AGH

AGH UNIVERSITY OF KRAKOW

FIELD OF SCIENCE: ENGINEERING AND TECHNOLOGY

SCIENTIFIC DISCIPLINE: AUTOMATION, ELECTRONICS, ELECTRICAL ENGINEERING AND SPACE TECHNOLOGIES

DOCTORAL DISSERTATION

Real-Time Generation of Safe Trajectories for Autonomous Vehicles in Dynamic Environments

Author: *Wojciech Turlej*

First supervisor: *prof. dr hab. inż. Wojciech Mitkowski*

Auxiliary supervisor: *dr inż. Krzysztof Kogut*

Completed at: AGH University of Krakow, Faculty of Electrical Engineering, Automatics, Computer Science, and Biomedical Engineering

Kraków, 2023

AGH

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH

AUTOMATYKA, ELEKTRONIKA, ELEKTROTECHNIKA I TECHNOLOGIE KOSMICZNE

ROZPRAWA DOKTORSKA

Generacja bezpiecznych trajektorii w czasie rzeczywistym dla pojazdów poruszających się w dynamicznym środowisku

Autor:	*Wojciech Turlej*
Promotor rozprawy:	*prof. dr hab. inż. Wojciech Mitkowski*
Promotor pomocniczy:	*dr inż. Krzysztof Kogut*
Praca wykonana:	Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Kraków, 2023

Abstract

Driving in a traffic environment is a notoriously difficult task, both for human drivers and autonomous driving algorithms. Differently than in the case of static environments, often considered in the design of planning algorithms for robotic purposes, the generation of a road vehicle's trajectory requires a deep understanding of the dynamic environment of public roads. The presence of other road users, the behavior of which is often unpredictable and depends on complex interactions between traffic participants, creates a need for new motion planning algorithms that would result in a safe yet efficient motion of the controlled vehicle.

In this thesis, several challenges related to the design and evaluation of motion planning algorithms for autonomous driving purposes are addressed.

To solve the problem of planning a safe trajectory for the vehicle in uncertain situations, a novel Multiple Hypothesis Planning method is introduced. The proposed method takes into account several hypotheses regarding the behavior of other road users to plan an efficient motion of the controlled vehicle, which will remain collision-free in all predicted plausible scenarios. Additionally, the proposed approach can be used to execute a fail-safe planning task, in which reasonably foreseeable worst-case hypotheses regarding the behavior of other traffic participants are taken into account to ensure a safe motion.

As the proposed method is intended mainly for short-term planning purposes in difficult situations, it can be used in conjunction with methods based on machine learning techniques, such as Reinforcement Learning, that are capable of long-term strategic planning. However, since such methods are typically trained in a simulation environment, they are often susceptible to perception errors that are often present in real systems. To address this problem, a set of low-fidelity sensor models is introduced in this thesis for training and evaluation purposes. The models simulate common error modalities of dynamic objects perception systems and lane marker detection systems. A set of driving policies has been trained in a reinforcement learning setup to closely assess how the use of proposed models affects the robustness of neural networks that perform vehicle control tasks.

Finally, to enable thorough testing of motion planning systems needed to ensure their safety, a novel automated scenario generation method is introduced. The method is capable of producing adversarial test scenarios that uncover potential weaknesses and issues in the evaluated vehicle motion planning algorithms. Unlike the existing adversarial testing methods, the proposed approach generates not only the trajectories of the surrounding vehicles, but also the corresponding perception error patterns. The effectiveness of the proposed method is demonstrated in the task of generating adversarial scenarios for machine learning-based driving policies, in which the method has been used to produce a varied set of safety-critical test scenarios.

Streszczenie

Prowadzenie samochodu stanowi złożone zadanie, zarówno dla kierowców, jak i dla algorytmów jazdy autonomicznej. W odróżnieniu od przypadku planowania ruchu w środowiskach statycznych, często uwzględnianych robotyce, generacja trajektorii pojazdu wymaga dogłębnego zrozumienia złożonego środowiska dynamicznego, jakim są publiczne drogi. Obecność innych pojazdów oraz skomplikowanych interakcji między nimi wymaga opracowania nowych algorytmów planowania, które zagwarantowałyby efektywny i bezpieczny ruch kontrolowanego pojazdu. W niniejszej pracy zaadresowano szereg problemów związanych z projektowaniem oraz ewaluacją takich algorytmów.

Celem rozwiązania problemu planowania ruchu w niepewnych sytuacjach, zaproponowano nową metodę planowania wielohipotezowego. Zaproponowana metoda uwzględnia kilka hipotez dotyczących przyszłego zachowania innych użytkowników drogi aby zagwarantować bezkolizyjność i efektywność ruchu kontrolowanego pojazdu we wszystkich prawdopodobnych scenariuszach. Proponowane podejście może zostać również użyte w zadaniu planowania z manewrem awaryjnym, w którym najgorsze prawdopodobne rozwinięcia danej sytuacji drogowej są uwzględniane dla zagwarantowania istnienia bezkolizyjnego manewru bezpieczeństwa.

Zaproponowana metoda planowania wielohipotezowego skupia się na planowaniu krótkich manewrów w trudnych sytuacjach i może być użyta w połączeniu z metodami planowania opartymi na technikach uczenia maszynowego, które umożliwiają strategiczne planowanie w odległym horyzoncie czasowym. Jednakże, ponieważ wykorzystywane do tego celu metody, takie jak uczenie ze wzmocnieniem, zazwyczaj trenowane są jedynie w środowisku symulacyjnym, często są one podatne na błędy percepcji występujące w rzeczywistych systemach. Aby rozwiązać ten problem, w niniejszej pracy zaproponowano szereg wysokopoziomowych modeli sensorów dla celów ewaluacji i trenowania takich algorytmów. Zaproponowane modele symulują rodzaje błędów powszechnie występujące w systemach pecepcji obiektów dynamicznych oraz wykrywania pasów ruchu. Wpływ zaproponowanych modeli na trenowane algorytmy zbadano na przykładzie trenowania sieci neuronowych sterujących ruchem pojazdu, których odporność na błędy percepcji została zbadana w szeregu eksperymentów symulacyjnych.

Aby umożliwić dokładne testowanie systemów planowania ruchu pojazdu, w szczególności tych opartych o techniki uczenia maszynowego, zaproponowano nową metodę automatycznej generacji scenariuszy testowych. Zaproponowana metoda generuje scenariusze antagonistyczne, pozwalając na aktywną eksplorację potencjalnych słabości i błędów w testowanych algorytmach. W przeciwieństwie do istniejących metod generacji scenariuszy antagonistycznych, metoda generuje nie tylko trajektorie użytkowników ruchu, ale także odpowiadające im błędy percepcji stanowiące wyzwanie dla testowanych systemów. Skuteczność zaproponowanej metody została zademonstrowana w zadaniu generowania scenariuszy testowych dla systemów planowania ruchu opartych o uczenie ze wzmocnieniem, pozwalając na wygenerowanie zróżnicowanego zestawu scenariuszy krytycznych z punktu widzenia bezpieczeństwa testowanego systemu.

Acknowledgements

First and foremost I would like to express my sincere gratitude to my PhD advisor prof. dr hab. inż. Wojciech Mitkowski and my auxiliary supervisor dr inż. Krzysztof Kogut for their guidance and support throughout my research and preparation of this thesis.

I would also like to thank all my colleagues at Aptiv for countless discussions, research collaboration, and creating a great working atmosphere, which helped me immensely in the preparation of this thesis and all related research. In particular, I am deeply grateful to Mateusz Orłowski, Nikodem Pankiewicz, Tomasz Wrona, Paweł Kowalczyk, and Michał Sokół - being able to collaborate with you, learn from you, and rely on you as my friends has been an honor and joy.

My sincere gratitude goes to my parents Andrzej and Danuta, as well as my wonderful brothers Piotr and Marcin. I am deeply grateful for your faith, support, and encouragements.

Last but not least, I would like to thank my dear friends Zuzanna, Krzysztof, Joanna and Dariusz. Although the last four years were definitely challenging for me, you filled them with amazing moments and adventures, memories of which I will cherish for years to come. Without your support, none of this would have been possible. Above all, thank you Dagmara, my love, my friend and my soulmate, for your endless support, patience and unconditional love.

Contents

1. Introduction 1

1.1. Driving Automation 1

1.2. Safety 3

1.3. Motivation and Scope 4

1.4. Research Hypotheses 5

1.5. Thesis Outline and Contributions 6

2. Background 9

2.1. Advanced Driver Assistance Systems and Autonomous Driving Systems 9

2.2. Perception Systems Used in Automotive and Their Limitations 10

2.2.1. Sensors 11

2.2.2. Sensor Fusion 22

2.3. Autonomous Driving System Architecture 26

2.4. Reinforcement Learning 28

2.4.1. Proximal Policy Optimization 28

2.5. Safety 29

2.5.1. Hardware and Software Faults 29

2.5.2. Sensors Errors and Performance Limitations 30

2.5.3. Nominal Safety 31

3. Multiple Hypothesis Trajectory Planning 35

3.1. Introduction and Motivation 35

3.1.1. Motion Planning 36

3.1.2. Prediction Methods 37

3.1.3. Motivation and General Idea 40

3.1.4. Contributions 43

3.2. Problem Formulation 44

3.2.1. Assumptions 44

3.2.2. Static Environment 45

3.2.3. Dynamic Constraints 47
3.2.4. Control Trajectories 48
3.2.5. Vehicle Model 48
3.3. Constraints Generation 50
3.3.1. Worst-case Occupancy Set 50
3.3.2. Reasonably Foreseeable Occupancy Set 52
3.3.3. Multi-modal Prediction Occupancy Sets 58
3.4. Planning 58
3.4.1. Cost Terms 60
3.4.2. Constraints 62
3.5. Evaluation 63
3.5.1. Experimental Setup 64
3.5.2. Evaluated Scenarios 65
3.5.3. Computational Performance 72
3.6. Conclusions 73
3.6.1. Limitations and Further Work 73
4. Sensor Modeling 77
4.1. Introduction and Motivation 77
4.1.1. Existing Approaches 78
4.1.2. Motivation and General Idea 81
4.1.3. Contributions 82
4.2. Problem Formulation 83
4.3. Dynamic Objects Detection Model 84
4.3.1. Range Limitations and Occlusions 86
4.3.2. False Negative Detection Errors 86
4.3.3. False Positive Detection Errors 87
4.3.4. State Estimation Errors 88
4.3.5. Complete Model of the Dynamic Environment Perception 89
4.4. Lane Markers Detection Model 90
4.4.1. Range Limitations and Occlusions 92
4.4.2. False Negative Detection Errors 93
4.4.3. Geometry Estimation Errors 94
4.5. Application of Sensor Models in RL-based Driving Policy Training 95
4.5.1. Network Inputs 96

4.5.2. Network Architecture 97
4.5.3. Reward Components........ 98
4.5.4. Training Setup........ 99
4.6. Evaluation methodology 100
4.6.1. Baseline Models........ 101
4.6.2. Test Scenarios........ 103
4.7. Experimental results 106
4.7.1. Highway Driving Performance........ 106
4.7.2. Scripted Scenarios Performance........ 109
4.7.3. Computational Performance........ 110
4.8. Conclusions 111
5. Adversarial Trajectories Generation 115
5.1. Introduction and Motivation........ 115
5.1.1. Background........ 116
5.1.2. Existing Approaches........ 117
5.1.3. Motivation and General Idea........ 120
5.1.4. Contributions........ 121
5.2. Problem Formulation 122
5.2.1. Assumptions 123
5.2.2. Scenario Description........ 123
5.2.3. System Under Test 125
5.2.4. Traffic Participants Model........ 126
5.3. Optimization Problem 127
5.3.1. Cost Terms 128
5.3.2. Constraints 130
5.3.3. Iterative Generation of Multiple Scenarios........ 130
5.3.4. State Estimation Errors........ 131
5.4. Evaluation........ 132
5.4.1. System Under Test 132
5.4.2. Evaluation Examples........ 134
5.5. Conclusions 141
5.5.1. Limitations and Further Work........ 142
5.5.2. Further Applications........ 144
6. Conclusions........ 145

6.1. Summary and Contributions 145
6.1.1. Multiple Hypothesis Planning 145
6.1.2. Sensors Modeling 146
6.1.3. Adversarial Scenarios Generation 147
6.2. Perspectives and Further Work 148
6.2.1. Multiple Hypothesis Planning 148
6.2.2. Sensor Modeling 149
6.2.3. Adversarial Scenarios Generation 149
Bibliography 169
List of Figures 170
List of Tables 172

List of Symbols

Symbols - Chapter 2

Symbol	**Description**
$\mathcal{S}$	State space in RL context.
$\mathcal{A}$	Action space in RL context.
σ	Environment's state $\sigma \in \mathcal{S}$.
α	Agent's action $\alpha \in \mathcal{A}$.
$\pi_\theta(\alpha\|o)$	Probability of taking an action α given observation o and parameters vector θ under a stochastic policy $\pi_\theta : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$.
θ	Policy's parameters vector (e.g., neural network weights).
θ_k	Policy's parameters vector during the training iteration k.
$\mathcal{L}$	Clipped loss function.
$\pi_\theta(\alpha_t\|\sigma_t)$	Probability of action α_t in state σ_t according to a policy parameterized by a vector θ.
$\pi_\theta(\alpha_t\|\sigma_t)$	Probability of action α_t in state σ_t according to a policy parameterized by a vector θ.
$\hat{A}^{\pi_{\theta_k}}(\sigma, \alpha)$	Advantage estimate.
ϵ	Training hyperparameter.
r_t	Reward value at time. t
$V_\phi(\sigma)$	Value function estimate for a state σ in PPO.
ϕ	Critic network parameters in PPO.
λ	PPO training hyperparameter.
γ	PPO training hyperparameter.
$\mathcal{D}_k = \{\tau_i\}$	Set of trajectory segments collected during the training iteration k.
n_{episodes}	Number of trajectory segments used in each PPO training iteration.

Symbols - Chapter 3

Symbol	**Description**
t_{h}	Trajectories overlap duration.
$\mathcal{T}_{\mathrm{p}}$	Set of planned control trajectories.
n_{traj}	Number of planned control trajectories.
k_{par}	Number of parameters that describe a single trajectory.
$T_i(\mathbf{q}_i, t)$	i-th control trajectory.
$\mathbf{q}_i$	Parameters of an $i-th$ control trajectory.
t_{f}	Duration of planned trajectories.
H_i	Hypothesis regarding current and/or future states of the environment.
$\mathcal{R}_i$	Occupancy set based on i-th hypothesis H_i.
$M_{\mathrm{occupancy}}$	Operation of mapping hypotheses set to a set of occupancy sets.
$\mathbf{c}$	Control vector.
δ	Desired steering angle.
a	Desired acceleration.
$\mathbf{p}_{\mathrm{veh}}$	Vehicle's parameters.
$\mathbf{s}$	State of the ego vehicle.
$\mathbf{s}_{\mathrm{tp}}$	State of a traffic participant other than the ego.
x	Longitudinal position of the ego vehicle in the (X, Y) World Coordinate System (WCS).
y	Lateral position of the ego vehicle in WCS.
ψ	Orientation of the ego vehicle in WCS.
v	Absolute speed of the ego vehicle in WCS.
β	Angle between ego's longitudinal axis and its velocity vector.
l_f	Distance between ego's front axis and CoM.
l_r	Distance between ego's rear axis and CoM.
$\mathcal{R}_{\mathrm{wc}}$	Worst-case occupancy set.
H_{wc}	Worst-case hypothesis.
$\delta_{\mathrm{tp}_{\mathrm{min}}}, \delta_{\mathrm{tp}_{\mathrm{max}}}$	Steering angle limits of vehicles other than the ego.
$a_{\mathrm{tp}_{\mathrm{min}}}, a_{\mathrm{tp}_{\mathrm{max}}}$	Acceleration limits of vehicles other than the ego.
$\mathbf{p}_{\mathrm{veh,tp}}$	Parameters of vehicles other than ego.
$\mathbf{s}_0$	Initial state of a vehicle.
$\mathcal{S}_{\mathrm{pt}}$	Set of plausible trajectories of a vehicle other than ego.
$\mathcal{C}_a$	Set of plausible acceleration values of a vehicle other than ego.
$\mathcal{C}_\delta$	Set of plausible steering angle values of a vehicle other than ego.
n_t	Number of discrete time steps used in the generation of the occupancy set.
t_i	Discrete time step used in the generation of the occupancy set.

Continued: Symbols - Chapter 3

Symbol	Description
$\mathcal{R}_{\mathrm{ll}}$	Lane-limited occupancy set.
n_{corr}	Number of relevant driving corridors.
τ_i	i-th driving corridor.
n_{sv}	Number of the vehicles in a scenario excluding the ego.
ρ_{RSS}	Max response time in the Responsibility-Sensitive Safety (RSS) framework.
$a_{\mathrm{brake,RSS}}$	Braking deceleration in the RSS framework.
$a_{\mathrm{max,RSS}}$	Maximum acceleration in RSS framework.
$a_{\mathrm{brake,min,RSS}}$	Min braking deceleration in RSS framework.
$a_{\mathrm{brake,max,RSS}}$	Max braking deceleration in RSS framework.
$a_{\mathrm{ego,lon}}$	Longitudinal acceleration of the ego vehicle.
t_{viol}	Safety violation time.
$d_{\mathrm{min}}^{\mathrm{lon}}$	Minimum safe longitudinal distance.
t_h	Trajectory replanning interval.
n_{pred}	Number of alternative trajectories returned by a prediction method.
$\mathcal{R}_{\mathrm{pred}_i}$	Occupancy set based on i-th trajectory prediction.
$\mathbf{q}$	Vector of optimized variables.
$\mathbf{q}_i$	Optimization variables that describe the i-th control trajectory.
n_{params}	Number of parameters of a single trajectory.
$\mathbf{p}_{\mathrm{veh}}$	Vehicle parameters.
C_{lc_i}	Distance to centerline cost term of i-th trajectory.
C_{ctrl_i}	Squared control values cost term of i-th trajectory.
C_{v_i}	Speed keeping cost term of i-th trajectory.
C_{brake_i}	Braking cost term of i-th trajectory.
w_{lc_i}	Weight of C_{lc_i} cost term.
w_{acc_i}	Acceleration weight in C_{ctrl_i} cost term.
w_{δ_i}	Steering angle weight in C_{ctrl_i} cost term.
w_{v_i}	Weight of C_{v_i} cost term.
w_{brake_i}	Weight of C_{brake_i} cost term.
d_i^{min}	Min distance between the ego and i-th occupancy set.
$\mathbf{q}_{\mathrm{init}}$	Initial guess for optimized vector $\mathbf{q}$.

Symbols - Chapter 4

Symbol	Description
c	Feature class.

Continued: Symbols - Chapter 4

Symbol	Description
$\mathfrak{C}$	Feature classes set.
$\mathcal{S}_c$	Set of feature states of a class c.
$\mathcal{S}_c$	Set of features states of a class c.
$\hat{\mathcal{S}}_c$	Set of features state estimates of a class c.
n_c	Number of features of a class c.
n_{ce}	Number of feature state estimates of a class c.
$\mathbf{s}_c$	State of a feature of a class c.
$\hat{\mathbf{s}}_c$	State estimate of a feature of a class c.
$M^{(i)}$	i-th mapping operation in a sensor model.
n_k	Number of feature state estimates after k-th mapping operation.
n_m	Number of mapping operations in a sensor model.
$\mathbf{g}$	Dimensions of a dynamic object.
$\mathbf{x}$	Position of a dynamic object.
$M^{\mathrm{d,occ}}$	Dynamic objects occlusion mapping operation.
$\mathbf{p}_{\mathrm{d,occ}}$	Parameters of dynamic objects occlusion mapping[1].
$n_{\mathrm{d,occ}}$	Number of unoccluded dynamic objects.
$\{\mathbf{s}_{\mathrm{d,occ}_j}\}_{j=1..n_{\mathrm{d,occ}}}^{(\mathrm{d,occ})}$	Set of objects remaining after occlusion mapping.
T_{delay_i}	Detection delay of i-th object.
$T_{\mathrm{d,fn,dur}}$	Duration of a false negative error.
$M^{\mathrm{d,fn}}$	False negative errors mapping operation.
$\mathbf{p}_{\mathrm{d,fn}}$	Parameters of false negative errors mapping[1].
$n_{\mathrm{d,fn}}$	Number of dynamic objects after false negative errors mapping.
$\{\mathbf{s}_{\mathrm{d,fn}_j}\}_{j=1..n_{\mathrm{d,fn}}}^{(\mathrm{d,fn})}$	Set of objects remaining after false negative errors mapping.
$T_{\mathrm{d,fp,dur}}$	Duration of a false positive error.
$M^{\mathrm{d,fp}}$	False positive errors mapping operation.
$\mathbf{p}_{\mathrm{d,fp}}$	Parameters of false positive errors mapping[1].
$n_{\mathrm{d,fp}}$	Number of dynamic objects after false positive errors mapping.
$\{\mathbf{s}_{\mathrm{d,fp}_j}\}_{j=1..n_{\mathrm{d,fp}}}^{(\mathrm{d,fp})}$	Set of objects after false positive errors mapping.
P_{OU}	State estimate error model based on Ornstein-Uhlenbeck noise.
$M^{\mathrm{state,est}}$	False positive errors mapping operation.

[1] All parameters related to dynamic objects perception modeling with their description and values are listed in the Table 4.1

Continued: Symbols - Chapter 4

Symbol	Description
$\mathbf{p}_{\text{state,est}}$	Parameters of false positive errors mapping[1].
$n_{\text{state,est}}$	Number of dynamic objects after false positive errors mapping.
$\{\mathbf{s}_{\text{state,est}_j}\}_{j=1..n_{\text{state,est}}}^{(\text{state,est})}$	Set of objects after false positive errors mapping.
$\mathcal{S}_r$	Set of lane markers geometries.
$\mathbf{s}_r$	Lane marker geometry.
$\mathbf{c}$	Vector of lane marker polynomial coefficients.
h	Observed length of a lane marker.
$\hat{\mathcal{S}}_r$	Set of lane markers estimates (model output).
$\hat{\mathbf{c}}$	Lane marker coefficients model output.
$\hat{h}$	Observed length of a lane marker model output.
$M^{\text{r,smpl}}$	Lane markers sampling operation.
$\mathbf{p}_{\text{r,smpl}}$	Parameters of lane markers sampling operation[2].
n_{lm}	Number of lane markers.
$\mathbf{z}_{\mathbf{ij}}$	j-th sample of i-th lane marker.
$M^{\text{r,occ}}$	Lane markers samples occlusion operation.
$M^{\text{r,lsa}}$	Lane markers polynomials approximation operation.
$M^{\text{r,fn}}$	Lane markers false negative errors mapping operation.
P_{discard}	Probability of marking lane marker as a false negative.
$P_{\text{lm,recovery}}$	Probability of false negative lane marker recovery.
$M^{\text{r,geom}}$	Lane markers geometry estimation errors mapping.
$\mathbf{o}_{\text{ego}}^{(t)}$	Observation of the ego state.
$\mathbf{o}_{\text{obj}_i}$	Observation of an i-th dynamic object's state.
$n_{\text{obj,max}}$	Max number of observed dynamic objects.
$\mathbf{o}_{\text{lm}_i}$	Observation of an i-th lane marker's state.
$\mathbf{d}_{\text{lm}_i}$	Lateral offsets vector of the i-th observed lane marker.
γ_{lm_i}	Rotation of the i-th observed lane marker.
m_{lm_i}	Type of the i-th observed lane marker.

[2] All parameters related to static environment perception modeling with their description and values are listed in the Table 4.2

Symbols - Chapter 5

Symbol	Description
$\mathcal{S}_{\text{scen}}$	Set of test scenarios.
n_{scen}	Number of test scenarios.
n_{tp}	Number of traffic participants other than ego in scenario.
t_{sf}	Duration of the test scenario.
$\mathbf{p}_{\text{adv,veh,params}}$	Parameters of an adversarial road user.
$\mathcal{S}_{\text{tp}}$	Set of state trajectories of other traffic participants.
$\mathbf{s}_{\text{tp}_i}$	State of an i-th traffic participant.
$\hat{\mathcal{S}}_{\text{tp}}$	Set of state estimate trajectories.
$\hat{\mathbf{s}}_{\text{tp}_i}$	State estimate (ego's perception of object's state) of an i-th traffic participant.
$\mathbf{s}_{\text{ego}}$	State of the ego vehicle.
$T_{\text{state,tp}_i}$	State trajectory of i-th traffic participant.
$\mathcal{S}_{\text{r}}$	Set of static environment features.
$\mathbf{s}_{\text{r}_i}$	Vector describing i-th lane marker.
$T_{\text{ctrl,ego}}$	Ego's control trajectory.
$\mathcal{T}_{\text{ctrl}}$	Set of control trajectories of other traffic participants.
$T_{\text{state,tp}_i}$	Control trajectory of i-th traffic participant.
$\mathbf{q}$	Vector of trajectories' parameters.
n_{q}	Number of parameters that describe a single trajectory.
$\mathbf{q}_{\text{se}}$	Vector of parameters of state estimates trajectories.
$\hat{\mathfrak{S}}_{\text{env}}$	Environment description.
$\mathbf{c}$	Control vector.
δ	Steering angle.
a	Acceleration.
$\mathcal{R}_{\text{phys}}$	Set of constraints related to vehicles' physical limitations.
$\mathbf{p}$	Set of parameters used to calibrate generated trajectories.
v_s	Longitudinal velocity of a vehicle.
$\mathbf{O}_{\text{env}}$	Observation of environment's state.
$\mathbf{o}_{\text{ego}}$	Observation of ego's state.
$\mathbf{o}_{\text{obj}_i}$	Observation of i-th traffic participant.
$\mathbf{o}_{\text{lm}_i}$	Observation of i-th lane marker.
C_{coll}	Collisions-related cost term[3].
C_{TTC}	Time To Collision cost term[3].

[3] Weights and additional parameters related to cost terms can be found in Table 5.1

Continued: Symbols - Chapter 5

Symbol	**Description**
C_{dist}	Euclidean distance cost term^3.
C_{sim}	Scenario similarity cost term^3.

Operators and distributions

Symbol	**Description**
$\subset$	Subset of.
$\in$	Is an element of; e.g. $s \in \mathcal{S}$.
$\cup$	Union of sets.
$\cap$	Intersection of sets.
$\setminus$	Set difference.
Box	Planar bounding box of a vehicle.
Hull	Convex hull of polygons.
$\mathcal{N}$	Normal distribution.
$\mathcal{N}_k$	k-dimensional multivariate normal distribution.
$\mathcal{U}$	Uniform distribution.

1. Introduction

Road traffic accidents are a leading cause of death among children and young adults, with a staggering number of 1.35 million traffic-related deaths reported annually [131] and over 50 million injures [137], that often lead to permanent disabilities. Road accidents cause immense economic burdens related to treatment and rehabilitation costs, property loss, post-trauma productivity loss, and legal costs. Traffic injuries are estimated to impose a global economic burden of approximately 1.8 trillion dollars over the period of 2015-2030 [27].

Despite immense amounts of work and costs put into the development of safer infrastructure, improvements in vehicle safety systems, as well as legislative and educational efforts, the annual number of severe injuries and deaths related to traffic remains relatively constant [131], suggesting that new disruptive technological advancements or legislative changes are needed in order to put an end to this global crisis. Many researchers and automotive engineers share the belief that autonomous driving (AD) systems are such an advancement that may revolutionize traffic safety and significantly reduce the number of traffic accidents [72].

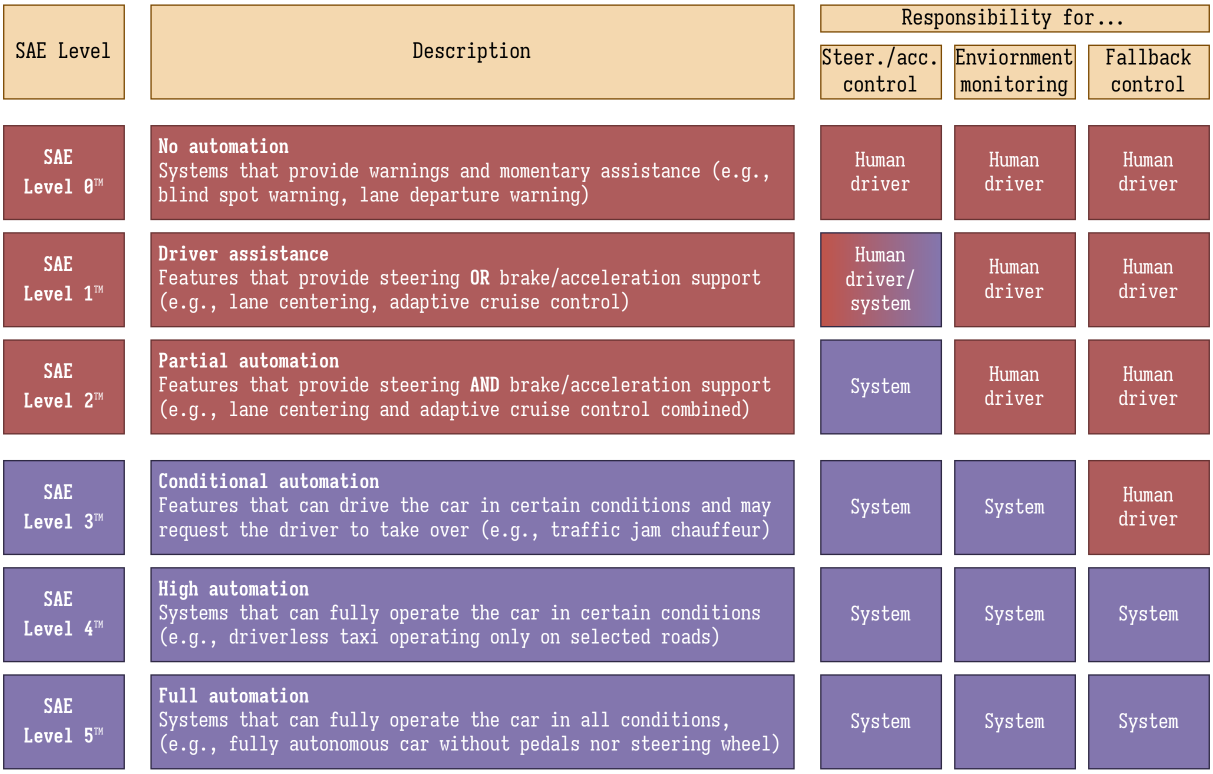
Numerous studies confirm that the overwhelming majority of accidents are caused by drivers' mistakes, with an estimated fraction of crashes caused by human errors reaching 90% [33] or even 94% [161]. Common causes behind such errors include distraction [149, 186] (present in up to 68% crashes [33]), alcohol-impaired driving (with approximately 30% traffic fatalities being related to alcohol consumption in the US [40]) and drowsiness [176].

In the context of these statistical findings, the wide adaptation of Advanced Driver Assistance Systems and AD systems that are immune to such impairments seems to be in fact a very promising premise. Unfortunately, with the extreme complexity of the driving task, the multitude of safety risks, and the potentially immense cost of wrong decisions during driving, a wide adaptation of such systems still requires a vast amount of research and a social discourse to set safety goals for AD vehicles and ensure that they meet them.

1.1. Driving Automation

Advanced Driver Assistance Systems (ADAS) are becoming increasingly common in modern vehicles, and most large automobile manufacturers offer a variety of systems that not only actively support drivers in their decisions, but even exercise a certain level of direct control over the vehicle. Features such as Autonomous Emergency Braking (AEB), Automated Lane Centering (ALC), or Adaptive Cruise Control (ACC) are slowly becoming a standard equipment in new cars.

In order to categorize the growing variety of driver assistance systems, the Society of Automotive Engineers (SAE) introduced a classification of vehicle autonomy levels [171], summarized in Fig. 1.1.



SAE Level	Description	Responsibility for...		
		Steer./acc. control	Enviornment monitoring	Fallback control
SAE Level 0™	**No automation** Systems that provide warnings and momentary assistance (e.g., blind spot warning, lane departure warning)	Human driver	Human driver	Human driver
SAE Level 1™	**Driver assistance** Features that provide steering **OR** brake/acceleration support (e.g., lane centering, adaptive cruise control)	Human driver/ system	Human driver	Human driver
SAE Level 2™	**Partial automation** Features that provide steering **AND** brake/acceleration support (e.g., lane centering and adaptive cruise control combined)	System	Human driver	Human driver
SAE Level 3™	**Conditional automation** Features that can drive the car in certain conditions and may request the driver to take over (e.g., traffic jam chauffeur)	System	System	Human driver
SAE Level 4™	**High automation** Systems that can fully operate the car in certain conditions (e.g., driverless taxi operating only on selected roads)	System	System	System
SAE Level 5™	**Full automation** Systems that can fully operate the car in all conditions, (e.g., fully autonomous car without pedals nor steering wheel)	System	System	System

Figure 1.1. Summary of SAE automation levels, based on [171]

.

SAE automation levels span from 0, which means that the vehicle is equipped only with systems that provide warnings or momentary braking/steering assistance, to 5, which describes fully autonomous vehicles that can operate without a driver on all roads.

Rapid automation of vehicles in recent years brought many Level 1 and Level 2 cars to the market. With such quick advancements in automation technology, it may seem that reaching higher levels of autonomy is only a matter of time and iterative improvements of existing technologies, considering that many systems are already capable of planning the motion of the vehicle and controlling it in an efficient manner.

Unfortunately, even if the planning and control algorithms needed for autonomy levels 2 and 3 may seem similar, there is an excessive difference between these levels in terms of required robustness, performance, and testing efforts. Since in the Level 3 vehicles the system is responsible for monitoring the environment, it must be able to properly detect dangerous situations and react to them. Ensuring that it will be able to do so in a sufficient number of situations and conditions requires not only robust planning algorithms, but also precise perception systems, and extremely thorough validation and verification.

The design of a safe AD system remains an open research area, with many problems requiring further investigation, such as safe trajectory planning, setting safety goals, designing safety constraints for planning and control algorithms, or developing efficient testing methods.

1.2. Safety

Safety concerns are an inseparable part of most engineering sciences. Their ubiquitous presence in the aviation, automotive, nuclear energy, and space industries may create the illusion that practices and standards in this area are already well established, and their application to the design of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) requires merely minor daptation of existing approaches. Safety standards, such as ISO 26262 [65] (Functional Safety), are in fact already widely used in the automotive industry, and newer standards, such as ISO/PAS 21448 [66] (Safety of the Intended Functionality, SOTIF) are being introduced to help develop reliable driver assistance systems. With advanced ADAS already available in commercial vehicles, it may seem that the development of fully autonomous driving systems is imminent.

Unfortunately, lessons learned from other industries, as well as from prior developments in the area of vehicle safety, are only partially applicable to the development of AD systems. Vehicles routinely operate in a complex, often unpredictable, dynamic environment. The handover of the driving task to an onboard system is thus related to a broad set of safety challenges unprecedented in other industries, as all actions undertaken by such a system are fraught with a considerable level of risk. Furthermore, many variables that are important for the safe execution of the driving task, such as the future behavior of other vehicles, remain largely unknown.

Existing standards only partially address these challenges. The Functional Safety standard (ISO 26262) focuses primarily on safety issues resulting from hardware and software failures, while in ADAS and AD systems, many hazardous situations can be a consequence of erroneous decisions, limited sensor performance, or behavior of other road users instead. SOTIF, on the other hand, while in fact focusing on hazards in the absence of faults, provides only a set of guidelines related mainly to the development and validation process, without complete solutions nor precise recommendations on how to set and achieve safety goals.

Recently introduced and less known IEEE Standard for Assumptions in Safety-Related Models for Automated Driving Systems (IEEE 2846-2022) [63] approaches the problem of planning a vehicle's motion in a dynamic environment more directly, proposing to use a set of assumptions regarding reasonably foreseeable behavior of other road users to define constraints for the planned motion. The standard remains relatively general; however, it requires further work to define precise requirements and develop planning and validation approaches that would fulfill the standard's requirements.

A vast amount of research is thus still needed in several areas to set safety goals for an AD system, develop perception, planning, and control algorithms that would fulfill them, and design feasible verification and validation methods.

1.3. Motivation and Scope

An immense amount of work has already been invested in the development of motion planning and vehicle control algorithms for the purpose of ADAS features with autonomy levels from 0 to 2. Still, considering the complexity and difficulty of the driving task, Autonomous Driving algorithms remain a very active research field, significantly accelerated by recent advancements in the area of Machine Learning (ML) algorithms. One of the most promising approaches for vehicle motion planning is the use of the Reinforcement Learning (RL) methods to develop a driving policy that would plan the near-future motion of the controlled car based on the data from the vehicle's sensors and perception algorithms [164]. RL-based policies have been shown to plan vehicle movement in an efficient and reliable manner while being able to take into account complex interaction between multiple vehicles, exhibit human-like on-road negotiation skills, and predict the behavior of other road users [82, 10].

While the capabilities of such algorithms are promising, several challenges related to ensuring their safety and reliability must be solved before they can be introduced on a large scale to commercial vehicles, including the issues listed below.

- Driving policies trained in RL setups often struggle to learn proper responses to events that are very rarely observed in the training process, or in which several possible outcomes of a given situation must be taken into account in the planning. In situations where one outcome of a given situation is considered significantly more plausible than the others, RL-based policies tend to disregard less plausible ones. For this reason, a more transparent trajectory planning method could be used in conjunction with the RL policy, either as a way to execute in a safe manner the high-level maneuvers chosen by the policy or to be used as an alternative way of planning in rare, atypical situations in which several hypotheses regarding the behavior of other vehicles can be formed.

- Perception systems utilized in road vehicles often suffer from performance limitations, producing various types of perception errors. RL-based policies typically trained in simulation environments may not be able to maintain desired robustness and performance in the presence of such errors, potentially leading to erratic, unsafe behaviors. More research is needed to evaluate the impact of sensing systems deficiencies on such algorithms and to improve their robustness to various types of perception errors.

- ○ The lack of transparency is an inherent characteristic of ML-based solutions that makes their validation and verification challenging. Erroneous decisions in such policies may not necessarily be directly correlated with the objective difficulty of a road situation, and thus manually designed test cases may fail to expose certain failures. Large-scale test drives, either performed on the road or in a simulation, while important for the final validation of the system, tend to be a costly and inefficient way of uncovering issues in AD systems related to rare, uncommon situations. Therefore, more research into automated adversarial testing is needed to explore potential issues in such systems.

Addressing these challenges is not only an important step required for the commercialization of RL-based AD systems, but could also provide valuable insights and methods for the development of various ADAS features, not necessarily based on ML algorithms.

In this thesis, I focus on these challenges, proposing several methods related to vehicle motion planning and validation of ADAS/AD systems. While the design and training of RL-based driving policies are not a direct focus of this thesis, proposed methods are mainly meant to be used in conjunction with such policies, as well as to ensure their robustness in the presence of perception errors, and help to validate them. To allow research related to proposed methods, exemplary RL-based policies are trained and used to test research hypotheses related to the defined challenges. It should be, however, noted, that most of the proposed methods are designed in a universal manner, remaining applicable to various types of driving policies, including ones that are not necessarily based on RL or even ML methods.

1.4. Research Hypotheses

The research hypotheses investigated in this thesis are directly related to the challenges identified in the previous section. Three main hypotheses are investigated, which are listed below.

1. It is possible to create a safe driving plan for an automated vehicle that considers several hypotheses regarding the future state of the vehicle's surroundings. In particular, reasonably foreseeable worst-case assumptions regarding the behavior of other road users can be taken into account in the motion planning algorithm, ensuring the existence of feasible collision avoidance maneuvers during the execution of the motion plan.

2. The use of stochastic models of perception systems in the training process of a Reinforcement-Learning driving policy improves the policy's robustness to perception errors.

3. Optimization-based adversarial scenario generation methods can be used in simulation-based validation of motion planning algorithms to expose potential weaknesses or issues in the evaluated systems.

1.5. Thesis Outline and Contributions

Chapter 2 contains a general introduction to the topic of autonomous vehicles, briefly describing their history, architecture, and methods used for motion planning.

Chapters 3 - 5 all share a similar structure. All of these chapters start with the motivations and review of the literature relevant to the discussed problems, followed by the problem statement, the description of the proposed solutions, the outline of the evaluation methods, and end with the experimental results and conclusions. Each chapter refers to one of the research hypotheses defined in Section 1.4.

In Chapter 3 a novel motion planning approach is introduced. The approach utilizes optimization-based trajectory generation methods to plan the vehicle motion while taking into account multiple hypotheses regarding the future state of the scene. The hypotheses may be generated using an arbitrary multimodal trajectory prediction method. In the proposed method, several vehicle control trajectories are planned, where each trajectory is related to one of the hypotheses, and all trajectories are identical in a predefined initial time period. This solution helps to ensure the existence of a feasible collision-free trajectory in this period, regardless of which hypothesis turns out to be true, allowing the postponement of safety-critical decisions until more data are collected regarding the environment and behavior of other road users. One of the important applications of the described method is Fail-Safe Planning, in which one trajectory is planned based on the most plausible hypothesis regarding the future behavior of other road users, while another respects safety constraints based on reasonably foreseeable worst-case assumptions regarding the future behavior of other road users. The main contribution presented in this chapter is a formulation of the optimization problem that enables planning of all the trajectories simultaneously. This allows ensuring that planning the trajectory based on the most plausible hypothesis will not render planning other trajectories infeasible, allowing one to successfully plan a safe motion of the vehicle in a significantly wider variety of scenarios compared to existing methods.

Although the method described in Chapter 3 can be used in conjunction with Reinforcement-Learning-based motion planning methods, e.g., to execute high-level maneuvers chosen by the RL-based driving policy in a safe manner, Chapter 4 focuses on the problem of robustness of such policies to perception errors. In this chapter, a set of efficient low-fidelity sensor models is introduced for modeling both the dynamic environment perception systems (e.g., radar-based object detection) and static environment perception (e.g., camera-based lane markers detectors). The proposed sensor models can be used both for the validation of driving policies and for their training. An exemplary driving policy is trained in a simulation environment with the use of the proposed sensor models and compared to the policies trained with simpler baseline sensor models and with a perfect environment perception.

While multiple sensor modeling techniques were already proposed in the literature, their application in training RL-based driving policies and their impact on the system's final performance are rarely analyzed. The main contributions presented in this chapter thus include the introduction of efficient models for both static and dynamic environment perception and the investigation of the impact of sensor modeling on driving policies' performance.

Chapter 5 addresses the problem of automatic validation of ADAS and AD systems. In this chapter, a novel method for the automated generation of adversarial test scenarios is introduced. The proposed method can be used to effectively explore potential issues in the developed driving policies, by using an optimization-based trajectory generation to find the behavior of other road users, that trigger safety-critical failures in the tested system.

A distinct feature of the presented method is the ability to generate adversarial scenarios with perception error patterns. Thanks to this feature, the method can be used to actively explore combinations of other's behaviors with plausible perception errors occurrences that could result in incorrect decisions of the driving policy.

The effectiveness of the method was demonstrated in the task of generating a database of adversarial scenarios for an exemplary RL-based driving policy.

The last chapter summarizes the contributions of this thesis and outlines possible further steps that could be taken to improve the proposed methods.

2. Background

The safety and performance of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) systems depend on many factors, such as the choice of sensors utilized in the system, perception and sensor fusion algorithms, the system's architecture, as well as planning and control algorithms. In this chapter, I provide an introduction to these topics, focusing on subjects that are most relevant to the approaches proposed in further chapters.

2.1. Advanced Driver Assistance Systems and Autonomous Driving Systems

Autonomous Driving is undoubtedly a very active and interesting research area pursued both by the automotive industry and the academic community. The introduction of efficient and safe AD vehicles may not only be very financially profitable but also bring great social benefits, saving many lives, and making roads safer and more environmentally friendly.

Analyzing the history of research in the AD area, one may notice that the efforts to create a truly AD system came from two distinct directions. One approach, which could be mainly observed in academic communities, was related to the development of heavily experimental systems with a high level of autonomy and working on increasing the Operational Design Domain (ODD) of the developed system. Another approach, pursued mainly by automobile manufacturers, was to start with SAE Level 0 driver assistance systems and iteratively develop and commercially introduce ADAS features with higher levels of autonomy.

Experiments with remote-controlled driverless cars began very early in the history of the automotive industry, with the first patents for such technologies appearing as early as in the 1890s [172], and on-road experiments reported to take place in 1920s [1]. However, the first experimental cars that could navigate on the road without human control started to appear significantly later, in the 1950s and 1960s, with a notable example of a self-driving system shown by RCA Labs in 1958 [16]. Nevertheless, most of the experimental systems showcased in these years by research laboratories and car manufacturers required specialized infrastructure, such as magnetic cables or electronic devices embedded in the roadways, and could not operate on public roads.

Experimental vehicles that could potentially operate without a specialized infrastructure began appearing in the 1980s, with a vision-guided vehicle developed at Carnegie Mellon University in 1986 [75], and a robotic van developed at Bundeswehr University Munich [16]. A decade later, autonomous vehicles capable of long highway driving with minimal human intervention were demonstrated, with VaMP and Vita-2 vehicles developed at Bundeswehr University Munich driving around 1000 kilometers in heavy highway traffic in 1994 [184], followed by a Navlab project driving nearly 5000 kilometers across the United States in 1995 [3] and few other similar successes [16].

In the years 2000-2010, the rapid growth of the autonomous driving research field could be observed, accelerated mainly by famous DARPA (Defense Advanced Research Projects Agency) challenges. The DARPA Grand Challenges, held in 2004, 2005, and 2007, offered substantial prizes to research teams that could create autonomous vehicles that would be able to complete predefined courses in a desert environment (first two challenges) and in an urban environment (third challenge). The challenges were well received, sparking significant interest in the development of AD systems in an academic community, and many research teams prepared their own systems for each challenge [173, 21].

Approximately from 2010 onward it is possible to observe significant investments in the development of AD technology made by big automotive manufacturers and technological companies. Examples include Google's creation of the Waymo autonomous driving company, the introduction of Tesla's autopilot feature, Uber's self-driving research, and the establishment of many startups and companies focused directly on AD systems, such as Motional, Zoox, Cruise, or Argo AI [4, 195, 31, 155].

At the same time, a wide variety of ADAS features were introduced to commercial vehicles [60, 198, 29]. Nowadays many Level 0 and Level 1 features are considered almost essential in modern vehicles, with systems such as Adaptive Cruise Control (ACC), Automatic Emergency Braking (AEB), Cross Trafic Alert (CTA), or Lane Keep Assistance (LKA) progressing from practically unavailable in commercial vehicles to being equipped in more than 60-70% new vehicles in 2022 [68].

Currently, many automotive companies already offer vehicles with Level 2 autonomy [26], with first attempts to commercialize Level 3 systems ongoing [2].

2.2. Perception Systems Used in Automotive and Their Limitations

Modern vehicles are equipped with a large number of sensors that provide information about the state of the vehicle's mechanical components, its occupants, and the environment in which it operates. For the design of ADAS / AD systems, the last position will be the most relevant, as information about other traffic participants, as well as road geometry, is critical for motion planning tasks, and the performance of widely used sensors poses a challenge for planning algorithms. It should be noted however that the performance of ego's state estimation, which may utilize, for instance, GPS, wheel encoders, and Inertial Measurement Units (IMUs), may also severely impact the motion planning task.

The most commonly used sensors for exterior sensing are cameras, LiDARs, radars, and ultrasonic distance sensors. In this section we will focus on the first three, as ultrasonic sensors are less relevant for on-road planning tasks because of their limited range, finding their application in parking-related tasks instead.

Understanding the operation principles of sensors and the underlying physical phenomena that may impact their performance is important for the design of motion planning algorithms, as each sensor may produce unique error patterns that have to be taken into account.

2.2.1. Sensors

Each of the types of sensors commonly utilized in automotive applications suffers from certain performance limitations, often associated with environmental conditions or road situations. As perception systems often play a safety-critical role in ADAS and AD systems, multiple types of sensors are often used to provide redundant environment state estimations, ensuring the system's robustness to temporary performance degradation of a single sensor or sensor type. It should be noted, that not all sensor types are able to provide a complete estimation of all environmental features that may be of interest for ADAS/AD system design - for instance, automotive radars

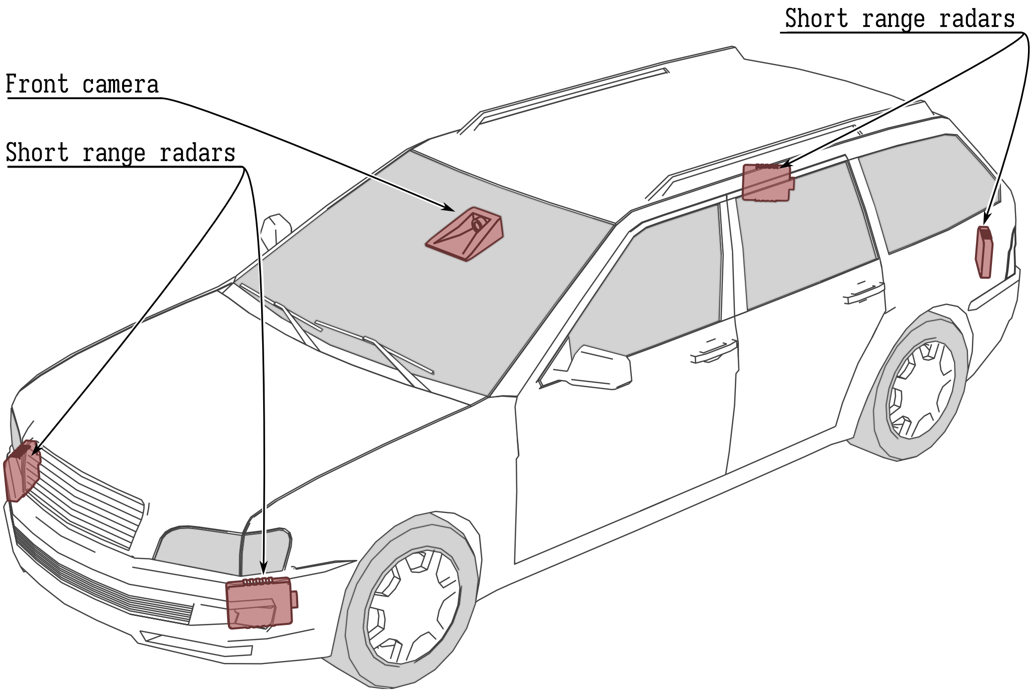


Figure 2.1. Example of the ADAS/AD sensor stack and mounting positions.

do not have the capability of the lane markers geometry estimation, while cameras do not have the ability of instantaneous object velocity measurement.

The capabilities of various automotive sensors, as well as their limitations, are summarized in Table 2.1.

Capability	Radar	Camera	LiDAR
Robustness to rain and mist	$+++$	$-$	$+$
Robustness to poor lighting conditions	$+++$	$--$	$+++$
Object classification	$---$	$+++$	$+$
Velocity measurement	$+++$	$---$	$---$
Range estimation	++	$-$	$+++$
Lane markers geometry estimation	$---$	$+++$	$---$
Traffic signs recognition	$---$	$+++$	$---$
Computational efficiency	++	$-$	$---$
Cost-effectiveness	++	++	$---$

Table 2.1. Comparison of common types of automotive sensors. $+++$ denotes the highest capability/robustness in a given area, while $---$ the lowest.

2.2.1.1. Radar

Radar (RAdio Detection and Ranging) sensors are widely used in the automotive industry due to their low cost and robustness to various adverse weather conditions. The lack of moving parts makes modern radars durable and resistant to vibrations, temperature changes, and other adverse conditions present during vehicle operation.

In principle, radar sensors operate by emission of electromagnetic signals and detection of the returning waves that were reflected from surfaces of the objects in the sensors' vicinity. Since radar waves propagate with a known speed (namely, the speed of light), it is possible to determine the range of the object that reflects the wave based on the time-of-flight of the emitted signal, i.e., the time elapsed from the signal's emission to the return of the reflected wave.

Modern automotive radar utilizes wavelengths on the order of millimeters, most commonly operating at frequencies between 76 GHz and 81 GHz, which corresponds to a wavelength of approximately 4 mm [188]. Being considered relatively short in the electromagnetic frequency spectrum, millimeter-range wavelengths enable relatively precise measurements, while providing robustness to environmental conditions, such as rain or fog, that severely impact the performance of sensors that operate in the visible light spectrum.

2.2.1.1.1 Features and operating principles

One of the important features of radar sensors is their ability to measure the radial velocity of the objects that move relative to the sensor. Velocity measurement utilizes the Doppler effect - a shift in the frequency of the wave that is reflected from a moving object proportional to the speed with which the distance between the object and the sensor changes. Knowing the shift in wave's frequency, radial velocity v_r can be computed as:

$$v_\mathrm{r} = \frac{cf_\mathrm{D}}{2f_\mathrm{T}}, \tag{2.2.1}$$

where c is the speed of light, f_D is Doppler's frequency (frequency of the returning wave), and f_T is the frequency of the emitted wave.

The ability to measure the object's radial velocity using the Doppler effect distinguishes the radar sensors as the only source of reliable velocity measurement that does not rely on differentiation of multiple range measurements, making them exceptionally useful in applications that require relatively precise velocity measurement, such as Adaptive Cruise Control systems.

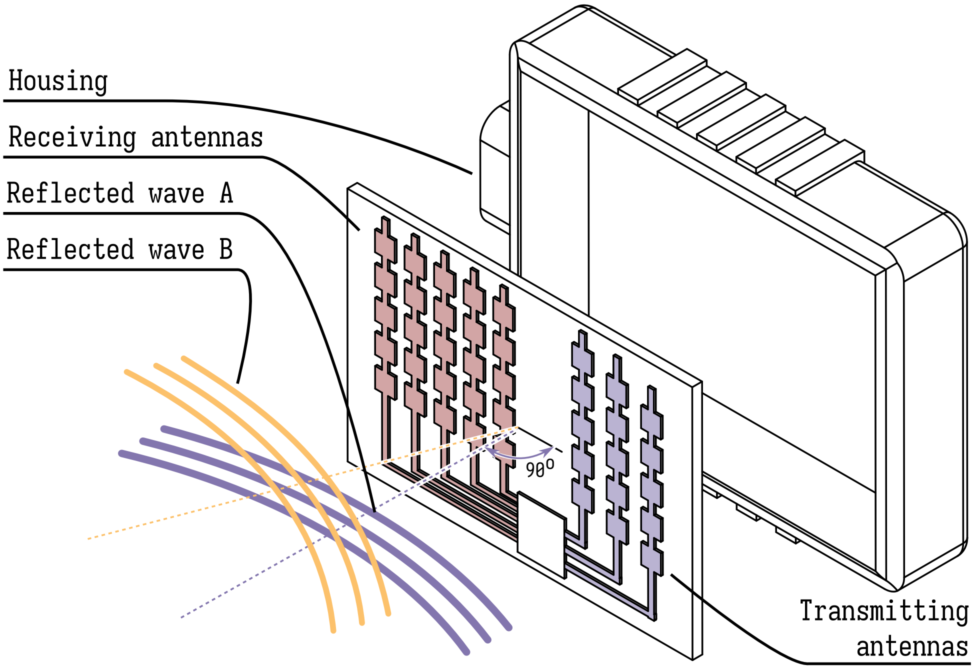


Figure 2.2. Determination of the object's azimuth based on the phase shift between multiple antenna elements. If the reflected wave is incoming from the direction perpendicular to the antenna array (Reflected wave A, in blue), the phase difference of the wave on antenna elements is minimal. For the objects situated at different angles (example: Reflected wave B, yellow), the phase shift between array elements is proportional to the object's azimuth angle.

Although historically a dominant way of determining an object's azimuth (angle relative to the sensor) was based on the mechanical rotation of the radar's antenna, such a solution would be difficult to execute reliably in a reliable and cost-effective way in the vehicle. Instead, an array of multiple receiver antennas is used to determine an azimuth angle based on the phase shift between the antennas placed within a known distance from each other, as shown in figure 2.2. In particular, a fast Fourier transform (FFT) can be performed across the antenna elements to determine the frequency of the phase change between elements, which will be proportional to the azimuth angle [136].

The power of the signals received by the radar sensor varies significantly depending on multiple factors, mainly the distance of the object and its Radar Cross Section (RCS). RCS is a property unique to all objects that describes how well a given object reflects the radar waves to its source. RCS itself depends on many aspects of the object, including but not limited to the object's size, geometry, the material of its surface, and orientation relative to the source of the radar wave.

Assuming that the receiving antenna is close to the transmitting antenna, the power of the wave p_{r} on the receiving antenna can be approximated using Equation 2.2.2.

$$p_{\mathrm{r}} = \frac{p_{\mathrm{t}} g_{\mathrm{t}} a_{\mathrm{r}} \sigma}{(4\pi)^2 r^4}, \tag{2.2.2}$$

where:

p_{t} - power of the transmitter,

g_{t} - transmitting antenna's gain,

a_{r} - effective aperture of the receiving antenna

σ - RCS of the object,

r - radial distance of the object.

2.2.1.1.2 Radar data processing

There are several ways in which the raw signals received by the radar sensor can be processed to acquire a useful description of the environment [136]. In this section, a simplified example of a data processing pipeline will be presented, but it should be noted that the topic of radar object detection and tracking constitutes a vast research area, and many different algorithms are proposed for this task, including models based on machine learning that often follow an entirely different approach [9, 35]. However, in commercial applications, classic solutions based on deterministic algorithms and tracking methods remain the most widely used.

Although the design and implementation of object detection and tracking algorithms are beyond the scope of this work, understanding the base principles behind these algorithms is important, as each major step of these algorithms introduces a potential for certain errors that will impact the performance of the whole system.

An example of a data processing pipeline is shown in figure 2.3. Raw data from Analog/Digital converters are used to compose the so-called Radar Data Cube (RDC). RDC is a multidimensional array composed of complex-valued baseband samples whose dimensions represent radial velocity, range, and angle of the detection. RDC is then used to extract a sparse map of radar detections, based on a signal-to-noise threshold. Since especially in short ranges multiple

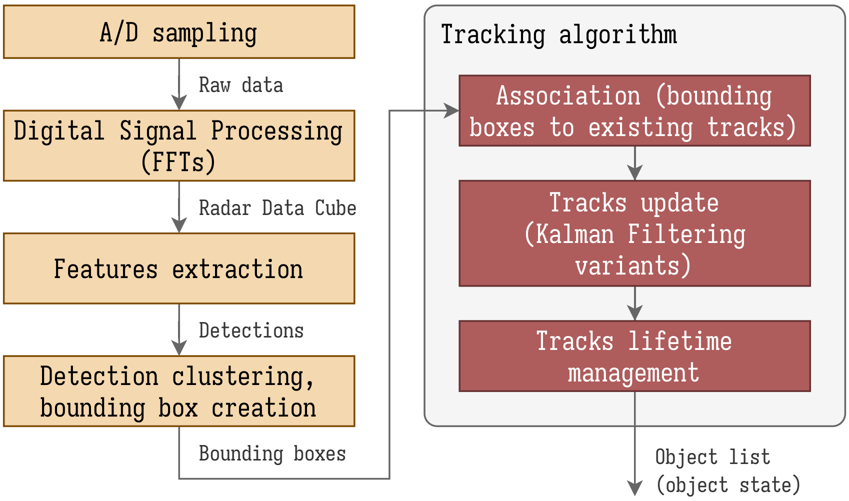

A/D sampling
Raw data
Digital Signal Processing (FFTs)
Radar Data Cube
Features extraction
Detections
Detection clustering, bounding box creation
Bounding boxes
Tracking algorithm
Association (bounding boxes to existing tracks)
Tracks update (Kalman Filtering variants)
Tracks lifetime management
Object list (object state)

Figure 2.3. Radar data processing workflow.

detections may be caused by a single object, e.g. due to multiple areas with strong radar reflection (scattering centers), the detections are clustered to form objects' bounding boxes and candidates.

The candidate bounding boxes serve as input to the tracking algorithm, usually based on derivatives of the Kalman Filter algorithm [104]. Bounding box candidates are associated with already tracked objects (tracks) or are used to create new tracks if no strong associations can be found. After association, previously existing tracks can be updated based on the state of the new bounding box candidate.

It should be noted that the use of the tracking algorithm introduces a certain correlation between previous objects' state estimates and current values. Kalman-based approaches utilize simplified models of the observed objects to update the object's state estimate based on the previous measurement, e.g., the object's velocity measurement may be used to update its position in the next time step. The estimates are then corrected on the basis of new measurements.

Lastly, the track lifetime management algorithm is used to oversee the existence of the tracks. This type of algorithm is typically used to decide when existing tracks should be removed (e.g., when in a few subsequent steps no bounding-box candidates can be associated with them) and when the new tracks should be created.

2.2.1.1.3 Error modalities

Physical properties of electromagnetic waves in automotive radar's spectrum, hardware limitations of cost-effective radar sensors, as well as algorithms utilized for object detection and tracking, introduce several types of errors that may lead to safety hazards in the ADAS/AD systems that utilize them.

Differently than in the case of waves in the visible light spectrum, a wide range of surfaces present in a typical road environment causes a specular reflection of the millimeter-range wave.

Contrarily to the diffuse reflection, mirror-like specular reflections introduce multipath propagation, that is, situations in which the transmitted radar wave does not return directly to the sensor after direct reflection from the detected object, but instead bounces from one or more other surfaces (e.g., surfaces of metal barriers, buildings, or vehicles) on a way to or from the object. This phenomenon can lead to false positive detection errors, where the sensor detects such undesirable echo as a real object [90]. Such falsely detected objects are often referred to as multipath ghosts or ghost objects, in short.

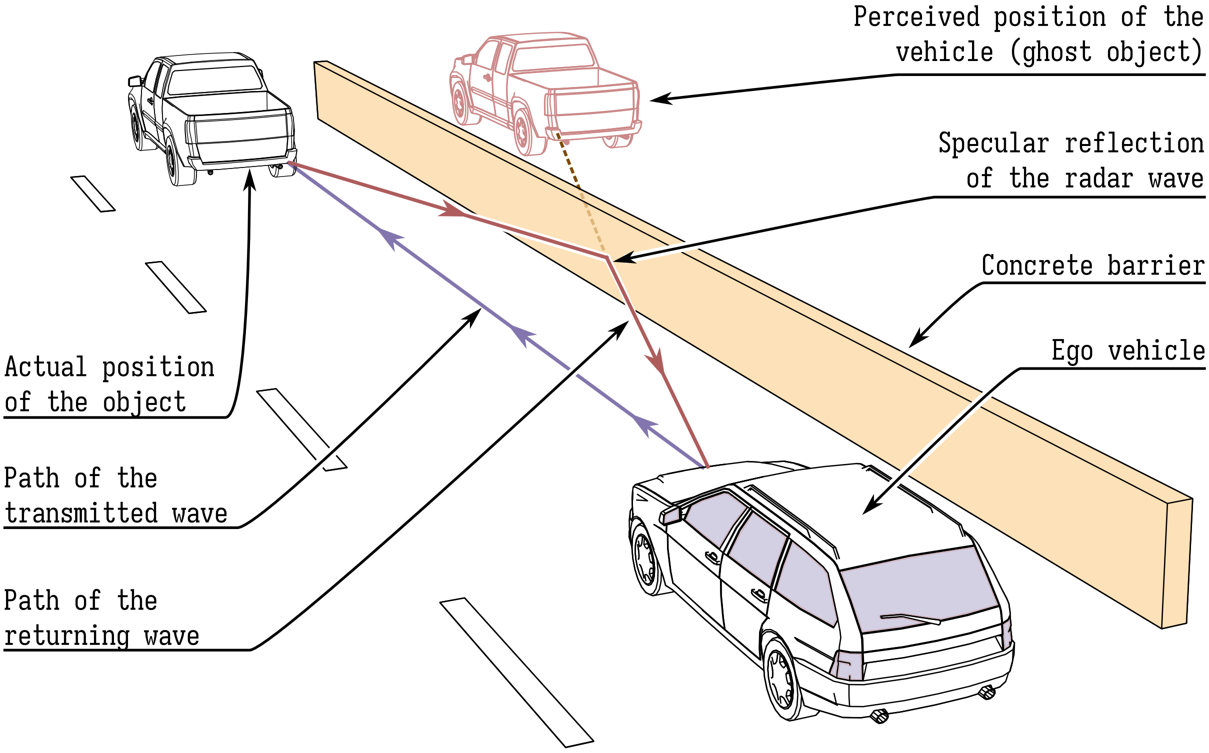

Perceived position of the vehicle (ghost object)
Specular reflection of the radar wave
Concrete barrier
Ego vehicle
Actual position of the object
Path of the transmitted wave
Path of the returning wave

Figure 2.4. Example of a false positive detection error caused by the multipath radar wave propagation.

Distinguishing multipath ghosts from real objects is a difficult task, as they often are detected in subsequent radar measurements in positions consistent with the measured velocity, e.g., when the echo of the existing vehicle is reflected from a large flat surface, such as a concrete barrier along the side of the road. Several methods have been proposed to detect such ghosts and thus improve the system performance [112, 108], although their applicability in small cost-effective sensors varies.

While radar sensors are considered relatively robust to difficult weather conditions, a certain level of performance degradation may still be observed when operating in rain or snow. Both water splashes and snow debris falling from other vehicles can be falsely detected as objects relevant to ADAS/AD systems. Traction losses of the ego vehicle also may be detrimental to the radar performance, as they impact the ego's state estimation, which plays an important role in both tracking and object detection. For instance, ego's velocity estimation is typically used to filter out stationary detections, e.g., from rough road surfaces, and its erroneous estimates may result in false positive detection of dynamic objects.

It should be noted that the level of noise detected by the radar sensor is significant compared to other commonly used sensor types. Many factors contribute to radars' noisiness, including interference with other radar sensors, multipath reflections, and electromagnetic pollution. Especially in cluttered urban environments with a large number of traffic participants and static objects in the sensors' vicinity, the noise level may severely impact the performance of radar sensors, introducing false positive detections. To partially alleviate this issue, potential detections with low signal strength compared to background noise are filtered out according to certain signal-to-noise ratio thresholds. This in turn introduces a certain probability of false negative detection errors.

Another class of potentially safety-critical errors is state estimate errors in which a particular object is detected but one or more state variables are disturbed. It should be noted, though, that severe errors in certain state estimates, especially position, may be classified as a false positive and false negative object detection pair. In the case of testing and validation methods in which ground truth data is compared to the sensor stream, usually, certain criteria are chosen to distinguish between these two situations, e.g., based on calculating the Intersection over Union (IoU) between the detected bounding box and the ground truth.

Multipath reflections and ego state estimation errors described in previous paragraphs both may lead to state estimation errors if the position disturbance is not high enough to result in a false positive.

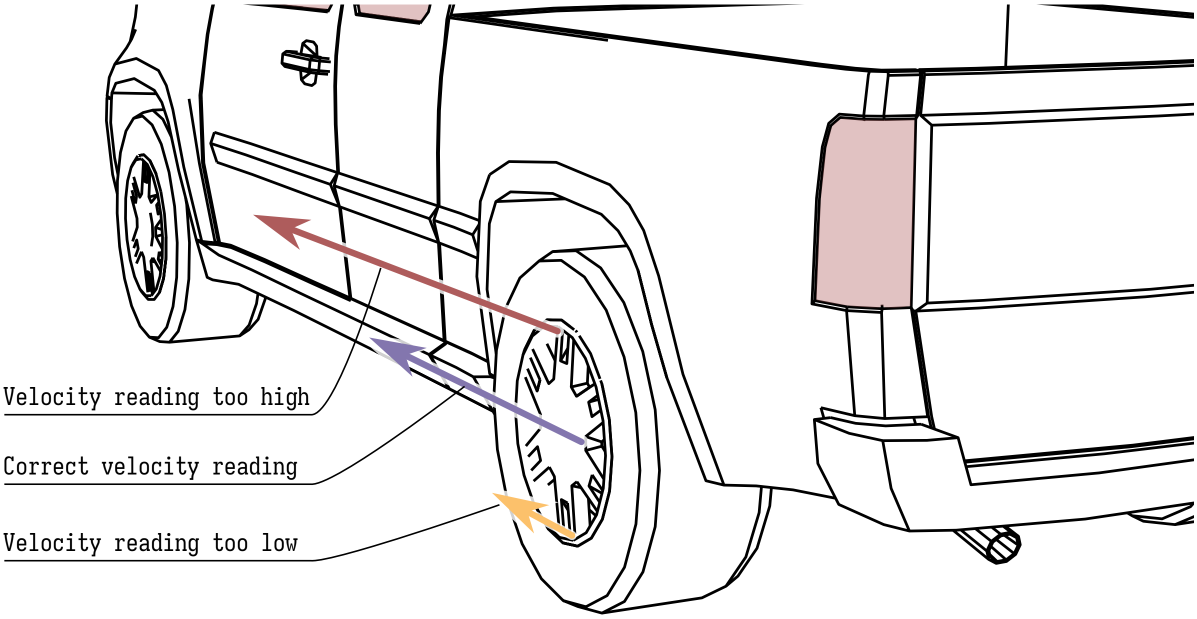


Figure 2.5. Mechanism of velocity measurement errors due to spinning wheels reflection. Since the wheel rotates relative to the vehicle, strong reflection from its different parts may result in severely disturbed velocity estimates.

Object velocity estimation errors constitute an important subset of errors. Various parts of the detected objects may differ substantially in RCS depending on their orientation relative to the sensor. Because of this, there is a certain chance that part of the object that moves relative to the object's local coordinate system (e.g., the rotating wheel of the vehicle or one of the pedestrian's limbs) will reflect radar's wave strong enough to contribute significantly to the object's velocity estimation. This is especially important in the case of the vehicle's wheels because complex rim shapes often serve as efficient scattering centers [78]. Since the magnitude of the instantaneous velocity of the different parts of the wheel relative to the ground varies from zero (at the bottom of the wheel) to the double vehicle's speed (at the top part of the wheel), they may lead to severe velocity estimation errors, as shown in Fig. 2.5.

2.2.1.2. Camera

Modern vehicles equipped with ADAS/AD systems use cameras in a wide range of subsystems and applications. Their high resolution and ability to operate in a visible and/or infrared light spectrum make them useful not only for object detection tasks but also for traffic sign recognition, lane markers detection, driver monitoring, object classification, semantic segmentation of vehicle surroundings [34], and many other functions. At the same time, cameras remain relatively inexpensive sensors, offering long-term reliability.

2.2.1.2.1 Operating principles

A typical camera module is composed of several hardware parts. The light enters the device through a lens system, that focuses the image on the image sensor. The sensor is typically covered by an infrared filter and a subsequent Color Filter Array (CFA), which is a mosaic of pixel-size color filters that enables the capturing of color information by the image sensor, which cannot effectively measure the wavelength of captured light directly. The use of the CFA allows reconstruction of the colors of the captured image in a demosaicing process, by measuring the light intensities of the pixels covered by each of the colors in the CFA separately, effectively capturing several single-color images at the same time (e.g. red, green, and blue).

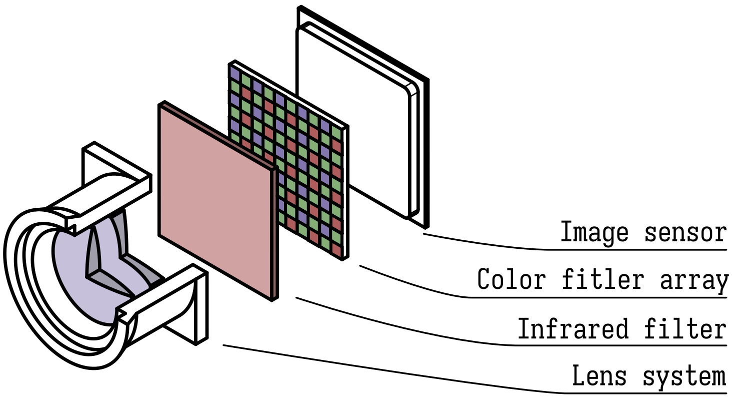


Figure 2.6. Simplified camera hardware model.

Two types of image sensors are commonly used in digital cameras, including camera-based automotive sensors: Charge-Coupled Devices (CCD), and active-pixel sensors, most commonly based on Complementary Metal-Oxide-Semiconductors (CMOS).

CCD sensors are effectively an array of light-sensitive analog devices that become electrically charged when exposed to light. The charge is amplified by a series of amplifiers and converted to a digital signal by Analog/Digital Converters (ADC) in a sequential manner, i.e., charges of the pixels closest to the amplifiers are read, and then charges of all pixels in the array are shifted towards amplifiers by a single pixel. This solution enables the amplifiers to be placed outside the photosensitive pixel array.

In CMOS sensors, on the other hand, each pixel in the array is coupled with its own amplifier. As the amplifiers are effectively placed along each of the pixels on the sensor, the effective light-capturing area of the sensor is smaller than that of CCDs. To increase the amount of light that affects each pixel, an array of microlenses is often placed in front of the sensors to focus the light that would otherwise hit the amplifiers on the photosensitive part of the sensors.

Both types of sensors have their advantages: CCD offers a higher dynamic range and lower noise, while CMOS is faster, less expensive, and more power efficient.

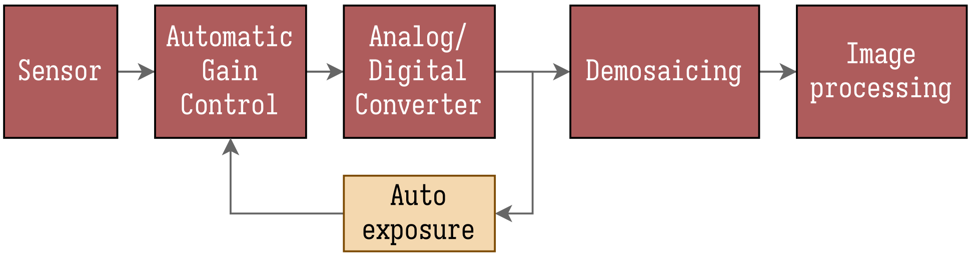


Figure 2.7. Camera operating scheme.

Independently from the sensor type, amplified charges are converted to digital signals and further processed - starting with a demosaicing process, which is used to reconstruct a color image from the pixels overlaid with a color filter array. Subsequently, the image can be further enhanced by gamma correction and noise reduction, as well as subjected to color conversion, scaling, and compression, if necessary.

Pre-processed images can be used in object detection algorithms. Historically, various methods were proposed for this purpose, including methods based on optical flow [22, 94], detecting vehicle shadows on an even road surface [183], detecting vehicle lights [28], or finding symmetric features on the image, taking advantage of the fact that front and rear views of vehicles are vertically symmetric [91]. With recent advances in machine learning technology, currently, the dominating approach is to use ML-based object detection algorithms for this task. Especially the introduction of approaches that do not require computationally expensive sliding window object classification, such as YOLO (You Only Look Once) [146] or Single-Shot Detectors (SSD) [111], enabled accurate real-time object detection for automotive applications [190, 200, 159].

Knowing the lens characteristics of the camera and its mounting position, the 3D dimensions and position of the detected objects can be easily approximated using simple geometric transformations, providing information usable in ADAS/AD algorithms.

2.2.1.2.2 Error modalities

One of the important aspects of the camera sensor is its limited performance with regard to state estimation. While the azimuth of an object usually can be measured in a direct, relatively precise manner, its distance from the sensor can only be estimated based on its size and/or its position projected onto an image plane.

One of the unfortunate results of the distance estimation's limited performance is difficulty in velocity estimation, as it cannot be measured directly but has to be derived as a change of the velocity estimate over time.

Both distance and velocity measurement play a safety-critical role in many ADAS features, as well as in AD systems. For this reason, cameras are often coupled with radar sensors in such applications, but camera-only AD systems proposals are not unheard of, as such an approach significantly reduces the cost and complexity of the entire system.

Missed detection errors are one of the most safety-critical issues in camera sensors. Inability to detect a vehicle, traffic sign, lane marker, or static obstacle may lead to dangerous situations in ADAS and AD systems. This class of errors may be caused by external factors, such as poor lighting conditions, or internal factors, such as the limited performance of object detection algorithms.

There is a large number of factors that can affect the detection performance, the most common of which are listed below.

High dynamic range. While camera sensors are usually able to adapt to lighting conditions by exposure modification (by changing the sensitivity or shutter speed), situations in which different parts of the image have drastically different lightness levels usually pose a difficult challenge. A common example of such a situation may be observed on tunnel exits during the day - usually lighting conditions inside the tunnel are much worse than outside and sensors with insufficient dynamic range tend to either overexpose the external part of the road or underexpose the internal one. Overexposure and underexposure can cause missed detections in the affected regions of the image.

Atypical appearance of objects. Detecting the presence of an object by the camera sensor usually requires the use of object recognition algorithms, most often based on supervised learning techniques. Although modern algorithms offer a good generalization, i.e., are able to correctly detect objects that were not present in the training data, they still may fail to detect objects of severely atypical presence. Examples of such objects may include vehicles with complex painting patterns, uncommon animals, or large objects transported by traffic participants.

Poor visibility environmental conditions. The performance of object detection by camera sensors can be severely affected by environmental conditions such as fog, heavy rain, or snow. This limitation is particularly important, as such conditions also affect the perceptive abilities of the driver and other traffic participants, potentially leading to the aggregation of dangerous factors.

Occlusions. Although occlusions impact all sensors used for object detection, they pose an additional challenge in the case of traffic signs recognition and lane markers perception, as an accumulation of even a thin layer of dirt or snow may lead to false negative detection errors.

False positive detection errors, while less common, may also lead to dangerous situations, although typically less severe than in the case of false negatives. One of the main sources of such errors is the confusion of environmental features observed in the background with relevant objects [59]. Although this kind of error may be caused by deficiencies in algorithm design or model training, certain patterns and objects pose particular challenges to most object detection algorithms. Due to the two-dimensional nature of the input data, images of the relevant objects (e.g., vehicles' photos on advertisement billboards, or markings similar to traffic signs painted on the vehicles) may be easily confused with real relevant objects. Other particularly difficult situations include the presence of the old lane markers alongside the new ones (e.g., during the roadwork) and light sources during the night that may be easily confused with the vehicle's headlights.

The last type of common error is misclassification. Cameras often play a central role in object classification. Since the class of the object provides vital cues to the object's motion model and expected behavior, this class of errors may lead to safety-critical situations as well. To provide an example, incorrect classification of the motorbike as a bicycle may lead to issues in tracking, ADAS, and AD algorithms, e.g., due to the choice of incorrect motion model or incorrect assumptions regarding possible maneuvers and achievable acceleration levels of the relevant object.

2.2.1.3. Other sensors

A wide variety of sensors is utilized in addition to cameras and radars both in the development of ADAS and AD systems, and in a final system on board commercial vehicles, including, but not limited to, the sensor types listed below.

LiDARs. LiDAR stands for *Light Detection and Ranging* and is a sensor based on short light impulses sequentially cast in the form of narrow infrared light beams around the device [105]. Although the use of strong lasers and time-of-flight measurement allows us to build a 3D model of the environment with precision often unmatched by other sensors, currently LiDARs rarely find applications in commercial vehicles. While the development of solid-state LiDARs is ongoing, most of the commercially available devices are relatively expensive and utilize moving elements, such as rotating mirrors or detecting modules, making them prone to failures in the long run. For these reasons, the use of LiDARs is typically limited to experimental platforms and test vehicles used in the development process.

GPS and DGPS. GPS (*Global Positioning System*), while frequently used for tasks related to road planning and navigation, typically does not have sufficient accuracy to be used in path planning or vehicle control. There are, however, augmentation methods that allow to enhance the GPS' accuracy, such as ground-based Differential GPS (DGPS), which utilizes one or more reference stations placed in known fixed positions to provide corrections to the GPS measurement. As the use of DGPS can result in positioning accuracy in a range of 1-3 cm, they are often used during the development of ADAS and AD systems, providing ground truth reference data on the position of test vehicles during test drives [199].

Ultrasonic sensors. This type of sensor uses ultrasound waves to estimate a distance to the nearest obstacle by measuring the time after which the emitted sound wave returns to the sensor after being reflected from the obstacle's surface [23]. While ultrasonic sensors are inexpensive and reliable, allowing measurement of distance with relatively high precision, they operate only on short distances and cannot be used to reliably determine a precise radial position of detected obstacles. For these reasons, their use is typically limited to parking applications.

Last but not least, the ADAS and AD systems rely on a variety of sensors that are used to estimate the state of the controlled vehicle itself, such as Inertial Measurement Units (IMU) or wheel encoders.

2.2.2. Sensor Fusion

Perception systems used for AD/ADAS vary in terms of sensors and architecture utilized; however, it is relatively common to combine several types of redundant sensors for certain tasks in highly autonomous AD systems. One of the typical examples is the use of cameras and radars for the task of detecting and estimating the state of dynamic objects.

As described in the previous section, both types of sensors are burdened with certain types of errors that are largely uncorrelated between sensor types. For example, low-light conditions may pose a severe challenge for camera-based perception systems, while not having any influence on the radars. On the other hand, false positive errors due to multipath reflections that are problematic in the case of radar-based object detection do not impact camera-based systems.

To achieve the high accuracy and low error rates required for highly autonomous AD systems, a reasonable choice is thus to use both types of sensor to create an accurate representation of the environment. The process of combining data from several sensors is referred to as *sensor fusion*, or *fusion* in short.

The algorithms used for fusion as well as the architecture of the fusion module depend on the type of sensors used in the system, but in most cases fusion systems can be classified into two main types: high-level and low-level sensor fusion.

2.2.2.1. High-level Fusion

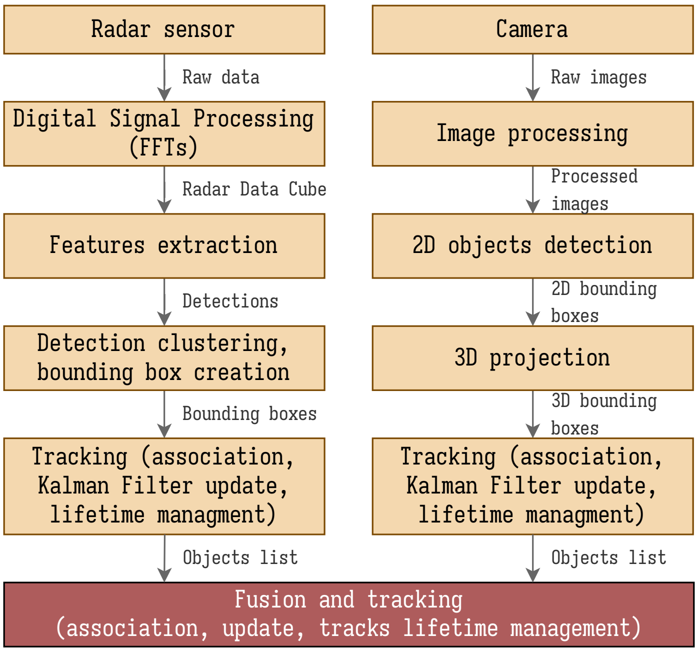

Radar sensor
Raw data
Digital Signal Processing (FFTs)
Radar Data Cube
Features extraction
Detections
Detection clustering, bounding box creation
Bounding boxes
Tracking (association, Kalman Filter update, lifetime managment)
Objects list
Camera
Raw images
Image processing
Processed images
2D objects detection
2D bounding boxes
3D projection
3D bounding boxes
Tracking (association, Kalman Filter update, lifetime managment)
Objects list
Fusion and tracking (association, update, tracks lifetime management)

Figure 2.8. Example of a high-level (track-to-track) sensor fusion arhcitecture. Sensor-specific detection and tracking steps are marked yellow.

In high-level approaches, also referred to as late-fusion approaches, the raw data from sensors is typically heavily processed before the fusion itself, often using the same algorithms as ones that would be used for perception and tracking in systems with only one type of sensor. As the raw data from various sensors is in many instances multimodal, the sensor-specific preprocessing often additionally converts the data to a common format, e.g., an object list consisting of objects' state estimates in the case of dynamic object detection applications.

As an example, the mentioned fusion of radar and camera sensors for the detection of dynamic objects can be considered. An exemplary overview of a high-level fusion algorithm used for these sensors is presented in Fig. 2.8.

The input to the fusion algorithm in high-level fusion architecture may be composed of complete object lists, fully preprocessed by perception and tracking algorithms distinct for each sensor type. This relatively common high-level fusion setup is often referred to as "track-to-track fusion", as what is effectively fused are the tracks, i.e., outputs of sensor-specific tracking algorithms.

Algorithms used for high-level fusion often resemble perception and tracking algorithms used for single-sensor data processing. An exemplary fusion algorithm may consist of the components listed below.

- Association - an algorithm step that is used to assign new measurements from each sensor to the existing *tracks*, i.e., objects' state estimate vectors from the object list created in previous iterations of the fusion algorithm. Simplest association approaches may be based on the measurement of the geometric distance between input objects and existing tracks, or the measurement of Intersection over Union (IoU) [17]. Other approaches to the association problem may take into account the Mahalanobis Distance between existing tracks and new ones, as well as utilize more advanced data association algorithms, such as the Jonker-Volgenant-Castanon (JVC) algorithm [70] or Joint Probabilistic Data Association (JPDA) techniques [49].
- Tracks lifetime management - a step in which new tracks are created if measurements from the sensors confirm (with sufficient certainty) the existence of an object that cannot be associated with any of the existing tracks, and tracks which existence is no longer confirmed by measurements are removed.
- Update - a step in which the state estimation of the relevant object is updated based on new measurements from a sensor associated with it in the Association step. One of the most popular approaches to tracking and updating the state of objects is the use of the Kalman Filter [104], or its various derivatives. Other popular types of methods use the Integrating Multiple Model algorithms for the state estimation task [44, 49, 43].

High-level fusion has several advantages: the data bandwidth required for sending object lists to a central computing component is low (compared to raw data), the entire architecture is relatively transparent, and methods for performing this type of fusion are well-researched and already established in the automotive industry. One significant disadvantage, however, is the data loss that occurs before the fusion itself.

2.2.2.2. Low-level Fusion

Low-level fusion, also referred to as *early fusion* is often proposed as an alternative to high-level methods. In this approach, the data is not significantly preprocessed before the fusion module, and the fusion methods vary greatly with the type of sensors. Due to the higher bandwidth and general difficulty of this type of fusion, it is relatively less frequently utilized in commercial systems.

Recent developments in the ML area, however, enable easier preprocessing of raw sensor data for low-level fusion applications that often utilize deep neural networks for the task of fusion itself as well. An example of such a system is presented in Fig. 2.9.

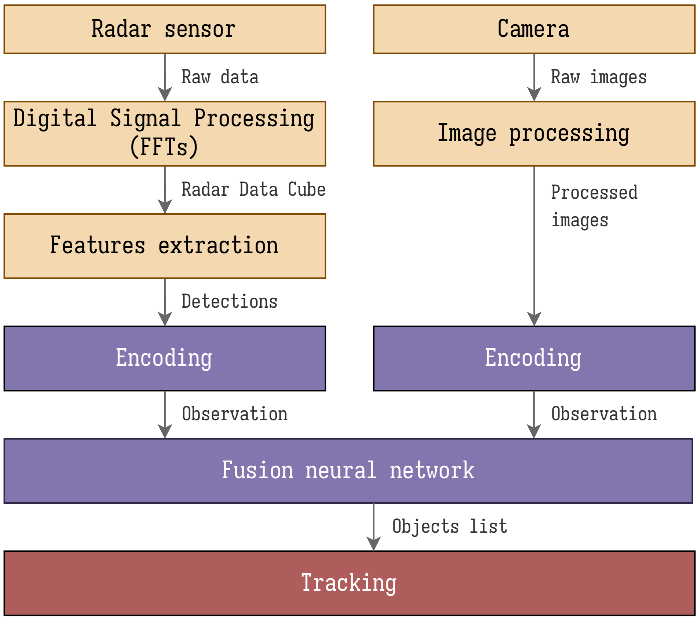


Figure 2.9. Example of a low-level sensor fusion architecture. Yellow blocks indicate sensor-specific initial processing steps, blue - parts of the ML-based fusion module, and red - the tracking module.

A similar architecture has been proposed in [79], where the authors used a modified VGG16 [160] model with a feature pyramid network [107] to preprocess a radar range-azimuth heatmap and a monocular camera image into feature maps further processed by a detection network. Another similar approach has been presented by Chadwick et al. [24], with radar detections preprocessed to a form of range and range rate image channels. Camera images and radar channels are preprocessed separately by convolutional networks and ResNet blocks [54], and then passed to a detection network based on the SSD architecture [111].

Other approaches to low-level radar-camera fusion in which radar detections are preprocessed to the form of an image-like matrix that can be further processed by a deep neural network similarly to the camera input were presented in [69] and [25].

2.3. Autonomous Driving System Architecture

AD systems vary significantly in terms of the utilized sensors, software modules, algorithms used, and general architecture. There are, however, certain elements of the system architecture that are relatively commonly utilized in such systems. A high-level example of a minimal AD system that utilizes them is presented in Fig. 2.10.

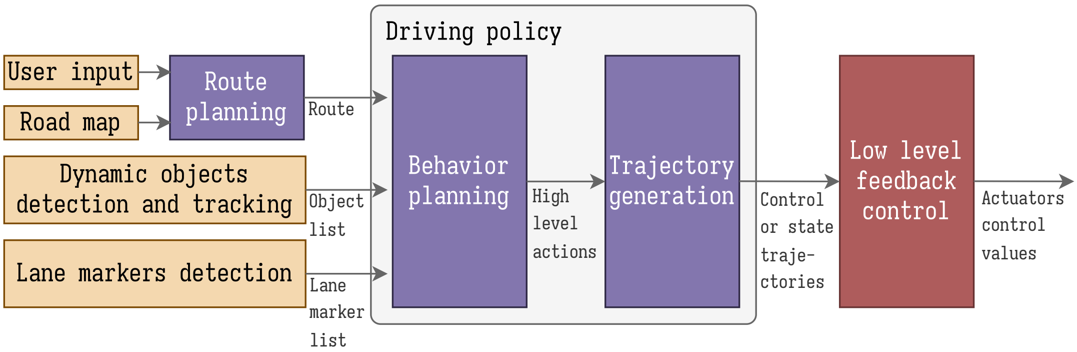


Figure 2.10. Example of a high-level architecture of an AD system. Yellow indicates algorithms' inputs, blue - planning modules, and red - low level control module.

The element of the system that will be at the core of this thesis' considerations is the *driving policy*, also referred to as *AD policy*, or simply *policy* for the remainder of this thesis.

For the purpose of this thesis, I define a driving policy as a software module that maps a certain representation of the ego's environment to control values or control trajectories that can be used to control the vehicle's motion.

The most common inputs used by most driving policies are listed below.

Dynamic objects list, which describes the current (and possibly past) states of the dynamic objects in the vicinity of the ego vehicle. The objects may include other vehicles, bicycles, pedestrians, animals, etc. Description of the objects' state is typically limited to their simplified geometry (for instance the bounding boxes), class of the object (examples of the object classes are: vehicle, motorcycle, pedestrian, etc.), its position, velocity, acceleration, yaw angle, and yaw rate. To ensure accurate estimations of objects' states, the object list is typically created using tracing and fusion algorithms based on data from multiple sensors [119].

Static environment description, which provides information about the geometry of the road and static obstacles. The source of static environment description varies significantly - the simplest systems typically utilize lane markers geometry estimates provided by a front-facing automotive camera, but there are many other approaches that utilize a precise

LiDAR-based model of the environment, High-Definition maps, the output of Simultaneous Localization and Mapping algorithms that utilize various sensors to create a precise description of the environment or even raw camera streams [130].

Ego state, including its current velocity, steering angle, and a variety of other sensor readings related to the state of the vehicle and its internal components (gear, power mode, error codes, etc.).

Depending on the level of autonomy of AD systems, additional information may also be required. Higher levels of autonomy, for instance, require a route planning module that would prepare a high-level driving plan based on a road map and the desired driving destination selected by the user. Other relevant information may include local speed limits, traffic sign recognition outputs, information about local traffic laws, information about the user's driving preferences, or the Driver Monitoring Systems (DMS) outputs.

The output of the driving policy also vary between different systems. Simple experimental setups may utilize direct control signals utilized by vehicle actuators, such as the desired steering angle, the desired steering angle rate, and the desired throttle position/braking force. More commonly, the driving policy outputs a desired trajectory that is followed using low-level feedback control methods.

The driving policy itself is often split into two modules, a high-level behavior planner, that provides information about a desired maneuver to be executed (e.g., following the vehicle in front or changing the lane), and a trajectory generation module, that executes the desired maneuver in a safe and efficient manner [157].

Splitting the driving policy into these two modules offers several benefits. Behavior planning typically requires complex reasoning regarding the behavior of other vehicles, long-term tactical planning, and general road situation. The use of Machine Learning (ML) based methods to decide on a high-level maneuver to be executed thus may be a desirable solution. The execution of the maneuver in a safe and efficient manner can be carried out with the use of more transparent and well-researched trajectory planning methods, such as optimization-based trajectory generation [47]. This allows for leveraging the advantages of both ML-based and classical planning methods.

An alternative approach that currently gains certain attention in the context of the rapid development of ML techniques is the end-to-end control approach, in which the driving policy is implemented as a single neural network. This approach, although less transparent, has certain advantages as well, allowing to leverage of reasoning skills of ML-based driving policies to plan precise movements of the vehicle in a way that enables the execution of more complex short-term strategies [170].

Note, that an in-depth review of planning methods that can be used in the trajectory generation module is provided in Section 3.1.1, and an introduction to Reinforcement Learning methods that can perform the behavior planning task is presented in the next section (2.4).

2.4. Reinforcement Learning

Public roads constitute a complex dynamic environment, and efficiently navigating it requires considerable planning, prediction, and scene understanding skills. Taking into account all important details regarding the environment and complex interactions between road users is difficult in the case of deterministic rules-based algorithms. For this reason, methods based on Reinforcement Learning (RL) techniques are often proposed for the execution of the behavior planning task [204].

RL-based driving policies are trained in a simulation environment, in which the trained policy is used to explore a variety of plausible scenarios and gather experience regarding the effects of the ego's actions on other road users.

For the purpose of the definition of the RL-based driving policy, the environment ϵ with which the policy interacts is defined as a Markov Decision Process (MDP).

MDP definition utilizes the components listed below.

$\mathcal{S}$ - *state space*, set of states, including both environment's and agent's (AD system's) states. Each state constitutes a complete description of the environment needed to determine its next state after performing certain actions (in the case of the AD application, that is, states of vehicles, description of static environment, internal states of the simulation, and relevant algorithms).

$\mathcal{A}$ - *action space*, or a set of actions that the policy can choose. In the case of the driving policy, action space often constitutes possible combinations of control values (in the case of end-to-end driving policies) or available maneuvers (in the case of RL-based behavior planning modules).

$P_a(\sigma, \sigma') = Pr(\sigma_{t+1} = \sigma' | \sigma_t = \sigma, \alpha_t = \alpha)$ denotes the probability that performing an action $\alpha \in \mathcal{A}$ while being in a state $\sigma \in \mathcal{S}$ will result in a transition to a state $\sigma' \in \mathcal{S}$ at time $t+1$.

$R_a(\sigma, \sigma')$ is an expected immediate reward assigned to a transition from a state σ to σ' as a consequence of performing the action α.

A stochastic policy $\pi_\theta(\alpha|o)$ returns the probability of taking an action $\alpha \in \mathcal{A}$ to be performed in time t based on the observation of the environment o_t and is parameterized by a parameter vector θ.

2.4.1. Proximal Policy Optimization

The main goal of training in a Reinforcement Learning setup is to find values of the vector θ that parameterizes the policy (typically a neural network) that maximize the expected cumulative reward.

Proximal Policy Optimization (PPO) [154] is one of the most commonly used on-policy RL algorithms, frequently used in control-related tasks.

PPO interactively updates the parameter values based on the following equation:

$$\theta_{k+1} = \arg\max_{\theta} \underset{\theta \sim \pi_{\theta_k}}{\hat{\mathbb{E}}} \left[\mathcal{L}\left(\sigma, \alpha, \theta_k, \theta\right) \right], \tag{2.4.1}$$

with $\mathcal{L}\left(\sigma, \alpha, \theta_k, \theta\right)$ denoting a clipped loss function defined as follows:

$$\mathcal{L}\left(\sigma, \alpha, \theta_k, \theta\right) = \min\left(\frac{\pi_\theta(\alpha_t|\sigma_t)}{\pi_{\theta_k}(\alpha_t|\sigma_t)} \hat{A}^{\pi_{\theta_k}}(\sigma, \alpha), \text{clip}\left(\frac{\pi_\theta(\alpha_t|\sigma_t)}{\pi_{\theta_k}(\alpha_t|\sigma_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}^{\pi_{\theta_k}}(\sigma, \alpha) \right), \tag{2.4.2}$$

where $\pi_\theta(\alpha_t|\sigma_t)$ denotes the probability of action a_t under the new policy, $\pi_{\theta_k}(\alpha_t|\sigma_t)$ describes the action probability under the current policy, $\hat{A}^{\pi_{\theta_k}}(\sigma, \alpha)$ is the advantage estimate at the time t, and ϵ is a training hyperparameter.

In the PPO, estimation of the advantage $\hat{A}^{\pi_{\theta_k}}(\sigma, \alpha)$ is calculated using a Generalized Advantage Estimator (GAE) [153]:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \left(r_{t+l} + \gamma V_\phi(\sigma_{t+l+1}) - V_\phi(\sigma_{t+l}) \right), \tag{2.4.3}$$

r_t is the reward at time t, and $V\phi(\sigma)$ denotes the value function estimate, which is performed by a neural network (critic network) parameterized with a parameter vector ϕ. λ and γ denote calibratable parameters.

Advantage estimation utilizes a set $\mathcal{D}_k = \{\tau_i\}_{i=1..n_{\text{episodes}}}$ of n_{episodes} trajectory segments $\tau_i = \{\sigma_0, \alpha_0, r_1, \sigma_1, \alpha_1, r_2, \sigma_2, \alpha_2, ...\}$ collected at each training iteration k through observation of the policy's π_{θ_k} interaction with the environment. To increase the training speed, the trajectory segments are typically collected in a parallelized simulation setup.

2.5. Safety

Ensuring the safety of ADAS and AD systems is one of the most difficult and expensive aspects of their development. Due to the difficulty and complexity of this task, several standards were introduced to provide a structured approach to the development of safe systems.

2.5.1. Hardware and Software Faults

A wide set of standards and good practices for the development of ADAS features with low autonomy levels already exist. The most well-known standard in this area is perhaps the ISO 26262 [65] titled "Road Vehicles - Functional Safety" (often referred to as FuSa). The standard, widely used in the automotive industry, recommends practices for the development of vehicles and their components that are intended to address hazards related to malfunctions of electronic components and electrical systems in the designed products. The standard is primarily focused on assessing and reducing safety hazards and risks in the system.

The standard specifies a safety life cycle for developed products that divides the development process into a series of phases, with related activities that help to identify risks, define requirements and safety goals, and ensure their fulfillment.

The standard also describes a development process that helps create a safe product, proposes a way in which potential hazardous events can be classified, introduces methods to specify safety-related requirements, and ways to analyze and evaluate the system from a safety perspective.

One of the most important concepts introduced by the FuSa standard is the Automotive Safety Integrity Level (ASIL) analysis. In the initial phases of the safety life cycle, a hazard analysis is performed to identify potential hazardous events and assess related risks.

Identified hazardous events are classified according to the expected injuries that they may cause (*severity* classification), the likelihood of such event (*exposure* classification), and the likelihood of the driver being able to prevent an injury in the case of such event (*controllability* classification). Depending on the combination of these factors, a safety level is assigned, and the standard provides a set of guidelines to assign safety goals related to the analyzed event.

The use of the FuSa standard helps to evaluate and reduce risks related to electronic and software faults. Unfortunately in ADAS and AD systems, hazardous events may occur in the absence of such errors - e.g., as a consequence of the limited performance of the perception systems, incorrect predictions of other road users' behavior, or dangerous road situations.

2.5.2. Sensors Errors and Performance Limitations

Sensors used to determine the state of other vehicles are inherently inaccurate and various environmental factors can result in missed detections, false positive detection errors, and state estimation errors, as discussed in Section 2.2. Depending on the situation on the road and the error itself, this may result in severely hazardous events.

In order to address safety hazards in the absence of hardware and software faults, an ISO/PAS 21448 [66] standard has been introduced, titled Safety Of The Intended Functionality (SOTIF). The SOTIF focuses on relevant use cases, i.e., situations in which the system may operate. The standard distinguishes between four types of use cases: known safe, known unsafe, unknown unsafe, and unknown safe. Its main goal is to provide a set of guidelines that may help to increase the number of known scenarios and reduce the number of unsafe use cases.

The standard proposes a general set of activities for the improvement of intended functionalities to ensure their safety. Activities start with functional and system specification, hazard identification, and risk evaluation, needed to identify potentially hazardous events. If the events may lead to harm, their triggering events (e.g. false negative detection errors) are analyzed. If the triggering events are considered unacceptable, the functional and system specification is modified appropriately; otherwise a verification and validation strategy is defined to ensure that the known scenarios are sufficiently covered, the system behaves as expected, and the system does not cause an unreasonable risk in such scenarios [165]. If any of these conditions are not met, the functional and system specifications are modified until the residual risk is considered acceptable.

The SOTIF standard can be useful in the context of ADAS to address the issues related to limited sensor performance and potential sensing errors. Unfortunately, the number of possible road situations and their combinations with sensing error patterns is practically infinite, and addressing all plausible situations in a scenario-based manner remains a very difficult task.

Furthermore, the SOTIF focuses mainly on foreseeable misuse of the system and performance limitations of the system, e.g., related to perception system abilities. One can easily imagine a dangerous situation that is not related to these conditions; for instance, a vehicle stopped in a traffic jam can be hit from behind by another road user.

2.5.3. Nominal Safety

Scenarios, in which the driver of a vehicle involved in a dangerous situation cannot do anything to prevent this situation or a subsequent collision are relatively common. A described scenario in which a vehicle stopped behind other road users (e.g., waiting for a green light at an intersection or in a traffic jam) is the subject of a rear-end collision is a good example of such a situation. Rear-end crashes are, in fact, the most common type of collisions in the United States, constituting 29% of reported traffic accidents [123], and in 80-90% of such cases, the front vehicle is, in fact, stationary according to naturalistic driving studies [99, 13]. Since the front vehicle is usually stopped due to other vehicles in front of it or a front cross-traffic [13], it cannot perform any maneuver that could prevent the accident.

It is unclear how to set safety goals for AD vehicles considering these facts. Many sources suggest setting goals for such systems in terms of accident or fatality rates, where often a fraction of the human driver's fatality rate is suggested as a goal [41, 73, 156]. However, as a very significant subset of collisions is unpreventable as the discussed rear-end collision case suggests, such safety goals may be unfeasible as long as human-driven vehicles constitute the majority of road traffic.

However, simply setting safety goals based on accident or fatality rates close to those observed in the case of human drivers is not a socially acceptable solution - as analysis of a case of a fatal accident involving the Uber autonomous vehicle in 2018 suggests [138]. Despite the presence of a human safety driver in the company's AD cars and the overall lower accident rate involving AD vehicles [147] compared to human-driven ones, the lack of a proper reaction of the vehicle to a dangerous situation led to societal outrage and eventually resulted in Uber resigning from the internal development of AD technology [118].

The main conclusion that can be drawn from the above consideration is the fact that setting safety goals for an AD system with higher levels of autonomy is a very difficult task, as the context of dangerous situations plays a significant role in societal acceptance, and avoiding collisions altogether is infeasible, as driving in realistic traffic is always burdened with a relatively significant collision risk.

In approaching the problem of setting safety goals, a definition of safety proposed in ISO 26262 may be considered - the standard defines safety as "the absence of unreasonable risk", where the unreasonable risk is "a risk judged to be unacceptable in a certain context according to valid societal moral complex".

Several influential works that attempt to formally define acceptable risk in the context of AD systems have been published in recent years.

Shalev-Shwartz et al. proposed a safety framework based on a notion of responsibility, named "Responsibility-Sensitive Safety" (RSS) [157]. Authors of the framework note that all traffic participants follow certain rules - both written (e.g., traffic laws) and unwritten (e.g., avoiding close cut-in maneuvers or otherwise reckless but lawful driving). Human drivers routinely plan their movements based on the assumption that others will follow these rules.

The framework attempts to formalize these rules and define reasonable assumptions related to the worst-case behavior of other road users. With clear assumptions regarding reasonably worst-case behavior of other road users and accurate perception of the environment, one may plan a motion that guarantees a collision-free movement as long as all other vehicles in fact follow these rules. This allows us to define an acceptable residual risk as a risk of hazards related to other vehicles performing unreasonable and unsafe maneuvers. As long as the rules are reasonably defined and align with societal views on what constitutes safe driving, in all collision cases, the responsibility for the accident can be clearly assigned to the vehicle that failed to follow them, that is, was driving recklessly.

The considerations regarding what constitutes proper or safe behavior of a vehicle are often discussed with the assumption that even a completely functionally safe vehicle in the absence of internal faults or perception errors may make incorrect decisions. To distinguish these considerations from a Functional Safety domain, authors of the RSS framework propose to refer to them as a *Nominal safety* domain. To be precise, the authors state that "Nominal safety is the concern whether AV is making safe logical decisions assuming that the HW and SW systems are operating error-free (i.e., are functionally safe)" [157], with AV denoting the Autonomous Vehicle, HW - hardware, and SW - software.

Similar concepts were used in the definition of a "Safety Force Field" safety framework proposed by Nvidia [128]. The framework also focuses on the definition of reasonable assumptions about the worst-case behavior of other road users and demonstrates how constraints for driving policies can be derived from them.

Research and discussions on the topic of nominal safety and formalization of reasonable expectations regarding the behavior of other road users led to the creation of a recently published IEEE 2846-2022 standard (IEEE Standard for Assumptions in Safety-Related Models for Automated Driving System) [63]. The standard introduces a minimum set of assumptions about reasonably foreseeable behaviors of other road users that can be used for setting the requirements, test scenarios, and safety goals for the AD systems. The standard proposes to formally define, e.g., expectations regarding reaction times of other road users or maximum braking deceleration that they are expected to exhibit in various situations. The expectations can be defined using a series of parameters that describe them numerically (e.g., acceleration value or response time value), but the standard does not define the values themselves. The authors of the standard provide, however, an extensive literature review that can be used as supplementary material for setting these values.

It should be noted that despite active research efforts and the multitude of proposals regarding the design and safety of driving policies, nominal safety remains a relatively new research area, and there is no simple consensus regarding the development and evaluation of highly autonomous AD systems.

3. Multiple Hypothesis Trajectory Planning

Motion planning for Autonomous Driving (AD) applications is a very difficult task, mainly due to limited information about both the present and future state of the vehicle's surroundings.

It is rarely possible to predict the future behaviors of other road users in a precise way. In many situations, more than one plausible hypothesis regarding future behavior can be formulated, posing a significant challenge to the planning algorithms. Dangerous situations on the road are particularly problematic; drivers often react to atypical events in an unpredictable way. As an example, typical reactions to the sudden appearance of an obstacle (e.g., a wild animal jumping onto the road in front of the vehicle) may include braking or steering maneuvers, and it is rarely clear which one will be used by other drivers. To ensure safety in such a situation, planning the emergency maneuvers of an AD vehicle would have to take into account both possibilities.

Unfortunately, this is not the only uncertainty that has a significant impact on planning algorithms. While vehicles with AD capabilities are typically equipped with a powerful set of sensors, such as radars and cameras, all perception systems have limitations, both in terms of performance and detection range. Furthermore, significant parts of the environment can be occluded for the sensors by other road users, elements of the road infrastructure, as well as vegetation or buildings in proximity of the road, which are particularly problematic at road intersections.

In this chapter, I present a vehicle motion planning method designed to plan a safe motion of an AD vehicle, taking into account various uncertainties related to the future state of other road users. In particular, the presented method can be used to plan a motion taking into account two or more hypotheses related to the future state of the vehicle surroundings, as well as to implement fail-safe planning, in which worst-case assumptions regarding the behavior of other road users are taken into account.

3.1. Introduction and Motivation

Predicting the motion of other road users and planning the motion of a vehicle are both large and active research areas. While many approaches have already been proposed in both fields, they remain very difficult problems, with new methods constantly being developed. It should be noted that predicting future trajectories of other road users is outside of the scope of this thesis; nevertheless, the proposed method may be used in conjunction with various prediction approaches, and thus I briefly review existing prediction and planning methods.

3.1.1. Motion Planning

Motion planning is one of the fundamental problems in robotics, and a large number of efficient methods have already been proposed to control wheeled robots that navigate static environments [191, 197]. Successful use of various search algorithms, such as A* [53], D* [166], or Rapidly-exploring random trees (RRT) [96], has been demonstrated numerous times in the context of wheeled robots path planning.

A major obstacle in the application of such algorithms to the problem of vehicle motion planning is the fact that the environment in which an AD vehicle has to operate is usually only partially observable, dynamic, and uncertain in terms of both current and future state. Navigation in such environments requires a deep understanding and consideration of the vehicle's surroundings, interactions between road users, and possible future movements of other vehicles in the proximity of the ego vehicle.

One of the methods that has been shown to achieve satisfactory performance in the task of motion planning in such environments is Reinforcement Learning (RL) [132]. RL-based methods are often used for AD applications, as RL-based policies are known to produce complex behaviors, demonstrating skills such as long-term strategic planning, behavior prediction, exploration, and navigation in partially observable environments [12].

One significant downside of RL-based driving policies is the lack of transparency, which is especially problematic in the context of proving the safety of planned behaviors. To address this issue, various methods were proposed to constrain the RL policies [15, 113]. Still, to ensure robustness in particularly difficult and severe situations, Machine Learning (ML) methods are often combined with more transparent methods, such as Model Predictive Control (MPC), or search algorithms [83]. This combination is often performed by splitting the planning task into two subtasks: high-level decision making and low-level trajectory generation [95, 132]. In such a setup, an ML-based policy typically is responsible for choosing a high-level maneuver to be performed, and another method, such as MPC is used to plan the trajectory for a safe execution of this maneuver. Other ways to combine ML-based planning with a more transparent method may include using ML for typical situations and switching to another method when the situation is considered uncertain or dangerous [158, 157, 83].

The use of non-ML algorithms, such as MPC or RTT, for planning in difficult situations provides transparency beneficial from the perspective of ADAD/AD system's safety analysis, but comes with a significant downside: most of such classic approaches require explicit constraints that would take into account plausible future behaviors of other road users. The generation of such constraints is typically based on various behavior and trajectory prediction algorithms, increasing the complexity of the entire system.

However, while systems based solely on ML methods may be able to account for possible behaviors of other road users in an implicit way, the safety of such systems is difficult to assess and prove. Utilization of a separate prediction and/or constraints generation module increases the transparency of the system's architecture and helps to evaluate the expected performance of the entire system.

In the next subsections, I will describe existing prediction approaches that could be used for the constraints generation for planning purposes and analyze existing planning solutions.

3.1.2. Prediction Methods

Vehicle behavior prediction modules are most commonly used to provide input to various Autonomous Driving and Advanced Driver Assistance Systems (ADAS) features, such as Autonomous Emergency Braking (AEB) or Adaptive Cruise Control (ACC). In the case of the planning task, information about the plausible future behavior of other road users can be used to generate constraints for the planning algorithms, serve as an auxiliary input to ML-based driving policies, or be utilized to validate generated trajectories and detect dangerous situations.

In the literature, many prediction approaches have been proposed, varying in both their interfaces and used methods. Existing methods can be categorized in several ways - survey [121] presents three of the most common categorization approaches, classifying deep learning prediction methods based on their features listed below.

- **Input representation.** The input type has a significant impact on the complexity and performance of the prediction. While it is common to use inputs that include information about road infrastructure and some representation of other road users, the way these are represented can vary significantly. Following the categorization used in [121], one can specify a few main input representations: state history of one or more road users in proximity of the ego vehicle, bird-eye view rendering of the scene, and raw sensor data input. Choice of the input type impacts the architecture of the module - for instance, bird-eye-view renderings are most commonly processed using convolutional neural networks, while objects' state histories will likely utilize other methods, such as neural networks with inputs based on transformer encoders.

- **Output type.** The type of output information is a very natural way to categorize prediction methods, as it is one of the main factors in choosing the prediction method for planning applications. The authors of [121] specify four main output types: Maneuver Intention, Unimodal Trajectory, Multimodal Trajectory, and Occupancy Map. A similar distinction was proposed in [20], where prediction methods were divided into categories such as single future behavior, countable and uncountable sets of future behaviors, and probability distribution of future behaviors. In a more general way, prediction methods can be categorized on the basis of a prediction horizon (e.g., into short-term and long-term prediction).

- **Algorithms utilized for prediction.** Categorizing approaches based on the utilized methodology itself is another natural way to present existing approaches. The mentioned survey [121] focuses mainly on ML-based methods, categorizing them according to the type of neural network utilized for the prediction task, but other categorizations are also used, e.g.,[100] splits prediction approaches into physics-based, maneuver-based, and interaction-aware. This categorization is also related to the prediction horizon - with simple physics-based methods being suitable for short-term predictions and long-term trajectory forecasts requiring approaches aware of interactions between road users.

From the context of the planning method that will utilize a behavior prediction, the output structure of the prediction module will have the largest impact on the planning approach, and thus I will focus on the output-type-based categorization, with additional remarks on the input type as well as utilized methods.

A minimal input to the prediction module consists of a state estimation of one or more vehicles in the proximity of the ego vehicle. While the information provided in the state is typically very limited, for example, to the position orientation and velocity of a vehicle, it still can be used to generate meaningful short-term predictions. This type of prediction is commonly used in vehicle tracking and sensor fusion algorithms [150].

Accurate predictions in a longer time horizon require additional information about the context, both in terms of the state of surrounding vehicles, as well as lane markers and/or road geometry. Information about the geometry of the road is often provided in the form of a two-dimensional map, for example, in [133] a 2D binary mask is used as input to the convolutional layers of the prediction network. The authors of [148] also utilized grid-like inputs representing a bird-eye view of the ego's surroundings but with additional information about past trajectories of other vehicles and multiple channels used for encoding various features of the static environment. This type of input is often preprocessed using specialized neural networks or convolutional layers - e.g., in [46] the authors proposed providing the context description in a form of latent encoding - a network based on GAN architecture is trained alongside the pedestrian's trajectory prediction network to encode the description of the environment.

Although input type and processing techniques vary significantly between methods, we can categorize prediction approaches on the basis of the output they produce with relative ease. Since the context for introducing prediction methods is their use for planning constraints generation, I will assign them to five categories: short-term prediction, maneuver recognition, unimodal trajectory prediction and multimodal prediction, and worst-case prediction.

- **Short-term prediction** finds many applications in the ADAS and AD systems. One of the most common ones can be found in sensor fusion and object tracking algorithms, which often utilize variations of the Kalman Filter [115], which uses a short-term prediction step [151] to update estimations of objects' states.

  Short-term prediction is often based on simple mathematical models that assume straight motion of the vehicles (e.g. Constant Velocity and Constant Acceleration models), or a constant turn rate (Constant Turn Rate and Velocity, Constant Turn Rate and Acceleration models) [71, 143]. The idea of using simple models with constant control inputs can also be extended to take into account the geometry of vehicles, for example, by using a bicycle kinematic model [142], which allows the motion of a vehicle to be simulated based on its steering angle and velocity. The bicycle model can be used for a short-term prediction using assumptions regarding its control values, examples of which are Constant Steering Angle and Velocity or Constant Steering Angle and Acceleration models.

  In certain applications, the use of more advanced dynamic models may be beneficial for the prediction problem. Dynamic models were, for example, used in threat assessment applications [19, 38], as well as in planning [139].

  As the models used for short-term prediction typically do not take the road geometry nor interaction with other road users into account for motion calculation, their application to path planning problems is, however, limited, as they may fail to provide an accurate prediction in complex road situations.

- **Maneuver recognition**. The behavior of road users, especially vehicle drivers, typically consists of distinct maneuvers, such as lane changes, overtaking maneuvers, braking, or lane following. Since maneuvers are performed to fulfill certain intentions (e.g., changing the lane), one can often assume that the vehicle driver that started performing a maneuver will continue executing a predictable series of operations typical for this maneuver. For this reason, recognition of a currently executed maneuver as well as early detection of an intention to perform a given maneuver can be used to relatively reliably predict future trajectories of other vehicles.

  Various maneuver recognition algorithms were proposed in the literature, including rule-based heuristics [48], Support Vector Machines [93], and Convolutional Neural Networks [97]. While the classification of other vehicles' maneuvers may be useful for Adaptive Cruise Control (ACC) and Autonomous Emergency Braking (AEB) algorithms, lack of the details about their expected or plausible trajectories limits their usage in trajectory planning algorithms.

- **Unimodal trajectory prediction** can be used to provide a single trajectory that is deemed the most plausible for a given road user.

Unimodal trajectory prediction methods provide clear information about the most plausible trajectory, making them easy to use in planning systems, but fail to capture less plausible behaviors of other road users, potentially making the planning methods based on such predictions prone to dangerous behaviors if a road user executes a less likely maneuver.

- **Multimodal prediction** represents the distribution of plausible behaviors of other road users, either as a list of possible trajectories and their probabilities, the distribution of plausible maneuvers, or plausible future states as a function of time, for example, in the form of an occupancy grid.

  Multimodal prediction approaches are typically based on ML methods, with a particular prevalence of Long-Short Term Memory (LSTM) [58] cells use, especially for a longer-term prediction. One of the LSTM-based approaches has been presented in [32], where the authors prepare plausible maneuver sequences and use a neural network with LSTM cells to estimate the probability of the occurrence of each maneuver. The authors of [77] demonstrated the use of a Convolutional Neural Network (CNN) for multimodal prediction in the absence of a detailed map of the environment, using only LIDAR-based perception of a scene for this task.

  Lee et al. [98] utilize a Conditional Variational Auto-Encoder (CVAE) [80, 81] to generate multiple samples of plausible trajectories, which are ranked during the training by a separate recurrent neural network. CVAE is also utilized in [189], where future trajectories of pixels in arbitrary videos are predicted from a single image.

  Non-recurrent convolutional neural networks are also commonly used for the task of multimodal prediction. As an example, in [30] authors encoded the scene context in the form of a multi-channel image and used a CNN to predict multiple plausible trajectories and estimate their respective probabilities.

  For many motion planning approaches, including one presented in this chapter, representing plausible future positions of other road users in the form of an occupancy grid may constitute a more convenient interface. One of the exemplary approaches that uses grid-based output was presented in [148], where the authors used a rasterized scene representation with a recurrent convolutional neural network to predict the motion of other vehicles, producing a grid with occupancy probabilities.

3.1.3. Motivation and General Idea

Planning a vehicle motion in a traffic environment inherently requires taking into account possible future behaviors of other road users, either implicitly, as in Reinforcement-Learning-based planning, or explicitly, e.g., using the trajectory prediction as input. The behavior of other road users rarely can be predicted with perfect accuracy - especially since typically more than one possible behavior hypothesis can be formulated.

Various planning methods often approach this problem using only the most plausible hypothesis as input. This approach, however, limits and requires adding conservative safety precautions to ensure that accidents can be avoided if another road user will perform unexpected maneuvers.

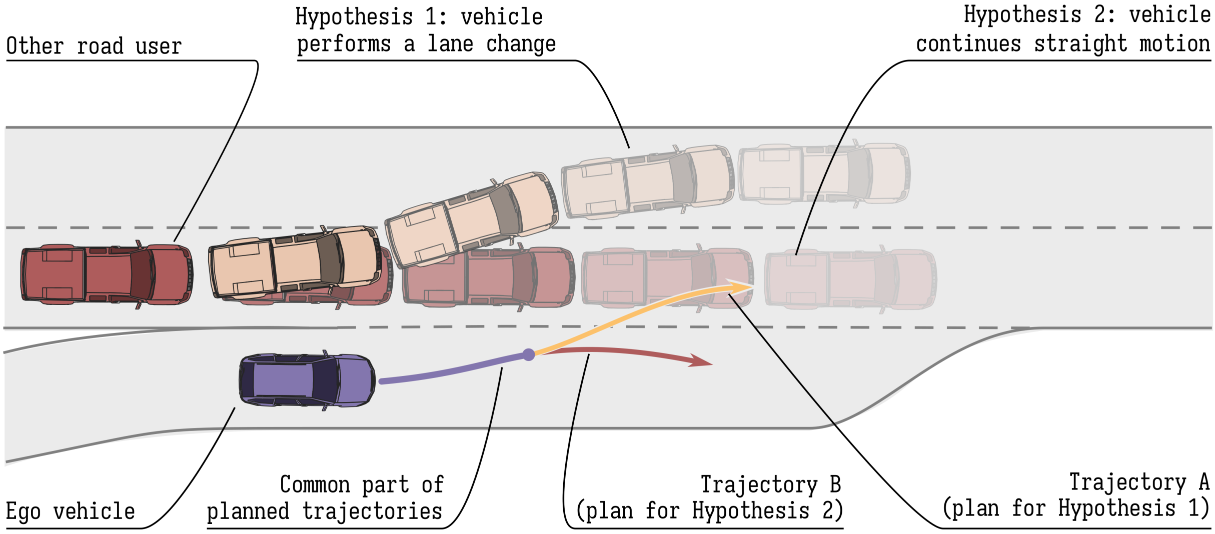

Other road user
Hypothesis 1: vehicle performs a lane change
Hypothesis 2: vehicle continues straight motion
Ego vehicle
Common part of planned trajectories
Trajectory B (plan for Hypothesis 2)
Trajectory A (plan for Hypothesis 1)

Figure 3.1. General idea of the multiple hypotheses planning algorithm. Multiple trajectories are planned simultaneously based on different hypotheses. The problem is formulated in a way that enforces that the trajectories remain identical for a certain duration.

The Multiple Hypothesis Planning algorithms described in this chapter are intended to alleviate these issues, allowing to plan safe and efficient trajectories in situations, where multiple plausible hypotheses regarding the behavior of other road users can be formulated. The algorithm does so by planning multiple trajectories based on different hypotheses simultaneously. Planned trajectories are constrained to overlap in a certain initial time horizon, allowing one to switch between executed trajectories depending on future observations performed during this time horizon.

An example of a possible application of the proposed algorithm is presented in Fig. 3.1, where the ego vehicle (that will execute the planned trajectories) merges into the traffic on a highway. Another road user is expected to change the lane to the left based on observation (e.g., it may be expressing this intention with indicator lights). Two main hypotheses can be formulated with regard to its near future behavior: either it will change the lane successfully, enabling the ego to safely merge into the traffic (Hypothesis 1), or it will fail to do so, e.g., due to an unsafe situation on the adjacent lane (Hypothesis 2). Based on these two hypotheses, two trajectories are planned: one in which Hypothesis 1 is assumed to be true, and the ego can perform the lane change (Trajectory A), and one in which the other hypothesis is taken into account and the ego performs braking, staying on its current lane (Trajectory B). Trajectories overlap in a certain time horizon, allowing for a postponed choice between trajectories to be executed until more information is gathered.

Note that the proposed trajectory generation method can be used to formulate a control scheme in which the trajectories are re-planned after execution of the common part of trajectories, potentially taking into account new or updated hypotheses.

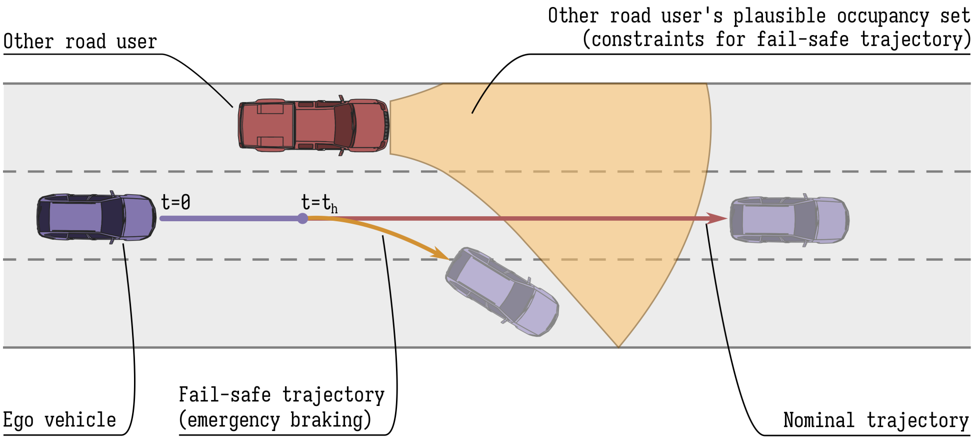


Figure 3.2. Application of multiple hypotheses planning to a Fail Safe Planning problem. Ego vehicle follows a nominal trajectory, that assumes the most plausible behavior of other road users. At the same time, a fail-safe trajectory is planned based on a worst-case hypothesis regarding the behavior of another vehicle. Replanning in periods shorter than t_h guarantees the existence of a collision-free trajectory under worst-case assumptions, as previously planned fail-safe trajectory can always be executed to avoid accidents.

One of the important applications of the proposed method is Fail-Safe Motion Planning [116]. In this control scheme, one trajectory, denoted the nominal trajectory, is planned based on the most plausible behavior of other road users. At the same time, another trajectory (a fail-safe trajectory) that branches from the nominal trajectory at a certain time $t = t_h$ is planned based on certain worst-case assumptions regarding the future behavior of other road users. The fail-safe trajectory utilizes spatio-temporal constraints that can be created based on all possible behaviors of other road users in such a way, that no overlap exists between the position of ego and the other vehicle, no matter what behavior will be executed by the other road user.

Fail-safe motion planning can be executed in a loop, where trajectories are re-planned during the period $t \in [0; t_h]$, ensuring a fail-safe trajectory always exists. As long as the behavior of other road users is within the derived worst-case constraints, a previously planned fail-safe trajectory can always be executed to avoid accidents.

The planning method proposed in this work extends existing fail-safe planning methods in a number of ways, as described in 3.1.4. Most notably, the simultaneous generation of nominal and fail-safe trajectories allows finding more conservative nominal trajectories in dangerous situations to enable the generation of fail-safe ones, instead of enforcing overly cautious emergency execution of a previously planned fail-safe trajectory.

3.1.4. Contributions

While a wide selection of planning methods has already been proposed for applications in ADAS and AD systems, finding trajectories that would produce reliable behaviors in potentially difficult situations taking into account various hypotheses regarding the state of the environment and the future behavior of other road users remains a difficult problem. While frameworks such as Fail-Safe Motion Planning [20] provide a transparent way of planning with safety guarantees, further work is needed to extend their usefulness, allowing the generation of versatile nominal trajectories that consider possible risks, generation of trajectories in situations where multiple hypotheses with similar plausibility must be considered, and providing a convenient way of tuning comfort/safety trade-offs in generated motions.

Taking into account the described limitations of existing methods, the main contributions of the approaches presented in this chapter are listed below.

- Proposal of the planning architecture, capable of taking into account various hypotheses regarding future behaviors of other road users, as well as the current state of the environment surrounding the ego vehicle. The proposed method generates multiple trajectories that overlap in a defined initial time period, allowing us to gather additional information before adhering to a trajectory based on one of the plausible hypotheses.
- One of the possible applications of the proposed method is fail-safe planning, in which a nominal trajectory is planned based on a most plausible hypothesis, and a fail-safe trajectory is planned based on certain worst-case assumptions. In contrast to existing methods, the proposed method applied to the fail-safe planning problem is capable of producing a nominal trajectory while taking into account the need for the generation of fail-safe trajectories as well. As a result, generated nominal trajectories naturally display a notion of cautiousness, i.e. they are altered compared to the nominal trajectory optimal in terms of comfort and/or efficiency in such a way, that enables safe execution of an emergency maneuver if a worst-case hypothesis turns out to be true. This approach extends to several situations in which fail-safe planning can be applied without unnecessary execution of emergency maneuvers.
- Contrary to the existing methods that plan trajectories related to different hypotheses either independently from each other or sequentially one after another, trajectories planned using the proposed method are planned simultaneously, affecting each other. While each trajectory fulfills different goals and/or utilizes different constraints, they all must overlap in a certain time horizon and different comfort/efficiency weights can be assigned to particular trajectories in order to fulfill this objective. For instance, one can decide to sacrifice the efficiency of a trajectory based on a least plausible hypothesis to ensure the high efficiency of the trajectory that is most likely to be executed.

Parts of the approach presented in this chapter are based on my previous work: the patent application [180] and the publication [181]. The approaches presented in these works were improved in a number of ways, including but not limited to new constraint formulation methods, application to situations with multiple plausible hypotheses, and new evaluation scenarios.

3.2. Problem Formulation

In this chapter, the Multiple Hypothesis Planning method is presented. The described method is intended to be used to generate control trajectories to be executed by a low-level vehicle control system that performs an autonomous driving task.

The method produces a set $\mathcal{T}_{\text{p}}$ of n_{traj} control trajectories, where each control trajectory $T_i(\mathbf{q}_i, t)$ for $i = i..n_{\text{traj}}$ is described with a k_{par}-dimensional vector $\mathbf{q}_i \in \mathbb{R}^{k_{\text{par}}}$ for and represents certain control values of the ego vehicle in a time frame $t \in [t_0, t_{\text{f}}]$ (for further considerations, it will be assumed that t_0 is equal to 0). Generated trajectories overlap in an initial period of time $t \in [0, t_{\text{h}}]$, where t_{h} is the arbitrarily chosen value $0 \leq t_{\text{h}} \leq t_{\text{f}}$. Due to this initial overlap, either trajectory can be safely executed by the ego vehicle during this time period.

3.2.1. Assumptions

Approaches presented in this chapter, unless otherwise stated, follow a set of general assumptions, listed below.

- The state of the ego vehicle is known, e.g., thanks to the availability of a precise host state estimation subsystem in the vehicle.
- An accurate state estimation of other vehicles is available in the system, e.g., as an output of a perception system with a set of precise sensors. Perception and tracking problems, as well as estimation of the perception systems' performance, are beyond the scope of this thesis, although certain aspects of the sensor modeling and the impact of perception errors on the planning systems are described in Chapter 4.
- Information about the road geometry is available, e.g., in the form of an HD map or a perception system that is capable of providing a precise estimation of the lane markers' geometry based on camera-based sensors.
- Ego vehicle is equipped with a low-level control system that is capable of precise execution of control commands e.g., using feedback control systems.
- Environmental conditions, in combination with control constraints used in the planning approach, do not have a significant impact on the execution of the planned trajectories, that is, constraints are conservative enough to guarantee the safe execution of the planned trajectories in all intended environmental conditions.

- Vertical road profiles do not have an impact on the execution of planned trajectories; i.e., vehicle performance is sufficient to ensure that the planned trajectories will be executed correctly on all roads the system is intended to be used in, independently of the road profile. With this assumption, the vertical profile of the road is neglected in the described planning method.

- The high-level route required to be taken by the ego vehicle is either known (e.g., as an output from a high-level route planning module) or irrelevant to the problem (e.g. when the method is used for highway drive assist applications, with the assumption that the ego should follow current road).

3.2.2. Static Environment

Planning a long-term trajectory of the vehicle usually requires knowledge of the geometry of the road and static obstacles that may limit the drivable area.

Various formats for representing road networks have already been proposed, most notably OpenStreetMap [50] commonly used in crowd-sourced map creation, and OpenDrive format [37] that enables the detailed representation of road features suitable for automotive applications.

To enable a transparent description of the method presented in this chapter, I introduce a set of definitions needed to represent a road network for planning purposes.

Lane fragment is an atomic section of a road represented by a *centerline* and *lane fragment boundaries.* Lane fragments may represent marked sections of the road, for example, a highway lane enclosed by lane markers or logical connections between road lanes, for example, geometry that encompasses the logical driving path at an intersection. *Centerline* of the lane fragment is a curve that represents a logical driving path related to this lane fragment. *Lane fragment boundaries* define an area relevant to a given lane fragment, for example, an area between the relevant lane markers.

Driving corridor is a road feature composed of a set of lane fragments that can be used by a vehicle to fulfill a given driving task (e.g., following the current lane and driving straight at an upcoming intersection, or taking a highway exit). *Corridor centerline* is a curve composed of centerlines of lane fragments that constitute the corridor, and *corridor area* is a sum of areas defined by its lane fragment boundaries. Note that a single-lane fragment may be used in several driving corridors.

Intended driving corridor is a driving corridor that is needed to perform current driving tasks.

Available driving area is a sum of corridor areas of the corridors that can be legally used by the ego vehicle in the driving task (including the intended driving corridor). Note that this may include corridors that are irrelevant for fulfilling the current driving tasks (but may be used, e.g., for the execution of emergency maneuvers).

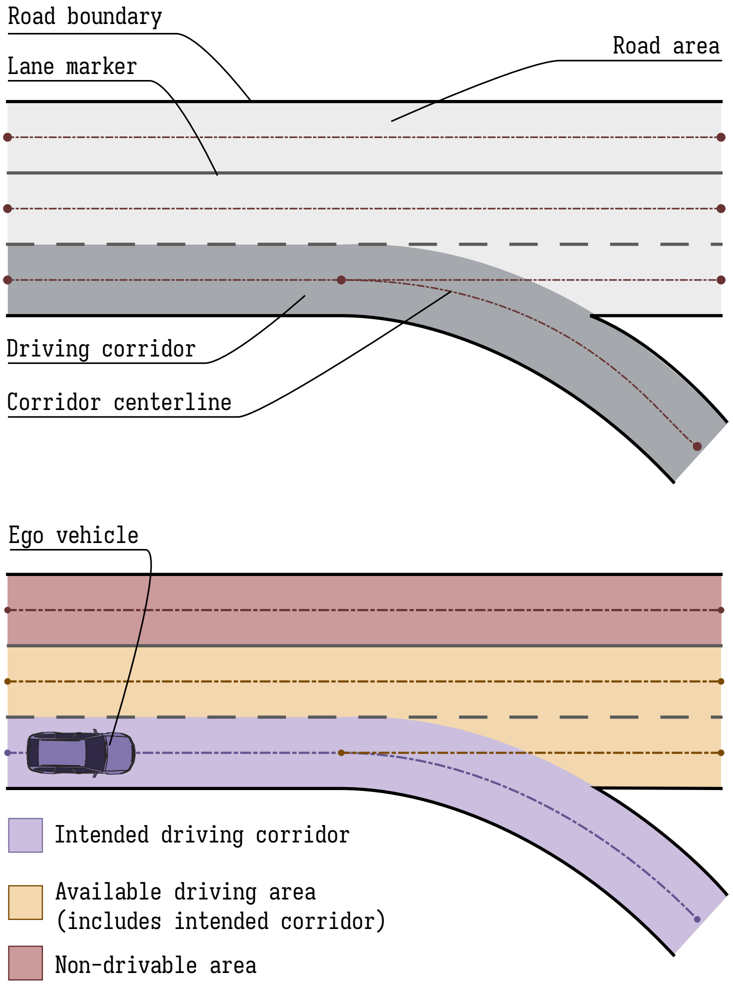


Figure 3.3. Road model used in this chapter. Road area is composed of possibly overlapping driving corridors, where each driving corridor represents a logical set of subsequent lane sections that can be used to fulfill a given driving task.

Non-drivable area is a sum of areas that cannot be legally accessed by the ego vehicle, e.g., an area of the opposite traffic lane behind a solid line. Non-drivable areas may also include areas of the ego surrounding that are physically inaccessible by the vehicle (e.g., areas occupied by static obstacles, and areas behind road barriers).

Road representation composed of the road features listed above can be created based on simple lane marker geometries (e.g., output from the camera-based lane markers detection system), or more complex maps, such as automotive grade high-definition maps [109].

Note that the definition of the intended driving corridor allows one to encode the current high-level driving task. The task may represent a simple intention constant for a given ADAS system, for example, the lane-following task of a lane-centering ADAS feature, as well as the output of advanced autonomous driving policies. The intended driving corridor can also be used to represent a desirable driving path chosen by a driving policy, as shown in Fig. 3.4, making

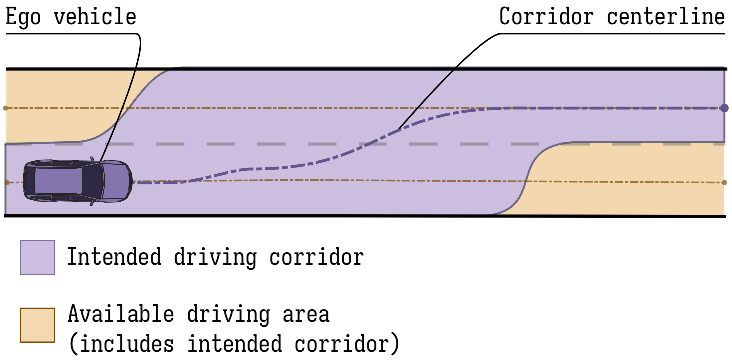


Figure 3.4. An example showing how the intended driving corridor can be used to specify a lane change maneuver. The corridor area specifies an area suitable for performing the maneuver, while the corridor centerline defines the desired lane change path.

the proposed representation a base for a viable interface between a high-level behavior planning module and the trajectory generation subsystem.

3.2.3. Dynamic Constraints

The main application of the proposed method is planning efficient ego control trajectories based on a set $\{H_i\}_{i=1..n_{\mathrm{traj}}}$ of n_{traj} hypotheses regarding the current and / or future states of the environment. Hypotheses may be formulated, for example, on the basis of the output of a prediction module or on the basis of an analysis of worst-case future behaviors of other road users. For further consideration, I will assume that the hypotheses also contain a description of the static environment relevant for planning. While in the cases considered in further sections the static environment is assumed to be known and identical in all hypotheses, proposed methods can be also used if this is not the case - e.g., when alternative hypotheses can be formed with regard to the static environment state based, e.g., on contradicting data from multiple redundant sensors.

Depending on the output interface of the prediction method, each hypothesis H_i for $i = 1..n_{\mathrm{traj}}$ can be expressed as, or used to derive, an occupancy set $\mathcal{R}_i(t)$ for $i = 1..n_{\mathrm{traj}}, t \in [0, t_f]$ that represents areas of the environment that should be avoided by the ego due to the possible presence of other vehicles in them according to this hypothesis. In other words, the occupancy set $\mathcal{R}_i(t)$ represents an area of the environment, entering which would be dangerous if hypothesis H_i proved true. The occupation sets can be used to derive state constraints used to plan each of the trajectories $T_i(\mathbf{q}_i, t)$. The set of all occupancy sets used for planning is denoted $\{\mathcal{R}_j\}_{j=1..n_{\mathrm{traj}}}$.

The operation of the $\{\mathcal{R}_j\}_{j=1..n_{\mathrm{traj}}}$ set generation based on the hypothesis set $\{H_i\}_{i=1..n_{\mathrm{traj}}}$ is denoted as a mapping:

$$M_{\mathrm{occupancy}} : \{H_i\}_{i=1..n_{\mathrm{traj}}} \rightarrow \{\mathcal{R}_j\}_{j=1..n_{\mathrm{traj}}}, \tag{3.2.1}$$

implementation of which depends on the type of available hypotheses.

Trajectory $T_i(\mathbf{q}_i, t)$ generated based on hypothesis H_i fulfills constraints defined by an occupancy set $\mathcal{R}_i(t)$ if the bounding box $\text{Box}(\mathbf{q}_i, t)$ of an ego vehicle controlled according to trajectory $T_i(\mathbf{q}_i, t)$ do not overlap with the constraint at any time, i.e.:

$$\mathcal{R}_i(t) \cap \text{Box}(\mathbf{q}_i, t) = \emptyset, \quad \forall t \in [0, t_\text{f}]. \tag{3.2.2}$$

3.2.4. Control Trajectories

The method generates a set $\mathcal{T}_\text{p}$ of n_traj control trajectories that describe desired control values $\mathbf{c}_i(t)$ for $i = 1..n_\text{traj}$, $t \in [0, t_\text{f}]$ as a function of time. The control vector $\mathbf{c}_i(t)$ related to the i-th hypothesis is defined as follows:

$$\mathbf{c}_i(t) = [\delta_i(t), a_i(t)]^T, \tag{3.2.3}$$

where $\delta_i(t)$ denotes the value of the vehicle's steering angle and $a_i(t)$ the value of the acceleration at time t according to the i-th trajectory.

Note that typically the desired control values described by a control trajectory $T_i(\mathbf{q}_i, t)$ are not used directly as input to vehicle actuators but serve as input to a low-level feedback controller that controls steering torque/rate and throttle/brake control values. As a result, the steering angle and acceleration $\mathbf{c}_i(t)$ achieved by the ego at time t are not necessarily equal to the desired control values $T_i(\mathbf{q}_i, t)$. Nevertheless, the problem of low-level control is beyond the scope of this thesis, and for the purpose of further considerations, the desired control values are assumed to be executed immediately, and thus $\mathbf{c}_i(t) = T_i(\mathbf{q}_i, t)\ \forall t \in [0, t_\text{f}]$.

3.2.5. Vehicle Model

The relation between the control trajectory and the state of a vehicle can be modeled as a differential equation of the following form:

$$\dot{\mathbf{s}}(t) = f\left(\mathbf{s}(t), \mathbf{c}(t), \mathbf{p}_\text{veh}\right), \tag{3.2.4}$$

where $\mathbf{s}(t)$ is a state of the vehicle at a time t, $\mathbf{c}(t)$ denotes the control vector, and $\mathbf{p}_\text{veh}$ the vector of vehicle's parameters, such as wheelbase, length and width (defined in further parts of this section).

State vector $\mathbf{s}(t)$ utilized in the approach described in this chapter is defined as:

$$\mathbf{s} = [x, y, \psi, v]^T, \tag{3.2.5}$$

where x and y describe the position of the vehicle's Center of Mass (CoM) in a Cartesian coordinates system (relative to a certain stationary reference frame (X, Y)), ψ denotes the vehicle's orientation in this system, and v its absolute speed.

A wide selection of dynamic and kinematic models was proposed for solving optimization-based trajectory generation problems and Model Predictive Control (MPC) schemes. The choice between lower-fidelity kinematic models and more accurate dynamic ones remains relatively difficult in the design of such algorithms, with a number of trade-offs to be considered. Kinematic models tend to be less computationally expensive but may fail to capture complex phenomena related to tire friction forces. Dynamic models, while more accurate, often struggle with accuracy and robustness at low speeds, e.g., tire force models often estimate tire slip angle using a velocity in the denominator, leading to singularities in stop-and-go scenarios.

To provide insight into the practical differences between these models, the authors of [85] compared the performance of dynamic and static models in a set of experiments, including test drives in the vehicle controlled by the MPC control scheme with both types of models. Their conclusions suggest that the use of the kinematic model does not impact the algorithm's performance in a significant manner while maintaining a significantly lower computation overhead. Taking these results into account, I decided to use a kinematic vehicle model to be used in the design of the described method. In particular, a commonly used Bicycle Kinematic Model [141] will be utilized for both the constraints computation and trajectory generation tasks.

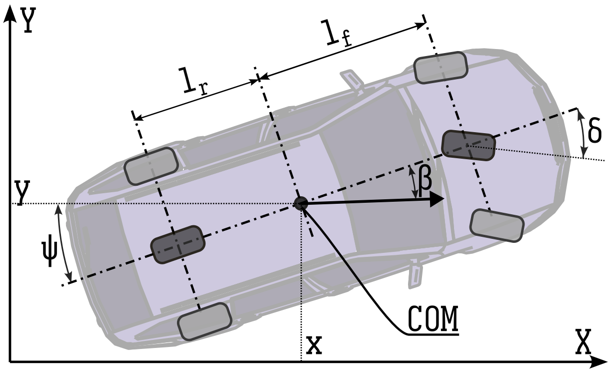


Figure 3.5. Bicycle model of the vehicle used for trajectory planning and constraints generation.

The model is defined with a set of equations:

$$\begin{aligned} \dot{x} &= v\ cos\left(\psi + \beta\left(\delta\right)\right) \\ \dot{y} &= v\ sin\left(\psi + \beta\left(\delta\right)\right) \\ \dot{\psi} &= \frac{v}{l_r} sin\left(\beta\left(\delta\right)\right) \\ \dot{v} &= a, \end{aligned} \tag{3.2.6}$$

where l_r is the distance between the CoM of the vehicle and the center of its rear axis, and $\beta(\delta)$ is the angle between the longitudinal axis of the vehicle and its velocity vector. β can be calculated as:

$$\beta(\delta) = tan^{-1}\left(\frac{l_r}{l_f + l_r} tan\left(\delta\right)\right), \tag{3.2.7}$$

where l_f denotes the distance between the vehicle's front axis and CoM. Both l_r and l_f are included in the parameters vector $\mathbf{p}_{\text{veh}}$.

3.3. Constraints Generation

A main goal that each of the trajectories generated by the described method needs to fulfill is to be collision-free in worst-case scenarios described by their respective hypotheses. This is facilitated by the introduction of spatial and spatiotemporal constraints that ensure no collisions with static obstacles (e.g., road barriers) nor dynamic ones (e.g., other vehicles) may happen assuming the given hypothesis is true.

As described in section 3.2.3, the state constraints used for the generation of i-th control trajectory are determined based on an occupancy set $\mathcal{R}_i(t)$ that describes areas potentially occupied by other road users according to the respective hypothesis H_i.

The formation of constraints may vary depending on the type of hypotheses provided by a prediction algorithm. In this section, the formation and utilization of two types of hypotheses will be presented: one related to the worst-case prediction that can be used for fail-safe planning approaches and one related to several plausible driving hypotheses, in which other vehicles are expected to fulfill more restrictive limitations, such as driving within a certain lane.

3.3.1. Worst-case Occupancy Set

The worst-case occupancy set is intended to represent the area reachable by other vehicles that do not necessarily adhere to traffic laws. Planning and following an ego's state trajectory in which its area does not overlap with such a set ensures a collision-free resolution of dangerous situations that may incorporate erratic driving of other vehicles (e.g., executed by intoxicated drivers) or sudden emergency maneuvers (such as severe braking and steering reaction to another dangerous event). In other words, a worst-case occupancy set $\mathcal{R}_{wc}(t)$ for $t \in [0, t_f]$ is calculated based on the hypothesis H_{wc} that other traffic participants can apply arbitrary control values $\delta_{\text{tp}} \in [\delta_{\text{tp}_{\text{min}}}, \delta_{\text{tp}_{\text{max}}}]$ and $a_{\text{tp}} \in [a_{\text{tp}_{\text{min}}}, a_{\text{tp}_{\text{max}}}]$, where δ_{tp} and a_{tp} are the steering angle and acceleration of other traffic participants, respectively, while $\delta_{\text{tp}_{\text{min}}}, \delta_{\text{tp}_{\text{max}}}, a_{\text{tp}_{\text{min}}}, a_{\text{tp}_{\text{max}}}$ are the parameters that approximate their extreme values.

In the following section, calculation of a worst-case occupancy set for a single vehicle is presented. Note that a complete occupancy set used for the trajectory generation can be calculated as a sum of occupancy sets of all the vehicles in the ego's proximity.

The derivation of the worst-case occupancy set is based on simulations of another vehicle's behavior under various plausible control inputs. To allow fast execution of the proposed algorithm, the occupancy set is approximated as a convex hull of the areas occupied by the vehicle under several plausible control trajectories.

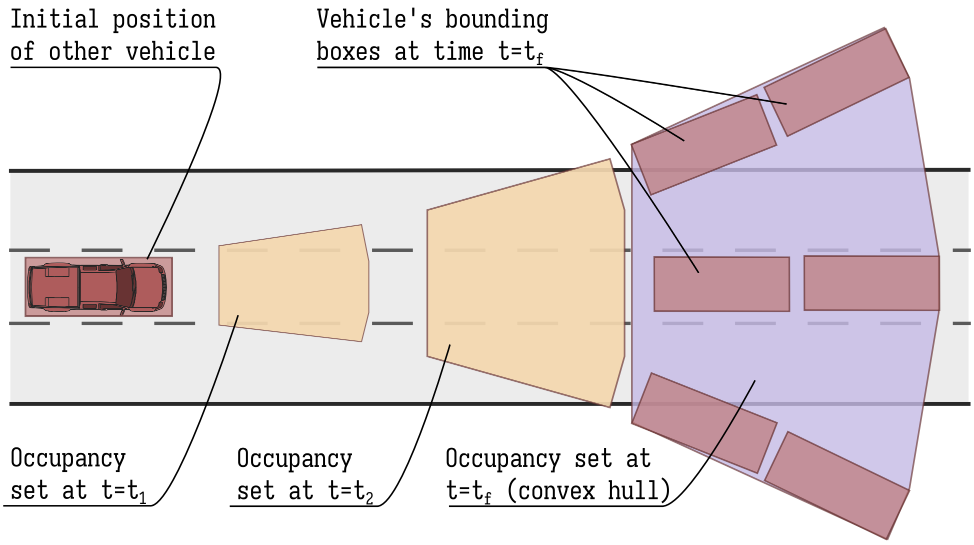


Figure 3.6. Generation of the worst-case occupancy set as a convex hull over extreme (and intermediate) occupancy polygons of a vehicle.

In particular, another vehicle's response to all permutations of constant extreme control, values is considered to determine the extreme positions of the vehicle. Additionally, to provide a better approximation of the occupancy set's boundaries, response to constant intermediate control values can also be considered. Overall, a set of relevant plausible (including extreme) state trajectories of a vehicle described by a parameter vector $\mathbf{p}_{\mathrm{veh,tp}}$ with an initial state $\mathbf{s}_{\mathrm{tp}}(0) = \mathbf{s_0}$ is denoted $\mathcal{S}_{\mathrm{pt}}$ and can be defined as:

$$\mathcal{S}_{\mathrm{pt}}(\mathcal{C}_\delta, \mathcal{C}_a, \mathbf{s}_0, t) = \{\mathbf{s}_{\mathrm{tp}}(\delta_{\mathrm{tp}}, a_{\mathrm{tp}}, \mathbf{s}_0, \mathbf{p}_{\mathrm{veh,tp}}, t)\}_{\forall(\delta_{\mathrm{tp}}, a_{\mathrm{tp}}):\delta_{\mathrm{tp}}\in\mathcal{C}_\delta, a_{\mathrm{tp}}\in\mathcal{C}_a}, \tag{3.3.1}$$

where each state trajectory $\mathbf{s}_{\mathrm{tp}} \in \mathcal{S}_{\mathrm{pt}}$ is generated through a simulation of the response of the kinematic model (3.2.6) to all combinations of the constant control values $(\delta_{\mathrm{tp}}, a_{\mathrm{tp}}) : \delta_{\mathrm{tp}} \in \mathcal{C}_\delta, a_{\mathrm{tp}} \in \mathcal{C}_a$, where $\mathcal{C}_\delta$ is a set of extreme and intermediate plausible steering angles, and $\mathcal{C}_a$ is a set of extreme and intermediate plausible accelerations.

The occupancy set $\mathcal{R}_{wc}(t), t \in [0, t_{\mathrm{f}}]$ is then calculated as a convex hull:

$$R_{wc}(\mathcal{C}_\delta, \mathcal{C}_a, \mathbf{s}_0, t) = \mathrm{Hull}\left(\{\mathrm{Box}(\mathbf{s}(t))\}_{\forall \mathbf{s}\in\mathcal{S}_{\mathrm{pt}}}\right) \tag{3.3.2}$$

over the vehicle's bounding box polygons at time $t \in [0, t_f]$.

However, the approach presented in (3.3.2) has one significant disadvantage: over longer time horizons t_{f}, large values of extreme steering angles $\delta_{\mathrm{tp}_{\min}}, \delta_{\mathrm{tp}_{\max}}$ may result in severe over-approximations of occupancy sets. To improve this, the following approach can be used instead:

$$\begin{aligned} R_{wc}(\mathcal{C}_\delta, \mathcal{C}_a, \mathrm{s}_0, t) = \bigcup_{i=1}^{|\mathcal{C}_\delta|-1} \mathrm{Hull}\Big(& \left\{\mathrm{Box}\left(\mathbf{s}_{\mathrm{tp}}(\delta_{\mathrm{tp}_i}, a_{\mathrm{tp}}, \mathbf{s}_0, \mathbf{p}_{\mathrm{veh,tp}}, t)\right)\right\}_{\forall a_{\mathrm{tp}} \in \mathcal{C}_a} \\ & \cup \left\{\mathrm{Box}\left(\mathbf{s}_{\mathrm{tp}}(\delta_{\mathrm{tp}_{i+1}}, a_{\mathrm{tp}}, \mathbf{s}_0, \mathbf{p}_{\mathrm{veh,tp}}, t)\right)\right\}_{\forall a_{\mathrm{tp}} \in \mathcal{C}_a}\Big), \end{aligned} \tag{3.3.3}$$

where $\delta_{\mathrm{tp}_i} \in \mathcal{C}_\delta$, and $\delta_{\mathrm{tp}_1} \leq \delta_{\mathrm{tp}_2} \leq ... \leq \delta_{\mathrm{tp}_{|\mathcal{C}_\delta|}}$. In this approach, a sum of convex hulls generated over trajectories with pairs of consecutively large steering angles is used to achieve a closer occupancy set approximation.

In practical implementation, the geometry of the occupancy set is approximated in n_t evenly spaced discrete-time instances t_i for $t = 1..n_t$, where $t_1 = 0, t_{n_t} = t_f$.

3.3.2. Reasonably Foreseeable Occupancy Set

Depending on the application of the proposed method, the use of strictly worst-case predictions is not always desirable, as they tend to result in overly cautious driving behaviors. As noted in [157], human drivers routinely take certain risks while driving, as it may not be feasible nor productive to take precautions against all possible situations on the road. A good example of this is driving on urban roads without physical separation between lanes with opposite traffic. Assuming that any vehicle driving in the opposite lane could steer into oncoming traffic at any time, could easily lead to the conclusion that no action of the ego vehicle can guarantee absolute safety in many typical road situations.

For this reason, it is often useful to take additional, less conservative assumptions with regard to the behavior of other road users, e.g., based on an expectation that all road users will respect certain traffic laws or avoid blatantly reckless maneuvers. While such assumptions do not reflect all possible traffic scenarios, they may help form a transparent and reasonable set of expectations with respect to the outcomes of a given road situation. This notion is reflected in traffic rules formalization efforts, which are currently an active research area [157, 129, 63].

A set of plausible future states of other road users can also be narrowed down based on various behavior prediction methods 3.1.2. While choice and calibration of a particular prediction method and safety framework are outside the scope of this thesis, in this section, I demonstrate occupancy set creation examples in common road scenarios.

3.3.2.1. Lateral Constraints

In many road situations, it is reasonable to restrict the worst-case occupancy set to the geometry of the road and/or a certain lane. An example of such a situation may be the merge-in scenario presented in Fig. 3.7. One may intuitively assume that the behavior of another road user in the near future will likely be limited to a lane change onto the *Lane 2* or a braking on a current lane.

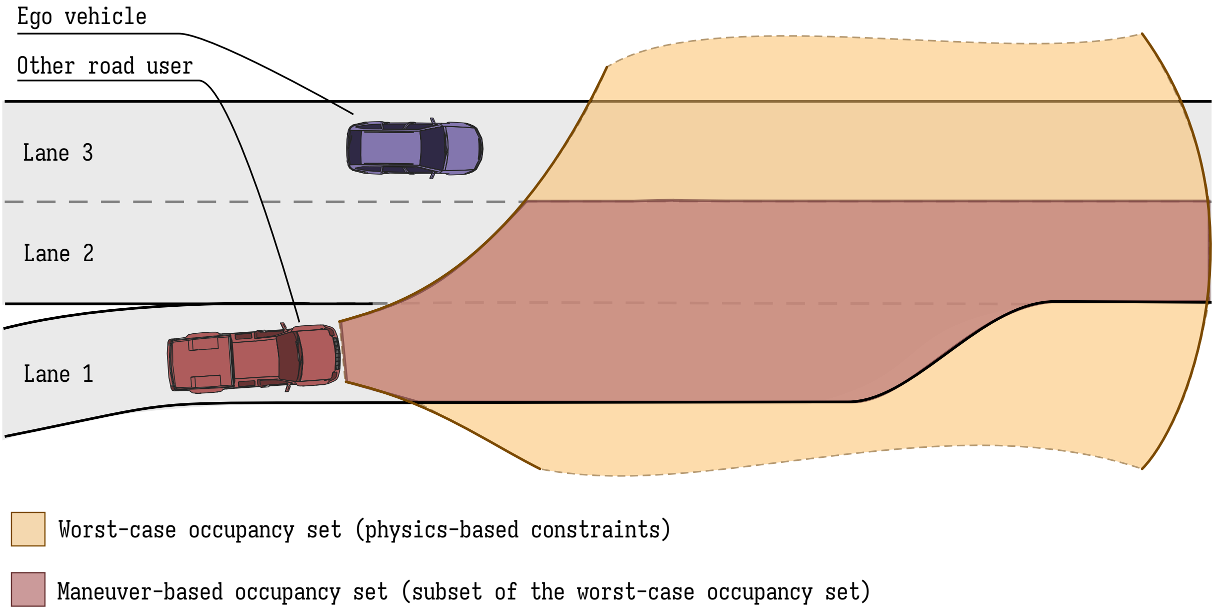

Ego vehicle
Other road user
Lane 3
Lane 2
Lane 1
Worst-case occupancy set (physics-based constraints)
Maneuver-based occupancy set (subset of the worst-case occupancy set)

Figure 3.7. Example of a maneuver-based occupancy set (presented as a sum of occupancy set geometries over a certain time period $t \in [0, t_f]$). While the other road user may perform a severe steering maneuver of moving onto the *Lane 3* or outside the road, for planning purposes it may be more reasonable to assume, that its near-future movement will be limited only to *Lane 1* and *Lane 2*.

It is important to note that such assumptions introduce a risk to the system, as the other vehicle is physically able to reach the *Lane 3* as well, and thus there is a non-zero probability that the planned ego's trajectory may lead to a collision even if it does not intersect with the calculated occupancy set. While this may seem concerning, a common approach to the ADAS design is to allow for a certain level of risk, as long as it is reasonably low and justifiable [66]. Nevertheless, the assessment of whether the given assumption is reasonable may depend on local driving culture, the design of other systems implemented in the vehicle, and thorough safety analyses. While such assessments are outside of the scope of this thesis, constraints similar to the one described above have the advantage of being relatively transparent and simple to assess - one may, for example, estimate the probability that the maneuver-based occupancy set presented in Fig. 3.7 will be in fact respected by other vehicles through an offline analysis of similar scenarios in naturalistic trajectories datasets, such as the highD [88] or inD [18] datasets.

In the example presented in Fig. 3.7, the maneuver-based occupancy set that constrains the behaviors of another road user to a lane following and a single left lane change can be derived as an intersection of the worst-case occupancy set with driving corridors representing lanes 1 and 2.

More generally, assuming that a list of the driving corridors that may be used by other vehicles is available (it can be generated by a maneuver prediction module, a safety framework [157, 129], or a set of heuristics based on an observation of a static and dynamic environment), the worst-case occupancy set can be restricted to these lanes (described as driving corridors) in a following operation:

$$\mathcal{R}_{\mathrm{ll}}(t) = \bigcup_{i=1}^{n_{\mathrm{corr}}} \mathcal{R}_{wc}(t) \cap \tau_i \text{ for } t \in [0, t_f], \tag{3.3.4}$$

where $\mathcal{R}_{\mathrm{ll}}(t)$ is the resulting lanes-limited occupancy set, $\mathcal{R}_{wc}(t)$ is the worst-case occupancy set, derived as described in Section 3.3.1, n_{corr} is the number of relevant driving corridors, and τ_i is an i-th avaialble driving corridor.

3.3.2.2. Longitudinal Constraints

Limiting occupancy sets to particular lanes helps to form reasonable assumptions with regard to future lateral positions of other vehicles, but it does not limit the possible longitudinal positions of another vehicle (unless the driving corridor has limited length). Taking simple worst-case assumptions with regard to these positions may result in unreasonably restrictive occupancy sets in long prediction horizons t_{f}.

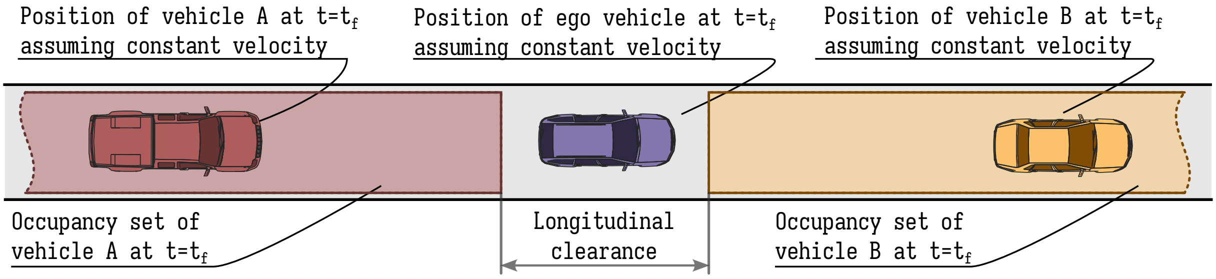

Position of vehicle A at t=t$_f$ assuming constant velocity
Position of ego vehicle at t=t$_f$ assuming constant velocity
Position of vehicle B at t=t$_f$ assuming constant velocity
Occupancy set of vehicle A at t=t$_f$
Longitudinal clearance
Occupancy set of vehicle B at t=t$_f$

Figure 3.8. Single-lane example. Distance between vehicle A's occupancy set $\mathcal{R}_{wc_1}(t)$ and vehicle B's occupancy set $\mathcal{R}_{wc_2}(t)$ is defined as a *longitudinal clearance* $s_{\mathrm{clear}}(t)$.

To illustrate this issue, a simple single-lane driving scenario presented in Fig. 3.8, in which the ego drives between two other vehicles, can be considered. To analyze this type of scenario more closely, I define a *longitudinal clearance* $s_{\mathrm{clear}}(t)$ as a longitudinal distance between occupancy sets $\mathcal{R}_{wc_1}(t)$ and $\mathcal{R}_{wc_2}(t)$ of front and rear vehicles at time t respectively, as shown in Fig. 3.8.

Assuming that the initial speed of all vehicles (including the ego) at $t = 0$ is equal ($v_{\mathrm{ego}}(0) = v_0$, $v_{\mathrm{A}}(0) = v_0$, $v_{\mathrm{B}}(0) = v_0$, with v_{ego}, v_{A}, v_{B} denoting speed of the ego, vehicle A and vehicle B respectively, and v_0 being an arbitrary initial speed value), the longitudinal clearance can be estimated as:

$$s_{\mathrm{clear}} = d_{\mathrm{AB}}(0) - \frac{1}{2} a_{\mathrm{tp_{max}}} * {t_{\mathrm{f}}}^2 + \frac{1}{2} a_{\mathrm{tp_{min}}} * {t_{\mathrm{f}}}^2, \tag{3.3.5}$$

where $d_{\mathrm{AB}}(0)$ denotes the initial distance between vehicles A and B.

Taking this into account, it is possible to find that in many relatively common scenarios, the occupancy sets of n_{sv} vehicles $\mathcal{R}_{wc_i}(t)$ for $i = 1..n_{\text{sv}}$ may overlap in relatively short time horizons t, potentially resulting in a situation where there is no feasible ego trajectory in which the ego does not violate occupancy sets. This coincides with a common-sense observation that in such situations, there is no feasible maneuver that would allow the ego to avoid a collision if the front vehicle would apply severe braking, while the rear one would accelerate.

Observation that in many traffic situations, it is impossible to guarantee absolute safety, and the responsibility for avoiding collisions is split between multiple road users, while relatively obvious, inspired a lot of attempts to formalize unwritten rules that human drivers follow intuitively, as it is often believed to be an important enabler for commercialization of AD systems [157, 129].

Perhaps one of the most important works in this area was presented by an automotive company Mobileye [157], which proposed a safety framework *Responsibility-Sensitive Safety* (RSS), which later greatly influenced the creation of an IEEE 2846-2022 standard [63]. The RSS framework, while published before IEEE 2846-2022, remains one of the possible realizations of the standard's proposals [39].

The standard and the RSS framework formalize traffic laws and certain common-sense rules that traffic participants are expected to follow (e.g. keeping a certain distance from other vehicles, or avoiding reckless cut-ins). While adherence to these rules by human drivers is not guaranteed, their transparent definition simplifies safety analysis, that is, with clearly defined rules it is possible to estimate how often traffic users violate them, e.g., through an analysis of naturalistic driving trajectories datasets [88, 18]. As the rules are transparently parameterized, they enable the design of a system that limits risks related to the behavior of other users to a certain desired level. In other words, if the system is designed to respect the RSS rules and the probability that other vehicles will follow them as well is accurately estimated, the risk of a collision in the absence of perception errors and/or system malfunctions can be estimated and limited by framework's parameterization if needed.

According to the RSS framework, the responsibility of all traffic participants in longitudinal situations similar to the example presented in Fig. 3.8 is defined by a rule, that all vehicles shall keep at least a minimum longitudinal distance d_{RSS} to vehicles in front of them, that would be sufficient to notice its sudden severe braking (e.g., in a response to some dangerous situation), and safely execute a predefined emergency maneuver. The emergency maneuver itself for longitudinal safe distance violations is defined as a necessity to react within a defined response time ρ_{RSS} by applying braking deceleration $a_{\text{brake,RSS}}(t)$ within certain limits $a_{\text{brake,RSS}}(t) \in [a_{\text{brake,min,RSS}},\ a_{\text{brake,max,RSS}}]$.

Based on the RSS proposals, several assumptions related to the planning constraints in the example scenario presented in Fig. 3.8 can be formulated:

- Ego's minimum and maximum accelerations should be constrained to $[a_{\text{brake,max,RSS}},\ a_{\text{max,RSS}}]$ limits.
- Occupancy set of the rear vehicle can be omitted in the final constraints - as long as the ego's acceleration is bounded, and the ego does not change lanes, responsibility for keeping proper distance to the ego is assigned to the rear vehicle.
- If a safe distance to the front vehicle is violated, the ego vehicle should apply emergency braking with an acceleration $a_{\text{ego,lon}}(t) \in [a_{\text{brake,max}},\ a_{\text{brake,min}}]$ for $t \in [t_{\text{viol}} + \rho_{\text{RSS}},\ t_{\text{brake,end}}]$, where t_{viol} is the time at which the violation of the safety distance occurred, and $t_{\text{brake,end}}$ is the time at which no further deceleration is necessary (because the ego vehicle reaches 0 velocity or the situation is no longer dangerous).
- Assuming that no safe distance is currently violated, the safe longitudinal distance of the front vehicle can be calculated as a distance sufficient to apply an emergency maneuver.

IEEE 2846-2022 standard and the RSS framework do not specify how planning itself should be executed, focusing only on the general requirements and assumptions regarding the behavior of other road users. The standard, however, provides a set of guidelines useful for the design of planning algorithms, including a reasonable approach to setting the longitudinal constraints.

The requirements defined in the RSS framework can be fulfilled by the proposed Multiple Hypothesis Planning algorithm through the use of two distinct hypotheses: one related to a situation in which no safe distance is violated (further referred to as *safe conditions*), and another related to an emergency situation, i.e., violation of safety distances (referred to as *dangerous situation*). This approach constitutes a relatively natural extension of the Fail-Safe Planning algorithm.

The *dangerous situation*, corresponds to a reasonably foreseeable worst-case situation, and thus the relevant occupancy set can be longitudinally constrained based on the IEEE 2846-2022 assumptions. As it is assumed that any vehicle can apply emergency braking with acceleration of at most $a_{\text{brake,max,RSS}}$, it can be used as a minimal control value $a_{\text{tp}_{\text{min}}}$ to derive an occupancy set as described in 3.3.1.

In the *safe conditions*, the main responsibility of the ego vehicle is to avoid causing dangerous situations. While being able to avoid the collision with a front vehicle in a worst-case scenario fulfills the ego's responsibility according to the RSS in the single-lane scenario, it is not sufficient in many multi-lane scenarios, such as a lane change scenario presented in Fig. 3.9.

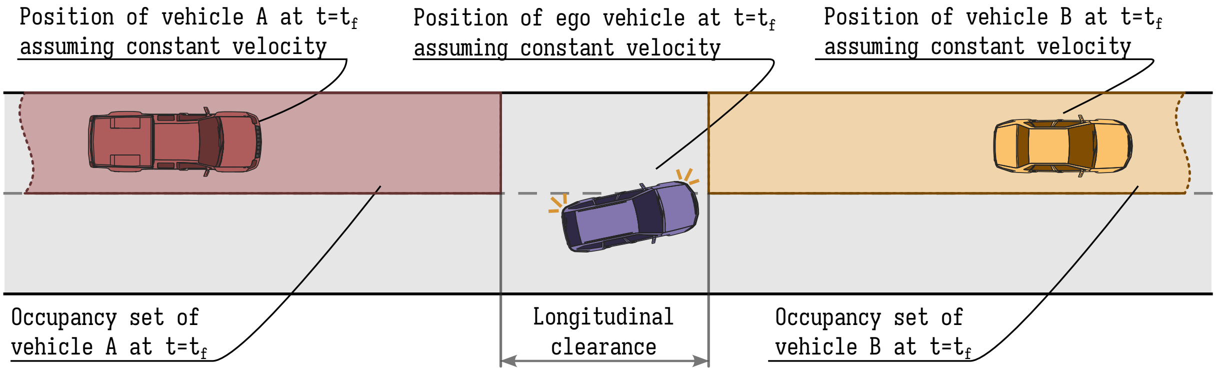

Position of vehicle A at t=t$_f$ assuming constant velocity
Position of ego vehicle at t=t$_f$ assuming constant velocity
Position of vehicle B at t=t$_f$ assuming constant velocity
Occupancy set of vehicle A at t=t$_f$
Longitudinal clearance
Occupancy set of vehicle B at t=t$_f$

Figure 3.9. Lane change example - the ego vehicle is responsible for keeping the proper distance to the rear vehicle to avoid a reckless cut-in.

In the lane change scenario, the ego vehicle needs to maintain sufficient distance not only to the front vehicle but also to the rear one, to ensure that in case of an emergency that requires ego's severe braking, the rear vehicle will be able to avoid the collision (by noticing the dangerous situation within ρ_{RSS} and applying the braking acceleration below $a_{\text{brake,min,RSS}}$). To achieve this, the longitudinal distance between the ego and rear vehicle must be at least:

$$d_{\text{min}}^{\text{lon}}(t) = \max\left(0, v_r(t)\rho + \frac{1}{2} * a_{\text{max,RSS}}\rho_{\text{RSS}}^2 + \frac{(v_r(t) + \rho_{\text{RSS}}a_{\text{max,RSS}})^2}{2a_{\text{brake,max,RSS}}} - \frac{v_f(t)^2}{2a_{\text{max,RSS}}},\right) \tag{3.3.6}$$

where $v_r(t)$ denotes the longitudinal velocity of the rear vehicle at time t, $a_{\text{max,RSS}}$ the assumed maximum foreseeable acceleration of a rear vehicle, ρ_{RSS} is the assumed response time in which the rear vehicle should start the braking maneuver in the dangerous situation, and $v_f(t)$ denotes the longitudinal velocity of the front vehicle (in this case ego) at time t. Note that this is a simplified version of the longitudinal safe distance formula proposed in [39].

Equation (3.3.6) can be used to derive rear vehicle occupancy sets for ego lane change scenarios, with two possible hypotheses: one that the rear vehicle will maintain current velocity ($v_r(t) = v_t(0)$ for $t \in [0, t_{\text{f}}]$), and one that assumes the rear vehicle applying a worst-case acceleration ($v_r(t) = v_t(0) + a_{\text{max,RSS}} * t$ for $t \in [0, t_{\text{f}}]$).

Reasonably foreseeable occupancy sets derived based on RSS rules can be used to create a control scheme similar to Fail-Safe Motion Planning, where the trajectory generation is repeated in intervals $t_p \in [0, t_h]$, ensuring that the emergency maneuver trajectory (analogous to a fail-safe trajectory) is always available, and the nominal trajectory considers the need for potential execution of the emergency maneuver. To distinguish this approach from Fail-Safe Planning, it will be referred to as Reasonably Forseeable Fail-Safe Planning (RFFS Planning) for the remainder of this thesis.

It should be noted that IEEE 2846-2022 defines a large number of other assumptions as well as possible scenarios [39], and only a subset of them are presented in this section and the examples provided in the next sections, as RFFS planning is only one of the possible applications of the presented Multiple Hypothesis Planning method.

3.3.3. Multi-modal Prediction Occupancy Sets

Fail-Safe and RFFS planning approaches described in previous subsections can be executed without the use of any external prediction method, as certain prediction capabilities are built into the described occupancy sets generation algorithms. The method can however be used in conjunction with various trajectory prediction algorithms, especially multimodal prediction methods, where each modality can be used to form a single hypothesis.

The derivation of the occupancy sets based on multi-modal prediction methods will vary significantly depending on the prediction method. For the simplest example, a multimodal prediction method that returns n_{pred} possible sets of trajectories can be considered, where each set is composed of n_{sv} state trajectories $\mathbf{s}_{\text{pred}_i}(t)$ for $t \in [0, t_f]$, $i = 1..n_{\text{sv}}$, one for each of the road users surrounding the ego vehicle. In this case, the occupancy sets $\mathcal{R}_{\text{pred}_i}(t)$ for $i = 1..n_{\text{pred}}$, $t \in [0, t_f]$ corresponding to each predicted set can be generated as a simple sum of n_{sv} vehicles' bounding boxes:

$$\mathcal{R}_{\text{pred}_i}(t) = \bigcup_{j=1}^{n_{\text{sv}}} \text{Box}(\mathbf{s}_j(t)) \text{ for } t \in [0, t_f], \tag{3.3.7}$$

where $\text{Box}(\mathbf{s}_j(t))$ denotes the bounding box polygon of a j-th vehicle at time t, with a current state $\mathbf{s}_j(t)$.

3.4. Planning

Occupancy sets discussed in a previous section have a central role in the planning problem, serving as a base for trajectory planning constraints. Each of the control trajectories planned with the proposed method must result in an ego movement that is constrained based on the respective occupancy set.

In order to fulfill the goals of the Multiple Hypothesis Planning method (i.e., that several partially overlapping trajectories are planned), planning all the trajectories is formulated as a single optimization problem. This is achieved through optimization of the parameter vector containing the parameters that describe multiple potentially different trajectories.

Each of the n_{traj} generated control trajectories $T_i(\mathbf{q}_i, t)$ for $t \in [0, t_{\text{f}}]$, $i = 1..n_{\text{traj}}$ is described using a respective vector $\mathbf{q}_i$ of n_{params} parameters. All the parameter vectors $\mathbf{q}_i$ for $i = 1..n_{\text{traj}}$ are gathered in the vector $\mathbf{q}_o = [\mathbf{q}_1, ..., \mathbf{q}_{n_{\text{traj}}}]$, allowing to formulate a following optimization problem:

$$
\begin{array}{rll}
\min\limits_{\mathbf{q}_o} & \sum\limits_{i=1}^{n_{\text{traj}}} w_i C_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}\right) & \\
\text{subject to} & \mathbf{s}_1 \left(T_1(\mathbf{q}_1), \mathbf{s}_0, \mathbf{q}_{\text{veh}}, t\right) = & \\
& = \mathbf{s}_2 \left(T_2(\mathbf{q}_2), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) = & \\
& = ... = & \\
& = \mathbf{s}_{n_{\text{traj}}} \left(T_{n_{\text{traj}}}(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) & \text{for } t \in [0, t_c]; \\
& d_i^{\min} \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) \geq 0, & \text{for } t \in [0, t_{\mathrm{f}}]\,,\ \forall i \in \{1, ..., n_{\text{traj}}\}; \\
& \delta_{\min} \leq \delta_i \left(T_i(\mathbf{q}_i), t\right) \leq \delta_{\max}, & \text{for } t \in [0, t_f],\ \forall i \in \{1, ..., n_{\text{traj}}\}; \\
& a_{\min} \leq a_i \left(T_i(\mathbf{q}_i), t\right) \leq a_{\max}, & \text{for } t \in [0, t_f],\ \forall i \in \{1, ..., n_{\text{traj}}\}; \\
& 0 \leq v_i^{\text{lon}} \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) \leq v_{\max}, & \text{for } t \in [0, t_{\mathrm{f}}],\ \forall i \in \{1, ..., n_{\text{traj}}\};
\end{array}
\tag{3.4.1}
$$

where $C_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}\right)$ is the cost value related to the i-th control trajectory $T_i(\mathbf{p}_i)$ (explained in Section 3.4.1), $\mathbf{s}_0$ denotes the initial state of the ego vehicle, $\mathbf{p}_{\text{veh}}$ is the vector that describes the ego vehicle (geometry, kinematic properties).

$\mathbf{s}_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right)$ denotes the state of the ego vehicle at time t, assuming that the vehicle followed the i-th control trajectory $T_i(\mathbf{q}_i)$, starting from the state $\mathbf{s}_0$, where $\dot{\mathbf{s}}_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) = f \left(\mathbf{s}_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right), \mathbf{c}_i(T_i(\mathbf{q}_i), t), p_{\text{veh}}\right)$, as described in Section 3.2.5.

$d_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right)$ denotes the minimal distance between the ego's bounding box $\text{Box} \left(\mathbf{s}_i \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right)\right)$ (assuming that the ego follows the i-th trajectory) and the respective occupancy set $\mathcal{R}_i(t)$ at time t, or the boundary of the drivable area.

$\delta_i \left(T_i(\mathbf{q}_i), t\right)$, $a_i \left(T_i(\mathbf{q}_i), t\right)$ and $v_i^{\text{lon}} \left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right)$ denote the steering angle, acceleration, and longitudinal velocity of the ego at time t, respectively, assuming that the ego follows the i-th control trajectory $T_i(\mathbf{q}_i)$.

It can be noted that the proposed formulation of the optimization problem corresponds to Direct Single Shooting methods used commonly for solving optimization-based trajectory generation problems. Depending on the application, other formulations are often proposed as an alternative to direct shooting methods, such as Direct Multiple Shooting methods, or a family of Direct Collocation methods. The robustness, efficiency, and implementation efforts related to each method vary depending on the application. While the use of Multiple Shooting and Direct Collocation methods was not investigated for the proposed approach, as Direct Single Shooting offered satisfactory performance for proposed applications, the described problem can be re-formulated to follow concepts of the other methods, with relatively minor modifications.

3.4.1. Cost Terms

The cost terms are intended to ensure that generated trajectories are efficient and comfortable for the passengers. Choice of the cost terms varies depending on an application, and while often similar cost terms are used to shape all the trajectories within a single optimization problem, certain variations can be applied as well.

3.4.1.1. Weight assignment strategy

The weights w_i for $i = 1..n_{\text{traj}}$ assigned to the cost terms can also vary depending on the application. In the Fail-Safe Planning schemes, the comfort-related cost terms corresponding to the fail-safe trajectories may have significantly lower values compared to the nominal trajectories, as the fail-safe trajectories are intended to be executed only in rare dangerous situations, and do not need to be nearly as efficient or comfortable as the nominal trajectory.

In systems, where Multiple Hypothesis Planning is used in conjunction with a multimodal trajectory prediction method, different strategies can be taken for choosing the cost terms values. Trajectory prediction methods often, apart from providing hypotheses $\{H_i\}_{i=1..n_{\text{traj}}}$ regarding future state of the ego surroundings, estimate probabilities $\{P_{H_i}\}_{i=1..n_{\text{traj}}}$ that each hypothesis is true (within certain tolerances), or grade hypotheses' quality in some other way. If the prediction method used in conjunction with the proposed method in fact provides some plausibility measures, they can be used to assign weights to the generated trajectory, where the largest weight would be given to the cost term corresponding to the trajectory planned based on the most plausible hypothesis.

Assigning weights based on the plausibility of the hypotheses allows increasing comfort and efficiency of the control trajectory that is most likely to be executed, at a cost of other trajectories - which is not always the desired approach, although may be beneficial in terms of the overall efficiency of the final behavior of the ego vehicle, especially if a lot of low-plausibility hypotheses are considered.

Examples of the assignment of weight values in several applications are presented in Section 3.5.

3.4.1.2. Terms

Cost value is calculated for each trajectory as a weighted sum of several cost terms that are listed below. Note that the choice of the cost terms, as well as their weights, may be modified depending on the application and end-user requirements.

- Distance from the closest driving corridor centerline cost term is defined as:

$$C_{\text{lc}_i}(\mathbf{q}_i) = w_{\text{lc}_i} \int_{t=0}^{t_f} d_{\text{lc}_i}(T_i(\mathbf{q}_i), \mathbf{s}_0, t)^2 dt, \tag{3.4.2}$$

where w_{lc_i} is the weight assigned to the C_{lc_i} cost term calculated for $i-th$ generated trajectory, and $d_{lc_i}(t)$ denotes the distance between the ego's center of mass and the closest centerline of one of available driving corridors at time t, assuming that the ego's initial state is $\mathbf{c}_0$, and it is controlled with i-th control trajectory $T_i(\mathbf{q}_i)$. This cost term encourages driving close to the lane center and helps to limit the duration of lane change maneuvers.

- Squared control values cost term used to penalize large acceleration and steering angle values is given by the equation:

$$C_{\mathrm{ctrl}_i}(\mathbf{q}_i) = w_{\mathrm{acc}_i} \int_{t=0}^{t_f} a_i(T_i(\mathbf{q}_i), t)^2 dt + w_{\delta_i} \int_{t=0}^{t_f} \delta_i(T_i(\mathbf{q}_i), t)^2 dt, \tag{3.4.3}$$

where w_{acc_i} is the weight assigned to acceleration control value, and w_{δ_i} is the weight assigned to the steering angle value. Minimization of this cost term helps to minimize G-forces, increasing the passengers' comfort and fuel efficiency.

- Speed keeping term, intended to encourage maintaining the desired (e.g., set by the user or matching the local speed limit) velocity, is defined as:

$$C_{v_i}(\mathbf{q}_i) = w_{v_i} \int_{t=0}^{t_f} (v_i(T_i(\mathbf{q}_i), \mathbf{s}_0, t) - v_d)^2 dt, \tag{3.4.4}$$

where $v_i(T_i(\mathbf{q}_i), \mathbf{s}_0, t)$ denotes the absolute speed of the ego vehicle at time t, that followed the control trajectory $T_i(\mathbf{q}_i)$, starting from the state $\mathbf{s}_0$; v_d is the desired velocity, and w_{v_i} is the weight corresponding to this cost term. In Fail-Safe Planning applications weight w_{v_i} can be set to 0 for the fail-safe trajectory, as velocity keeping is typically not a priority in emergency situations.

- The braking term is intended for use for the fail-safe trajectories in Fail-Safe Planning systems, as it encourages braking until the full stop, which is often a desirable outcome in emergency scenarios. The term is defined as:

$$C_{\mathrm{brake}_i}(\mathbf{q}_i) = w_{\mathrm{brake}_i} * v_i(T_i(\mathbf{q}_i), \mathbf{s}_0, t_f)^2, \tag{3.4.5}$$

where w_{brake_i} is the weight corresponding to this cost term.

In the practical implementation of the presented problem, ego's state trajectory is approximated in $n_{\Delta t} = \frac{t_f}{\Delta t}$ discrete time steps Δt using numerical integration methods, and thus the constraints involving integration are in fact approximated as a Riemann sum. Cost term of a form $C_{\mathrm{ex}}(\mathbf{q}_{\mathrm{ex}}) = w_{\mathrm{ex}} \int_{t=0}^{t_f} f(\mathbf{q}_{\mathrm{ex}}) dt$, where w_{ex} and p_{ex} are the weight and parameters corresponding to this cost term, is thus approximated as $C_{\mathrm{ex}}(\mathbf{q}_{\mathrm{ex}}, t) \approx \sum_{i=1}^{n_{\Delta t}} f(\mathbf{q}_{ex}, t_i)\Delta t$, where $\sum_{i=1}^{n_{\Delta t}} \Delta t = t_f$.

3.4.2. Constraints

Constraints in the proposed setup serve several purposes: ensuring overlapping of all state trajectories $\mathbf{s}_i(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t)$ for $i \in \{1, ..., n_{\text{traj}}\}$ over the initial time interval $t \in [0, t_c]$, restraining the control values to safe and feasible limits, limiting the ego's motion to the drivable parts of the road, and ensuring that its area does not overlap with relevant occupancy sets of other vehicles.

Trajectory overlap in the original problem formulation (3.4.1) is enforced by a set of equality constraints applied to the ego state functions:

$$\mathbf{s}_1\left(T_1(\mathbf{q}_1), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) = \mathbf{s}_2\left(T_2(\mathbf{q}_2), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) = ... = \mathbf{s}_{n_{\text{traj}}}\left(T_{n_{\text{traj}}}(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) \text{ for } t \in [0, t_c]; \tag{3.4.6}$$

As the initial state $\mathbf{s_0}$ and the vehicle parameters $\mathbf{p}_{\text{veh}}$ are identical for all alternative trajectories, the described set of constraints will be fulfilled if the control values $\mathbf{c}_i(T_i(\mathbf{q}_i), t), t \in [0, t_c], \forall i \in \{1, ..., n_{\text{traj}}\}$ are equal in the initial time window $[0, t_c]$. For practical implementation, this is enforced by introducing a set of constraints that enforce the equality of the control values in discrete time instances, as indicated in Equation (3.4.7).

$$\mathbf{c}_1\left(T_1(\mathbf{q}_2, t_k)\right) = \mathbf{c}_2(T_2(\mathbf{q}_2, t_k)) = ... = \mathbf{c}_{n_{\text{traj}}}(T_{n_{\text{traj}}}(\mathbf{q}_{n_{\text{traj}}}, t_k)) \quad \forall k \in \{0, \Delta t, ..., n_{\Delta t} * \Delta t\} \tag{3.4.7}$$

All control values are also constrained to certain limits that reflect the physical limitations of the vehicle, or, as in the case of safety frameworks based on IEEE 2846-2022 standard, safety-related constraints. This is reflected in the constraints:

$$\begin{aligned} \delta_{\min} &\leq \delta_i\left(T_i(\mathbf{q}_i), t\right) \leq \delta_{\max}, \quad \text{for } t \in [0, t_f],\ \forall i \in \{1, ..., n_{\text{traj}}\}; \\ a_{\min} &\leq a_i\left(T_i(\mathbf{p}_i), t\right) \leq a_{\max}, \quad \text{for } t \in [0, t_f],\ \forall i \in \{1, ..., n_{\text{traj}}\}, \end{aligned} \tag{3.4.8}$$

from the Equation (3.4.1). Practical implementation of this constraint, similar to constraints (3.4.6), is based on the evaluation of control values in discrete time steps, as in (3.4.7). The speed limitation $0 \leq v_i^{\text{lon}}\left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) \leq v_{\max}$ for $t \in [0, t_f],\ \forall i \in \{1, ..., n_{\text{traj}}\}$ is enforced in the same way.

The last set of inequality constraints is based on the occupancy sets derived in section 3.3. The geometry of the occupancy sets varies in time and between the alternative trajectories, and thus an alternative set of constraints is introduced for each of the n_{traj} generated trajectories. The constraint in the original equation (3.4.1) is defined as:

$$d_i^{\min}\left(T_i(\mathbf{q}_i), \mathbf{s}_0, \mathbf{p}_{\text{veh}}, t\right) \geq 0, \quad \text{for } t \in [0, t_f]\,,\ \forall i \in \{1, ..., n_{\text{traj}}\}, \tag{3.4.9}$$

where the distance $d_i^{\min}$ between the ego (assuming that the ego started from state s_0 and executed the $i-th$ trajectory $T_i(\mathbf{q}_i)$) and the closest polygon of i-th occupancy set $\mathcal{R}_i$ at time t (notation omitted in the following equation) is approximated as:

$$d_i^{\min} = \min\left(\{\text{-dist}(\varepsilon,\vartheta)\}_{\forall\varepsilon\in\mathcal{E}_{\mathcal{R}_i},\forall\vartheta\in\left(\mathcal{V}_{\text{Box}_{\text{ego}}}\cap\mathcal{R}_i\right)} \cup \{\text{dist}(\varepsilon,\vartheta)\}_{\forall\varepsilon\in\mathcal{E}_{\mathcal{R}_i},\forall\vartheta\in\left(\mathcal{V}_{\text{Box}_{\text{ego}}}\setminus(\mathcal{V}_{\text{Box}_{\text{ego}}}\cap\mathcal{R}_i)\right)}\right), \tag{3.4.10}$$

where $\text{dist}(\varepsilon,\vartheta)$ denotes the shortest distance between a line ε and a point ϑ, $\mathcal{E}_{\mathcal{R}_i}$ is the set of the edges of $\mathcal{R}_i$ occupancy set polygon, $\mathcal{V}_{\text{Box}_{\text{ego}}}$ is the set of ego's bounding box' vertices. It can be noted that for the ego's vertices that are within an occupancy set (denoted $\vartheta \in \mathcal{V}_{\text{Box}_{\text{ego}}} \cap \mathcal{R}_i$), a negative distance is used, meaning that negative values $d_i^{\min}$ denote the occupancy set's polygon penetration depth.

Similarly as in the case of the other constraints, the constraint (3.4.9) in the practical implementation of the optimization problem is replaced by a series of constraints related to discrete time steps.

3.5. Evaluation

The proposed method has been tested in a set of simulated scenarios that at the same time serve as examples for its various applications described in previous sections, such as the Fail-Safe Motion Planning. The experiments presented in this section were performed to fulfill several research goals related to the proposed method, as listed below.

- Investigate whether the method can be used in applications such as Fail-Safe Motion Planning in conjunction with worst-case occupancy sets generation, planning nominal and emergency maneuvers with limitations based on IEEE 2846-2022 standard, and trajectory planning in conjunction with a multimodal trajectory prediction modules.

- Investigate how the trajectory planned based on the most plausible hypothesis is impacted by a simultaneous generation of several trajectories based on less plausible hypotheses.

- Provide insight into the method's computational performance.

- Expose potential issues and limitations of the presented method to identify areas for future research and improvements.

Each of the scenarios presented in the following section is defined by a description of a static environment (defined by a road model, as described in section 3.2.2), initial states of ego and other agents, as well as a set of parameters that include the cost weights and parameters of the vehicles in the scenario.

Table 3.1. Vehicle parameters used for ego state trajectory calculation and occupancy sets generation. Note that steering angle and accelerations are arbitrarily limited and do not necessarily reflect the vehicle's physical limitations.

Parameter	Value	Unit	Description
l_r	2.0	m	Distance between vehicle's Center of Mass (CoM) and rear axis
l_f	2.0	m	Distance between vehicle's CoM and front axis
δ_{max}	0.3	rad	Max steering angle (absolute)
a_{min}	-5.0	$\frac{m}{s^2}$	Min acceleration
a_{max}	3.0	$\frac{m}{s^2}$	Max acceleration

3.5.1. Experimental Setup

The optimization problem described in Section 3.4 can be parametrized and implemented in multiple ways, depending on the intended application, available computational resources, and other requirements. For the purpose of the evaluation presented in this section, the method has been implemented as a Python module, with implementation details and adjustments listed below.

- Unless stated otherwise, all generated trajectories were described using 1st-order 2-dimensional B-splines with 16 free parameters, where both acceleration and steering angle were constrained to 0 at $t = 0$ using additional parameters.

- SLSQP optimization method [87] has been used, where constraints of the control values were implemented as actual bounds, and all other constraints were represented as additional cost terms with large weight parameters.

- The initial guess for the vector of the optimized parameters $\mathbf{q}_{\text{init}}$ is created by drawing the value of each parameter q_i from a normal distribution $q_i \sim \mathcal{N}(0, 0.1)\ \forall i \in \{1, ..., |\mathbf{q}_{\text{init}}|\}$.

- The state trajectories of the ego vehicle under various control values, as well as cost terms that include integrals, were calculated numerically with a time step $dt = 0.1[s]$.

- Curved roads were implemented as third degree B-splines, but for cost and constraints calculations, they were approximated using linear interpolation with 100 points for each driving corridor.

Details related to particular scenarios, as well as weights and time-horizon parameters, are described in the sections corresponding to each scenario.

Vehicle parameters used to acquire the state trajectories of the ego and the occupancy sets of other vehicles are presented in Table 3.1.

3.5.2. Evaluated Scenarios

As the proposed Multiple Hypothesis Planning method can be applied in several different setups, the example scenarios described in this section vary in terms of road geometry, goals, and type of occupancy sets.

3.5.2.1. Highway Fail-Safe Planning scenario

The highway scenario is intended to showcase the fail-safe planning application of the presented method with the worst-case occupancy sets.

The scenario defines a simple overtaking situation on a straight three-lane road. The ego vehicle drives with a $12\frac{m}{s}$ velocity in a middle lane, while the other vehicle, placed in a right lane at a longitudinal distance of $10m$ from the ego, maintains a velocity of $10\frac{m}{s}$.

The worst-case occupancy set of the other vehicle is derived based on assumptions that its acceleration is limited to $a \in [-5.5, 2]$, and the steering angle to $\delta \in [-0.05, 0.05]$. On the other hand, the most plausible hypothesis (further referred to as *nominal hypothesis*) is that the vehicle will continue to move in its current lane, maintaining its current speed.

Table 3.2 summarizes the values of other relevant weights and calibration parameters used in this scenario.

Table 3.2. Calibration parameters used in the urban intersection scenario

Parameter	Value	Unit	Description
t_h	1.0	s	Trajectories initial overlap time horizon
t_f	3.0	s	Planning time horizon
w_{lc_1}	1.0	-	Lane centering cost term weight for *Hypothesis 1*
w_{lc_2}	0.0	-	Lane centering cost term weight for *Hypothesis 2*
w_{acc_1}	5.0	-	Acceleration cost term weight for *Hypothesis 1*
w_{acc_2}	0.0	-	Acceleration cost term weight for *Hypothesis 2*
w_{δ_1}	1.0	-	Steering angle cost term weight for *Hypothesis 1*
w_{δ_2}	0.0	-	Steering angle cost term weight for *Hypothesis 2*
w_{v_1}	0.0	-	Velocity keeping cost term weight for *Hypothesis 1*
w_{v_2}	0.0	-	Velocity keeping cost term weight for *Hypothesis 2*
w_ct	1000	-	Trajectory overlapping cost term weight
v_d	11.0	m/s	Desired velocity
w_{brake_1}	0.0	-	Braking cost term weight for *Hypothesis 1*
w_{brake_2}	10.0	-	Braking cost term weight for *Hypothesis 2*

The execution of the method in the described scenario produces the results presented in Fig. 3.10. The control trajectory, planned based on the nominal hypothesis, results in the ego vehicle maintaining its current speed and yaw angle, with a slight slowdown at the beginning. On the other hand, the fail-safe trajectory, planned based on the worst-case hypothesis, consists of severe braking and steering executed after the initial trajectories overlap period. The ego changes the lane to the left while executing the braking to avoid crossing the occupancy set of the other vehicle.

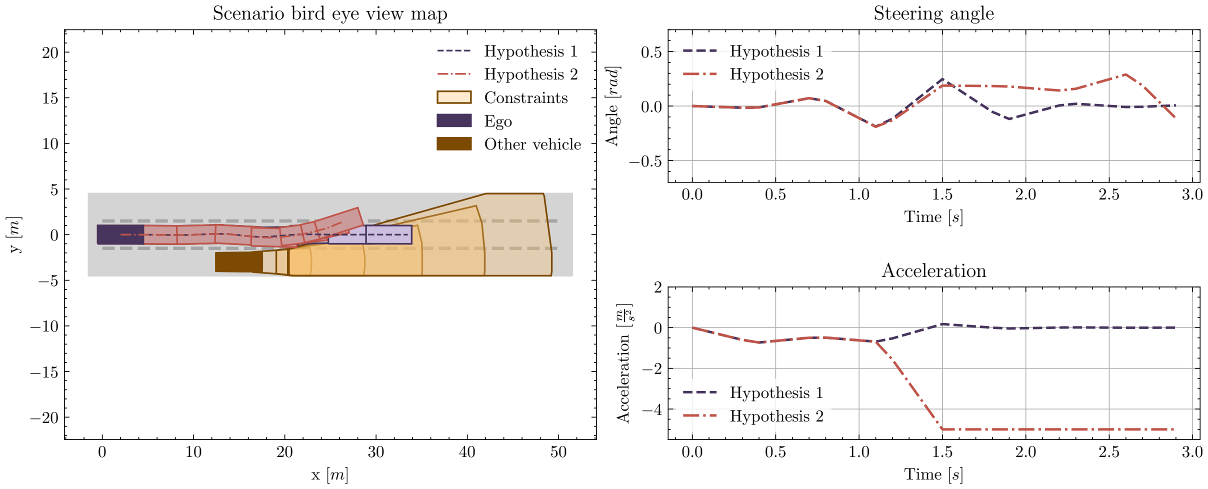


Figure 3.10. Trajectories generated for the highway fail-safe planning scenario.

It can be noted, however, that use of the strictly worst-case predictions in similar cases may result in overly cautious behaviors of the ego vehicle. This issue becomes especially apparent if we consider a variant of the discussed scenario, in which an additional vehicle is added beside the non-ego road user, driving with the same speed as the first one. As shown in Fig. 3.11, in such a configuration, the occupancy sets of both vehicles overlap, resulting in very restrictive constraints.

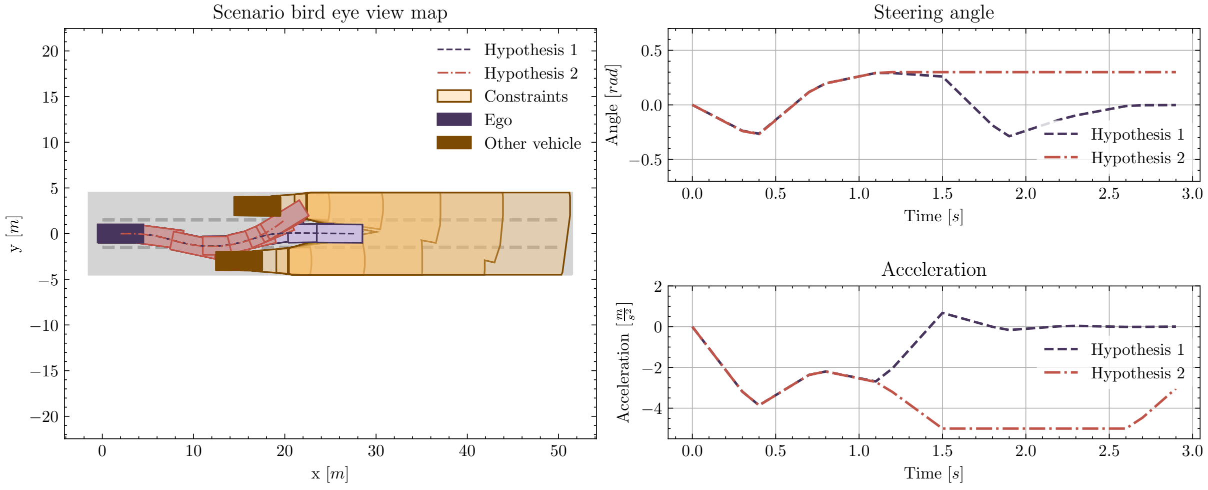


Figure 3.11. Trajectories generated for the highway fail-safe planning scenario with two other vehicles.

Although the method was able to find a collision-free fail-safe trajectory, it consists of severe braking and steering, utilizes all the three lanes, and severely impacts the nominal trajectory, causing the ego to brake heavily in $t \in [0, t_h]$.

For this reason, the use of strictly worst-case assumptions in all situations may not be a desirable solution. While they may still be beneficial in certain situations, e.g., in emergency situations in which other vehicles execute severe and unpredictable emergency maneuvers, other solutions, such as one featured in the next example, may be more appropriate for typical driving situations.

3.5.2.2. Urban intersection scenario

Urban intersection scenario is indented to evaluate the applicability of the proposed method to Reasonably Foreseeable Fail-Safe Planning (RFFSP) with assumptions based on IEEE 2846-2022 [63].

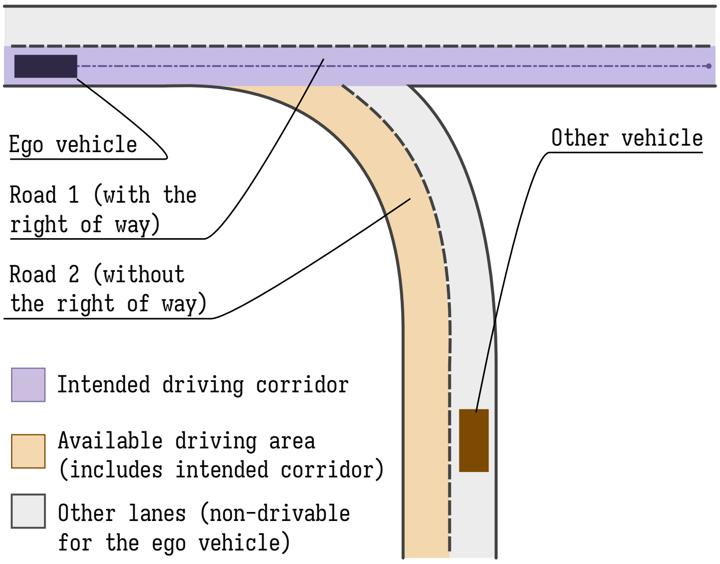

Ego vehicle
Other vehicle
Road 1 (with the right of way)
Road 2 (without the right of way)
Intended driving corridor
Available driving area (includes intended corridor)
Other lanes (non-drivable for the ego vehicle)

Figure 3.12. Urban intersection scenario overview.

In this scenario, the ego vehicle is driving on a two-lane straight main road (denoted *Road A*) approaching an intersection with a road without right of way (further referred to as *Road B*), as shown in Fig. 3.12. At the same time, another vehicle approaches the intersection on the *Road B* with the intention of merging into traffic on the *Road A*, on the lane with opposite traffic to the ego's lane. While the other vehicle has no right of way and should yield before the ego vehicle, its high speed may suggest that its driver may not intend to do so (which may be caused by the driver's failure to notice the ego vehicle, false assumption that they are on the road with the right of way, or otherwise incorrect situational assessment). The initial velocity of both vehicles is set to $11\frac{m}{s}$.

Use of the strictly worst-case prediction in such a scenario would be problematic - it is feasible for the other vehicle to swerve into oncoming traffic, making it practically impossible for the ego vehicle to avoid the collision in such a case. As similar scenarios are relatively common in urban environments, the use of the RFFSP may be a better alternative for urban applications.

In the discussed scenario, two major plausible hypotheses can be formed: one that the other vehicle will yield before the ego by stopping before the intersection (*Hypothesis 1*), and one that it will fail to do so (*Hypothesis 2*), which would correspond to the reasonably foreseeable worst-case scenario. In both hypotheses, the vehicle is assumed to remain within its driving corridor

Table 3.3. Calibration parameters used in the urban intersection scenario

Parameter	Value	Unit	Description
t_h	0.8	s	Trajectories initial overlap time horizon
t_f	3.0	s	Planning time horizon
w_{lc_1}	1.0	-	Lane centering cost term weight for *Hypothesis 1*
w_{lc_2}	0.1	-	Lane centering cost term weight for *Hypothesis 2*
w_{acc_1}	5.0	-	Acceleration cost term weight for *Hypothesis 1*
w_{acc_2}	0.0	-	Acceleration cost term weight for *Hypothesis 2*
w_{δ_1}	1.0	-	Steering angle cost term weight for *Hypothesis 1*
w_{δ_2}	0.0	-	Steering angle cost term weight for *Hypothesis 2*
w_{v_1}	0.5	-	Velocity keeping cost term weight for *Hypothesis 1*
w_{v_2}	0.0	-	Velocity keeping cost term weight for *Hypothesis 2*
w_ct	1000	-	Trajectory overlapping cost term weight
v_d	12.0	m/s	Desired velocity
w_{brake_1}	0.0	-	Braking cost term weight for *Hypothesis 1*
w_{brake_2}	20.0	-	Braking cost term weight for *Hypothesis 2*

composed of a union of the lanes consistent with its initial driving direction on both roads (i.e., the right lane from its perspective). As *Hypothesis 1* assumes that the other vehicle breaks successfully before the intersection, it does not need to use constraints based on occupancy sets, as long as the ego motion is restricted to the available driving area.

For *Hypothesis 2*, a reasonably foreseeable worst-case behavior of the other vehicle is considered. Longitudinal acceleration limits used to create its occupancy set were chosen based on the literature review [64] published by the authors of the IEEE 2846-2022 standard [63], in particular the analysis [135] of the InD dataset [18] of the naturalistic trajectories of road users at intersections. According to these analyses, the 99th percentile of the acceleration values in intersection scenarios is within -2.9 and $2.8[m/s]$ for the passenger cars, and thus $a_{\mathrm{brake,max,RSS}}$ and $a_{\mathrm{max,RSS}}$ are set to these values respectively.

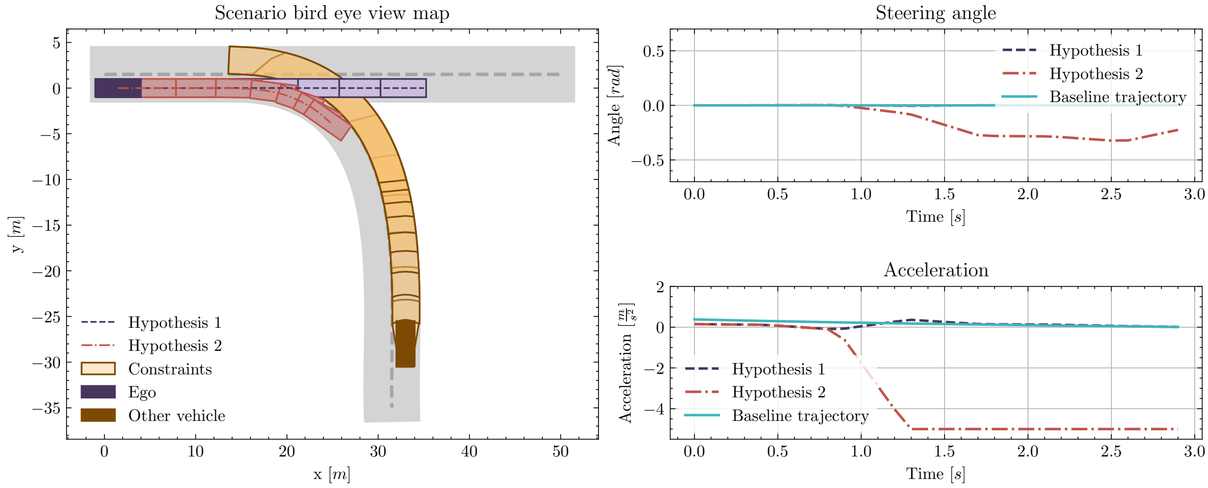
The lateral limits of the occupancy set are based on the steering angle limits in the range $\delta \in [-0.3, 0.3]$, which is sufficient to follow the geometry of the road. The resulting occupancy set is then restricted to lanes with a correct driving direction (i.e., opposite traffic lanes from the ego perspective).

The values of the parameters and the weights of the cost terms used in this example are summarized in the table 3.3.

Solving the optimization problem formulated for this case produces the trajectories presented in Fig. 3.13. Following the control trajectory generated based on *Hypothesis 1* results in a straight forward motion with the acceleration values close to 0. Following the second of the generated trajectories, on the other hand, results in a severe collision avoidance maneuver executed right after the initial overlap period. Since the initial distance to the intersection (approximately $17m$) is too short to allow braking to a full stop without crossing the other vehicle's occupancy set, *Road 2* is utilized for the collision avoidance maneuver, ensuring the collision-free trajectory.

The maneuver planned based on the *Hypothesis 2* consists therefore of braking with acceleration $a_{\text{brake,max,RSS}}$ applied after the overlap horizon and the steering angle trajectory that facilitates driving onto the *Road 2* and following its curvature.

Although such behavior is not always desirable and can be prevented by excluding the *Road 2* from the ego's drivable area, presented results demonstrate the method's ability to utilize the entire available driving area for potential emergency maneuvers.


Scenario bird eye view map

Steering angle

Acceleration

Figure 3.13. Trajectory generation results for the urban intersection scenario.

One of the important features that distinguishes the proposed method in the context of fail-safe planning applications from existing ones [116] is the fact that the nominal trajectory (planned based on *Hypothesis 1* in this case) is impacted by the necessity to plan the fail-safe trajectory. In other words, if the optimal trajectory (in terms of comfort, fuel efficiency, etc.) planned solely based on the most plausible hypothesis (the first one in this case) would not enable the generation of a collision-free fail-safe trajectory, the use of the proposed method will result in an emergence of "cautious" behaviors, such as slowing down to enable collision braking shall a dangerous situation occur.

The described feature of the proposed method is investigated by planning the baseline trajectory for the scenario analyzed. The trajectory is planned based solely on the *Hypothesis 1*, otherwise using the identical configuration as in the described example for *Hypothesis 1*. The resulting trajectory, as shown in Fig. 3.13, is similar to the nominal trajectory, but the acceleration in the initial time period t_h is noticeably higher. This suggests that in the nominal trajectory, the initial acceleration is kept lower to enable the smooth application of the emergency maneuver if needed, resulting in the ego exhibiting behavior that can be interpreted as a cautious one.

3.5.2.3. Multimodal trajectory prediction: merge-in scenario

To evaluate the applicability of the method to complex situations with more than two competing hypotheses, a scenario of merging into highway traffic has been prepared. In this scenario, the ego vehicle is driving at a speed of $14\frac{m}{s}$ on a short merge-in lane and has to merge into traffic on the two-lane straight highway road. On this road, the right lane is occupied by a vehicle that is driving with $15\frac{m}{s}$, while the left lane remains empty.

In this case, 3 hypotheses are arbitrarily generated for evaluation purposes, as listed below. The hypotheses were designed in an attempt to describe behaviors often observed in similar situations, and, while crafted manually, may reflect the plausible output of a multimodal trajectory prediction algorithm.

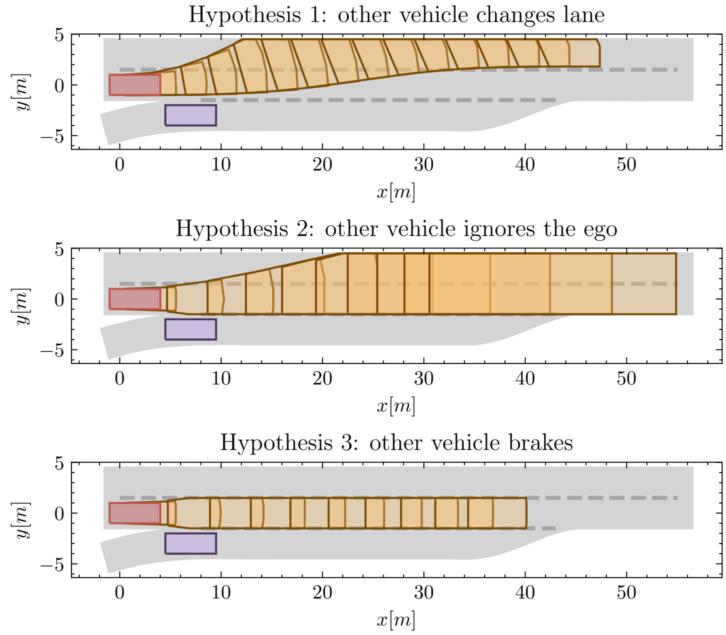


Figure 3.14. Constraints generated for different hypotheses. The ego vehicle is marked blue, the other vehicle - red, and the constraints - yellow.

- *Hypothesis 1* assumes, that the other vehicle will change the lane to the left, allowing the ego to safely merge into the traffic.
- *Hypothesis 2* is related to a situation in which another vehicle ignores the ego and encompasses various plausible maneuvers limited to the area of the main road, similar to the reasonably foreseeable worst-case hypotheses described in the previous example. Acceleration of the other vehicle is limited to $a \in [-2.5, 2.5]\frac{m}{s^2}$.
- *Hypothesis 3* assumes, that the other vehicle will perform a braking maneuver (with a constant braking acceleration $a = -1.5\frac{m}{s^2}$) in its current lane, helping ego to merge into the traffic.

The constraints generated for each of the hypotheses are shown in Fig. 3.14, and the calibration parameters are collected in Table 3.4. In this example, the lane-keeping term is calculated as a distance from the right lane of the main road, as it is the target lane for the merge-in maneuver.

Table 3.4. Calibration parameters used in the scenario. All parameters with the index i, are identical for all hypotheses ($i \in \{1, 2, 3\}$).

Parameter	Value	Unit	Description
t_{h}	0.8	s	Trajectories initial overlap time horizon
t_{f}	3.0	s	Planning time horizon
w_{lc_i}	1.0	-	Lane centering cost term weight
w_{acc_i}	1.0	-	Acceleration cost term weight
w_{δ_i}	1.0	-	Steering angle cost term weight
w_{v_i}	0.0	-	Velocity keeping cost term weight
w_{ct}	1000	-	Trajectory overlapping cost term weight
v_{d}	12.0	m/s	Desired velocity
w_{brake_i}	0.0	-	Braking cost term weight

The execution of the proposed method in the described scenario results in the generation of three distinct trajectories. All trajectories overlap in the time horizon $t \in [0, t_h]$, resulting in driving close to the center of the merge lane while applying moderate braking. After t_h, the trajectory generated based on Hypothesis 1 (that the other vehicle will perform a lane change) consists of a near zero acceleration and early execution of a merge-in maneuver. The trajectory based on *Hypothesis 2* consists of sudden and severe braking that restricts the movement of the ego to the merge-in lane, ensuring a collision-free trajectory. *Hypothesis 3* that assumes the braking maneuver of the other vehicle results in the ego's trajectory that consists of a speed-up with a late execution of a lane change maneuver. A summary of these results is presented in Fig. 3.15.

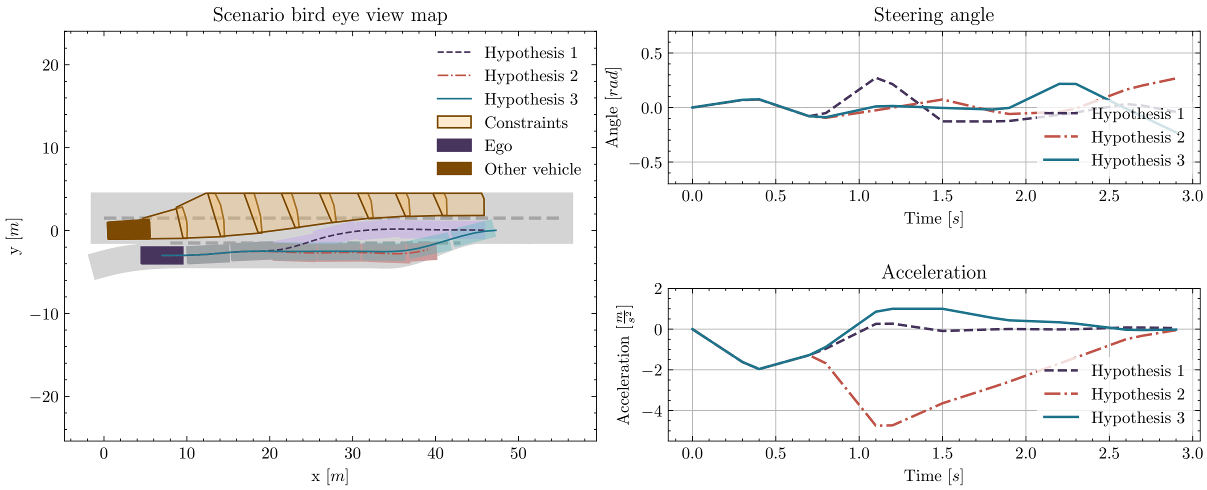


Figure 3.15. Trajectory generation results for the merge-in scenario. Note, that only the constraints for the $Hypothesis 2$ were plotted for clarity.

The behavior of the ego vehicle controlled according to the generated control trajectories seems to reflect the typical behavior of human drivers in similar situations. In the initial overlap period, moderate braking allows one to gain more time to gather the necessary information about the evolution of the surrounding situation, while preparing the ego vehicle for a potential braking maneuver. The lane change of the other vehicle allows the ego to safely perform a merge-in without additional requirements, which can be seen in the first of the generated trajectories. The third trajectory, on the other hand, represents a reasonable trajectory for a case in which another vehicle yields by performing a braking maneuver - the ego in this case speeds up, performing a lane change at the very end of the merge-in lane. In the event that the other vehicle continues to move, preventing the ego from performing a lane change, the ego may break on its current lane. Note, however, that due to a short planning horizon, this does not guarantee that the ego will reach a full stop at the end of the lane. To alleviate this issue, an additional constraint based on safe distance (described in equation (3.4.9)) to a static lane end point may be introduced for similar cases.

3.5.3. Computational Performance

All experiments were implemented in the Python language using the SLSQP optimization method [87] implemented in the *scipy* library and executed on a single thread of Intel(R) Core(TM) i7-12800H CPU. The resulting execution times for all scenarios except the last one took approximately $0.5 - 1.0s$ assuming that linear B-splines with eight parameters were used for each trajectory. Although this performance is sufficient to execute the method in a loop with a replanning frequency below t_h, especially if previously generated trajectories would be used as initial guesses for subsequent optimization runs, further work may be needed to ensure satisfactory and reliable performance in difficult cases. This is especially apparent in the third scenario, which took several seconds to compute due to a significantly larger dimensionality of the solution space.

Profiling the application suggests that relatively slow calculation can be attributed mainly to the geometrical calculations in the cost and constraints evaluations. Since the cost function gradient in the utilized optimization method is approximated numerically, the evaluation of cost terms is repeated multiple times at each iteration of the optimization, requiring substantial processing power.

These findings suggest that the performance of the proposed method can be significantly improved using techniques such as the use of automatic differentiation for the gradient estimation, approximation of certain geometries (e.g., the ego vehicle's bounding box) with simpler shapes, or linearization of the vehicle model. Potential improvements are further discussed in the Conclusions section.

However, it should be noted that the use of the proposed method for fail-safe planning applications brings a significant advantage to real-time systems. In setups where the algorithm is executed in a loop, a collision-free fail-safe trajectory from previous iterations is always available, and thus can be utilized by the system if the optimization algorithm fails to produce new trajectories in a predefined calculation time limit.

3.6. Conclusions

In this chapter, I presented a novel vehicle motion planning method intended for AD and ADAS applications, where several different hypotheses can be formed with regard to the current and future state of the vehicle's environment. The hypotheses can be produced by a separate multimodal trajectory prediction module or formed based on a set of assumptions related to the plausible future behavior of other road users. The method generates several control trajectories that overlap in an arbitrarily long initial time period, allowing postponement of the decision on which trajectory should be followed. During this time additional data regarding vehicle surroundings can be gathered, most notably, further behavior of other road users can be observed, helping to assess the validity of the utilized hypotheses.

The presented method can be utilized to fulfill the Fail-Safe Motion Planning principles, i.e., planning a nominal trajectory based on the most plausible hypothesis regarding the future state of the other road users, as well as additional fail-safe trajectory based on worst-case assumptions regarding others' future behavior.

The proposed method is distinct from the existing Fail-Safe Planning methods, as it allows for simultaneous planning of all the trajectories (e.g., nominal and fail-safe) using a single optimization problem. As a consequence of this approach, the nominal trajectory is affected by the need to establish a feasible fail-safe trajectory. This feature results in the generation of safe and cautious nominal trajectories in situations where existing fail-safe approaches would enforce the execution of emergency maneuvers.

Furthermore, in this chapter, a methodology for the generation of worst-case occupancy sets was presented, together with an alternative proposal for the creation of occupancy sets based on reasonably foreseeable assumptions regarding the behavior of other road users. This approach, inspired by safety frameworks such as RSS [157] or SFF [129], can be used to fulfill the principles of the new IEEE 2846-2022 standard [63], ensuring a collision-free trajectory under reasonable assumptions outlined in the standard.

The use of the method has been demonstrated in several simulation examples, showcasing its applicability to various road situations and system architectures.

3.6.1. Limitations and Further Work

The proposed approach has several limitations that may be addressed at least partially by further work.

3.6.1.1. Perception errors

While the main objective of fail-safe planning is to provide certain guarantees with regard to safety, it does not address the problem of perception errors that remain a major issue in ADAS and AD systems, and all considerations presented in this chapter assume the availability of a perfect model of both static and dynamic environment. Further work is required to ensure that the method is robust in the presence of such errors. Sensor modeling methods presented in Chapter 4, as well as Adversarial Trajectory Generation methods described in Chapter 5 can be used to further test the robustness of the method to perception errors and atypical behaviors of other road users in simulation. Additional evaluation of the method in the setups similar to Model Predictive Control is also required to ensure the method's stability in closed-loop control applications.

3.6.1.2. Computational performance improvements

One of the main limitations of the proposed method is its relatively low computational performance. This can be mainly attributed to the cost gradient approximation method, as mentioned in Section 3.5.3. Since the gradient is approximated numerically, its computation time is proportional to the number of optimization parameters. Furthermore, the cost value, as well as constraints violation values (in the analyzed examples actually included in the cost), depend on the optimized variables in a heavily nonlinear way, due to the nonlinear vehicle model and problem formulation, which can be described as a variant of the Direct Single Shooting Method. Because of this, the resulting optimization problem is relatively difficult to solve, and finding a satisfactory solution requires a large number of iterations of the optimization algorithm. Depending on the optimization method used to solve the problem, the proposed formulation may also make it difficult to find a globally optimal solution instead of a local optimum.

The performance of the method can be improved in a number of ways, including but not limited to the ones listed below.

- Use of the Direct Multiple Shooting or Direct Collocation trajectory optimization techniques. Direct multiple shooting is an extension of the utilized single shooting method, in which the trajectory is generated as a series of short segments, with additional constraints that enforce the continuity of the resulting trajectory. On the other hand, in direct collocation, both the control trajectory and state trajectory are generated by optimization of their respective parameters, and a series of additional constraints based on the dynamic model of the vehicle ensure the physical feasibility of the resulting solution. Both methods are commonly utilized for similar classes of problems, and while these formulations tend to increase the dimensionality of the problem and the number of constraints, the resulting problem is typically sparse and significantly easier to solve.

- Efficient cost gradient estimation techniques. As the proposed problem is relatively complex, the gradient is not calculated analytically but is estimated numerically through a finite differences method. Numerical estimation of the cost gradient, as well as constraints Jacobian in constrained problems, tend to be the most computationally expensive aspect of optimization-based trajectory generation. Fortunately, several techniques can be utilized to address this problem, such as the use of parallel computing or the utilization of gradient-free optimization techniques. One of the most convenient solutions, however, is to utilize automatic differentiation techniques [144], which exploit a chain rule to calculate accurate partial derivatives of arbitrary functions in an efficient way.

- Problem simplification. In the proposed method, the vehicle is described using a nonlinear kinematic model, and the geometric computations used for the constraints and cost calculations are often relatively time-consuming. Several simplifications can be introduced to improve the performance of the computation, such as approximation of the vehicle and/or constraints geometries using multiple circles rather than polygons, use of simplified point-mass kinematic models, or use of efficient approximations of trigonometric functions based on lookup tables.

With sufficient performance, the proposed method in a fail-safe planning setup is particularly useful for real-time applications, as it can be used to guarantee constant existence of a collision-free trajectory that can be safely executed if the system fails to produce new trajectories within a predetermined time limit.

3.6.1.3. Further applications

While the discussed method has been presented mainly in fail-safe planning and multimodal prediction-based planning applications, it can also be extended to enable different applications. Since conflicting hypotheses can be formed in the context of motion planning with regard to not only the behavior of other road users, but also the current state of the ego and its environment, the proposed method can be used in conjunction with such hypotheses as well.

As an example, one may consider the situation in which a perception system detects an object but with a low detection confidence, which may happen, e.g., in situations where only one of the sensors with overlapping sensing areas in a sensor fusion setup indicates the object's existence. In such situations, one of the hypotheses used for multiple hypothesis planning may be that the object in fact exists, while the other hypothesis is that the object was falsely detected and in fact does not exist. Use of the multiple hypothesis planning is a very convenient way to address such situations, as often gathering further observations from the sensors allows us to confirm or disprove the hypotheses, and thus initial overlap of the planned trajectories helps to make informed decisions regarding the ego's behavior.

As the use of redundant sensor sets (e.g., radars and cameras) is common in ADAS/AD systems, the proposed method can also be utilized to directly address situations in which conflicting observations are provided by different sensors. Planning a trajectory that takes into account all observations would greatly reduce the collision probability, especially if the error occurrences between sensor types are not strongly correlated. Assuming that each sensor set provides a hypothesis regarding the environment's state, planning a trajectory that respects all the worst-case constraints derived from these hypotheses ensures a collision-free driving plan as long as at least one hypothesis is true. Although the existence of a feasible trajectory that fulfills this goal is not guaranteed, the proposed method may still sufficiently reduce the system's susceptibility to perception errors. Nevertheless, further work is needed to evaluate the correlation between sensor sets' error occurrences and the method's performance in conjunction with imperfect sensing systems.

Automotive sensors are rarely able to detect road users in areas that are occluded, e.g., by other vehicles, vegetation, or buildings. Safety frameworks such as RSS [157] propose to assume that occluded areas may in fact contain other road users, taking into account reasonably foreseeable worst-case predictions regarding their behavior. The approach presented in this chapter could as well be utilized to address such situations, as the hypotheses can be also formed with regard to such an object's existence and state. Formation of such hypotheses requires, however, further extension of the proposed worst-case occupancy sets creation method.

4. Sensor Modeling

Perception systems that estimate the state of the static and dynamic environment of the vehicle play an essential role in ADAS and AD systems. Modern vehicles are often equipped with several sensors that provide information about road geometry, static obstacles, and other road users in their vicinity. Since AD/ADAS systems utilize this information to make safety-critical decisions, setting the performance requirements for perception systems and ensuring that they are fulfilled is immensely important in the design and validation of such systems.

Estimates of the environment's state provided by perception systems are, however, inherently stochastic, and sporadic occurrence of severe errors is inevitable. Sensors used in the automotive industry are susceptible to several types of errors, the nature of which often differs between sensor types. For instance, the performance of the camera-based object detection systems may be negatively impacted by adverse environmental conditions, such as heavy rain, while the performance of radar-based sensors may suffer in a cluttered environment with many surfaces that may cause undesired multi-bounce reflections of the radar wave. Since the creation of perfect sensing systems is impossible, the designer of ADAS/AD systems must ensure that they operate acceptably even in the presence of perception errors and inaccuracies. This requirement presents several challenges in algorithm design and validation.

Ensuring a robust behavior of the ADAS/AD systems in the presence of perception errors remains a difficult problem that lacks simple and universal solutions. Understanding the characteristics of the errors and modeling them in an efficient manner plays a significant role in the testing and development of such systems.

4.1. Introduction and Motivation

Deterministic ADAS/AD algorithms often require complex heuristics to handle potential errors, e.g., by ensuring that safety-critical decisions are supported by several subsequent measurements or data from several sensors with weakly correlated errors. However, these solutions come with their own problems, such as slow reaction times caused by the waiting for subsequent measurements or the need for an expensive set of sensors to ensure adequate redundancy.

One commonly proposed alternative is to utilize machine learning-based algorithms instead, harnessing their ability to adapt to various complex environments. However, learning such models requires a large amount of exemplary data that would incorporate realistic sensor inaccuracies and errors. While in certain learning approaches perception system output registered during real-world test drives can be utilized for this purpose, a large subset of approaches utilized in AD systems relies on closed-loop simulation techniques, in which artificial sensor data that incorporate realistic error patterns must be created on demand. A good example of such an approach is Reinforcement Learning, which typically utilizes simulation of the driving environment, and thus requires synthetic perception data generation.

The need for automatic generation of realistic sensor data streams based on a perfect description of the vehicle's environment (e.g., generated by a traffic simulator) is even more prominent in the testing and validation of ADAS/AD systems. Due to the immense efforts related to end-to-end on-road ADAS/AD systems testing, virtual simulation environments are often proposed as an efficient way to support testing efforts. Since the precise simulation of all physical phenomena that impact the sensor's performance is prohibitively difficult, various techniques of approximation of expected sensor outputs are proposed to facilitate the provision of realistic perception data streams to the tested algorithms.

As a consequence of previously defined design and validation needs, the automatic generation of artificial perception data from simulation-based ground truth, further referred to as sensor modeling, plays a critical role in the development of ADAS / AD systems.

The types of commonly used automotive sensors and their corresponding error types are presented in Chapter 2. In this chapter, I present a short introduction to existing sensor modeling approaches, as well as propose a set of sensor modeling methods that can be efficiently used for the testing and development of ADAS/AD systems. The effectiveness of the proposed sensor models is presented in the example of Reinforcement-Learning-based behavior and trajectory planning system, which utilizes them in a simulation-based learning process. Finally, the impact of sensor model utilization in the development of AD algorithms on system performance is analyzed and discussed, based on the results of the extensive simulation-based evaluation.

4.1.1. Existing Approaches

Sensor modeling finds a number of applications in the development, validation, and verification of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving Systems (ADS). While on-road physical testing of vehicle systems provides valuable information about their real-world performance, it is often not a cost-effective or even viable way to test ADAS/ADS, especially in the early stages of their development.

A commonly used alternative is the use of simulation tools that enable testing of ADAS/ADS algorithms in virtual environments. Simulation tools, however, rarely are able to efficiently and realistically re-create sensor data streams that would closely mimic the realistic performance of the perception sensors and algorithms, especially since they tend to vary significantly depending on the number and type of sensors, as well as the type of the perception and tracking algorithms.

Thus, models of the sensing systems are often developed for specific applications, depending on their desired performance and accuracy, as well as the architecture of the modeled system.

4.1.1.1. Automotive Sensor Models

Due to the sheer quantity of existing sensor modeling techniques, they are often categorized in various ways. The most common and simple categorization can be performed, depending on the models' fidelity, by assigning them to one of two classes: high-level or low-level models.

Low-level (high-fidelity) models aim to accurately recreate error patterns observed in given perception systems through a detailed simulation of the physical phenomena that affect a given class of sensors. High-level (low-fidelity) models, on the other hand, tend to take a statistical approach, modeling the outputs of the sensors as stochastic processes with statistical properties similar to the values observed in real systems. High-level models typically generate a coarser approximation of sensor output compared to low-level models, omitting the complex physical simulations used in high-fidelity approaches.

High-fidelity models vary significantly depending on the type of sensor they attempt to simulate, as well as the intended balance between accuracy and computational performance.

4.1.1.1.1 Radar models

Depending on the application, automotive radars can be modeled on various levels, providing raw Radar Data Cubes, point-detections, or even output of a sensor-specific object detection and tracking algorithm, e.g., in the form of bounding boxes. In this section, I will focus mostly on the point-detections level, given that this type of data is often used for further processing and/or fusion with other sensors.

Models of the automotive radar sensors often aim to re-cerate errors related to the undesired reflections of the radar wave, such as mirroring, or multi-path errors. To achieve this, wave reflections are typically approximated using ray casting methods [57]. To estimate the strength of wave reflections, as well as the expected distribution of the detections on road users and static obstacles, simplified geometric models are often utilized [67]. Alternatively, the radar wave scattering centers of a given object (e.g., vehicle) can be estimated a priori, e.g., using ray casting with a detailed geometric model [152] for later use in the simulation.

Ray-casting methods that take into account the geometries and material properties of the relevant objects tend to be computationally expensive. In order to achieve satisfactory performance in setups with stricter time and/or computational power constraints, methods based on various machine learning techniques are often proposed as an alternative. Machine-learning models utilized for the simulation of the radar's errors may be trained on a labeled dataset. The authors of [122] present a comparison of various ML models used for this task. Wheeler et al. [193] present an application of deep learning for stochastic simulation of radar sensors, taking into account both static and dynamic environmental features.

4.1.1.1.2 Camera models

While methods used for modeling low-level radar sensors vary significantly, most camera-based perception models involve rendering the simulated 3D scene from the sensor perspective, applying optional filters and distortions, and running the perception model itself [86].

The environment in proximity of the ego vehicle in this method is represented using detailed 3D models of the relevant static objects and traffic participants [177]. The scene is then rendered with the use of rasterization or ray-tracing techniques.

Rasterization methods allow rendering of complex scenes quickly with relatively limited computational resources, and thus most commonly find applications in 3D game engines, where performance is one of the key requirements. As a drawback, rasterization techniques require certain complex pre-computations and are not as general as ray-tracing methods.

Ray-tracing methods, while significantly slower, accurately simulate complex lighting effects even in a heavily cluttered environment, where multiple light bounces between objects' surfaces with varied light scattering properties. For this reason, these techniques often are utilized in applications that do not require real-time rendering performance, e.g., in the movie industry. However, it should be noted that the growing popularity of specialized hardware (graphic cards with dedicated ray-tracing cores) significantly improves the computation speed of the ray-tracing rendering techniques, enabling their use in real-time applications.

2D renderings of the ego's surroundings are typically further processed to take into account the lens distortions [101], as well as apply the required color space conversions [102].

One significant downside of approaches based on the rendering techniques is that the rendered scenes must have a high level of realism to ensure that the object detection algorithm, more often than not trained on datasets composed of real images, would have similar performance on the synthetic data. Achieving such realism requires a wide variety of detailed 3D models, high-resolution textures, and realistic simulation of weather conditions. Both the preparation of such a simulation and its execution tend to be time-consuming and costly.

High-level models are one of the possible alternatives that may greatly reduce the complexity and computational requirements of sensor modeling. Such models typically operate on object lists, similarly to the high-level radar sensor models [120, 45].

4.1.1.2. Applications in Reinforcement Learning

Applications of sensor models in the automotive industry are closely connected to simulation-based testing and validation, as well as to training machine learning-based algorithms that require realistic road situations samples with corresponding sensor data. While in many cases testing and learning needs can be fulfilled with the use of pre-recorded sensor streams, together with reference data that describe the actual state of the environment around the test vehicle, it is not always a viable solution. Testing AD/ADAS algorithms that control the vehicle in actual traffic in the early stages of development may pose a danger to other road users and tend to be prohibitively expensive to perform on a large scale. Certain algorithms, such as Reinforcement Learning, utilize extensive trial-and-error attempts in their learning process, which cannot be safely executed outside the simulation environment.

The problem of insufficient realism of the simulated environments used for training RL algorithms is widely known and commonly addressed as the Sim-to-Real Gap or Sim-to-Real Transfer problem [203]. Since RL-based control algorithms are most widely trained using a physical simulation, they are susceptible to erroneous behaviors in the real world caused by simplifications and inaccuracies of the simulation environments, that may fail to provide a sufficiently close re-creation of the targeted real-world environment. This limitation of RL-based algorithms is often listed as one of the main safety issues that limit their wider adaptation in the robotics and automotive industries [42].

One of the most commonly proposed approaches to preventing the issues caused by the sim-to-real gap is the use of domain randomization methods. This type of method is based on the assumption that the agent trained to operate properly in the presence of a wide spectrum of randomized modifications of the simulation environments should be also able to operate properly in the real world, as long as the distribution of modifications includes errors caused by simplifications of the simulation [175].

Domain randomization techniques are widely used in systems with camera-based perception modules that fulfill tasks such as object detection or semantic segmentation. The simulated environment can be easily rendered with a wide spectrum of modifications that include changes in lighting conditions, object texture, colors, and camera properties, as well as the addition of random noise or blur filters [55]. This approach has been proven to be effective in alleviating sim-to-real transfer problems in certain robotics tasks [174].

Randomization techniques can also be applied to the physical properties of simulated objects, by modifying their surface friction properties, locations, or geometries [8].

Although it could be argued that the randomization methods successfully used for systems with camera-based perception systems could be applied to the training of RL-based vehicle control algorithms that utilize camera-based perception, data on the applicability of domain randomization methods to the systems with radar-based or multi-sensor fusion perception systems is sparse. Atypical error patterns caused by both the physical properties of the radar wave and the limitations of perception, tracking, and fusion algorithms may not be sufficiently represented by the random error distributions often used in domain randomization techniques.

4.1.2. Motivation and General Idea

Reinforcement Learning appear to be one of the most promising approaches to vehicle behavior and trajectory planning. Its reliance on massive amounts of closed-loop simulation driving creates a need for efficient sensor modeling techniques to ensure the resulting system's robustness to sensor errors. Although many sensor modeling techniques have already been proposed for simulation-based testing of ADAS and AD systems, they rarely provide the faster than real-time performance required for efficient training of neural networks. Simpler domain randomization techniques frequently utilized in RL-based systems training, on the other hand, fail to capture the complex nature of automotive perception errors.

In this chapter, I propose a set of sensor models that are intended to be used for RL training applications, closing the gap between specialized high-fidelity models and simple domain randomization techniques. While the models are generic in their nature and designed to be easily applicable for various AD/ADAS systems and different types of sensors, their application is presented in an example of an RL-based direct control AD system (further described in section 4.5) equipped with 360-degree short range radar-based objects perception system and a front camera with the lane detection system.

Due to the immense amount of simulations required for most RL training techniques, the models must remain simple and computationally efficient, yet at the same time should be able to mimic common error patterns observed in automotive perception systems. Calibration parameters should allow simple adaptation of the proposed models to the different setups and sensor characteristics.

4.1.3. Contributions

Considering the restrictions of existing techniques and motivations discussed in the previous section, the primary contributions of the methods proposed in this chapter are listed below.

- The proposal of an efficient model of an arbitrary dynamic objects detection system. The proposed model simulates objects' state estimation errors, false positive detection errors, false negative detection errors, as well as object detection delays.

- Model of a lane marker detection perception system capable of simulating geometry estimation errors, viewing distance limitations, and false negative detection errors. The model can take into account lane occlusions, simulating their impact on the perception quality and false negative detection error occurrence probability.

- The proposal of a RL-based driving policy. The policy is capable of performing a highway driving task efficiently (faster than real-time). The underlying neural network has been trained with the use of the proposed sensor models, resulting in robust behavior in the presence of various types of perception errors.

- Analysis of an impact of the use of sensor models in RL training on the robustness of the trained driving policy. The trained policy has been compared with policies trained based on ground-truth sensors data, and baseline stochastic sensor models resembling existing domain randomization techniques. A set of simulation experiments involving highway driving as well as a set of predefined test scenarios has been used to provide insight into the impact of sensor models and domain randomization on the robustness of the policy.

The methods presented in this chapter are partially based on my previous work [179], with several notable extensions, including, but not limited to, extended evaluation and performance analyses.

4.2. Problem Formulation

The main role of sensor models discussed in this chapter is to utilize traffic simulation ground truth information to generate synthetic data about dynamic objects and lane markers in proximity to the ego vehicle, as perceived by its perception systems. This allows for the creation of a Software-in-Loop testing and training setup, where the AD/ADAS algorithm controls an ego vehicle in a simulated traffic environment, performing decisions based on synthetic sensor data.

There is no common agreement on the interfaces used to interact with the traffic simulation software, even though certain standardization attempts were made, the most notable example being the Open Simulation Interface (OSI) [51]. Depending on the intended use, typically either a subset of OSI or application-specific interfaces is utilized.

Sensor models described in this chapter are split into two subsystems: dynamic environment models, which utilize a list of traffic participants (e.g., vehicles) in the proximity of the ego with their properties, and static environment models, which use a list of lane markers, with their types and description of their geometries. There are no substantial differences between the sensor models' input and output interfaces.

As both dynamic and static models operate on lists of entities (vehicles, lane markers) with their respective parameters (such as geometry or position), both models will use a similar notation.

The simulation package provides information about the environment in discrete time steps, consisting of n_{class} sets of environmental features of distinct classes $c \in \mathfrak{C}$, including, e.g., dynamic object detection and lane marker detection classes. At each time step t, the environmental features are represented as sets $\mathcal{S}_c(t) = \{\mathbf{s}_{ci}(t)\}_{i=1..n_c}$ of n_c state vectors $\mathbf{s}_{ci}$, where $\mathbf{s}_{ci} \in \mathbb{R}^{n_{\mathbf{s}_c}}$. Note that the size of the state vector $n_{\mathbf{s}_c}$ may vary between object types, and the number of objects n_c in each set is not constant and may vary over time.

Perception, as a process of acquiring information from sensors and optionally processing it further (e.g., filtering, tracking), can be modeled as a generation of an objects' set estimate $\hat{\mathcal{S}}_c = \{\hat{\mathbf{s}}_{cj}\}_{j=1..n_{ce}}$. The set $\hat{\mathcal{S}}_c$ describes n_{ce} objects registered by a perception stack, where each object is described by a state estimate $\hat{\mathbf{s}}_c \in \mathbb{R}^{n_{\hat{\mathbf{s}}_c}}$.

Following the notation proposed in [52], I describe the sensing process of features of a class c using a mapping:

$$M\left(\mathbf{p}_c\right) : \left\{\mathbf{s}_c\right\}_{i=1..n_c} \to \left\{\hat{\mathbf{s}}_{cj}\right\}_{j=1..n_{ce}}, \tag{4.2.1}$$

where $\mathbf{p}_c$ is a vector of parameters that have an impact on a sensing process related to a feature class c, describing e.g., sensor properties, weather conditions, etc.

In general, the sensing process does not preserve the number of objects, as false positive and false negative detection errors can impact the number of perceived objects, and thus $n_c \neq n_{ce}$.

Further following the notation proposed in [52], the sensing model M can be composed of an arbitrary number n_m of subsequent mapping operations $M^{(k)}$ for $k = 1..n_m$.

The resulting sensor model is given by the following equation:

$$M(\mathbf{p}) = M^{(n_m)}(\mathbf{p}_{n_m}) \circ ... \circ M^{(2)}(\mathbf{p}_2) \circ M^{(1)}(\mathbf{p}_1) \tag{4.2.2}$$

where $\mathbf{p}_n$ is a parameters vector that impact the the n-th mapping operation $M^{(n)}$. The mapping $M^{(n)}$ is defined as:

$$M^n(\mathbf{p}_n) : \{\hat{\mathbf{s}}_{ci}\}^{(k-1)}_{i=1..n_{k-1}} \to \{\hat{\mathbf{s}}_{cj}\}^{(k)}_{j=1..n_k}, \tag{4.2.3}$$

where n_k denotes the number of objects after the k-th mapping operation. It can be noted that $\{\mathbf{s}_{ci}\}^{(1)}_{i=1..n_c} = \mathcal{S}_c$ (first mapping $M^{(1)}(\mathbf{p_1})$ maps the ground-truth environment description, or the simulator output, to a certain set $\{\mathbf{s}_{ci}\}^{(1)}_{i=1..n_1}$), and the output of the sensing process is given by the result of the last mapping operation, so $\{\hat{\mathbf{s}}_{cj}\}^{(k)}_{j=1..n_k} = \hat{\mathcal{S}}_c$.

A significant subset of error patterns observable in perception systems, especially those that filter their output in time, is time-correlated. Modeling them requires information about previous states of the environment and/or the use of the internal sensor state, which would enable preservation of certain information between the execution of the algorithm on the subsequent time samples. To achieve this, previous inputs and outputs of the sensor model, as well as additional variables that describe the internal state of the model, can be included in the parameter vectors $\mathbf{p}_i$ for $i = 1..n_m$.

4.3. Dynamic Objects Detection Model

The dynamic environment perception model utilizes object-level representation of traffic participants in proximity to the ego vehicle to simulate a sensing stack composed of multiple sensors, the output of which is fused and filtered in a central unit. One possible realization of such a system often utilized in commercial vehicles is a set of Phase-Modulate Continuous Wave (PMCW) Radars that operate in the millimeter range of the electromagnetic waves spectrum. As an example, radar-based AD/ADAS systems that utilize four short-range radars placed in each corner of the vehicle to provide a 360-degree environment model can be considered. In certain applications, additional long-range front radar or front-facing camera is added to provide additional information about the vehicles in front of the ego, critical for Automatic Cruise Control (ACC) and Automatic Braking (AB) applications.

Fusion algorithms, often based on Kalman filter concepts, gather information from all sensors, often in the form of object lists, to produce a single time-filtered object list.

The dynamic perception sensor model described in this section is intended to mimic such a system, simulating the following limitations and error patterns observed in similar systems:

- **Range limitations.** All dynamic environment sensor types have effective range limitations, above which objects cannot be reliably detected.
- **Occlusions.** Most sensor types cannot detect an object that is occluded by other objects or static obstacles. While certain exceptions from this rule exist (e.g., radars are sometimes able to detect an object occluded by a barrier or other vehicle based on radar wave's ground reflections), they rarely can be utilized in a reliable manner.
- **False positive detections.** Detection of non-existing objects in unoccupied areas, often observed in radar-based systems due to, e.g., multipath reflections.
- **False negative detections.** Missing detections, often caused by adverse weather conditions, or an object's unfavorable Radar Cross Section (RCS) in radar-based systems.
- **State estimation errors.** Errors in the estimation of object's state (e.g., geometry, velocity, orientation). It should be noted that sometimes severe state estimation errors can be indistinguishable from a pair of false positive and false negative detection errors.

The ground truth information about the dynamic environment produced by a simulation package in time t is described by a set $\mathcal{S}_\mathrm{d}(t) = \{\mathbf{s}_{\mathrm{d}i}(t)\}_{i=1..n_d}$. State of each of n_d relevant objects in is represented by a state vector $\mathbf{s}_{\mathrm{d}i}$ for $i = 1..n_d$ consisting of following state variables:

$$\mathbf{s}_{\mathrm{d}_i} = \begin{bmatrix} \mathbf{g}_i \\ \mathbf{x}_i \\ \psi_i \\ v_i \\ a_i \end{bmatrix}. \tag{4.3.1}$$

In the above equation, $\mathbf{g}_i \in \mathbb{R}^2$ represents the geometry of the i-th object, in particular the length and width of its bounding box. Vector $\mathbf{x}_i \in \mathbb{R}^2$ is the object's position in a Cartesian coordinate system, $\psi_i \in \mathbb{R}$ denotes its rotation (with respect to the coordinate system's origin), while $v_i \in \mathbb{R}$ and $a_i \in \mathbb{R}$ describe the object's absolute velocity and acceleration, respectively.

The dynamic perception model performs the mapping:

$$M^{\mathrm{dyn}}(\mathbf{p}_d) : \{\mathbf{s}_{\mathrm{d}i}\}_{i=1..n_\mathrm{d}} \rightarrow \{\hat{\mathbf{s}}_{\mathrm{d}j}\}_{j=1..n_\mathrm{de}}, \tag{4.3.2}$$

producing an estimate of the dynamic environment's state $\hat{\mathcal{S}}_\mathrm{d} = \{\hat{\mathbf{s}}_{\mathrm{d}j}\}_{j=1..n_\mathrm{de}}$ composed of n_de object's state estimates $\hat{\mathbf{s}}_{\mathrm{d}j} = [\hat{\mathbf{g}}_j,\ \hat{\mathbf{x}}_j,\ \hat{\psi}_j,\ \hat{v}_j,\ \hat{a}_j]^T$ for $j = 1..n_\mathrm{de}$. The perception system is modeled through a set of operations, described in following sections.

4.3.1. Range Limitations and Occlusions

Range limitations and occlusions model is described as $M^\mathrm{d,occ}(p_\mathrm{d,occ})$ mapping operation:

$$M^\mathrm{d,occ}(\mathbf{p}_\mathrm{d,occ}) : \{\mathbf{s}_{\mathrm{d}i}\}_{i=1..n_\mathrm{d}} \to \left\{\mathbf{s}_{\mathrm{d,occ}_j}\right\}^{(\mathrm{d,occ})}_{j=1..n_\mathrm{d,occ}}, \tag{4.3.3}$$

where the parameter vector $\mathbf{p}_\mathrm{d,occ}$ describes the sensor's range limitation, detection angle (if relevant), and the occlusion level above which the objects are filtered out. $n_\mathrm{d,occ}$ is a number of unoccluded objects in the detection area of a sensor.

Occlusion filtering is typically performed in a 2D projection of the environment, using rectangular bounding boxes for the relevant objects. Both range limitation and occlusion filtering operations are commonly available in the simulation packages and can be calculated using relatively trivial trigonometric operations, and thus will not be described in detail in this section.

4.3.2. False Negative Detection Errors

False negative object detection errors may be caused by several different phenomena, but for the purpose of sensor modeling, they can be categorized into two main types: detection delays and random disappearances.

The detection delays refer to the time delay between the object's appearance in an unoccluded detection area and its addition to the perception output objects list. One source of detection delays is related to the system's properties, such as the internal communication network's latency and the perception algorithm's execution time. Second, a more significant source of stochastic detection delays is related to the design of the tracking and fusion algorithms. In order to filter out possible false positive errors, such algorithms often require gathering sufficient evidence that objects, in fact, exist, before they will be added to the objects list. Such evidence may include the detection of the object's existence in several subsequent time instances, confirmation from several sensors, or acquisition of detection with a sufficient confidence level (e.g., the sufficient signal-to-noise ratio in radar systems). The time needed to gather this evidence may be difficult to predict, resulting in potentially significant detection delays.

In order to model such delays, each newly observed object that entered the unoccluded detection area of the sensors (so each new object in the set $\{\mathbf{s}_{\mathrm{d,occ}_j}\}^{(\mathrm{d,occ})}_{j=1..n_\mathrm{d,occ}}$) is assigned a random detection delay T_{delay_i} for $i = 1..n_\mathrm{d,occ}$. The delay value is sampled from a normal distribution:

$$T_{\text{delay}_i} = \max\left(p_{\mu,\text{delay}}, |\sim \mathcal{N}(0, p^2_{\sigma,\text{delay}})|\right), \tag{4.3.4}$$

configured by $p_{\mu,\text{delay}}$ and $p_{\sigma,\text{delay}}$ calibration parameters.

False negative detection errors may also affect already detected objects, for example, due to partial occlusions, weak radar reflections in certain orientations, or weather conditions. Such disappearances of objects from the perception output are modeled as random drops with a certain duration assigned. At each sensing update, each of the objects from the set $\{\mathbf{s}_{\text{d,occ}_j}\}^{(\text{d,occ})}_{j=1..n_{\text{d,occ}}}$ can be marked as a false negative (and removed from the subsequent mapping output) with probability $p_{\text{d,fn,prob}}$. The object newly marked as a false negative is assigned a duration $T_{\text{d,fn,dur}} = \max\left(p_{\mu,\text{fn,dur}}, |\sim \mathcal{N}(0, p^2_{\sigma,\text{fn,dur}})|\right)$ for which it will remain removed from the output objects list. $p_{\mu,\text{delay}}$, $p_{\sigma,\text{delay}}$, $p_{\text{d,fn,prob}}$, $p_{\mu,\text{fn,dur}}$, $p_{\sigma,\text{fn,dur}}$ are the calibration parameters included in the $\mathbf{p}_{\text{d,fn}}$ parameters vector.

Additional state variables are required to keep track of false negatives and their duration, i.e. detection delays T_{delay}, detection durations $T_{\text{d,fn,dur}}$, timestamps that allow computing the current duration of a given false negative, and false negative flags are also included in the parameter vector.

All operations related to false negatives are described as a mapping:

$$M^{\text{d,fn}}(\mathbf{p}_{\text{d,fn}}) : \left\{\mathbf{s}_{\text{d,occ}_i}\right\}^{(\text{d,occ})}_{i=1..n_{\text{d,occ}}} \to \left\{\mathbf{s}_{\text{d,fn}_j}\right\}^{(\text{d,fn})}_{j=1..n_{\text{d,fn}}}. \tag{4.3.5}$$

4.3.3. False Positive Detection Errors

False positive detection errors pose a significant hazard in AD/ADAS, being able to falsely trigger various emergency responses. False positive errors are especially problematic in radar-based systems. False detections caused by multipath reflections of the radar wave are difficult to distinguish from the real objects - especially since they tend to exhibit similar characteristics to the objects that caused them.

False positives are modeled by random injection of the non-existing objects, that persist for a certain amount of time. The persistence of the objects for several time frames is often the case even in camera-based systems, as tracking and fusion algorithms often artificially prolong the existence of the tracked objects to avoid false negative errors, updating their state according to simplified motion models.

In the proposed model, at each sensing update, a single false positive object can be randomly introduced with a probability $p_{\text{fp,prob}}$. Newly created false positive detections are assigned a duration $T_{\text{d,fp,dur}} = \max\left(p_{\mu,\text{fp,dur}}, |\sim \mathcal{N}(0, p^2_{\sigma,\text{fp,dur}})|\right)$, for which they will persist, and their state $\mathbf{s}_{\text{d,fp}}$ is sampled from the following vector of normal distributions:

$$\mathbf{s}_{\mathrm{d,fp}} = \begin{bmatrix} \mathbf{g} \sim \mathcal{N}_2(\mu_g, \Sigma_g) \\ \mathbf{x} \sim \mathcal{N}_2(\mu_x, \Sigma_x) \\ \psi \sim \mathcal{N}(\mu_\psi, \sigma_\psi) \\ v \sim \mathcal{N}(\mu_v, \sigma_v) \\ a \sim \mathcal{N}(\mu_a, \sigma_a) \end{bmatrix}, \tag{4.3.6}$$

where μ_g, Σ_g, μ_x, Σ_x, μ_ψ, σ_ψ, μ_v, σ_v, μ_a, and σ_a denote the calibration parameters. All parameters are stored in a vector $\mathbf{p}_{\mathrm{d,fp}}$.

During false positive detections' existence, their stat is updated according to a basic constant acceleration model, i.e., objects move in the direction coincident with their orientation keeping the initial acceleration. The current state of the object, as well as its duration of existence, are stored in the parameter vector $\mathbf{p}_{d,fp}$ as well.

All operations related to the false positive detection errors, i.e., their creation, state updates, and removal are described with a mapping:

$$M^{\mathrm{d,fp}}(\mathbf{p}_{\mathrm{d,fp}}) : \left\{\mathbf{s}_{\mathrm{d,fn}_i}\right\}_{i=1..n_{\mathrm{d,fn}}}^{(\mathrm{d,fn})} \rightarrow \left\{\mathbf{s}_{\mathrm{d,fp}_j}\right\}_{j=1..n_{\mathrm{d,fp}}}^{(\mathrm{d,fp})}. \tag{4.3.7}$$

4.3.4. State Estimation Errors

State estimation errors are common in all automotive perception systems, due to the physical limitations of the sensors, as well as the imperfect nature of the detection, tracking, and fusion algorithms. This type of errors in automotive systems tends to be time-correlated, as not only do the environmental triggers that cause the errors tend to persist for multiple sensing updates, but also the tracking and fusion algorithms usually have filtering properties that introduce time correlations.

Thus, state estimation errors are simulated using a stochastic process that allows one to mimic the time-correlated nature of the errors. In particular, a multivariate stochastic model based on the Ornstein-Uhlenbeck process is used for the update of the state estimate values of each observed object. At each time update i, each n_c-dimensional chunk $\hat{\mathbf{s}}_c$ of the state estimate vector is calculated according to the following model:

$$P_{OU_c}(\hat{\mathbf{s}}_c^{(i)}|\mathbf{s}^{(i)}, \hat{\mathbf{s}}^{(i-1)}, \mathbf{p}_{s_c}) = \begin{cases} \mathbf{p}_{\mathrm{ou},\lambda_c} * (\mathbf{s}^{(i)} - \hat{\mathbf{s}}^{(i-1)}) * dt + \mathbf{W}^{(i)} * dt, & \text{for } i > 1 \\ \mathcal{N}_{n_c}(\mathbf{s}^{(i)}, \mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{init}_c}), & \text{for } i = 0, \end{cases} \tag{4.3.8}$$

where the ground truth value of the state vector chunk is denoted as $\mathbf{s}^{(i)} \in \mathbb{R}^{n_c}$, the previous value of the state estimate is denoted as $\hat{\mathbf{s}}^{(i-1)} \in \mathbb{R}^{n_c}$, dt denotes the time between perception updates, and $\mathbf{W}_i \in \mathbb{R}^{n_c}$ is a random variable, which is sampled according to a multivariate normal distribution $\mathbf{W}_i \sim \mathcal{N}_{n_c}(\mathbf{0}, \mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{u}_c})$. The process can be calibrated using the parameters: $\mathbf{p}_{\mathrm{ou},\lambda_c}$, $\mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{init}_c}$ and $\mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{u}_c}$, gathered in a parameter vector $\mathbf{p}_{s_c}$.

Operations that constitute the update of the objects' state estimates are described with a mapping:

$$M^{\text{state,est}}\left(\mathbf{p}_{\text{state,est}}, \hat{\mathbf{s}}^{(t-1)}_{\text{state,est}}\right) : \left\{\mathbf{s}_{\mathrm{d,fn}_i}\right\}^{(\mathrm{d,fn})}_{i=1..n_{\mathrm{d,fn}}} \rightarrow \left\{\hat{\mathbf{s}}^{(t)}_{\text{state,est}_j}\right\}^{(\text{state,est})}_{j=1..n_{\text{state,est}}}, \tag{4.3.9}$$

where $\hat{\mathbf{s}}^{(t)}_{\text{state,est}_j}$ is computed according to the following mapping:

$$\hat{\mathbf{s}}^{(t)}_{\text{state,est}_j} = P_{OU}(\hat{\mathbf{s}}^{(\mathbf{t})}|\mathbf{s}^{(t)}, \hat{\mathbf{s}}^{(t-1)}, \mathbf{p}_{\text{state}}), \tag{4.3.10}$$

where $\mathbf{p}_{\text{state,est}}$ is the calibration vector that includes OU process calibration matrices: $\mathbf{p}_{\mathrm{ou},\lambda}$, $\mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{init}}$ and $\mathbf{p}_{\mathrm{ou},\Sigma,\mathrm{u}}$.

4.3.5. Complete Model of the Dynamic Environment Perception

The model is composed of all previously described operations: simulation of detection area limitations, occlusions, false negative and positive errors, and state estimate errors. All these operations can be described as mapping:

$$M_d(\mathbf{p}) = M^{(\text{state,est})}(\mathbf{p}_{\text{state,est}}) \circ M^{(\mathrm{d,fp})}(\mathbf{p}_{\mathrm{d,fp}}) \circ M^{(\mathrm{d,fn})}(\mathbf{p}_{\mathrm{d,fn}}) \circ M^{(\mathrm{d,occ})}(\mathbf{p}_{\mathrm{d,occ}}). \tag{4.3.11}$$

The values of the parameters used for model calibration depend on the particular configuration of sensors and perception algorithms used in the ego vehicle. An exemplary set of values used for experiments presented in the following sections is shown in Table 4.1.

Table 4.1. Values of sensor models calibration parameters used in the experimental setup.

Parameter Name	Value	Unit	Description
$p_{\mu,\text{delay}}$	0.3	s	Mean object detection delay
$p_{\sigma,\text{delay}}$	0.55	s	Object detection delay standard deviation
$p_{\text{d,fn,prob}}$	0.001	-	Probability of false negative object detection
$p_{\mu,\text{fn,dur}}$	1.47	s	Mean duration of false negative object detection
$p_{\sigma,\text{fn,dur}}$	1.5	s	False negative object detection duration standard deviation
$p_{\text{d,fp,prob}}$	0.0175	-	Probability of false positive object detection
$p_{\mu,\text{fp,dur}}$	0.5	s	Mean duration of false positive object detection
$p_{\sigma,\text{fp,dur}}$	2.8	s	False positive object detection duration standard deviation
μ_q	$[4.34, 1.89]^T$	-	Mean false positive object size
Σ_q	$\begin{bmatrix} 0.21 & 0 \\ 0 & 0.01 \end{bmatrix}$	-	False positive object size covariance matrix
μ_x	$[45.1, 0]^T$	-	Mean false positive object position
Σ_x	$\begin{bmatrix} 19.3 & 0 \\ 0 & 0.97 \end{bmatrix}$	-	False positive object position covariance matrix
μ_ψ	0.0	-	Mean rotation of false positive object
σ_ψ	0.44	-	Standard deviation of false positive object rotation
μ_v	0.0	$\frac{\text{m}}{\text{s}}$	Mean speed of false positive object relative to ego
σ_v	11.7	$\frac{\text{m}}{\text{s}}$	Standard deviation of false positive object speed
μ_a	0.0	$\frac{\text{m}}{\text{s}^2}$	Mean acceleration of false positive object relative to ego
σ_a	3.46	$\frac{\text{m}}{\text{s}^2}$	Standard deviation of false positive object speed
$\mathbf{p}_{\text{ou},\lambda}$	diag(0.5, 0.65, 0.11, 0.45, 0, 0.5, 0)	-	State estimation noise parameter
$\mathbf{p}_{\text{ou},\Sigma,\text{init}}$	diag(1.3, 1.0, 1.4, 0.7, 0, 2.2, 0)	-	State estimation noise parameter
$\mathbf{p}_{\text{ou},\Sigma,\text{u}}$	diag(2.0, 1.6, 1.3, 0.7, 0, 2.5, 0)	-	State estimation noise parameter

4.4. Lane Markers Detection Model

Static environment perception can refer to the detection and tracking of various elements of the ego's surroundings, such as lane markers, road barriers, parked vehicles, debris, or elements of the road infrastructure.

In this section, I will focus on the elements that have the greatest impact on AD systems, which are the lane markers. Lane markers are one of the most important elements of the infrastructure that are utilized even in relatively basic ADAS features, such as Lane Centering or Automatic Cruise Control. However, it should be noted that both parked vehicles and static elements of road infrastructure may also play a significant role in such features - both can be represented in a way identical to the elements of the dynamic environment, that is, using the state vectors $\mathbf{s}_d$, with the acceleration and velocity set to 0.

Lane markers are usually detected by a camera-based sensor and thus are prone to errors and performance degradation related to bad weather conditions, such as rain or snow. Additionally, the detection of lane markers can be affected by unfavorable light reflections on a wet road, snow residues, complex road geometry, and wear of the markers themselves.

The model presented in this section is intended to simulate the following types of errors and performance degradations:

- **Range limitations.** Limited resolution of the cameras hinders their ability to detect farther objects, including the lane markers. For this reason, the perceived length of the lane markers is typically limited and tends to decrease on roads with complex geometries that create unfavorable perspective effects.

- **Occlusions.** Range is further limited on roads with heavy traffic, due to other traffic participants that may occlude the road markers.

- **False negative detections.** Adverse weather conditions and severe occlusions can lead to missing lane marker detections. Missing detections are often the case for markers that have already been partially occluded by other vehicles.

- **State estimation errors.** In the case of lane markers, state estimation errors most often refer to the incorrect estimation of the marker's geometry. The geometry error tends to grow with the longitudinal distance from the ego vehicle.

The lane markers are represented as a set $\mathcal{S}_r(t) = \{\mathbf{s}_{r_i}(t)\}_{i=1..n_{\mathrm{lm}}}$ of n_{lm} l-dimensional vectors $\mathbf{s}_{r_i} \in \mathbb{R}^l \ for \ i = 1..n_{\mathrm{lm}}$, which describe the geometry of the lane markers in the vicinity of the ego vehicle. The lane markers' geometries are defined as:

$$\mathbf{s}_{r_i} = \begin{bmatrix} \mathbf{c} \\ h \end{bmatrix}, \tag{4.4.1}$$

where $\mathbf{c} \in \mathbb{R}^4$ is a vector of polynomial coefficients $\mathbf{c} = [c_0,\ c_1,\ c_2,\ c_3]$ that define the lateral offset d of a lane marker as a function of the longitudinal distance s from the ego vehicle: $d(s) = c_3 * s^3 + c_2 * s^2 + c_1 * s + c_0$. h is the longitudinal length of the observed lane marker.

The static environment model uses a ground truth of lane markers $\mathcal{S}_{r_i}(t)$ generated by the simulator, and produces an estimate of the road model $\hat{\mathcal{S}}_r$, which includes estimates of the state of the lane markers $\hat{\mathbf{s}}_{r_i}(t) = \left[\hat{\mathbf{c}}_{\mathbf{i}}(t), \hat{h_i}(t)\right]^T$.

4.4.1. Range Limitations and Occlusions

The lane marker detection range limitation model limits the observed marker length in a stochastic manner, emulating the difficulties of detecting markers at large distances observed in camera-based sensors. Range limitation directly impacts the lane marker geometry estimation - in further steps of the processing, the lane marker geometry is estimated based on observed unoccluded parts of the ground truth marker.

The range limitation is also closely related to the occlusions. If a large part of the marker is occluded, the observed length is often limited to the distance from the ego vehicle to the appearance of the occlusion.

Occlusions of the ground truth lane markers are calculated on the basis of discrete samples of the lane markers' geometry. Each ground truth lane marker $\mathbf{s}_{r_i}$ is sampled uniformly to a set of samples $\mathbf{s}_{i_j} = [x_s, y_s]^T$, where x_s denotes the longitudinal position of a sample and y_s its lateral position, both in the Cartesian Vehicle Coordinates System (VCS).

The sampling operation can be described as a mapping:

$$M^{(\mathrm{r,smpl})}(\mathbf{p}_{\mathrm{r,smpl}}) : \{\mathbf{s}_{r_i}(t)\}_{i=1..n_{\mathrm{lm}}} \rightarrow \left\{\left\{\mathbf{z}_{ij}^{\mathrm{r,smpl}}\right\}_{j=1..n_{\mathrm{r,smpl}}}\right\}_{i=1..n_{\mathrm{lm}}} \tag{4.4.2}$$

where $\mathbf{z}_{ij}^{\mathrm{r,smpl}}$ is the j-th sample of i-th lane marker, and $\mathbf{p}_{\mathrm{r,smpl}}$ is a parameter vector that allows configuring the number of samples to be gathered and the maximum sampling distance.

The next operation is the removal of the occluded samples. The occlusion is evaluated on the basis of a 2D projection of bounding boxes of other vehicles, making the occlusion check trivial to perform. Additionally, if at least n_{cons} consecutive samples in a single marker are occluded, the rest of the samples (farther from the ego vehicle) are discarded as well. A mapping $M^{(\mathrm{r,occ})}(\mathbf{p}_{\mathrm{r,occ}})$ consists of the execution of occluded samples removal operation for each of the lane markers:

$$M^{(\mathrm{r,occ})}(\mathbf{p}_{\mathrm{r,occ}}) : \left\{\left\{\mathbf{z}_{ij}^{\mathrm{r,smpl}}\right\}_{j=1..n_{\mathrm{r,smpl}}}\right\}_{i=1..n_{\mathrm{lm}}} \rightarrow \left\{\left\{\mathbf{z}_{ij}^{\mathrm{r,occ}}\right\}_{j=1..n_{\mathrm{r,occ}_i}}\right\}_{i=1..n_{\mathrm{lm}}} \tag{4.4.3}$$

where $\mathbf{p}_{\mathrm{r,occ}}$ is the calibration vector.

The distance of the farthest sample of each marker is considered its base length used for subsequent range limitation operations and marked h_{gt}.

The marker length value is calculated using a stochastic model inspired by an Ornstein-Uhlenbeck process, similar to the model used in state estimate errors for dynamic objects. The model is defined as follows:

$$P_h(\hat{h}^{(t)}|h_{\text{gt}}^{(t)}, h_{\text{gt}}^{(t-1)}, \hat{h}^{(t-1)}, \mathbf{p}_h) = \begin{cases} \left(p_{\text{lm,h},\lambda}\left(\left(h_{\text{gt}}^{(t)} - p_{\text{lm,lim}}\right) - \hat{h}^{(t-1)}\right) + W^{(t)}\right) * dt \\ \qquad \text{if } t > 0 \text{ and } \left(h_{\text{gt}}^{(t)} - h_{\text{gt}}^{(t-1)}\right) \geq p_{\text{lm,jump}} \\ \mathcal{N}\left(h_{\text{gt}}^{(t)} - p_{\text{lm,lim}}, p_{\text{lm,h},\sigma}\right) \\ \qquad \text{otherwise,} \end{cases} \tag{4.4.4}$$

where $\mathbf{p}_h$ is the calibration vector consisting of the calibration parameters $p_{\text{lm,h},\lambda}$, $p_{\text{lm,lim}}$, $p_{\text{lm,jump}}$, and $p_{\text{lm,h},\sigma}$. It should be noted that according to the above equation, the process is reset if the perceived length h_{gt} of the ground truth marker experiences a sudden drop, i.e., $\left(h_{\text{gt}}^{(t)} - h_{\text{gt}}^{(t-1)}\right) \geq p_{\text{lm,jump}}$. After the reset, the process is initialized by drawing the length value from a normal distribution. This mechanism is implemented to enable severe length limitations during sudden occlusions.

Since the final representation of the lane markers $\hat{\mathbf{s}}_{r_i}(t) = \left[\hat{\mathbf{c}}_{\mathbf{i}}(t), \hat{h}_i(t)\right]^T$ uses cubic polynomials to represent their geometry, the samples $\{\mathbf{z}_{ij}^{\text{r,occ}}\}_{j=1..n_{\text{r,occ,i}}}$ are used to find an approximation of the polynomial $\hat{\mathbf{c}}_i(t)$ using the least squares method. This approximation is denoted as a mapping:

$$M^{(\text{r,lsa})} : \left\{\left\{\mathbf{z}_{ij}^{\text{r,occ}}\right\}_{j=1..n_{\text{r,occ}_i}}\right\}_{i=1..n_{\text{lm}}} \rightarrow \left\{\mathbf{s}_i^{(r,lsa)}\right\}_{i=1..n_{\text{lm}}} \tag{4.4.5}$$

Note that since the approximation is performed using only already filtered-out samples, this operation at the same time emulates the geometry estimation performance degradation in presence of occlusions - i.e., loss of information about the markers' geometry in occluded areas.

4.4.2. False Negative Detection Errors

The false negative model is responsible for the random removal of the lane markers from the set $\left\{\mathbf{s}_i^{(\text{r,lsa})}\right\}_{i=1..n_{\text{lm}}}$. The model is intended to implement the intuition that unoccluded lane markers in close proximity to the ego vehicle have a higher chance of being detected than markers that are heavily occluded or are placed farther from the vehicle laterally. Removal of the lane markers is performed in a mapping:

$$M^{\text{r,fn}}(\mathbf{p}_{\text{r,fn}}) : \left\{\mathbf{s}_i^{\text{r,lsa}}\right\}_{i=1..n_{\text{lm}}} \rightarrow \left\{\mathbf{s}_j^{\text{r,fn}}\right\}_{j=1..n_{\text{r,fn}}}. \tag{4.4.6}$$

The mapping takes into account the lateral offset of a lane marker, or rather a number of markers $o \in \mathbb{N}$ between the ego and the marker, with $o = 0$ corresponding to the markers delimiting the lane currently occupied by the ego vehicle.

The probability P_{discard} that the lane marker will be discarded from a final set is defined as:

$$P_{\text{discard}_i}(\mathbf{p}_{\text{fn}}) = \begin{cases} p_{\text{lm,disc,c}_0} + p_{\text{lm,disc,l}_0} * \frac{h_{\max}-h_i}{h_{\max}} & \text{if } o = 0 \\ p_{\text{lm,disc,c}_1} + p_{\text{lm,disc,l}_1} * \frac{h_{\max}-h_i}{h_{\max}} & \text{if } o = 1 \\ p_{\text{lm,disc,c}_2} + p_{\text{lm,disc,l}_2} * \frac{h_{\max}-h_i}{h_{\max}} & \text{otherwise,} \end{cases} \quad \text{for } i = 1..n_{\text{lm}}, \tag{4.4.7}$$

where the vector $\mathbf{p}_{\text{fn}}$ is composed of calibration parameters $p_{\text{lm,disc,c}_0}$, $p_{\text{lm,disc,l}_0}$, $p_{\text{lm,disc,c}_1}$, $p_{\text{lm,disc,l}_1}$, $p_{\text{lm,disc,c}_2}$, and $p_{\text{lm,disc,l}_2}$. Additionally, the vector includes a value $h_{\max}$ that denotes the effective detection range of the static environment perception system, that is, the maximum length of the lane marker.

Lane markers assigned a false negative status may regain a true positive status at each perception update with a probability $P_{\text{lm,recovery}}$ calculated as follows:

$$P_{\text{lm,recovery}} = p_{\text{rec,hyst}} * (1 - P_{\text{discard}}) + \min(p_{\text{rec,pps}} * t_{\text{disc}}, p_{\text{rec,sat}}), \tag{4.4.8}$$

where t_{disc} denotes the time for which the lane marker had the false negative status, and $p_{\text{rec,hyst}}$, $p_{\text{rec,pps}}$, $p_{\text{rec,sat}}$ are the configuration parameters.

4.4.3. Geometry Estimation Errors

The geometry estimation error model is based on the Ornstein-Uhlenbeck-based processes $P_{OU}\left(\hat{\mathbf{c}}^{(t)}|\mathbf{c}^{(t)}, \hat{\mathbf{c}}^{(t-1)}, \mathbf{p_c}\right)$, with the calibration vector $\mathbf{p}_c$ including parameters $\mathbf{p}_{\text{lm,c},\lambda}$, $\mathbf{p}_{\text{lm,c},\Sigma_{\text{init}}}$ and $\mathbf{p}_{\text{lm,c},\Sigma_u}$, analogous to the one used for dynamic objects' state estimation errors modeling. In combination with the length model defined in Section 4.4.1, the complete geometry model can be defined as:

$$\mathbf{s}_j^{(t)} = \begin{bmatrix} P_{OU}(\hat{\mathbf{c}}^{(t)}|\mathbf{c}^{(t)}, \hat{\mathbf{c}}^{(t-1)}, \mathbf{p_c}) \\ P_{OU}(\hat{h}^{(t)}|h_{gt}^{(t)}, h_{gt}^{(t-1)}, \hat{h}^{(t-1)}, \mathbf{p}_h) \end{bmatrix} \text{ for } j = 1..n_{\text{r,fn}}. \tag{4.4.9}$$

Calibration vectors $\mathbf{p}_c$ and $\mathbf{p}_h$ are included in the calibration vector $\mathbf{p}_{\text{r,geom}}$.

With the geometry model denoted as a mapping:

$$M^{(\text{r,geom})}(\mathbf{p}_{\text{r,geom}}) : \left\{\mathbf{s}_i^{(\text{r,fn})}\right\}_{i=1..n_{\text{r,fn}}} \to \left\{\mathbf{s}_j^{(\text{r,geom})}\right\}_{j=1..n_{\text{r,fn}}}, \tag{4.4.10}$$

a complete model of a static environment perception system can be defined as:

$$M_{\text{static}}(\mathbf{p}_r) = M^{(\text{r,geom})} \circ M^{(\text{r,fn})} \circ M^{(\text{r,lsa})} \circ M^{(\text{r,occ})} \circ M^{(\text{r,smpl})}. \tag{4.4.11}$$

The model in the experiments presented in the following sections is calibrated according to the parameters in Table 4.2

Table 4.2. Calibration values used for lane markers perception model.

Parameter Name	Value	Unit	Description
$p_{\mathrm{lm,h},\lambda}$	0.4	-	Lane markers length noise parameter
$p_{\mathrm{lm,lim}}$	5.0	m	Mean lane markers length shortening
$p_{\mathrm{lm,jump}}$	15.0	m	Min lane markers length change for noise reset
$p_{\mathrm{h},\sigma}$	5.6	-	Lane markers length noise parameter
$p_{\mathrm{lm,disc,c_0}}$	0.001	-	Marker false negative probability parameter
$p_{\mathrm{lm,disc,l_0}}$	0.01	-	Marker false negative probability parameter
$p_{\mathrm{lm,disc,c_1}}$	0.01	-	Marker false negative probability parameter
$p_{\mathrm{lm,disc,l_1}}$	0.01	-	Marker false negative probability parameter
$p_{\mathrm{lm,disc,c_2}}$	0.02	-	Marker false negative probability parameter
$p_{\mathrm{lm,disc,l_2}}$	0.01	-	Marker false negative probability parameter
h_{max}	90.0	m	Maximum length of lane marker
$p_{\mathrm{rec,hyst}}$	0.005	-	Marker false negative recovery probability parameter
$p_{\mathrm{rec,pps}}$	0.05	-	Marker false negative recovery probability parameter
$p_{\mathrm{rec,sat}}$	0.3	-	Marker false negative recovery probability parameter
$\mathbf{p}_{\mathrm{lm,c},\lambda}$	diag(5.5, 5.5, 1.5, 2.5)	-	Lane marker geometry noise parameter
$\mathbf{p}_{\mathrm{lm,c},\Sigma_{\mathrm{init}}}$	diag(2.5, 0.05, 0.001, 0.0001)	-	Lane marker geometry noise parameter
$\mathbf{p}_{\mathrm{lm,c},\Sigma_u}$	diag(0.15, 0.007, 10^{-4}, 10^{-6})	-	Lane marker geometry noise parameter

4.5. Application of Sensor Models in RL-based Driving Policy Training

The main purpose of the presented sensor models is to enable efficient training of driving policies based on Reinforcement Learning methods that would be robust to perception errors. In this subsection I describe a direct control driving policy, which will be used for further experiments related to the sensor models that will be described in the next sections.

A vast selection of network architectures have already been proposed for the task of autonomous driving. As the network described in this section will be used primarily for experiments related to perception errors, the main goal of its design is to utilize state-of-the-art solutions (such as the use of transformers for input processing) while remaining relatively simple.

To fulfill these goals, I propose a direct-control driving policy that outputs acceleration and steering angle values. This design choice allows for a close investigation of the impact of the perception errors, as such issues may result in easily observable, immediate reactions to the policy. In contrast to higher-level control designs, such as behavior-planning policies that output semantic actions, in this setup incorrect inputs may lead directly to dangerous steering actions or a lack of appropriate braking reactions in dangerous situations.

4.5.1. Network Inputs

To prevent data loss and maintain efficient execution, object lists were used as input to the network. The input to the network is made up of several components, which are described in the following paragraphs.

State of the ego vehicle at time update t defined as:

$$\mathbf{o}_{ego}^{(t)} = \left[v_{s_e}^{(t)}, v_{\text{ex}_e}^{(t)}, a_{s_e}^{(t)}, a_{s_e}^{(t-1)}, \gamma_e^{(t)}\right], \tag{4.5.1}$$

where $v_{s_e}^{(t)}$ denotes a longitudinal velocity at time update t, $v_{\text{ex}_e}^{(t)}$ describes the absolute speed of the ego as a ratio of the speed of the ego at time update t to the current speed limit, $a_{s_e}^{(t)}$ and $a_{s_e}^{(t-1)}$ denote the acceleration of the ego at time updates t and $t-1$, and $\gamma_e^{(t)}$ denotes the ego's yaw rate divided by its absolute velocity.

Description of other road users in the vicinity of the ego vehicle. In the testing setup used for the experiments, only vehicles were considered for road users, although the description remains sufficiently general to describe other types of traffic participants, such as bicycles and pedestrians. Each road user is described using a vector:

$$\mathbf{o}_{obj_i} = [\mathbf{g}_i, \mathbf{x}_i, \psi_i, \mathbf{v}_i, \mathbf{a}_i]^T \text{ for } i = 1..n_{\text{obj\_max}}, \tag{4.5.2}$$

where $\mathbf{g}_i \in \mathbb{R}^2$ describes the width and length of the bounding box of the road user, $\mathbf{x}_i \in \mathbb{R}^2$ denotes the position of the center of the bounding box relative to the ego vehicle, $\psi_i \in \mathbb{R}$ is the rotation with respect to the ego's orientation, $\mathbf{v}_i = [v_{s_i}, v_{d_i}]^T$ describes the relative velocity of the road user, and $\mathbf{a}_i \in \mathbb{R}^2$ its acceleration relative to the ego.

Description of lane markers geometry, where each relevant (i.e., belonging to the drivable road in ego's proximity) lane marker is described with a state vector:

$$\mathbf{o}_{\text{lm}_i} = [\mathbf{d}_{\text{lm}_i}, h_{\text{lm}_i}, \gamma_{\text{lm}_i}, m_{\text{lm}_i}]^T, \tag{4.5.3}$$

where $\mathbf{d}_{\text{lm}_i} \in \mathbb{R}^{10}$ describes the marker geometry in the form of a vector of lateral positions. The vector is created by sampling the polynomial that describes the marker's geometry in a set of uniformly distributed longitudinal distances. $h_{\text{lm}_i} \in \mathbb{R}$ describes the distance up to which the marker is reliably observed, that is, the length of the marker, $\gamma_{\text{lm}_i} \in \mathbb{R}$ describes the rotation of the marker relative to the ego vehicle at the point closest to the ego, and $m_{\text{lm}_i} \in [0, 1]$ describes the type of marker, with $m_{\text{lm}_i} = 0$ denoting a broken line, and $m_{\text{lm}_i} = 1$ continuous one.

The description of lane markers and other road users, depending on the setup, is created either based on ground-truth information from the simulation package or from the sensor models' outputs. All values are updated at each time step, i.e., every time the network inference is performed.

4.5.2. Network Architecture

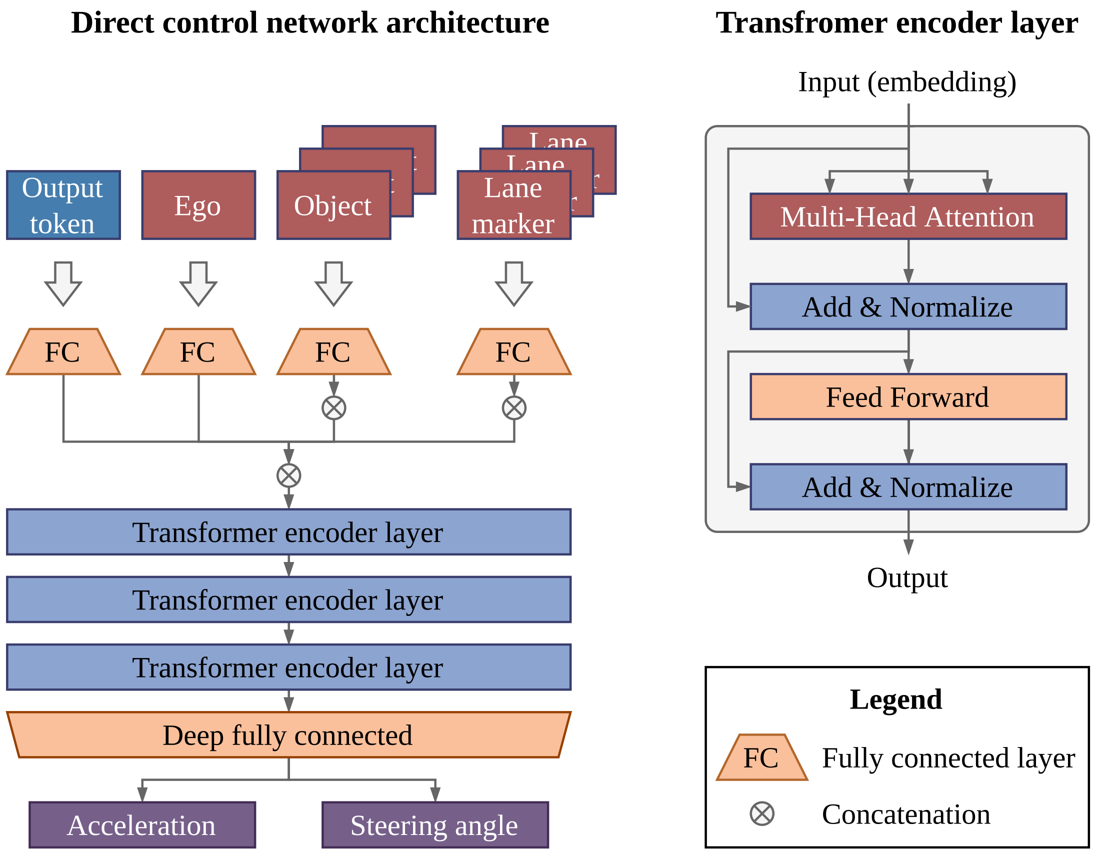


Figure 4.1. Architecture of the direct control network. n_{embd}-dimensional input embeddings of a const output token, Ego features ($\mathbf{o}_{\text{ego}}$), objects $\mathbf{o}_{\text{obj}_i}$ for $i = 1..n_{\text{max,obj}}$, and lane markers $\mathbf{o}_{\text{lm}_i}$ for $i = 1..n_{\text{max,lm}}$ are concatenated into matrix $I \in R^{n_{\text{embd}} \times (1+1+n_{\text{obj}}+n_{\text{lm}})}$ and consumed by the transformer encoder layers. Encoders use a masking mechanism to prevent non-existing objects or lane markers from impacting the outputs. Ultimately, the transformer encoder layers produce the output $O \in R^{n_{\text{embd}} \times (1+1+n_{\text{obj}}+n_{\text{lm}})}$. The first column of O is then processed by a deep fully connected network to generate the control values α_{acc} and α_{steer}. Note that the transformer structure used for observation lacks positional encodings, as there is no need to sort or prioritize the environmental features.

The use of variable-length object lists as an input to the network, while having the advantage of small memory requirements, creates a few challenges. For one, since the number of objects observed in the proximity of the ego vehicle varies, an adequate architecture is needed to distinguish between actual inputs and empty placeholders if not all possible object fields are filled. Another problem is related to the order of objects - while separate inputs can be used for various objects depending, e.g., on their position in relation to the ego, such solutions tend to result in sudden changes in output changes when objects change their categorization (e.g., object categorized as "adjacent left lane marker" changes when ego change lanes, possibly resulting in action changes).

To alleviate these issues, the transformer encoder model [185] is used for input processing. All the observation values are scaled and processed by the transformer model as presented in Fig. 4.1. The output of the model is further processed by a fully connected deep network that returns the output:

$$\alpha_\phi(\mathbf{o}_{\mathrm{ego}}, \mathbf{o}_{\mathrm{obj}}, \mathbf{o}_{\mathrm{lm}}) = [\alpha_{\mathrm{acc}}, \alpha_{\mathrm{steer}}]^T \tag{4.5.4}$$

The output values, ranged $[-1, 1]$ are scaled by gains $p_{\alpha_{\mathrm{acc}}}, p_{\alpha_{\mathrm{steer}}}$ to acquire actual control values.

The parameters related to the scaling of the inputs and outputs of the network are presented in Table 4.3.

Table 4.3. Calibration parameters related to the direct control network.

Parameter Name	Value	Description
$p_{\alpha_{\mathrm{acc}}}$	3.5	Acceleration output scaling.
$p_{\alpha_{\mathrm{steer}}}$	0.125	Steering output scaling.
$n_{\mathrm{max,obj}}$	10	Max number of observed objects.
$n_{\mathrm{max,lm}}$	6	Max number of observed lane markers.

4.5.3. Reward Components

The policy is trained using a PPO algorithm using a set of reward components that are intended to promote smooth and safe driving in a highway traffic environment. PPO training utilizes a composite reward $r^{(t)}$ defined as:

$$r^{(t)} = r^{(t)}_{\mathrm{speed\_limit}} + r^{(t)}_{\mathrm{acc}} + r^{(t)}_{\mathrm{steer}} + r^{(t)}_{\mathrm{centering}} + r^{(t)}_{\mathrm{TTC}} + r^{(t)}_{\mathrm{term}}, \tag{4.5.5}$$

where $r_{\mathrm{speed\_limit}}$ is a speed limit execution component, calculated as $r^{(t)}_{\mathrm{speed\_limit}} = w_{\mathrm{speed\_limit}} * v^{(t)}_{exe}$, with $w_{\mathrm{speed\_limit}}$ being a calibratable weight, and $v^{(t)}_{exe}$ the ratio between ego's speed and the speed limit. The second and third reward components penalize high control values and are defined as $r_{\mathrm{acc}} = w_{\mathrm{acc}} * \alpha^{(t)}_{\mathrm{acc}}$ and $r_{\mathrm{steer}} = w_{\mathrm{steer}} * \alpha^{(t)}_{\mathrm{steer}}$, with w_{acc} and w_{steer} denoting their respective weights. Lane centering behaviors are encouraged by the $r_{\mathrm{centering}} = w_{\mathrm{centering}} * d^{(t)}$ component, with $d^{(t)}$ denoting the lateral distance between the center of the ego and the center of the current lane.

Although strong penalties for collisions are typically sufficient to achieve satisfactory collision avoidance performance, an additional time-to-collision (TTC) term $r^{(t)}_{\mathrm{TTC}}$ is introduced to promote keeping a safe distance from other vehicles. TTC is a metric often used in ADAS features and can be calculated as a time after which a collision between the ego and another vehicle is expected to occur under the assumption that both vehicles will maintain their current acceleration. The component is set to 0 if the collision is not expected to happen; otherwise, the value of $w_{\mathrm{TTC}} * \frac{1}{max(1, v^{(t)}_{\mathrm{ttc}})}$ is assigned, where w_{TTC} is a calibratable weight and $v^{(t)}_{\mathrm{ttc}}$ is the TTC at time t.

The last cost term is assigned on the termination of the episode based on the termination reason. w_{term} value is assigned if the episode ends due to the ego's collision (with barriers or other road users), or severe speeding (defined as exceeding the current speed limit by $10\frac{m}{s}$ or more). Otherwise, the value of this reward component is zero.

The scaling parameters used for the reward components in the training are presented in Table 4.4.

Table 4.4. Values of reward components weights for the direct control driving policy training.

Parameter Name	Value	Description
$r_{\mathrm{speed\_limit}}$	0.04	Speed limit execution squared.
r_{acc}	-0.003	Ego acceleration squared.
r_{steer}	-1.0	Steering angle squared.
$r_{\mathrm{centering}}$	-0.006	Lane centering.
$r_{\mathrm{TTC\_max}}$	6.0	Max Time To Collision to be included in reward in m/s.
r_{TTC}	-0.01	Time to collision (inversed).
r_{terminal}	-10.0	Terminal states (collisions, speed limit violations.)

4.5.4. Training Setup

Traffic simulation software is the main component of the environment used to both train and test the proposed driving policy. For this purpose, a proprietary traffic simulation tool TrafficAI was used.

The simulation environment allows simulation of realistic road traffic on arbitrarily defined maps. The movement of the road users is governed by a set of semi-random heuristic rules, with various agents having different driving styles (defined by, e.g., aggressiveness levels).

Training is performed on randomly generated maps consisting of multilane highway roads with features typical for highway networks, such as exit and merge-in lanes. A simulated highway is populated with road traffic of various density, ranging from empty road setup to heavy traffic that results in traffic jams.

The simulation is updated in $t_{su} = 0.05s$ steps, while the ego's policy inference resulting in an update of control values is performed every two simulation steps.

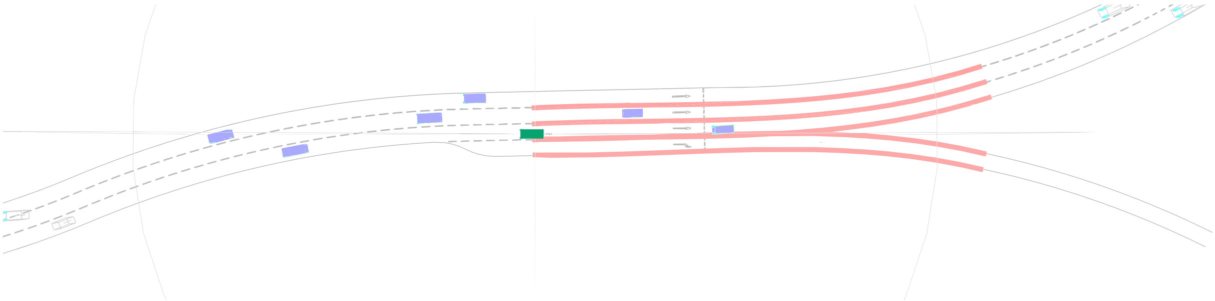


Figure 4.2. Visualization of the simulation environment used for trainings and evaluations.

Training of direct control policies presented in the following sections is performed in a distributed computing setup, where 100 simulation threads are used for data collection, with the entire training lasting approximately 24 hours.

Training hyperparameters are summarized in the table 4.5.

Table 4.5. Proximal Policy Optimization training hyperparameters.

Hyperparameter	**Value**
Train batch size	250000
Minibatch size	5000
Num epochs	15
Discount (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.3
KL coefficient	0
Entropy coefficient	0
VF coefficient	1.0

4.6. Evaluation methodology

Training of RL-based driving policies is the main application of the proposed method, which can be used to alleviate sim-to-real gap issues in the training setup. Developed sensor models in such a setup can be used to enhance the simulator's output, producing a realistic set of observations for the trained agent.

One way to evaluate the models would be to perform statistical comparisons between the sensors' output and data produced by the models based on a certain ground truth. While this approach could provide some insight into the models' accuracy in terms of mimicking the real world, it is not very useful in the context of the described application. Since high-fidelity sensor models tend to be prohibitively slow for RL training, achieving the best accuracy of the models is not necessarily the main concern in their design.

For this reason, a more informative evaluation approach is to train the driving policy with the use of the proposed sensor models and evaluate its performance instead of the sensor models themselves. Comparison of such policy to the agents trained on ground-truth simulation data or with the use of certain baseline sensor models would help to understand a crucial question - whether and how the proposed method improves the performance of the final policy.

Unfortunately, the evaluation of the policy itself is not without its own challenges. Since the behavior of the ego vehicle inevitably impacts the decisions of other road users, one of the most informative ways to evaluate AD systems is to perform closed-loop driving tests on public roads in realistic traffic. Such tests, however, pose severe safety hazards and would require extreme safety precautions, as well as well-tested trajectory generation systems, and thus are not viable for safety evaluations and comparisons of new driving policies, especially if a comparison to baseline policies is required.

Open-loop testing techniques, which do not involve feedback from the environment, cannot be used to fully evaluate a driving policy, as it is difficult to consider the influence of the policy on the behavior of other drivers. Pre-recorded sensor data can still, however, be used to investigate the impact of the sensors' errors on the policy's choices. An interesting way to do so was described in [157], where a Probably Approximately Correct Sensing System was defined as a perception system that is sufficiently accurate to not cause frequent severe mistakes of the driving policy, compared to ground truth data. Extending this idea, one can formulate an open-loop test based on pre-recorded sensor data (e.g., bounding boxes from a radar-based sensing system) with additional ground-truth information (e.g., manually labeled bounding boxes based on accurate LiDAR and camera data). A comparison of the policy's response to both ground truth and sensor data provides information about how the sensors' performance limitations impact the policy's decisions. While this approach does not provide information about the actual performance of the policies themselves, it can be used to compare different policies in terms of their robustness to realistic sensors' errors. Unfortunately, the lack of open datasets consisting of object-level radar-based streams with ground-truth data prevents the use of this method for the evaluation of the method described in this chapter.

The policies are instead evaluated in a set of closed-loop simulation environments. In particular, two types of simulation-based test environments are used: a highway environment for large-scale evaluation in typical traffic and a set of short test scenarios that evaluate agents' performance in challenging situations. Each of the test environments is further specialized by utilizing three types of sensor models: ground truth, evaluated sensor models, and an additional set of baseline models described in the next subsection.

4.6.1. Baseline Models

Domain randomization techniques used to alleviate sim-to-gap issues in RL setups often utilize simple noise models in the training environment to improve the policy's robustness to real-life conditions. The intuition behind such an approach is that policy trained with severe input noise may acquire robustness to a wide spectrum of possible errors, hopefully including ones that will be observed in the targeted environment. If this assumption were true for a given environment, there would be no strong basis for the use of more complex sensor models for training purposes.

To investigate whether simple domain randomization techniques could be useful for the case of automotive sensors, I developed a set of baseline sensor models that utilize Gaussian noise for state estimation errors and simplify false negative detections generation.

The mapping $M^{\mathrm{d,occ}}$ defined in previous sections is used to perform deterministic removal of objects outside the sensing area, resulting in a set $\{\mathbf{s}_{d_j}\}^{(v)}_{j=1..n_{d,occ}}$, where $n_{d,occ}$ is the number of unoccluded objects in a detection area.

Object state estimation errors are modeled as a mapping:

$$M^{\mathrm{bl,state}}\left(\Sigma_{\mathrm{bl,state}}\right) : \left\{\mathbf{s}^{(v)}_{\mathrm{d}_j}\right\}_{j=1..n_{\mathrm{d,occ}}} \rightarrow \left\{\mathbf{s}^{(\mathrm{d})}_{\mathrm{d}_i}\right\}_{i=1..n_{\mathrm{d,occ}}}, \tag{4.6.1}$$

where $\Sigma_{\mathrm{bl\_state}}$ is a diagonal covariance matrix used to calibrate the model, and the state estimates in the resulting set $\mathbf{s}^{(d)}_{\mathrm{d}_i}$ for $i = 1..n_{\mathrm{d,occ}}$ are calculated as:

$$\mathbf{s}^{(d)}_{\mathrm{d}_i} =\sim \mathcal{N}(0, \Sigma_{\mathrm{bl,state}}) + \mathbf{s}^{(v)}_{\mathrm{d}_i} \text{ for } i = 1..n_{\mathrm{d,occ}}. \tag{4.6.2}$$

Perhaps the most impactful difference between this way of modeling the state estimation errors and the models based on Ornstein-Uhlenbeck noise is the lack of time correlations. Similarly, false negative object detection errors are modeled as single timestep events, where each object at each step can be removed from the detection set with a probability $p_{\mathrm{bl,disc}}$. False negatives removal is described as a mapping:

$$M^{\mathrm{bl,fn}}(p_{\mathrm{bl,disc}}) : \left\{\mathbf{s}^{(d)}_{\mathrm{d}_i}\right\}_{i=1..n_{\mathrm{d,occ}}} \rightarrow \left\{\mathbf{s}^{(\mathrm{fn})}_{\mathrm{d}_i}\right\}_{i=1..n_{\mathrm{bl,fn}}} \tag{4.6.3}$$

False positive detections are also modeled as single-step events, where the additional object is created at each timestep with a probability $p_{\mathrm{bl,fp}}$. The state of the false-positive object $\mathbf{s}_{\mathrm{bl,fp}} = [g_{\mathrm{lon}}, g_{\mathrm{lat}}, x_{\mathrm{lon}}, x_{\mathrm{lat}}, \psi, v, a]^T$ is sampled according to a distribution $\sim \mathcal{N}_7(\mu_{\mathrm{bl,fp}}, \Sigma_{\mathrm{bl,fp}})$. The creation of false positive object detections is described with a mapping:

$$M^{\mathrm{bl,fp}}(p_{\mathrm{bl,fp}}, \mu_{\mathrm{bl,fp}}, \Sigma_{\mathrm{bl,fp}}) : \left\{\mathbf{s}^{(\mathrm{fn})}_{\mathrm{d}_j}\right\}_{j=1..n_{\mathrm{bl,fn}}} \rightarrow \left\{\mathbf{s}^{(\mathrm{fp})}_{\mathrm{d}_i}\right\}_{i=1..n_{\mathrm{bl,fp}}}, \tag{4.6.4}$$

with $n_{\mathrm{bl,fp}}$ denoting the number of objects after the generation of false positive detections, and $\mu_{\mathrm{bl,fp}}, \Sigma_{\mathrm{bl,fp}}$ being calibratable parameters.

The baseline model of the lane markers detection system is created in a similar way, with the use of Gaussian noise for markers length and coefficients, as well as random single-timestep false negative detections.

The state of $i-th$ lane marker $\hat{\mathbf{s}}_{\mathrm{bl,lm}} = \left[\hat{c}_{0i}, \hat{c}_{1i}, \hat{c}_{2i}, \hat{c}_{3i}, \hat{h}_i\right]^T$ for $i = 1..n_{\mathrm{lm}}$ of the lane markers is thus modeled as:

$$\mathbf{s}^{(\mathrm{lm,state})}_{\mathrm{bl,lm}_i} =\sim \mathcal{N}(0, \Sigma_{\mathrm{bl,lm}}) + \mathbf{s}^{(\mathrm{lm,fd})}_{\mathrm{bl,lm}_i} \text{ for } i = 1..n_{\mathrm{lm,fd}}, \tag{4.6.5}$$

where $\Sigma_{\mathrm{bl\_lm}} \in \mathbb{R}^{5\times 5}$ is the covariance matrix used for calibration.

Furthermore, at each time step a random lane marker can be removed from the set of observed markers with a probability $p_{\mathrm{lm,fn}}$, as well as a false positive lane marker can be added with a probability $p_{\mathrm{lm,fp}}$. False positive lane markers are initialized with a state drawn from a distribution $\sim N_5(\mu_{\mathrm{bl,lm,fp}}, \Sigma_{\mathrm{bl,lm,fp}})$.

Table 4.6. Parameters of baseline sensor models (G-SM).

Parameter	Description	Value
$\Sigma_{\mathrm{bl,state}_{\mathbf{x}}}$	Object position error covariance matrix	$\begin{bmatrix} 1.2 & 0 \\ 0 & 0.7 \end{bmatrix}$
$\sigma_{\mathrm{bl,state}_v}$	Velocity error variance	2.0
$\sigma_{\mathrm{bl,state}_{\mathbf{q}}0}$	Object length error variance	0.5
$\sigma_{\mathrm{bl,state,min}_{\mathbf{g},\mathrm{lon}}}$	Object length error lower limit	-1.0
$\sigma_{\mathrm{bl,state}_{\mathbf{g},\mathrm{lat}}}$	Object width error variance	0.5
$\sigma_{\mathrm{bl,state,min}_{\mathbf{g},\mathrm{lat}}}$	Object width error lower limit	-1.0
$p_{\mathrm{bl,fp}}$	False positive detection probability [1]	0.0575
$p_{\mathrm{bl,fn}}$	False negative detection probability [2]	0.1
$\mu_{\mathrm{bl,h}}$	Lane marker mean observed length	87.0
$\sigma_{\mathrm{bl,h}}$	Lane marker observed length variance	5.0
$h_{\mathrm{bl,max}}$	Lane marker observed length upper limit	90.0
$\Sigma_{\mathrm{bl,lm}}$	Lane marker coefficients errors covariance matrix	diag(0.005, 0.0005, 0.00005, 0.000005)

[1] Parameters of the false positive object detections are drawn from the same distributions as described in Table 4.1.
[2] Duration of the false negatives is fixed to a single time step.

4.6.2. Test Scenarios

While the use of baseline sensor models both as an additional testing setup and as a baseline for the evaluation of OU-based sensor models helps to gain insight into the performance of the method, testing policies in the described simulation environments is far from ideal. All the setups utilize the same simulation package, which may be unable to generate certain edge-case scenarios that could happen in the real traffic environment. Furthermore, even though a setup with baseline sensor models can be used to compare ground truth policy and OU-based sensor models policy in a third environment, it cannot be considered a fully independent one, due to the underlying simulation being common for all setups.

In order to address these issues, a set of semi-random low-level test scenarios is introduced. The scenarios are designed to provide insight into the policies' behaviors in difficult situations related to sensing errors, emulating cases such as late detection of a vehicle, false-positive detections in front of the ego, or long-lasting false negatives.

The parameters of particular instances of a given scenario are drawn from random distributions, allowing to test policies in a large number of different versions of a given situation. Parameters that describe the distributions themselves are chosen manually to produce an intentionally difficult set of scenarios, possibly including unsolvable scenarios, that is, situations in which avoiding collisions with other road users is impossible considering control constraints. Testing investigated models in such extreme conditions allows exposure of possible issues in the tested policies.

Scenario A: Late detection of a slow-moving vehicle in front of the ego
The first test scenario consists of the ego and another vehicle in front of it, both driving on a 2-lane highway. The object remains invisible to the ego (false negative detection) for the first 2 seconds of the scenario. After this time, it is detected correctly with no state estimation errors.

Ego's initial velocity as well as the initial state of the vehicle are drawn from the distributions listed in table 4.7. The scenario is intended to evaluate the ego's responses to late detection of a slow-moving object.

Depending on a particular instance of the scenario, the correct response may consist of severe braking or lane change maneuvers applied to avoid a collision with other vehicles.

Table 4.7. Scenario A parameters distributions.

Parameter	**Value**
Ego's initial velocity $[\frac{m}{s}]$	$\mathcal{U}(20.0, 30.0)$
Object's initial relative longitudinal position $[m]$	$\mathcal{N}(30.0, 3.0^2)$
Object's initial relative lateral position $[m]$	$\mathcal{N}(0.0, 0.3^2)$
Object's initial absolute speed $[\frac{m}{s}]$	$\mathcal{N}(10.0, 2.0^2)$
Object's acceleration $[\frac{m}{s^2}]$	$\mathcal{N}(0.0, 1.0^2)$.

Scenario B: Severe speed estimation error
The scenario is composed of two vehicles (ego and another road user) moving on a 3-lane highway. The vehicles are placed on a random lane, with relative positions and absolute speeds drawn from the distributions described in Table 4.8.

The front object's state estimation performed by ego is accurate, except for the speed estimation, which is disturbed by a constant-velocity estimation error (the observed speed of the object is higher than the actual speed).

The scenario is intended to evaluate whether the driving policy is able to make correct decisions in the presence of a severe estimation error affecting one of the state variables based on other values.

Table 4.8. Scenario B parameters distributions.

Parameter	Value
Ego's initial velocity $[\frac{m}{s}]$	$\mathcal{U}(20.0, 30.0)$
Object's initial relative longitudinal position $[m]$	$\mathcal{N}(40.0, 3.0^2)$
Object's initial relative lateral position $[m]$	$\mathcal{N}(0.0, 0.3^2)$
Object's initial absolute speed $[\frac{m}{s}]$	$\mathcal{N}(10.0, 2.0^2)$
Object's acceleration $[\frac{m}{s^2}]$	$\mathcal{N}(0.0, 1.0^2)$.
Constant velocity estimation error $[\frac{m}{s}]$	20

Scenario C: Random speed estimation error

Scenario C is set up identically to Scenario B, with one difference: the speed estimation error is sampled at each time step according to the normal distribution described in the table 4.9.

Table 4.9. Scenario C parameters distributions. Parameters that describe distributions of vehicles' initial states are identical to the *Scenario B*.

Parameter	Value
Velocity estimation error $[\frac{m}{s}]$	$\mathcal{N}(0.0, 10.0^2)$

Scenario D: Random lateral position estimation error

In this scenario, in which the ego vehicle follows a slower moving object, the estimation of the lateral position of the other vehicle is disturbed by an error drawn from a normal distribution described in table 4.10. Other aspects of the scenario remain identical to those of Scenarios B and C.

Table 4.10. Scenario D parameters distributions. Parameters that describe distributions of vehicles' initial states are identical to the *Scenario B*.

Parameter	Value
Lateral position estimation error $[\frac{m}{s}]$	$\mathcal{N}(0.0, 3.5^2)$

Scenario E: Randomly occurring false negative detections.

Scenario E is set up similarly to scenarios B-D, but instead of state estimation errors, randomly occurring false negative detection errors are simulated. Errors that affect front vehicle detections have a constant duration and can occur at each time step with a probability given in the Table 4.11.

Table 4.11. Scenario E parameters distributions. Parameters that describe distributions of vehicles' initial states are identical to the *Scenario B*.

Parameter	Value
Probability of false negative detection occurrence in each time step.	0.1
Duration of false negative detection error events. $[s]$	0.2

Scenario F: Randomly occurring false negative road markers detections. Scenario F consists of the ego vehicle moving on an empty 3-lane highway. The vehicle is placed in a random lane. During the scenario, lane marker detections are randomly removed for a single timestep to simulate false negative detection errors.

The scenario allows testing policy's ability to properly navigate on a road even if certain lane markers are missing.

Table 4.12. Distributions of Scenario F parameters. Ego initial state parameters are identical as in *Scenario B*.

Parameter	**Value**
Probability of lane marker false negative detection.	0.4

4.7. Experimental results

The main focus of the experiments carried out was to evaluate the impact of the use of proposed sensor models in the training of the policy on the performance of the policy itself. For this reason, three training policies were trained for subsequent evaluations: GT policy, trained on ground truth data from the simulator, Baseline policy, trained with the use of baseline models, and OU policy, trained on the proposed sensor models based on Ornstein-Uhlenbeck processes. Except for the sensor models utilized, all training setups and network architectures were identical.

The simulation environment utilized for training consisted of randomly generated highway maps. The maps represented simple highway networks, with roads between 1 and 4 lanes, and features such as curves, merge-in lanes, and highway exits.

The policies were trained with the PPO algorithm, tuned with a set of hyperparameters presented in table 4.5. Trainings were carried out on a computing cluster, each taking approximately 24 hours and utilizing 80 CPU cores.

The use of sensor models did not have a significant impact on the training process itself. All policies were successfully trained, using a similar number of training iterations. As shown in Fig. 4.3, the final mean reward values achieved in the GT environment have been higher compared to the environments with sensor models, which is an expected outcome, as the baseline and OU environments are significantly more difficult. Most notably, the reward component related to absolute accelerations in these environments achieves noticeably lower values in these environments, most likely due to more frequent sudden brakings caused by late detection errors.

4.7.1. Highway Driving Performance

All of the trained policies were evaluated in the environments described in a previous section: one with Ground Truth (GT) sensor models, one with Gaussian-based baseline sensor models (G-SM) and one with Ornstein-Uhlenbeck-based sensor models (OU-SM environment). Evaluation

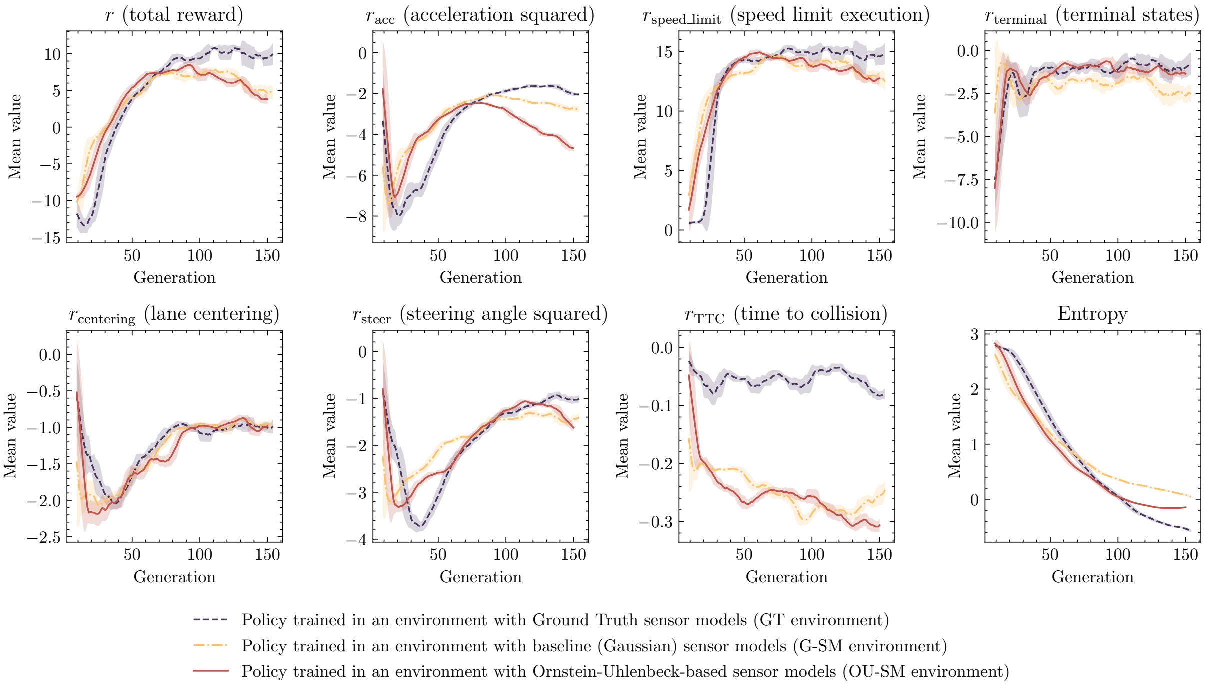


Figure 4.3. Progress of the policy's training in all of the tested environments. The final rewards of the agents in G-SM and OU-SM environments have lower values due to the increased difficulty of these setups. Otherwise, all the trainings progressed in a similar manner.

of each agent was performed by running 100 simulation episodes in all test environments. The episodes were terminated after reaching a horizon of 1000 simulation steps or after a collision with other road users or road barriers.

Based on performed simulations, Key Performance Indicators (KPIs) were calculated. listed in the table 4.13.

The policy trained in the GT environment highlights potential sim-to-real gap issues of the policies trained in the perfect environment, as it achieves very poor performance in the presence of simulated perception errors. Almost half of the episodes were in the G-SM environment, and the vast majority of OU-SM simulations were ended prematurely due to collisions. A large number of heavy braking events and increased values of control actions observed in the G-SM environment seem to indicate that the policy has been frequently undertaking severe collision avoidance attempts. Interestingly, the number of heavy braking events and average steering angles in the OU-SM environment is lower compared to the G-SM environment. This observation can be explained by very short mean episode lengths - the policy was likely not able to successfully execute collision avoidance maneuvers in this setup, and most of the dangerous situations ended in collision instead.

Manual inspection of the policy's behavior in the generated episodes supports these hypotheses. The policy trained in the GT environment tends to keep short distances from other road users, and, as a consequence, many dangerous situations are observed. The policy tends to exhibit

Table 4.13. Key Performance Indicators calculated based on highway driving episodes. The performance of three driving policies are summarized here - the policy trained in the ground-truth environment (GT agent), the policy trained in the baseline environment (with gaussian-noise-based sensor models, denoted G-SM agent), and the policy trained with the Ornstein-Uhlenbeck-noise-based sensor models proposed in previous sections, denoted OU-SM agent. Each policy has been evaluated in all three training environments.

	GT Agent			**G-SM Agent**			**OU-SM Agent**		
Performance Indicator	**GT env**	**G-SM env**	**OU-SM env**	**GT env**	**G-SM env**	**OU-SM env**	**GT env**	**G-SM env**	**OU-SM env**
Mean episode length (sim steps)	977.4	716.8	291.5	925.8	927.2	804.0	907.6	941.8	910.8
Average abs speed $[\frac{m}{s}]$	27.6	27.9	26.0	29.1	29.4	29.0	26.0	26.2	27.1
Average abs steering angle [rad]	0.35	0.56	0.30	0.54	0.57	0.62	0.41	0.46	0.48
Average abs acceleration $[\frac{m}{s^2}]$	0.64	0.91	1.23	0.68	0.76	0.82	0.98	0.91	0.99
Heavy braking events	1.7	8.7	1.5	2.9	7.6	4.1	2.7	8.2	3.5
Fraction of episodes failed	0.03	0.48	0.84	0.04	0.07	0.27	0.03	0.02	0.03

various severe reactions to such situations, applying extreme steering angles and accelerations. However, as most of the errors in this environment are short-lasting, policy's reactions are often sufficient to avoid collisions, and overcompensating behaviors are short enough not to cause collisions themselves. Furthermore, the policy in the G-SM environment exhibits frequent oscillations of both control values.

Observation of the policy behavior in the OU-SM environment highlights even more concerns. Apart from the lack of caution, oscillations, and overcompensation behaviors observed in G-SM environments, time-correlated errors observed in the OU-SM environment exposed critical safety issues of the GT policy. Most notably, the policy reacts to both false-positive detections appearing in front of the vehicle, as well as late-detected front objects with extreme collision avoidance attempts. The vehicle typically tries to avoid the collision by applying severe steering values, which often lead to a collision with road barriers or other road users. Road barrier collisions caused by such behaviors are categorically the most frequent reason for premature episode termination.

Training the policy in the G-SM environment, being similar to the domain randomization techniques used occasionally in Reinforcement Learning, was expected to alleviate at least part of the issues observed in the GT policy. In fact, the policy achieved significantly better performance in both G-SM and OU-SM environments, with 7% and 27% failed episodes, respectively. The policy exhibited more cautious behaviors in all the environments, including the GT environment - the distances kept to other vehicles were larger, and the policy often applied collision

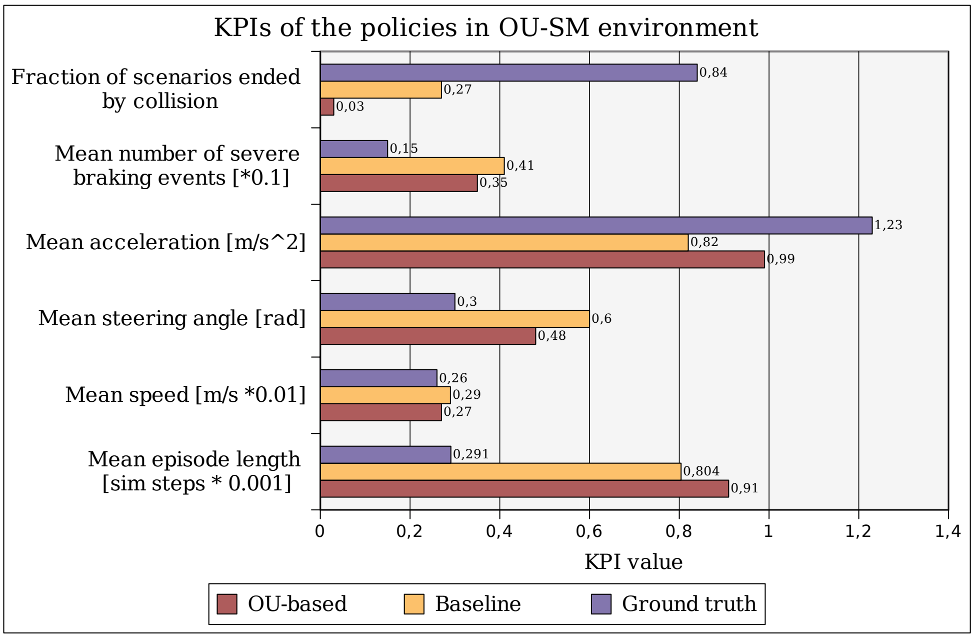


Figure 4.4. Performance of the trained policies in OU-SM environment. The results show the inability of both G-SM and GT driving policies to cope with time-correlated errors present in the OU-SM environment, most apparently visible in the fraction of failed episodes. Note that a complete set of evaluation results in all environments can be found in Table 4.13.

avoidance maneuvers in moderately dangerous situations. Evaluation of the policy in the OU-SM environment has exposed, however, similar, yet less severe, issues to the problems observed in the GT environment. Time-correlated errors sporadically caused severe reactions, often resulting in collisions. The final performance, with approximately 27% failed episodes, remains unacceptably poor.

Training the policy in the OU-SM environment expectedly resulted in significantly better results in this environment, but also led to smooth and cautious behaviors in other environments, maintaining a good overall performance. The policy was able to drive in a cautious manner, handle late detections and false positives correctly by applying appropriate collision avoidance measures, and overall achieve good performance in all setups. Interestingly, the policy tested in the G-SM environment achieved significantly better collision avoidance performance compared to the G-SM policy, with only 2% failed episodes (compared to 7% achieved by the G-SM agent).

4.7.2. Scripted Scenarios Performance

Evaluation in highway driving environments provides valuable insight into the limitations of GT and G-SM policies, as well as helps to evaluate the robustness and versatility of all policies. To fully understand the advantages and limitations of all policies, however, additional tests were performed in a set of scripted scenarios, to ensure an independent testing environment.

Distributions from which the parameters of scenarios were sampled were intentionally calibrated in such a way that the resulting scenarios posed a significant challenge, where part of the scenarios may be possibly too difficult to resolve without a collision.

The results of the test scenarios are presented on the chart in Fig. 4.5, where a fraction of the scenarios that led to collisions was used as a safety performance indicator. The scenarios confirmed the advantage of the policy trained in the OU-SM environment over the other two policies, with significantly fewer collisions caused by this policy. Interestingly, the performance of the GT and G-SM policies varied between all scenarios, with the G-SM policy achieving worse performance in several scenarios.

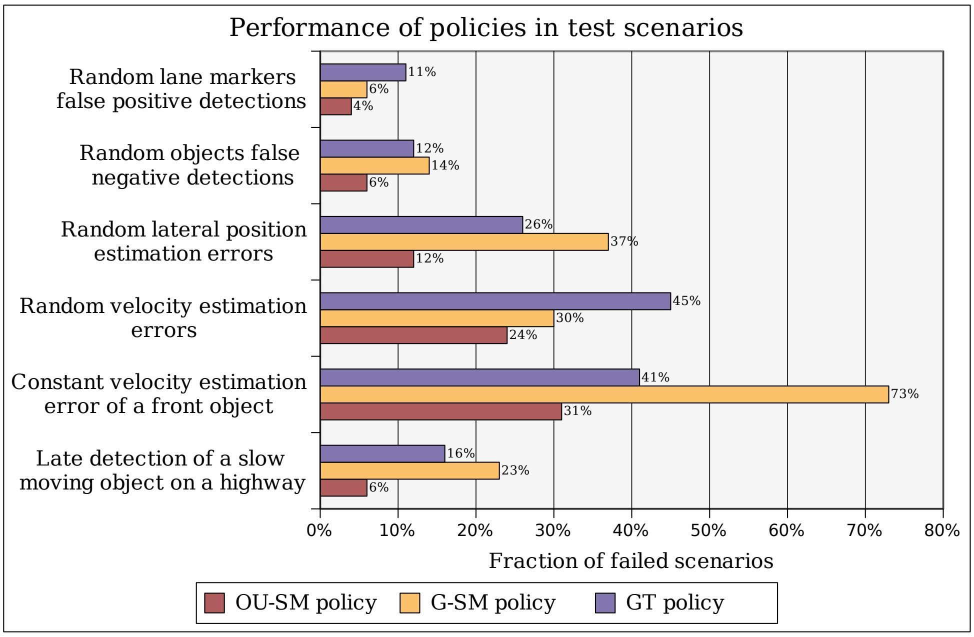


Figure 4.5. Performance of the trained policies in different test scenarios .

4.7.3. Computational Performance

One of the main goals of the proposed sensor models is to enable effective training of RL-based driving policies in a closed-loop simulation environment. While achieving real-time sensor models simulation performance is not required for successful training, the massive scale of computation efforts required for RL training creates an incentive to avoid slow and complex computations to reduce costs related to high-performance computing.

The computational performance of the proposed methods has been investigated in a closed-loop highway driving setup, with the OU-SM environment. The proposed models and the driving policy were executed for 800 time steps in a three-lane highway scenario with relatively heavy traffic (with 8-10 other traffic participants in the ego's sensing area throughout the entire scenario). The experiments were executed in a single thread on the Intel(R) Core(TM) i7-12800H CPU, resulting in the computation times presented in Table 4.14.

Table 4.14. Single timestep computation time of different components (in seconds).

Component	Average	Median	Std. dev.	Max
Objects perception model	0.0071	0.0068	0.0008	0.0114
Static env. perception model	0.0369	0.0406	0.0075	0.0590
Network inference	0.0030	0.0027	0.0008	0.0059

Despite a very demanding test scenario of a congested multilane highway, the average computation time observed for the proposed sensor models remained below $44ms$, allowing for faster than real-time training of a policy with the network inference performed every $0.1s$, as proposed in previous sections.

Calculations related to static sensor models took significantly longer, due to relatively slow sampling and polynomial approximation operations. While the achieved performance has been satisfactory for training purposes, it can be easily improved if needed by decreasing the number of samples used for the lane markers geometry approximation, as well as by using a single-precision floating point format for related calculations.

In contrast to sensor models, the driving policy is intended to be employed in a closed-loop vehicle control system, and thus its performance is significantly more crucial to the final product. Fortunately, the performance of the proposed lightweight network architecture observed in the described computing setup is more than satisfactory, with the mean inference time achieved on a single CPU thread reaching 3 milliseconds. Furthermore, the complexity of the road situation, the number of vehicles surrounding the ego, and the number of lanes do not significantly impact the inference time, with the observed standard deviation of the inference time below 1 ms. As the performance of modern embedded computing platforms used for AD purposes often matches the performance of the CPU used for the performed tests, real-time performance in a target vehicle should be easily achievable with the proposed network architecture, as even a 10-fold decrease in computing power would still leave significant computation time margin even in complex road situations.

As the policy trained in the described setup with proposed sensor models exhibits robustness against even severe sensor errors, while producing smooth, efficient movements with an exceptionally low computing power needed, it appears to be a viable option for a component of a commercial real-time ADAS/AD system.

4.8. Conclusions

A wide variety of perception systems is used in the automotive industry for ADAS and AD systems. Errors observed in such systems vary significantly depending on the number of sensors, their type, individual properties, as well as utilized algorithms, system architecture, and other factors. Furthermore, various road situations, as well as weather conditions, road surface types, and light sources, can affect sensors' performance and may produce errors that are difficult to predict.

The modeling of perception systems is thus a very difficult challenge, especially if one intends to create general models while maintaining satisfactory computational performance.

Fortunately, not all applications require models to be perfectly accurate. The main application considered in this chapter is the 'use of sensor models in the training of driving policies based on reinforcement learning, where discrepancies between the real world and the simulation used in training can lead to incorrect behavior of the trained policy in the targeted environment. One of the most frequently proposed ways to address this problem is the use of domain randomization techniques, which introduce various random modifications to the input data. This approach has been shown to produce good results in robotic applications, even if the errors introduced to the observation were relatively random and did not resemble the error profiles observed in realistic systems.

These observations seem to suggest that it is more desirable as well as practical to train a model to generalize well to various domains than to attempt to recreate the intended environment perfectly. Automotive perception systems suffer, however, from certain safety-critical error patterns, such as long-lasting false positive detections, that may not be accurately captured by naive domain randomization techniques.

In this chapter, I described several experiments that I performed to confirm these observations. Training an RL-based driving policy in a ground truth environment and testing it in both scripted scenarios and highway environments with sensor models has shown that such systems may in fact be severely impacted by the gap between the training simulation and real-world (sim-to-real gap).

The applicability of naive domain randomization techniques to this problem has been evaluated by performing similar experiments, but with a policy trained in an environment with baseline sensor models, based on random false detection events and Gaussian noise added to state estimations. While the use of such models had a positive impact on the policy's robustness to perception errors, evaluations in scripted scenarios exposed several issues, especially related to time-correlated errors.

The poor performance of driving policies trained with ground truth and baseline sensor models justifies the need for the development of efficient sensor models that would be able to mimic more complex errors characteristics of automotive perception systems.

A set of sensor models related to the detection of lane markers, as well as object detection and state estimation, have been designed and evaluated in scripted scenarios and various highway traffic environments. Models maintain faster than real-time performance appropriate for RL training, while approximating various error patterns observed in automotive perception systems, such as time-correlated state estimation errors, long-lasting false detections, occlusions, or detection delays. The policy trained in the environment with developed sensor models has shown good performance in all test environments as well as in scripted scenarios. The proposed policy was able to generate smooth and efficient motion while remaining resilient to considerable perception errors with minimal computational requirements. Therefore, it could be a viable choice for real-time ADAS/AD systems.

The satisfactory results achieved in the simulation-based evaluation suggest that the proposed sensor models may serve as an efficient domain randomization technique appropriate for training robust driving policies. However, further testing in real-life situations is needed to confirm that the proposed methods fully address the sim-to-real gap issues. The effectiveness of the proposed methods may vary depending on the sensor stack and the architecture of the perception system in the target vehicle.

Future work on the proposed methods may include the evaluation of the method on sensor data from test drives in public traffic. While performing closed-loop driving policy evaluation in public traffic is a potentially dangerous endeavor with severe formal requirements, certain open-loop testing techniques can be used to gain insight into the method's performance with a realistic sensor stack, such as performing various comparisons of the policy's response to the actual sensors data streams to the response to the manually labeled ground truth data.

5. Adversarial Trajectories Generation

One of the biggest challenges in the development of Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) systems is their validation and verification. The use of complex and often nontransparent algorithms, a potentially infinite number of road situations in which the system may operate, and a diversity of sensing system error modalities make prediction of all failure modes infeasible in such systems.

The automotive industry employs a variety of exploratory testing techniques to identify potential safety problems in developed ADAS and AD systems. These include Real-World User Profiling (RWUP), which involves driving a large fleet of vehicles in real-world traffic, and a variety of virtual testing methods, which assess the safety of systems in a simulated environment. However, both types of methods have their disadvantages in the context of testing systems capable of controlling the vehicle's movements. Real-world tests tend to be extremely costly and may pose a danger to test drivers and other road users, while simulation-based testing may fail to produce certain complex and atypical situations that may be challenging for the tested system.

In this chapter, I describe a testing method designed to explore the vulnerabilities of ADAS and AD systems to atypical road situations, as well as to perception errors. The method is designed to produce a variety of virtual test scenarios that are particularly challenging to a given ADAS/AD system (adversarial scenarios). The use of the proposed method helps to ensure that a driving policy is robust to plausible but complex road situations that may be difficult to observe during RWUP or standard simulation-based testing.

The method is particularly suited for testing driving policies that are based on Machine Learning methods. Its effectiveness in the generation of adversarial scenarios is demonstrated in the context of Reinforcement-Learning-based driving policy testing.

5.1. Introduction and Motivation

This section introduces the background of simulation-based ADAS/AD systems testing, reviews existing work in this area, and explains the motivation for the development of the proposed method.

5.1.1. Background

Due to the immense costs and potential risks of road driving tests, simulation-based testing has gained a lot of attention, especially in the context of AD systems testing. Systems are typically tested in one of the setups listed below.

Model-in-Loop (MiL), where the model of algorithms used in the system controls the virtual vehicle in the simulation. The model typically makes its decision in a closed-loop manner, utilizing an approximation of the sensor data that would be generated in the simulated situations. Models that perform well in the simulation can then be used to develop actual software that implements the algorithms in a way suitable for the target hardware (e.g., in low-level programming languages for embedded platforms with limited resources).

Software-in-Loop (SiL), where the software implementation of the algorithms is tested in a closed-loop manner. The software is not executed on the target hardware in this setup, instead utilizing a computing cluster or a personal computer that often runs the simulation at the same time.

Processor-in-Loop (PiL), which resembles the SiL setup, but the software is executed on a dedicated processor (typically a microcontroller or a digital signal processor).

Hardware-in-Loop (HiL), where the actual electronic component with the developed software is tested.

Note that certain types of algorithms do not benefit from a closed-loop setup and can be effectively tested in a simpler open-loop manner. This is usually the case for the algorithms that do not exercise direct control over the vehicle, such as passive warnings (e.g., Blind Spot Warning (BSW) that indicates the presence of a vehicle in a black spot of side mirrors or Cross Traffic Alert (CTA) that warns the driver about vehicles oncoming from the side, e.g., during reversing into the perpendicular road).

Open-loop testing is especially useful due to its ability to reuse realistic data registered during the test drives for testing various algorithms' versions. Raw data registered previously by the sensors can be provided to a new version of the tested algorithm to observe changes in the algorithm performance without the limitations of the virtual environments. This type of regression testing is often referred to as resimulation.

The use of simulation techniques in the testing process enables an efficient evaluation of driving policies on a massive scale. Thanks to parallel computing, the vehicle model with the tested system can cover distances in the virtual world that are orders of magnitude larger compared to the capability of an actual vehicle with the test driver.

Unfortunately, in order to provide meaningful information regarding system performance in various situations and environmental conditions, the simulation software would have to model the static environment, behavior of other road users, and physical phenomena that impact sensor performance with extraordinary precision. The ability to do that is limited by both the computational cost of complex simulation (e.g., precise high-resolution rendering of the environment for camera sensor inputs) and technical limitations. For these reasons, the incorporation of simulation software into the testing process requires a careful approach and ensuring that the generated situations are actually representative and/or useful in the system's safety and performance evaluations [167].

5.1.2. Existing Approaches

The main objective of both simulation-based and real-world driving tests is to uncover potential issues and limitations of the tested algorithms. The tests are often performed on a large scale, covering thousands or even millions of kilometers, for the purpose of uncovering weaknesses of the system that manifest themselves only in edge cases, specific conditions, situations, or combinations of these, which cannot be easily predicted by the developers. Such situations may be related to sensors' performance degradation in certain situations (e.g., in adverse weather conditions), atypical behavior of other road users (e.g., erratic driving due to intoxication, traction loss), or static obstacles that impede the understanding of road situation (e.g., construction zones).

The rarity of edge cases, in combination with their uniqueness, makes them difficult to predict or explore efficiently. Typically, the vast majority of the samples collected in the large-scale driving tests, both on-road and virtual ones, represent situations that do not pose challenges to the tested system. Due to this, large-scale driving tests, while remaining an important performance evaluation tool, are an inefficient method of exploring edge cases, as pointed out in [74].

Virtual driving tests suffer from an additional limitation that further impedes the edge case exploration - difficulty in generating unique atypical scenarios, as algorithms used for traffic simulation rarely can simulate a wide variety of individual driving styles and behaviors observed in human traffic participants [117].

At the perception level, a significant disadvantage of simulators is related to the limited number of textures and 3D models of traffic participants, which hinders their ability to test the reaction of the system to road users with atypical appearance or geometry [169]. Since these are significant sources of errors in perception systems, relying on simulation-based testing to explore edge cases in such systems creates a considerable risk of missing important errors.

On the behavior level, the limitations of simulation packages are related to the repeatability of traffic participants' behaviors. Movement trajectories of road users are typically scripted or generated by machine learning-based algorithms [127]. Because of this, exploration of the system's behavior in the presence of traffic participants who behave erratically (e.g., due to intoxication, distraction, or fatigue) may be difficult or even impossible, depending on the particular software.

5.1.2.1. High-Level Test Scenarios

The rarity of edge cases critical for ADAS / AD system evaluation observed during real-world test drives results in increased use of predefined test scenarios for these applications. Scenario-based testing, both simulation-based and real-world-based, is used for performance evaluation, validation, and certification of various ADAS/AD features [126, 124].

Scenario-based testing allows test efforts to be focused directly on critical situations, ensuring that AD/ADAS reacts accordingly to safety requirements in predictable challenging situations. On the downside, the design and execution of test scenarios that would explore the system's behavior in situations that incorporate complex situations in a cluttered environment (e.g., a large number of vulnerable road users in the vehicle's vicinity) may be difficult. Thus, tests limited to predefined scenarios only may lead to a false sense of safety, especially if tests fail to recreate complex environments and behaviors of other traffic participants that may be observed in the real world.

However, numerous studies have shown that scenario-based testing is a convenient tool to assess the safety of ADAS / AD systems, showing a strong correlation between performance scores achieved in scenario-based evaluation and real-life injury risk levels [106, 92, 168].

The type of test scenarios and the creation methods depend on the purpose of the testing procedures. Manual creation of scenarios is widely used to ensure that the system meets its functional requirements and to measure the performance of its features in the most relevant situations [194, 62, 114].

While manually crafted scenarios provide a convenient way of atomic requirements testing, the calculation of Key Performance Indicators defined for given features often requires more extensive use of real-world data. Numerous methods have been proposed to utilize information collected during test drives and naturalistic driving studies [89, 201] to derive a portfolio of test scenarios suitable for the testing of ADAS / AD systems [11, 205]. Similar methods are often proposed for testing systems in safety-critical situations, where recordings of real-life safety-critical situations can be used to derive test procedures [110].

5.1.2.2. Adversarial Scenarios Generation

While manually crafted scenarios, as well as randomly generated ones, provide important insight into the system's performance in difficult situations, not all edge cases can be easily predicted. Especially in the case of driving policies based on machine learning, there is a certain risk of triggering critical failures through seemingly minor anomalies in the observed patterns. This characteristic of deep neural networks is often exposed in the works on adversarial attacks [61, 125, 6].

In order to explore difficult situations and edge cases more efficiently, a number of automatic test scenario generation methods were proposed [163, 162]. In [202] authors proposed to generate test scenarios through the increase in difficulty of observed or random scenarios. The authors utilize reachability analysis [7] to measure the size of the solution space and apply optimization techniques to decrease it by changing the initial conditions of the scenario. Similarly, [76] proposes a method that increases the criticality of automatically generated test cases, utilizing Reinforcement Learning techniques to influence RSS safety metrics[156]. [145] focuses on the development of the scenario derivation method for the purpose of SOTIF safety analysis [66], proposing a hierarchical framework that describes scenarios with a set of variables that can be modified to acquire new potentially unsafe situations.

The methods described above enable the generation of a wide set of difficult scenarios, providing means for edge case exploration, but do not focus on finding flaws in any particular algorithm directly. This may be an inefficient approach in the case of machine learning-based techniques, where failures are not necessarily directly related to the objective difficulty of the scenario [196]. To address this issue, methods that are intended to search for vulnerabilities of particular algorithms directly are proposed. The authors of [5] presented a method for falsification of ADAS/AD systems based on searching through a set of possible decisions of other road users in discrete time points using Bayesian optimization techniques. Another search-based method was presented by Tuncali et al. in [178], where search methods were used to modify a set of predefined scenarios to trigger situations challenging for a predefined ADAS/AD system. The authors focused mainly on searching for a combination of conditions that would lead to the violation of predefined system requirements.

Since adversarial methods are often used in the machine learning context, both in falsification-based testing [196], as well as Robust Adversarial Reinforcement Learning (RARL) [140, 134], the use of the neural networks for scenario generation is often proposed. An example of such an approach is given in [187], where Reinforcement Learning (RL) in a multi-agent setup is used to generate test scenarios for a driving policy based on deterministic rules.

Adversarial networks trained in an RARL setup tend to achieve particularly good performance in finding vulnerabilities in the trained networks. As their training process is, however, strictly tied to the training of the evaluated network, the driving policy may achieve robustness to the adversarial policy, while remaining vulnerable to situations that the adversarial policy fails to generate. This issue is relatively similar to the overfitting problem common in supervised learning setups, where trained networks may learn to work perfectly in conjunction with the training data, while failing to produce satisfactory results on the independent testing datasets.

RL-based adversarial networks can, however, be trained outside the RARL setup, using an already trained driving policy. While this approach may successfully produce adversarial test scenarios, it is difficult to ensure the creation of a database with a vast variety of dissimilar scenarios. Once the adversarial agent based on an RL policy finds one way to trigger a safety failure in the tested algorithm, it may learn to exploit only this particular issue, failing to uncover further ones.

5.1.3. Motivation and General Idea

A rapidly growing demand for commercially viable Autonomous Driving systems with a high level of autonomy creates a need for effective virtual testing techniques that could uncover potential issues in increasingly complex driving policies, which often rely on non-transparent ML-based techniques. Automatic creation of adversarial scenarios may significantly help to achieve this, allowing to focus only on the scenarios that pose a significant challenge to the tested policies.

As discussed in the previous section, a number of methods for automatic test scenario generation have already been proposed. Unfortunately, most of them are not fully suitable for the generation of larger scenario databases, especially in the context of RL-based driving policy testing. Most of the methods intended for the creation of scenario databases do not necessarily produce road situations that are difficult for a particular driving policy. As in the case of RL-based policies, the objective difficulty of the road scenario may not correlate directly with the subjective difficulty for a particular policy, it is desirable to incorporate the tested policy itself in the generation of test scenarios. While RL-based falsification techniques fulfill this goal, they are rarely suitable for finding variations of a given scenario that would uncover different issues in a given policy.

The adversarial trajectory generation algorithm presented in this chapter utilizes the evaluated driving policy in the process of scenario database creation, allowing for the discovery of issues in a particular AD algorithm. Contrary to techniques focused on the increase of objective scenario difficulty measures, this approach helps to uncover failure modes not necessarily related to objectively difficult situations, which is particularly important in the case of ML-based driving policies. The proposed approach also utilizes a measure, which allows us to estimate similarity of the generated scenario to the scenarios already present in a database. The use of this measure enables the generation of multiple scenarios iteratively, ensuring that they will represent difficult situations dissimilar to the ones already generated.

The proposed method is based on stochastic optimization techniques that are used to generate a set of control trajectories for the vehicles surrounding the vehicle controlled by tested driving policy in a simulation environment. The cost terms used in the trajectories optimization process are designed to encourage solutions that pose a challenge to the driving policy. This is achieved by assigning a negative cost value based on a set of safety measures, such as the minimal distance observed between the ego and other vehicles in the simulated scenario.

Optimization problems formulated in this way can be used to generate a single adversarial scenario. To incorporate the generation of a scenario database, the optimization can be repeated, with an addition of the cost term, that encourages finding original trajectories that are dissimilar to the already generated ones. In this way, each subsequent solution of the optimization problem yields a novel scenario.

The general idea of the proposed method is presented in Fig. 5.1.

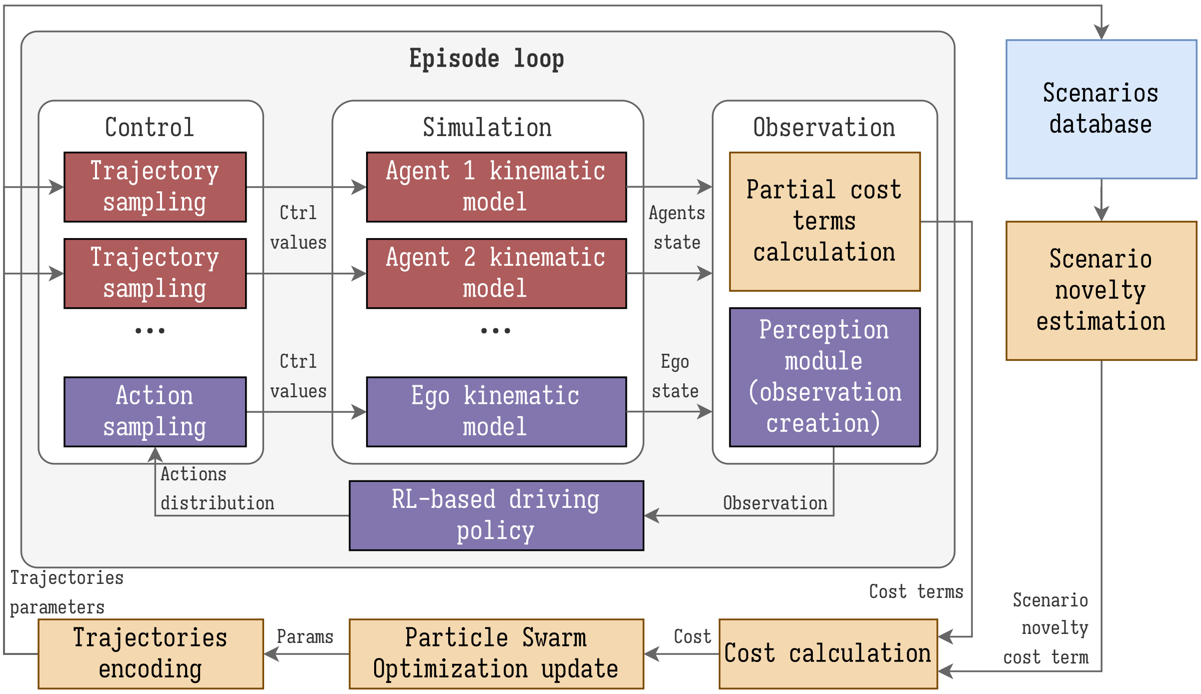


Figure 5.1. General idea of the adversarial trajectory generation method proposed in this chapter.

In addition, an extension to this method that enables the simultaneous generation of state trajectories and perception error patterns is proposed. In this extension, an *error trajectory* is generated alongside the state trajectory of each of the traffic participants (apart from the ego), which represents plausible state estimation errors that impact the ego's perception of this road user. As minimization of the state estimation errors is part of the underlying optimization problem used for trajectory generation, the method helps to identify small but critical error modalities that may trigger safety-critical mistakes of the tested policy.

5.1.4. Contributions

Virtual testing of the ADAS and AD systems is an area of extensive research that already resulted in numerous proposals for automated test scenario generation methods. Popularization of the Machine-Learning-based approaches to vehicle control and path planning creates, however, a need for a method that would be capable of an in-depth exploration of potential issues in such systems. The ability to generate multiple variants of an adversarial scenario is particularly desired, as it may help to ensure that less apparent vulnerabilities of the policy will be exposed apart from the most obvious ones.

Considering this need and the limitations of existing methods described in the previous section, in this chapter I describe a novel automated scenario generation method, bringing several contributions listed below.

- The proposal of the adversarial scenario creation method based on stochastic optimization, capable of generating multiple variants of a scenario starting from manually defined or randomly generated initial conditions.

- The proposed method allows the generation of situations that pose a challenge to an arbitrary driving policy, including the stochastic ML-based driving policies. While falsification methods (e.g., based on Reinforcement Learning) that explore vulnerabilities of particular algorithms have already been proposed, they are rarely capable of generating multiple dissimilar variants of a given scenario.

- The proposed method can be used to generate safety-critical perception error patterns alongside the trajectories of other vehicles in the scenario. While certain existing methods enable the generation of critical error patterns, the simultaneous generation of both critical errors and state trajectories of other vehicles is rarely explored and may be used to uncover vulnerabilities that are difficult to trigger using existing methods.

- The proposed method is applied to find vulnerabilities in an exemplary RL-based driving policy, providing insight into potential issues in such systems, and demonstrating the effectiveness of the proposed method.

Parts of the described method have been presented in my publication [182]. In this thesis, the proposed approach is extended in several ways: a new, more modern driving policy architecture has been used for the evaluation, further calibration allowed to improve the performance of the method, and new scenario types have been generated to further investigate the method's effectiveness. Additionally, a potential for the method's extension to the generation of adversarial perception error patterns has been identified and described.

5.2. Problem Formulation

This chapter presents a method that can be used to generate a set $\mathcal{S}_{\text{scen}}$ of n_{scen} test scenarios. Each scenario incorporates an arbitrarily (or randomly) chosen number n_{tp} of traffic participants, where the properties (geometry, kinematic parameters) of each road user are described by a parameter vector $\mathbf{p}_{\text{adv,veh,params}}$. The description of the generated scenario is composed of the ego's initial state $\mathbf{s}_{\text{ego}}(0)$ and n_{tp} state trajectories $\mathbf{s}_{\text{tp}_i}(t), \forall i \in \{1, ..., n_{\text{tp}}\}, t \in [0, t_{\text{sf}}]$, where t_{sf} is an arbitrarily chosen duration of the scenario.

Additionally, the method can be used to generate a set of the state estimate trajectories $\hat{\mathbf{s}}_{\text{tp}_i}(t), \forall i \in \{1, ..., n_{\text{tp}}\}, t \in [0, t_{\text{sf}}]$, that represent ego's imperfect perception of other traffic participants in a given scenario.

5.2.1. Assumptions

Several assumptions regarding the tested policy and generated scenarios are applied in this chapter. Unless stated otherwise, the assumptions listed below are used in all considerations in this chapter.

- The driving policy can be executed in a Software-in-Loop setup, and can be used to derive an ego's control trajectory $T_{\text{ctrl,ego}}(t)$, where $t \in [0, t_{\text{sf}}]$, based on a representation of the environment, that can be created using the description of the road geometry in ego's proximity, information about other road users' bounding boxes, and state of other road users $T_{\text{state,tp}_i}(t), \forall i \in \{1, ..., n_{\text{tp}}\}, t \in [0, t_{\text{sf}}]$.
- Ego's control trajectory $T_{\text{ctrl, ego}}(t), t \in [0, t_{\text{sf}}]$ and initial state $\mathbf{s}_{\text{ego}}(0)$ are sufficient to derive an ego's state trajectory $\mathbf{s}_{\text{ego}}(t)$ for $t \in [0, t_{\text{sf}}]$, e.g., using a known ego's dynamic model.
- False positive and false negative object detection errors, as well as static environment perception errors (or lack of them), do not have a significant impact on a policy and can be omitted from the considerations. Note that, as stated in a previous chapter, this is rarely the case in actual systems. However, such errors are mostly omitted in this chapter, as the proposed approach can be used in conjunction with the sensor models described in the previous chapter to at least partially alleviate the resulting issues.
- Ego's state estimation errors are assumed to be negligible and therefore omitted.
- The vertical profile of the road as well as environmental conditions such as weather have no impact on a policy or on constraints applied to other road users and thus are neglected in the described approach.

5.2.2. Scenario Description

A traffic scenario also referred to as a test scenario can be defined in a relatively broad manner. It may refer to both simulated situations as well as real-life ones, it may define the state of the scene and movement of the traffic participants precisely or only in a broad manner, and it may describe short, atomic situations, or long-term traffic conditions.

For the purpose of this section, I define a *passive test scenario*, as a short-term virtual scenario that describes precise movements of traffic participants in the vicinity of the ego vehicle. The movement of the road users excluding the ego in a defined passive test scenario remains unchanged between executions of the virtual scenario, even if the ego's movement changes, i.e., the environment does not actively react to the ego's behavior.

The passive test scenario is composed of several main components listed below.

Static environment description consisting of the road description (lanes geometry, lane markings description, etc.), static obstacles, traffic signs, and other elements of the environment relevant to the ego's control algorithm. The exact description may depend on the tested policy and availability of certain information (e.g., related to the presence of certain perception modules in the system). In this chapter, the static environment is always predefined (that is, the map is available), using the description defined in Section 3.2.2. As driving policies often build the static environment model based on on-board sensors, the static environment in proximity of the ego can be represented using lane markers geometry in the ego's coordinate frame, using the definition provided in Section 4.4. The environment can be thus represented as:

$$\mathcal{S}_r(\mathbf{s}_{\text{ego}}(t)) = \{\mathbf{s}_{r_i}(\mathbf{s}_{\text{ego}}(t))\}_{i=1..n_s}, \tag{5.2.1}$$

where n_s is number of the relevant lane markers, and $\mathbf{s}_{r_i} \in \mathbb{R}^5$ for $i = 1..n_s$ describe the lane markers geometry, as defined in (4.4.1).

Traffic participants properties - number of traffic participants, their type, and corresponding parameters relevant to the ego's control algorithm and/or impacting their movement (such as the size of the vehicle). In this chapter, number of the road users, as well as their geometry and type are assumed to be predefined for each of the scenarios and remain constant during the scenario itself.

State trajectories that describe the movement of traffic participants as a function of time. The set of state trajectories is defined similarly as in Section 4.3 as $\mathcal{S}_{\text{tp}}(t) = \{\mathbf{s}_{tp_i}(t)\}_{i=1..n_{aa}}$, where each of n_{aa} relevant objects is represented using a state vector:

$$\mathbf{s}_{\text{tp}_i} = \begin{bmatrix} \mathbf{g}_i \\ \mathbf{x}_i \\ \psi_i \\ v_i \\ a_i \end{bmatrix}. \tag{5.2.2}$$

In equation 5.2.2, the vector $\mathbf{g}_i \in \mathbb{R}^2$ consists of the length and width of the i-th vehicle, $\mathbf{x}_i \in \mathbb{R}^2$ denotes its lateral and longitudinal position ($\mathbf{x}_i^{\text{lon}}$ and $\mathbf{x}_i^{\text{lat}}$ respectively) in a predefined reference frame, ψ_i is the vehicle rotation in reference to this frame, v_i - its absolute speed and a_i - acceleration. Note that while $\mathbf{g}_i$ is predefined and constant for each scenario, it remains part of the state vector for convenience.

State estimate trajectories (optional) that describe the state estimates of traffic participants, i.e., the state of other vehicles with possible errors or perturbations. The set of state estimate trajectories is defined as $\hat{\mathcal{S}}_{\text{tp}}(t) = \left\{\hat{\mathbf{s}}_{\text{tp}_i}(t)\right\}_{i=1..n_{\text{tp}}}$. The state estimate vector contains estimates of the state variables described in (5.2.2):

$$\hat{\mathbf{s}}_{tp_i} = [\hat{\mathbf{g}}_i, \hat{\mathbf{x}}_i, \hat{\psi}_i, \hat{v}_i, \hat{a}_i]^T, \tag{5.2.3}$$

where values $\hat{\mathbf{g}}_i$, $\hat{\mathbf{x}}_i$, $\hat{\psi}_i$, $\hat{v}_i$, and $\hat{a}_i$ represent estimates of state variables $\mathbf{g}_i$, $\mathbf{x}_i$, ψ_i, v_i, and a_i respectively.

Initial state consisting of the ego's initial state:

$$\mathbf{s}_{\text{ego}}(0) = [\mathbf{g}_{\text{ego}}(0), \mathbf{x}_{\text{ego}}(0), \psi_{\text{ego}}(0), v_{\text{ego}}(0), a_{\text{ego}}(0)], \tag{5.2.4}$$

defined the same way as in the case of other vehicles (5.2.2), and the initial state of other traffic participants $\mathbf{s}_{\text{tp}}(0)$.

The proposed method is intended for finding the state trajectories of vehicles in proximity of the ego that fulfill given requirements, e.g., lead to situations that pose a challenge to the control algorithm in a simulation. While other elements of the scenario can be included in the optimization problem used for scenario generation as well, for the purpose of this chapter, I will assume that they are predefined, either manually or automatically, e.g., based on situations observed in real-world test drives, or through a random generation process. Since the method focuses on the generation of the trajectories that lead to challenging road situations, that is, adverse behaviors toward ego vehicles, it can be referred to as *Adversarial Trajectories Generation* method.

5.2.3. System Under Test

The method is intended to enable the validation and verification of ADAS/AD systems that are used for vehicle control. In particular, the method can be used to test a policy π_{sut} that maps the environment model $\hat{\mathbf{E}}_{\text{env}}(t)$ to an action $\mathbf{c}_{\text{ego}}(t)$ that will be executed by a system.

In the examples used in this chapter, the environment model is defined as:

$$\hat{\mathfrak{S}}_{\text{env}}(t) = \{\mathcal{S}_{\text{tp}}(t), \mathcal{S}_r(t), \mathbf{s}_{\text{ego}}(t)\} \tag{5.2.5}$$

Note that the description includes the ego's state $\mathbf{s}_{\text{ego}}(t)$, which, while technically not a part of the environment, has been included in the model for convenience.

The action $\mathbf{c}_a(t)$ describes a vector of control values sufficient to solve the forward dynamics problem of the ego, knowing its dynamic model and state $\mathbf{s}_{\text{ego}}(t)$. Control variables as well as utilized ego's mathematical model are arbitrary, as long as they can be used to compute its state in next-time instances.

The definition of a System Under Test (SUT) is intentionally broad. It allows for relatively complex setups, e.g., the system may include sensor models (for example, the models defined in the previous chapter) that will map the environment's state $\mathfrak{S}_{\text{env}}$ to a vector of state estimates $\left\{\hat{\mathcal{S}}_{\text{tp}}(t), \hat{\mathcal{S}}_r(t), \hat{\mathbf{s}}_{\text{ego}}(t)\right\}$. The system may also be composed of several modules, e.g., a planning module that outputs the reference trajectory and a low-level control module that implements feedback control algorithms to follow the reference. Note, however, that such setups were not tested with the proposed method, as a Reinforcement-Learning-based direct control policy has been used for this purpose.

5.2.4. Traffic Participants Model

While generated scenarios describe the dynamic environment of the ego through a set of state trajectories of traffic participants $\mathcal{S}_{\text{tp}}(t)$, the underlying stochastic optimization method actually generates their control trajectories $\mathbf{c}_{\text{tp}_i}$, which are later used to derive the state trajectories.

The state trajectories of traffic participants in each scenario $S \in \mathcal{S}_{\text{scen}}$ are calculated based on a set $\mathcal{T}_{\text{ctrl}}$ of n_{tp} control trajectories $\mathbf{c}_i(\mathbf{q}_i, t)$ for $i = 1..n_{\text{tp}}$, where $\mathbf{q}_i$ is a vector of n_{q} parameters describing the $i-th$ trajectory. Each control trajectory $\mathbf{c}_i \in \mathcal{T}_{\text{ctrl}}$ describes values of the control variables $\mathbf{c}_i(t)$ defined as:

$$\mathbf{c_i}(\mathbf{q}_i, t) = \begin{bmatrix} \delta_i(\mathbf{q}_i, t) \\ a_i(\mathbf{q}_i, t) \end{bmatrix},$$

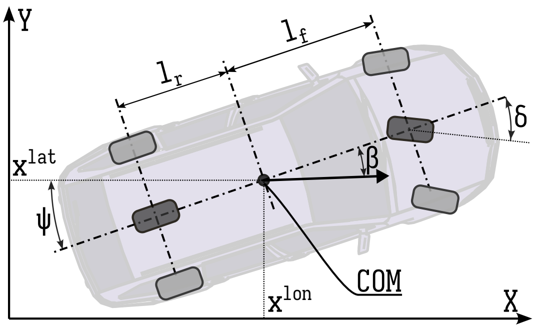
where $\delta_i(\mathbf{q}_i, t)$ denotes the steering angle and $a_i(\mathbf{q}_i, t)$ the acceleration of $i-th$ vehicle in the scenario at time t. Each of the road users present in the scenario, except for the ego vehicle, moves according to a predefined dynamic model, which, knowing the initial state and control variables $\mathbf{c}_i(t)$ for $t \in [0, t_{\text{sf}}]$, allows one to derive its state $\mathbf{s}_{\text{tp}_i}(t)$.

5.2.4.1. Kinematic Model

The evolution of the vehicle state in time depends on the control variables and is described using a kinematic model. The model is a variant of a bicycle kinematic model of a road vehicle [141], which is commonly used to describe the movement of vehicles in ADAS/AD systems. The bicycle model approximates the kinematic properties of a four-wheeled vehicle using a two-wheeled model that resembles a bicycle with negligible tire widths.

Note that the identical model has been used in the previously described Multiple Hypothesis Planning methods, and thus its full description with the additional rationale behind its choice can be found in Section 3.2.5.

It should be noted that the model neglects many phenomena that impact vehicle movement, such as wheel friction, air drag, the influence of road profile or engine dynamics. These simplifications are frequently used in traffic simulation and perception/prediction systems, as their impact on the final results of relevant algorithms is often not significant [85] and requires knowledge of variables that are difficult to measure (such as friction coefficients).



$$\begin{aligned}
\dot{\mathbf{x}}^{\text{lon}} &= v\ cos\left(\psi + \beta\left(\delta\right)\right) \\
\dot{\mathbf{x}}^{\text{lat}} &= v\ sin\left(\psi + \beta\left(\delta\right)\right) \\
\dot{\psi} &= \frac{v}{l_r} sin\left(\beta\left(\delta\right)\right) \\
\dot{v} &= a, \\
&\text{where} \\
\beta(\delta) &= tan^{-1}\left(\frac{l_r}{l_f + l_r} tan\left(\delta\right)\right).
\end{aligned} \tag{5.2.6}$$

Figure 5.2. Kinematic model used for state trajectory calculation. The model is defined in detail in Section 3.2.5.

5.3. Optimization Problem

The scenario generation is performed through a stochastic optimization of the parameter vector $\mathbf{q} = [\mathbf{q}_1, ..., \mathbf{q}_{n_{\text{tp}}}]$ composed of parameters vectors $\mathbf{q}_i \in \mathbb{R}^{n_q}$, that describe the multi-dimensional control trajectories of n_{tp} non-ego vehicles in the scenario. Parameters may encode the trajectories in an arbitrary way, e.g., using B-splines or polynomials. The trajectories describe the values of the control variables as a function of time for $t \in [t_0, t_f]$, where t_0 is typically set to 0, as the trajectories start at the beginning of the scenario, and the duration $t_s = t_f - t_0$ is chosen arbitrarily (although it can easily be included in the vector of parameters $\mathbf{q}$ and optimized alongside other scenario parameters).

The generation of adversarial trajectories is performed by solving the following non-linear constrained optimization problem.

$$\begin{aligned}
\min_{\mathbf{q}} \qquad & f(\mathbf{q}) = \sum_{j=1}^{m} w_j C_j(\mathbf{q}) \\
\text{subject to:} \qquad & \mathcal{R}_{\text{phys}}(\mathbf{q}) \geq \mathbf{0}
\end{aligned} \tag{5.3.1}$$

where $C_j(\mathbf{q}_o)$ for $j = 1..m$ denotes the cost terms evaluated based on simulation of the traffic scenario, where m is the number of cost terms. Cost terms are scaled by heuristically chosen weights w_j for $j = 1..m$. $\mathcal{R}_{\text{phys}}(\mathbf{q})$ describes a set of inequality constraints used to ensure the physical feasibility of the generated trajectories.

It should be noted that the evaluation of the cost terms requires performing the simulation of a traffic scenario with the ego vehicle controlled by the tested algorithm. Since movements of the ego impact the cost values and depend on possibly severely nonlinear and/or stochastic driving policies, the problem itself may also be stochastic and nonlinear; thus, use of gradient-based optimization methods is rarely a viable way of solving it. For this reason, the use of stochastic derivative-free optimization algorithms is advisable.

5.3.1. Cost Terms

The cost terms used for the generation of adversarial trajectories are listed below.

Front collision cost term intended to promote solutions that lead to a collision between the ego's front bumper and one of the traffic participants. In particular, the front collision of the ego is defined as a state in the simulation, where there is an overlap between the bounding box of the traffic participant and the bounding box of the ego vehicle, and the ego's Center of Mass (CoM) position ($\mathbf{x}_{\text{ego}}$) is behind the other vehicle's CoM position ($\mathbf{x}_{\text{other}}$), so $\mathbf{x}_{\text{ego}}^{\text{lon}} > \mathbf{x}_{\text{other}}^{\text{lon}}$.

The cost term is evaluated after the simulation, which is terminated at the time of the first collision observed in the simulated scenario, or at t_f if the collision is not observed during the episode. The time at which the simulation is terminated is denoted t_t.

$$C_{\text{front,collision}}(\mathbf{q}) = \sum_{i=1}^{n} \begin{cases} -C_{\text{coll}_i}(\mathbf{q}_i) & \text{if } v_{s,\text{ego}}(t_t) > v_{s,\text{tp}_i}(t_t) \text{ and } f_{\text{coll}_i}(\mathbf{q}, t_t) = 1, \\ -p_{\text{side,coll}} & \text{if } v_{s,\text{ego}}(t_t) \leq v_{s,\text{tp}_i}(t_t) \text{ and } f_{\text{coll}_i}(\mathbf{q}_o, t_t) = 1, \\ 0 & \text{if } f_{\text{coll}_i}(\mathbf{q}_o, t_t) = 0. \end{cases} \tag{5.3.2}$$

where $f_{\text{coll}_i}(\mathbf{q}, t)$ is equal to 1 if the $i-th$ traffic participant collides with the ego at time t, while the ego's center is behind the vehicle's center (in a road-aligned Frenet coordinate system), and 0 otherwise.

The term C_{coll_i} is defined as:

$$C_{\text{coll}_i}(\mathbf{q}_i) = p_{\text{fc},1} * (v_{s,\text{tp}_i}(\mathbf{q}_i, t_t) - v_{s,\text{ego}}(\mathbf{q}_i, t_t)) + p_{\text{fc},2}, \tag{5.3.3}$$

where $v_{s,\text{tp}_i}(\mathbf{q}, t_t)$ is the longitudinal velocity of the i-th agent in the Frenet coordinate system, $v_{s,\text{ego}}$ is the longitudinal velocity of the ego vehicle, while $p_{\text{fc},1}$, $p_{\text{fc},2}$, and $p_{\text{side,coll}}$ are calibration parameters.

Values of the parameters $p_{\text{fc},1}, p_{\text{fc},2}$ are chosen in a way that rewards collisions, in which the ego vehicle moves faster than the relevant traffic participant, following the intuition that such front collisions are of the greatest importance, as it typically indicates the ego's responsibility for the collision. The cost term in this case depends on the longitudinal velocity difference, to promote solutions in which the accident is more severe.

Front collisions in which other traffic participant moves faster are less plausible, but can still occur under the proposed definition of the front collision - e.g., when ego's front bumper collides with the rear part of the other vehicle during the lane change maneuver, while one of the vehicles moves laterally. In such situations, a predefined negative cost depending on the $p_{\text{side,coll}}$ parameter is assigned to promote them.

The parameters are chosen in such a way that the 0 value assigned for lack of collisions is always larger than the values assigned for the collision occurrence, so that the *sum* operation will result in assigning the negative cost when any of the traffic participants collide with the ego.

Rear collision occurrences, while less desirable than front collision, are still promoted by an additional cost term $C_{\text{rear,coll}}$. Similarly, as in the case of the front collision, the rear collision is defined as an overlap between the rear part of the ego vehicle and another traffic participant. The cost term $C_{\text{rear,coll}}$ is defined as:

$$C_{\text{rear,coll}}(\mathbf{q}) = \sum_{i=1}^{n} \begin{cases} p_{\text{rc},1} & \text{if } \dot{v}_{s,\text{ego}} > 0 \\ \dot{v}_{s,\text{ego}}(t_t) * p_{\text{rc},2} & \text{otherwise}, \end{cases} \tag{5.3.4}$$

where $v_{s,\text{ego}}$ denotes the longitudinal velocity of the ego vehicle, and $p_{\text{rc},1}, p_{\text{rc},2}$ are calibration parameters. $p_{\text{rc},1}$ has a positive value, resulting in the assignment of positive values when the ego accelerates. This choice discourages solutions in which other traffic participants deliberately chase the accelerating ego vehicle, as such situations are implausible. On the other hand, the deceleration of the ego during the rear-end collision may suggest that the collision is caused by the sudden braking of the ego, and thus $p_{\text{rc},2}$ has a negative value, resulting in a negative cost proportional to the ego's acceleration.

Time to Collision (TTC) is a measure commonly used in ADAS features, expressing the time in which a collision involving the ego is expected to occur, assuming that all road users (including the ego) will sustain their current acceleration. If the collision is not expected to occur under these conditions, the TTC has a ∞ value.

The cost term that promotes the achievement of low TTC values is defined as:

$$C_{\text{TTC}}(\mathbf{q}) = \min \left\{ \min_{t \in [t_0, t_f]} (f_{\text{TTC,i}}(\mathbf{q}, t)), p_{\text{TTC,max}} \right\} \; for \; i = 1..n_{\text{tp}}, \tag{5.3.5}$$

where $f_{\text{TTC,i}}(\mathbf{q}, t)$ denotes $i - th$ agent TTC at time t, while $p_{\text{TTC,max}}$ is a calibration parameter.

The TTC cost term serves as a way to guide the optimization process toward dangerous situations and promote dangerous low-TTC solutions.

Euclidean distance cost term promotes solutions in which the distance between ego and at least one of the other traffic participants is small. It should be noted that without this term, the gradient of the cost function may be zero in potentially large areas of solution space, i.e. changes in the optimization vector $\mathbf{q}$ do not impact the cost for a wide range of the $\mathbf{q}$ values. In particular, cost remains the same for all $\mathbf{q}$ that result in lack of collisions and $f_{\text{TTC,i}}(\mathbf{q}, t) > p_{\text{TTC,max}} \; for \; i = 1..n_{\text{tp}}, t \in [t_0, t_f]$.

While it is strongly suggested to use gradient-free optimization methods for solving the proposed optimization problem, this situation may still severely impact the performance of the stochastic solvers, or even render the problem impossible to solve with certain algorithms. To avoid these issues, the following c_{ed} cost term is introduced:

$$C_{\text{dist}}(\mathbf{q}) = \min_{i=1..n_{\text{tp}}} \left(\min_{t\in[t_0,f_t]} \left(\sqrt{(\mathbf{x}^{\text{lon}}_{\text{ego}}(\mathbf{q},t) - \mathbf{x}^{\text{lon}}_{\text{i}}(\mathbf{q},t))^2 + (\mathbf{x}^{\text{lat}}_{\text{ego}}(\mathbf{q},t) - \mathbf{x}^{\text{lat}}_{\text{i}}(\mathbf{q},t))^2} \right) \right), \tag{5.3.6}$$

where $\mathbf{x}^{\text{lon}}_{\text{ego}}(\mathbf{q},t), \mathbf{x}^{\text{lat}}_{\text{ego}}(\mathbf{q},t)$ $\mathbf{x}^{\text{lon}}_{\text{i}}(\mathbf{q},t), \mathbf{x}^{\text{lat}}_{\text{i}}(\mathbf{q},t)$ are CoM coordinates of ego and $i-th$ agent respectively in an arbitrary Cartesian coordinate system at the time t.

The weight assigned to this cost term is relatively small; the main goal of the C_{dist} term is to guide the optimization process in its initial stages toward the $\mathbf{q}$ values that result in low TTC with respect to one or more traffic participants.

5.3.2. Constraints

Due to the physical limitations of the vehicles, inequality constraints $\mathcal{R}_{\text{phys}}(\mathbf{q}) \geq \mathbf{0}$ are needed to enforce limits on the control values. For an arbitrary trajectory encoding method, the constraints for control values $\mathbf{c}(\mathbf{q})$ can be defined in a general way as:

$$\begin{aligned} a_{\min} \leq a_i(\mathbf{q},t) \leq a_{\max} \text{ for } i = 1..n_{\text{tp}},\ t \in [t_0, t_f]; \\ \delta_{\min} \leq \delta_i(\mathbf{q},t) \leq \delta_{\max} \text{ for } i = 1..n_{\text{tp}},\ t \in [t_0, t_f], \end{aligned} \tag{5.3.7}$$

where $a_{\max}, a_{\min}$ are acceleration limits that approximate the typical performance of road vehicles, and $\delta_{\max}, \delta_{\min}$ denote the steering angle limits typical for vehicles. Constraints are typically evaluated in discrete time steps.

It should be noted that for certain trajectory encoding methods, the problem of constraining the control values can be reduced to a trivial problem of limiting the trajectory parameters themselves - this is the case for the proposed piece-wise linear trajectory encoding.

5.3.3. Iterative Generation of Multiple Scenarios

The method described so far is suitable for the generation of a singular scenario. While the proposed optimization problem is likely to contain multiple local minima that can be explored using a suitable optimization algorithm, the generation of multiple dissimilar scenarios may be troublesome, especially if there is a pronounced global solution.

In order to enable efficient generation of multiple scenarios, additional cost terms can be added to the optimization problem:

$$C_{\text{sim}}(\mathbf{q}) = p_{\text{sim},1} \sum_{j=0}^{n_s} \frac{d(\mathbf{q}_j, \mathbf{q})}{n_s} + p_{\text{sim},2} \min_{\mathbf{q}_k \in \mathfrak{S}} d(\mathbf{q}_k, \mathbf{q}), \tag{5.3.8}$$

where $\mathcal{S}_{\text{scen}}$ is a set of n_s previously generated scenarios (parameter vectors), and $d(\mathbf{q_k}, \mathbf{q})$ is an Euclidean distance between vectors $\mathbf{q}_k$ and $\mathbf{q}$. $p_{\text{sim},1}$ and $p_{\text{sim},2}$ are the calibration parameters.

The scenarios in this setup are generated iteratively, and each scenario is added to the set $\mathcal{S}_{\text{scen}}$, impacting the next execution of the optimization problem. This way an arbitrary number of dissimilar scenarios can be generated.

The C_{sim} term promotes solutions that are dissimilar to a mean solution $\mathbf{q}_{\text{mean}} = \sum_{j=0}^{n_s} \frac{d(\mathbf{q}_j, \mathbf{q})}{n_s}$, as well as ones that have a large distance to the closest already generated scenario.

5.3.4. State Estimation Errors

As mentioned in the introduction to this chapter, the described method can be extended to generate scenarios that include dynamic object state estimation errors. The weighted sum of the state variables' error integrals in time is included as a cost term of the optimization problem (5.3.1), enabling the pursuit of situations in which minuscule state estimation errors lead to critical safety failures.

The general idea of such an extension is presented in Fig. 5.3.

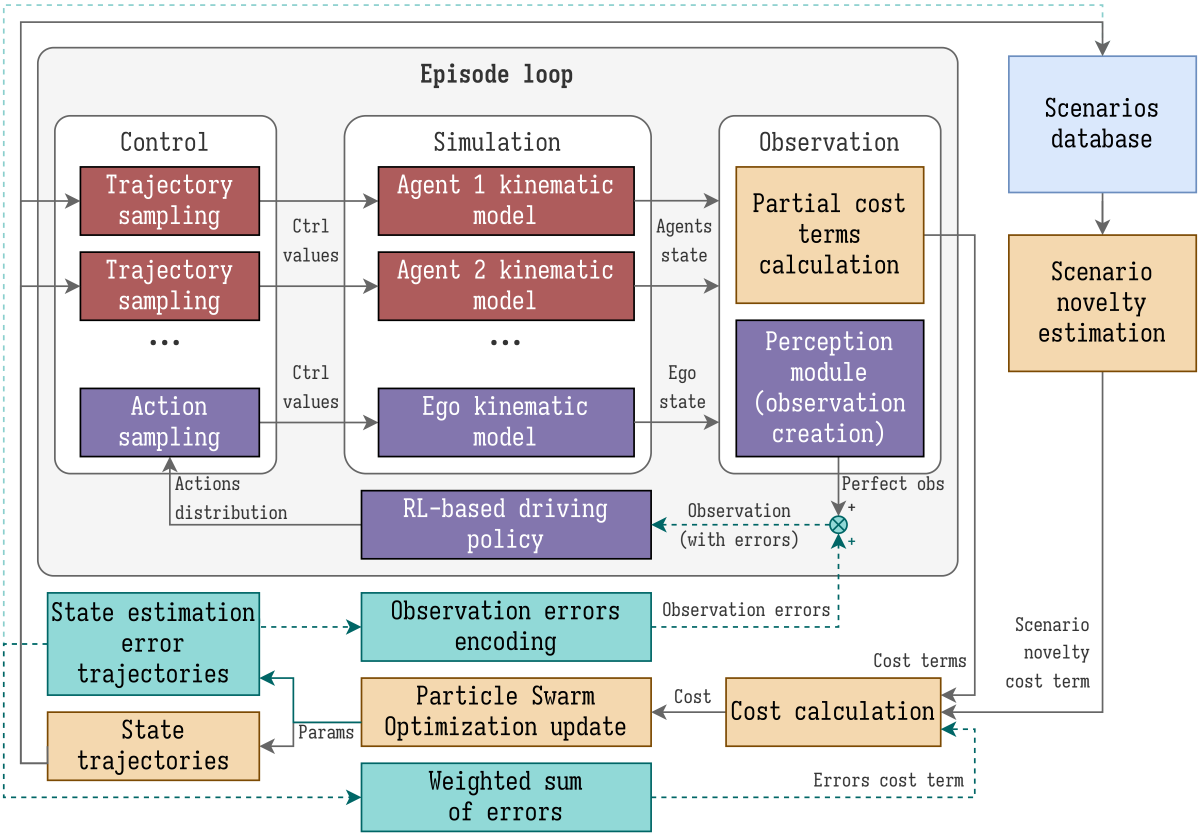


Figure 5.3. General idea of adversarial scenario generation with state estimation errors.

The scenario generation problem, in this case, is defined similarly as in 5.3.1:

$$\begin{aligned} \min_{\mathbf{q},\mathbf{q}_{\mathrm{se}}} \quad & f(\mathbf{q},\mathbf{q}_{\mathrm{se}}) = \sum_{j=1}^{m} w_j C_j(\mathbf{q},\mathbf{q}_{\mathrm{se}}) + w_{\mathrm{se}} C_{\mathrm{se}}(\mathbf{q},\mathbf{q}_{\mathrm{se}}) \\ \text{subject to:} \quad & \mathcal{R}_{\mathrm{phys}}(\mathbf{q}) \geq \mathbf{0} \end{aligned} \tag{5.3.9}$$

where $\mathbf{q}_{\mathrm{se}}$ is a vector of parameters that describe the state estimates trajectories of traffic participants $\hat{\mathbf{s}}(\mathbf{q}_{\mathrm{se}}, t) = \{\hat{\mathbf{s}}_{\mathrm{se}_i}(\mathbf{q}_{\mathrm{se}}, t)\}_{i=1..n_{\mathrm{tp}}}$ for $t \in [0, t_{\mathrm{sf}}]$.

The cost terms $C_j(\mathbf{q}\forall j \in \{1, ..., m\}$ in the described problem remain identical to the previous formulation, but a supplementary term $C_{\mathrm{se}}(\mathbf{q}, \mathbf{q}_{\mathrm{se}})$ is added, defined as:

$$C_{\mathrm{se}}(\mathbf{q}, \mathbf{q}_{\mathrm{se}}) = \sum_{i=1}^{n_{\mathrm{tp}}} \int_0^{t_f} \|\mathbf{s}(\mathbf{q}_o) - \hat{\mathbf{s}}(\mathbf{q}_{\mathrm{se}})\|_1 \, dt. \tag{5.3.10}$$

The cost term (5.3.10) scaled by a weight w_{se} promotes solutions, in which the difference between the state trajectory $\mathbf{s}(\mathbf{q}_o, t)$ (approximated using numerical integration methods) and the state estimate trajectory $\hat{\mathbf{s}}(\mathbf{q}_{\mathrm{se}}, t)$ (defined explicitly as a B-spline parameterized by the vector $\mathbf{q}_{\mathrm{se}}$) is minimal.

It can be noted that the problem can be simplified by using the $\mathbf{q}_{\mathrm{se}}$ vector to describe a state estimate error trajectory instead of a state estimate trajectory directly, as outlined in Fig. 5.3. In addition, in the practical implementation, the integral in the equation (5.3.10) is approximated using a Riemann sum.

5.4. Evaluation

The described method has been tested in an experimental setup, where the proposed approach was used to generate a set of challenging scenarios for an AD system controlled by a driving policy based on Reinforcement Learning (RL).

5.4.1. System Under Test

The driving policy subjected to evaluation using the proposed method in the described experiment is the G-SM or GT policy described in Section 4.5, i.e., a direct control driving policy based on a neural network trained in RL setup with ground-truth sensor models.

The policy outputs a control vector $\mathbf{c}_{\mathrm{ego}}(\mathbf{O}_{\mathrm{env}}) = [\delta, a]$, where $\mathbf{O}_{\mathrm{env}} \in \mathbb{R}^{n_{embd} \times (1+1+n_{tp}+n_{lm})}$ is an observation vector, which includes, as described in Section 4.5.1, the ego's state observation $\mathbf{o}_{\mathrm{ego}}(\mathbf{s}_{\mathrm{ego}}(t))$, description of other road users $\mathbf{o}_{\mathrm{obj}_i}(\mathbf{s}_{\mathrm{tp}}(t)), \forall i \in \{1, ..., n_{\mathrm{obj,max}}\}$, and the description of lane markers geometry $\mathbf{o}_{\mathrm{lm}_i}(\mathbf{s}_r(t))\forall i \in \{1, ..., n_{\mathrm{lm,max}}\}$.

The observation vector is created based on the environment model $\hat{\mathfrak{S}}(t)$ in a relatively straightforward manner, where the state of other road users and lane markers is transformed to the ego's coordinate system and scaled to $[-1, 1]$ limits.

As the policy outputs a control vector, and values of the cost terms in the optimization problem (5.3.1) depend mostly on the ego's state $\mathbf{s}_{\text{ego}}(t)$, a kinematic model (5.2.6) is used to derive the ego's state trajectory.

Observation creation and policy inference are performed every $\delta t_{\text{infer}} = 0.1s$, and after the inference, the ego's control values are kept constant until the next control update. The entire simulation loop, i.e., vehicle state updates and calculation of partial rewards, is repeated every $\delta t_{\text{sim}} = 0.05s$, and the ego's state trajectory is, in fact, approximated using numerical integration.

The policy is capable of performing sharp evasive maneuvers, such as braking and steering, in response to dangerous situations. To demonstrate this ability, a manually-crafted passive test scenario has been used, as shown in Fig. 5.4.

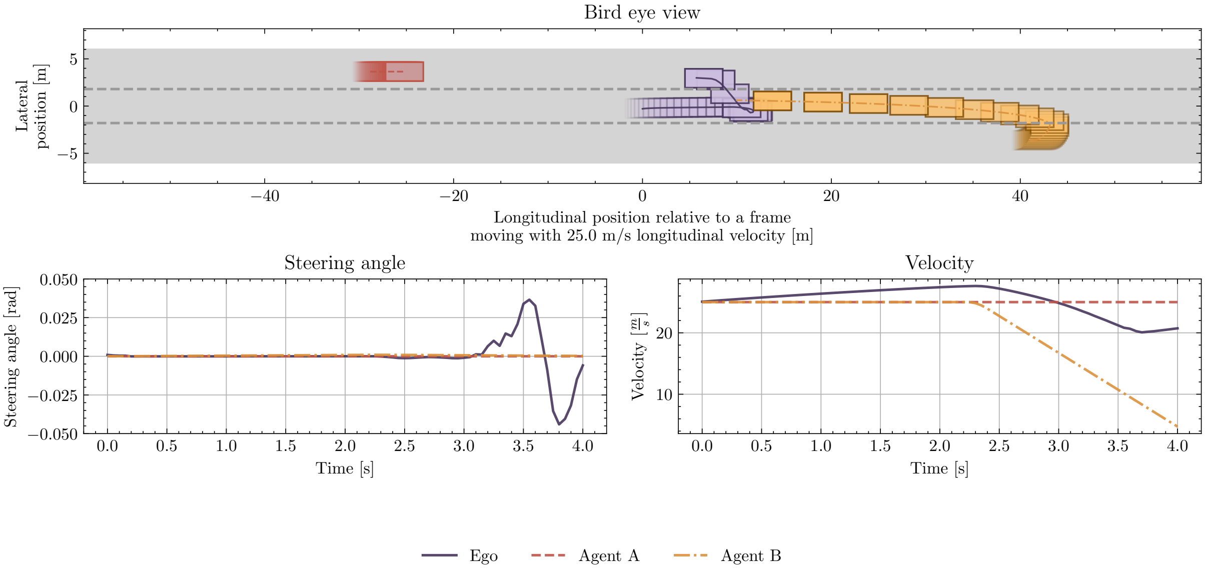


Figure 5.4. Successful collision avoidance maneuver executed by the evaluated policy in a manually-crafted test scenario.

The scenario consisted of the ego vehicle driving in the middle lane of a 3-lane highway, and two traffic participants, where one (denoted Agent A on Fig. 5.4), driving behind the ego on a left-most lane, keeps its velocity, while another (Agent B), initially driving on the right lane in front of the ego, performs a lane change to the ego's lane, followed by severe braking. All vehicles, including the ego, start the scenario with an initial velocity of $25\frac{m}{s}$.

In the described scenario, the ego controlled by the evaluated policy initially kept its lane, while moderately accelerating. As soon as the adverse agent entered the ego's lane and started braking (around $t = 2.3s$, as seen in the velocity plot), the policy applied severe braking. Since the agent maintained its severe acceleration, the ego followed its emergency braking with an evasive lane change maneuver, executed around $t = 3.3s$, as shown on the steering angle plot in Fig. 5.4. The maneuvers performed were sufficient to avoid a collision in this situation.

Further details on the environment in which the policy was trained, as well as its performance, can be found in Section 4.5. The experiment presented in Fig. 5.4 indicates, however, that the policy is capable of relatively complex and severe collision avoidance maneuvers, and thus finding the scenarios that lead to a collision may be a nontrivial task.

5.4.2. Evaluation Examples

A set of experiments was performed to observe the ability of the method to produce adversarial scenarios for the driving policy described in 5.4.1. Initial conditions for the experiments, as well as the number of adversarial agents, were created manually.

Table 5.1. Weights and parameters used for generation of adversarial scenarios.

Parameter Name	Value	Unit	Description
w_{coll}	15.0	-	Collision weight.
$p_{\text{side,coll}}$	0.2	-	Side collision multiplier.
$p_{\text{fc},1}$	0.01	-	Front collision velocity cost parameter.
$p_{\text{fc},2}$	1.0	-	Front collision constant cost parameter.
w_{rc}	1.0	-	Rear collision weight.
$p_{\text{rc},2}$	10.0	-	Rear collision constant cost parameter.
$p_{\text{rc},2}$	1.0	-	Rear collision velocity-dependent cost parameter.
w_{TTC}	-2.0	-	Time to collision weight.
$p_{\text{TTC,max}}$	10.0	[s]	TTC saturation parameter.
w_{dist}	0.3	-	Euclidean distance cost weight.
a_{min}	-4.0	$\frac{m}{s^2}$	Min acceleration of traffic participants.
a_{max}	4.0	$\frac{m}{s^2}$	Max acceleration of traffic participants.
δ_{min}	0.125	-	Min steering angle of traffic participants.
δ_{min}	0.125	-	Max steering angle of traffic participants.
w_{sim}	1.0	-	Weight of the scenario similarity cost term.
$p_{\text{sim},1}$	1.0	-	Scenario similarity cost term parameter.
$p_{\text{sim},2}$	1.0	-	Scenario similarity cost term parameter.

5.4.2.1. Experiment 1: Highway driving with two adverse agents

The main purpose of Experiment 1 was to evaluate the method's ability to generate a larger set of dissimilar scenarios involving multiple agents. The scenario generation in this example did not involve the state estimate errors.

For the first experiment, a straight three-lane road with four traffic participants (apart from the ego) was used. The ego vehicle starts the scenario in the middle lane, driving at a speed of $25\frac{m}{s}$. Two agents are placed on adjacent lanes: one at the right lane in front of the ego, at a distance of approximately $40m$, and one behind the ego at the left lane, at a distance of approximately $30m$. Both agents move at a speed of $25\frac{m}{s}$ at the beginning of the scenario, and their trajectories are generated using the described method. In addition, two passive agents are added on the middle lane in front of and behind the ego vehicle. Passive agents move at a constant speed of $25\frac{m}{s}$ throughout the scenario and maintain a steering angle of 0. The maximum duration of the scenario was set to $9s$.

An overview of the initial conditions is presented in Fig. 5.5.

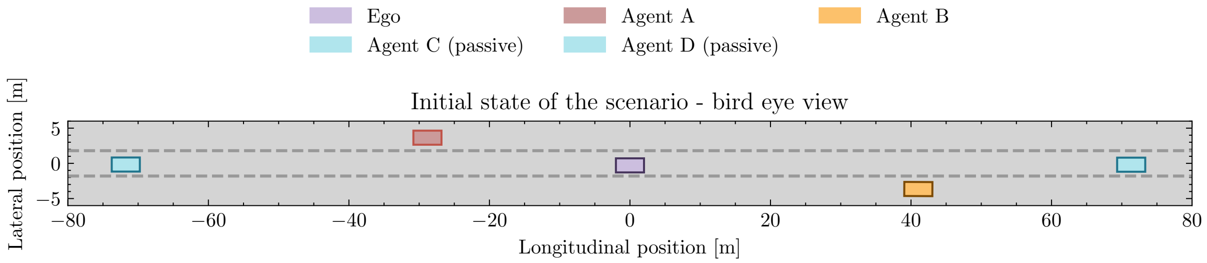


Figure 5.5. Initial conditions for the first experiment. Trajectories of Agents A and B are generated using the described method, while passive agents C (in front of the ego) and D (in the rear) maintain zero acceleration and steering angle. The ego is controlled by the driving policy under test.

The proposed method has been executed iteratively to generate 20 adversarial scenarios. Examples of the generated scenarios are discussed in the following paragraphs.

5.4.2.1.1 Example 1

In example 1, shown in Fig. 5.6, the front vehicle (Agent B) performed a severe braking maneuver, followed by an acceleration and a sudden change of lane to the left. In response to the severe cut-in maneuver, the ego executed an aggressive steering maneuver, which resulted in a collision with Agent A, which had been accelerating in the left lane.

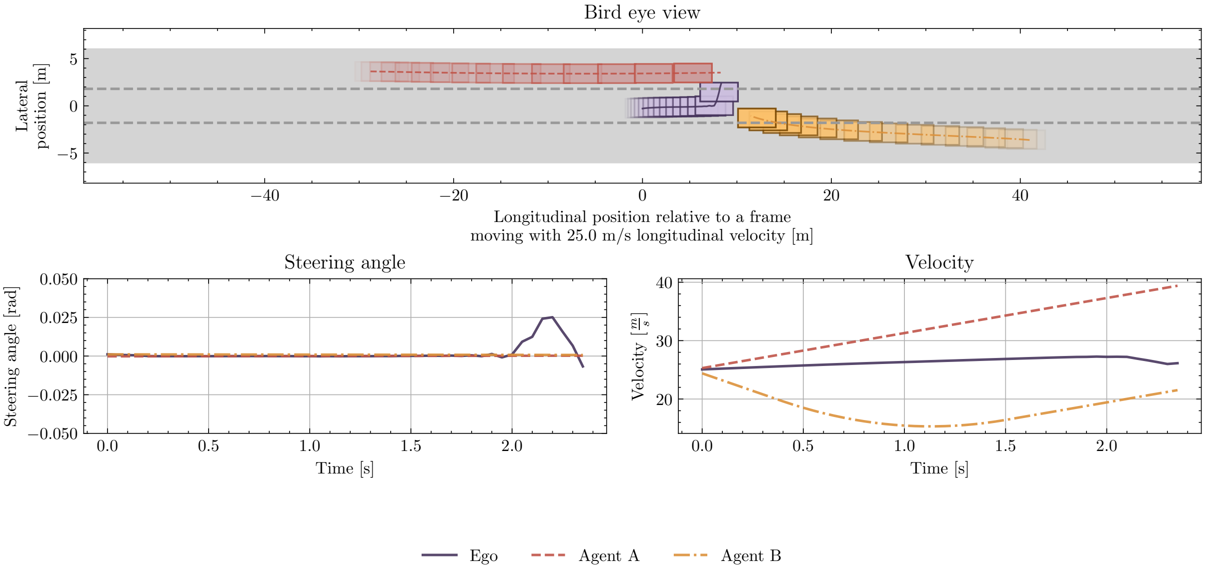


Figure 5.6. Experiment 1, scenario example 1: front vehicle braking with a late cut-in.

The method exposed an important vulnerability of the policy. While in many cases lane change is a viable collision avoidance maneuver, the policy failed to recognize a danger posed by a fast-moving traffic participant on the target lane. One possible way to address this issue may be to increase the number of dense traffic scenarios generated during the training.

5.4.2.1.2 Example 2

Example 2 presented in Fig. 5.7 shares certain similarities with Example 1, also consisting of Agent A accelerating on its lane, and Agent B performing a cut-in with braking. However, the cut-in is performed in a less severe manner in this scenario, with Agent B only partially entering the central lane. The policy fails to recognize this as a dangerous situation, maintaining the ego's velocity and steering angle.

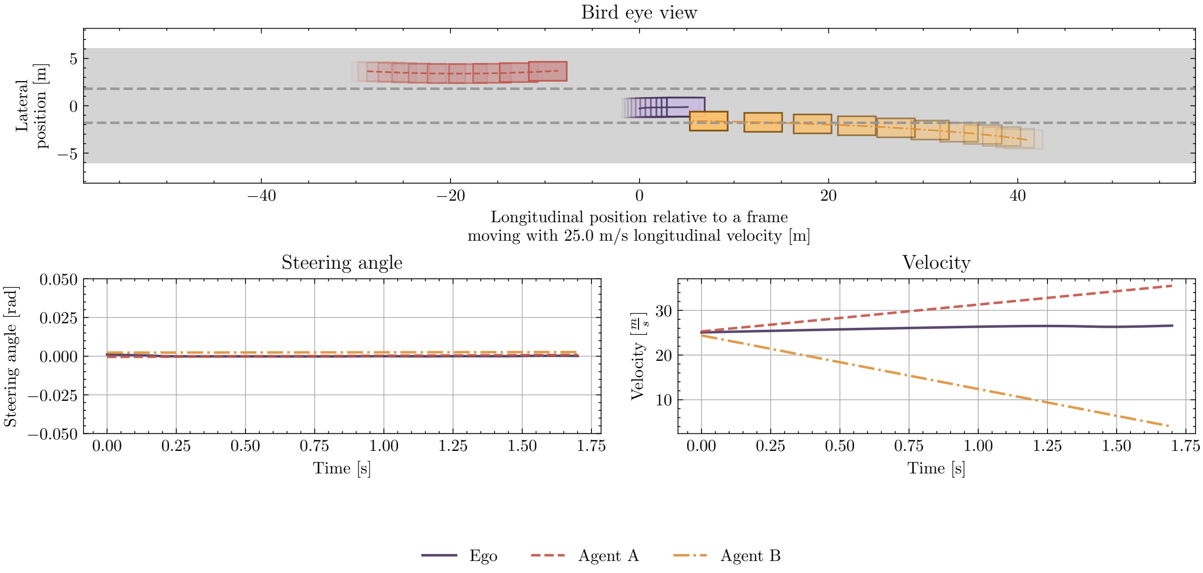


Figure 5.7. Experiment 1, scenario example 2: partial cut-in.

Significant differences in the policy's reaction to scenario examples 1 and 2 may be worth further investigation. While it is not certain why the policy did not react to Agent B's maneuver in this case, one possible theory is that the simulation package utilized for the policy's training generated only relatively fast lane change maneuvers, and the policy rarely observed situations in which the other road user drives in between the lanes for prolonged periods of time.

5.4.2.1.3 Example 3

In this example, shown in Fig. 5.8, the front vehicle (Agent B) changed the lane to the central one and then applied a prolonged severe braking. Agent A did not participate in the scenario, as it applied severe braking, promptly leaving the ego's proximity.

The ego reacted by braking, yet failed to apply sufficient braking deceleration to avoid the accident. At the same time, the ego failed to perform an evasive steering maneuver, though,

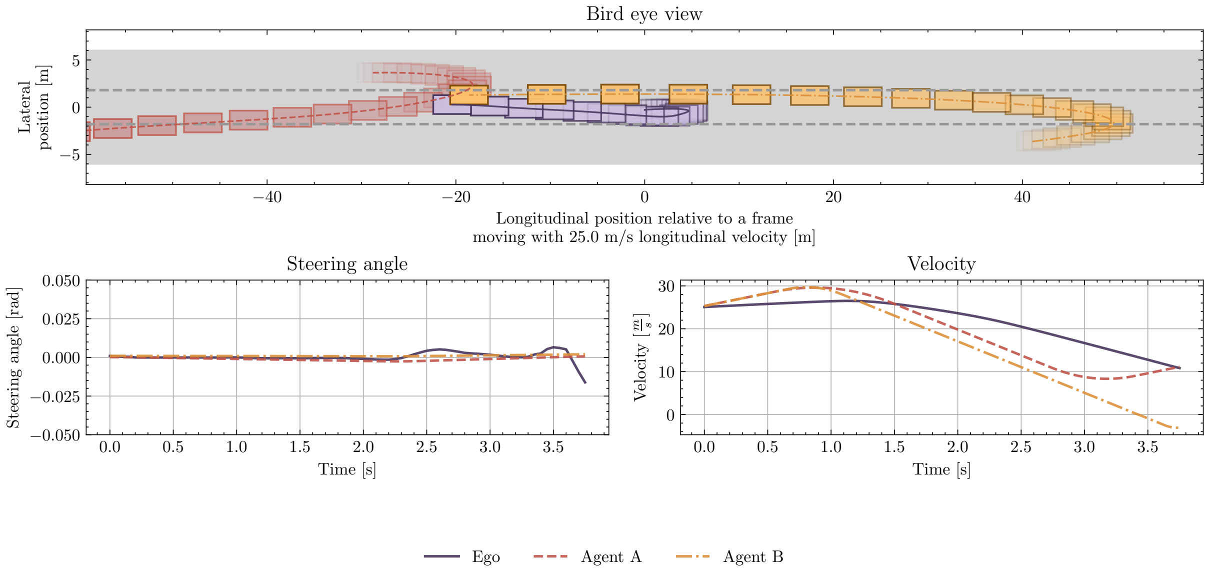


Figure 5.8. Experiment 1, scenario example 3: front vehicle braking on the ego's lane.

interestingly, its initial reaction consisted of moderate steering to the left but was followed by returning to the lane's center.

Although in this case, the initial reaction of the policy was correct, the collision could easily be avoided by changing the lane to the right. It is not certain why the police failed to perform such a maneuver - it may be related to Agent B driving at the edge of the lane, similarly as in the previous example.

5.4.2.1.4 Scenarios distribution

Iterative execution of the method to generate 20 scenarios based on identical initial conditions produced a varied set of situations that ended in the collision with the ego vehicle. The generated scenarios can be assigned to several classes depending on the behavior of adversarial agents, including collisions front agent (Agent B) performing cut-ins to trigger the collision, rear agent (Agent A) accelerating and cutting in front of the ego, or both agents approaching the ego at the same time.

To further investigate the scenarios, all of them were manually assigned one of seven arbitrarily chosen classes, listed in Fig. 5.9. A dimensional reduction algorithm t-Distributed Stochastic Neighbor Embedding (t-SNE) [56] was then utilized to represent the scenarios' parameters in a 2-dimensional space, as shown in Fig. 5.9.

The scenarios represented on 2-dimensional space after t-SNE dimensionality reduction do not form easily recognizable clusters related to each scenario class, although two main groups can be distinguished - one containing mostly scenarios in which the B agent collides with the ego and one in which Agent A causes the collision.

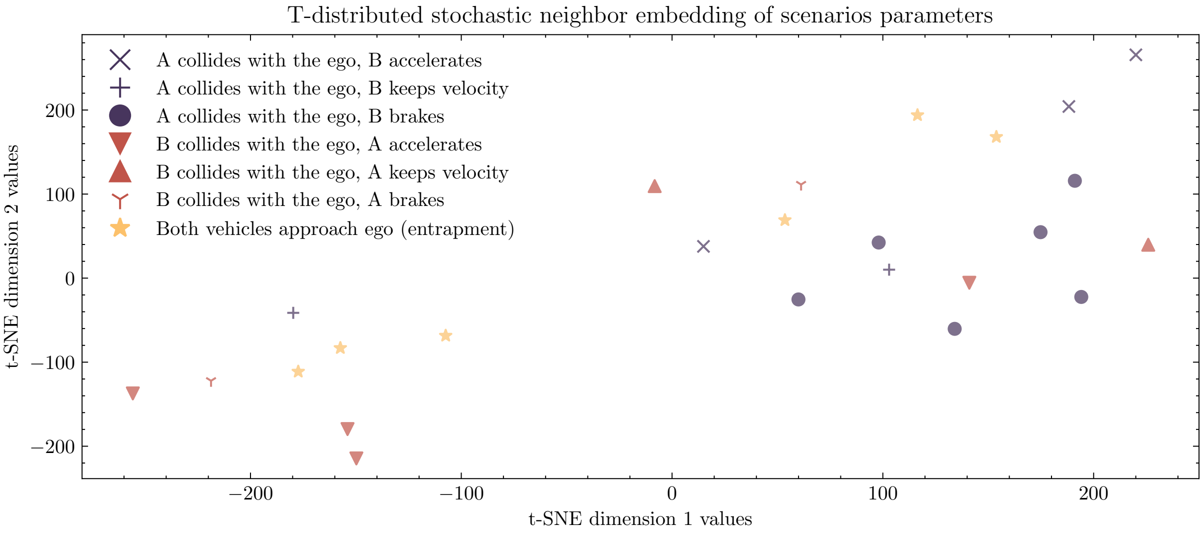


Figure 5.9. t-SNE decomposition of scenarios parameters in Experiment 1.

The lack of smaller, class-related clusters may suggest that the scenarios are relatively varied even within one class. In most cases, the scenarios in a single class differ in the behavior of one of the agents, e.g., the agent that did not participate in the collision was performing different maneuvers, producing variations of the scenario.

5.4.2.2. Experiment 2

Experiment 2 was designed to examine the ability of the method to generate scenarios with state estimate errors.

In this experiment a relatively common merge-in scenario is explored, in which the ego vehicle starts on a merge-in lane that closes in approximately 60 meters, as shown in Fig. 5.10, driving with an initial speed of $10\frac{m}{s}$. Another vehicle starts the scenario with identical speed on the right lane of a 2-lane road, to which the ego's lane merges.

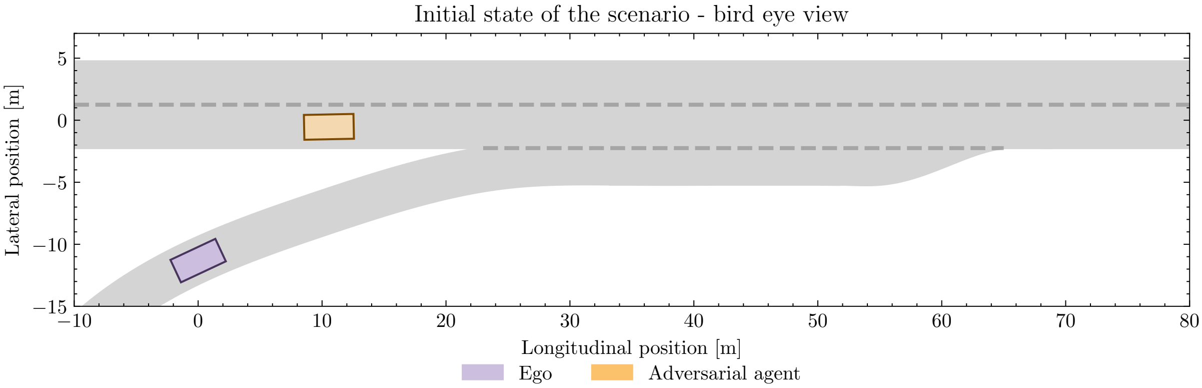

Initial state of the scenario - bird eye view
Lateral position [m]
5
0
−5
−10
−15
−10
0
10
20
30
40
50
60
70
80
Longitudinal position [m]
Ego
Adversarial agent

Figure 5.10. Initial conditions for the second experiment. Both vehicles start with an initial speed of $10\frac{m}{s}$.

Similarly, as in the previous experiment, the method is used to generate 20 adversarial scenarios, with the difference of also generating the state estimate trajectory of the other vehicle. The state estimate errors are generated only for the longitudinal and lateral positions of the vehicle, that is, other state variables remain unmodified. The maximum duration of the scenario is set to 9 seconds.

5.4.2.2.1 Example 1

In the first example, presented in Fig. 5.11, the adversarial agent applied moderate braking in the first part of the scenario, while driving near the center of the lane. However, the generated state estimate trajectory involved a moderate lateral error that started from approximately $-1.5m$, and grew to $1m$ over the duration of the scenario. The error, while relatively insignificant, could lead to a false interpretation of the vehicle's motion, suggesting an ongoing execution of a lane-change maneuver.

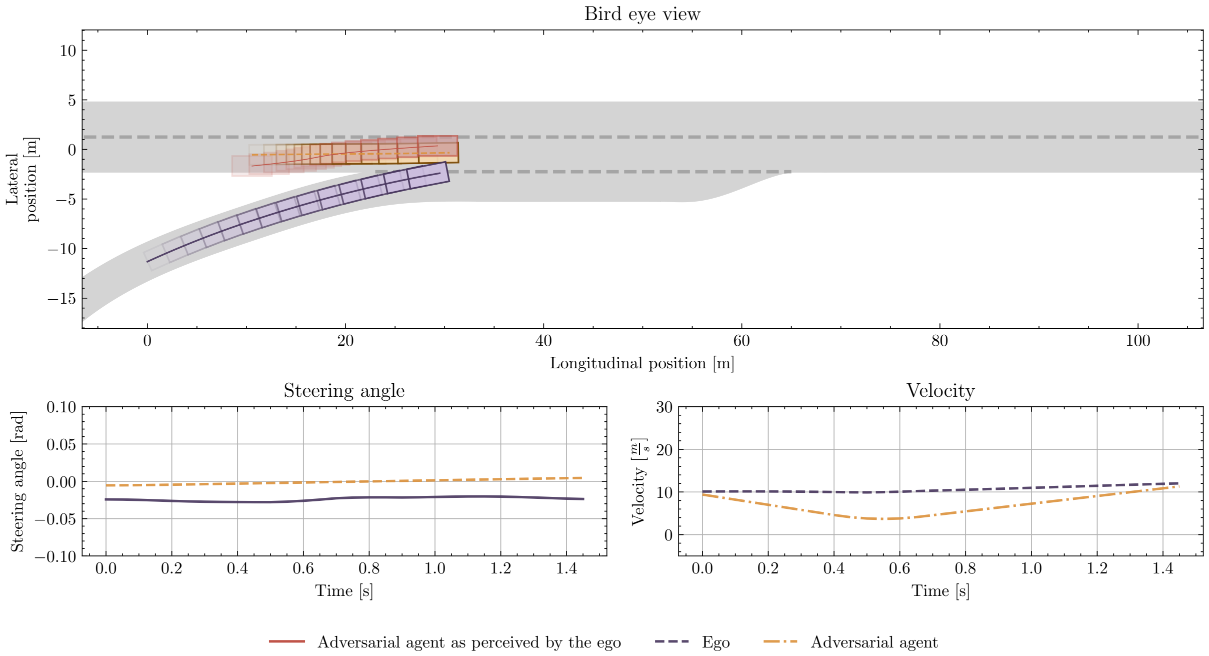


Figure 5.11. Experiment 2, scenario example 1: the false assumption of lane change due to perception errors.

As shown in Fig. 5.11, the driving policy under test in fact reacted incorrectly to the observed state estimates, performing a lane change maneuver as soon as possible due to the geometry of the road, leading to a collision with the adversarial agent.

5.4.2.2.2 Example 2

The second example represents a fairly similar situation, in which the other vehicle performs a braking maneuver on its lane, but with a different perception error pattern produced. As can be seen in Fig. 5.12, the position of the vehicle is perceived by the ego as closer to the merge-in lane than in the actual scenario, potentially leading to an incorrect assumption that the vehicle is performing a lane change to the right.

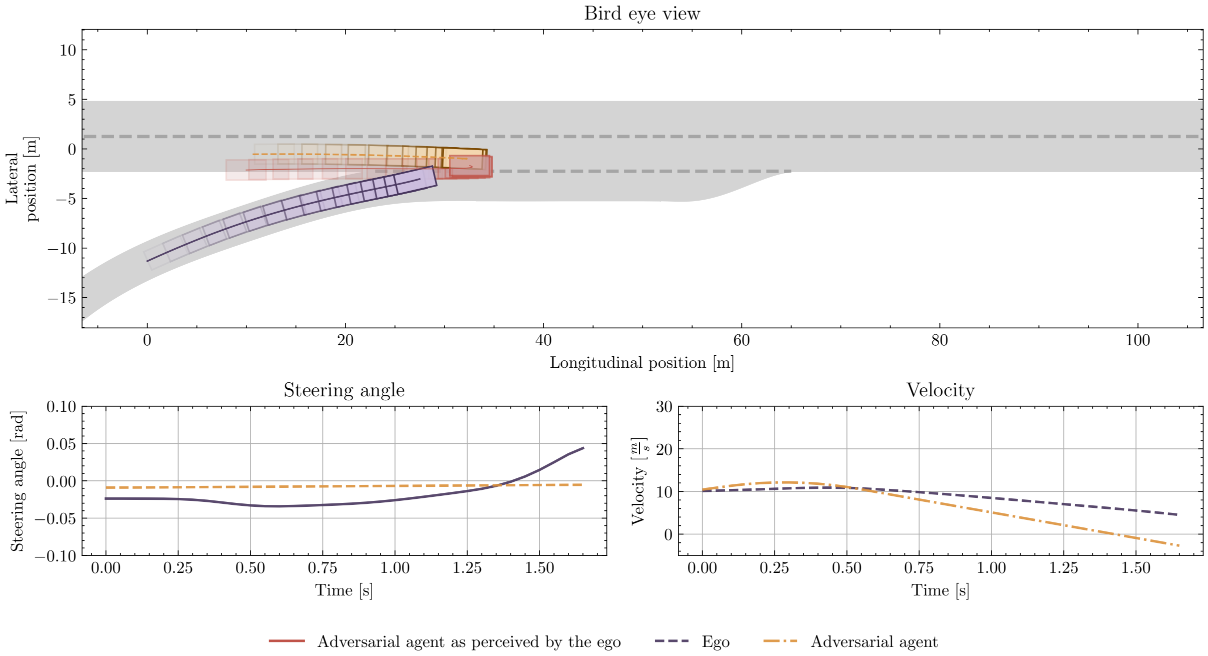


Figure 5.12. Experiment 2, scenario example 2.

The ego vehicle reacts to this observation in an improper manner, by performing a sudden steering maneuver to the left, which directly leads to a collision with the other vehicle. Such behavior could be caused by a false assumption that the other vehicle will perform a right-lane change maneuver, in which case a left-lane change maneuver could potentially constitute a viable collision avoidance maneuver.

5.4.2.2.3 Example 3

The third example exposes another issue in the tested system, not directly related to the merge-in maneuver. In this scenario, the adversarial agent brakes at the beginning of the scenario, allowing the ego vehicle to safely merge into the traffic on a main road. The ego vehicle performed a double lane change, followed by centering on the left lane. The adversarial agent subsequently accelerated, performing a slow lane change.

The state estimation errors in this case were again relatively small, as the lateral position estimation error was maintained at approximately $-1m$, while the longitudinal state estimation remained almost accurate, with near-zero errors.

The tested policy failed to react to the lane change maneuver of the other vehicle in any way, resulting in the collision. While the precise reason behind such behavior remains unknown, one possible explanation is that the lateral state estimation error could lead to a false assumption that the lane-change maneuver was conducted slower than in the actual scenario. As the adversarial

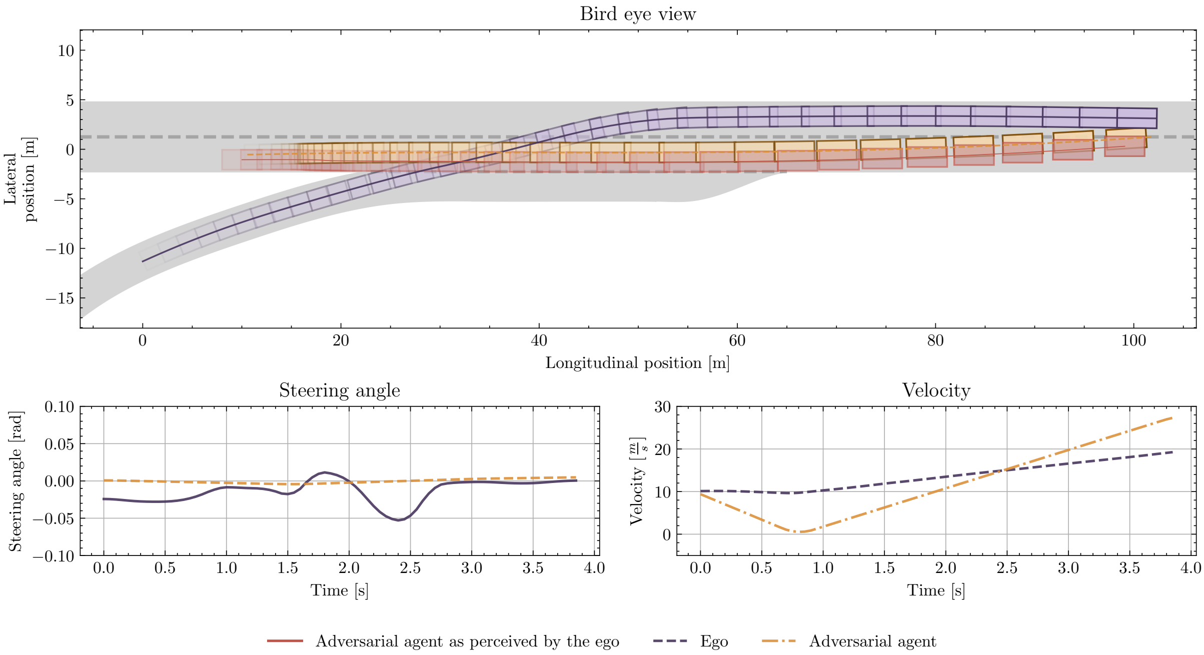


Figure 5.13. Experiment 2, scenario example 3

vehicle's speed was significantly higher than the ego's during the collision, even a minimal delay in the lane change execution could result in a safe resolution of the analyzed situation.

This exposes an important issue in the tested policy, as similar state estimation errors could occur as a consequence of delays or incorrectly calibrated filtering algorithms in the perception system.

5.5. Conclusions

In this chapter, I presented a novel adversarial scenario generation method intended for the testing of ADAS / AD systems. The method can be used to generate test scenarios based on provided or randomly generated initial conditions. The scenario generation process utilizes stochastic optimization methods that are used to pursue scenarios that pose the greatest challenge to the system under test.

The optimization problem used to generate trajectories of the traffic participants surrounding the ego vehicle is formulated in a way that penalizes the generation of scenarios similar to other scenarios within a certain scenario database. This formulation enables the use of the proposed method for the iterative generation of multiple dissimilar scenarios, allowing for the exploration of different failure modes of the tested driving policy.

One of the key features of the proposed method that distinguishes it from the existing approaches is the ability to simultaneously generate the state trajectories of the traffic participants, as well as state estimates trajectories intended to mimic the way in which the ego vehicle may perceive other vehicles. The difference between state trajectories and generated state estimates is minimized during the scenario generation, enabling the exploration of minimal perception system mistakes that may lead to critical safety failures in certain situations.

The use of the presented approach for the validation of an AD system has been demonstrated in the task of generating sets of adversarial test scenarios for a driving policy based on Reinforcement Learning techniques. The algorithm that implements the proposed method was able to successfully generate a wide variety of test scenarios that exposed several failure modes in the tested policy. Additionally, the described method has been used to explore the policy's susceptibility to perception system deficiencies, generating a set of scenarios with state estimation errors.

With a growing demand for AD systems, the proposed method can play an important role in ensuring the safety of the underlying planning and control algorithms, allowing to actively explore potential issues in the developed systems.

The method is particularly useful in applications that involve machine learning-based techniques, such as driving policies based on Reinforcement Learning. The proposed method can be used to quickly identify situations in which the developed driving policy struggles to achieve the desired performance. The scenarios generated by the proposed method can be utilized by the developers as guidance for the creation of the training sets, or even used directly in policy training, e.g., in Falsification-Based Robust Reinforcement Learning [192] setups.

5.5.1. Limitations and Further Work

The described method, while functional and applicable for testing purposes in the described form, can be further improved in several ways, which could significantly extend its applications and increase its versatility.

5.5.1.1. Initial conditions and static environment generation

All the examples presented in this section were based on manually defined initial conditions. While workflow in which the user of the proposed method defines the initial conditions is useful for the exploration of policy's vulnerabilities in situations that are known to occur frequently or be challenging, the generation of large scenario databases this way may prove to be a tedious task. For this reason, one of the important improvement areas of the proposed method is the automatic generation of initial conditions.

One way to generate initial conditions automatically is to generate the static environment in a randomized manner, e.g., randomly choosing the presence of various road features (merge-in lanes, intersections) and their parameters (number of lanes, road curvature, etc.) and then place the traffic participants on drivable parts of the road using various semi-random heuristics (e.g., generate traffic participants on a center of a random lane in a normally distributed distance from the lane beginning). Most commercially available traffic simulation packages already offer such random generation capabilities out of the box [14].

Another approach to the generation of initial conditions is to utilize situations registered during Real-World User Profiling (RWUP) test drives [103], or naturalistic trajectories studies [18]. Such an approach can be used to source both static environment information (either in the form of map data or lane markers detections registered by the test vehicle), and realistic states of other vehicles. Pairs of static and dynamic environment descriptions can be obtained at random time instances in the source logs, or various heuristics can be defined to find potentially interesting configurations, as proposed, for example, in [84].

5.5.1.2. Plausibility of the generated scenarios

Scenarios generated by the proposed method vary significantly in terms of the behavior of traffic participants, often consisting of atypical or illogical maneuvers of other vehicles. It is often useful to investigate the behavior of the tested system in such situations, as they may happen, e.g., in situations where the driver of the other vehicle is intoxicated or incapacitated. Nevertheless, a large subset of such scenarios is unlikely to happen in real traffic (e.g., situations that could be interpreted as active and conscious efforts to cause an accident), and investigation of such scenarios may be counterproductive.

While one of the main ideas behind adversarial trajectory generation is to explore scenarios that are rarely observed in real traffic, finding plausible situations that trigger critical safety failures in the tested system is often more desirable than finding scenarios that do so through complex, implausible series of adverse, illogical maneuvers.

Furthermore, as mentioned in Section 3.3.2, ensuring a complete lack of collisions on the road is practically impossible, and attempts to do so are suggested to be counterproductive by many researchers [72, 63, 157]. For this reason, it could be beneficial to focus the adversarial scenario search on situations in which the responsibility for a collision can be assigned to the driving policy's decisions, or which could be reasonably foreseen and avoided through its actions.

The examples described in a previous section already partially utilize this approach, as the cost values assigned for front and rear collisions are significantly different in the optimization problem used for trajectory generation. In particular, collisions in which the other traffic participant collides with the ego from the back are discouraged by assigning a positive cost value to them, since in such situations the responsibility for the collision is often considered to be on the rear-vehicle side [157, 63].

Depending on the priorities of the method's user, as well as requirements for the validation efforts in which the method is used, this approach can be further extended to first and foremost produce scenarios that are most plausible, or in which the ego is responsible for the accident. Estimation of the trajectories' plausibility, as well as responsibility assignment, remains, however, a complex topic, and further work would be needed to apply it in the proposed approach.

As a base for advanced responsibility assignment, the IEEE 2846-2022 standard [63], as well as frameworks such as the Responsibility-Sensitive Safety [157], or Safety Force Field [128] could be used. Knowing which vehicle in an adversarial scenario is responsible for the accident allows us to adjust the cost value in a way that promotes the generation of scenarios, in which the ego vehicle causes the accident.

To generate more plausible scenarios, various heuristics can be used to grade scenarios' plausibility, e.g., during the generation process, scenarios in which adversarial agents perform sharp, sudden maneuvers may be assigned positive cost values. Various machine learning-based techniques can also be used for this purpose, e.g., autoencoder-based neural networks can be trained in a supervised manner on naturalistic trajectory databases to recognize plausible trajectories in setups similar to autoencoder-based anomaly detection and trajectory prediction techniques [36].

The choice of a particular scenario plausibility enhancement method depends heavily on the end-user needs. While the current approach may be sufficient for use by engineers seeking to explore potential deficiencies in a developed policy, use in large-scale validation or verification endeavors or automated policy performance evaluation may require further work on the adaptation of one of the proposed methods.

5.5.2. Further Applications

One of the most evident applications of the proposed method is in the workflow, where an engineer developing the driving policy explores the vulnerabilities of the algorithm developed using this approach and utilizes the insights gained this way to improve the policy, possibly in an iterative workflow.

Depending on the type of driving policy, this process can be automated, which is especially apparent in the case of reinforcement learning-based driving policies. Generated adversarial scenarios can be used directly in the training process, in which the trained policy will explore alternative behaviors in a given situation and will be able to learn how to react in such situations. The generation of adversarial scenarios can be performed alongside the training, in a setup where new scenarios are generated for each new iteration of the driving policy. Similar approaches have already been shown to significantly improve the robustness of the trained policy to difficult situations [192], and novel features of the proposed approach could help to further immunize the developed policy against perception errors.

6. Conclusions

Recent advancements in Machine Learning (ML) technology have revolutionized many fields of science and technology, including the automotive industry. In particular, Reinforcement Learning (RL) methods have been demonstrated to achieve unprecedented performance in motion planning tasks, being able to take into account complex interactions between road users, plan long-term driving strategies, and compensate for the imperfect performance of perception systems.

Unfortunately, there are several challenges related to the introduction of such systems, including limited transparency, lack of safety guarantees, and susceptibility to errors in situations that were not sufficiently covered by the simulated scenarios used in the RL training. Furthermore, ensuring the sufficient performance of the motion planning module is a challenging task, as issues in ML-based systems may be triggered by seemingly innocuous road situations and/or small perception errors.

The research hypotheses investigated in this thesis are related to these problems, aiming to enable the introduction of RL-based motion planning methods to commercial Advanced Driving Assistance Systems (ADAS) and Autonomous Driving (AD) systems.

6.1. Summary and Contributions

In this section, I summarize the research presented in each of the chapters in the context of the research hypotheses formulated in Section 1.4.

6.1.1. Multiple Hypothesis Planning

Research presented in Chapter 3 has been motivated by a need for a trajectory generation method that could produce safe control trajectories considering multiple hypotheses regarding the future state of the dynamic environment surrounding the controlled car. Especially in the context of RL-based behavior planning methods that decide on high-level maneuvers to be performed, such trajectory planning method could be used to execute said maneuvers in a safe manner, additionally taking into account a reasonably foreseeable worst-case scenario hypothesis, effectively adding additional safety guarantees to the system.

The described considerations led to the formulation of the first research hypothesis that *it is possible to create a safe driving plan for an automated vehicle that considers several hypotheses regarding the future state of the vehicle's surroundings. In particular, reasonably foreseeable worst-case assumptions regarding the behavior of other road users can be taken into account in the motion planning algorithm, ensuring the existence of feasible collision avoidance maneuvers during the execution of the motion plan.*

To investigate this hypothesis, I proposed a novel trajectory generation method that can be used to generate multiple control trajectories that result in a collision-free movement of the ego vehicle according to their respective hypotheses. Generated trajectories overlap in a certain initial time period, allowing formulation of a control scheme in which the planning is repeated periodically, ensuring a collision-free driving plan as long as at least one of the hypotheses is accurate at any time.

The main contribution presented in this chapter is the formulation of the optimization problem used for trajectory generation that enables the simultaneous generation of all the trajectories. Thanks to this formulation, each of the planned trajectories takes into account the need for the potential execution of another, potentially less plausible, trajectory. This formulation distinguishes the proposed approach from the existing methods which either plan only a single trajectory or perform the planning for several hypotheses in a sequential manner.

Further contribution presented in this chapter is a proposal of a foreseeable worst-case hypothesis generation based on the recently introduced IEEE Standard for Assumptions in Safety-Related Models for Automated Driving System (IEEE 2846-2022) [63]. While the proposed method can be used in conjunction with various multimodal trajectory prediction modules, the proposed worst-case hypothesis generation method allows its use for a fail-safe trajectory planning task, ensuring the existence of a collision-free emergency maneuver even in reasonably foreseeable worst-case scenarios.

The effectiveness of the proposed method has been demonstrated in several numerical experiments. The method has been used to generate control trajectories for a simulated ego vehicle in several typical but challenging road situations, presenting its application with predefined multiple hypotheses regarding plausible trajectories of other road users, as well as in a fail-safe planning setup, where a worst-case hypothesis was generated by a proposed algorithm. The method has been able to generate safe (with respect to formulated hypotheses) and efficient trajectories in the presented examples, confirming the research hypothesis.

6.1.2. Sensors Modeling

Motion planning systems in autonomous vehicles rely on perception systems that provide a model of the dynamic environment (including the state of other road users), as well as the static environment (including the geometry of the road and static obstacles). Due to the limitations of the existing sensors and perception algorithms, the accuracy of such systems remains limited, and they may produce erroneous models of the environment.

Motion planning systems are often susceptible to perception errors, that, depending on their type, severity, and duration, may lead to safety-critical mistakes. Due to the complexity of the underlying physical phenomena, it is difficult to predict or model such perception errors in an accurate manner.

An advantage of RL-based motion planning systems is their generalization ability, which may potentially help to enable efficient operation even in the presence of perception errors. Unfortunately, since training such systems is typically performed in a simulation, perception errors that were not observed in the training environment may hinder their performance and result in safety-critical errors.

These observations lead to the formulation of the second research hypothesis that *the use of stochastic models of perception systems in the training process of a Reinforcement-Learning driving policy improves the policy's robustness to perception errors.*

To investigate this research hypothesis, I proposed a set of stochastic sensor models for modeling errors in both static and dynamic environment perception systems. While many sensor modeling methods were previously proposed for evaluation purposes, the impact of their use in the training of RL-based motion planning systems on the final performance of such systems remains poorly understood. For this reason, the proposal of models designed for use in RL training, as well as subsequent investigation of the models' use on the systems' performance constitutes a main contribution presented in this chapter.

The models described in Chapter 4 efficiently simulate state estimation errors that occur in radar-based and camera-based vehicle detection systems, as well as false positive and false negative detection errors. Additionally, a model of the road markers detection system was proposed, enabling the simulation of lane markers geometry estimation errors as well as false-negative detection errors. Proposed models offer performance that exceeds real-time execution needs, enabling their efficient use in large-scale training of RL-based driving policies.

To investigate the impact of the proposed models on the RL-based motion planning systems, several driving policies were trained in setups with the proposed error models, without them, and with simple baseline models. The performance of all the trained policies has been evaluated both in large-scale driving tests in the simulation, as well as in a set of predefined test scenarios.

Performed experiments have shown that the policy trained with the proposed sensor models achieved significantly better performance compared to the policies trained without the sensor models, or with the simpler baseline sensor models, confirming the proposed research hypothesis.

6.1.3. Adversarial Scenarios Generation

Testing and evaluation of motion planning systems is a challenging and expensive task. Exploring potential problems in the developed system through large-scale driving tests in real traffic is not only inefficient, but may also be dangerous. Performing similar evaluation in the simulation environment, on the other hand, depends on the traffic simulator's ability to generate rare, atypical situations that may happen in the real world, which is often limited.

Although the use of predefined test scenarios is a certain alternative for such testing approaches, issues in RL-based systems are not necessarily related to the objective difficulty of road situations, making it difficult to explore issues in such systems using predefined scenarios.

Based on these observations, I formulated the third research hypothesis: *optimization-based adversarial scenario generation methods can be used in simulation-based validation of motion planning algorithms to expose potential weaknesses or issues in the evaluated systems.*

In Chapter 5 I investigated this hypothesis, proposing a novel adversarial scenario generation method. The method utilizes optimization-based trajectory generation to actively search for scenarios that are challenging to the evaluated motion planning algorithm and enables iterative generation of an arbitrary number of dissimilar challenging scenarios.

One of the main contributions that distinguishes the proposed method from the existing ones is the ability to simultaneously generate the trajectories of the road users surrounding the vehicle controlled by evaluated policy, as well as perception error patterns that may lead to safety-critical mistakes of the tested motion planning module. This ability is especially desirable for testing RL-based driving policies, as it is difficult to predict combinations of trajectories and perception errors that would be challenging for a particular system.

The method has been used for the generation of adversarial test scenarios for an exemplary RL-based driving policy, being able to expose several issues in the tested system. Iterative execution of the proposed generation scheme allowed the acquisition of datasets of dissimilar test scenarios that involve dangerous situations. The successful generation of the test scenarios confirms the investigated research hypothesis.

6.2. Perspectives and Further Work

The proposed methods open up several areas for further research related to improving the performance of the methods and the extension of their capabilities and applications.

6.2.1. Multiple Hypothesis Planning

The Multiple Hypothesis Planning method described in Chapter 3 has been demonstrated in planning applications, where the alternative hypotheses are related to the future behavior of other road users. Conflicting hypotheses may, however, also be formulated with regard to the current state of the environment. Thus, further research can be conducted to address uncertainties in the environment model related to the limited performance of perception systems using the proposed method.

Similarly as in the described fail-safe planning approach, one could formulate a worst-case hypothesis that encompasses all plausible states of the ego's environment based on perception data with known state estimation uncertainties. The use of the proposed method in such cases could help to ensure the safety of the generated trajectories in the presence of state estimation errors. The proposed method could also be used in systems with redundant perception systems (e.g., camera-based and radar-based perception modules), allowing safe trajectories to be planned, taking into account potentially conflicting environment models generated by these systems.

Further work on the Multiple Hypothesis Planning method may also include computational performance improvements. While the observed performance is sufficient to enable re-planning within the initial trajectories overlap period, the resulting behavior of the ego may be suboptimal if the state of the environment changes in a rapid and unexpected manner. Several methods can thus be used to improve computational performance, including, but not limited to, the use of the collocation-based trajectory generation method, the implementation of the proposed method in a low-level compiled programming language, the use of the automatic differentiation method to enable fast estimation of the cost gradient, and the use of the previously generated trajectories to formulate an initial guess for subsequent iterations of the trajectory generation.

6.2.2. Sensor Modeling

Further research on the sensor modeling methods presented in Chapter 4 may include their evaluation and calibration using sensor data collected in test drives and ground truth information based, e.g., on precise reference sensors.

While the use of sensor models in the training process of RL-based driving policies does not necessarily require precise replication of the sensors' characteristics, their application in simulation-based testing may definitely benefit from accurate calibration of the models. Further research is thus needed to propose a methodology for statistical analysis of pre-recorded perception data streams that would enable the identification of relevant sensor models' parameters.

Another potential research area is related to the evaluation of proposed sensor models and driving policies trained with their use. Both real-world vehicle testing and evaluation methods based on pre-recorded perception streams can be used to evaluate the safety and performance of the trained driving policies, as well as ensure that the proposed models correctly reflect the performance of modeled perception systems.

6.2.3. Adversarial Scenarios Generation

The adversarial scenarios generation method described in Chapter 5 can be used for validation and verification purposes, but also the method can be extended to enable its use for

RL-based driving policies training purposes. Automatic generation of challenging scenarios during the training in a Falsification-based Robust Adversarial Reinforcement Learning setup could significantly improve the robustness of the resulting driving policy.

The method is designed to generate relatively plausible scenarios, with cost terms in the underlying optimization problem that penalize certain less plausible types of collisions and constraints that limit control values applied by other vehicles in a scenario. Still, it is difficult to define what constitutes a plausible scenario, and the method occasionally produces scenarios that include the atypical behaviors of other road users.

As described in Section 5.5.1.2, various additional constraints and heuristics can be implemented to increase the credibility of generated scenarios, including machine learning-based methods, such as discriminative networks known from generative adversarial models (GAN).

Other possible improvements of the proposed approach include automated generation of initial conditions, as well as extending the method with static environment model generation capabilities.

Bibliography

[1] *""Phantom Auto" to Be Operated Here". The Free-Lance Star. 17 June 1932. https://news.google.com/newspapers?id=PthNAAAAIBAJ&sjid=yYoDAAAAIBAJ&pg=6442%2C3879017*. Accessed: 31-07-2023.

[2] *"Mercedes-Benz self-driving car technology approved for use"*. https://web.archive.org/web/20211209192401/ https://www.fleetnews.co.uk/news/manufacturer-news/2021/12/09/mercedes-benz-self-driving-car-technology-approved-for-use. Archived from the original on 9 December 2021. Accessed 31-07-2023. 2021.

[3] Carnegie Mellon University (1995). *"No Hands Across America" home page. https://www.cs.cmu.edu/~tjochem/nhaa/nhaa_home_page.html*. Accessed: 31-07-2023.

[4] Official Google Blog. 27 May 2014. *"Official Google Blog: Just press go: designing a self-driving vehicle". https://blog.google/alphabet/just-press-go-designing-self-driving/*. Accessed: 31-07-2023.

[5] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. "Generating adversarial driving scenarios in high-fidelity simulators". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 8271–8277. ISBN: 9781538660263. DOI: *10.1109/ICRA.2019.8793740*.

[6] Naveed Akhtar and Ajmal Mian. *Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey*. 2018. DOI: *10.1109/ACCESS.2018.2807385*.

[7] Matthias Althoff. "Reachability analysis and its application to the safety assessment of autonomous cars". PhD thesis. Technische Universität München, 2010.

[8] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. "Learning dexterous in-hand manipulation". In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[9] Aleksandar Angelov, Andrew Robertson, Roderick Murray-Smith, and Francesco Fioranelli. “Practical classification of different moving targets using automotive radar and deep neural networks”. In: *IET Radar, Sonar and Navigation* 12.10 (Oct. 2018), pp. 1082–1089. ISSN: 17518784. DOI: *10.1049/iet-rsn.2018.0103*.

[10] Szilárd Aradi. “Survey of deep reinforcement learning for motion planning of autonomous vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.2 (2020), pp. 740–759.

[11] Johannes Bach, Jacob Langner, Stefan Otten, Eric Sax, and Marc Holzapfel. “Test scenario selection for system-level verification and validation of geolocation-dependent automotive control systems”. In: *2017 International Conference on Engineering, Technology and Innovation: Engineering, Technology and Innovation Management Beyond 2020: New Challenges, New Approaches, ICE/ITMC 2017 - Proceedings*. Vol. 2018-Janua. 2018, pp. 203–210. ISBN: 9781538607749. DOI: *10.1109/ICE.2017.8279890*.

[12] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. “Emergent tool use from multi-agent autocurricula”. In: *arXiv preprint arXiv:1909.07528* (2019).

[13] Matthew Robert Justin Baldock, Alexandra Denise Long, Vicki Lee Ann Lindsay, and Jack McLean. “Rear end crashes”. In: (2005).

[14] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. “SUMO–simulation of urban mobility: an overview”. In: *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind. 2011.

[15] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. “Safe model-based reinforcement learning with stability guarantees”. In: *Advances in neural information processing systems* 30 (2017).

[16] Keshav Bimbraw. “Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology”. In: *2015 12th international conference on informatics in control, automation and robotics (ICINCO)*. Vol. 1. IEEE. 2015, pp. 191–198.

[17] Erik Bochinski, Volker Eiselein, and Thomas Sikora. “High-speed tracking-by-detection without using image information”. In: *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE. 2017, pp. 1–6.

[18] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. “The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections”. In: 2019.

[19] Mattias Brännström, Erik Coelingh, and Jonas Sjöberg. “Model-Based Threat Assessment for Avoiding Arbitrary Vehicle Collisions”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (2010), pp. 658–669. DOI: *10.1109/TITS.2010.2048314*.

[20] Silvia Bresug. “Motion Planning of Autonomous Vehicles with Safety Guarantees”. PhD thesis. Technische Universität München, 2021.

[21] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer, 2009.

[22] Yanpeng Cao, Alasdair Renfrew, and Peter Cook. “Vehicle motion analysis based on a monocular vision system”. In: (2008).

[23] Alessio Carullo, Marco Parvis, et al. “An ultrasonic sensor for distance measurement in automotive applications”. In: *IEEE Sensors journal* 1.2 (2001), p. 143.

[24] Simon Chadwick, Will Maddern, and Paul Newman. “Distant vehicle detection using radar and vision”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8311–8317.

[25] Shuo Chang, Yifan Zhang, Fan Zhang, Xiaotong Zhao, Sai Huang, Zhiyong Feng, and Zhiqing Wei. “Spatial attention fusion for obstacle detection using mmwave radar and vision sensor”. In: *Sensors* 20.4 (2020), p. 956.

[26] Long Chen, Yuchen Li, Chao Huang, Bai Li, Yang Xing, Daxin Tian, Li Li, Zhongxu Hu, Xiaoxiang Na, Zixuan Li, et al. “Milestones in autonomous driving and intelligent vehicles: Survey of surveys”. In: *IEEE Transactions on Intelligent Vehicles* 8.2 (2022), pp. 1046–1056.

[27] Simiao Chen, Michael Kuhn, Klaus Prettner, and David E Bloom. “The global macroeconomic burden of road injuries: estimates and projections for 166 countries”. In: *The Lancet Planetary Health* 3.9 (2019), e390–e398.

[28] Yen-Lin Chen, Yuan-Hsin Chen, Chao-Jung Chen, and Bing-Fei Wu. “Nighttime vehicle detection for driver assistance and autonomous vehicles”. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 1. IEEE. 2006, pp. 687–690.

[29] Seunghyuk Choi, Florian Thalmayr, Dominik Wee, and Florian Weig. “Advanced driver-assistance systems: Challenges and opportunities ahead”. In: *McKinsey & Company* (2016), pp. 1–11.

[30] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. “Multimodal trajectory predictions for autonomous driving using deep convolutional networks”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2090–2096.

[31] Michael A Cusumano. “Self-driving vehicle technology: progress and promises”. In: *Communications of the ACM* 63.10 (2020), pp. 20–22.

[32] Nachiket Deo and Mohan M Trivedi. "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms". In: *2018 IEEE intelligent vehicles symposium (IV)*. IEEE. 2018, pp. 1179–1184.

[33] Thomas A Dingus, Feng Guo, Suzie Lee, Jonathan F Antin, Miguel Perez, Mindy Buchanan-King, and Jonathan Hankey. "Driver crash risk factors and prevalence evaluation using naturalistic driving data". In: *Proceedings of the National Academy of Sciences* 113.10 (2016), pp. 2636–2641.

[34] Marek Długosz, Michał Brodzicki, Paweł Skruch, Marcin Szelest, and Dariusz Cieślar. "Intelligent road recognition system for an autonomous vehicle". In: *2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. 2022, pp. 122–128. DOI: *10.1109/ICETA57911.2022.9974839*.

[35] Anand Dubey, Avik Santra, Jonas Fuchs, Maximilian Lubke, Robert Weigel, and Fabian Lurz. "A Bayesian Framework for Integrated Deep Metric Learning and Tracking of Vulnerable Road Users Using Automotive Radars". In: *IEEE Access* 9 (2021), pp. 68758–68777. ISSN: 21693536. DOI: *10.1109/ACCESS.2021.3077690*.

[36] Albert Dulian and John C Murray. "Multi-modal anticipation of stochastic trajectories in a dynamic environment with Conditional Variational Autoencoders". In: *arXiv preprint arXiv:2103.03912* (2021).

[37] Marius Dupuis, Martin Strobl, and Hans Grezlikowski. "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks". In: *Proc. of the Driving Simulation Conference Europe*. 2010, pp. 231–242.

[38] Andreas Eidehall and Lars Petersson. "Statistical Threat Assessment for General Road Scenes Using Monte Carlo Sampling". In: *IEEE Transactions on Intelligent Transportation Systems* 9.1 (2008), pp. 137–147. DOI: *10.1109/TITS.2007.909241*.

[39] "Example Applications of IEEE Std 2846-2022 to Formal Safety-Related Models". In: *Example Applications of IEEE Std 2846-2022 to Formal Safety-Related Models* (2023), pp. 1–26.

[40] Traffic Safety Facts. "Alcohol-Impaired Driving". In: *National Highway Traffic Safety Administration Reports* (2014).

[41] Laura Fraade-Blanar, Marjory S Blumenthal, James M Anderson, and Nidhi Kalra. *Measuring Automated Vehicle Safety: Forging a Framework*. Santa Monica, CA: RAND Corporation, 2018. DOI: *10.7249/RR2662*.

[42] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[43] Shivam Gautam, Gregory P Meyer, Carlos Vallespi-Gonzalez, and Brian C Becker. "Sdv-tracker: Real-time multi-sensor association and tracking for self-driving vehicles". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3012–3021.

[44] Anthony F Genovese. "The interacting multiple model algorithm for accurate state estimation of maneuvering targets". In: *Johns Hopkins APL technical digest* 22.4 (2001), pp. 614–623.

[45] Simon Genser, Stefan Muckenhuber, Selim Solmaz, and Jakob Reckenzaun. "Development and experimental validation of an Intelligent Camera Model for Automated Driving". In: *Sensors* 21.22 (2021), p. 7583.

[46] Igor Gilitschenski, Guy Rosman, Arjun Gupta, Sertac Karaman, and Daniela Rus. "Deep Context Maps: Agent Trajectory Prediction Using Location-Specific Latent Maps". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5097–5104. DOI: *10.1109/lra.2020.3004800*.

[47] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. "A review of motion planning techniques for automated vehicles". In: *IEEE Transactions on intelligent transportation systems* 17.4 (2015), pp. 1135–1145.

[48] DH Greene, JJ Liu, JE Reich, Yukio Hirokawa, T Mikami, Hayuru Ito, and A Shinagawa. "A computationally-efficient collision early warning system for vehicles, pedestrian and bicyclists". In: *Proc. of the 15th World Congress on Intelligent Transportation Systems*. 2008.

[49] Melita Hadzagic, Hannah Michalska, and Alexandre Jouan. "IMM-JVC and IMM-JPDA for closely maneuvering targets". In: *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat. No. 01CH37256)*. Vol. 2. IEEE. 2001, pp. 1278–1282.

[50] Mordechai Haklay and Patrick Weber. "Openstreetmap: User-generated street maps". In: *IEEE Pervasive computing* 7.4 (2008), pp. 12–18.

[51] T Hanke, N Hirsenkorn, C van-Driesten, P Garcia-Ramos, M Schiementz, S Schneider, and E Biebl. "A generic interface for the environment perception of automated driving functions in virtual scenarios". In: *Internet: https://www. hot. ei. tum. de/forschung/automotive-veroeffentlichungen* (2019).

[52] T. Hanke, N. Hirsenkorn, B. Dehlink, A. Rauch, R. Rasshofer, and E. Biebl. "Generic architecture for simulation of ADAS sensors". In: *Proceedings International Radar Symposium* 2015-August (Aug. 2015), pp. 125–130. ISSN: 21555753. DOI: *10.1109/IRS.2015.7226306*.

[53] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer. 2016, pp. 630–645.

[55] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. "On pre-trained image features and synthetic images for deep learning". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[56] Geoffrey E Hinton and Sam Roweis. "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15 (2002).

[57] Nils Hirsenkorn, Paul Subkowski, Timo Hanke, Alexander Schaermann, Andreas Rauch, Ralph Rasshofer, and Erwin Biebl. "A ray launching approach for modeling an FMCW radar system". In: *2017 18th International Radar Symposium (IRS)*. 2017, pp. 1–10. DOI: *10.23919/IRS.2017.8008120*.

[58] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[59] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. "Diagnosing error in object detectors". In: *European conference on computer vision*. Springer. 2012, pp. 340–353.

[60] Jonathan Horgan, Ciarán Hughes, John McDonald, and Senthil Yogamani. "Vision-based driver assistance systems: Survey, taxonomy and advances". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE. 2015, pp. 2032–2039.

[61] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. "Adversarial attacks on neural network policies". In: *5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings*. 2019.

[62] Wu Ling Huang, Kunfeng Wang, Yisheng Lv, and Feng Hua Zhu. "Autonomous vehicles testing methods review". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2016, pp. 163–168. ISBN: 9781509018895. DOI: *10.1109/ITSC.2016.7795548*.

[63] "IEEE Standard for Assumptions in Safety-Related Models for Automated Driving Systems". In: *IEEE Std 2846-2022* (2022), pp. 1–59. DOI: *10.1109/IEEESTD.2022.9761121*.

[64] ITS IEEE VT. "Literature Review on Kinematic Properties of Road Users for Use on Safety-Related Models for Automated Driving Systems". In: *Literature Review on Kinematic Properties of Road Users for Use on Safety-Related Models for Automated Driving Systems* (2022), pp. 1–35.

[65] *ISO 26262-1:2018 Road Vehicles: Functional Safety.* ISO, 2018.

[66] *ISO/PAS 21448:2019 Road vehicles — Safety of the intended functionality.* ISO, 2019.

[67] Michał Jasiński. “A Generic Validation Scheme for real-time capable Automotive Radar Sensor Models integrated into an Autonomous Driving Simulator”. In: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR).* 2019, pp. 612–617. DOI: *10.1109/MMAR.2019.8864669*.

[68] James Jeffs. *A History of ADAS: Emergence to Essential. https://www.idtechex.com/en/research-article/a-history-of-adas-emergence-to-essential/25592*. Accessed: 31-07-2023. 2022.

[69] Vijay John and Seiichi Mita. “RVNet: Deep sensor fusion of monocular camera and radar for image-based obstacle detection in challenging environments”. In: *Image and Video Technology: 9th Pacific-Rim Symposium, PSIVT 2019, Sydney, NSW, Australia, November 18–22, 2019, Proceedings 9.* Springer. 2019, pp. 351–364.

[70] Roy Jonker and Ton Volgenant. “A shortest augmenting path algorithm for dense and sparse linear assignment problems”. In: *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR.* Springer. 1988, pp. 622–622.

[71] Nico Kaempchen, Kristian Weiss, Michael Schaefer, and Klaus CJ Dietmayer. “IMM object tracking for high dynamic driving maneuvers”. In: *IEEE Intelligent Vehicles Symposium, 2004.* IEEE. 2004, pp. 825–830.

[72] Nidhi Kalra and David G Groves. *The Enemy of Good: Estimating the Cost of Waiting for Nearly Perfect Automated Vehicles.* Santa Monica, CA: RAND Corporation, 2017. DOI: *10.7249/RR2150*.

[73] Nidhi Kalra and Susan M Paddock. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Santa Monica, CA: RAND Corporation, 2016. DOI: *10.7249/RR1478*.

[74] Nidhi Kalra and Susan M. Paddock. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Santa Monica, CA: RAND Corporation, 2016. DOI: *10.7249/RR1478*.

[75] Takeo Kanade, Chuck Thorpe, and William Whittaker. “Autonomous land vehicle project at CMU”. In: *Proceedings of the 1986 ACM fourteenth annual conference on Computer science.* 1986, pp. 71–80.

[76] Dhanoop Karunakaran, Stewart Worrall, and Eduardo Nebot. “Efficient statistical validation with edge cases to evaluate Highly Automated Vehicles”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems, ITSC 2020.* 2020. ISBN: 9781728141497. DOI: *10.1109/ITSC45102.2020.9294590*.

[77] Atsushi Kawasaki and Akihito Seki. “Multimodal trajectory predictions for autonomous driving without a detailed prior map”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 3723–3732.

[78] Dominik Kellner, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Klaus Dietmayer. “Wheel extraction based on micro doppler distribution using high-resolution radar”. In: *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, ICMIM 2015*. Institute of Electrical and Electronics Engineers Inc., June 2015. ISBN: 9781479972159. DOI: *10.1109/ICMIM.2015.7117951*.

[79] Jinhyeong Kim, Youngseok Kim, and Dongsuk Kum. “Low-level sensor fusion network for 3d vehicle detection using radar range-azimuth heatmap and monocular image”. In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[80] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[81] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. “Semi-supervised learning with deep generative models”. In: *Advances in neural information processing systems* 27 (2014).

[82] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. “Deep reinforcement learning for autonomous driving: A survey”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2021), pp. 4909–4926.

[83] Marvin Klimke, Benjamin Völz, and Michael Buchholz. *Integration of Reinforcement Learning Based Behavior Planning With Sampling Based Motion Planning for Automated Driving*. 2023. arXiv: *2304.08280* [cs.RO].

[84] Moritz Klischat, Edmond Irani Liu, Fabian Holtke, and Matthias Althoff. “Scenario factory: Creating safety-critical traffic scenarios for automated vehicles”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–7.

[85] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 1094–1099. DOI: *10.1109/IVS.2015.7225830*.

[86] Paweł Kowalczyk, Paulina Bugiel, Marcin Szelest, and Jacek Izydorczyk. “Fault injection in optical path - detection quality degradation analysis with Wasserstein distance”. In: *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2021, pp. 121–126. DOI: *10.1109/MMAR49549.2021.9528441*.

[87] Dieter Kraft. "A software package for sequential quadratic programming". In: *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[88] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2118–2125. DOI: *10.1109/ITSC.2018.8569552*.

[89] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Vol. 2018-Novem. 2018, pp. 2118–2125. ISBN: 9781728103235. DOI: *10.1109/ITSC.2018.8569552*.

[90] Florian Kraus, Nicolas Scheiner, Werner Ritter, and Klaus Dietmayer. "The Radar Ghost Dataset – An Evaluation of Ghost Objects in Automotive Radar Data". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021.

[91] Andreas Kuehnle. "Symmetry-based recognition of vehicle rears". In: *Pattern recognition letters* 12.4 (1991), pp. 249–258.

[92] Anders Kullgren, Anders Lie, and Claes Tingvall. "Comparison Between Euro NCAP Test Results and Real-World Crash Data". In: *Traffic Injury Prevention* 11.6 (Dec. 2010), pp. 587–593. ISSN: 15389588. DOI: *10.1080/15389588.2010.508804*.

[93] Puneet Kumar, Mathias Perrollaz, Stéphanie Lefevre, and Christian Laugier. "Learning-based approach for online lane change intention prediction". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2013, pp. 797–802.

[94] Ying-Che Kuo, Neng-Sheng Pai, and Yen-Feng Li. "Vision-based vehicle detection for a driver assistance system". In: *Computers & Mathematics with Applications* 61.8 (2011), pp. 2096–2100.

[95] Ray Lattarulo and Joshué Pérez Rastelli. "A hybrid planning approach based on MPC and parametric curves for overtaking maneuvers". In: *Sensors* 21.2 (2021), p. 595.

[96] Steven M LaValle et al. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[97] Donghan Lee, Youngwook Paul Kwon, Sara McMains, and J Karl Hedrick. "Convolution neural network-based lane change intention prediction of surrounding vehicles for ACC". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[98] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. "Desire: Distant future prediction in dynamic scenes with interacting agents". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017, pp. 336–345.

[99] Suzanne E Lee, Eddy Llaneras, Sheila Klauer, and Jeremy Sudweeks. "Analyses of rear-end crashes and near-crashes in the 100-car naturalistic driving study to support rear-signaling countermeasure development". In: *DOT HS* 810 (2007), p. 846.

[100] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH journal* 1.1 (2014), pp. 1–14.

[101] Kamil Lelowicz. "Camera model for lens with strong distortion in automotive application". In: *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR).* IEEE. 2019, pp. 314–319.

[102] Kamil Lelowicz, Michał Jasiński, and Adam Krzysztof Piłat. "Discussion of novel filters and models for color space conversion". In: *IEEE Sensors Journal* (2022).

[103] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. "Av-fuzzer: Finding safety violations in autonomous driving systems". In: *2020 IEEE 31st international symposium on software reliability engineering (ISSRE).* IEEE. 2020, pp. 25–36.

[104] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. "Kalman filter and its application". In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS).* IEEE. 2015, pp. 74–77.

[105] You Li and Javier Ibanez-Guzman. "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems". In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[106] Anders Lie and Claes Tingvall. "How do Euro NCAP results correlate with real-life injury risks? A paired comparison study of car-to-car crashes". In: *Traffic Injury Prevention* 3.4 (Dec. 2002), pp. 288–293. ISSN: 15389588. DOI: *10.1080/15389580214632*.

[107] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017, pp. 2117–2125.

[108] C. Liu, S. Liu, C. Zhang, Y. Huang, and H. Wang. "Multipath propagation analysis and ghost target removal for FMCW automotive radars". In: *IET International Radar Conference (IET IRC 2020).* Vol. 2020. IET, 2020.

[109] Rong Liu, Jinling Wang, and Bingqi Zhang. "High definition map for automated driving: Overview and analysis". In: *The Journal of Navigation* 73.2 (2020), pp. 324–341.

[110] Wansong Liu, Danyang Luo, Changxu Wu, and Minghui Zheng. “Vehicle-Human Interactive Behaviors in Emergency: Data Extraction from Traffic Accident Videos”. In: *Proceedings of the American Control Conference*. Vol. 2020-July. 2020, pp. 2526–2531. ISBN: 9781538682661. DOI: *10.23919/ACC45564.2020.9147923*.

[111] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. “Ssd: Single shot multibox detector”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[112] Oren Longman, Shahar Villeval, and Igal Bilik. “Multipath Ghost Targets Mitigation in Automotive Environments”. In: *IEEE National Radar Conference - Proceedings* 2021-May (May 2021). ISSN: 10975659. DOI: *10.1109/RADARCONF2147009.2021.9455253*.

[113] Haitong Ma, Jianyu Chen, Shengbo Eben, Ziyu Lin, Yang Guan, Yangang Ren, and Sifa Zheng. “Model-based constrained reinforcement learning using generalized control barrier function”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4552–4559.

[114] Jing Ma, Xiaobo Che, Yanqiang Li, and Edmund M.K. Lai. *Traffic scenarios for automated vehicle testing: A review of description languages and systems*. 2021. DOI: *10.3390/machines9120342*.

[115] Adrian Macaveiu and Andrei Câmpeanu. “Automotive radar target tracking by Kalman filtering”. In: *2013 11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*. Vol. 02. 2013, pp. 553–556. DOI: *10.1109/TELSKS.2013.6704439*.

[116] Silvia Magdici and Matthias Althoff. “Fail-safe motion planning of autonomous vehicles”. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, pp. 452–458.

[117] Gustav Markkula, Ola Benderius, Krister Wolff, and Mattias Wahde. “A review of near-collision driver behavior models”. In: *Human Factors* 54.6 (2012), pp. 1117–1143. ISSN: 15478181. DOI: *10.1177/0018720812448474*.

[118] Aarian Marshall. *"Uber Gives Up on the Self-Driving Dream"*. https://www.wired.com/story/uber-gives-up-self-driving-dream/. Accessed 31-07-2023. 2020.

[119] Enrique Marti, Miguel Angel De Miguel, Fernando Garcia, and Joshue Perez. “A review of sensor technologies for perception in automated driving”. In: *IEEE Intelligent Transportation Systems Magazine* 11.4 (2019), pp. 94–108.

[120] Pallavi Mitra, Apratirn Choudhury, Vimal Rau Aparow, Giridharan Kulandaivelu, and Justin Dauwels. "Towards Modeling of Perception Errors in Autonomous Vehicles". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3024–3029. DOI: *10.1109/ITSC.2018.8570015*.

[121] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review". In: *IEEE Transactions on Intelligent Transportation Systems* 23.1 (2020), pp. 33–47.

[122] Stefan Muckenhuber, Eniz Museljic, and Georg Stettinger. "Performance evaluation of a state-of-the-art automotive radar and corresponding modeling approaches based on a large labeled dataset". In: *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations* (2021). ISSN: 15472442. DOI: *10.1080/15472450.2021.1959328*.

[123] Wassim Najm, Mary Stearns, Heidi Howarth, Jonathan Koopmann, John S Hitz, et al. *Evaluation of an automotive rear-end collision avoidance system.* Tech. rep. United States. Department of Transportation. National Highway Traffic Safety ..., 2006.

[124] Demin Nalic, Hexuan Li, Arno Eichberger, Christoph Wellershaus, Aleksa Pandurevic, and Branko Rogic. "Stress Testing Method for Scenario-Based Testing of Automated Driving Systems". In: *IEEE Access* 8 (2020), pp. 224974–224984. ISSN: 21693536. DOI: *10.1109/ACCESS.2020.3044024*.

[125] Nina Narodytska and Shiva Kasiviswanathan. "Simple Black-Box Adversarial Attacks on Deep Neural Networks". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops.* Vol. 2017-July. 2017, pp. 1310–1318. ISBN: 9781538607336. DOI: *10.1109/CVPRW.2017.172*.

[126] Christian Neurohr, Lukas Westhofen, Tabea Henning, Thies De Graaff, Eike Mohlmann, and Eckard Bode. "Fundamental Considerations around Scenario-Based Testing for Automated Driving". In: *IEEE Intelligent Vehicles Symposium, Proceedings.* 2020, pp. 121–127. ISBN: 2005.04045v2. DOI: *10.1109/IV47402.2020.9304823*.

[127] Johannes Nguyen, Simon T Powers, Neil Urquhart, Thomas Farrenkopf, and Michael Guckert. "An overview of agent-based traffic simulators". In: *Transportation research interdisciplinary perspectives* 12 (2021), p. 100486.

[128] David Nistér, Hon-Leung Lee, Julia Ng, and Yizhou Wang. *The Safety Force Field.* Tech. rep.

[129] David Nistér, Hon-Leung Lee, Julia Ng, and Yizhou Wang. "The safety force field". In: *NVIDIA White Paper* (2019).

[130] Ryosuke Okuda, Yuki Kajiwara, and Kazuaki Terashima. “A survey of technical trend of ADAS and autonomous driving”. In: *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test.* IEEE. 2014, pp. 1–4.

[131] World Health Organization. *Global Status Report on Road Safety 2018: Summary.* Tech. rep. 2018.

[132] Mateusz Orłowski, Tomasz Wrona, Nikodem Pankiewicz, and Wojciech Turlej. “Safe and Goal-Based Highway Maneuver Planning with Reinforcement Learning”. In: *Advanced, Contemporary Control.* Springer, 2020, pp. 1261–1274.

[133] Vyshakh Palli-Thazha, David Filliat, and Javier Ibañez-Guzmán. “Trajectory Prediction of Traffic Agents: Incorporating context into machine learning approaches”. In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring).* 2020, pp. 1–6. DOI: *10.1109/VTC2020-Spring48590.2020.9128848*.

[134] Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. “Risk averse robust adversarial reinforcement learning”. In: *Proceedings - IEEE International Conference on Robotics and Automation.* Vol. 2019-May. 2019, pp. 8522–8528. ISBN: 9781538660263. DOI: *10.1109/ICRA.2019.8794293*.

[135] Frederik Pasch, Fabian Oboril, Bernd Gassmann, and Kay-Ulrich Scholl. “Vulnerable Road Users in Structured Environments with Responsibility-Sensitive Safety”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC).* 2021, pp. 270–277. DOI: *10.1109/ITSC48978.2021.9564554*.

[136] Sujeet Milind Patole, Murat Torlak, Dan Wang, and Murtaza Ali. “Automotive radars: A review of signal processing techniques”. In: *IEEE Signal Processing Magazine* 34.2 (2017).

[137] Margie M Peden. *World report on road traffic injury prevention.* World Health Organization, 2004.

[138] Praveena Penmetsa, Pezhman Sheinidashtegol, Aibek Musaev, Emmanuel Kofi Adanu, and Matthew Hudnall. “Effects of the autonomous vehicle crashes on public perception of the technology”. In: *IATSS research* 45.4 (2021), pp. 485–492.

[139] R. Pepy, A. Lambert, and H. Mounier. “Reducing Navigation Errors by Planning with Realistic Vehicle Model”. In: *2006 IEEE Intelligent Vehicles Symposium.* 2006, pp. 300–307. DOI: *10.1109/IVS.2006.1689645*.

[140] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. “Robust adversarial reinforcement learning”. In: *34th International Conference on Machine Learning, ICML 2017.* Vol. 6. 2017, pp. 4310–4319. ISBN: 9781510855144.

[141] Philip Polack, Florent Altche, Brigitte DAndrea-Novel, and Arnaud De La Fortelle. “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” In: *IEEE Intelligent Vehicles Symposium, Proceedings.* 2017, pp. 812–818. ISBN: 9781509048045. DOI: *10.1109/IVS.2017.7995816*.

[142] Philip Polack, Florent Altché, Brigitte d’Andréa-Novel, and Arnaud de La Fortelle. “The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?” In: *2017 IEEE intelligent vehicles symposium (IV).* IEEE. 2017, pp. 812–818.

[143] Aris Polychronopoulos, Manolis Tsogas, Angelos J Amditis, and Luisa Andreone. “Sensor fusion for predicting vehicles’ path for collision avoidance systems”. In: *IEEE Transactions on Intelligent Transportation Systems* 8.3 (2007), pp. 549–562.

[144] Louis B Rall. *Automatic differentiation: Techniques and applications.* Springer, 1981.

[145] Paul Rau and Christopher Becker. “Approach for Deriving Scenarios for Safety of the Intended Functionality”. In: *Esv* (2019), pp. 1–15.

[146] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 779–788.

[147] Gintux Market Data Report. *"Driverless Car Accident Statistics And Trends in 2023"*. https://blog.gitnux.com/driverless-car-accident-statistics/. Accessed 15-08-2023. 2023.

[148] Maximilian Schäfer, Kun Zhao, Markus Bühren, and Anton Kummert. “Context-aware scene prediction network (caspnet)”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC).* IEEE. 2022, pp. 3970–3977.

[149] Paul Schroeder, Mikelyn Meyers, Lidia Kostyniuk, et al. *National survey on distracted driving attitudes and behaviors–2012.* Tech. rep. United States. National Highway Traffic Safety Administration. Office of . . ., 2013.

[150] Robin Schubert, Eric Richter, and Gerd Wanielik. “Comparison and evaluation of advanced motion models for vehicle tracking”. In: *2008 11th international conference on information fusion.* IEEE. 2008, pp. 1–6.

[151] Robin Schubert, Eric Richter, and Gerd Wanielik. “Comparison and evaluation of advanced motion models for vehicle tracking”. In: *2008 11th International Conference on Information Fusion.* 2008, pp. 1–6.

[152] Karin Schuler, Denis Becker, and Werner Wiesbeck. “Extraction of Virtual Scattering Centers of Vehicles by Ray-Tracing Simulations”. In: *IEEE Transactions on Antennas and Propagation* 56.11 (2008), pp. 3543–3551. DOI: *10.1109/TAP.2008.2005436*.

[153] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. “High-dimensional continuous control using generalized advantage estimation”. In: *arXiv preprint arXiv:1506.02438* (2015).

[154] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[155] Matthew Schwall, Tom Daniel, Trent Victor, Francesca Favaro, and Henning Hohnhold. “Waymo public road safety performance data”. In: *arXiv preprint arXiv:2011.00038* (2020).

[156] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “On a Formal Model of Safe and Scalable Self-driving Cars”. In: (Aug. 2017).

[157] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “On a formal model of safe and scalable self-driving cars”. In: *arXiv preprint arXiv:1708.06374* (2017).

[158] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. “Safe, multi-agent, reinforcement learning for autonomous driving”. In: *arXiv preprint arXiv:1610.03295* (2016).

[159] Lei Shao, Han Wu, Chao Li, and Ji Li. “A Vehicle Recognition Model Based on Improved YOLOv5”. In: *Electronics* 12.6 (2023), p. 1323.

[160] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[161] Santokh Singh. *Critical reasons for crashes investigated in the national motor vehicle crash causation survey.* Tech. rep. 2015.

[162] Pawel Skruch, Marek Dlugosz, Wojciech Mitkowski, and Marcin Szelest. “A Model-Based Approach to Testing Software Control Systems Described by Linear Differential Equations”. In: *Proceedings of the XXI Polish Control Conference.* Springer. 2023, pp. 215–224.

[163] Pawel Skruch, Marek Dlugosz, Wojciech Mitkowski, and Marcin Szelest. “Software System Testing as an Optimization Problem”. In: *Proceedings of the XXI Polish Control Conference.* Springer. 2023, pp. 205–214.

[164] Paweł Skruch, Wojciech Mitkowski, Piotr Bania, and Marek Długosz. “Systemy dynamiczne i teoria sterowania w nowoczesnej automatyce”. In: Jan. 2022, pp. 33–41. ISBN: 978-83-67427-00-5. DOI: *10.7494/978-83-67427-00-5_2*.

[165] Paweł Skruch, Marcin Szelest, Marek Długosz, and Dariusz Cieślar. “Safety of Perception Systems in Vehicles of High-Level Motion Automation”. In: Oct. 2022. DOI: *10.1109/ICETA57911.2022.9974838*.

[166] Anthony Stentz. “Optimal and efficient path planning for partially-known environments”. In: *Proceedings of the 1994 IEEE international conference on robotics and automation.* IEEE. 1994, pp. 3310–3317.

[167] Andrea Stocco, Brian Pulfer, and Paolo Tonella. “Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems”. In: (Dec. 2021).

[168] Johan Strandroth, Matteo Rizzi, Simon Sternlund, Anders Lie, and Claes Tingvall. “The Correlation Between Pedestrian Injury Severity in Real-Life Crashes and Euro NCAP Pedestrian Test Results”. In: *Traffic Injury Prevention* 12.6 (Dec. 2011), pp. 604–613. ISSN: 15389588. DOI: *10.1080/15389588.2011.607198*.

[169] Tomasz Sulkowski, Paulina Bugiel, and Jacek Izydorczyk. “In Search of the Ultimate Autonomous Driving Simulator”. In: *2018 International Conference on Signals and Electronic Systems (ICSES).* 2018, pp. 252–256. DOI: *10.1109/ICSES.2018.8507288*.

[170] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. “A survey of end-to-end driving: Architectures and training methods”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.4 (2020), pp. 1364–1384.

[171] SAE Taxonomy. “Definitions for terms related to driving automation systems for on-road motor vehicles”. In: *SAE: Warrendale, PA, USA* 3016 (2018).

[172] Nikola Tesla. *Method of and apparatus for controlling mechanism of moving vessels or vehicles.* U.S. Patent No. 613809 (1898). 1898.

[173] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[174] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS).* IEEE. 2017, pp. 23–30.

[175] Joshua P Tobin. *Real-World Robotic Perception and Control Using Synthetic Data.* University of California, Berkeley, 2019.

[176] U.S. Department of Transportation. “Drowsy Driving - A Brief Statistical Summary”. In: *National Highway Traffic Safety Administration Reports* (2011).

[177] Apostolia Tsirikoglou, Joel Kronander, Magnus Wrenninge, and Jonas Unger. “Procedural modeling and physically based rendering for synthetic data generation in automotive applications”. In: *arXiv preprint arXiv:1710.06270* (2017).

[178] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. "Simulation-based Adversarial Test Generation for Autonomous Vehicles with Machine Learning Components". In: *IEEE Intelligent Vehicles Symposium, Proceedings*. Vol. 2018-June. Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 1555–1562. ISBN: 9781538644522. DOI: *10.1109/IVS.2018.8500421*.

[179] Wojciech Turlej. "High-Level Sensor Models for the Reinforcement Learning Driving Policy Training". In: *Electronics* 12.1 (2022), p. 71.

[180] Wojciech Turlej, Mateusz Orlowski, Tomasz Wrona, and Nikodem Pankiewicz. *Method and system for planning the motion of a vehicle*. US Patent 11,584,393. 2023.

[181] Wojciech Turlej and Nikodem Pankiewicz. "Adversarial Trajectories Generation for Automotive Applications". In: *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE. 2021, pp. 115–120.

[182] Wojciech Turlej and Nikodem Pankiewicz. "Adversarial Trajectories Generation for Automotive Applications". In: *2021 25th International Conference on Methods and Models in Automation and Robotics, MMAR 2021* (2021), pp. 115–120. DOI: *10.1109/MMAR49549.2021.9528492*.

[183] Christos Tzomakas and Werner von Seelen. *Vehicle detection in traffic scenes using shadows*. Inst. für Neuroinformatik, Ruhr-Univ., 1998.

[184] Berthold Ulmer. "Vita ii-active collision avoidance in real traffic". In: *Proceedings of the Intelligent Vehicles' 94 Symposium*. IEEE. 1994, pp. 1–6.

[185] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[186] Maria Vegega, Brian Jones, Chris Monk, et al. *Understanding the effects of distracted driving and developing strategies to reduce resulting deaths and injuries: a report to congress*. Tech. rep. United States. Office of Impaired Driving and Occupant Protection, 2013.

[187] Akifumi Wachi. "Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving". In: *IJCAI International Joint Conference on Artificial Intelligence* 2019-August (2019), pp. 6006–6012. ISSN: 10450823. DOI: *10.24963/IJCAI.2019/832*.

[188] Christian Waldschmidt, Juergen Hasch, and Wolfgang Menzel. "Automotive Radar — From First Efforts to Future Systems". In: *IEEE Journal of Microwaves* 1.1 (2021).

[189] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. "An uncertain future: Forecasting from static images using variational autoencoders". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer. 2016, pp. 835–851.

[190] F Wang, Jun Zhang, and G Lu. “Vehic YOLOv4: Optimal Speed and Accle Information Detection and Tracking System Based on YOLO”. In: *Ind. Control. Comput* 7 (2018), pp. 89–91.

[191] Jiankun Wang, Tianyi Zhang, Nachuan Ma, Zhaoting Li, Han Ma, Fei Meng, and Max Q-H Meng. “A survey of learning-based robot motion planning”. In: *IET Cyber-Systems and Robotics* 3.4 (2021), pp. 302–314.

[192] Xiao Wang, Saasha Nair, and Matthias Althoff. “Falsification-Based Robust Adversarial Reinforcement Learning”. In: (July 2020).

[193] Tim A. Wheeler, Martin Holder, Hermann Winner, and Mykel J. Kochenderfer. “Deep stochastic radar models”. In: *IEEE Intelligent Vehicles Symposium, Proceedings* (2017), pp. 47–53. DOI: *10.1109/IVS.2017.7995697*.

[194] Carsten Wiecher, Sergej Japs, Lydia Kaiser, Joel Greenyer, Roman Dumitrescu, and Carsten Wolff. “Scenarios in the loop: Integrated requirements analysis and automotive system validation”. In: *Proceedings - 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings.* Association for Computing Machinery, Inc, Oct. 2020, pp. 199–208. ISBN: 9781450381352. DOI: *10.1145/3417990.3421264*.

[195] Wei Xiao, Noushin Mehdipour, Anne Collin, Amitai Y Bin-Nun, Emilio Frazzoli, Radboud Duintjer Tebbens, and Calin Belta. “Rule-based optimal control for autonomous driving”. In: *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems.* 2021, pp. 143–154.

[196] Han Xu, Yao Ma, Hao Chen Liu, Debayan Deb, Hui Liu, Ji Liang Tang, and Anil K. Jain. *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review.* Apr. 2020. DOI: *10.1007/s11633-019-1211-x*.

[197] Yajue Yang, Jia Pan, and Weiwei Wan. “Survey of optimal motion planning”. In: *IET Cyber-Systems and Robotics* 1.1 (2019), pp. 13–19.

[198] Hairi Zamzuri, Umar Zakir Abdul Hamid, Konstantin Pushkin, Djahid Gueraiche, and Mohd Azizi Abdul Rahman. “Current collision mitigation technologies for advanced driver assistance systems–a survey”. In: *Perintis eJournal* 6.2 (2016), pp. 78–90.

[199] Shuqing Zeng. “Performance evaluation of automotive radars using carrier-phase differential GPS”. In: *IEEE Transactions on Instrumentation and Measurement* 59.10 (2010), pp. 2732–2741.

[200] FK Zhang, Feng Yang, and Ce Li. “Fast vehicle detection method based on improved YOLOv3”. In: *Computer engineering and applications* 55.02 (2019), pp. 12–20.

[201] Ding Zhao, Yaohui Guo, and Yunhan Jack Jia. "TrafficNet: An open naturalistic driving scenario library". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Vol. 2018-March. 2018, pp. 1–8. ISBN: 9781538615256. DOI: *10.1109/ITSC.2017.8317860*.

[202] Ding Zhao, Yaohui Guo, and Yunhan Jack Jia. "TrafficNet: An open naturalistic driving scenario library". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. Vol. 2018-March. Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 1–8. ISBN: 9781538615256. DOI: *10.1109/ITSC.2017.8317860*.

[203] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE. 2020, pp. 737–744.

[204] Zeyu Zhu and Huijing Zhao. "A Survey of Deep RL and IL for Autonomous Driving Policy Learning". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–23. DOI: *10.1109/TITS.2021.3134702*.

[205] Marc Rene Zofka, Florian Kuhnt, Ralf Kohlhaas, Christoph Rist, Thomas Schamm, and J. Marius Zollner. "Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems". In: *2015 18th International Conference on Information Fusion, Fusion 2015*. 2015, pp. 1422–1428. ISBN: 9780982443866.

List of Figures

1.1 SAE levels of automation. 2

2.1 Example of the ADAS/AD sensor stack and mounting positions. 11

2.2 Azimuth determination in automotive radar. 13

2.3 Radar data processing workflow. 15

2.4 Multipath errors in automotive radar. 16

2.5 Wheelspin velocity estimation error in radar. 17

2.6 Simplified camera hardware model. 18

2.7 Camera operating scheme. 19

2.8 High level fusion architecture. 23

2.9 Low level fusion architecture. 25

2.10 AD system architecture. 26

3.1 Multiple hypothesis planning algorithm - general idea. 41

3.2 Fail-safe planning - general idea. 42

3.3 Road model. 46

3.4 Driving corridor - use for maneuver planning. 47

3.5 Bicycle model of the vehicle. 49

3.6 Worst-case occupancy set generation. 51

3.7 Maneuver-based occupancy set. 53

3.8 Single-lane fail safe planning example. 54

3.9 Lane change example. 57

3.10 Trajectories generated for the highway fail-safe planning scenario. 66

3.11 Trajectories generated for the highway fail-safe planning scenario with two other vehicles. 66

3.12 Urban intersection scenario overview. 67

3.13 Trajectory generation results for the urban intersection scenario. 69

3.14 Constraints generation example. 70

3.15 Merge-in scenario results. 71

4.1 Direct control network architecture. 97
4.2 Simulation environment 100
4.3 Training progress. 107
4.4 Performance of the trained policies in OU-SM environment. 109
4.5 Performance of the trained policies in different test scenarios 110

5.1 Adversarial trajectory generation - general idea. 121
5.2 Kinematic model of the vehicle. 127
5.3 General idea of adversarial scenario generation with state estimation errors. . . . 131
5.4 Successful collision avoidance maneuver executed by the tested policy. 133
5.5 Initial conditions for adversarial scenario generation Experiment 1. 135
5.6 Adversarial trajectory generation: Experiment 1, Example 1. 135
5.7 Adversarial trajectory generation: Experiment 1, Example 2. 136
5.8 Adversarial trajectory generation: Experiment 1, Example 3. 137
5.9 t-SNE decomposition of scenarios parameters in Experiment 1. 138
5.10 Initial conditions in the second adversarial scenario generation experiment. 138
5.11 Adversarial trajectory generation: Experiment 2, Example 1. 139
5.12 Adversarial trajectory generation: Experiment 2, Example 2. 140
5.13 Adversarial trajectory generation: Experiment 2, Example 3 141

List of Tables

2.1 Comparison of common types of automotive sensors. 12

3.1 Vehicle parameters. 64

3.2 Calibration parameters used in the urban intersection scenario 65

3.3 Calibration parameters used in the urban intersection scenario 68

3.4 Calibration parameters used in the merge-in scenario. 71

4.1 Values of sensor models calibration parameters used in the experimental setup. . 90

4.2 Calibration values used for lane markers perception model. 95

4.3 Calibration parameters related to the direct control network. 98

4.4 Values of reward components weights for the direct control driving policy training. 99

4.5 Proximal Policy Optimization training hyperparameters. 100

4.6 Parameters of baseline sensor models (G-SM). 103

4.7 Scenario A parameters distributions. 104

4.8 Scenario B parameters distributions. 105

4.9 Scenario C parameters distributions. 105

4.10 Scenario D parameters distributions. 105

4.11 Scenario E parameters distributions. 105

4.12 Scenario F parameters distributions. 106

4.13 Highway driving key performance indicators. 108

4.14 Single timestep computation time of different components (in seconds). 111

5.1 Weights and parameters used for generation of adversarial scenarios. 134