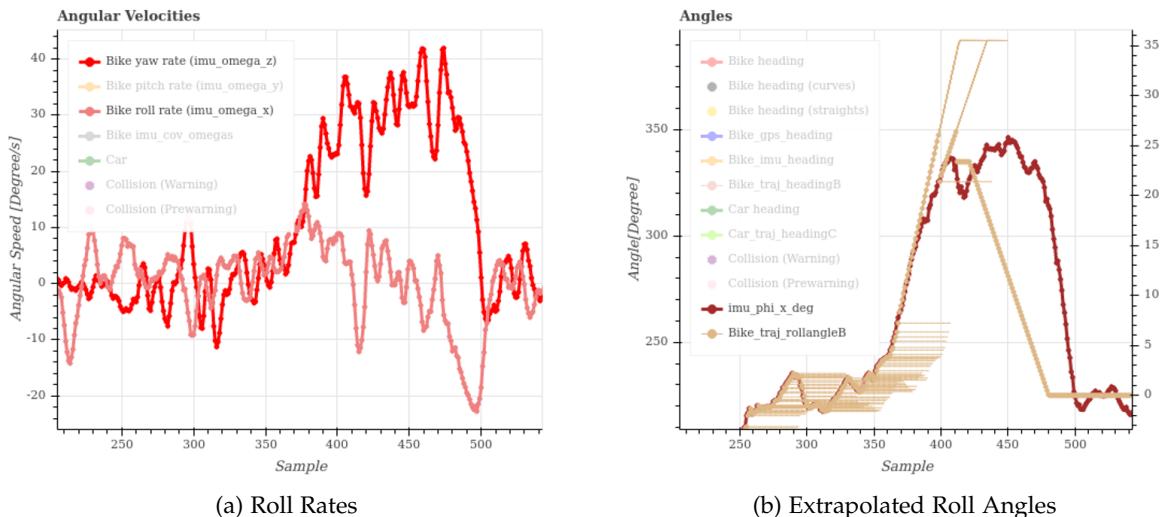


Figure 5.18.: Signals for 180 Degree Curve of Large Radius at High Speed

In this case, the rider truly balances around the roll angle equilibrium for an extended period of time opposed to just hitting it once. Figure 5.18a show the roll rates crossing zero multiple times and falsely signalizing the adaption that the curve is soon over. Figure 5.18b show how the constant duration parameter t_{const} is adapted to a value that is way too small. As discussed in chapter 6, the parameters of the model may in future be predicted online via ML anyway, thus eliminating the need for this roll rate based adaption.

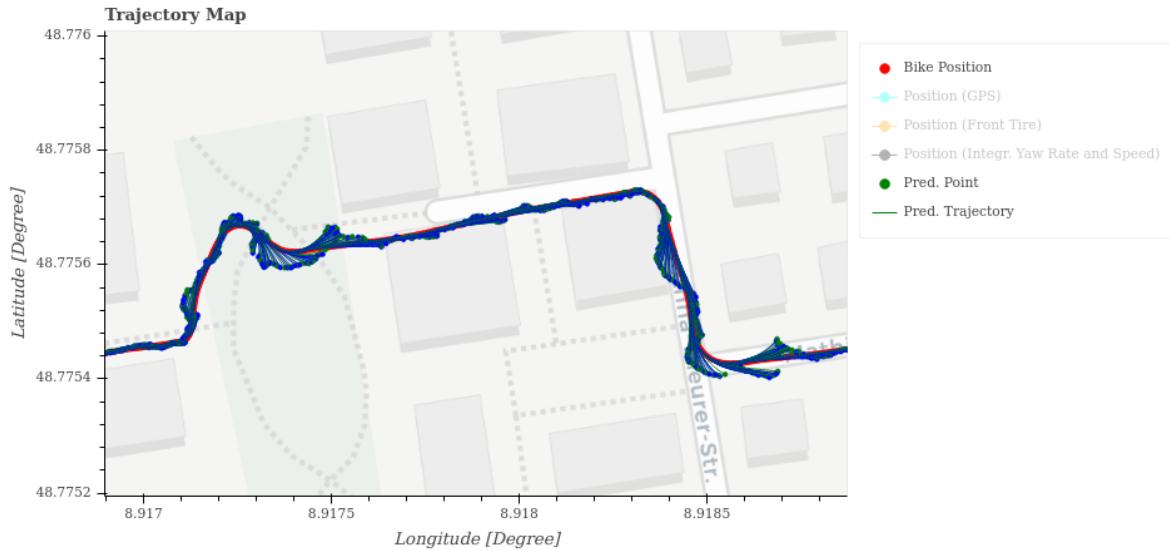


Figure 5.17.: Trajectories of `extrapol_speed_trapezoid_roll` for $t_{\text{pred}} = 2 \text{ s}$

5.2. Extrinsic Evaluation via Collision Warning Function

As mentioned already, improving the intrinsic metrics of the trajectory prediction component may not necessarily guarantee a noticeable improvement of the overall collision warning system. Moreover, this thesis enhanced other components that might be crucial to the collision warning system as well. Consequently, this section evaluates the impacts of the improved components extrinsically on the collision warning system as a whole. It examines how the new reference INS, trajectory prediction models, CD and warning strategy improve the overall collision warning system. It finishes off by comparing the new collision warning system with the previous one on selected scenarios.

5.2.1. Influence of State Estimation and Inertial Measurements

Section 5.1.2 already qualitatively compared the influence of the new sensor setup on the trajectory prediction errors. This subsection evaluates the impact of the enhanced state estimation (position and heading) as well as inertial motion data as a result of integrating a high-precision INS in section 4.6 on the collision warning system as a whole. Since this thesis does not focus about evaluating the positioning or inertial data against ground truth data, this comparison is of qualitative nature. A follow-up thesis will evaluate the positioning (and maybe attitude estimation) quantitatively.

As explained in subsection 5.1.2, if the state estimation and internal measurements provide e.g. an offset, this error will carry over directly to the trajectory predictions. This is because the trajectory prediction does not originate from the actual ground truth data and

rather has to rely on the measured data.

Consider the simplest scenario where a cyclist rides on a sidewalk on a straight road segment and assume the system had a perfect trajectory prediction that would accurately predict perfectly straight trajectories. Other road users are assumed to have perfect localization. If the measured initial positions of the pedelec is offset by e.g 3 m, all of the predicted positions would also be offset by 3 m as well. After all, the trajectory prediction cannot magically correct this by guessing what the actual initial position should have been. In this case, the collision warning system might incorrectly think that the cyclist is riding near the median strip of the road (opposed to the side walk). Consequently, the system may mistakenly signalize a collision with oncoming traffic from the other lane because the two trajectories intersect or come too close to each other. If the 3 m offset in pedelec position was measured to the opposite side of the side walk, the system "sees" the cyclist riding next to the road. In such cases, the system may not trigger any collision warnings at all because other road users cannot physically drive where the system thinks the pedelec is located, e.g. through houses. There exist many of such cases that could negatively affect the FP and FN rate of the system as a result of bad positioning.

Similarly, a wrong state estimation e.g. an offset in the roll angle while riding straight would incorrectly tell the system that the cyclist is riding a curve and thus predict curve trajectories. Again, this could either mean ghost-collisions are detected with oncoming traffic from the other lane or no collisions are detected at all e.g. turning through houses where other road users will never be located.

As covered in section 4.6 and subsection 5.1.2, the newly integrated INS greatly improves the roll angle estimation and positioning. A correct initial state allows the trajectory prediction models to forecast the actual positions in the real-world, opposed to pseudo-positions e.g. through houses. Accurately predicted positions allow the CD to correctly identify oncoming crashes with other road users. Thus, the new INS greatly improves the overall collision warning system.

5.2.2. Influence of New Trajectory Prediction Models

Section 4.6 already quantitatively investigated the influence of various factors on the trajectory prediction error metrics. This subsection discusses the impact of the novel physics-based trajectory prediction models on the collision warning system as a whole.

Since the trajectory prediction errors are averaged over an entire trip or over specific segments as described in subsection 4.8.1, there might always exist some other segments where a specific model performs poorly. If those segments are underrepresented, then the model statistically still performs well on the metrics. Obviously, inappropriately predicted pedelec trajectories may then come unintentionally come close or divert from those of other road users if in their vicinity. This may cause unintentional intersections of both trajectories or cause them to not intersect even though they should. In the end, this may then lead to FP or FN collision warnings by the system.

5.2.3. Influence of New Collision Detection and Warning Strategy

This subsection evaluates the impact of the new CD and warning strategy on the overall collision warning system on some simple but very common real-world scenarios. The new CD is compared against the previous one in a separate simulation environment with ideal positioning and trajectory predictions. This way, the impact of the pure CD and warning strategy can be evaluated without errors or inconsistencies arising from other factors such as an offset positioning or diverging trajectories. However, bounding rectangles approximate the shape of the road users instead of their true silhouettes.

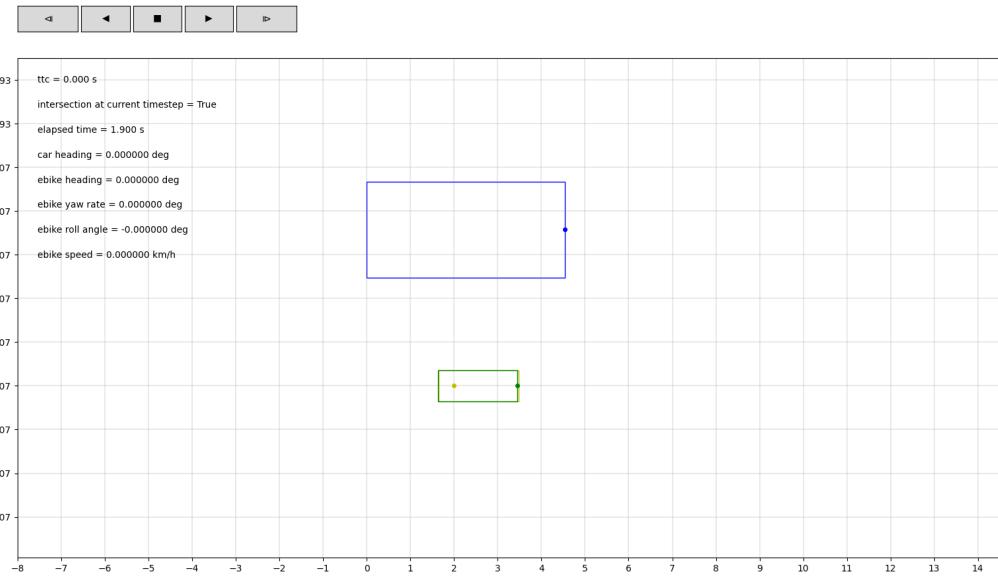
The simulations only model the simplest interactions along a straight road segment with a side walk. It uses measurements defined by the European Commission on the minimum widths of side walks and roads. The European Concept for Accessibility norms the width of the pure walking surfaces as follows: A single direction side walk needs to be at least 0.90 m wide. Sidewalks with frequent bidirectional usage need to be at least 1.80 m wide [98]. Note that there additionally needs to be a safety buffer to houses and heavy goods traffic. The German government recommends the total width of 2.50 m for sidewalk, measured from the walls of houses nearby to the actual road. However, note that many older towns may still have sidewalks adhering to the outdated regulation of 1.50 m [99]. According to several court decisions, the width of a road has to be at least 3.00 m (composed of 2.50 m width of car and 0.50 m safety buffer) [100] [101]. The car's rectangle is 2.20 m wide and 4.54 m long. The pedelec's rectangle is 0.77 m wide and 1.81 m long.

The simulation assumes a 2.50 m wide sidewalk with the bicycle riding near the house edge and the car in the middle of its 3.00 m wide lane. As a result, a driving car is therefore separated by a margin of 2.09 m from the pedelec riding in the same direction on the sidewalk. A car driving in the opposite lane is separated by a margin of 5.09 m from the pedelec.

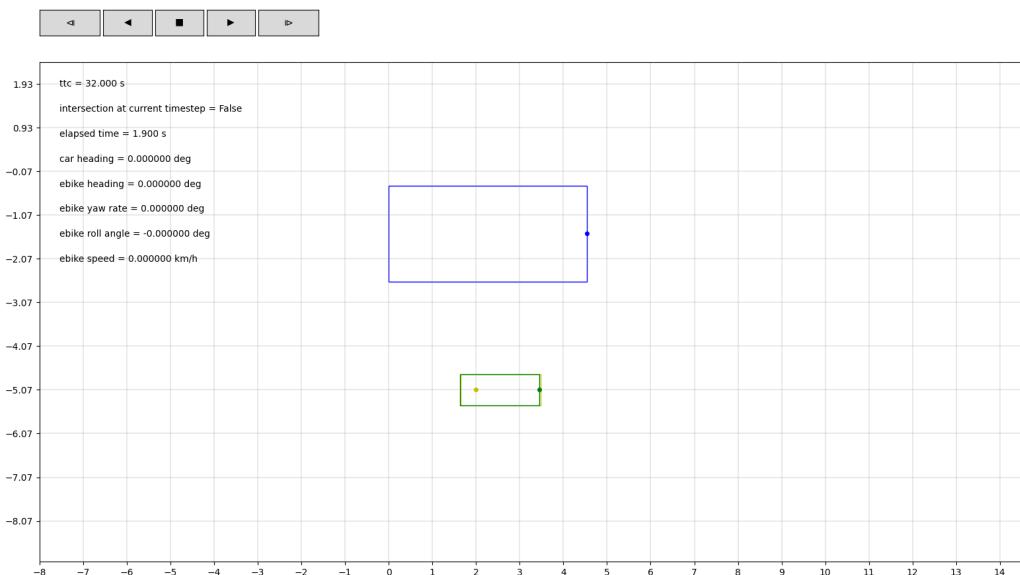
The dynamic range-based CD, which serves as a baseline, is parameterized with a base warning range of `warn_range= 4.0m` as shown in Figure 4.60 of [55]. The safety buffer is parameterized to `warn_range_buffer= 0.25m` for the old and new CD to ensure a fair comparison.

The simplest scenario involves both road users to be stationary and next to each other. The car is on the road and the bicycle on the sidewalk adjacent to the road in the same direction as shown in Figure 5.19. For example, the car and pedelec together wait at a red light. Despite both road users being stationary and absolutely no risk of a collision, the dynamic range-based CD triggers a warning as seen in the text field `ttc = 0.000 s` as well as `intersection at current timestep = True` in Figure 5.19a. In contrast, the new SAT-based CD shows a TTC of 32 seconds (the highest default value) and the intersection flag is False as depicted in Figure 5.19b. The outcome is the same with the old sidewalk width of 1.50 m. Figure A.5 in the appendix shows a similar scenario with the same setup as in Figure 5.19, expect the pedelec passes the stationary car from West to East. Once again, the old CD triggers a warning, whereas the new one rightfully does not.

5. Evaluation



(a) Dynamic Range-Based CD



(b) SAT-Based CD

Figure 5.19.: Collision Detection Comparison for Car and Pedelec Waiting on Red Light Next to Each Other

Figure 5.20 depicts another common scenario, where the pedelec rides on the sidewalk and car on the opposing lane approaches. Of course, the massive gap of 5.09 m ensures collision will not happen. The pedelec rides from West to East, and the car drives from East to West. Even in this case, the dynamic-range based CD triggers a collision warning with the oncoming car at a TTC of 0.50 s as seen in Figure 5.20a. The SAT-based CD does not trigger a warning as expected.

Finally, there exist cases, where the old CD detects incorrect stopping locations and warns too late. For example, consider a 90 degree junction without a line of sight between the road users. The pedelec enter the junction first from West to East, quickly followed by the car from North to South. However, the car does not miss the pedelec but clips its rear wheel instead 1.50 s into the future as depicted in Figure 5.21. In this scenario, the pedelec should come to a stop before the junction without intersecting with the car's predicted driving tube. Figure 5.22 compares the CD for the aforementioned scenario. The old CD predicts a collision at a TTC of 1.100 s. However, as depicted in Figure 5.22a, if the pedelec comes to a stop at this point in time, the car would hit the body and head of the cyclist instead of just the rear-wheel. In contrast, the new CD predicts a TTS of 0.950 s as shown in Figure 5.22b. The new system thus actually prevents the crash by ensuring the pedelec rider comes to a stop before the junction and does not end up in the driving tube of the car.

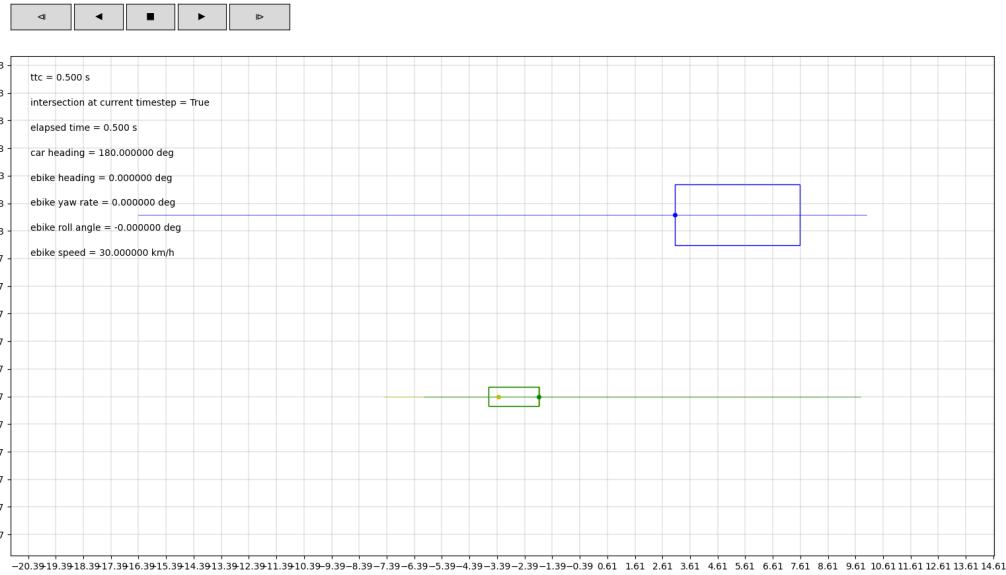
This subsection demonstrates the huge positive impact of the new SAT-based CD on the overall collision warning system. In simulations, it successfully solved a lot of common real-life scenarios that used to trigger FP alarms.

5.2.4. Qualitative Comparison

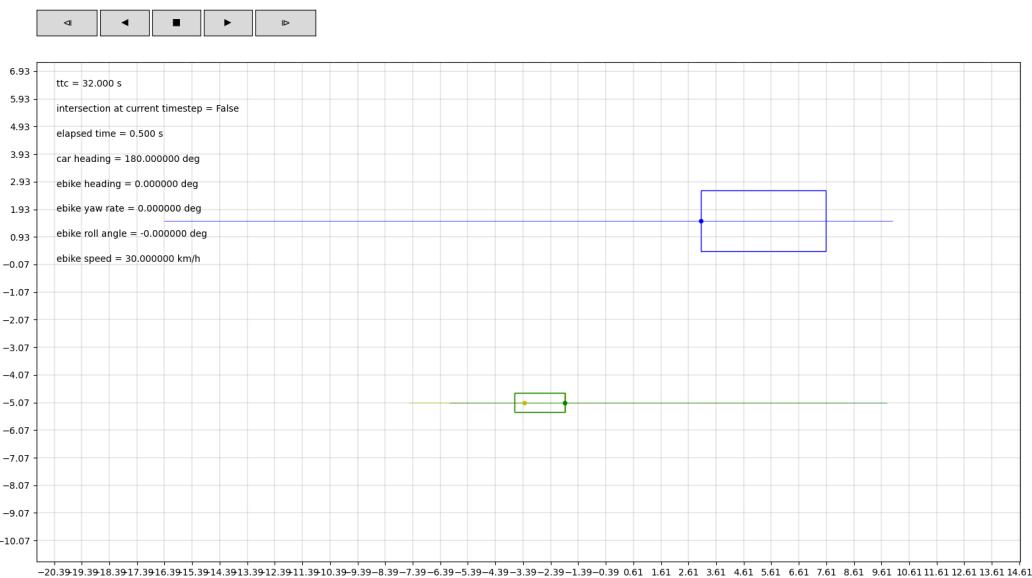
This subsection compares the new trajectory prediction, CD and warning strategy with the previous approach on a few common scenarios. Note that these comparisons use the same newly integrated INS to collect a high accuracy positioning and attitude estimation. This eliminates the possibly huge impact of e.g. an offset in the positioning as evaluated in subsection 5.2.1. Technically, in order to compare the collision warning functions as a whole, the exact same scenarios should have been repeated with the old sensor setup.

Revisiting the scenario from Figure 1.2 and comparing it to the new `extrapol_speed_const_roll` trajectory prediction model and the new SAT-based CD in Figure A.6 clearly shows the impact of the improvements on the target collision warning system. A similar improvement can be noted by using the `extrapol_speed_trapezoid_roll` model as shown in Figure A.6 of the appendix. The previous system uses the `const_a_yawr` model and dynamic-range CD. Figure A.6 compares the collision warning systems for the common scenario where the pedelec rides on the side walk or in a bike lane on a straight road segment next to a car that is driving on the road. When making a right turn, the driver of the car oversees the cyclist e.g. because of the blind spot and nearly crashes into the cyclist who

5. Evaluation



(a) Dynamic Range-Based CD



(b) SAT-Based CD

Figure 5.20.: Collision Detection Comparison for Car Driving in the Opposite Lane and Pedelec Riding on Sidewalk

5. Evaluation

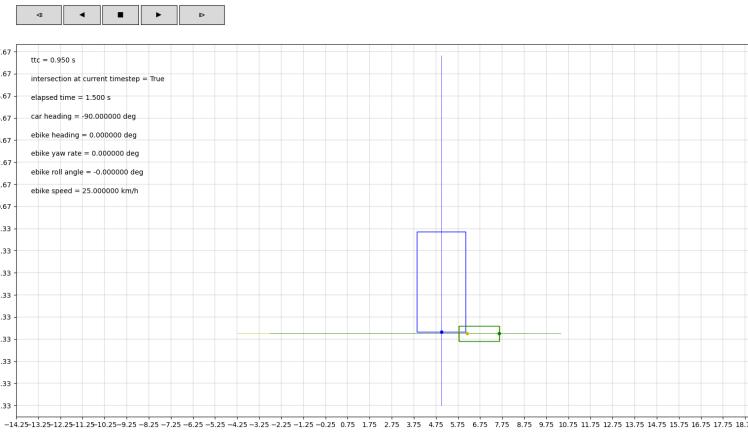


Figure 5.21.: Car Clipping the Rear Wheel of a Pedelec in a 90 Degree Junction

intends to go straight. In this case, the collision warning system needs to alert the cyclist only once, shortly before the car turns right. The trajectories in red represent the pedelec's predicted movement two seconds into the future at every computation step. Similarly, the green trajectories represent the car's predicted movements. A collision warning triggered by the system gets marked with a purple cross. Note how all of the FP collision alarms when riding parallel to the car disappear in A.6b with the new collision warning system. Also, the last FP in A.6a, where the pedelec is already at rest, disappeared as well in A.6b⁵.

Figure 5.24 further compares the timings of the collision alarms from both methods for the same scenario. The plots only show all of the alarm requests which do not necessarily trigger a new warning sound every time. One reason is that the sound file of the siren needs to finish playing first before getting triggered again. Thus $t_{\text{sound}} = 2 \text{ s}$ equate to a duration of 40 samples at $f_{\text{node}} = 20 \text{ Hz}$. In addition, the previous approach uses a hard-coded duration $t_{\text{sleep}} = 3 \text{ s}$ for suppressing warnings that follow each other closely. The yellow dots in 5.24a mark the start of an auditory signal to the rider, and the gray boxes symbolize t_{sleep} . The time slot for the actual collision alarm is marked with a green box in each of the subfigures of Figure 5.24. For this specific scenario, all of the four warnings from the previous approach result in FP while the appropriate time window for the actual collision warning is falsely suppressed by t_{sleep} and thus missed. In contrast, the new approach only warns the rider once in the appropriate time window. To be fair, this single warning should ideally arrive a bit sooner, near the beginning of the green box, to really ensure that the cyclist can come to a complete stop. However, the slightly late warning is a result of the car's trajectories still pointing outwards of the curve at this point in time, such that they do not yet cross or come close enough to the predicted trajectories of the pedelec.

⁵Refer to 5.24a where the speed is zero for the last collision alarm.

5. Evaluation

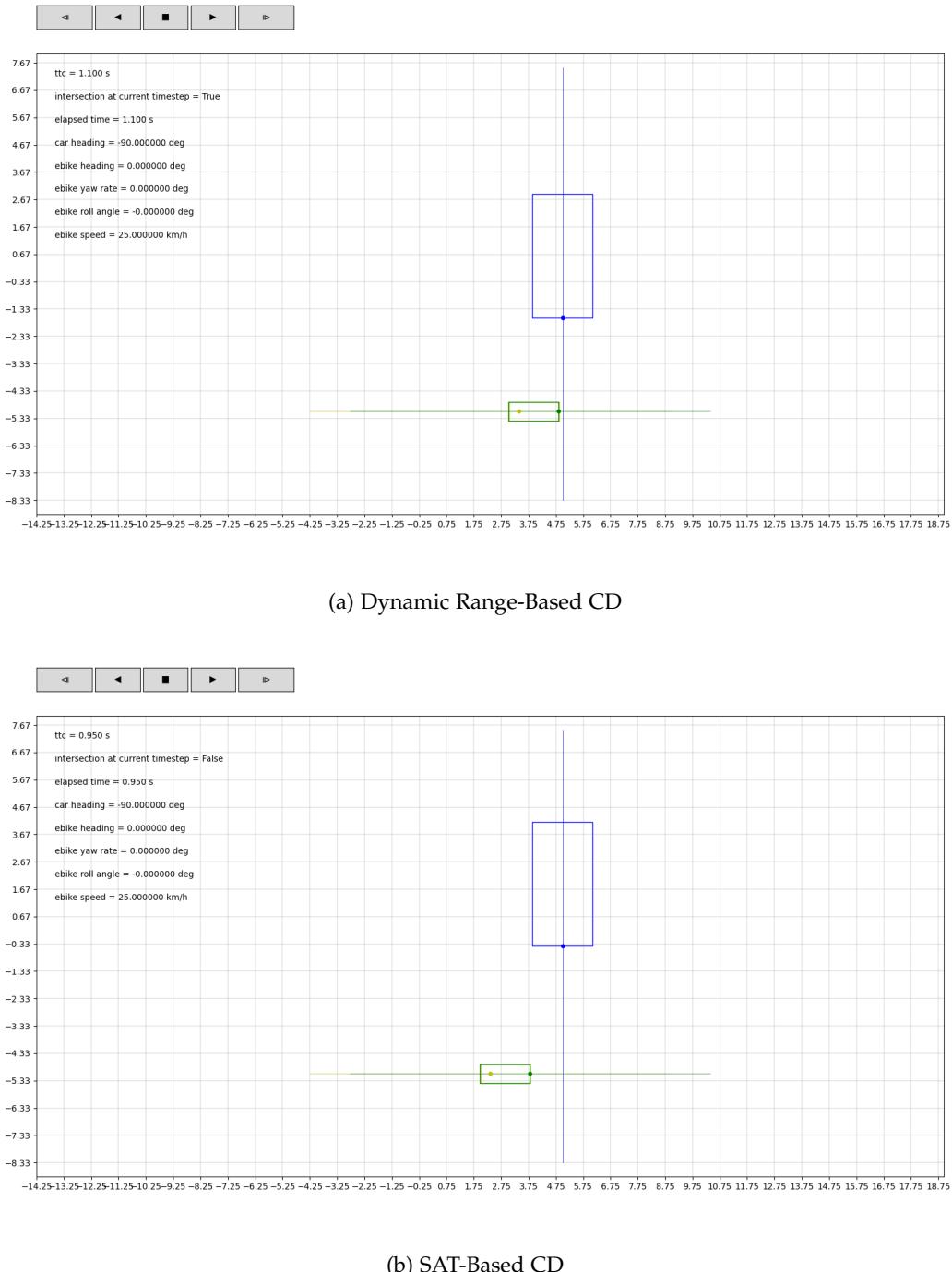


Figure 5.22.: Collision Detection Comparison for Car Clipping the Rear Wheel of a Pedelec in a 90 degree Junction

5. Evaluation

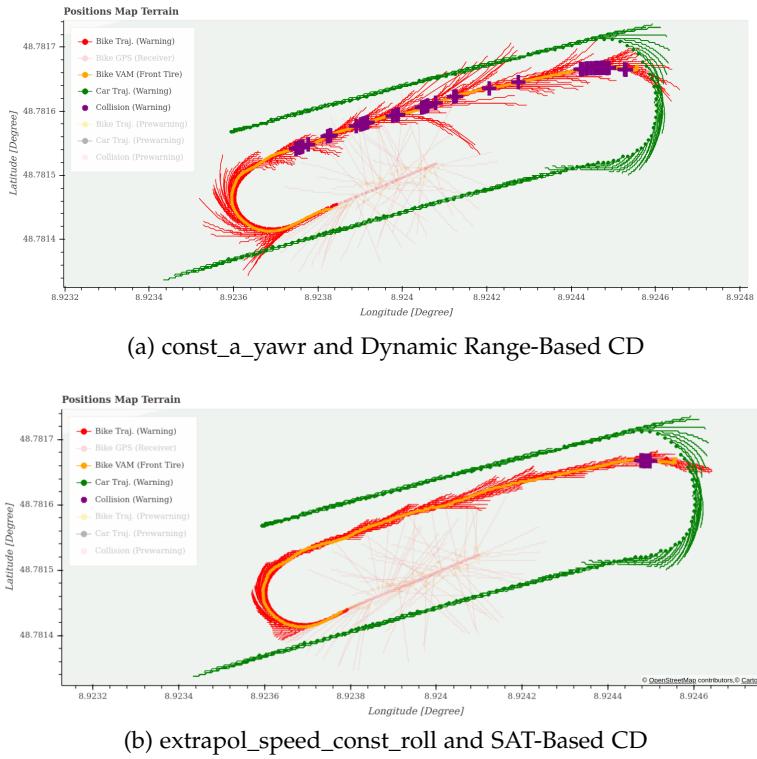


Figure 5.23.: Trajectory Comparison for Parallel Driving Scenario with Car Turning Into Pedelec

In the following, the pedelec and car partake in another common scenario. The pedelec is riding on a priority road, and the car in the intersection yields to the right-of-way of the pedelec until it passes. Figure 5.25 compares the collision warning system for the aforementioned scenario. As shown in the figures, the car is at a complete stop and does not move (see green dot). The pedelec rides along a straight road segment past the intersection where the car is waiting. Obviously, this scenario does not pose a threat to the pedelec as the car sees the oncoming pedelec from the West and shows no intention of pulling into the intersection. However, the previous collision warning system triggers a lot of FP collision warnings for this scenario as shown in Figure 5.25b. The new system, shown in Figure 5.25b does not trigger any collision warning as a result of the improved pedelec trajectories and CD.

Finally, a scenario similar to the previous one is compared. The pedelec is riding on a priority road again from West to East, but the car does not see it and pulls into the intersection (from South to North). Figure 5.26 compares the collision warning system for the aforementioned scenario. In this case, both collision warning systems behave as expected and trigger a warning that is necessary to slow the pedelec down. The newly developed trajectories, CD and warnings strategy thus also detects actual collisions, opposed to simply ignoring everything to achieve a low FP rate.

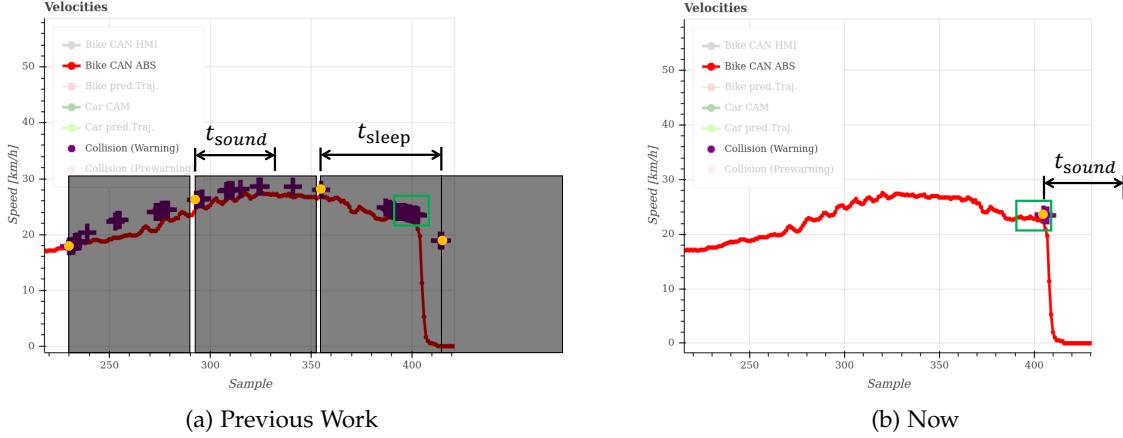


Figure 5.24.: Alarm Time Comparison for Parallel Driving Scenario with Car Turning Into Pedelec

Moreover, the new collision warning system features a pre-warning functionality, as explained in subsection 4.9.4, that gently suggests a lower speed to the pedelec by vibrating its handlebars well before the actual urgent warning for an emergency brake is triggered. Figure 5.27 shows the same scenario with respect to the pre-warning functionality. Hereby, the `const_speed_heading` model predicts the movements of both participants six seconds into the future by assuming that both road users will keep their current speed and heading (ride straight). The yellow trajectories belong to the pedelec and the black ones to the car. A triggered pre-warning event gets marked with a pink cross. In this specific scenario, had the cyclist reacted to the vibrating pre-warning and slowed down a bit, the car may have already pulled out of the intersection. Then, the path of the cyclist would have been clear without the need to initiate an emergency braking.

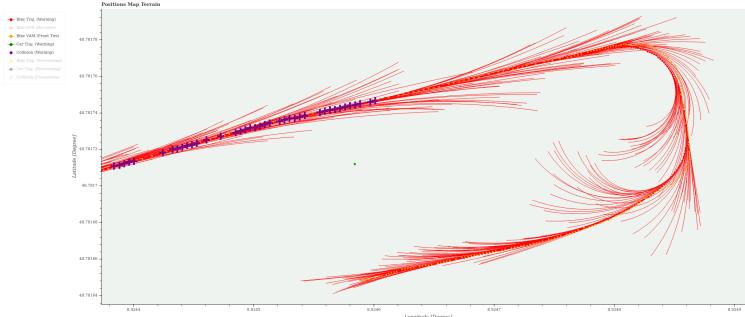
This subsection showed that not only do the individual improved components outperform the corresponding counterparts from the baseline individually, they work nicely together to form a far superior collision warning system. The new system immensely outperform the base system not only theoretically but more importantly in real-life.

5.3. Discussion of Results

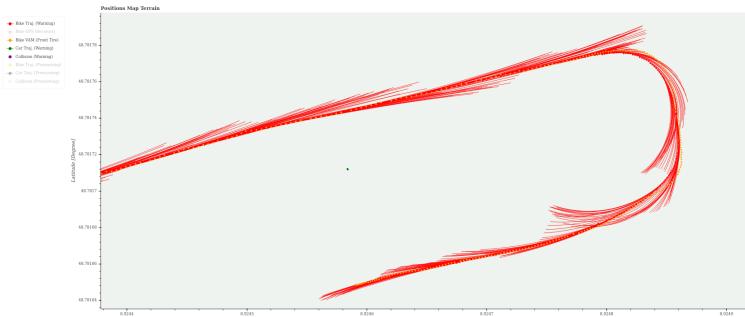
This section discusses the results of the intrinsic and extrinsic evaluations further. It also shows the theoretical potential of the `extrapol_speed_trapezoid_roll` model when the model parameters match closely to those of the actual curve.

Generally speaking, doubling the prediction horizon seems to at least quadruple the precisions for all models, even those where a linear relationship with respect to time was expected. Fortunately, the prediction errors of the position for $t_{\text{pred}} = 2 \text{ s}$ could be reduced drastically

5. Evaluation



(a) `const_a_yawr` and Dynamic Range-Based CD



(b) `extrapol_speed_const_roll` and SAT-Based CD

Figure 5.25.: Comparison of Collision Warning Systems for Car Yielding to Pedelec

enough from a covariance of roughly 2.5 m reported in [55] to around a meter for similar routes through Malsheim. Note that this does not yet fully include the positioning errors from the INS. As discussed in subsection 4.1.3, a minimum goal of the covariance was 2.0 m. Thus, the new trajectory prediction models can generally be used for the urgent warning function with $t_{\text{pred}} = 2$ s. However, The prediction errors for $t_{\text{pred}} = 4$ s are way too high to request an emergency brake of the cyclist with FP rates greater than 50%. However, these long trajectories may still be useful for the pre-warning functionality as introduced in subsection 4.9.4

During testing it was noticed that collision warnings arrive a bit too late when going fast and using $t_{\text{pred}} = 2$ s. Equation 5.2 shows a simple relationship for roughly determining the maximum initial speed v_i that a time horizon of $t_{\text{pred}} = 2$ s can handle at most in order to ensure a complete stop:

$$v = v_i + a_{\text{brake}} \cdot (t_{\text{TTS}} - t_{\text{react}}) \quad (5.1)$$

$$v_i = a_{\text{brake}} \cdot (t_{\text{TTS}} - t_{\text{react}}) \quad (5.2)$$

Under the best-case assumptions of $a_{\text{brake}} = -4.6 \frac{\text{m}}{\text{s}^2}$ and $t_{\text{react}} = 0.3$ s from subsection 4.9.3 and the time to stop is at the very end of the prediction horizon $t_{\text{TTS}} = t_{\text{pred}}$ the maximum initial pedelec speed thus amounts to $v_i = 7.65 \frac{\text{m}}{\text{s}} = 27.54 \frac{\text{km}}{\text{h}}$. This shows that $t_{\text{pred}} = 2$ s may not be long enough and extending the time horizon may be necessary at the cost of a

5. Evaluation

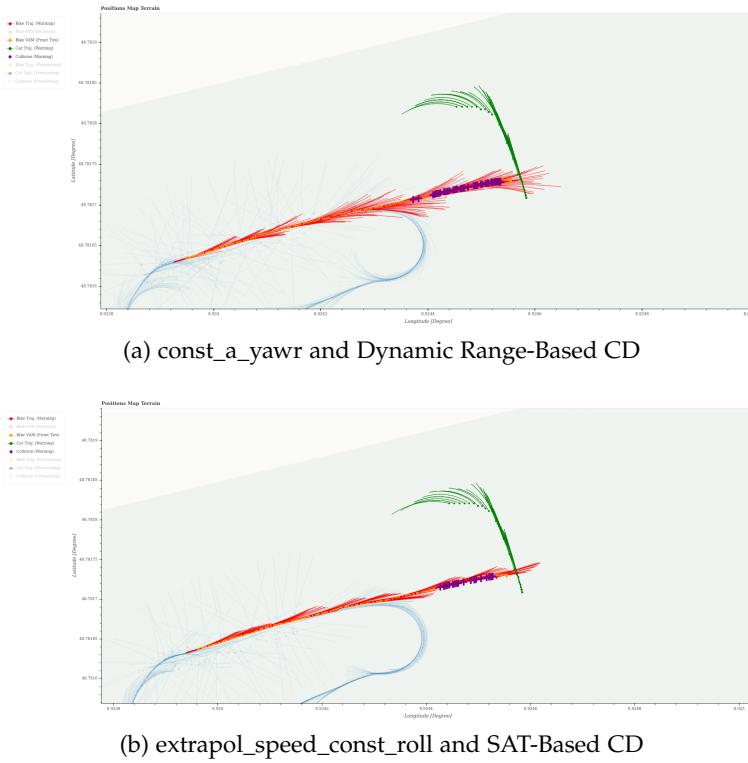


Figure 5.26.: Comparison of Collision Warning Systems for Car Taking Pedelec's Right-of-Way

higher FP rate because of roughly quadratically increasing errors. A better solution would be to improve the prediction errors for longer time horizons, e.g. by investigating some of the proposals in the next chapter.

Subsubsection 5.1.6.2 showed that setting the parameters of the `extrapol_speed_trapezoid_roll` as the mean values of statistical evaluation over all curves does not work well in practice. However, this model is still believed to have the greatest potential, despite the `extrapol_speed_extrapol_roll` model reaching better quantitative and qualitative results in subsection 5.1.6 for both $t_{\text{pred}} = 1 \text{ s}$ and $t_{\text{pred}} = 2 \text{ s}$. The model can compactly describe and predict various curves with a handful of parameters that can easily be limited to physical boundaries. For example, the maximum lateral acceleration through any curve may never exceed e.g. $7 \frac{\text{m}}{\text{s}^2}$. Figure 5.28b shows a scenario where the generated roll angle profile closely matches the actual roll angle signal for $t_{\text{pred}} = 2 \text{ s}$. Note that this is not a perfect match at first and the online adaptation corrects this profile slightly. The yaw rates in Figure 5.28a are just for comparison purposes to see how much smoother the roll angle signal is. Figure 5.29 shows the resulting trajectories and prediction errors in the aforementioned well-matched case. The predicted trajectories in Figure 5.29a nearly perfectly resemble the actual curve. The worst-case errors in the positions in Figure 5.29b do not even exceed 1.0 m. The worst-case angle error does not exceed 6 degrees as shown in Figure 5.29c. For comparison, Figure 5.30

5. Evaluation

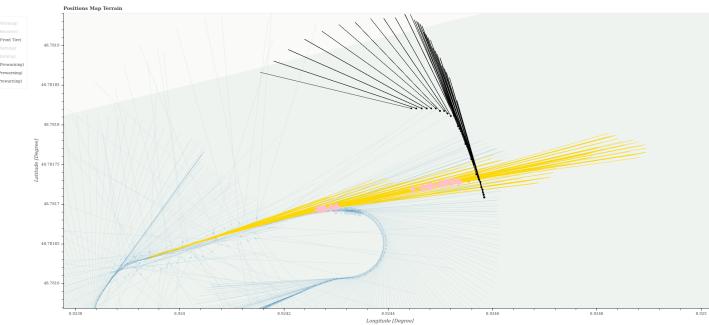


Figure 5.27.: Pre-Warning Functionality of New Collision Warning System

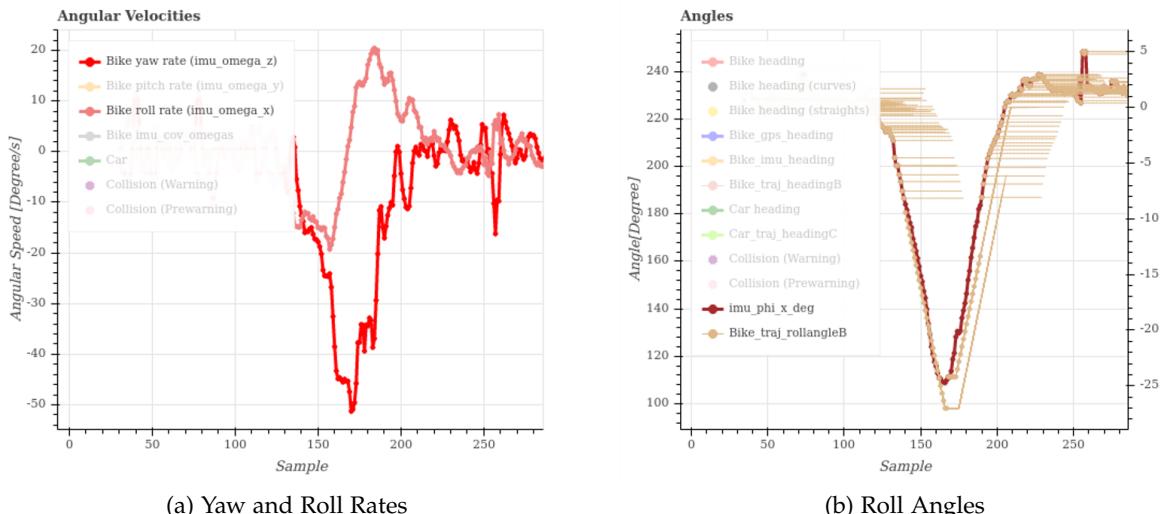


Figure 5.28.: Generated Roll Angle Profile Closely Matching Actual Roll Angle Signal

shows the trajectories and prediction errors for the exact same curve section with the base `const_a_yawr` model. In this case, the worst-case position errors reach 8.0 m, and the worst-case angle error reaches around 25 degree.

However, the simplistic switching mechanism of the `extrapol_speed_trapezoid_roll` model also has some drawbacks not accounted for in these evaluations. The current implementation of the VAM does not allow to communicate a useful confidence of the predicted trajectories from such a highly nonlinear switching model in a trustworthy manner. From the point of other road users, it does not seem trustworthy for a road user to suddenly switch from e.g. completely straight trajectory to a 90 degree turn from one time step to the other.

Of course, some real-life scenarios may be more complex than the exemplary ones shown in subsection 5.2.4. For example, in Figure A.6, the pedelec could evade a small object or overtake a pedestrian while staying on the side walk without any intention to momentarily

5. Evaluation

ride on the road. While in the first moment it may look like the cyclist ends up on the street, the ADAS would have to somehow recognize or learn that such a cyclist behavior is unlikely or accept FP in such cases.

All in all, the intrinsic and extrinsic evaluations showed that the newly integrated INS, novel trajectory prediction models, collision detection and warning strategy vastly improve the overall collision warning system on exemplary common real-life scenarios by reducing the number of FP drastically while retaining a high TP rate. Hence, the main goal of this thesis is achieved.

5. Evaluation

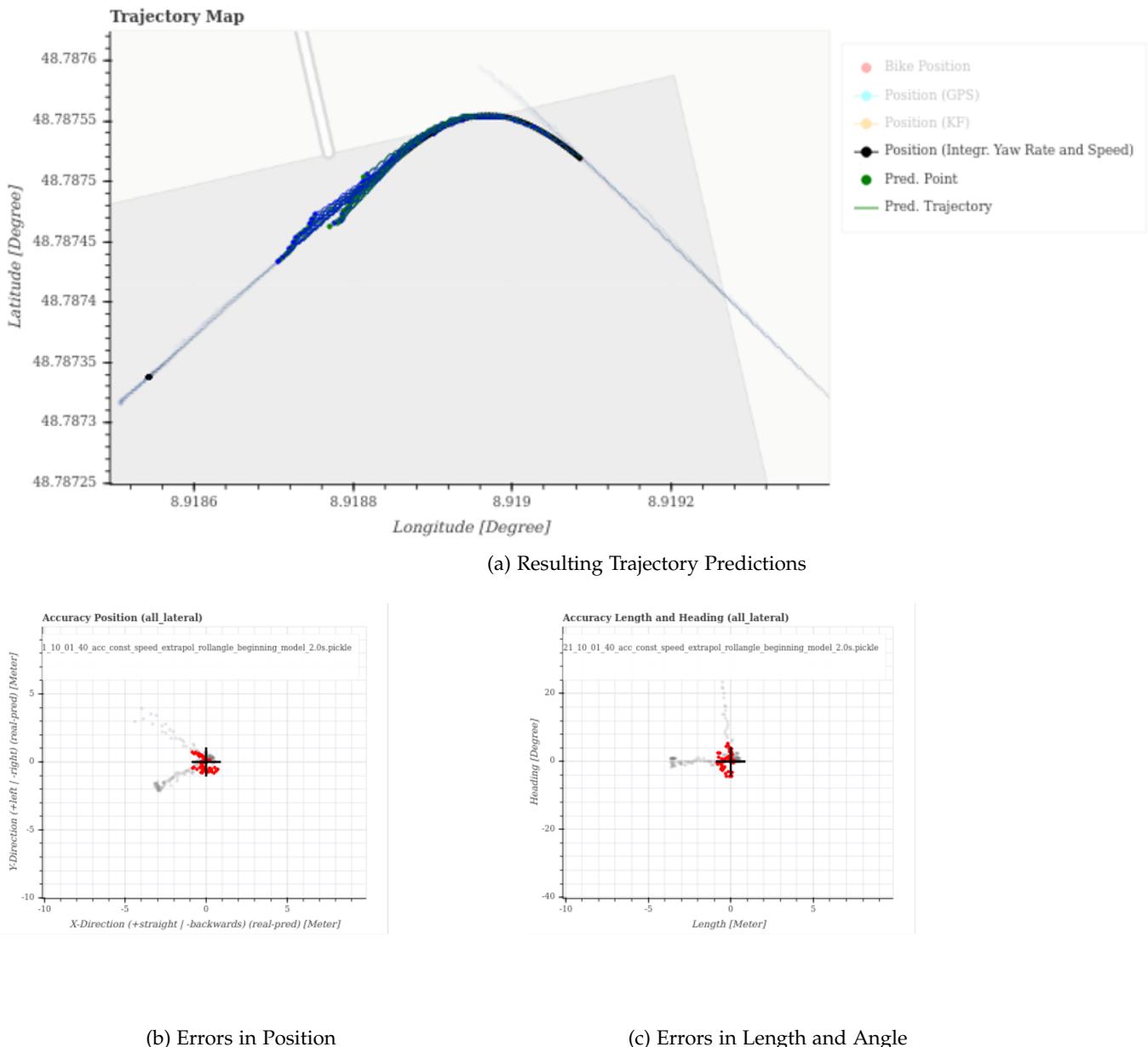


Figure 5.29.: Potential of Trajectories and Prediction Errors of `extrapol_speed_trapezoid-roll` Model

5. Evaluation

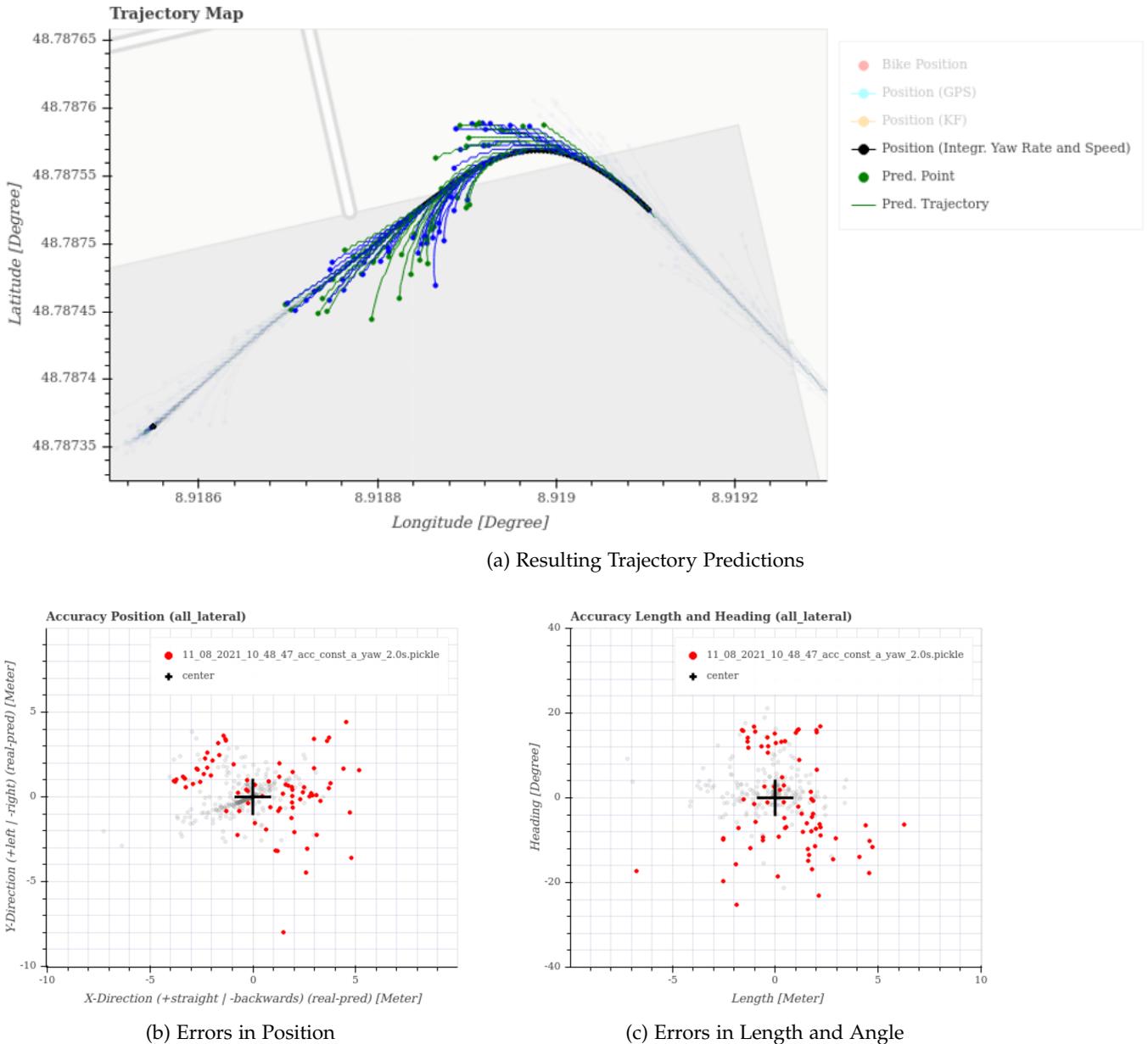


Figure 5.30.: Trajectory and Prediction Errors of Baseline for Scenario of Figure 5.29

6. Future Work

This chapter discusses what further investigations and improvements could be made to the collision warning system in future. First, it highlights expected challenges and ideas regarding the positioning and attitude estimation. Then, it proposes ideas to improve the trapezoidal extrapolation such as how to guess its parameters, more sophisticated switching between models and incorporating confidence values. Afterwards, it lists some to-be-solved issues with the collision detection and warning strategy. Towards the end, it summarizes some minor open problems from testing. Finally, it discusses general future topics such as resource optimizations, deployment and real-time capabilities.

The positioning and attitude accuracy, reliability and trustworthiness needs to be properly evaluated against a true reference system. Especially in urban environments, where GNSS outages or signal reflections are common, this may pose major challenges. Connecting the ABS WSS signal to the ZED-F9R receiver should improve the positioning accuracy of the current INS that already uses an internal EKF for sensor fusion.

As mentioned several times, the current EKF will likely have to get reworked completely in order to achieve sufficient positioning and state estimation in urban environments or when downgrading to low-cost sensors again. The current INS is way too expensive for series production. One approach could be replacing the current 2D state model of the EKF that can only rotate about the z-axis with a 3D bicycle model that incorporates the rolling motion as well (see subsection 3.2.3). This would allow the state estimation to also provide online filtered and fused roll angle, roll rate, and yaw rate signals. Moreover, this physical bicycle model may then also be used for the trajectory prediction by running the state estimation KF without updating. This would automatically return growing covariances regarding the predicted positions that can be used to describe the confidence of the trajectory prediction. Another approach would be to only estimate the relative positions of the pedelec another road users to each other e.g. via a moving base algorithm or fusing their data in a joint KF. As shown in subsubsection 5.1.4.1, the higher precision of the 56-toothed ABS WSS greatly improves the overall trajectory prediction accuracies. However, since most pedelecs do not (yet) feature an ABS, this high precision sensor will not be available in series. Instead, the ADAS function has to rely on the factory 1-impulse WSS of the HMI. Thus, it would be worthwhile to research into predicting more precise wheel speeds from the low-cost factory sensor e.g. via ML approaches.

The novel trapezoidal extrapolation method did not yet achieve its full theoretical potential in practice. Instead of statically setting pre-defined values for the $|\phi_{\min}|$, $|a_{y_{\max}}|$ and t_{const} parameters once, it would make sense to learn how to guess these roll angle parameters

6. Future Work

online for every specific situation. Predicting the correct parameters essentially has the same effect as predicting the switching point from one model to another. For example, these physically-limited parameter ranges can be binned/quantized into discrete values and used to train a ML classifier that tries to predict the correct bin for each parameter. The classification probability could be used as a confidence for the trajectory prediction, which is currently still missing and necessary for other road user to define appropriate reactions to the received pedelec trajectories. For example, high confidence colliding trajectories from the pedelec may enable a car to trigger a AEB without sacrificing their extremely high FP-requirements. On the other hand, low confidence trajectories may get ignored by the car even if they indicate a collision. A simple way to improve the timing of the the switches at the curve entry and exits may be to incorporate the roll rate as well. A high dynamic tight curve, typically signalized by high roll rates, should demand a lower roll angle threshold to switch to the roll angle extrapolation earlier. On the other hand, slowly changing roll angles at relatively large magnitude indicate a steady-state curve riding where the constant roll angle assumption works well. Switching to the extrapolation may be delayed to a higher roll angle threshold in this case.

This thesis showed that the roll angle signal generally is a very suitable signal for the lateral component of physics-based trajectory prediction models and can effectively replace the yaw rates. Similarly, the ML models may also have an easier time to extract and learn useful features from the much smoother and less complex roll angle signal compared to the yaw rate signal. However, it is believed that in addition to physical constraints developed in this thesis, additional higher-level constraints e.g. from map matching can greatly improve the trajectory predictions. For example, if the pedelec approaches a junction in a dense urban environment, the trajectory prediction model should have a higher probability of slowing down and/or turning opposed to a lonely country road where keeping the speed constant and following the road seems more probable. Additionally, any unfeasible trajectory predictions crossing buildings or other landmarks should get a really low confidence or ignored.

As described in section 5.3, the simplistic switching mechanism of the `extrapol1_speed_trapezoid_roll` model has some drawbacks. In future, it may make sense to perform some kind of probabilistic merge between these models to report a useful confidence and enable smoother transitions. Finally, the trapezoidal extrapolation method introduced a multi-model approach only for the lateral component of the trajectory prediction model. A future work could investigate switching between different models for the longitudinal component as well.

Eventually, the CD needs to extend to 3D space e.g. because a car could pass the pedelec under a bridge. The simplest way would be to suppress the alarm if the difference in altitude (only horizontal component) is above a predefined threshold under the assumption that an extremely rapid change in altitude is unlikely. Alternatively, the rectangles could be modelled as 3D boxes and compute their intersection in 3D space. However, this may add significant computational effort and may also require a 3D trajectory prediction as well. Furthermore, it may make sense to extend the CD with (probabilistic) indicators of crash criticality and/or likelihood. For example, SAT could compute a penetration vector to eval-

6. Future Work

uate how critical or likely a collision actually is under the uncertainties of the positioning and trajectory prediction: The larger the penetration vector is, the more severe or more likely a collision is in reality because a small deviation in the vehicles' predicted positions would still likely result in a crash. Another way could be to compute Intersection over Union (IoU) that measures the polygon overlap. Similarly, the more overlap, the more critical or likely a collision actually is. In both cases, instead of stopping at the first collision, the CD would need to proceed until the objects are safely apart again and potentially evaluate multiple crash configurations to obtain a maximum or average IoU. The idea is to use the collision criticality/liability of as an additional condition (e.g. above certain threshold) to trigger an emergency warning in order to further improve the FP rate. This approach may put less focus on the accuracy of the trajectories. A simplistic substitute of this may be using a k-out-of-n majority voter. Hereby, a warning is only triggered if k out of the last n positions resulted in predicted collisions. This would delay the collision warning but would eliminate warnings due to trajectory outliers and thus lower the FP rate. Generally, more real-world end-to-end evaluations of the entire ADAS are necessary as this thesis only touched upon some of the most basic scenarios.

The speed recommendation of the pre-warning functionality is not yet finished. In some cases it may be better if the pedelec went faster because the cyclist will not achieve a full stop before the stopping location anymore, similar to Figure 4.24. Alternatively, such a scenario could be coordinated using V2V. For example, the pre-warning function could gently request the cyclist via vibrations to speed up such that the cyclist makes it out of the critical collision zone in time. However, this would require coordination among the vehicles to ensure that e.g. the car does not accelerate as well which would result in a collision again.

During final testing with a real car and truck, the collision warning system generally worked well. Some minor problems seem to exist with the visualization in the HMI and outputting of the warnings. The screens sometimes toggles between the pre-warning and urgent warning display. Also, the vibration and speakers sometimes only quickly blip instead of playing the entire vibration profile or sound file. There may be an electromagnetic interference in the audio card of the NUC or the python multiprocessing for outputting the analog signal does not work reliably. The `extrapol_speed_trapezoid_roll` model needs to be made more robust. There seems to be a sporadic bug, which has not yet been analyzed.

While some of the crucially time-consuming operations such as the collision detection and major loops have been vectorized for efficient computation with *NumPy*, there still exists plenty of optimization potential. For example, Equation 4.28 is called within a loop to iterate through every point in time of the trajectory. There may exist a way to execute all of these transformations at once, by storing all points in a matrix and performing matrix multiplication in a higher dimensional space. Also, intermediate variables are often introduced for better readability which take up more memory resources than needed. These optimizations are not a high priority for now and become more relevant once the software needs to be ported to an embedded system target. In this case, migrating the Python code entirely to C++ would be

6. Future Work

advisable. However, depending on how sophisticated the collision warning system will get, especially the trajectory prediction, it may make sense to deploy parts or the whole system on the cloud and only send the raw sensor data and inference results back and forth with a low-latency wireless connection. Managing the bandwidth of thousands of users trying to run inference on the trajectory prediction model could pose a major challenge, though. Because the collision warning function is safety critical, the system should actively enforce real-time capability at some point. If the system stays in ROS, a switch from ROS1 to ROS2, which supports real-time computing better, would make sense. Alternatively, a light-weight real-time operating system e.g. FreeRTOS could be used if the final system is deployed on a microcontroller. In case of the cloud computing deployment, ensuring real-time capabilities could mean significant efforts because of the communication latency.

7. Conclusion

The overall goal of this thesis revolved around improving the FP rate of a prototypical collision warning system for pedelecs based on V2V communication.

First, relevant components of the existing hardware and software from previous works were analyzed. The thesis noted the noisy and oscillating signals due to vibrations and balancing motion of the cyclist. It came to the conclusion that filtering these artefacts out without over-damping the useful signal and introducing a large delay would be non-trivial.

The positioning from the *Navilock NL-8012U* GNSS with an accuracy between 2.0 - 4.0 m turned out way too imprecise for the collision warning function to trigger any useful alarms. Furthermore, the thesis unraveled a previously unknown problem with the roll angle estimation of the BNO055 AHRS that showed an offset of roughly 5 degrees after a single curve. For these reasons, both sensors were replaced with a *simpleRTK2B-F9R* INS that offers up to cm-level accurate positions and 0.5 degree accuracy in its roll angle estimation. However, even this system showed slightly inaccurate positioning in curves with respect to the pedelec. Achievable roll angles of up to 40 degree can cause a wrong projection of the positions onto the street plane that are offset by up to 0.27 m inward to the curve. The thesis derives a coordinate transformation that corrects this projection error based on the mounting position and attitude readings of the INS. Moreover, this transformation virtually offsets the reported positions at the INS frame to a point in the front wheel of the pedelec for communication purposes to other road users.

Besides, an automation that sequentially executes a number of experiments by iterating through a set of pre-defined parameters is developed. All configurations, source code, tables and plots are stored in a unique folder that allows easily matching and reproducing results. Similarly, an end-to-end validation method using a pre-defined virtual vehicles was developed to test the entire collision warning system without depending on the availability of another vehicle.

Then, the thesis investigated what signals and prediction methods are most suitable for the longitudinal and lateral components of physics-based trajectory prediction models. It turns out that a linearly extrapolated speed significantly performs best for the longitudinal component. Hereby, using the ABS WSS instead of the HMI WSS is key as this improved the precision in x-direction by roughly half a meter for $t_{\text{pred}} = 2 \text{ s}$. In total, the precision in x-direction is improved by roughly 0.4 m, 1.6 m and 5.0 m for $t_{\text{pred}} = 1 \text{ s}$, $t_{\text{pred}} = 2 \text{ s}$ and $t_{\text{pred}} = 4 \text{ s}$ compared to the constant acceleration baseline.

Moreover, the thesis finds that the roll angle seems to be the one of the first and most reliable signals to use when predicting future trajectories of curves. The thesis derives a relationship to compute a yaw rate signal from the roll angles and validated it. The yaw rates from roll

7. Conclusion

angles generally match the overall signal curve from the measured yaw rates from the gyro. As desired, the roll angle-based signal slightly leads ahead and generally is much smoother. Evaluations confirmed that incorporating the roll angle instead of the measured yaw rates for the lateral component leads to higher accuracies. For example, replacing the constant yaw rate assumption with a constant roll angle assumption lead to an improvement of the precision in y-direction by roughly half a meter for $t_{\text{pred}} = 2 \text{ s}$. Then, a novel trapezoidal extrapolation method is designed for predicting curves. It aims to improve the trajectories at the curve entries and exits by using a limited linear extrapolation of the roll angle signal and modelling the steady-state part using the constant roll angle assumption. This hypothesis of switching between multiple models theoretically offers great potential for predicting many types of curves but in practice still falls short of its expectations. Visual analysis showed that the statistically set parameters do not match the actual curve behavior well in many cases and lead to great under- or overestimated curves. The thesis proposes a simple improvement to adapt the roll angle threshold based on current roll rates. Moreover, it provides ideas for hybrid models that use ML to predict the parameters of this model for each curve specifically by additionally incorporating information from maps. It also notes that incorporating a prediction confidence needs to be implemented as well and that the hard-switching needs improvement as well.

In overall, all of the novel physics-based trajectory prediction models showed vastly superior performance to the physics-based and ML-based baselines from this project. For example, the trapezoidal extrapolation method improves the baseline significantly by 38% and 59% in the predicted position accuracy and precision at the end of a two second time horizon.

However, section 5.3 discussed that $t_{\text{pred}} = 2 \text{ s}$ might be too short to ensure a complete stop at higher speeds and that trajectory prediction errors for $t_{\text{pred}} = 4 \text{ s}$ are not yet small enough to be used as the urgent warning. Therefore, further improving the trajectory predictions is necessary. While pure ML approaches applied to the raw speed and gyro sensor signals performed poorly thus far, hybrid models with the trapezoidal extrapolation of the roll angle and ML predicted parameters seem promising. The resulting trajectories could be more accurate, explainable, safe and trustworthy. The improved accuracy would arise from appropriate predicted parameter for every individual curve by the ML part. The physics-based part would enable improved safety by physically limiting the parameters to prevent unrealistic trajectories. The classification probability of the network may be used in some way as a confidence measure for the trajectory.

The thesis split up the collision warnings into two parts: detecting the collisions via intersection of rectangles that enclose the road users and and a braking distance based alarm. An initial polygon intersection library that computes the intersection points proved insufficiently slow. Thus, a binary SAT algorithm is used to detect rectangle collisions about 800 times faster. Moreover, the thesis notes that the last safe stopping point, which may lie before the actual collision point, should be used. Ergo, a custom algorithm that uses the SAT algorithm in a backtracking fashion is developed to find this safe stopping point. The timing of the collision alarm to the cyclist is investigated and set to depend on the remaining braking

7. Conclusion

distance to this last safe stopping location. For example, this prevents premature alarms when a bicycle is riding slowly. Simulations with perfect positioning and trajectory predictions confirm that the new CD greatly reduces the FP rate for many common urban scenarios compared to the previous CD. Moreover, the thesis proposes a pre-warning functionality that aims to additionally stimulate a more gentle but earlier reaction of the cyclist in order to prevent the pedelec from ending up in the critical zone in the first place. However, the actual speed recommendation for the cyclist is not yet computed and coordination with the other road user via V2V may be necessary.

Final testing with a real car and truck confirm the effectiveness of all improvements by greatly reducing the FP rate on common real-world scenarios. Minor findings mostly revolved around displaying and outputting the warnings. All in all, this thesis immensely improved the collision warning system and thus offers great potential to improve road safety, especially for cyclists.

A. Appendix

A.1. Review of Homogeneous Transformation

Quick review of homogeneous transformation in three-dimensional space. Translation of a vector:

$$\mathbf{t} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation of vector by an angle of ϕ about x-axis:

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}$$

Rotation of vector by an angle of θ about y-axis:

$$\mathbf{R}_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$$

Rotation of vector by an angle of ψ about z-axis:

$$\mathbf{R}_z = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

General rotations are described via matrix multiplication of the three rotation matrices in the following order:

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_z \cdot \mathbf{R}_y \cdot \mathbf{R}_x = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix} \\ &= \begin{bmatrix} \cos \psi \cos \theta & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\ \sin \psi \cos \theta & \sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi \\ -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi \end{bmatrix} \end{aligned}$$

The general homogeneous transformation contains a rotation matrix and a translation vector:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

A. Appendix

Finally, the homogeneous transformation from a position i expressed in frame $\{1\}$, denoted with ${}^1\mathbf{p}_i$ to the position i in frame $\{0\}$, denoted with ${}^0\mathbf{p}_i$, states:

$${}^0\mathbf{p}_i = {}^0\mathbf{T} \cdot {}^1\mathbf{p}_i$$

Note that the position vectors are in homogeneous coordinates, which means a one is appended to each of them.

A.2. Full Derivations

Full derivation of quadratic acceleration term with respect to time when integrating a constant acceleration twice over time (Equation 4.5):

$$a(t) = \frac{d}{dt}v(t) \quad \text{and} \quad v(t) = \frac{d}{dt}x(t) \quad (\text{A.1})$$

$$\int a(t)dt + C_1 = \int \frac{d}{dt}v(t)dt \quad (\text{A.2})$$

$$\int a(t)dt + C_1 = v(t) \quad a(t) \text{ constant w.r.t. time} \quad (\text{A.3})$$

$$at + C_1 = v(t) \quad \text{where} \quad a \cdot 0 + C_1 = v(0) \Rightarrow C_1 = v_0 \quad (\text{A.4})$$

$$at + v_0 = v(t) \quad (\text{A.5})$$

$$at + v_0 = \frac{d}{dt}x(t) \quad (\text{A.6})$$

$$\int (at + v_0)dt + C_0 = \int \frac{d}{dt}x(t)dt \quad (\text{A.7})$$

$$\frac{1}{2}at^2 + v_0t + C_0 = x(t) \quad \text{where} \quad \frac{1}{2}a \cdot 0^2 + v_0 \cdot 0 + C_0 = x(0) \Rightarrow C_0 = x_0 \quad (\text{A.8})$$

$$\frac{1}{2}at^2 + v_0t + x_0 = x(t) \quad (\text{A.9})$$

Full derivation of the linear yaw rate term with respect to time when integrating a constant yaw rate once (Equation 4.6):

$$\omega_z(t) = \frac{d}{dt}\psi(t) \quad (\text{A.10})$$

$$\int \omega_z(t)dt + C_0 = \int \frac{d}{dt}\psi(t)dt \quad (\text{A.11})$$

$$\int \omega_z(t)dt + C_0 = \psi(t) \quad \omega_z(t) \text{ constant w.r.t. time} \quad (\text{A.12})$$

$$\omega_z t + C_0 = \psi(t) \quad \text{where} \quad \omega_z \cdot 0 + C_0 = \psi(0) \Rightarrow C_0 = \psi_0 \quad (\text{A.13})$$

$$\psi(t) = \omega_z t + \psi_0 \quad (\text{A.14})$$

A. Appendix

Full derivation of quadratic speed term with respect to time when integrating a linearly extrapolated speed (Equation 4.7):

$$v(t) = \frac{d}{dt}x(t) \quad (\text{A.15})$$

$$\int v(t)dt + C_0 = \int \frac{d}{dt}x(t)dt \quad (\text{A.16})$$

$$\int v(t)dt + C_0 = x(t) \quad v(t) \text{ linear w.r.t. time (first order polynomial)} \quad (\text{A.17})$$

$$\int (p_1t + p_0)dt + C_0 = x(t) \quad (\text{A.18})$$

$$\frac{1}{2}p_1t^2 + p_0t + C_0 = x(t) \quad \text{where } \frac{1}{2}p_1 \cdot 0^2 + p_0 \cdot 0 + C_0 = x(0) \Rightarrow C_0 = x_0 \quad (\text{A.19})$$

$$\frac{1}{2}p_1t^2 + p_0t + x_0 = x(t) \quad (\text{A.20})$$

Full derivation of the quadratic yaw rate term with respect to time when integrating a linearly extrapolated yaw rate (Equation 4.8):

$$\omega_z(t) = \frac{d}{dt}\psi(t) \quad (\text{A.21})$$

$$\int \omega_z(t)dt + C_0 = \int \frac{d}{dt}\psi(t)dt \quad (\text{A.22})$$

$$\int \omega_z(t)dt + C_0 = \psi(t) \quad \omega_z(t) \text{ linear w.r.t. time (first order polynomial)} \quad (\text{A.23})$$

$$\int (p_1t + p_0)dt + C_0 = \psi(t) \quad (\text{A.24})$$

$$\frac{1}{2}p_1t^2 + p_0t + C_0 = \psi(t) \quad \text{where } \frac{1}{2}p_1 \cdot 0^2 + p_0 \cdot 0 + C_0 = x(0) \Rightarrow C_0 = x_0 \quad (\text{A.25})$$

$$\frac{1}{2}p_1t^2 + p_0t + \psi_0 = \psi(t) \quad (\text{A.26})$$

Full derivation of rider's reaction distance (Equation 4.39):

$$v(t) = \frac{d}{dt}x(t) \quad (\text{A.27})$$

$$x(t) = \int v(t)dt + C_0 \quad (\text{A.28})$$

$$x(t_{\text{react}}) = v_i \cdot \int_0^{t_{\text{react}}} dt + x_0 \quad (\text{A.29})$$

$$x(t_{\text{react}}) - x_0 = v_i(t_{\text{react}} - 0) \quad (\text{A.30})$$

$$d_{\text{react}} = v_i \cdot t_{\text{react}} \quad (\text{A.31})$$

A. Appendix

Full derivation of pedelec's braking distance (Equation 4.40):

$$a = \frac{dv}{dt} = \frac{dv}{dx} \cdot \frac{dx}{dt} = \frac{dv}{dx} \cdot v \quad (\text{A.32})$$

$$\int_{x_i}^{x_t} adx = \int_{v_i}^{v_t} vdv \quad (\text{A.33})$$

$$a[x]_{x_i}^{x_f} = [\frac{v^2}{2}]_{v_i}^{v_f} \quad (\text{A.34})$$

$$a(x_f - x_i) = \frac{v_f^2}{2} - \frac{v_i^2}{2} \quad (\text{A.35})$$

$$v_f^2 - v_i^2 = 2a(x - x_i) \quad (\text{A.36})$$

$$v^2 = v_i^2 + 2a(x - x_i) \quad (\text{A.37})$$

$$0 = v_i^2 + 2a_{\text{brake}}(d_{\text{brake}}) \quad (\text{A.38})$$

$$d_{\text{brake}} = \frac{v_0^2}{2a_{\text{brake}}} \quad (\text{A.39})$$

A.3. Supplementary Tables

Model (using HMI WSS)	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
const_a_LSTM_yawr (as in [55])	-0.14	2.15	0.25	2.13	-0.27	2.97	1.89	12.90
LSTM_speed_const_yawr (as from [55])	-0.05	1.77	0.29	1.23	-0.25	2.02	-1.18	20.82

Table A.1.: Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 1$ s Averaged over 1724 s Malmsheim 1 Trip

Model (using HMI WSS)	$\mu_{\Delta x}$ [m]	$\sigma_{\Delta x}$ [m]	$\mu_{\Delta y}$ [m]	$\sigma_{\Delta y}$ [m]	$\mu_{\Delta l}$ [m]	$\sigma_{\Delta l}$ [m]	$\mu_{\Delta \sigma}$ [deg]	$\sigma_{\Delta \sigma}$ [deg]
const_a_LSTM_yawr (as in [55])	-0.15	3.87	0.82	1.99	-0.29	3.81	4.83	20.44
LSTM_speed_const_yawr (as from [55])	-0.03	2.94	0.06	2.00	-0.11	3.01	0.20	19.41

Table A.2.: Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 2$ s Averaged over 1724 s Malmsheim 1 Trip

A.4. Supplementary Figures

A. Appendix

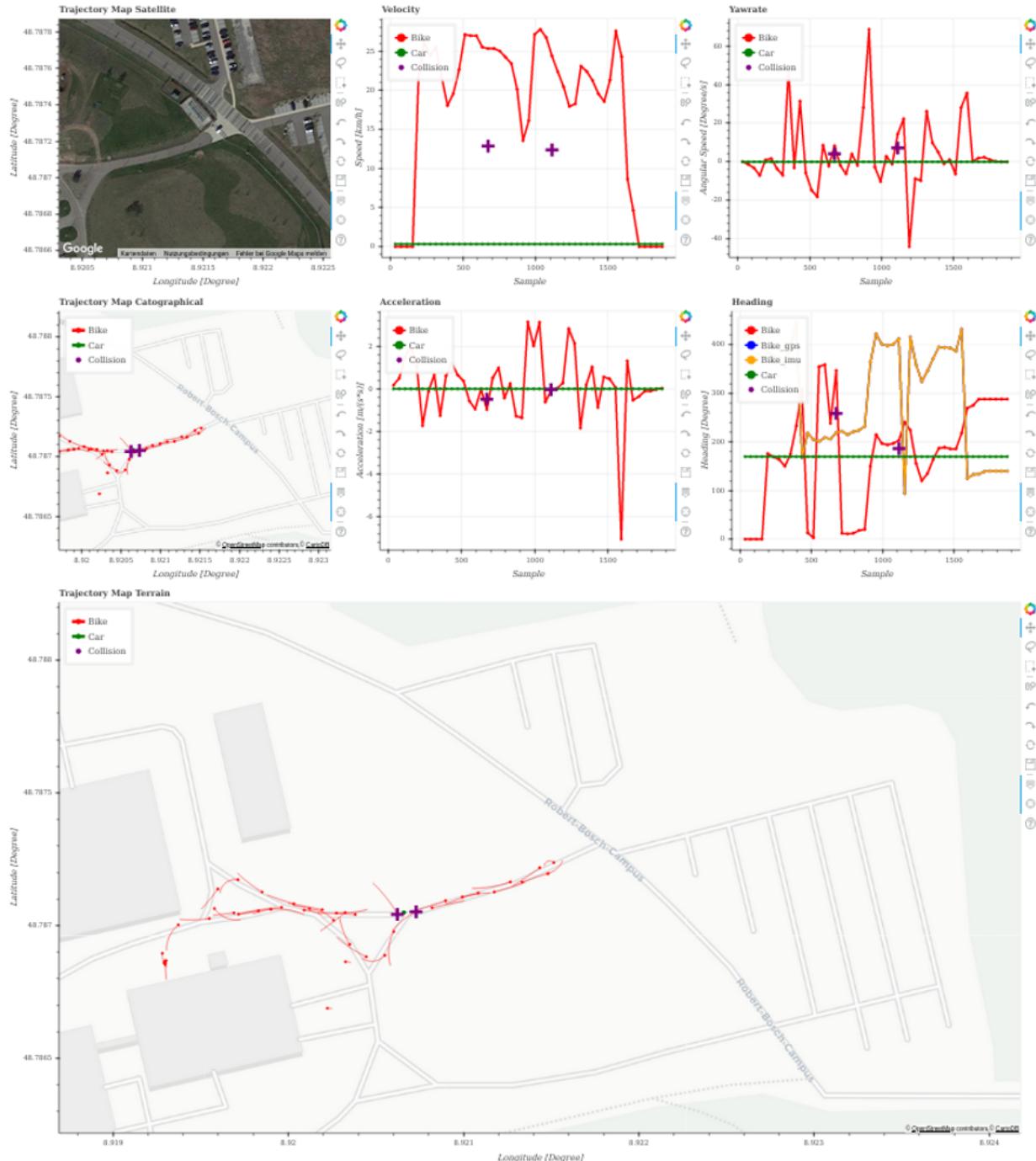


Figure A.1.: Initial Visualizations of the Signals

A. Appendix

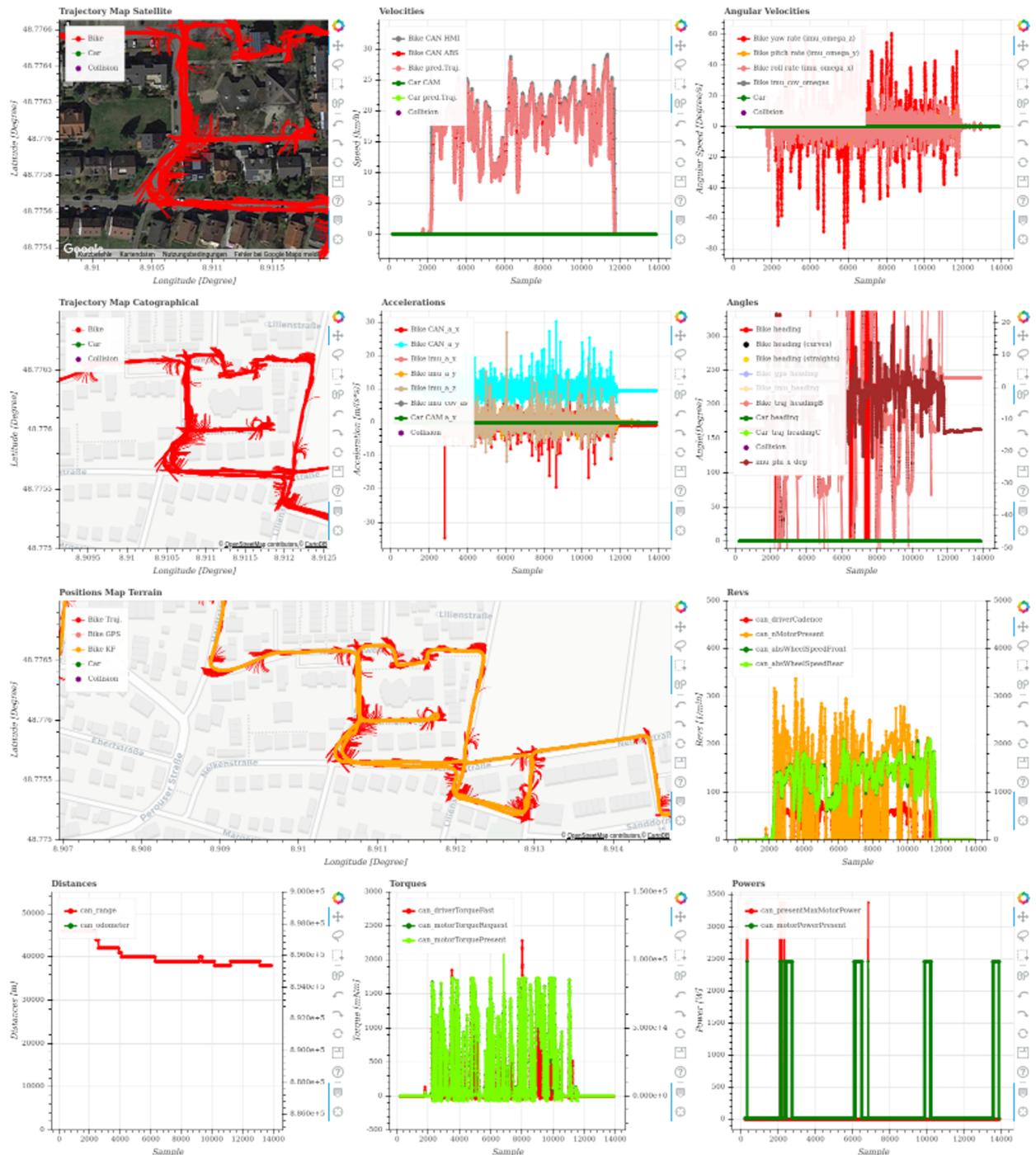


Figure A.2.: Adapted Visualizations of the Signals

A. Appendix

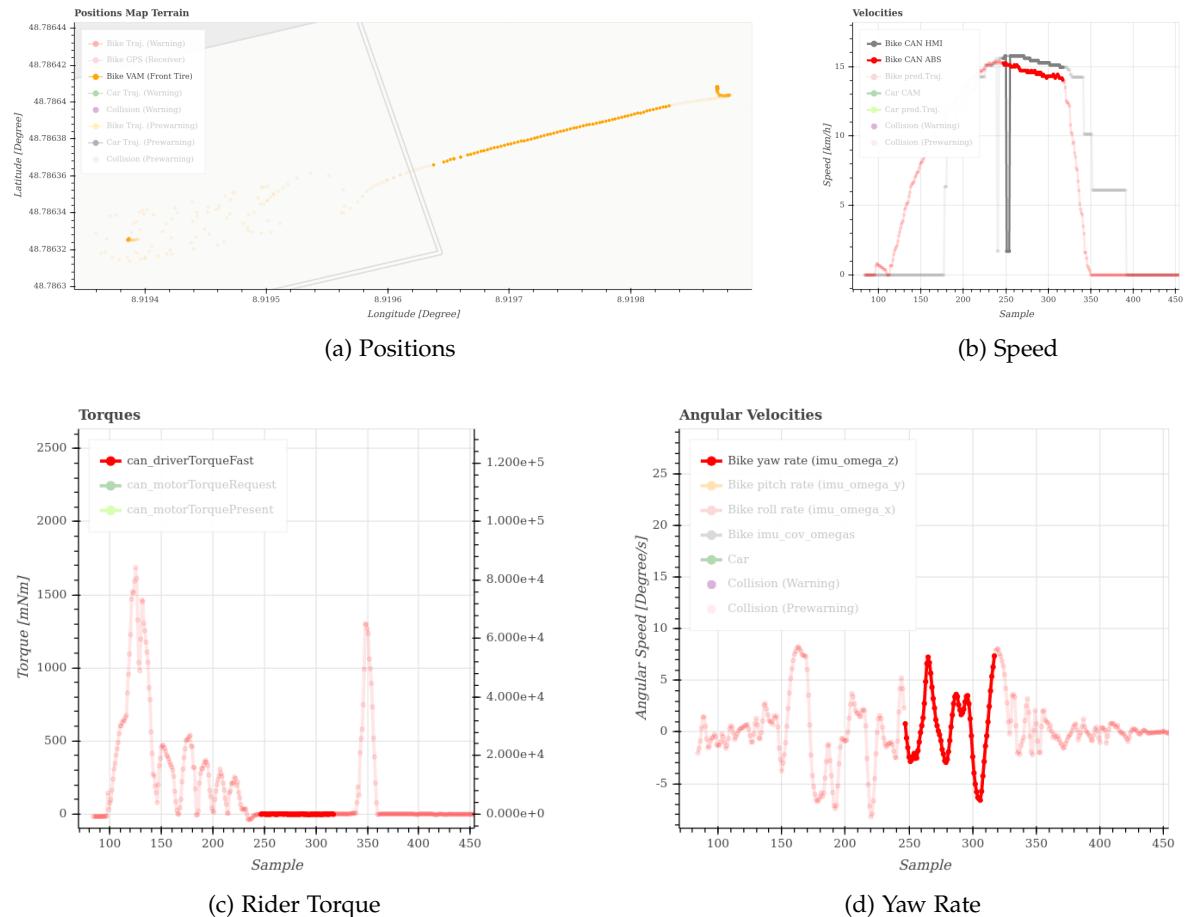


Figure A.3.: SimpleRTK2B: Oscillating Yaw Rates When Coasting for Around 4 Seconds

A. Appendix

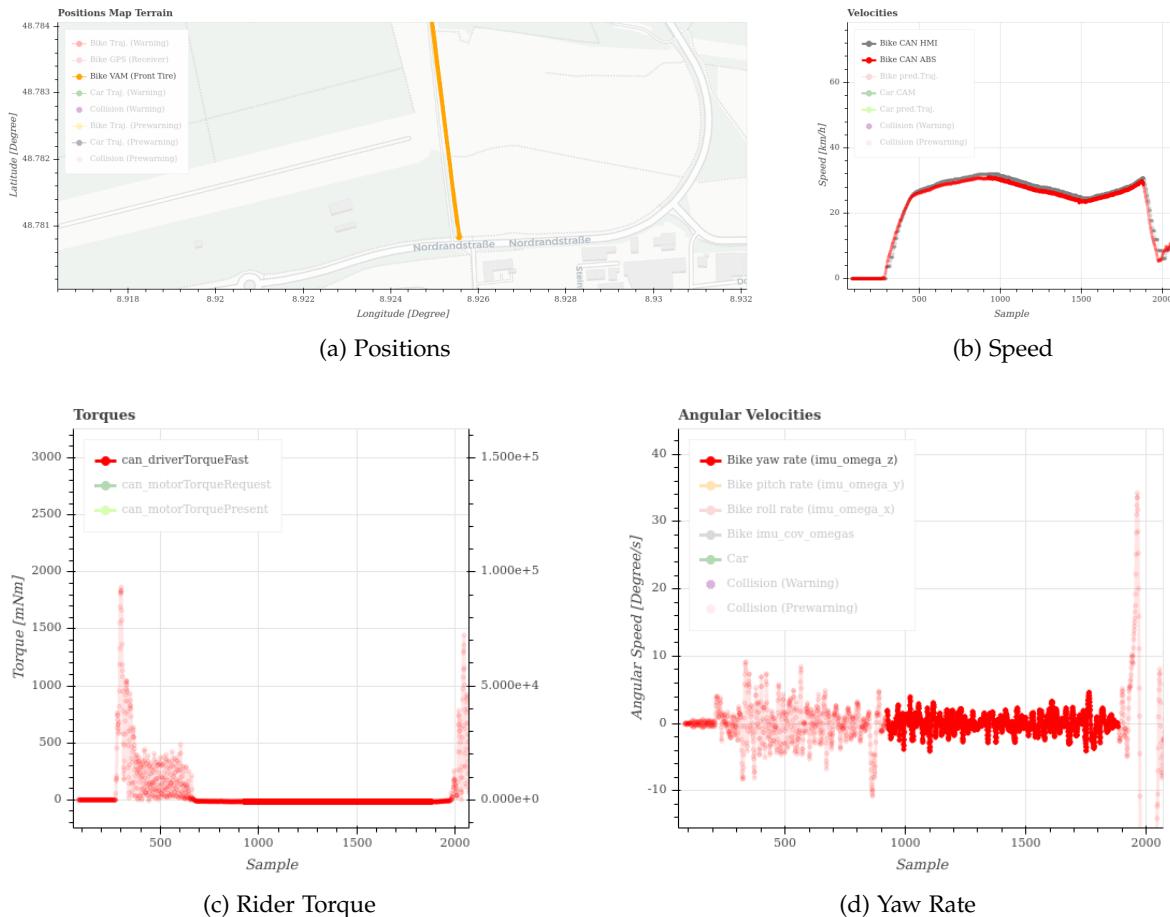
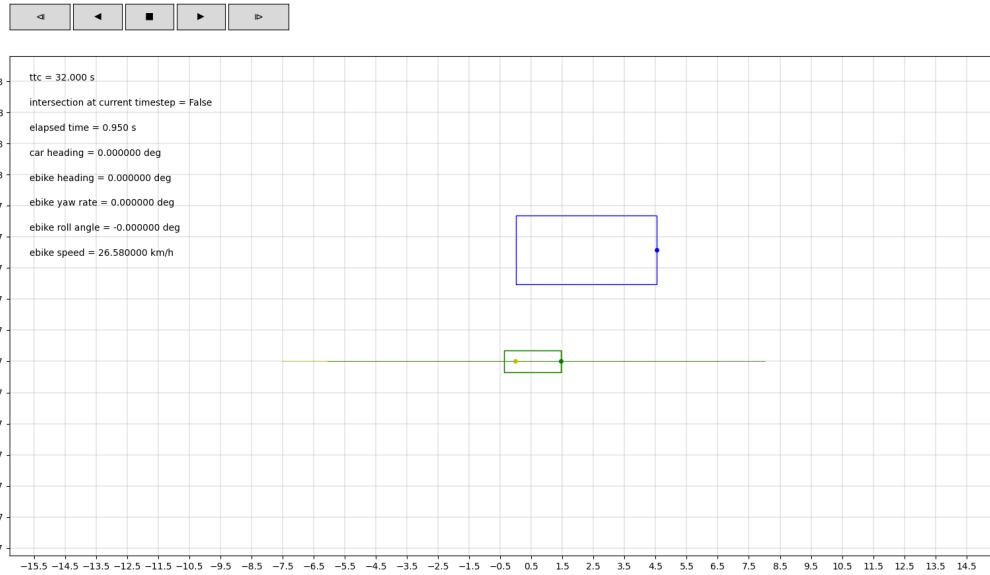
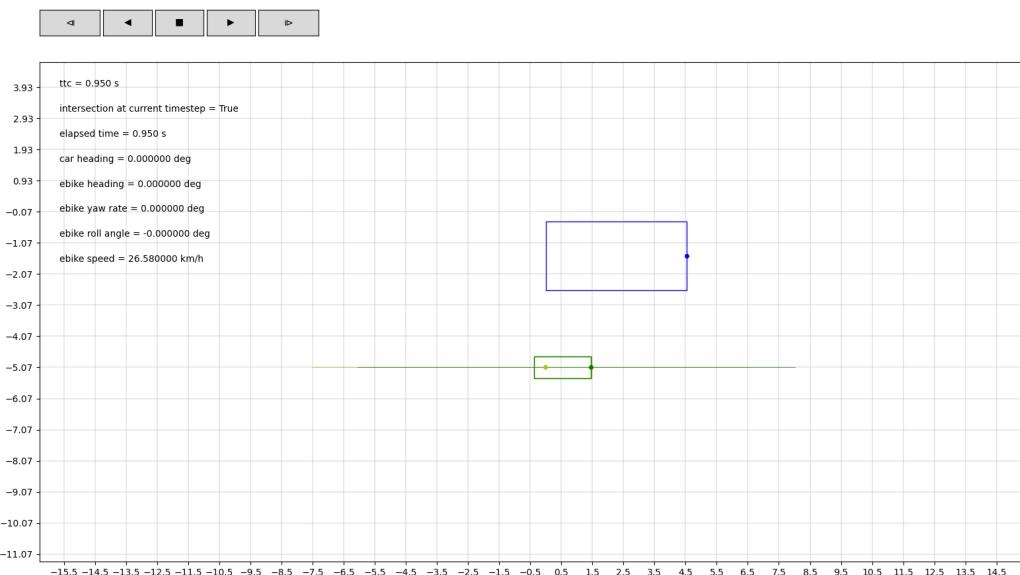


Figure A.4.: Ellipse-N: Oscillating Yaw Rates when Coasting for Around 50 Seconds

A. Appendix



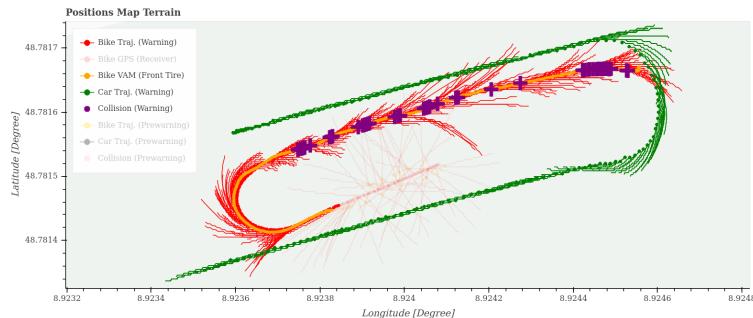
(a) Dynamic Range-Based CD



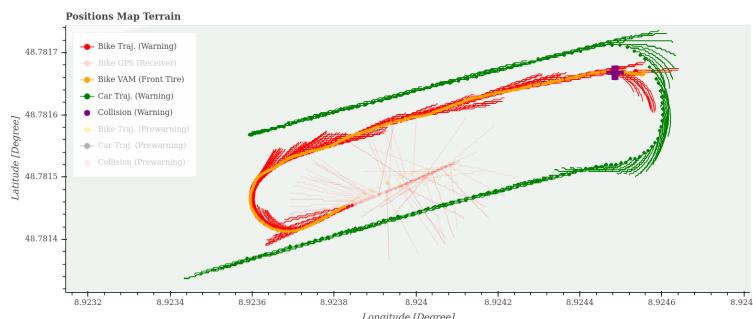
(b) SAT-Based CD

Figure A.5.: Collision Detection Comparison for a Stationary Car Passed by a Pedelec on the Side Walk

A. Appendix



(a) const_a_yawr Model and Dynamic Range-Based CD



(b) extrapol_speed_trapezoid_roll Model and SAT-Based CD

Figure A.6.: Comparison of Collision Warning Systems for Parallel Driving Scenario with Car Turning Into Pedelec

Acronyms

1D	one-dimensional
1T2W	Single-Track Two-Wheelers
2D	two-dimensional
3D	three-dimensional
4D	four-dimensional
AABB	Axis Aligned Bounding Box
ABS	Anti-lock Braking System
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
AEB	Autonomous Emergency Braking
AHRS	Attitude and Heading Reference System
AUC	Area Under the Curve
BNO055	Bosch Sensortec BNO055 USB-Stick Eval Kit
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CCD	Continuous Collision Detection
CCU	Communication Control Unit
CD	Collision Detection
CNN	Convolutional Neural Network
CoG	Center of Gravity
DC-DC	Direct Current-to-Direct Current
DoF	Degree of Freedom

A. Appendix

DU	Drive Unit
ECEF	Earth-centered - Earth-fixed
ECI	Earth-centered inertial
EKF	Extended Kalman Filter
ETSC	European Transport Safety Council
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FIS	Fuzzy Inference System
FN	False Negative
FP	False Positive
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMI	Human Machine Interface
IERS	International Earth Rotation Service
IIR	Infinite Impulse Response
IMM	Interacting Multiple Model
I-MoLDS	Informed Mixture of Linear Dynamical Systems
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IoU	Intersection over Union
ITS	Intelligent Transport Systems
KF	Kalman Filter
LDS	Linear Dynamical System
Lidar	Light Detection and Ranging
LSTM	Long Short-Term Memory
MCU	Microcontroller Unit
MEMS	Micro Electro Mechanical System

A. Appendix

ML	Machine Learning
MLP	Multilayer Perceptron
NCAP	European New Car Assessment Programme
NTRIP	Networked Transport of RTCM via Internet Protocol
NUC	Next Unit of Computing
OBB	Oriented Bounding-Boxes
OEM	Original Equipment Manufacturer
PWM	Pulse-Width Modulation
Radar	Radio Detecting and Ranging
RFR	Random Forest Regressor
ROS	Robot Operating System
RTK	Real Time Kinematics
SAE	Society of Automotive Engineers
SAT	Separating Axis Theorem
SHT	Separating Hyperplane Theorem
SNR	Signal-to-Noise Ratio
TP	True Positive
TTC	Time-To-Collision
TTS	Time-To-Stop
TUM	Technical University of Munich
U-MoLDS	Uninformed Mixture of Linear Dynamical Systems
USB	Universal Serial Bus
V2V	Vehicle-to-Vehicle
VAM	Vulnerable Road User Awareness Message
VRU	Vulnerable Road User
WGS 84	World Geodetic System 1984
WSS	Wheel Speed Sensor
ZED-F9R	u-blox F9 high precision dead reckoning module

List of Figures

1.1.	Intersection Without Line of Sight	2
1.2.	Visualized Trajectories for Pedelec and Car	4
2.1.	Test Vehicle with Mounted Processing and Measurement Hardware	7
2.2.	Geodetic Latitude ϕ and Geocentric Latitude $theta$ [32]	10
2.3.	Reference Frames $\{i\}$, $\{WGS84\}$ and $\{Street\}$	11
2.4.	Pedelec Reference Frames	12
2.5.	Coordinate Frame of the BNO055 USB-Stick AHRS. Adapted from [17]	13
2.6.	Sensor Reference Frames	14
2.7.	Separating Hyperplane and Separating Axis	17
2.8.	Minimum and Maximum Projections of Edges onto the Normal Vectors	18
3.1.	Software Architecture of the Collision Warning. Adapted from [17]	21
3.2.	Trajectory Computed from Piece-Wise Linear Segments	22
3.3.	Distance between Corresponding Trajectory Points and CD Radius. Adapted from [17]	23
3.4.	Simplified Architecture of the ML-Based Multi-Step Prediction Models [54] . .	25
3.5.	Trajectory Prediction Error Metrics. Adapted from [55]	27
3.6.	Comparison of <i>static_range</i> - and <i>dynamic_range</i> -Methods from an Aerial Perspective [55]	28
3.7.	Sketch With Parameters of the Mathematical Getz and Zhang Bicycle Model [66]	32
4.1.	Aliasing Effect When Visualizing Only Trajectory Endpoints	40
4.2.	Comparison of Signal Quality between Car and Pedelec	41
4.3.	BNO055-USB-Stick: Pedelec at Rest Showing Negligible Sensor Noise from IMU	42
4.4.	BNO055-USB-Stick: Oscillating Yaw Rates when Coasting for around 20 Seconds	44
4.5.	BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence	45
4.6.	BNO055-USB-Stick: FFT of Rider Torque Compared to Yaw Rates at Constant Cadence	45
4.7.	<i>Navilock NL-8012U</i> : Inaccurate Positioning	47
4.8.	Converging Positions	48
4.9.	Filtered Positions Oscillating around Stationary Pedelec	49
4.10.	BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence	49
4.11.	Ambiguity in Metric	50
4.12.	BNO055-USB-Stick: Oscillations in Yaw Rates at Constant Cadence	52
4.13.	const_a_yawr: Trajectories and Yaw Rates During a Curve	53
4.14.	Comparison of Trajectory Lengths for $t_{pred} = 2$ s and $t_{pred} = 4$ s	54

4.15. Two Succeeding Trajectory Predictions Jumping in Length	55
4.16. Overview of the Entire Software Framework	59
4.17. Overview of the Software Architecture in ROS	61
4.18. Angular Velocities for Dynamic Left Turn Transformed to {Street} Frame	64
4.19. Roll Angle Offset after Turn	65
4.20. Collision Test with Virtual Car	66
4.21. Reference Coordinate Frames With New Sensors	68
4.22. Roll Angle Offset after Turn	69
4.23. Coordinate Frames with Bike Roll (3D view)	71
4.24. Ignoring Roll Angle Correction leading to FN	72
4.25. Comparison of straight and curved segments for exemplary urban riding	75
4.26. Comparison of heading difference between various riding sections in an urban environment	76
4.27. Linear extrapolation of the speed signal	79
4.28. Overview of Two Routes through Malmsheim	79
4.29. Physically Illogical Trajectories	80
4.30. Pedelec's response compared to brake pressure	81
4.31. Steer angles and yaw rates for a curve	83
4.32. Running Mean-Filtered Lateral Acceleration	84
4.33. Lateral Signals for Left Turn (highlighted curve entry)	85
4.34. Bicycle Simplified as Inverted Pendulum	86
4.35. Validating Yaw Rate as a Function of Roll Angle	88
4.36. Trapezoidal Extrapolation Logic	90
4.37. Adaption of Overestimated $a_{y_{\max}}$ and t_{const} Parameters Based on Roll Rate	91
4.38. Effect of Useful Online Parameter Adaption on Predicted Trajectories	92
4.39. Using Geometric Centers Can Solve Certain Crash Configurations	94
4.40. Positions at the Geometric Centers Still Lead to Different Distances	94
4.41. Rectangular Convex Hulls around Road Users	95
4.42. Non-Colliding Circles	98
4.43. Finding Stopping Location for Pedelec	100
4.44. Comparison of Pre-Warning Trajectories for $t_{\text{pred}} = 4\text{s}$	104
5.1. <i>SimpleRTK2B-F9R Noard</i> : Accurate Positioning positioning	109
5.2. Overview of Two Routes through Malmsheim	111
5.3. Maximum Roll Angles of All Curves in Malmsheim 1 Trip	117
5.4. Speed Ripple Linearly Extrapolated with Different t_{pred} Parameters	118
5.5. Trajectory Comparison When Replacing Yaw Rate with Roll Angle for $t_{\text{pred}} = 2\text{s}$	122
5.6. Maximum Roll Angles of All Curves in Malmsheim 1 Trip	123
5.7. Statistics of Curves Above $ \phi_{\min} = 14.0\text{ deg}$ for 372 s of Malmsheim 1 Trip	124
5.8. Comparison of Positional Errors for $t_{\text{pred}} = 1\text{s}$	131
5.9. Comparison of Positional Errors for $t_{\text{pred}} = 2\text{s}$	131
5.10. Comparison of Positional Errors for $t_{\text{pred}} = 4\text{s}$	131
5.11. Comparison of Length and Angle Errors for $t_{\text{pred}} = 1\text{s}$	132

List of Figures

5.12. Comparison of Length and Angle Errors for $t_{\text{pred}} = 2 \text{ s}$	132
5.13. Comparison of Length and Angle Errors for $t_{\text{pred}} = 4 \text{ s}$	132
5.14. Trajectories of <code>const_a_const_yaw</code> for $t_{\text{pred}} = 2 \text{ s}$	134
5.15. Trajectories of <code>extrapol_speed_const_roll</code> for $t_{\text{pred}} = 2 \text{ s}$	135
5.16. Trajectories of <code>extrapol_speed_extrapol_roll</code> for $t_{\text{pred}} = 2 \text{ s}$	136
5.18. Signals for 180 Degree Curve of Large Radius at High Speed	136
5.17. Trajectories of <code>extrapol_speed_trapezoid_roll</code> for $t_{\text{pred}} = 2 \text{ s}$	137
5.19. Collision Detection Comparison for Car and Pedelec Waiting on Red Light Next to Each Other	140
5.20. Collision Detection Comparison for Car Driving in the Opposite Lane and Pedelec Riding on Sidewalk	142
5.21. Car Clipping the Rear Wheel of a Pedelec in a 90 Degree Junction	143
5.22. Collision Detection Comparison for Car Clipping the Rear Wheel of a Pedelec in a 90 degree Junction	144
5.23. Trajectory Comparison for Parallel Driving Scenario with Car Turning Into Pedelec	145
5.24. Alarm Time Comparison for Parallel Driving Scenario with Car Turning Into Pedelec	146
5.25. Comparison of Collision Warning Systems for Car Yielding to Pedelec	147
5.26. Comparison of Collision Warning Systems for Car Taking Pedelec's Right-of-Way	148
5.27. Pre-Warning Functionality of New Collision Warning System	149
5.28. Generated Roll Angle Profile Closely Matching Actual Roll Angle Signal	149
5.29. Potential of Trajectories and Prediction Errors of <code>extrapol_speed_trapezoid_- roll</code> Model	151
5.30. Trajectory and Prediction Errors of Baseline for Scenario of Figure 5.29	152
A.1. Initial Visualizations of the Signals	164
A.2. Adapted Visualizations of the Signals	165
A.3. SimpleRTK2B: Oscillating Yaw Rates When Coasting for Around 4 Seconds . .	166
A.4. Ellipse-N: Oscillating Yaw Rates when Coasting for Around 50 Seconds . . .	167
A.5. Collision Detection Comparison for a Stationary Car Passed by a Pedelec on the Side Walk	168
A.6. Comparison of Collision Warning Systems for Parallel Driving Scenario with Car Turning Into Pedelec	169

List of Tables

1. Key Points of Utilized Notation	vii
4.1. Performance Rating of ZED-F9R	69
4.2. Comparison of Computation Times for Single Intersection	98
5.1. Impact of stationary phase at beginning	107
5.2. Impact of stationary phase at end	107
5.3. Comparison of Riding Situations for Malsmheim Route 1 and 2	111
5.4. Trajectory Prediction Accuracies for Different Sections of Malmsheim 1 Trip with const_a_yawr Model and $t_{\text{pred}} = 2 \text{ s}$	112
5.5. Trajectory Prediction Accuracies for Different Sections of Malmsheim 2 Trip with const_a_yawr Model and $t_{\text{pred}} = 2 \text{ s}$	112
5.6. Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip	114
5.7. Trajectory Prediction Accuracies for Different Speed Ranges of Malmsheim 1 Trip with HMI WSS, const_speed_const_yawr Model and $t_{\text{pred}} = 2 \text{ s}$	114
5.8. Trajectory Prediction Accuracies for Different Speed Ranges of Malmsheim 1 Trip with ABS WSS, const_speed_const_yawr Model and $t_{\text{pred}} = 2 \text{ s}$	115
5.9. Trajectory Prediction Errors when Extrapolating Speed with a Limit of $v_{\min} = 0 \frac{\text{m}}{\text{s}}$ and $v_{\max} = 30 \frac{\text{m}}{\text{s}}$ for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip .	115
5.10. Trajectory Prediction Errors for Speed Extrapolation Parameter Combinations with $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	116
5.11. Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{\text{pred}} = 1 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	118
5.12. Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	118
5.13. Trajectory Prediction Errors for Extreme Speed Extrapolation Parameter Combinations with $t_{\text{pred}} = 4 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	119
5.14. Trajectory Prediction Errors for Extreme Acceleration Extrapolation Parameter Combinations with $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	119
5.15. Trajectory Prediction Accuracies for $t_{\text{pred}} = 1 \text{ s}$ Averaged over 1724 s Malmsheim Trip	120
5.16. Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim Trip	120
5.17. Trajectory Prediction Accuracies for $t_{\text{pred}} = 4 \text{ s}$ Averaged over 1724 s Malmsheim Trip	120

List of Tables

5.18. Trajectory prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ averaged over 1724 s Malmsheim Trip	121
5.19. Average, Over- and Underestimated Parameters for extrapol_lim_speed_trap_roll Trajectory Prediction Model	124
5.20. Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 372 s of Malmsheim 1 Trip	125
5.21. Trajectory Prediction Accuracies for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip	126
5.22. Selected Parameters of the Trajectory Prediction Models for Comparisons	127
5.23. Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 1 \text{ s}$ averaged over 1724 s Malmsheim 1 Trip	128
5.24. Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 2 \text{ s}$ averaged over 1724 s Malmsheim 1 Trip	129
5.25. Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 4 \text{ s}$ averaged over 1724 s Malmsheim 1 Trip	129
5.26. Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 1 \text{ s}$	133
5.27. Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 2 \text{ s}$	133
5.28. Relative Improvements of the Combined Positional Accuracy and Precision for $t_{\text{pred}} = 4 \text{ s}$	133
A.1. Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 1 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip	163
A.2. Comparison of Trajectory Prediction Errors for $t_{\text{pred}} = 2 \text{ s}$ Averaged over 1724 s Malmsheim 1 Trip	163

Bibliography

- [1] *Formelschreibweise und Formelsatz: DIN 1338*. Beuth, 2011.
- [2] A. Thompson. *Guide for the use of the International System of Units (SI)*. National Institute of Standards and Technology, 2008.
- [3] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. 3rd ed. Pearson Education, Inc., 2008.
- [4] M. Kords. *E-Bikes - Absatz in Deutschland bis 2020*. Mar. 2021. URL: <https://de.statista.com/statistik/daten/studie/152721/umfrage/absatz-von-e-bikes-in-deutschland/#:~:text=Absatz%20von%20E-Bikes%20in%20Deutschland%20bis%202020%20Ver%C3%BCffentlicht,nahm%202020%20so%20stark%20zu%20wie%20nie%20zuvor>. (visited on 08/19/2021).
- [5] Z.-I.-V. (ZIV). *Zahlen-Daten-Fakten zum Fahrradmarkt in Deutschland 2020*. Mar. 2021. URL: https://www.ziv-zweirad.de/uploads/media/PM_2021_10_03_ZIV-Praesentation_10.03.2021_mit_Text.pdf (visited on 08/19/2021).
- [6] D. S. B. (DESTATIS). *Verkehrsunfälle - Fachserie 8 Reihe 7 - 2020*. July 2021. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Publikationen/Downloads-Verkehrsunfaelle/verkehrsunfaelle-jahr-2080700207004.pdf;jsessionid=F4627F3C215A8510C72653B82FEB93B4.live741?__blob=publicationFile (visited on 08/19/2021).
- [7] D. Spiegel. *Unfallstatistik: Viel weniger Verkehrstote – aber deutlicher Anstieg bei E-Bike-Fahrern*. Feb. 2021. URL: <https://www.spiegel.de/auto/deutschland-viel-weniger-verkehrstote-aber-deutlich-mehr-verunglueckte-e-bike-fahrer-a-5c11ff9f-0b9a-4433-aa36-ff2ebea9e75e> (visited on 08/19/2021).
- [8] D. Adminaité-Fodor and G. Jost. *How safe is walking and cycling in Europe? (PIN Flash 38)*. Jan. 2020. URL: https://etsc.eu/wp-content/uploads/PIN-Flash-38_FINAL.pdf (visited on 08/20/2020).
- [9] E. R. S. Observatory. *Traffic Safety Basic Facts 2018 – Cyclists*. May 2018. URL: https://ec.europa.eu/transport/road_safety/sites/default/files/pdf/statistics/dacota/bfs20xx_cyclists.pdf (visited on 08/19/2021).
- [10] M. Pschenitzka. *Kreuzungsunfälle*. München: Allgemeiner Deutscher Automobil-Club e.V., 2017.
- [11] ASSESSMENT PROTOCOL – PEDESTRIAN PROTECTION. EUROPEAN NEW CAR ASSESSMENT PROGRAMME (Euro NCAP), Sept. 2018.

Bibliography

- [12] D. S. B. (DESTATIS). *Verkehrsunfälle - Fachserie 8 Reihe 7 - 2015*. July 2016. URL: https://www.statistischebibliothek.de/mir/servlets/MCRFileNodeServlet/DEHeft_derivate_00030914/2080700157004_akt12102016.pdf (visited on 08/29/2021).
- [13] P.-Å. Fröberg. *Volvo Car Group introduces world-first Cyclist Detection with full auto brake*. Mar. 2013. URL: <https://www.media.volvocars.com/global/en-gb/media/pressreleases/48277#:~:text=Volvo%20Car%20Group%20has%20rolled%20out%20another%20Volvo,of%20the%20present%20detection%20and%20auto%20brake%20technology.> (visited on 08/20/2021).
- [14] *Collision warning system - detection of cyclists: Collision warning system: Driver support: V40 2019: Volvo Support*. Jan. 2020. URL: <https://www.volvocars.com/za/support/manuals/v40/2018w17/driver-support/collision-warning-system/collision-warning-system---detection-of-cyclists> (visited on 08/20/2021).
- [15] S. Fillenberg. *Digital Guardian Angel for Cyclists*. June 2021. URL: <https://www.continental.com/en/press/press-releases/2021-06-02-collision-warning/> (visited on 08/20/2021).
- [16] F. Leibiger. *Digital guardian angel for cyclists*. June 2021. URL: <https://www.telekom.com/en/media/media-information/archive/digital-guardian-angel-for-cyclists-628702> (visited on 08/20/2021).
- [17] B.-M. Piscol. *Entwicklung einer Fahrerassistenzfunktion für Zweiräder zur Früherkennung nicht sichtbarer Gefahrensituationen mittels 5G-Mobilfunktechnologie*. 2020.
- [18] *E-Fire Tour R2600i ABS*. URL: <https://www.centurion.de/de-de/bike/246/e-fire-tour-r2600i-abs> (visited on 08/20/2021).
- [19] *BNO055 USB-STICK*. URL: <https://www.digikey.de/product-detail/en/bosch-sensortec/BN0055-USB-STICK/828-1045-ND/6136288> (visited on 08/20/2021).
- [20] *Intel® NUC-Kit NUC8i7BEH*. URL: <https://www.intel.de/content/www/de/de/products/sku/126140/intel-nuc-kit-nuc8i7beh/specifications.html> (visited on 08/20/2021).
- [21] *Ixxat® USB-to-CAN V2 Active USB interface*. URL: <https://www.ixxat.com/products/products-industrial/can-interfaces/usb-can-interfaces/usb-to-can-v2-professional?ordercode=1.01.0281.12001> (visited on 08/20/2021).
- [22] *NL-8012U USB 2.0 Multi GNSS Empfänger u-blox 8 4,5 m*. URL: <https://www.navilock.de/produkt/62524/merkmale.html> (visited on 08/20/2021).
- [23] *Sinuslive GL-205 RCA EMI filter*. URL: <https://www.conrad.com/p/sinuslive-gl-205-rca-emi-filter-379234> (visited on 08/20/2021).
- [24] G. Rizzi and M. L. Ruggiero. *Relativity in rotating frames: relativistic physics in rotating reference frames*. Kluwer Academic Publishers, 2004.

Bibliography

- [25] B. D. Tapley, B. E. Schutz, and G. H. Born. "Chapter 2 - The Orbit Problem". In: *Statistical Orbit Determination*. Ed. by B. D. Tapley, B. E. Schutz, and G. H. Born. Burlington: Academic Press, 2004, pp. 17–91. ISBN: 978-0-12-683630-1. DOI: <https://doi.org/10.1016/B978-012683630-1/50021-7>. URL: <https://www.sciencedirect.com/science/article/pii/B9780126836301500217>.
- [26] M. Kok, J. D. Hol, and T. B. Schön. "Using Inertial Sensors for Position and Orientation Estimation". In: *Foundations and Trends® in Signal Processing* 11.1-2 (2017), pp. 1–153. ISSN: 1932-8354. DOI: 10.1561/2000000094. URL: <http://dx.doi.org/10.1561/2000000094>.
- [27] O. Montenbruck, E. Gill, and T. Terzibaschian. *Note on the BIRD ACS Reference Frames*. Tech. rep. LIDO-Berichtsjahr=2000, 2000. URL: <https://elib.dlr.de/11316/>.
- [28] J. R. Lynch. *Earth coordinates*. Tech. rep. LIDO-Berichtsjahr=2000, 2006. URL: <http://clynchg3c.com/Technote/geodesy/coorddef.pdf> (visited on 09/30/2021).
- [29] J. Sanz Subirana, J. M. Juan Zornoza, and M. Hernandez-Pajares. *Reference Frames in GNSS*. Feb. 2021. URL: https://gssc.esa.int/navipedia/index.php/Reference_Frames_in_GNSS (visited on 09/30/2021).
- [30] F. J. Lohmar. "World geodetic system 1984 — geodetic reference system of GPS orbits". In: *GPS-Techniques Applied to Geodesy and Surveying*. Ed. by E. Groten and R. Strauß. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 476–486. ISBN: 978-3-540-45962-0.
- [31] S. Malys, J. H. Seago, N. K. Pavlis, P. K. Seidelmann, and G. H. Kaplan. "Why the Greenwich meridian moved". In: *Journal of Geodesy* 89.12 (Dec. 2015), pp. 1263–1272. DOI: 10.1007/s00190-015-0844-y.
- [32] *File:Geocentric coords 03.svg*. URL: https://commons.wikimedia.org/wiki/File:Geocentric_coords_03.svg (visited on 09/30/2021).
- [33] *Projections/Spatial reference systems*. URL: <https://osmdata.openstreetmap.de/info/projections.html> (visited on 10/30/2021).
- [34] *EPSG:3857*. URL: <https://wiki.openstreetmap.org/wiki/EPSG:3857> (visited on 10/30/2021).
- [35] ISO Central Secretary. *Straßenfahrzeuge – Fahrzeugdynamik und Fahrverhalten – Begriffe (ISO 8855:2011)*. de. Standard DIN ISO 8855:2011. Berlin, Germany: Deutsches Institut für Normung e. V., International Organization for Standardization, 2013. URL: <https://www.iso.org/standard/51180.html>.
- [36] *BNO055 Intelligent 9-axis absolute orientation sensor*. BST-BNO055-DS000-17. Rev. 1.7. Bosch Sensortec GmbH. 2020. URL: <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf> (visited on 09/13/2021).
- [37] D. Titterton, J. L. Weston, and J. Weston. *Strapdown inertial navigation technology*. 2nd ed. Vol. 17. Institution of Engineering and Technology, 2005.

Bibliography

- [38] *Inertial Sensor Modules*. 2020. URL: <https://www.xsens.com/inertial-sensor-modules> (visited on 09/30/2021).
- [39] *What Difference between an IMU, an AHRS, and an Inertial Navigation System?* Nov. 2020. URL: <https://www.sbg-systems.com/attitude-heading-reference-systems-ahrs/> (visited on 09/30/2021).
- [40] A. Rietdorf, C. Daub, and P. Loef. *Precise Positioning in Real-Time using Navigation Satellites and Telecommunication*.
- [41] *Chapter 4: Source of Inaccuracy: The Cures*. URL: <https://northsurveying.com/index.php/soporte/gnss-and-geodesy-concepts#chapter-4-sources-of-inaccuracy-the-cures> (visited on 09/30/2021).
- [42] *u-blox F9 high precision sensor fusion GNSS receiver*. UBX-19054459. Rev. 05. u-blox AG. 2021. URL: https://www.u-blox.com/sites/default/files/ZED-F9R-01B%5C_Datasheet%5C_UBX-19054459.pdf (visited on 09/14/2021).
- [43] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. European Telecommunications Standards Institute, 2014. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf.
- [44] *Vulnerable Road User Safety Message Minimum Performance Requirements*. SAE International, 2017. DOI: https://doi.org/10.4271/J2945/9_201703. URL: https://doi.org/10.4271/J2945/9_201703.
- [45] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. 7th ed. Cambridge Univ. Pr., 2009.
- [46] J. Huynh. *Separating Axis Theorem*. Sept. 2009. URL: <http://www.jkh.me/tutorials.html> (visited on 08/30/2021).
- [47] S. Crozet. *Continuous collision detection§*. URL: [https://nphysics.org/continuous_collision_detection/#:~:text=Continuous%20collision%20detection%20\(CC%20is%20a%20technique%20that,of%20collision%20detection,%20causing%20visual%20and%20logical%20artifacts](https://nphysics.org/continuous_collision_detection/#:~:text=Continuous%20collision%20detection%20(CC%20is%20a%20technique%20that,of%20collision%20detection,%20causing%20visual%20and%20logical%20artifacts). (visited on 09/02/2021).
- [48] H. Eichner. *Collision Detection*. 2014. URL: <http://myselfph.de/gamePhysics/collisionDetection.html#:~:text=Collision%20detection%20usually%20consists%20of%20several%20phases,%20at,-%20that%20is,%20pairs%20of%20bodies%20that%20mightcollide>. (visited on 08/31/2021).
- [49] C. Ericson. “Collisions using separating-axis tests”. In: *Exocimes 2012*. San Francisco, CA, Mar. 2007. URL: http://realtimecollisiondetection.net/pubs/GDC07_Ericson_Physics_Tutorial_SAT.ppt.
- [50] C. Ericson. *Real-Time Collision Detection*. Elsevier, 2005.

Bibliography

- [51] S. Gottschalk, M. C. Lin, and D. Manocha. "OBBTree: A Hierarchical Structure for Rapid Interference Detection". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, pp. 171–180. ISBN: 0897917464. DOI: 10.1145/237170.237244. URL: <https://doi.org/10.1145/237170.237244>.
- [52] E.-H. Erdei, J. Steinmann, and C. Hagemeister. *Comparing perception of signals in different modalities during the cycling task: A field study*. Vol. 73. 2020, pp. 259–270. DOI: <https://doi.org/10.1016/j.trf.2020.06.011>. URL: <http://www.sciencedirect.com/science/article/pii/S1369847820304484>.
- [53] D. Zeradjanin. *Entwicklung und prototypischer Aufbau von Fahrerwarnsystemen für Pedelecs*. 2020.
- [54] V. R. Kailasam. *Multivariate Forecasting of E-Bike Sensor Data using Machine Learning*. 2020.
- [55] P. Muresan. *Entwicklung eines Kollisionswarnsystems basierend auf Fahrzeug zu Fahrzeug Kommunikation und Optimierung der Performance durch Machine Learning*. 2021.
- [56] J. Van Brummelen, B. Emran, K. Yesilcimen, and H. Najjaran. "Reliable and low-cost cyclist collision warning system for safer commute on urban roads". In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2016, pp. 003731–003735. DOI: 10.1109/SMC.2016.7844814.
- [57] K. Li, X. Yu, and J. Zhou. *Rear End Collision Warning System for Bikes*. 2017. URL: <https://courses.physics.illinois.edu/ece445/getfile.asp?id=11922>.
- [58] Z. X. Gooi. *Design and Development of Collision Warning Algorithm for Motorcycles and Bicycles by Using Ultrasonic Sensor*. 2019. URL: <https://eprints.tarc.edu.my/id/eprint/13104>.
- [59] K. Iwamoto and S. Ohtake. "A Warning System with Multiple Sensors for Avoiding Collision of Bicycles". In: *2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*. 2020, pp. 1–2. DOI: 10.1109/ICCE-Taiwan49838.2020.9258109.
- [60] W. Jeon and R. Rajamani. "A novel collision avoidance system for bicycles". In: *2016 American Control Conference (ACC)*. 2016, pp. 3474–3479. DOI: 10.1109/ACC.2016.7525451.
- [61] W. Jeon and R. Rajamani. "Two-dimensional active sensing system for bicyclist-motorist crash prediction". In: *2017 American Control Conference (ACC)* (2017), pp. 2315–2320.
- [62] *A Novel Collision Avoidance System for a Bicycle*. Center for Transportation Studies, University of Minnesota, 2018. URL: <https://hdl.handle.net/11299/195859>.
- [63] W. Jeon, Z. Xie, C. Craig, J. Achtemeier, L. Alexander, N. Morris, M. Donath, and R. Rajamani. "A Smart Bicycle That Protects Itself: Active Sensing and Estimation for Car-Bicycle Collision Prevention". In: *IEEE Control Systems Magazine* 41.3 (2021), pp. 28–57. DOI: 10.1109/MCS.2021.3062955.

Bibliography

- [64] P. Riekert and T. E. Schunck. "Zur Fahrmechanik des gummibereiften Kraftfahrzeugs". In: *Ingenieur-Archiv* 11.3 (June 1940), pp. 210–224. ISSN: 1432-0681. doi: 10.1007/BF02086921. URL: <https://doi.org/10.1007/BF02086921>.
- [65] N. H. Getz and J. E. Marsden. "Dynamic Inversion of Nonlinear Maps with Applications to Nonlinear Control and Robotics". PhD thesis. 1995.
- [66] F. Wirth, T. Wen, C. F. Lopez, and C. Stiller. "Model-Based Prediction of Two-Wheelers". In: *2020 IEEE Intelligent Vehicles Symposium (IV)* (2020), pp. 1669–1674.
- [67] Y. Zhang. "Modeling and control of single-track vehicles: A human-machine-environment interactions perspective". PhD thesis. 2014.
- [68] F. J. W. Whipple. "The stability of the motion of a bicycle". In: *Quart. J. Pure Appl. Math.* 30 (1899), pp. 312–348.
- [69] E. Carvallo. "Théorie du mouvement du monocycle et de la bicyclette". In: *Prix Fourneyron* (1899), pp. 312–348.
- [70] J. Meijaard, J. Papadopoulos, A. Ruina, and A. Schwab. "Linearized dynamics equations for the balance and steer of a bicycle: A benchmark and review". In: *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences* 463 (Aug. 2007), pp. 1955–1982. doi: 10.1098/rspa.2007.1857.
- [71] V. Cossalter. *Motorcycle dynamics*. Lulu, 2010.
- [72] J. Kooijman, A. Schwab, and J. Meijaard. "Experimental validation of a model of an uncontrolled bicycle". In: *Multibody System Dynamics* 19 (Feb. 2008), pp. 115–132. doi: 10.1007/s11044-007-9050-x.
- [73] J. Kooijman, J. Meijaard, J. Papadopoulos, A. Ruina, and A. Schwab. "A Bicycle Can Be Self-Stable Without Gyroscopic or Caster Effects". In: *Science (New York, N.Y.)* 332 (Apr. 2011), pp. 339–42. doi: 10.1126/science.1201959.
- [74] J. Yuan, H. Chen, F. Sun, and Y. Huang. "Trajectory planning and tracking control for autonomous bicycle robot". In: *Nonlinear Dynamics* 78.1 (Oct. 2014), pp. 421–431. ISSN: 1573-269X. doi: 10.1007/s11071-014-1449-3. URL: <https://doi.org/10.1007/s11071-014-1449-3>.
- [75] 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: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, pp. 833–838. doi: 10.1109/IVS.2016.7535484.
- [76] X. Ma and D. Luo. "Modeling cyclist acceleration process for bicycle traffic simulation using naturalistic data". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 40 (2016), pp. 130–144. ISSN: 1369-8478. doi: <https://doi.org/10.1016/j.trf.2016.04.009>. URL: <https://www.sciencedirect.com/science/article/pii/S1369847816300195>.
- [77] D. Luo and X. Ma. "Modeling of Cyclist Acceleration Behavior Using Naturalistic GPS Data". In: Jan. 2016. doi: 10.13140/RG.2.1.1196.7120.

Bibliography

- [78] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila. "Using road topology to improve cyclist path prediction". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 289–296. doi: 10.1109/IVS.2017.7995734.
- [79] K. Saleh, M. Hossny, and S. Nahavandi. "Cyclist Trajectory Prediction Using Bidirectional Recurrent Neural Networks". In: *AI 2018: Advances in Artificial Intelligence*. Ed. by T. Mitrovic, B. Xue, and X. Li. Cham: Springer International Publishing, 2018, pp. 284–295. ISBN: 978-3-030-03991-2.
- [80] H. Gao, H. Su, Y. Cai, R. Wu, Z. Hao, Y. Xu, W. Wu, J. Wang, Z. Li, and Z. Kan. "Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections". In: *Science China Information Sciences* 64.7 (May 2021), p. 172207. ISSN: 1869-1919. doi: 10.1007/s11432-020-3071-8. URL: <https://doi.org/10.1007/s11432-020-3071-8>.
- [81] Y. Ding, X. Zhou, H. Bao, Y. Li, C. Hamann, S. Spears, and Z. Yuan. "Cycling-Net: A Deep Learning Approach to Predicting Cyclist Behaviors from Geo-Referenced Egocentric Video Data". In: *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '20. Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 337–346. ISBN: 9781450380195. doi: 10.1145/3397536.3422258. URL: <https://doi.org/10.1145/3397536.3422258>.
- [82] A. Gavriilidou, W. Daamen, Y. Yuan, and S. Hoogendoorn. "Modelling cyclist queue formation using a two-layer framework for operational cycling behaviour". In: *Transportation Research Part C: Emerging Technologies* 105 (2019), pp. 468–484. ISSN: 0968-090X. doi: <https://doi.org/10.1016/j.trc.2019.06.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X18318576>.
- [83] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2018. URL: <https://bokeh.pydata.org/en/latest/>.
- [84] M. Dozza and A. Fernandez. "Understanding Bicycle Dynamics and Cyclist Behavior From Naturalistic Field Data (November 2012)". In: *IEEE Transactions on Intelligent Transportation Systems* 15.1 (2014), pp. 376–384. doi: 10.1109/TITS.2013.2279687.
- [85] S. W. Smith. *The Scientist and Engineers Guide to Digital Signal Processing*. California Technical Pub., 1997.
- [86] *u-blox 8 GNSS modules*. UBX-16000093. Rev. 06. u-blox AG. 2019. URL: https://www.u-blox.com/sites/default/files/MAX-8_DataSheet_%5C%28UBX-16000093%5C%29.pdf (visited on 09/09/2021).
- [87] B. Global. *BNO055*. Oct. 2021. URL: <https://www.bosch-sensortec.com/products/smart-sensors/bno055/#applications>.
- [88] *4G NTRIP Client hookup guide for simpleRTK2B GNSS board*. Sept. 2021. URL: <https://www.ardusimple.com/4g-ntrip-client-hookup-guide/> (visited on 09/14/2021).
- [89] *RTK INS simpleRTK2B-F9R hookup guide*. Mar. 2021. URL: <https://www.ardusimple.com/rtk-ins-simplertk2b-f9r-hookup-guide/> (visited on 09/14/2021).

Bibliography

- [90] T. Oliphant. *NumPy: A guide to NumPy*. USA: Trelgol Publishing. 2006–. URL: <http://www.numpy.org/> (visited on 08/26/2021).
- [91] *Welcome to SymPy's documentation!* Apr. 2021. URL: <https://docs.sympy.org/latest/index.html> (visited on 08/30/2021).
- [92] *Polygons*. Apr. 2021. URL: <https://docs.sympy.org/latest/modules/geometry/polygons.html> (visited on 08/30/2021).
- [93] J. A. Aldea and J. Eager. *Separating-Axis-Theorem*. Oct. 2017. URL: https://github.com/JuantAldea/Separating-Axis-Theorem/blob/master/python/separation_axis_theorem.py (visited on 08/23/2021).
- [94] N. R. Rose. *Braking Capabilities of Motorcyclists - A Literature Review*. 7AD. URL: <https://www.nathanarose.com/blog/braking-capabilities-of-motorcyclists-a-literature-review> (visited on 09/04/2021).
- [95] M. Paine. In: (). URL: https://www.researchgate.net/publication/352555720_ANALYSIS_OF_RELATIVE_SAFETY_PERFORMANCE_OF_BICYCLES_AND_SCOOTERS (visited on 09/04/2021).
- [96] N. Famiglietti, B. Nguyen, E. Fatzinger, and J. Landerville. “Bicycle Braking Performance Testing and Analysis”. In: *SAE Technical Paper Series* (Apr. 2020). doi: 10.4271/2020-01-0876. URL: <https://www.sae.org/publications/technical-papers/content/2020-01-0876/> (visited on 09/04/2021).
- [97] K. Buchholtz and T. Burgess. “An evaluation of bicycle-specific agility and reaction times in mountain bikers and road cyclists”. In: *South African Journal of Sports Medicine* 32 (Sept. 2020), pp. 1–5. doi: 10.17159/2078-516X/2020/v32i1a8576.
- [98] F. Aragall. *Europäisches Konzept für Zugänglichkeit*. Nov. 2003. URL: https://www.fdst.de/w/files/pdf/eca_deutsch_internet.pdf (visited on 10/01/2021).
- [99] *Wie breit müssen Gehwege sein?* Oct. 2020. URL: <https://www.bundesregierung.de/breg-de/aktuelles/faq-fusswegeplanung-1800308> (visited on 10/01/2021).
- [100] BayObLG. *Court decision NZV 1990, 402*. Feb. 22, 1990. URL: <https://dejure.org/1990,5418> (visited on 10/01/2021).
- [101] V. Berlin. *Court decision NZV 1998, 224*. Nov. 18, 1997. URL: <https://dejure.org/1997,8692> (visited on 10/01/2021).