

AUTONOMOUS AGGRESSIVE DRIVING: THEORY & EXPERIMENTS

A Dissertation
Presented to
The Academic Faculty

By

Changxi You

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace

Georgia Institute of Technology

May 2019

Copyright © Changxi You 2019

AUTONOMOUS AGGRESSIVE DRIVING: THEORY & EXPERIMENTS

Approved by:

Dr. Panagiotis Tsiotras, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Karen Feigh
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Eric Marie J Feron
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Byron Boots
College of Computing
Georgia Institute of Technology

Date Approved: January 22, 2019

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Panagiotis Tsiotras for his continuous support of my PhD study on the interesting research, for his great patience, immense knowledge and professional instructions. I also would like to thank the labmates in DCSL for the helpful discussions, comments and fun in the last four years. I obtain more than what I could imagine from a PhD program.

I am also grateful to the members of my committee: Dr. Karen Feigh, Dr. Samuel Coogan, Dr. Byron Boots and Dr. Eric Marie J Feron, for their patience, insightful comments and encouragement throughout writing this thesis.

Besides my advisor, I would like to thank Dr. Jianbo Lu and Ford Motor Company for their long term support, who provided me precious experimental data and a great opportunity to join them as an intern. I also would like to thank Justin Zheng, Luis Pimentel, Brian Goldfain, Kamil Saigol, Kelsey Hawkins and Grady Williams for their great assistance in the Auto-Rally experiments. Without their support it would not be possible to conduct this research.

Last but not the least, I would like to thank my family for supporting me spiritually throughout my PhD program.

TABLE OF CONTENTS

List of Tables	xi
List of Figures	xiii
Chapter 1: Introduction	1
1.1 Motivation	1
1.2 Literature Review	1
1.2.1 Driver Modeling	2
1.2.2 Autonomous Vehicles	3
1.2.3 Planning for Autonomous Driving	5
1.3 Goals and Challenges	9
1.4 Contributions	11
1.5 Outline of the Dissertation	12
Chapter 2: Kalman Filters	13
2.1 Introduction	13
2.2 Nonlinear Kalman Filter	13

2.3	Adaptive Limited Memory UKF	15
2.4	Nonlinear State Constraints	20
2.5	Conclusion	21
Chapter 3: Driver Modeling and Parameter Estimation		22
3.1	Introduction	22
3.2	System Modeling and Problem Formulation	23
3.2.1	Driver Model	24
3.2.2	Road and Perception Model	24
3.2.3	Problem Formulation	26
3.3	Field Tests	28
3.4	Data Analysis and Results	30
3.4.1	GPS Data Processing	30
3.4.2	Driver Parameter Identification	31
3.4.3	Driver Model Refinement	34
3.5	Driver Comparison and Analysis	36
3.5.1	Driver Parameter Analysis	37
3.5.2	Wavelet Analysis of Driver Steering Torque Command	39
3.6	Conclusion	43

Chapter 4: Vehicle Modeling and Parameter Estimation	44
4.1 INTRODUCTION	44
4.2 VEHICLE MODELING	44
4.2.1 Single-Track Model	44
4.2.2 Double-Track Model	45
4.2.3 Full Vehicle Model	46
4.2.4 Tire Force Model	47
4.3 Parameter Estimation	49
4.4 Results and Discussion	50
4.4.1 Standard UKF	50
4.4.2 Adaptive Limited Memory UKF	51
4.4.3 Experiments	53
4.5 Conclusion	54
Chapter 5: Highway Traffic Modeling and Optimal Decision Making	56
5.1 Introduction	56
5.2 Traffic Modeling	57
5.2.1 Markov Decision Process	58
5.2.2 System Modeling	58
5.2.3 Dynamic Cell	61

5.3	Reinforcement Learning	63
5.3.1	Reinforcement Learning Algorithms	63
5.3.2	Reward Function	64
5.3.3	Q-Learning	65
5.4	Maximum Entropy Principle	68
5.4.1	Maximum Entropy Principle	69
5.4.2	Nonparameterized Features	70
5.4.3	Parameterized Features	72
5.5	Inverse Reinforcement Learning	73
5.5.1	Reward Approximator	73
5.5.2	MaxEnt Deep IRL Algorithm	74
5.5.3	IRL Algorithm Refinement	76
5.6	Results and Analysis	79
5.6.1	Driving Behavior from Reinforcement Learning	79
5.6.2	Driving Behavior from Inverse Reinforcement Learning	83
5.7	Conclusion	85
	Chapter 6: Path Planning and Control: Highway Overtaking	87
6.1	Introduction	87
6.2	Path Planning	88

6.2.1	Preliminaries	88
6.2.2	Joint Quadratic Bézier curves	88
6.2.3	Fourth Order Bézier Curves	91
6.3	Speed Control	94
6.4	Lane-Switching Control	97
6.4.1	Optimal Driver Model	97
6.4.2	Output Regulation	101
6.5	Results and Analysis	102
6.5.1	Optimal driver parameters	103
6.5.2	Path Planning	105
6.5.3	Path Tracking Control	107
6.5.4	Overtaking Behavior	108
6.6	Conclusion	109
Chapter 7: Path Planning and Control: Off-Road Rally Racing		110
7.1	Introduction	110
7.2	High-Speed Cornering Trajectories	111
7.2.1	Problem Formulation	112
7.2.2	Optimal Trajectories	113
7.3	Trajectory Learning	114

7.3.1	Generative Model	114
7.3.2	Primitive High-Speed Cornering Trajectory	118
7.4	Differentially Flatness Trajectory Generation	120
7.4.1	Differential Flatness	120
7.4.2	Differential Flatness of Vehicle Model	121
7.5	High-Speed Cornering Trajectory Planning	122
7.5.1	Sliding Trajectory	123
7.5.2	Guiding Trajectory	124
7.6	Control Design	130
7.6.1	Tracking Controller	131
7.6.2	Sliding Controller	133
7.6.3	Exiting Controller	134
7.7	Numerical Simulations	134
7.7.1	Trajectory Design	135
7.7.2	Tracking Control	136
7.7.3	Late-Apex High-Speed Cornering	138
7.8	Experimental Validation	140
7.8.1	CarSim Simulation	140
7.8.2	Auto-Rally Experiments	141

7.9 Conclusion	145
Chapter 8: Conclusions & Future Research Directions	147
8.1 Conclusions	147
8.2 Future Work	148
References	167
Vita	168

LIST OF TABLES

1.1	Available path generating methods.	8
2.1	The EKF procedures.	14
2.2	The UT procedures.	15
2.3	The UKF procedures.	16
2.4	The ALM-UKF procedures.	18
3.1	Steering handling course (constant velocity); CW=clockwise, CCW=counter clockwise.	29
3.2	Constant parameters of the system.	32
3.3	Driver model parameters; JEKF=Joint EKF, DEKF=Dual EKF, UB=upper bound, LB=lower bound.	33
3.4	Driver model parameters.	37
4.1	Known / Unknown Vehicle model parameters.	50
5.1	The selected features and the weights for reinforcement learning.	81
5.2	IRL results summary.	84
6.1	Constant parameters of the vehicle.	103

6.2	Driver model parameters.	104
6.3	H ₂ -norms regarding to various drivers.	105
6.4	Vehicle model parameters.	107
7.1	Initial conditions.	113
7.2	Boundary conditions	124
7.3	Equilibrium for steady-state cornering.	135
7.4	Boundary conditions	135
7.5	Road geometry and high-speed cornering trajectory setup.	138
7.6	“Late apex” specification.	139
7.7	Boundary conditions	140
7.8	Vehicle/tire model parameters.	142

LIST OF FIGURES

1.1	A two point visual driver model of steering: the far-field visual point (white cross) and the near-field visual point (white dot) in different driving scenarios [24].	3
1.2	Five levels of autonomous driving defined by the Society of Automobile Engineers [33].	4
1.3	Autonomous control architecture at different levels.	5
1.4	Path planning using Rapidly-exploring Random Trees (RRT) in a merge test. Red paths are unsafe since they may cause a collision with the traffic vehicle [48].	6
1.5	Path planning based on joint cubic Bézier curves [63].	7
1.6	Typical high-speed cornering trajectory for rally racing. Three typical features of such trajectories may be observed: large sideslip angle, counter steering and “late apex”.	10
3.1	Human-vehicle-road closed-loop system.	23
3.2	Road geometries, vehicle states and driver’s visual perception.	25
3.3	The proving ground by the google map.	28
3.4	Experiment vehicles and some apparatus. 1st row: Fiesta (left), MKS (medium), F150 (right); 2nd row: power source (left), power converter (medium), CAN case (right).	28
3.5	Illustration of the CAN network on MKS.	29

3.6	Illustration on the different coordinate systems.	31
3.7	The data, the training curve and the simulated curve for the steering wheel torque.	32
3.8	The data, the training curve and the simulated curve from the Joint/Dual E-/UKE.	34
3.9	The time histories of the driver parameters during the training process. Steady state is reached after 45 seconds.	34
3.10	The data, the training curve and the simulated curve from the Joint UKE.	35
3.11	The trajectory of the driver parameters with $\pm 2\sigma$ error during the training process.	36
3.12	Detail of estimate of ℓ_s along with the 2σ confidence bounds.	36
3.13	The bode plots of G_c for the novice, the experienced and the racing drivers, 45 mph.	37
3.14	The bode plots of G_{fb} for the novice, the experienced and the racing drivers (45 mph).	38
3.15	The plots of K_a vs K_c for the three types of drivers.	39
3.16	The steering wheel torque of the racing, experienced and novice driver (MKS, 45 mph).	40
3.17	Wavelet transform of T_{dr} of the racing (above), experienced (medium) and novice (below) driver.	40
3.18	The absolute CWT coefficients $ Wf(s, \tau) $	41
3.19	The histogram of the Lipschitz exponents α for the experienced and racing driver.	42
3.20	The histogram of the Lipschitz exponents α for the novice and experienced driver.	42

4.1	Single-track vehicle model.	45
4.2	Double-track vehicle model.	46
4.3	Riding model.	47
4.4	Rolling and pitching model.	48
4.5	The magic formula.	48
4.6	The test track and the Auto-Rally vehicle model.	49
4.7	State estimation for the single-track model using JUKF.	51
4.8	Simulation results of the estimated vehicle models using standard UKF. .	52
4.9	Convergence of the vehicle parameters along with the estimation process.	52
4.10	Simulation results of the estimated vehicle models using ALM-JUKF. . . .	53
4.11	MPPI implementation using a neural-network model (left) and a single-track model (right).	54
4.12	Sideslip angle of Auto-Rally.	54
5.1	The agent-environment interaction.	57
5.2	The traffic on multi-lane road.	59
5.3	The cells and the definition of the state: ① 9-cell internal-lane state, ② 6-cell left-boundary state and ③ 6-cell right-boundary state	59
5.4	Overtaking during cornering.	60
5.5	State transition process.	61
5.6	Dynamic cells.	62

5.7	Deep neural-network feature function and reward.	71
5.8	Structures of the deep neural-network reward functions.	74
5.9	The convergence performance of the policy π in the learning process. . .	81
5.10	Overtaking scenarios in simulation by implementing π_1^*	82
5.11	The initial setup for simulation.	83
5.12	The tailgating in simulation by implementing $\hat{\pi}_2$	85
6.1	Path planning for the single lane change.	89
6.2	Path planning for the single lane change.	90
6.3	A symmetric fourth order Bézier curve.	93
6.4	Bézier curve reconstruction for smooth transition at endpoints.	94
6.5	Slip ratio circle.	96
6.6	Two-point visual steering control driver model.	98
6.7	Path tracking error.	101
6.8	Road and vehicle used in Carsim.	102
6.9	Trajectory of H_2 vs. the number of iterations.	103
6.10	Comparison of tracking errors for all four drivers.	104
6.11	Quadratic Bézier curves for lane switching.	105
6.12	The curvature of the quadratic Bézier curves.	105
6.13	Fourth order Bézier curves for lane switching.	106

6.14	The curvature of the fourth order Bézier curves.	106
6.15	Tracking control for fourth order Bézier curves.	107
6.16	Overtaking scenarios in simulation by implementing π_1^*	108
7.1	Road geometry.	113
7.2	Optimal trajectories for different initial positions and velocities.	114
7.3	Graphical observation model with time indexing τ_j^k	116
7.4	Multiple demonstrations and the learned primitive trajectory.	118
7.5	The velocity, side-slip and yaw motion of the learned primitive.	119
7.6	Road geometry and high-speed cornering trajectory.	123
7.7	Path planning for guiding control.	125
7.8	Scheme of flatness-based vehicle dynamics control.	132
7.9	The desired and simulated trajectories.	135
7.10	The desired and simulated controls.	136
7.11	The desired and simulated output.	137
7.12	The desired and simulated lateral tire forces.	137
7.13	Switching-mode control for steady-state cornering.	138
7.14	Trail braking maneuver generation for different road geometries.	139
7.15	Closed track in CarSim.	140
7.16	Simulated trajectories.	141

7.17	The test track and the Auto-Rally vehicle platform.	142
7.18	The estimated lateral tire force.	142
7.19	The trajectories of Auto-Rally (counter clockwise).	143
7.20	The speed profile of Auto-Rally.	143
7.21	Online path replanning.	144
7.22	Online speed profile generating.	144
7.23	The posture of Auto-Rally in a typical round.	145

SUMMARY

This dissertation intends to understand expert driving maneuvers in different scenarios such as highway overtaking and off-road rally racing, which are referred to as “aggressive” driving in the context of this work. By mimicking expert driving styles, one expects to be able to improve the vehicle’s active safety and traffic efficiency in the development of autonomous vehicles. This dissertation starts from the system modeling, namely, driver modeling, vehicle modeling and traffic system modeling, for which we implement different Kalman type filters for nonlinear parameter estimation using experimental data. We then focus on the optimal decision making, path planning and control design problems for highway overtaking and off-road autonomous rally racing, respectively.

The main contributions of this dissertation can be summarized as follows. 1) The Kalman type filters require good knowledge of the system noise, which makes it challenging to design the hyperparameters for a Kalman filter. In order to estimate the unknown model parameters from real-world driving data, we develop a new adaptive limited memory UKF (ALM-UKF) for nonlinear parameter estimation. The ALM-UKF is much easier to use since it is able to estimate the unknown noise statistics on-the-fly. 2) By estimating the driver parameters in the two-point visual driver model using experimental data, we are the first to show that the driver parameters are slightly changing with time, as far as the authors know. 3) We perform a wavelet analysis on the steering commands of different drivers and show that the steering command of an experienced driver is smoother than a novice driver, according to the number of singularities of the steering command and the corresponding Lipschitz exponents. 4) Based on driver modeling and the result of driver behavior analysis, we design a robust ADAS controller for lane-keeping using output regulation to assist all drivers having the parameters entering certain region in the parameter space. 5) We propose to use a stochastic MDP for highway traffic modeling. The new concept of “dynamic cell” is introduced to dynamically extract the essential state of the traffic according to different vehicle velocities, driver intents (signals) and sizes of the surrounding vehicles (i.e., truck, sedan, etc.). This allows us to solve the (inverse) reinforcement learning problem efficiently since the dimensionality of the state space can be maintained in a manageable level. This approach is easily scalable. 6) We propose new path planning algorithms using Bézier curves to generate everywhere C^2 continuous curvature-constrained paths for highway real-time lane-switching. We demonstrate expert overtaking maneuver by implementing the proposed decision making, path planning and control algorithms on an in-house developed traffic simulator. 7) Based on the trajectory learning result, we model high-speed cornering with a segment of steady-state cornering. We then propose a geometry-based path/nominal planning algorithm using the vehicle’s differential flatness. Our approach requires low computation effort since it avoids solving optimal control problems on-the-fly, while guaranteeing good racing performance in terms of the highest speed the vehicle achieved in off-road racing.

CHAPTER 1

INTRODUCTION

1.1 Motivation

More than six million motor vehicle crashes occurred in the US in 2015 alone, of which 27 percent resulted in injury or death [1]. From 2014 to 2015 the total number of vehicle crashes increased by 3.8 percent, and the number of fatal crashes increased by 7 percent [2]. Another study, sponsored by NHTSA, investigated 723 crashes and showed that driver behavioral error caused or contributed to 99 percent of these crashes [3].

Given the increased sophistication of automotive active safety systems, these studies show that driver behavior still remains the most important factor contributing to accidents. It is therefore necessary to understand, characterize and, if possible, predict driver behavior so as to design better, and more proactive (as opposed to merely reactive) advanced driver-assist systems (ADAS). Nevertheless, driver modeling is a difficult task since driver behavior is affected by different individual factors, such as gender, age, experience and driver's aggression. Such diverse driver behaviors have a significant effect on the performance of ADAS [4, 5].

Self-driving vehicles offer a solution to avoid driver behavioral error, by completely freeing the human from the burden of driving. Hence, instead of developing an ADAS controller based on the understanding of driver behavior characteristics, one can also mimic the driving behavior of an expert driver for optimal decision making and path planning in order to generate the desired driving style for autonomous vehicles. Autonomous vehicles are expected to significantly improve traffic congestion, reduce collisions and resulting injuries, enhance mobility for the children, the elderly and the disabled, and reduce the need for parking space in cities [6]. They represent a major trend in future intelligent transportation systems.

This dissertation reviews different driver modeling methodologies in the literature and selects the well-known two-point visual driver model to characterize the steering behavior of the driver. We then build the driver model, vehicle model and traffic system model. We implement different Kalman filters for nonlinear system parameter estimation. Expert driving maneuvers for autonomous vehicles are generated for two typical driving scenarios, namely, highway overtaking and off-road autonomous rally racing.

1.2 Literature Review

In this section we review the driver modeling methodologies and the techniques used to develop autonomous vehicles.

1.2.1 Driver Modeling

A controller for vehicle handling stability should take into account the diverse driver skills, habits and handling behavior of different drivers, and persistently provide good “intuitive” performance. In order to characterize driver behavior, researchers have proposed different driver models based on several methodologies over the past four decades [7, 8, 9, 10, 11, 12, 13, 14, 15].

Wier and McRuer [7] used transfer functions to describe the result of the driver’s actions on the vehicle’s position error and yaw angle, and built a quasi-linear model (crossover model) to approximately describe the nonlinear steering behavior of the driver. This model uses feedback control to eliminate tracking error, but it does not take the driver’s preview behavior into consideration. MacAdam [8, 9] assumed that the driver wants to minimize a pre-defined previewed output error, and modeled the driver’s steering strategy as an optimal preview process with a time lag. Hess and Modjtahedzadeh [10, 16] introduced a control-theoretic model for the steering behavior of the driver. This model consisted of a preview component along with low- and high-frequency compensation elements. The above models successfully achieve lane-tracking using only lateral control; braking is not considered in these works. Burgett and Miller [12] designed and optimized a parameterized driver model using a multi-variable nonlinear regression approach, based on data collected from test tracks and driving simulations. This model investigated the driver’s braking strategy in order to avoid rear-end driving conflicts. Chatzikomis and Spentzas [17] proposed a path-following driver model that regulated both the steering wheel and the throttle/brake by previewing the path ahead of the vehicle. Keen and Cole [18] linearized the vehicle model at different working points and used a multi-model structure to characterize the ability of the driver to predict the future vehicle path. By using different combinations of the internal models, this model predictive controller (MPC) achieves various driver expertises in the path-following task.

The driver’s mental work has also been taken into consideration for driver modeling. In [15] Flad et al. proposed a steering-primitive optimal selection driver model by defining a set of elementary control primitives to describe the driver’s neuromuscular system, limbs and control actions. This model assumes that the driver has a mental model of the vehicle and the steering task and determines the optimal sequence of control primitives to achieve the target maneuver. Different artificial intelligence approaches have also been introduced to model the driver’s mental work and behavior. In [11] the authors evaluated the driver’s mental influence from the environment with respect to a “risk level” and proposed a driver model based on fuzzy control theory. Lin et al. [13] built a neural network driver model and compared three typical model configurations in great detail. More recently, Hamada et al. [19] proposed a beta process autoregressive hidden Markov model (HMM). This model was trained in an unsupervised way using real driving data, and was used to predict the driving behaviors of the drivers.

All previous driver control-theoretic models can be categorized into three groups according to the methodology used to develop them: 1) classical control theory such as [7, 10, 16], where the system is represented using transfer functions and the stability is analyzed using frequency-response methods; 2) modern control theory such as [8, 9, 12,

17, 18], where the system is represented in state space and the stability is analyzed in the time domain; and 3) intelligent control theory such as [11, 13, 19], where the artificial intelligence approaches including neural network, fuzzy logic and HMM are used to develop the driver models [20]. These driver models focus on three kinds of driving tasks, including longitudinal control [12], lateral control [7, 8, 9, 10, 16, 13, 18, 15] and combined longitudinal-lateral control [11, 17, 19].

Non-parameterized models such as neural networks or HMMs have also been used to predict driver behavior. They have to be trained off-line by using supervised/unsupervised machine learning techniques and they typically need large amounts of data. We prefer to use the parameterized, transfer function based driver models, such as the crossover model [7, 21], the control theoretic model [10], and the two-point visual driver model [22, 23, 24] for control design tasks, since they are quasi/- linear and transparent to the user. The two-point visual driver model (see Figure 1.1) is considered to have both satisfactory model accuracy and good identification feasibility [25].

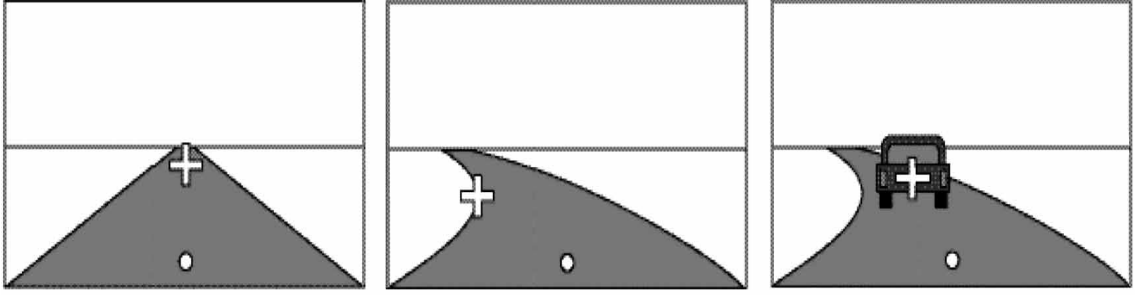


Figure 1.1: A two point visual driver model of steering: the far-field visual point (white cross) and the near-field visual point (white dot) in different driving scenarios [24].

Such parameterized, transfer function based driver models have certain number of unknown parameters requiring the researchers to design. One can either determine these parameters by solving an optimization problem, or alternatively, one can estimate these unknown parameters using real driving data. The most commonly used techniques for parameter estimation include least-squares fitting [26, 27], robust techniques [27, 28, 29], and various Kalman filtering techniques [27, 30, 31, 32].

Based on the understanding of driver's handling characteristics, one can either design an ADAS controller for better lane/path tracking performance, or develop fully autonomous vehicles to completely free the driver from the burden of driving, such that the driver behavioral error could be effectively eliminated.

1.2.2 Autonomous Vehicles

An autonomous vehicle is able to detect the environment and navigate without the driver's input, by using a variety of sensing techniques such as radar, lidar, ultrasound, localization and computer vision, along with advanced control techniques that can analyze the sensory data, in order to plan and achieve the desired path to the desired des-

tion. Autonomous vehicles are expected to significantly improve traffic congestion, reduce collisions and resulting injuries, enhance mobility for the children, the elderly and the disabled, and reduce the need for parking space in cities [6]. Figure 1.2 shows the definitions of different automated driving levels provided by the Society of Automobile Engineers (SAE). Levels 1-3 require a licensed driver to be behind the steering wheel, while levels 4 and 5 allow driverless operations.


























		 Human Driver	 Automated Systems	Steering and Acceleration/Deceleration	Monitoring of Driving Environment	Fallback When Automation Fails	Automated System is in Control	BASf level	NHTSA level
Human Driver Monitors the Driving Environment	0	NO AUTOMATION					n/a	Driver Only	0
	1	DRIVER ASSISTANCE					Some Driving Modes	Assisted	1
	2	PARTIAL AUTOMATION					Some Driving Modes	Partially Automated	2
Automated Driving System Monitors the Driving Environment	3	CONDITIONAL AUTOMATION					Some Driving Modes	Highly Automated	3
	4	HIGH AUTOMATION					Some Driving Modes	Fully Automated	3/4
	5	FULL AUTOMATION						-----	

Figure 1.2: Five levels of autonomous driving defined by the Society of Automobile Engineers [33].

Due to the rapid development of sensing and computing technologies over the past two decades, research in the field of autonomous vehicles has shown great progress, and related self-driving vehicle technology has matured significantly in the recent years [6].

The first autonomous vehicle was developed by Carnegie Mellon University's Navlab in 1988, and it was able to achieve lane-following using camera images [34]. Navlab completed the first autonomous coast-to-coast trip across the United States in 1995, traveling 2,849 miles between Pittsburgh and San Diego at an average speed of 63.8 mph[35]. Another important milestone in the self-driving vehicle technology was the DARPA Grand Challenge, which was held three times between 2004 and 2007 [36]. In these races the vehicles were required to drive autonomously in an off-road course (2004 and 2005) or an urban area course (2007) without any human intervention. These tests showed that fully autonomous off-road driving and fully autonomous urban driving are indeed technologically possible. Since then, many commercial companies, startups, and research organizations have launched their own development of autonomous vehicles.

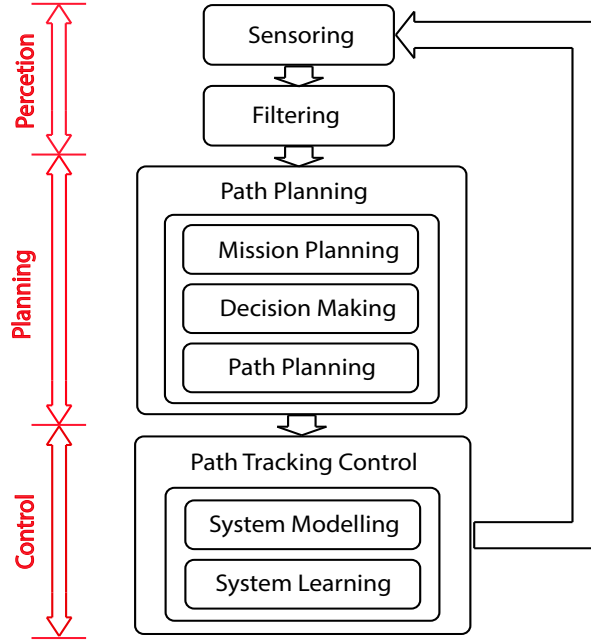


Figure 1.3: Autonomous control architecture at different levels.

Google started the self-driving car project in 2009 (called Waymo after 2016) and has already tested autonomous vehicles for more than 10 million miles in six states of the US. Waymo introduced a minivan based on a mass-production platform for the purpose of full autonomy [37]. The ride-sharing company Uber tested its first self-driving program in the mobility service sector in Pittsburgh in 2016, and plans to eventually replace all its drivers with self-driving cars in the not-so-distant future [38]. Tesla currently provides auto steering, lane changing and parking capabilities in their “Autopilot” system and plans to bring semi-autonomous and autonomous vehicle features to the mass market with the 2017 Model 3 [39]. In 2016, Ford became the first automaker to test its autonomous vehicles on snow and in darkness, and plans to deliver a commercially available fully autonomous vehicle by 2021 [40]. Volvo introduced the first large-scale autonomous drive project and plans to give 100 customers early-access to autonomous XC90 on Swedish public roads by 2017[41]. Although there were about 44 large corporations and numerous automotive driving startups working on autonomous vehicles by May 2017 [42], vehicles currently permitted on public roads are not fully autonomous and they all require a driver to take over control of the vehicle at a moment’s notice.

1.2.3 Planning for Autonomous Driving

The main technical issues in developing fully autonomous vehicles exist at three levels, namely, perception, planning and control, as shown in Figure 1.3. The perception level comprises of sensing and filtering of environmental data. The sensing system consists of a number of sensors and provides information about the vehicle’s state and the environment. The filtering system denoises the signals from the sensing system and provides a reasonable estimate for the unmeasurable states[30, 43, 44, 45]. The planning level

completes three tasks, which include mission planning, where the vehicle solves a routing problem in order to complete a task, decision making, where the vehicle chooses an appropriate action for the next time step from an available action set, and path planning, where the vehicle plans its future trajectory as a function of space or time [46, 47, 48, 49]. Finally, the control level receives the signals from the planning level, maintains the stability of the vehicle, and tracks the desired path.

Many vehicle control techniques have been developed to enhance stability and handling performance, such as differential braking [50, 51], torque vectoring [52, 53], active steering [54, 55] and integrated chassis control [56, 57] and advanced driver assist systems [58, 18, 59, 60]. Numerous control techniques are available to use at the control level. Higher-level path planning and decision making is another essential part for developing fully autonomous vehicles. During the past decade many techniques have been developed to solve the high-level planning problems for autonomous vehicles [46, 48, 61, 62, 63, 64, 65, 66, 67].

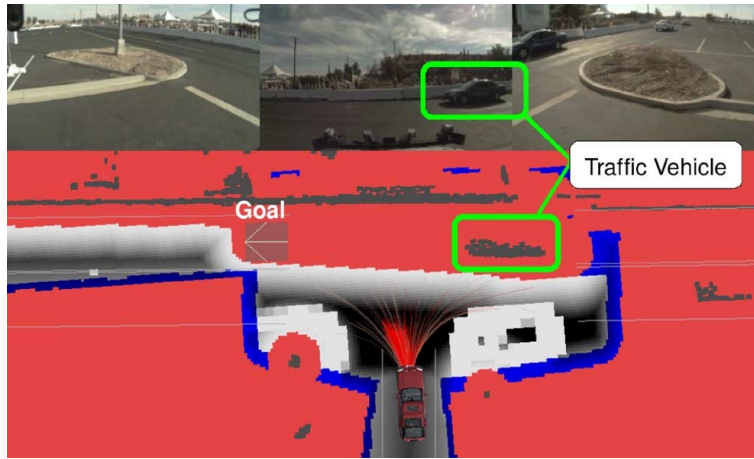


Figure 1.4: Path planning using Rapidly-exploring Random Trees (RRT) in a merge test. Red paths are unsafe since they may cause a collision with the traffic vehicle [48].

In [48] the authors proposed a real-time path planning algorithm based on Rapidly-exploring Random Trees (see Figure 1.4). This algorithm was implemented on an autonomous vehicle which completed a 60 mile simulated military supply mission in the 2007 DARPA Urban Challenge. The path planning approaches using RRTs can efficiently explore the space to handle obstacle avoidance problems. Nevertheless, since the tree is built incrementally from the direction of the samples randomly from the search space, an additional smoother may be required to smooth the path. Cimurs et al. used Dijkstra's algorithm to find the shortest viable path by connecting the Voronoi vertices, such that the path keeps a safe distance from all the obstacles in the environment [61]. They then used the Bézier curves to smooth the path with respect to the maximum curvature constraint by selecting and aligning the control points.

In order to generate a smooth path for an autonomous vehicle, Choi et al. [62, 63] presented a series of path planning algorithms based on Bézier curves. The planned paths have continuous curvature and satisfy the road boundary constraints (see Fig-

ure 1.5). Shim et al. [64] used a parameterized 6th-order polynomial to represent a smooth path, and planned a feasible path for the autonomous vehicle satisfying both the initial/final conditions and the constraint conditions. They implemented their path-planning algorithm in static/moving obstacle avoidance tasks and designed the tracking control module using model predictive control techniques. Instead of planning a path geometrically by solving an optimization problem [62, 63, 64], one can also design a path using optimal control theory. Mousavi et al. [65] applied an extended Kalman filter to predict the future trajectory of an autonomous vehicle, and used a linear time-varying model predictive control scheme to determine the optimal path and the associated optimal control. This approach was designed to achieve collision avoidance and stochastic target tracking in a dynamic environment. Similar work was devoted to developing fast path planning or decision making algorithms to achieve fully autonomous driving in real world scenarios. Ulbrich and Maurer [66] focused on real-time decision making for lane changes of an autonomous vehicle. They used a partial observable Markov decision process (POMDP) to model the decision making for lane changes, and implemented a two-step algorithm in real-time to obtain the optimal action for an autonomous vehicle in an urban driving task.

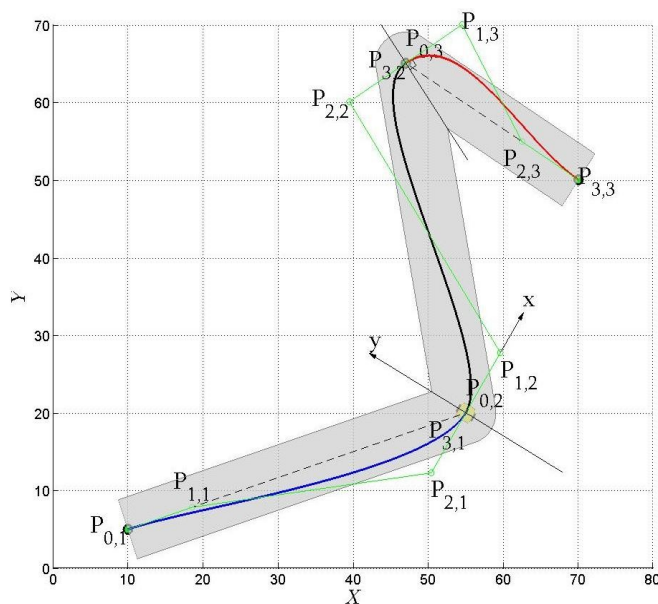


Figure 1.5: Path planning based on joint cubic Bézier curves [63].

Regarding to the single lane change maneuver, there are many path generating methods available in the literature[68, 69, 70, 71, 72]. A single lane change maneuver is required to be smooth and safe. A mathematical description of this maneuver naturally leads to two separated tasks, namely, path planning and path tracking. Traditional path planning algorithms are mainly based on sampling and searching, probability theory and geometry, among which the geometry-based methods are most practical for real-time planning. We summarize the advantages and disadvantages of some typical methods in Table 1.1. These approaches either plan a path without guaranteeing the continu-

ity or the constraint requirements on the curvature [68, 71], or require additional time for computing clothoids [69, 70] or tuning parameters [72]. A more extensive survey on path planning for autonomous vehicles can be found in [73, 36].

Table 1.1: Available path generating methods.

Method: circular trajectory[68].
Description: two constant radius arcs connected with a line segment.
Advantage: short computation time.
Disadvantage: discontinuities of the curvature.
Method: arcs combination[68].
Description: several arcs having different radii.
Advantage: smoother transition between arcs.
Disadvantage: discontinuities of the curvature.
Method: arcs and clothoids [69].
Description: constant radius arcs connected with clothoids.
Advantage: continuous curvature.
Disadvantage: More computing time.
Method: polynomial trajectory [68].
Description: the path is a 5th order polynomial trajectory.
Advantage: continuous curvature.
Disadvantage: hard to modify the shape of trajectory.
Method: Joint clothoids[70].
Description: four connected clothoids (use polynomial approximation).
Advantage: continuous curvature, lower costs than using clothoids.
Disadvantage: No obvious disadvantage.
Method: Joint Bézier curves [71, 72].
Description: two symmetrical cubic Bézier curves.
Advantage: curvature is continuous and minimized.
Disadvantage: curvature constraint is not guaranteed.

Other techniques using ideas from artificial intelligence (AI) have also been developed to solve planning problems for autonomous vehicles. These include supervised learning [74], deep learning [75] and reinforcement learning [76]. Lange et al. [67] used a deep neural encoder to extract feature representations from the raw visual input of camera images for a racing vehicle, and successfully learned the optimal control actions (i.e., steering, accelerating and braking) using reinforcement learning. The control performance was even better than an experienced human, in the sense that the car was able to move along a closed track as fast as possible without crashing. The approach in [67] concentrated on improving the driving performance of a single vehicle without considering the traffic. Shalev-Schwartz et al. [46] took the traffic into consideration and divided the planning problem into two phases. They first modeled the state transition of the traffic using a deep neural network, such that they could apply supervised learning to predict the near future states of the system. Subsequently, they used a recurrent neural network to model the trajectory and learn the optimal driving policy of

the autonomous vehicle. This approach does not rely on any Markovian assumption, and hence it is considered to be robust to the stochastic behavior of the environment. The learning procedure was validated using both an adaptive cruise control task and a roundabout merging task.

Application of reinforcement learning requires knowledge of the reward function, which needs to be carefully designed. An alternative is to learn the optimal driving strategy using demonstrations of the desired driving behaviors. Abbeel and Ng [77] used a driving simulator to collect two minutes of driving data from an expert driver, and assumed that the reward function of this expert driver is a linear combination of a number of known features. In order to recover the reward function and the expert driving policy, they proposed a max-margin algorithm along with a projection algorithm to solve the inverse reinforcement learning problem. Although one can approximately recover expert driving behaviors using this approach, the matching between the optimal policy/reward and the features is ambiguous, as indicated by Ziebart and his colleagues [78]. In order to address this ambiguity, Ziebart introduced the maximum entropy principle (MEP) to uniquely match the rewards with the features. He proposed the maximum entropy inverse reinforcement learning (MaxEnt IRL) algorithm [78, 79], which was shown to be computationally efficient in [78]. It was implemented on a routing problem (mission planning). The researchers in [80, 81, 82, 83] developed different versions of the MaxEnt IRL algorithm based on [78, 79]. Among these, the authors of [82] formulated the maximum entropy inverse reinforcement learning problem using a deep neural network (DNN) to represent the unknown reward function. All the above formulations of the MaxEnt IRL problem require complete knowledge of the environment dynamics.

1.3 Goals and Challenges

Most of the existing control driving techniques are designed to minimize, or restrict, the tire sideslip angle within the linear operation region, so that an average driver can maintain control of the vehicle during an emergency [84]. Instead of restricting the lateral dynamics of the vehicle, a better control strategy may be to take advantage of the full handling capacity of the vehicle and perform an accident avoidance maneuver. Such an approach has been studied in [85], where the authors used aggressive, but controlled, yaw motion to mitigate the effects of a T-bone collision. Good knowledge of aggressive driving maneuvers may be required in order to achieve better collision avoidance and mitigation strategies.

The present study deals with a form of aggressive driving that involves driving at high speed along with speeding/braking actions. Such maneuvers are performed primarily by expert drivers. These aggressive maneuvers may be utilized to improve vehicle safety of (semi-)autonomous vehicles and traffic efficiency, by duplicating expert driver behavior in different driving scenarios.

The authors in [86, 87, 88] suggested different definitions of aggressive driving, all of which, however, assumed that there exists at least one victim, and the driver intended to inflict psychological or physical harm to the victim or even attempts to injure or kill

the victim. A more precise definition of aggressive driving was provided by Tasca in [89], who performed a review on aggressive driving research. He concluded that, a driving behavior is aggressive if it is deliberate, likely to increase the risk of collision and is motivated by impatience, hostility and/or attempt to save time. Some typical aggressive driving behaviors on the road may include tailgating, weaving in and out of traffic, improper lane change, driving at speeds far in excess of the norm, improper passing etc. [89, 90]. These aggressive driving behaviors may violate traffic rules, and irritate the other drivers, and hence considered to be dangerous. These behaviors do not require the driver to have good driving skills, and the vehicle may not be driving at a high speed.

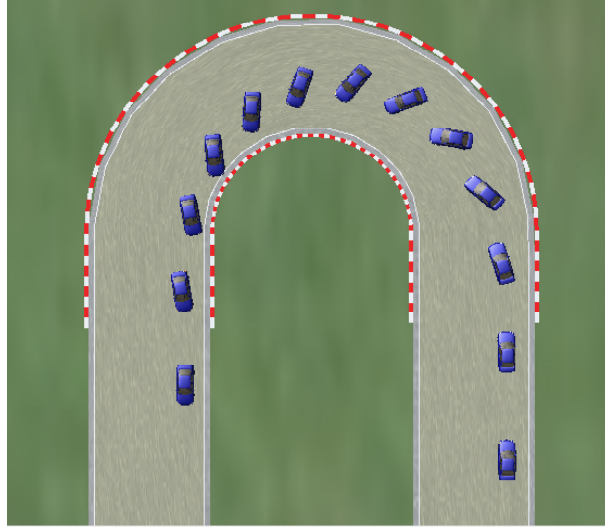


Figure 1.6: Typical high-speed cornering trajectory for rally racing. Three typical features of such trajectories may be observed: large sideslip angle, counter steering and “late apex”.

The term “aggressive driving” in this dissertation mainly represents certain high speed controlled driving maneuvers mostly performed by expert drivers, which may involve steering at high speed, along with certain speeding/braking operations possibly. We consider this kind of aggressive driving behavior since it is possible to be utilized to improve vehicle safety and traffic efficiency. This dissertation focuses on generating the aggressive driving maneuvers for autonomous vehicles in two different scenarios, namely, highway overtaking and off-road autonomous rally racing (see Figure 1.6). To this end, we need to solve high level optimal decision making problem and design low level path planning and control algorithms for the two driving tasks, respectively.

The goal of this dissertation is to solve the following problems:

- 1) Driver/vehicle modeling and parameter estimation. We use Kalman filters (i.e., EKF, UKF) for system parameter estimation. The main difficulty of this problem is the design of the unknown hyperparameters for the Kalman filter, such as the statistics of the process noise and the measurement noise.
- 2) Highway traffic modeling and optimal decision making for intelligent vehicles.

The first difficulty of this problem is the traffic modeling, where one must take into account the different velocities, different sizes (i.e., truck, sedan etc.) and different driving intents (i.e., lane-switching, braking etc.) of the surrounding vehicles during the decision making process.

Another difficulty of this problem is the design of the reward function for expert driving in order to achieve the desired driving styles. One may have to solve an inverse optimal control (inverse reinforcement learning) problem in order to recover the unknown parameters in the parameterized reward function using data.

3) Planning and control for highway overtaking. The main difficulty of this problem is the design of the path planning algorithms for lane switching, where a C^2 continuous, curvature constrained path is required to be generated in real-time. Reliable low-level controllers need to be designed for both longitudinal and lateral motion control for the autonomous vehicles.

4) Planning and control for off-road autonomous racing. The main difficulty of this problem is to generate high-speed cornering in real time without solving any optimal control problem on-the-fly. One needs to plan the nominal trajectory and design the corresponding controller using only the geometry of the path.

1.4 Contributions

The main contributions of this dissertation can be summarized as follows:

1) The design of the model-based controller for autonomous vehicles requires to model the system (i.e., driver, vehicle) with sufficient modeling accuracy. To this end, one needs to estimate the unknown model parameters from real-world driving data, using certain parameter estimation algorithms such as Kalman filters. Nevertheless, the Kalman type filters require good knowledge of the system noise, which makes it challenging to design the hyperparameters for a Kalman filter. We develop a new adaptive limited memory UKF (ALM-UKF) for nonlinear parameter estimation. The ALM-UKF is much easier to use since it is able to estimate the unknown noise statistics on-the-fly.

2) By estimating the driver parameters in the two-point visual driver model using experimental data, we are the first to show that the driver parameters are slightly changing with time, as far as the authors know. The design of ADAS controller should take this result into consideration for better control stability.

3) To better understand the steering behavior of different types of drivers, this dissertation performs a wavelet analysis on the steering commands of different drivers. The number of singularities and the corresponding Lipschitz exponents indicate that the steering command of an experienced driver is smoother than a novice driver.

4) Based on driver modeling and the result of driver behavior analysis, we design a robust ADAS controller for lane-keeping using output regulation to assist all drivers having the parameters entering certain predesigned polyhedron.

5) In order to reproduce the overtaking behavior of an expert driver in highway traf-

fic, we propose to use a stochastic MDP for highway traffic modeling. The new concept of “dynamic cell” is introduced in this dissertation to dynamically extract the essential state of the traffic according to different vehicle velocities, driver intents (signals) and the sizes of the surrounding vehicles (i.e., truck, sedan, etc.), such that we are able to solve the (inverse) reinforcement learning problem efficiently since the dimensionality of the state space can be maintained in a manageable level. This approach is easily scalable.

6) We propose new path planning algorithms using Bézier curves to generate everywhere C^2 continuous curvature-constrained paths for highway real-time lane-switching. We demonstrate expert overtaking maneuver by implementing the proposed decision making, path planning and control algorithms on an in-house developed traffic simulator.

7) Based on the trajectory learning result, we propose to model high-speed cornering with a segment of steady-state cornering. We then propose a geometry-based path/nominal planning algorithm using the vehicle’s differential flatness. Our approach requires low computation effort since it avoids solving optimal control problems on-the-fly, while guaranteeing good racing performance in terms of the highest speed the vehicle achieved in off-road racing.

1.5 Outline of the Dissertation

The contents of this dissertation are as follows. In Chapter 2 we introduce the Kalman type filters used in this work for system parameter estimation. In Chapters 3 and 4 we introduce the driver model and the vehicle models and estimate the unknown parameters using experimental data. Chapter 5 models the traffic and determines the optimal decision making strategy using (inverse) reinforcement learning techniques. Chapter 6 solves the path planning problem and designs tracking control for autonomous vehicles in highway traffic. Chapter 7 designs the controller for aggressive driving and generates real-time high-speed cornering using an auto-rally robotic platform. Finally, in Chapter 8 we summarize the main contribution of the dissertation and we propose several possible directions for future research.

CHAPTER 2

KALMAN FILTERS

2.1 Introduction

The most commonly used techniques for parameter estimation include least-squares fitting [26, 27], robust techniques [27, 28, 29], and various Kalman filtering techniques [27, 30, 31, 32]. Among these, Kalman filtering is especially suitable for problems where the measurements are collected in a sequential manner.

The extended Kalman filter (EKF) and the unscented Kalman filter (UKF) are probably the most popular filters used for system identification of nonlinear systems. Nevertheless, they can only achieve good performance under some prior knowledge including: 1) an accurate system model, 2) complete information of noise statistics, and 3) properly selected initial conditions, all of which may be either not accurate or not available in practice [91]. A commonly used approach to solve these problems is to make the Kalman filters work adaptively, by dynamically modifying the filtering algorithm using various schemes [92, 93, 94, 91, 95].

The adaptive limited memory filter (ALMF) in [95] estimates the process and observation noise statistics on-line, based on the past state estimations and observations. This algorithm improves the state estimation performance at little computational expense. Nonetheless, the ALMF was derived using a linear system model, and it has not been validated in parameter estimation applications, at least as far as the authors know. This dissertation builds on the work in [95], and develops a new adaptive limited memory unscented Kalman filter (ALM-UKF) for *nonlinear* applications.

This chapter is structured as follows. Section 2.2 introduces the standard Kalman filters for nonlinear state estimation. Section 2.3 proposes a new adaptive limited memory UKF for process and measurement noises estimation, while Section 2.4 solves the constrained state estimation problem. Finally, Section 2.5 summarizes the results of this chapter.

2.2 Nonlinear Kalman Filter

The extended Kalman filter is a classical approach to solve nonlinear estimation problems. This is achieved by means of linearizing the nonlinear state transition and nonlinear observation models. Let the discrete system

$$x_{k+1} = f(x_k, u_k, w_k), \quad (2.1a)$$

$$y_k = h(x_k, u_k, v_k), \quad (2.1b)$$

where w_k and v_k are the process noise and the measure noise, respectively, both of which are assumed to be with zero-mean white Gaussian with covariances given by

$$\mathbb{E}(w_t w_s^T) = Q_w \delta_{ts}, \quad \mathbb{E}(w_t v_s^T) = Q_c \delta_{ts}, \quad \mathbb{E}(v_t v_s^T) = Q_v \delta_{ts}, \quad (2.2)$$

where Q_w , Q_c and Q_v are the covariance matrices and δ_{ts} is the Kronecker delta function defined by

$$\delta_{ts} = \begin{cases} 1 & \text{if } t = s, \\ 0 & \text{if } t \neq s. \end{cases} \quad (2.3)$$

We assume that w_t and v_s are independent Gaussian random variables and hence the cross term Q_c in (2.2) is zero. The state estimates can then be computed using the EKF algorithm [44].

EKF Algorithm

1: Initialize with:

$$\hat{x}_0 = \mathbb{E}[x_0]$$

$$\hat{P}_0 = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$$

2: Prediction (time update):

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1})$$

$$\hat{y}_k^- = h(\hat{x}_k^-, u_k)$$

$$\hat{P}_k^- = F_{k-1} \hat{P}_{k-1} F_{k-1}^T + Q_w$$

3: Measurement update:

$$\epsilon_k = y_k - \hat{y}_k^-$$

$$S_k = H_k \hat{P}_k^- H_k^T + Q_v$$

$$K_k = \hat{P}_k^- H_k^T S_k^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \epsilon_k$$

$$\hat{P}_k = \hat{P}_k^- - K_k S_k K_k^T$$

Algorithm 2.1: The EKF procedures.

where \hat{x}_k^- is the predicted state estimate, \hat{y}_k^- is the predicted measurement, \hat{P}_k^- is the predicted estimate covariance, ϵ_k is the measurement residual, S_k is the innovation covariance, K_k is the Kalman gain, and \hat{x}_k and \hat{P}_k are the updated state estimate and the updated estimate covariance, respectively. The matrix-valued functions F_{k-1} and H_k are the Jacobian matrices of f and h in (2.1) and are given by

$$F_{k-1} = \left. \frac{\partial f}{\partial x} \right|_{\hat{x}_{k-1}}, \quad H_k = \left. \frac{\partial h}{\partial x} \right|_{\hat{x}_k^-}. \quad (2.4)$$

An alternative to EKF is to use an unscented Kalman filter (UKF). A UKF implements the unscented transform (UT) [30], and avoids calculating the Jacobian matrices at each time step. Hence, it captures the true mean and the covariance of the state Gaussian random variable to at least second order accuracy for any nonlinearity. If x is an L -

dimensional Gaussian random variable given by $x \sim (\hat{x}, P_x)$ and $g: \mathbb{R}^L \rightarrow \mathbb{R}^M$ is a nonlinear function, then the UT Algorithm calculates the statistics of $y = g(x)$ as follows:

UT Algorithm

$$\begin{aligned}
\lambda &= \alpha^2(L + \kappa) - L \\
W_0^{(m)} &= \lambda / (L + \lambda) \\
W_0^{(c)} &= \lambda / (L + \lambda) + 1 - \alpha^2 + \beta \\
W_i^{(m)} &= W_i^{(c)} = 0.5 / (L + \lambda), \quad i = 1, \dots, 2L \\
\gamma &= \sqrt{L + \lambda} \\
\mathcal{X}_0 &= \hat{x} \\
\mathcal{X}_i &= \hat{x} + (\gamma \sqrt{P_x})_i, \quad i = 1, \dots, L \\
\mathcal{X}_i &= \hat{x} - (\gamma \sqrt{P_x})_i, \quad i = L + 1, \dots, 2L \\
\mathcal{Y}_i &= g(\mathcal{X}_i), \quad i = 0, \dots, 2L \\
\hat{y} &\approx \sum_{i=1}^{2L} W_i^{(m)} \mathcal{Y}_i \\
P_y &\approx \sum_{i=1}^{2L} W_i^{(c)} (\mathcal{Y}_i - \hat{y})(\mathcal{Y}_i - \hat{y})^T
\end{aligned}$$

Algorithm 2.2: The UT procedures.

where \mathcal{X}_i , $i = 1, \dots, 2L$ are called sigma points, λ is the principal scaling parameter, α determines the spread of sigma points around the mean \hat{x} and is usually set to a small positive value, κ is a secondary scaling parameter which is usually set to zero or $3 - L$, and β is used to incorporate prior knowledge of the distribution of x . For Gaussian distribution, $\beta = 2$ is optimal [30, 31, 96, 32]. In the previous UT Algorithm $(\gamma \sqrt{P_x})_i$ is the i th column of the matrix square root.

Let us consider the system in (2.1a)-(2.1b). The UKF redefines the state vector as $x_k^a = [x_k^T, w_k^T, v_k^T]^T$ and estimates x_k^a recursively. The UT sigma point selection scheme is applied to calculate the sigma matrix \mathcal{X}_k^a for the augmented state x_k^a .

Although the UKF based algorithms (joint/dual UKF) are expected to have better accuracy, the choice between the joint estimation and the dual estimation is still not clear, since they show different performances when they are applying to different problems. More discussions can be found, for instance, in [30, 43]. The UKF equations are summarized in Algorithm 2.3.

2.3 Adaptive Limited Memory UKF

We propose a new estimation algorithm for nonlinear systems called Adaptive Limited Memory UKF (ALM-UKF). First, recall that the adaptive Kalman filter algorithm [95] adjusts the mean and the covariance of the noise on-line, which is expected to compensate for time-varying modeling errors. Define the set of unknown time-varying hyperparam-

eters for the Kalman filter corresponding to the noise statistics at the i^{th} time step, as

$$\mathcal{S}_i \triangleq \{q_i, Q_i, r_i, R_i\}. \quad (2.5)$$

\mathcal{S}_i is estimated simultaneously with the system state and parameters. Since an optimal estimator for \mathcal{S}_i does not exist, and many suboptimal schemes are either too restrictive for nonlinear applications or too computationally demanding [92, 93, 94, 91], this dissertation adopts the adaptive limited memory algorithm in [95], with the following two extensions: a) the algorithm is developed for a nonlinear application (i.e., UKF); b) we wish to estimate the unknown parameters of the system along with the state, instead of just the system state. In the following, we assume that \mathcal{S}_i is constant and is denoted by $\mathcal{S} = \{q, Q, r, R\}$.

UKF Algorithm

1: Initialize with:

$$\begin{aligned} \hat{x}_0 &= \mathbb{E}[x_0] \\ P_0 &= \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \\ \hat{x}_0^a &= \mathbb{E}[x_0^a] = [\hat{x}_0^T \ 0 \ 0]^T \\ P_0^a &= \mathbb{E}[(x_0^a - \hat{x}_0^a)(x_0^a - \hat{x}_0^a)^T] = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & Q_w & 0 \\ 0 & 0 & Q_v \end{bmatrix} \end{aligned}$$

2: Sigma-point calculation and prediction:

$$\begin{aligned} \mathcal{X}_{k-1}^a &= [\hat{x}_{k-1}^a \quad \hat{x}_{k-1}^a + \gamma \sqrt{P_{k-1}^a} \quad \hat{x}_{k-1}^a - \gamma \sqrt{P_{k-1}^a}] \\ \mathcal{X}_{k|k-1}^x &= f(\mathcal{X}_{k-1}^x, u_{k-1}, \mathcal{X}_{k-1}^w) \\ \hat{x}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x \\ P_k^- &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-)(\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-)^T \\ \mathcal{Y}_{k|k-1} &= h(\mathcal{X}_{k|k-1}^x, u_{k-1}, \mathcal{X}_{k|k-1}^v) \\ \hat{y}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1} \end{aligned}$$

3: Measurement update:

$$\begin{aligned} P_{y_k y_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-)^T \\ P_{x_k y_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-)^T \\ \mathcal{K} &= P_{x_k y_k} P_{y_k y_k}^{-1} \\ \hat{x}_k &= \hat{x}_k^- + \mathcal{K} (y_k - \hat{y}_k^-) \\ P_k &= P_k^- - \mathcal{K} P_{y_k y_k} \mathcal{K}^T \end{aligned}$$

Note: $x^a = [x^T \ w^T \ v^T]^T$, $\mathcal{X}^a = [(\mathcal{X}^x)^T \ (\mathcal{X}^w)^T \ (\mathcal{X}^v)^T]^T$.

Algorithm 2.3: The UKF procedures.

For the observation noise statistics r and R , we consider the nonlinear observation

at time k , which is given by $y_k = h(x_k, u_k) + v_k$. Since the true value of x_k is unknown, v_k is approximated by

$$r_k \approx y_k - \hat{h}(x_k, u_k), \quad (2.6)$$

where r_k represents a sample of the observation noise v at time k , and

$$\hat{h}(x_k, u_k) = \sum_{i=0}^{2L} W_i^{(m)} h(\mathcal{X}_{i,k}^x, u_k) \triangleq \hat{h}_k. \quad (2.7)$$

We define a new random variable $\xi \sim (r, C_r)$, and assume that there are N samples r_k ($k = 1, \dots, N$), such that the r_k 's are N empirical measurements for ξ . An unbiased estimator for r can be given by the sample mean

$$\hat{r} = \frac{1}{N} \sum_{k=1}^N r_k, \quad (2.8)$$

where the term “unbiased” implies that $\mathbb{E}[\hat{r}] = \mathbb{E}[\xi] = r$. An unbiased estimator for the covariance of ξ can be given by

$$\hat{C}_r = \frac{1}{N-1} \sum_{k=1}^N (r_k - \hat{r})(r_k - \hat{r})^T, \quad (2.9)$$

where the term “unbiased” implies $\mathbb{E}[\hat{C}_r] = \mathbb{E}[(\xi - r)(*)^T]$. Since $y_k = h(x_k, u_k) + v_k$, it follows from (2.6) that

$$r_k = h(x_k, u_k) - \hat{h}_k + v_k. \quad (2.10)$$

We can therefore calculate the covariance of ξ as follows

$$\begin{aligned} \mathbb{E}[(\xi - r)(*)^T] &= \frac{1}{N} \sum_{k=1}^N \mathbb{E}[(r_k - r)(*)^T] \\ &= \frac{1}{N} \sum_{k=1}^N \mathbb{E}[(h(x_k, u_k) - \hat{h}_k + v_k - r)(*)^T] \\ &= \frac{1}{N} \sum_{k=1}^N \mathbb{E}[(h(x_k, u_k) - \hat{h}_k)(*)^T] + \frac{1}{N} \sum_{k=1}^N \mathbb{E}[(v_k - r)(*)^T] + \frac{2}{N} \sum_{k=1}^N \mathbb{E}[(h(x_k, u_k) - \hat{h}_k)(v_k - r)^T] \\ &= \frac{1}{N} \sum_{k=1}^N (\mathbb{E}[(h(x_k, u_k))(*)^T] - (\hat{h}_k)(*)^T) + R. \end{aligned} \quad (2.11)$$

where

$$\mathbb{E}[(h(x_k, u_k))(*)^T] = \sum_{i=0}^{2L} W_i^{(m)} (h(\mathcal{X}_{i,k}^x, u_k))(*)^T. \quad (2.12)$$

Adaptive Limited Memory UKF

1: Initialize with:

$$\begin{aligned}
 \hat{x}_0 &= \mathbb{E}[x_0], \hat{q}_0 = \mathbb{E}[q_0], \hat{r}_0 = \mathbb{E}[r_0] \\
 P_0 &= \mathbb{E}[(x_0 - \hat{x}_0)(*)^T] \\
 Q_0 &= \mathbb{E}[(q_0 - \hat{q}_0)(*)^T] \\
 R_0 &= \mathbb{E}[(r_0 - \hat{r}_0)(*)^T] \\
 \hat{x}_0^a &= \mathbb{E}[x_0^a] = [\hat{x}_0^T \hat{q}_0^T \hat{r}_0^T]^T \\
 P_0^a &= \mathbb{E}[(x_0^a - \hat{x}_0^a)(*)^T] = \text{blkdiag}\left(\begin{bmatrix} P_0, & Q_0, & R_0 \end{bmatrix}\right)
 \end{aligned}$$

2: Sigma-point calculation and prediction:

$$\begin{aligned}
 \mathcal{X}_{k-1}^a &= [\hat{x}_{k-1}^a \quad \hat{x}_{k-1}^a + \gamma \sqrt{P_{k-1}^a} \quad \hat{x}_{k-1}^a - \gamma \sqrt{P_{k-1}^a}] \\
 \mathcal{X}_{k|k-1}^x &= f(\mathcal{X}_{k-1}^x, u_{k-1}) + \mathcal{X}_{k-1}^w \\
 \hat{x}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}_{i,k|k-1}^x \\
 P_k^- &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-)(*)^T \\
 \mathcal{Y}_{k|k-1} &= h(\mathcal{X}_{k|k-1}^x, u_{k-1}) + \mathcal{X}_{k|k-1}^v \\
 \hat{y}_k^- &= \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}
 \end{aligned}$$

3: Observation noise estimation ($k \geq N$):

$$\begin{aligned}
 r_k &= y_{k-1} - \hat{h}_{k-1} \\
 \Gamma_k &= \sum_{i=0}^{2L} W_i^{(m)} (h(\mathcal{X}_{i,k-1}^x, u_{k-1}))(*)^T - (\hat{h}_{k-1})(*)^T \\
 \hat{r}_k &= \hat{r}_{k-1} + \frac{1}{N} (r_k - r_{k-N}) \\
 R_k &= R_{k-1} + \frac{1}{N-1} \left((r_k - \hat{r}_k)(*)^T - (r_{k-N} - \hat{r}_k)(*)^T + \right. \\
 &\quad \left. \frac{1}{N} (r_k - r_{k-N})(*)^T + \frac{N-1}{N} (\Gamma_{k-N} - \Gamma_k) \right)
 \end{aligned}$$

4: Measurement update:

$$\begin{aligned}
 P_{y_k y_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-)(*)^T \\
 P_{x_k y_k} &= \sum_{i=0}^{2L} W_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{x}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{y}_k^-)^T \\
 \mathcal{K} &= P_{x_k y_k} P_{y_k y_k}^{-1} \\
 \hat{x}_k &= \hat{x}_k^- + \mathcal{K} (y_k - \hat{y}_k^-) \\
 P_k &= P_k^- - \mathcal{K} P_{y_k y_k} \mathcal{K}^T
 \end{aligned}$$

5: Process noise estimation ($k \geq M$):

$$\begin{aligned}
 q_k &= \hat{x}_k - \hat{f}_{k-1} \\
 \Pi_k &= \sum_{i=0}^{2L} W_i^{(m)} (f(\mathcal{X}_{i,k-1}^x, u_{k-1}))(*)^T - (\hat{f}_{k-1})(*)^T - P_k \\
 \hat{q}_k &= \hat{q}_{k-1} + \frac{1}{M} (q_k - q_{k-M}) \\
 Q_k &= Q_{k-1} + \frac{1}{M-1} \left((q_k - \hat{q}_k)(*)^T - (q_{k-M} - \hat{q}_k)(*)^T + \right. \\
 &\quad \left. \frac{1}{M} (q_k - q_{k-M})(*)^T + \frac{M-1}{M} (\Pi_{k-M} - \Pi_k) \right)
 \end{aligned}$$

Note: $x^a = [x^T \ w^T \ v^T]^T$, $\mathcal{X}^a = [(\mathcal{X}^x)^T \ (\mathcal{X}^w)^T \ (\mathcal{X}^v)^T]^T$.

Algorithm 2.4: The ALM-UKF procedures.

Note that we assume that x_k and v_k are independent in (2.11). By replacing $\mathbb{E}[(\xi - r)(*)^T]$ in (2.11) with the expression in (2.9), an unbiased estimate of R is given following (2.9) and (2.11):

$$\hat{R} = \frac{1}{N-1} \sum_{k=1}^N \left((r_k - \hat{r}_k)(*)^T - \frac{N-1}{N} \left(\mathbb{E}[(h(x_k, u_k))(*)^T] - (\hat{h}_k)(*)^T \right) \right). \quad (2.13)$$

For the process noise statistics q and Q , we consider the nonlinear state propagation at time k , which is given by $x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$. Since the true values of x_k and x_{k-1} are unknown, w_{k-1} is approximated by

$$q_k \approx \hat{x}_k - \hat{f}(x_{k-1}, u_{k-1}), \quad (2.14)$$

where q_k represents a sample of the process noise w at time step $k-1$, and

$$\hat{f}(x_{k-1}, u_{k-1}) = \sum_{i=0}^{2L} W_i^{(m)} f(\mathcal{X}_{i,k-1}^x, u_{k-1}) \triangleq \hat{f}_{k-1}. \quad (2.15)$$

We define a new random variable $\zeta \sim (q, C_q)$, and assume that there are M samples q_k ($k = 1, \dots, M$), where the q_k 's are M empirical measurements for ζ . An unbiased estimator for the mean value of ζ is given by the sample mean

$$\hat{q} = \frac{1}{M} \sum_{k=1}^M q_k. \quad (2.16)$$

Similarly with (2.8), the term “unbiased” indicates that $\mathbb{E}[\hat{q}] = \mathbb{E}[\zeta] = q$. An unbiased estimator for the covariance of ζ is given by

$$\hat{C}_q = \frac{1}{M-1} \sum_{k=1}^M (q_k - \hat{q})(q_k - \hat{q})^T, \quad (2.17)$$

such that $\mathbb{E}[\hat{C}_q] = \mathbb{E}[(\zeta - q)(*)^T]$, where $\mathbb{E}[(\zeta - q)(*)^T]$ is calculated by the following equation

$$\begin{aligned} \mathbb{E}[(\zeta - q)(*)^T] &= \mathbb{E}[(\zeta - q_k + q_k - q)(*)^T] \\ &= \frac{1}{M} \sum_{k=1}^M \mathbb{E} \left[\left((x_k - \hat{x}_k) - (f(x_{k-1}, u_{k-1}) - \hat{f}_{k-1}) + (q_k - q) \right) (*)^T \right] \\ &= \frac{1}{M} \sum_{k=1}^M \left(\mathbb{E}[(x_k - \hat{x}_k)(*)^T] + \mathbb{E}[(q_k - q)(*)^T] + 2\mathbb{E}[(x_k - \hat{x}_k)(q_k - q)^T] \right. \\ &\quad + \mathbb{E}[(f(x_{k-1}, u_{k-1}) - \hat{f}_{k-1})(*)^T] - 2\mathbb{E}[(x_k - \hat{x}_k)(f(x_{k-1}, u_{k-1}) - \hat{f}_{k-1})^T] \\ &\quad \left. - 2\mathbb{E}[(f(x_{k-1}, u_{k-1}) - \hat{f}_{k-1})(q_k - q)^T] \right) \\ &= \frac{1}{M} \sum_{k=1}^M \left(\mathbb{E}[(f(x_{k-1}, u_{k-1}))(*)^T] - (\hat{f}_{k-1})(*)^T - P_k \right) + Q, \end{aligned} \quad (2.18)$$

where

$$\mathbb{E}[(f(x_{k-1}, u_{k-1}))(*)^T] = \sum_{i=0}^{2L} W_i^{(m)}(f(\mathcal{X}_{i,k-1}^x, u_{k-1}))(*)^T. \quad (2.19)$$

Then Q can be estimated unbiasedly following the equations (2.17)-(2.18),

$$\hat{Q} = \frac{1}{M-1} \sum_{k=1}^M \left((q_k - \hat{q}_k)(*)^T - \frac{M-1}{M} \left(\mathbb{E}[(f(x_{k-1}, u_{k-1}))(*)^T] - (\hat{f}_{k-1})(*)^T - P_k \right) \right). \quad (2.20)$$

Equations (2.8), (2.13), (2.16) and (2.20) provide unbiased estimates for r , R , q and Q , which are based on N observation noise samples and M process noise samples, respectively. All samples r_k and q_k are assumed to be statistically independent and identically distributed. We summarize the algorithm of the Adaptive Limited Memory UKF based on equations (2.6)-(2.20), as shown in Algorithm 2.4.

2.4 Nonlinear State Constraints

The Kalman filtering constrained state estimation problem has been solved using a number of algorithms [97, 98, 99]. The available approaches for solving linear equality constraint problems include model reduction [100], perfect measurement [101], estimate projection [97], system projection [102] and soft constraints [103]. The available methods for solving nonlinear equality constraints problems include Taylor expansion approximation [104], smoothly constrained Kalman filter [105], moving horizon estimation [106], unscented Kalman filtering [107] and particle filters [108]. In this study, we use the estimate projection algorithm and the first-order Taylor expansion approximation method to solve the state estimation problem with nonlinear inequality constraints.

Geometrically, the idea is to project the unconstrained estimate $\hat{x}(k)$ onto the constraint surface. Mathematically, we solve the following minimization problem

$$\tilde{x}_k = \underset{x}{\operatorname{argmin}} \quad (x - \hat{x}_k)^T W (x - \hat{x}_k), \quad (2.21a)$$

$$\text{such that } g(x) \leq b, \quad (2.21b)$$

where \hat{x}_k and \tilde{x}_k are the unconstrained estimate and the constrained estimate of the state at the time step k , respectively, W is the weighting matrix, and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a nonlinear vector-valued function. Performing a Taylor series expansion of (2.21b) around $\hat{x}(k)$, yields

$$g(x) \approx g(\hat{x}_k) + g'(\hat{x}_k)(x - \hat{x}_k) + \dots, \quad (2.22)$$

and after ignoring higher order terms, we obtain a linear approximation of the constraint

inequalities in (2.21b),

$$g'(\hat{x}_k)x \leq b - g(\hat{x}_k) + g'(\hat{x}_k)\hat{x}_k. \quad (2.23)$$

The minimization problem (2.21a) subject to the linear inequality constraints in (2.23) can be solved using standard quadratic programming [12, 109].

2.5 Conclusion

This chapter introduces two typical nonlinear state estimators, namely, the EKF and UKF. The design of the standard EKF/UKF is hindered by the lack of knowledge of the unknown noise statistics. By tuning the noise statistics of the standard EKF/UKF, satisfactory estimates of the system state and model parameters can be obtained, but the tuning process is time consuming and hence can only be implemented off-line. We introduced an adaptive limited memory UKF algorithm (ALM-UKF), which estimates the system state and the Kalman filter hyperparameters related to the noise simultaneously, hence making possible to provide on-line estimates of the model parameters. The following work will implement the nonlinear state estimators for driver and vehicle parameters estimation.

CHAPTER 3

DRIVER MODELING AND PARAMETER ESTIMATION

3.1 Introduction

In the development of advanced driver-assist systems (ADAS) for lane-keeping or cornering, one important design objective is to appropriately share the steering control with the driver. The steering behavior of the driver must therefore be well characterized for the design of a high-performance ADAS controller. A controller for vehicle handling stability should take into account the diverse driver skills, habits and handling behavior of different drivers, and persistently provide good “intuitive” performance. In order to characterize driver behavior, researchers have proposed different driver models based on several methodologies over the past four decades [7, 8, 9, 10, 11, 12, 13, 14, 15].

The two-point visual driver model used in this chapter is derived from the concept of the two-level steering mechanism observed in a series of psychological experiments involving human drivers [110, 111, 112]. In [110] Donges divided the driver’s steering task into a guidance level and a stabilization level, and thereby built a two-level steering model. The guidance level interprets the driver’s perceptual response with respect to the oncoming road in an anticipatory open-loop control mode. The stabilization level interprets the driver’s compensatory behavior with respect to the deviation from the reference path in a closed-loop control mode. This idea has been widely accepted and has been further developed by subsequent researchers [111, 112, 24, 23, 113]. Among these researchers, Salvucci [24] first introduced the concepts of visual “near point” and “far point” into the model. By taking appropriate choices of the “near point” and “far point,” the two-point visual driver model achieves different tasks such as lane tracking [24] and collision avoidance [114].

The work of this chapter can be summarized as follows: First, we adopt the two-point visual driver model from [23], since this model characterizes driver steering behavior more precisely. This driver model combines both a two-level visual strategy and high-frequency kinesthetic feedback. The latter accounts for the interaction between the driver’s arms and the steering wheel [10]. Saleh et al in [115, 116, 22] also adopted the two-level visual strategy, but instead of the high-frequency kinesthetic feedback in [10, 23], a well-designed neuromuscular system was used. The identification of the parameters of the model in [115, 116, 22] was done using simulated data. We show the validity of the proposed model by comparing with actual recorded driver data collected during field experiments. Although previous work has validated the two-point visual driver model and identified the driver model parameters using a driving simulator [23, 115], this is the first instance that the model is validated using actual field test data. Second, by applying four different identification methods, namely, the joint EKF/UKF and the dual EKF/UKF [30, 43, 44] it is shown that the model parameters are indeed identifiable using minimal data, but that some of these parameters are not necessarily

constant but may vary with time. Our results thus reveal that *parameter-varying* versions of the two-point visual driver model may provide a much better explanation of actual human driver behavior. It is expected that these observations will pave the way for on-line driver behavior and cognitive driver state identification, which can be used downstream in the ADAS architecture in order to adapt the controller gains to the specific driver/vehicle/traffic configuration. Finally, we show that when comparing different driving types, the smoothness of the driver steering command may be a good discriminating feature for driver classification. Using wavelet signal analysis, it is shown that different driver styles correspond to different signal smoothness (i.e., degree of differentiability), as measured by the rate of decay of the wavelet coefficients. As far as we know, this is the first work that wavelet analysis has been applied to determine driver categories.

The chapter is structured as follows. Section 3.2 introduces the mathematical modeling of the driver. Section 3.3 describes the equipment and the driving scenarios used for the field tests. Section 3.4 outlines the data processing task and presents the results. Section 3.5 analyzes and compares different driver styles. Finally, Section 3.6 summarizes the results of this study and provides some directions for future work.

3.2 System Modeling and Problem Formulation

The proposed human-vehicle-road system consists of four subsystems, as shown in Figure 3.1: (a) the driver model that exerts a steering torque on the steering wheel; (b) the steering column model that converts steering torque to steering angle; (c) the vehicle model that provides the necessary position and state information of the vehicle; and (d) the road and perception model that provides the road geometry and kinematics, and also determines the driver's visual perception angles. The input to the system is the curvature of the road ρ_{ref} , which can be treated either as an external reference command to be tracked or a disturbance to be rejected, depending on the problem formulation. The primary performance variable is the lateral deviation Δy of the so-called “near point” directly in front of the vehicle to the centerline of the road (see Figure 3.1 and Figure 3.2).

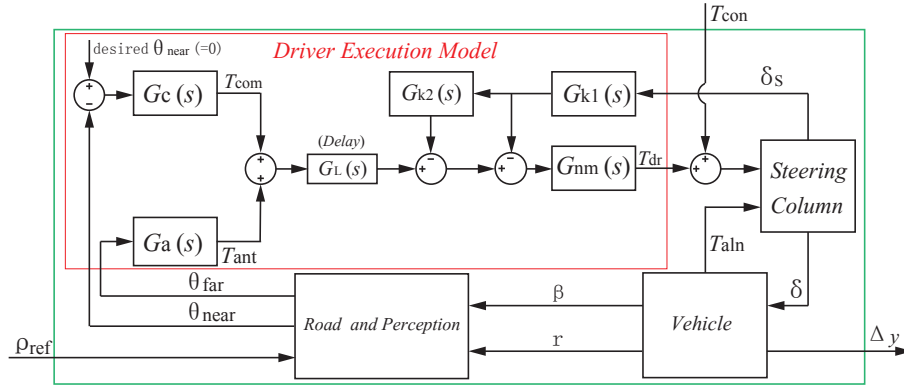


Figure 3.1: Human-vehicle-road closed-loop system.

3.2.1 Driver Model

We use the driver model proposed in [23], which introduces a kinesthetic force feedback from the steering wheel. The structure of this model is shown in the red rectangular box in Figure 3.1. The transfer functions $G_a(s)$ and $G_c(s)$ account for the anticipatory control and the compensatory control actions of the driver, respectively. The system $G_{nm}(s)$ approximately describes the neuromuscular response of the driver's arms. The “*Delay*” block indicates the driver's processing delay in the brain, and the transfer functions $G_{k1}(s)$ and $G_{k2}(s)$ account for the driver's kinesthetic perception of the steering system. The variables T_{ant} and T_{com} denote the driver's steering torques corresponding to the anticipatory control and the compensatory control paths, respectively; δ_s denotes the steering wheel angle; and the inputs θ_{near} and θ_{far} denote the near field and the far field visual angles, respectively (see Figure 3.2). Finally, T_{dr} denotes the driver's total steering torque delivered at the steering wheel. The transfer functions of the blocks shown in Figure 3.1 are given below

$$\begin{aligned} G_a(s) &= K_a, & G_c(s) &= K_c \frac{T_L s + 1}{T_I s + 1}, \\ G_{nm}(s) &= \frac{1}{T_N s + 1}, & G_{k1}(s) &= K_D \frac{T_{k1} s}{T_{k1} s + 1}, \\ G_L(s) &= e^{-t_p s}, & G_{k2}(s) &= K_G \frac{T_{k2} s + 1}{T_{k3} s + 1}, \end{aligned} \quad (3.1)$$

where K_a and K_c are static gains for the anticipatory and compensatory control subsystems, respectively; K_D and K_G are static gains for the kinesthetic perception feedback subsystems, respectively; T_L and T_I ($T_L > T_I$) are the lead time and lag time constants, respectively; T_{k1} , T_{k2} and T_{k3} are the three time constants of the driver's kinesthetic perception feedback from the steering wheel, t_p is the delay for the driver to process sensory signals, and T_N is the time constant of the driver's arm neuromuscular system. K_a , K_c , K_D , K_G , T_L , T_I , T_N , T_{k1} , T_{k2} , T_{k3} and t_p are the eleven parameters of the two-point visual driver model.

3.2.2 Road and Perception Model

The road and perception model interacts with both the vehicle model and the driver model (refer to Figure 3.1) and achieves two functions: (a) it determines the vehicle's position and posture relative to the road geometry; and (b) it determines the location of the driver's near and far visual points on the upcoming road. The near visual point is fixed at a certain distance along the heading direction of the vehicle, while the far visual point is taken as the tangent point on the inner road boundary for driving on a curved road, or the vanishing point of the road for driving along a straight road [24]. Figure 3.2 illustrates the relations between the geometry of driver's visual perception, the vehicle and the curved road[117, 118].

In Figure 3.2 the frame X_I -O- Y_I is fixed on the road. It is assumed that the vehicle is cornering with a certain lateral deviation from the road centerline. Let ψ denote the

vehicle's yaw angle, let ψ_t denote the angle between the tangent to the road centerline and the X_I axis, and let M denote the current position of the vehicle's center of mass. Let also A denote the driver's "lookahead" point in front of M at a distance ℓ_s along the vehicle's heading direction, let B denote the intersection of OA with the road centerline, let E denote the intersection of AB with the tangent to the road centerline, and let C denote the point of tangency of the line along the gaze direction on the road's inner boundary. Furthermore, let L_s denote the distance between C and M , let θ_{far} denote the visual angle between the gaze direction of the driver from a far away point and the heading direction of the vehicle, and let θ_{near} denote the near point visual angle between MB and the heading direction of the vehicle. Finally, in Figure 3.2 Δy denotes the length of the line segment AB —the predicted deviation from the road centerline at the near look-ahead point if the vehicles continues with the current heading, R_{ref} denotes the radius of the road's inner boundary, d denotes the distance from M to the road's inner boundary, and D denotes the width of the road. Henceforth, it will be assumed that d and D are small compared to R_{ref} . From Figure 3.2, the near and far distance visual perception angles can be approximated as [110, 23, 117, 119, 118, 59]

$$\theta_{\text{near}} \approx \frac{\Delta y}{\ell_s}, \quad (3.2a)$$

$$\theta_{\text{far}} \approx \frac{L_s}{R_{\text{ref}}} + \Delta\psi \approx L_s \rho_{\text{ref}} + \Delta\psi, \quad (3.2b)$$

where $\rho_{\text{ref}} = 1/R_{\text{ref}}$ is the road curvature, and $\Delta\psi = \psi_t - \psi$ is the angle between the tangent of the road centerline and the vehicle's heading direction.

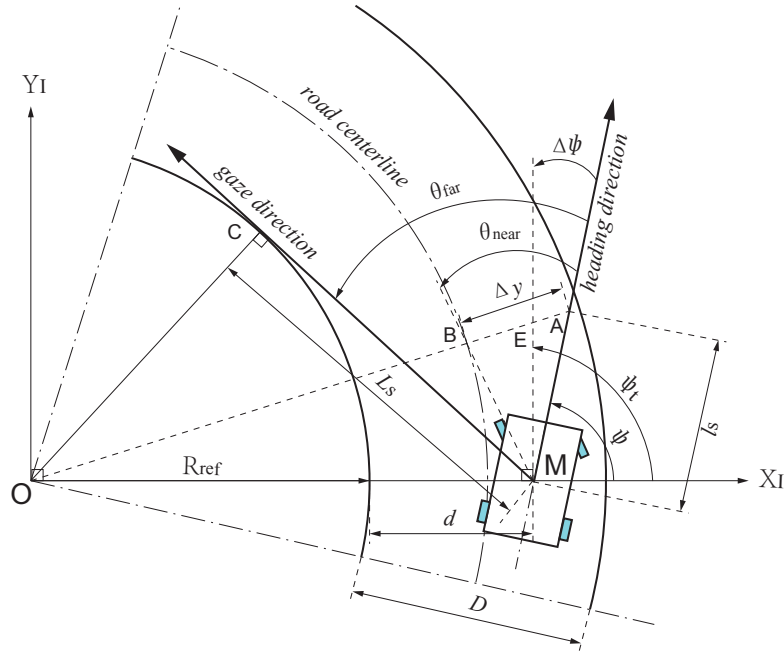


Figure 3.2: Road geometries, vehicle states and driver's visual perception.

3.2.3 Problem Formulation

We formulate the driver parameter estimation problem based on the driver model, road and perception model, steering column model and vehicle model summarized in the previous section. For notational simplicity, let $p_1 = K_a$, $p_2 = K_c$, $p_3 = T_L$, $p_4 = T_I$, $p_5 = T_N$ and $p_6 = t_p$. The driver's near field look-ahead distance ℓ_s is also an important feature of the driver steering characteristics. We thus take ℓ_s as an additional parameter, and let $p_7 = \ell_s$. We further let $p_8 = K_D$, $p_9 = K_G$, $p_{10} = T_{k_1}$, $p_{11} = T_{k_2}$ and $p_{12} = T_{k_3}$ for the high frequency kinesthetic feedback in the driver model. Since the human driver has physical limits, each model parameter is restricted to lie within some compact interval, $p_i \in [\underline{p}_i, \overline{p}_i]$, $i = 1, 2, \dots, 12$. Let $p = (p_1, p_2, \dots, p_{12})^T \in \mathcal{P} = [\underline{p}_1, \overline{p}_1] \times [\underline{p}_2, \overline{p}_2] \times \dots \times [\underline{p}_{12}, \overline{p}_{12}] \subset \mathbb{R}^{12}$. The upper and lower bounds (\overline{p}_i and \underline{p}_i) that define \mathcal{P} are given in Table 3.3.

The combined system of the driver model and the road and perception model can be written in the form

$$\dot{x}^c = A^c(p)x^c + B^c(p)u^c, \quad (3.3a)$$

$$y^c = C^c x^c, \quad (3.3b)$$

where the system state is $x^c = (\Delta\psi, \delta y, x_{d1}, x_{d2}, T_{dr}^{ff}, x_{d3}, x_{d4}, T_{dr}^{fb})^T$, the input is $u^c = (\rho, \beta, r, \delta_s)^T$ and the output is $y^c = T_{dr}^{ff} + T_{dr}^{fb} = T_{dr}$. In the previous expressions T_{dr}^{ff} and T_{dr}^{fb} denote the two components of the driver's steering torque, resulting from the feedforward path and the feedback path of the driver model, respectively. Specially, referring to Figure 3.1, T_{dr}^{ff} and T_{dr}^{fb} can be expressed as follows

$$T_{dr}^{ff} = (T_{com} + T_{ant})G_L G_{nm}, \quad (3.4a)$$

$$T_{dr}^{fb} = -\delta_s G_{k_1} (1 + G_{k_2}) G_{nm}. \quad (3.4b)$$

By measuring u^c and y^c we can identify the driver parameter vector p in (3.3a)-(3.3b). To this end, we define an alternative parameter vector $v = (v_1, v_1, \dots, v_{12})^T$ as follows

$$\begin{aligned} v_1 &= \frac{1}{p_4}, & v_2 &= \frac{1}{p_6}, & v_3 &= \frac{1}{p_5}, & v_4 &= \frac{p_1}{p_5}, \\ v_5 &= \frac{p_2 p_3}{p_4 p_6 p_7}, & v_6 &= \frac{p_2}{p_4 p_7}, & v_7 &= p_7, & v_8 &= p_8, \\ v_9 &= p_9, & v_{10} &= \frac{1}{p_{10}}, & v_{11} &= p_{11}, & v_{12} &= \frac{1}{p_{12}}. \end{aligned} \quad (3.5)$$

The mathematical expressions in the sequel can be simplified by using v instead of p . The system matrices in (3.3a)-(3.3b) are given explicitly by (3.6). It is worth mentioning that, V_x is assumed to be constant in (3.6). One can add V_x to the input vector u^c for varying velocity cases. Since we are interested in identifying the parameter vector v , we augment the state with v and define the new augmented state $x = \begin{bmatrix} (x^c)^T & v^T \end{bmatrix}^T$. The

$$\left[\begin{array}{c|c} A^c(\nu) & B^c(\nu) \\ \hline C^c & 0 \end{array} \right] = \left[\begin{array}{cccccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & V_x & 0 & -1 & 0 \\ V_x & 0 & 0 & 0 & 0 & 0 & V_x \nu_7 & -V_x - \nu_7 & & 0 \\ 0 & \nu_6 - \frac{\nu_1 \nu_5}{\nu_2} - \nu_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 \frac{\nu_2 \nu_4}{\nu_3} & 4 \nu_5 & 4 \nu_2 - 2 \nu_2 & 0 & 0 & 0 & 4 \frac{L_s \nu_2 \nu_4}{\nu_3} & 0 & 0 & 0 \\ -\nu_4 & -\frac{\nu_5 \nu_3}{\nu_2} & -\nu_3 & \nu_3 & -\nu_3 & 0 & -L_s \nu_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 - \nu_3 \nu_{10} - \nu_3 \nu_{12} - \nu_{10} \nu_{12} & 0 & 0 & 0 & -\nu_3 \nu_8 \nu_{12} (\nu_9 + 1) \\ 0 & 0 & 0 & 0 & 0 & 01 - \nu_3 - \nu_{10} - \nu_{12} & 0 & 0 & 0 & -\nu_3 \nu_8 (\nu_9 \nu_{11} \nu_{12} + 1) \\ \hline 0 & 0 & 0 & 0 & 1 & 00 & 0 & 0 & 0 & 0 \end{array} \right]. \quad (3.6)$$

augmented-state system is then given by

$$\dot{x} = \begin{bmatrix} A^c(\nu) & \\ & 0 \end{bmatrix} x + \begin{bmatrix} B^c(\nu) \\ 0 \end{bmatrix} u, \quad (3.7a)$$

$$y = \begin{bmatrix} C^c & 0 \end{bmatrix} x, \quad (3.7b)$$

where $u = u^c$. Notice that although the system in (3.3a)-(3.3b) is linear, the system in (3.7a)-(3.7b) is nonlinear, since the matrices A^c and B^c depend on the augmented state x . If we discretize the system in (3.7a)-(3.7b), we obtain the following discrete augmented system with additive noise terms

$$x_{k+1} = A_D(\nu) x_k + B_D(\nu) u_k + w_k, \quad (3.8a)$$

$$y_k = C_D x_k + v_k, \quad (3.8b)$$

where w_k and v_k are the process noise and the measure noise, respectively. As usual, these noise terms are included to model neglected/unmodeled uncertainties.

In the following sections we estimate the state vector of the system in (3.3) or (3.7) based on the available data, subject to the following constraints

$$\underline{p}_i \leq g_i(\nu) \leq \overline{p}_i, \quad i = 1, 2, \dots, 12, \quad (3.9)$$

where $g_i(\nu)$ is the i th element of the vector-valued function $g(\nu)$ given by

$$g(\nu) = \begin{bmatrix} 1/\nu_1 & 1/\nu_2 & 1/\nu_3 & \nu_4/\nu_3 & \nu_5/(\nu_2 \nu_6) & (\nu_6 \nu_7)/\nu_1 & \nu_7 & \nu_8 & \nu_9 & 1/\nu_{10} & \nu_{11} & 1/\nu_{12} \end{bmatrix}^T. \quad (3.10)$$

Note that some of the parameters in the feedback model, in particular in the neuromuscular system $G_{nm}(s)$, can be considered to be constants that do not change significantly from driver to driver [120, 10]. These parameters will be discussed in Section 3.5.1.

3.3 Field Tests

Several field tests were conducted to validate the previous driver model. The field tests took place at the Ford Dearborn Proving Ground (DPG) in Michigan during November 2015. The Ford DPG is about 1,750 meters from West end to East end and about 900 meters from South end to North end. The width of the double-lane road is about 6 meters. Three kinds of tests were conducted. A steering handling course (SHC) test, a fixed-radius circling (FRC) test and the public road test (PRT). The SHC and FRC tests were conducted at zone 1 and zone 2 of the proving ground, respectively (see Figure 3.3).

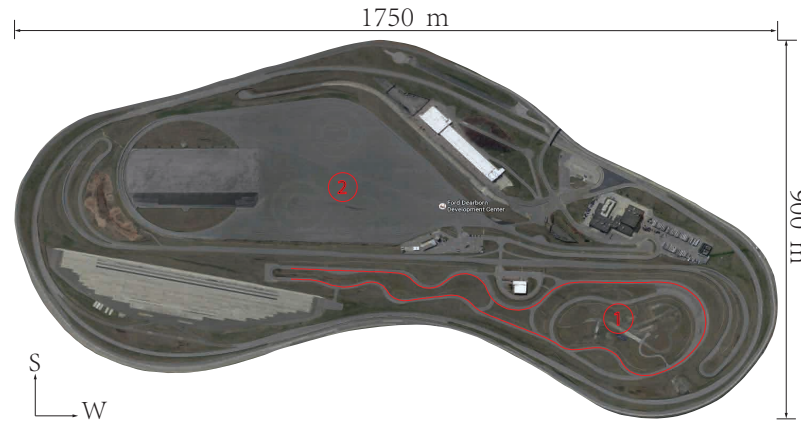


Figure 3.3: The proving ground by the google map.

Three vehicles differing in size and engine power were prepared and were driven by a professional driver mimicking three different types of drivers having distinct driving skills (novice, experienced, and racing). This was done mainly for safety reasons, as untrained novice drivers were not allowed to use the DPG. Consequently, a natural next step along this research direction would be the collection of more data (primarily from untrained novice drivers on the road) in order to further corroborate the conclusions of this chapter.

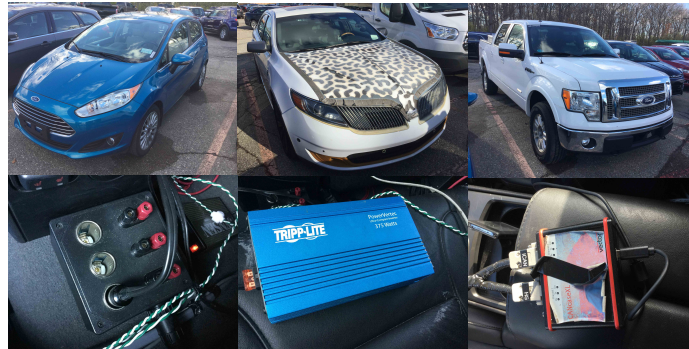


Figure 3.4: Experiment vehicles and some apparatus. 1st row: Fiesta (left), MKS (medium), F150 (right); 2nd row: power source (left), power converter (medium), CAN case (right).

In both the SHC test and the FRC test, the driver was required to maintain the vehicle at a constant velocity throughout the road, while in the PRT test, the driver drove freely on a section of a pre-chosen public road, considering the specific traffic conditions. The proving ground, the experimental vehicles and the drivers were provided by the Ford Motor Company. Figure 3.4 shows the experimental vehicles and the main equipment used for the tests.

All data were collected through a controller area network (CAN) analyzer that interconnected the computer and the in-vehicle CAN buses. There are two CAN channels, namely, the HS-CAN and INFO-CAN channel, both of which have a data transfer rate of 500 [kB/s]. The HS-CAN connects to most of the regular on-board electronic control units (ECU), such as the anti-skip braking system (ABS), the electric power assisted steering (EPAS) system and the restraints control module (RCM). These ECUs share the data on the HS-CAN bus. The INFO-CAN connects to the in-vehicle communications and entertainment system – called the SYNC system, which incorporates the global position system (GPS) and the navigation module.

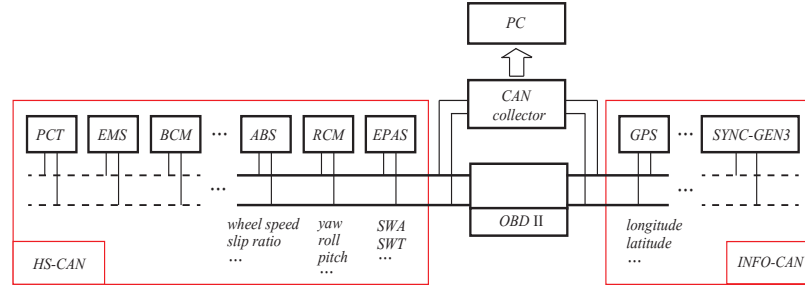


Figure 3.5: Illustration of the CAN network on MKS.

The data collected during the tests were the steering wheel angle, the steering column torque, the yaw rate, and the longitude and the latitude of the vehicle. The signals of the steering wheel angle and the steering column torque were provided by the EPAS, the yaw rate signal was provided by the RCM and position information was provided by the on-board GPS system. Additional variables such as the vehicle yaw angle, the velocity/acceleration of the vehicle, the side slip angle and the road curvature were estimated based on the yaw rate and the GPS position data. The CAN bus network of the MKS is shown in Figure 3.5. The setup of the test conditions for the SHC, FRC and PRT tests are summarized in Table 3.1.

Table 3.1: Steering handling course (constant velocity); CW=clockwise, CCW=counter clockwise.

SHC	Novice	Experienced	Racing
30 [mph]	1 lane, unsmooth	1 lane, smooth	2 lanes, smooth
45 [mph]	1 lane, unsmooth	1 lane, smooth	2 lanes, smooth
	FRC	R=85 [m]	R=200 [m]
	30 [mph]	CW and CCW	CW and CCW
	45 [mph]	CW and CCW	CW and CCW
	PRT	Novice	Experienced
	Westward	unsmooth	smooth
	Eastward	unsmooth	smooth

3.4 Data Analysis and Results

In this section we summarize the data processing step from the driving tests and we apply the joint EKF/UKF and the dual EKF/UKF to estimate the parameters of the assumed driver model. Based on the data analysis, a refined driver model is proposed to better reproduce the actual steering wheel torque command of the driver.

3.4.1 GPS Data Processing

Since the road curvature and the side slip angle of the vehicle were not directly measured, we first obtain the missing values by processing the GPS data, which are given in the form of latitude and longitude. We refer to the method proposed in [121], by which the GPS coordinates are transformed to local navigation coordinates East, North and Up (in this work, the height is zero since the vehicle is traveling on the ground). Three useful coordinate systems used in this transformation are shown in Figure 3.6, namely, the World Geodetic System 1984 (WGS84), the Earth Centered Earth Fixed (ECEF) system and the East, North and Up (ENU) system. The WGS84 system expresses the position vector in terms of the longitude, the latitude and the height (ϕ, λ, h) of the vehicle, while the ECEF system is in terms of the vehicle Cartesian coordinates (x, y, z) . The ENU system is represented locally, which usually works as the navigation coordinate system.

We first converted the GPS coordinates to ECEF coordinates using the following equations:

$$x = \frac{a \cos \phi \cos \lambda}{\chi}, \quad y = \frac{a \cos \phi \sin \lambda}{\chi}, \quad z = \frac{a(1 - e^2) \sin \phi}{\chi}, \quad (3.11)$$

where $\chi = \sqrt{1 - e^2 \sin^2 \phi}$, $a \approx 6.39 \times 10^6$ [m] and $e^2 \approx 6.69 \times 10^{-3}$ are the semi-major axis and the first numerical eccentricity of the earth, respectively. By performing a Taylor expansion of equation (3.11) about ϕ and λ and omitting terms higher than second order, we obtain

$$\begin{aligned} dx &= -\frac{a \cos \lambda \sin \phi (1 - e^2)}{\chi^3} d\phi - \frac{a \sin \lambda \cos \phi}{\chi} d\lambda + \frac{1}{4} a \cos \phi \cos \lambda (-2 \\ &\quad - 7e^2 + 9e^2 \cos^2 \phi) d\phi^2 - \frac{a \sin \lambda \sin \phi (1 - e^2)}{\chi^3} d\phi d\lambda - \frac{a \cos \lambda \cos \phi}{2\chi} d\lambda^2, \\ dy &= -\frac{a \sin \lambda \sin \phi (1 - e^2)}{\chi^3} d\phi + \frac{a \cos \lambda \cos \phi}{\chi} d\lambda + \frac{1}{4} a \cos \phi \sin \lambda (-2 \\ &\quad - 7e^2 + 9e^2 \cos^2 \phi) d\phi^2 - \frac{a \cos \lambda \sin \phi (1 - e^2)}{\chi^3} d\phi d\lambda - \frac{a \sin \lambda \cos \phi}{2\chi} d\lambda^2, \\ dz &= \frac{a \cos \phi (1 - e^2)}{\chi^3} d\phi + \frac{1}{4} a \sin \phi (-2 - e^2 + 9e^2 \cos^2 \phi) d\phi^2. \end{aligned}$$

We finally rotate the ECEF coordinates to obtain the ENU coordinates using the fol-

lowing equations:

$$\begin{pmatrix} de \\ dn \end{pmatrix} = \begin{pmatrix} -\sin \lambda & \cos \lambda & 0 \\ -\sin \phi \cos \lambda & -\sin \phi \sin \lambda & \cos \phi \end{pmatrix} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix}. \quad (3.13)$$

The trajectory of the vehicle can be obtained by integrating the ENU coordinates de and dn in (3.13). The side slip angle β is estimated using the equation [122],

$$\beta = \arctan\left(\frac{V_y}{V_x}\right) - \psi, \quad (3.14)$$

where V_x and V_y are the longitudinal velocity and the lateral velocity of the mass center of the vehicle chassis, respectively, and ψ is the yaw angle. The road curvature ρ is calculated by

$$\rho = \frac{Y''_{\text{cog}}}{(1 + Y'^2_{\text{cog}})^{3/2}}, \quad (3.15)$$

where $Y'_{\text{cog}} = \partial Y_{\text{cog}} / \partial X_{\text{cog}}$, $Y''_{\text{cog}} = \partial^2 Y_{\text{cog}} / \partial^2 X_{\text{cog}}$, and X_{cog} and Y_{cog} are the coordinates of the vehicle in the local ENU system.

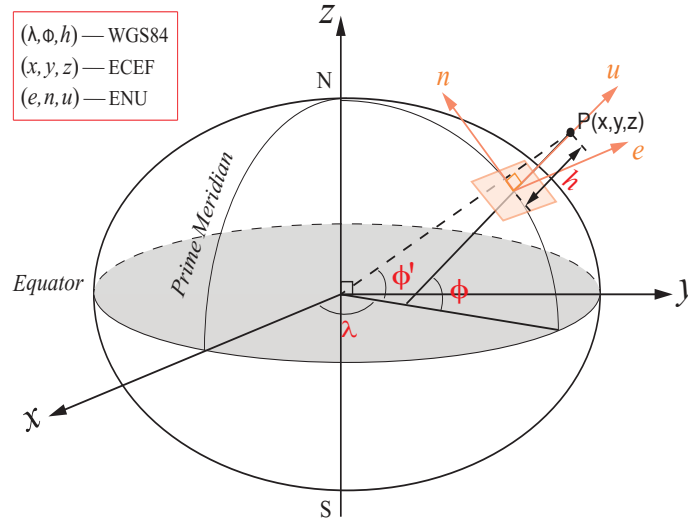


Figure 3.6: Illustration on the different coordinate systems.

3.4.2 Driver Parameter Identification

This section shows the results from the previous driver parameter identification and validation procedure, and provides a comparative analysis. Before processing the field test data, we first implemented the identification approach on a set of data obtained from CarSim/Matlab simulation. This was done in order to confirm the correctness and limitations of the identification algorithm.

CarSim Data Processing

The vehicle model used in the simulation was configured with Carsim 8.0 [123] and was initialized with a constant speed of 15 [m/sec] (54 [km/h]). Other vehicle constants can be found in Table 3.2. In addition, we assumed a high-adhesion asphalt pavement with a constant friction factor of 0.89 for all simulations. The length and the width of the road were configured as 1,000 [m] and 6 [m], respectively.

Table 3.2: Constant parameters of the system.

m	Mass of vehicle	1653	kg
ℓ_f	Distance from center of gravity to front axis	1.402	m
ℓ_r	Distance from center of gravity to rear axis	1.646	m
L_s	Distance from center of gravity to far field visual point	15	m
I_z	Moment of inertia of the vehicle	2765	kgm ²
J_s	Moment of inertia of the steering column	0.11	kgm ²

A path composed of a sequence of straight segments, circular segments, and clothoids was given as an input. The configured road curvature was obtained through a sensor provided by Carsim. Since the road curvature data from Carsim are noisy, we applied a first-order lowpass filter with a cut-off frequency at 2.5 rad/sec to eliminate the noise before inputting this signal to the driver model.

After collecting the necessary simulation data, namely, the steering wheel angle δ_s , the road curvature ρ , the side slip angle β and the yaw rate r in the input vector u^c , and the steering wheel torque T_{dr} in the output y^c . We then implemented the joint EKF to estimate the driver model parameters. The results are given in Figure 3.7. We only show the results of the joint EKF here, since the results given by the other filters were quite similar.

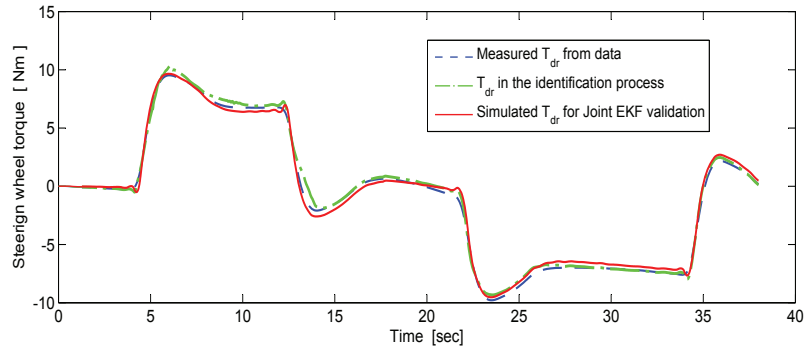


Figure 3.7: The data, the training curve and the simulated curve for the steering wheel torque.

In Figure 3.7, the blue curve shows the steering wheel torque from the data, the green curve shows the estimation of the steering wheel torque during the training process, and the red curve shows the validation result, which is obtained by using the identified driver model parameters from the training process in the simulation. The simulated

Table 3.3: Driver model parameters;

JEKF=Joint EKF, DEKF=Dual EKF, UB=upper bound, LB=lower bound.

Parameter	JEKF(§)	JEKF	JUKF	DEKF	DUKF	UB	LB
K_a	56.56	22.10	21.62	21.29	21.29	100	5
K_c	19.82	149.87	152.35	151.88	150.96	200	5
T_L [sec]	0.90	0.33	0.34	0.33	0.33	5	0
T_I [sec]	0.48	0.26	0.26	0.26	0.26	0.5	0
T_N [sec]	0.30	0.18	0.19	0.19	0.20	0.3	0.01
t_p [sec]	0.19	0.11	0.11	0.11	0.11	0.5	0.01
ℓ_s [m]	3.47	12.06	12.16	12.25	12.07	15	3
K_D [m]	1.50	0.37	0.27	0.11	0.31	1.5	0.1
K_G [m]	-0.41	-0.74	-0.64	-0.79	-0.43	-0.4	-1.5
T_{k_1} [m]	1.05	1.50	1.54	1.97	1.57	6	1
T_{k_2} [m]	5.13	3.82	3.71	3.42	3.81	6	1
T_{k_3} [m]	0.01	0.01	0.01	0.01	0.01	0.03	0.01

result agrees well with the data. The identified driver model parameters are given in the second column of Table 3.3.

Field Test Data Processing

We processed the field test data using the joint EKF, the joint UKF, the dual EKF and the dual UKF separately, so that we can compare the identified driver parameters obtained from these four different methods. For instance, we took the set of data from the SHC tests corresponding to the conditions of a “Racing Driver” with a constant velocity of 45 [mph] in Table 3.1. In each implementation, we used the first 60% of the data for parameter training and then used the remaining 40% of the data for validation.

By designing the appropriate Kalman filter parameters, such as the process noise covariance, the measurement noise covariance and the initial state covariance matrix, we obtained reasonably good estimation of the parameters. The process noise covariance is considered to be the most critical, and therefore had to be carefully tuned [124, 125]. Figure 3.8 illustrates the steering wheel torque from data, the training curve for each filter and the simulated output corresponding to the identified model parameters.

The green plots in Figure 3.8 show how the prediction of the steering wheel torque at the current time step, provided by the joint/dual E-/UKF based on past data, agrees with the current data. After about 1 minute the prediction results get stabilized and agree well with the data.

The trajectories of the estimated states (we only show the driver parameters, and each parameter is scaled such that the initial value is one) corresponding to the joint EKF are given in Figure 3.9. The red plots in Figure 3.8, which are drawn to validate the identified driver parameters, agree well with the data. Although one sees some difference between the validation results and the data, the results are reasonable, since the parameters of the real driver may change slowly with time. This effect is investigated next.

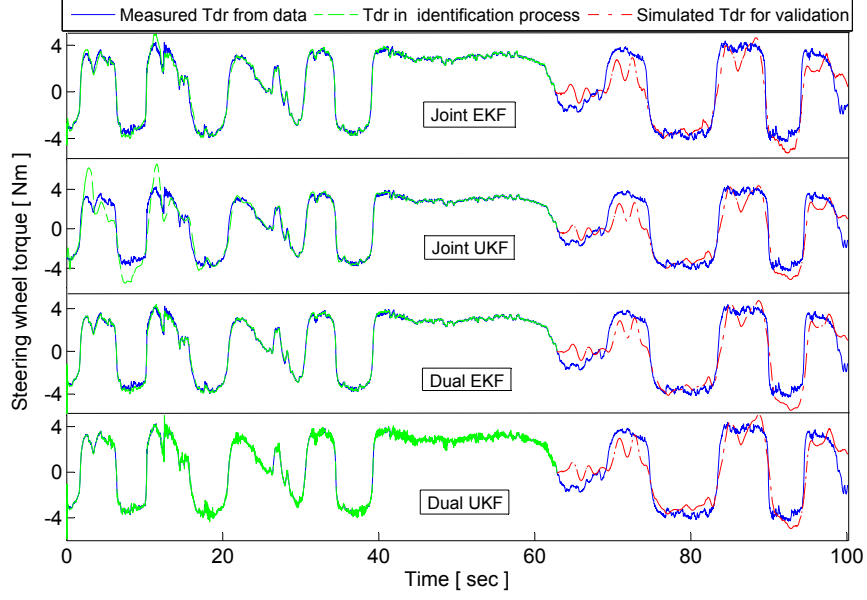


Figure 3.8: The data, the training curve and the simulated curve from the Joint/Dual E-/UKF.

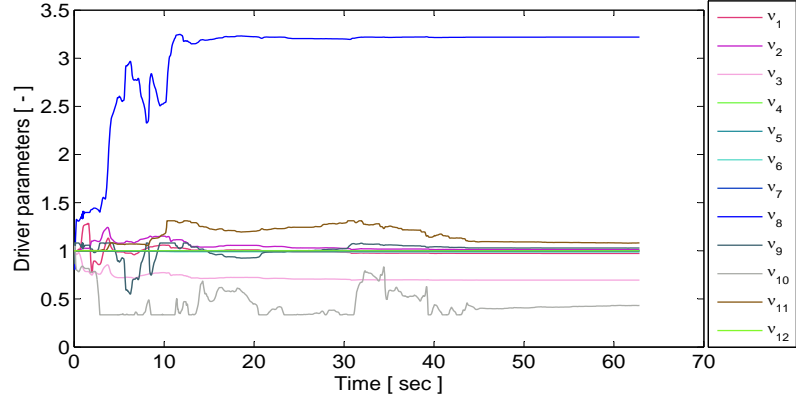


Figure 3.9: The time histories of the driver parameters during the training process. Steady state is reached after 45 seconds.

3.4.3 Driver Model Refinement

Based on the results from the previous section, we refined the model by assuming that the process noise for the parameter vector \mathbf{v} is colored. To this end, we let

$$\dot{\mathbf{v}} = \boldsymbol{\zeta}, \quad \dot{\boldsymbol{\zeta}} = \boldsymbol{\xi}, \quad (3.16)$$

where $\boldsymbol{\xi}$ is a zero-mean white process noise and $\boldsymbol{\zeta}$ is a colored process noise with unknown time-varying mean. By discretizing (3.16) with a sampling interval Δt , one ob-

tains

$$\mathbf{v}_k = \mathbf{v}_{k-1} + \Delta t \boldsymbol{\zeta}_{k-1}, \quad \boldsymbol{\zeta}_k = \boldsymbol{\zeta}_{k-1} + \Delta t \boldsymbol{\xi}_{k-1}. \quad (3.17)$$

If $\boldsymbol{\xi}_{k-1}$ is uncorrelated with $\boldsymbol{\zeta}_{k-1}$, $\boldsymbol{\zeta}_k$ is colored process noise in the sense that $\boldsymbol{\zeta}_k$ is correlated with itself at different time steps [126]

$$\mathbb{E}(\boldsymbol{\zeta}_k \boldsymbol{\zeta}_{k-1}^T) = \mathbb{E}(\boldsymbol{\zeta}_{k-1} \boldsymbol{\zeta}_{k-1}^T + \Delta t \boldsymbol{\xi}_{k-1} \boldsymbol{\zeta}_{k-1}^T) = \mathbb{E}(\boldsymbol{\zeta}_{k-1} \boldsymbol{\zeta}_{k-1}^T) \triangleq Q_\zeta, \quad (3.18)$$

where $Q_\zeta \neq 0$ is the covariance matrix. For the noise $\boldsymbol{\zeta}$ at any two different time steps t and s ($t > s$) one obtains that $\mathbb{E}(\boldsymbol{\zeta}_t \boldsymbol{\zeta}_s^T) = \mathbb{E}(\boldsymbol{\zeta}_{t-1} \boldsymbol{\zeta}_s^T) = \dots = \mathbb{E}(\boldsymbol{\zeta}_s \boldsymbol{\zeta}_s^T) = Q_\zeta$, and hence one can summarize the covariances for $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}$ as follows

$$\mathbb{E}(\boldsymbol{\zeta}_t \boldsymbol{\zeta}_s^T) = Q_\zeta, \quad \mathbb{E}(\boldsymbol{\zeta}_t \boldsymbol{\xi}_s^T) = 0, \quad \mathbb{E}(\boldsymbol{\xi}_t \boldsymbol{\xi}_s^T) = Q_\xi \delta_{ts}, \quad (3.19)$$

where Q_ξ is the covariance matrix and δ_{ts} is given by (7.32a).

Equations (3.16) allow the parameter vector \mathbf{v} to drift with time. We implemented all filters using this model and recorded the estimates of \mathbf{v} at each time step. We then performed simulations with the time-varying parameters. Figure 3.10 shows the results for the Joint UKF case. The results with the other filters are similar, and are thus, omitted.

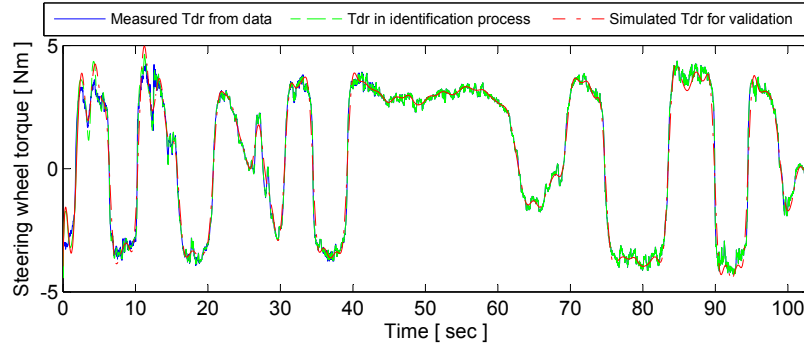


Figure 3.10: The data, the training curve and the simulated curve from the Joint UKF

Figure 3.10 indicates that, the parameterized driver model with time-varying parameters characterizes the driver's steering behavior much more accurately. This implies that the parameterized two-point visual driver model architecture shown in Figure 3.1 is valid, but the parameters are not necessarily constant, and may vary slowly with time.

The driver parameters convergence with time, along with their 2σ -bounds, are shown in Figure 3.11. Figure 3.12 shows in greater detail the estimate (solid red line) and the confidence levels (blue dotted line) for ℓ_s . The plots for the other parameters are similar. From Figure 3.11 one observes that, although most of the parameters converge to some constants, some of them exhibit drift, specifically, K_a , K_c and T_L . This behavior accounts for the difference between the simulation result (red line) and the test data (blue line) shown in Figure 3.8. In terms of driver parameter identification, this result suggests that K_a , K_c and T_L are the most important parameters to track in an on-line

identification scheme.

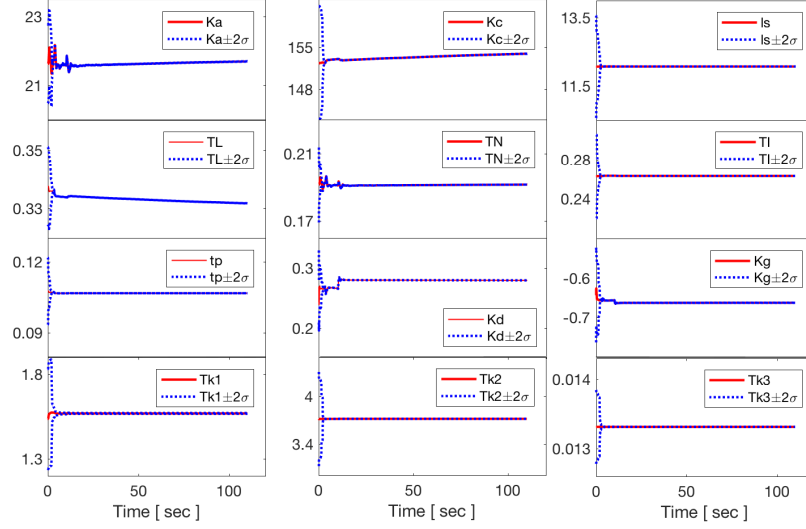


Figure 3.11: The trajectory of the driver parameters with $\pm 2\sigma$ error during the training process.

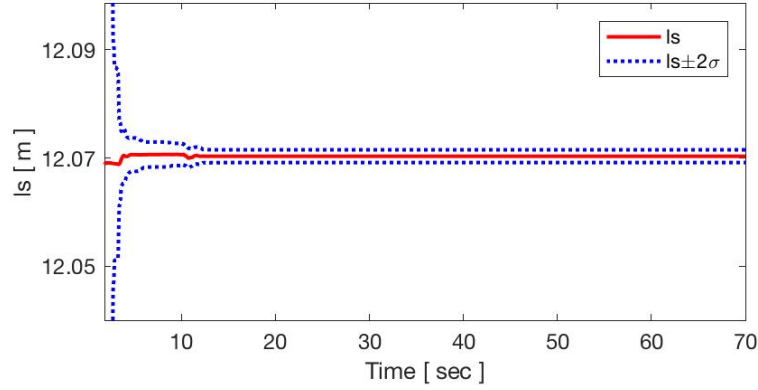


Figure 3.12: Detail of estimate of ℓ_s along with the 2σ confidence bounds.

3.5 Driver Comparison and Analysis

The previous section estimates the identified driver parameters using four different non-linear filters. Our main motivation for parameter estimation is to be able to distinguish the different driver styles based on the identified driver parameters from experimental data. Empirical evidence suggests that one potential strong distinguishing feature of driver style is the smoothness of the applied steering command [127, 128]. In order to test this theory we first analyzed the driver's steering behavior according to the identified driver parameters, and we then analyzed the driver's steering behavior by comparing the wavelet transform of the control signals from different drivers, since it is well-known that wavelet transform contains information about the local smoothness of a signal [129].

Table 3.4: Driver model parameters.

Parameter		K_a	K_c	T_L	T_I	T_N	t_p	ℓ_s	K_D	K_g	T_{k_1}	T_{k_2}	T_{k_3}
30 mph	Racing	21.7	153.5	0.33	0.26	0.19	0.11	12.1	0.28	-0.66	1.57	3.72	0.013
	Experienced	21.9	158.6	0.35	0.28	0.20	0.11	12.1	0.66	-0.40	5.95	3.73	0.013
	Novice	17.1	113.5	0.25	0.20	0.15	0.10	8.7	0.37	-0.40	2.92	3.29	0.013
45 mph	Racing	21.9	155.8	0.33	0.26	0.19	0.11	12.1	0.30	-0.66	1.53	3.72	0.013
	Experienced	21.8	156.8	0.35	0.28	0.19	0.11	12.2	0.35	-0.78	2.31	3.73	0.013
	Novice	17.4	121.4	0.27	0.21	0.16	0.08	8.8	0.33	-0.81	5.65	3.21	0.013

3.5.1 Driver Parameter Analysis

Table 3.4 shows the parameters for the racing, experienced and novice driver in the SHC tests (MKS vehicle). Since some of the parameters are varying with time (namely, K_a , K_c and T_L), we only show their time-average values in Table 3.4.

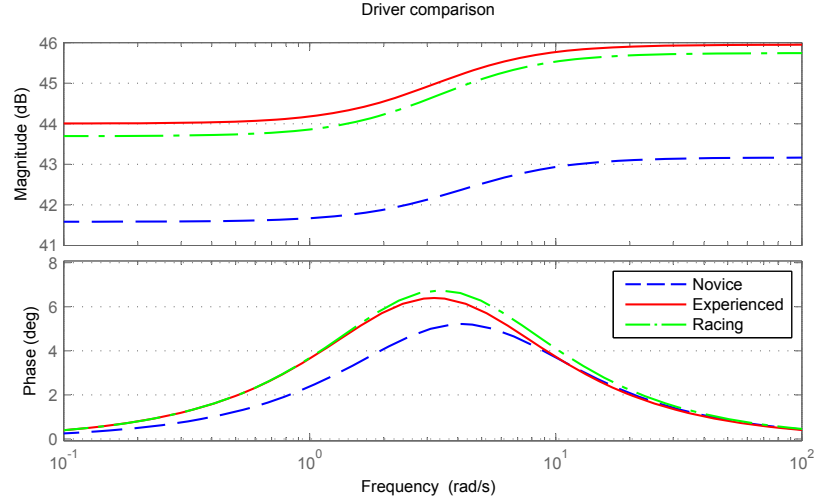


Figure 3.13: The bode plots of G_c for the novice, the experienced and the racing drivers, 45 mph.

From the tests shown in Table 3.4, one observes that K_c is much larger than K_a . This indicates that in the lane-keeping task the driver pays more attention to θ_{near} than θ_{far} , as expected. This result may change, however, for a different driving task [23]. In this section we wish to compare the experienced driver steering command with the novice driver steering command for the same task. The question we wish to answer is whether we are able to distinguish between these two (supposedly) distinct driver styles by analyzing only the driver steering command. From Table 3.4 one sees that the parameters K_a , K_c , T_L and T_I for the novice driver are smaller than that for the experienced driver. The anticipatory gain K_a and the compensatory gain K_c represent the attention the driver pays to θ_{far} and θ_{near} , respectively. An increase of K_a leads to $d\theta_{\text{far}}/dt < 0$ (oversteering), and the vehicle gets closer to the inner curb of the road. An increase of K_c leads to more compensation ($d\theta_{\text{near}}/dt < 0$), and the vehicle gets closer to the road

centerline. The authors of [115] mention that K_c may depend on the driver's cautiousness (e.g., the driver avoids driving too close to the border line) and small K_c leads to a great tendency to cut around the bends. The parameters T_L and T_I define a lead compensation in the compensatory control path of the driver. A larger T_L corresponds to higher compensation rate of θ_{near} (the speed of θ_{near} to reach the desired value), but the system will be oscillating if T_L is too large [115]. T_I determines the bandwidth of the frequencies of θ_{near} to be compensated. Small values of T_I mean that the driver compensates all frequencies including the high frequency noise, hence leading to an oscillatory system. If T_I is large ($T_I < T_L$), the bandwidth of the compensatory loop is narrow such that most frequencies of θ_{near} are filtered.

The Bode plots of the lead compensator G_c are shown in Figure 3.13. One sees that the magnitude of G_c for the novice driver is the smallest and the center frequency is the highest. This indicates that the compensatory control of the novice driver is slow and the driver is more likely to compensate the high frequencies of θ_{near} .

The near field of view visual distance ℓ_s for the novice driver in Table 3.4 is smaller than that for the experienced and the racing drivers. This does not necessarily imply, however, that the larger the preview distance ℓ_s the better. For instance, the authors in [130] observed that the driver's compensatory behavior is reduced with increasing preview distance (5 - 100 [m]) and pointed out that, with a preview distance above a certain point, the drivers no longer minimize the lateral error but use the additional preview to obtain a smooth path. The preview distance may also depend on the road geometry [131].

We analyzed the high frequency feedback part $G_{fb} = G_{k1}(1 + G_{k2})$ of the model (see Figure 1.3), where G_{k1} and G_{k2} are given by (1). The Bode plots for the three drivers using the identified parameters are shown in Figure 3.14.

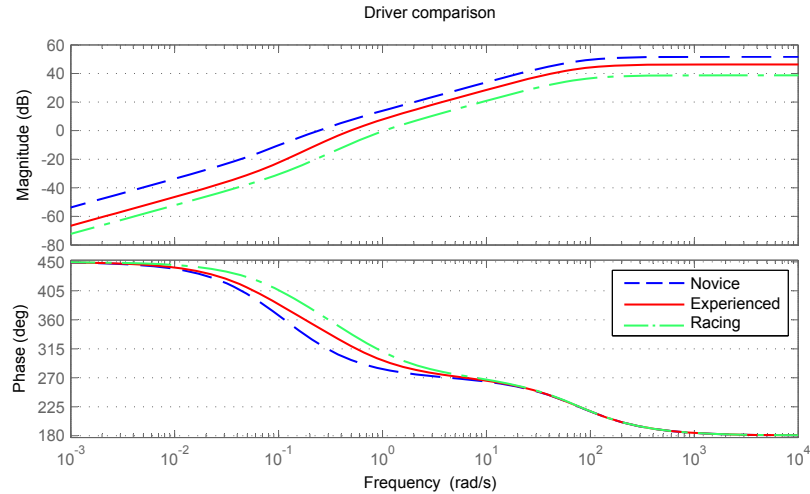


Figure 3.14: The bode plots of G_{fb} for the novice, the experienced and the racing drivers (45 mph).

Figure 3.14 shows that G_{fb} has phase lead and high pass properties, and the three

Bode plots look quite close to each other. Considering that the input signal to G_{fb} (steering wheel angle) typically does not have many high frequency components, and the low frequencies are filtered, only a narrow band of frequencies can effectively pass through G_{fb} as feedback to the driver. This result indicates that the effect of the high frequency feedback may be small, and hence the visual information is more important to the driver than the steering feel in his/her hands during a lane-keeping task. Figure 3.14 also indicates that the parameters in G_{fb} do not distinguish between the drivers since the Bode plots are close to each other. We may thus be able to fix the parameters in G_{fb} to represent the steering behaviors of different drivers, as in [10, 23] where K_D and T_{k_1} are considered to be constants (i.e., $K_D = 1$, $T_{k_1} = 2.5$). It is worth mentioning that, besides K_D and T_{k_1} in [10, 23], the time delay t_p and the parameter T_N in the neuromuscular system are also treated as constants (i.e., $t_p = 0.151$, $T_N = 0.11$).

Figure 3.15 shows the plots of K_a vs K_c . As shown from the analysis of the experimental data in Section 3.4.3, the gains K_a and K_c drift with time. Furthermore, the parameters of the novice driver change faster and take values in a larger range than both the experienced driver and the racing driver. This result may indicate that, at least for a lane-keeping task, the steering behavior of the experienced driver and the racing driver is smoother than the novice driver. To confirm this conjecture, in the next section we perform a wavelet analysis of the control signals of the experienced/racing driver and the novice driver and compare the two.

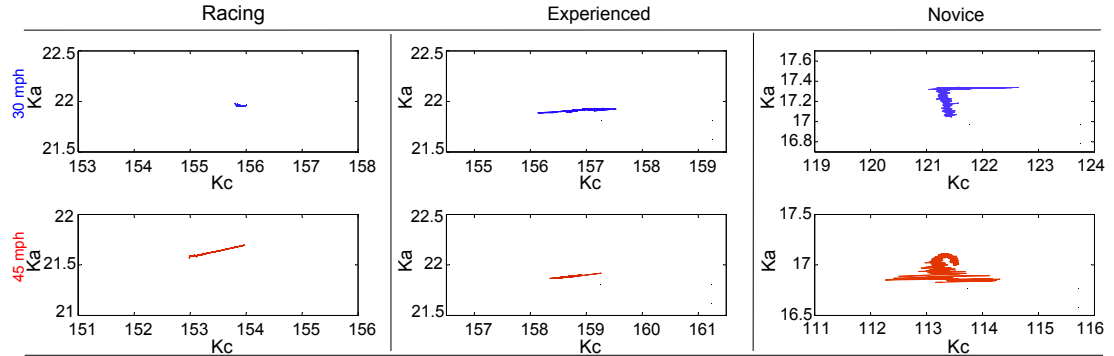


Figure 3.15: The plots of K_a vs K_c for the three types of drivers.

3.5.2 Wavelet Analysis of Driver Steering Torque Command

In this section we compare the steering commands of the novice, experienced and racing drivers in terms of their frequency characteristics and local smoothness properties. Recall that the continuous wavelet transform (CWT) for a given signal f at scale $s \geq 0$ and translation $\tau \in \mathbb{R}$ is written as [132, 133]

$$Wf(s, \tau) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} f(t) \psi^* \left(\frac{t - \tau}{s} \right) dt, \quad (3.20)$$

where ψ^* is the complex conjugate of the mother wavelet ψ . Many wavelet bases are available, such as Morlet, Paul, Haar, Daubechies, Coiflets and Symlets [134, 135].

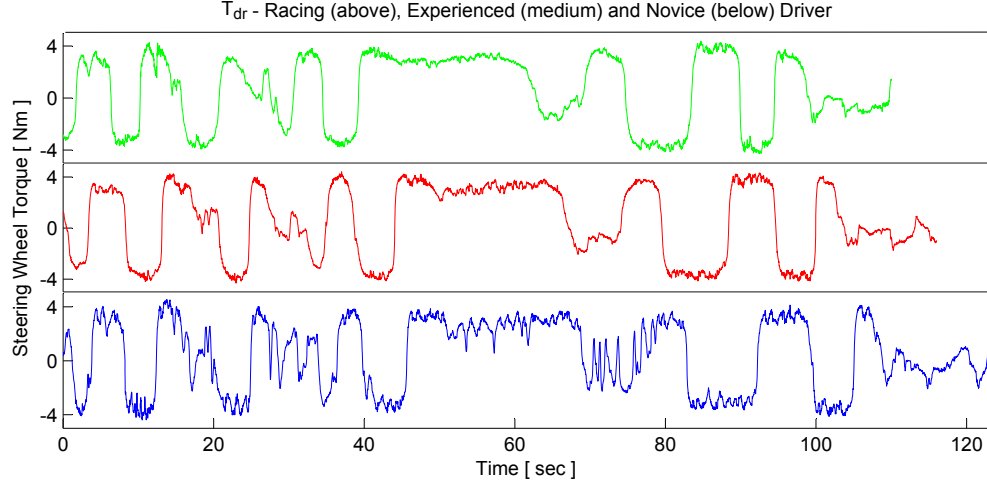


Figure 3.16: The steering wheel torque of the racing, experienced and novice driver (MKS, 45 mph).

Figure 3.16 shows the steering wheel torques for the novice, experienced and racing driver, respectively. In order to compare the frequency content of the signals, we performed a CWT of the steering wheel torques shown in Figure 3.16 using the real-valued Daubechies wavelet 3 function with respect to Equation (3.20). The graphs of the absolute coefficients of the CWT of the steering wheel torques in Figure 3.16 are shown in Figure 3.17. The color regions of the graph indicate the local modulus maxima.

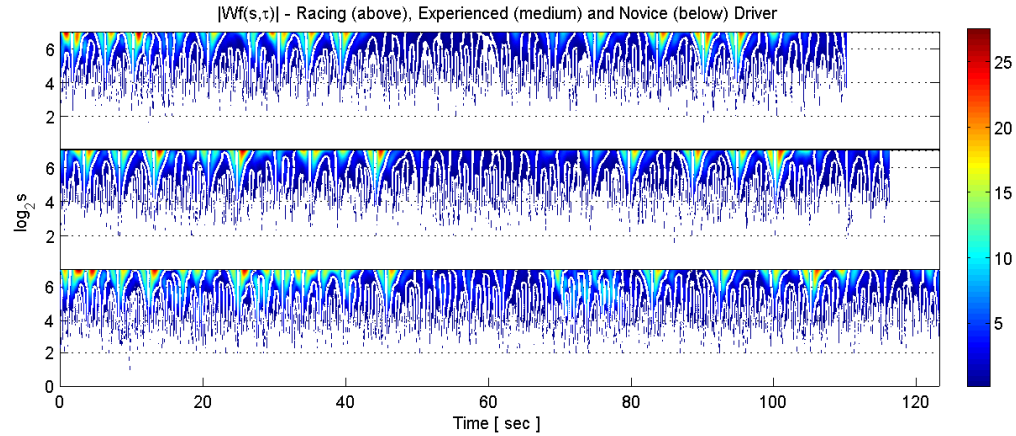


Figure 3.17: Wavelet transform of T_{dr} of the racing (above), experienced (medium) and novice (below) driver.

The results in Figure 3.17 show that, in the same steering handling course, the CWT of the control signal of the novice driver has more local maxima than the experienced and the racing driver. This may be used to evaluate the performance of the steering behavior of the driver. The local maxima can be used to detect the position of the local singularities, as well as to determine the associated Lipschitz exponents using the following theorem [129].

Theorem 3.5.1 Suppose that the wavelet $\psi(t)$ is the n^{th} derivative of a smooth function, is n times continuously differentiable, and has compact support. Let $f(t)$ be a tempered distribution whose wavelet transform is well defined over $[a, b]$, and let $\tau_0 \in [a, b]$. Assume that there exists $s_0 > 0$ and a constant C , such that for all $\tau \in [a, b]$ and $s < s_0$, the modulus maxima of $Wf(s, \tau)$ belong to the cone defined by

$$|\tau - \tau_0| \leq Cs. \quad (3.21)$$

Then $f(t)$ is uniformly Lipschitz n in a neighborhood of τ , for all $\tau \in [a, b], \tau \neq \tau_0$. Furthermore, $f(t)$ is Lipschitz α ($\alpha < n$) at τ_0 , if and only if there exists a constant A such that at each modulus maximum (s, τ) in the cone (3.21)

$$|Wf(s, \tau)| \leq As^\alpha. \quad (3.22)$$

By taking the logarithm of both sides of Equation (3.22), one obtains

$$\log |Wf(s, \tau)| \leq \log A + \alpha \log s. \quad (3.23)$$

The Lipschitz exponent α is therefore determined by the maximum slope of the straight lines of $\log |Wf(s, \tau)|$ on a logarithmic scale. Here we only perform CWT of the steering wheel torque of the experienced driver and show the process to determine the Lipschitz exponent. We adopt the Daubechies wavelet 3 that is orthogonal, compactly supported and has three vanishing moments, by which we can determine Lipschitz exponents $\alpha < 3$. Figure 3.18 plots the absolute CWT coefficients in the time-scale domain.

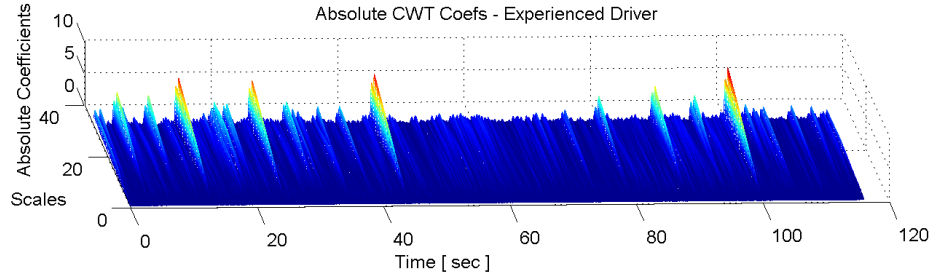


Figure 3.18: The absolute CWT coefficients $|Wf(s, \tau)|$.

We find the lines of maxima from Figure 3.18 and determine the positions of the singularities. The singularities are all the points on the time axis that the lines of maxima converge to. There may be multiple lines of maxima converging to the points that are close to each other, due to the number of the vanishing moments of the mother wavelet or the Lipschitz exponent of the singularity [129].

Figure 3.19 illustrates the histogram showing the distribution of the Lipschitz exponents corresponding to the experienced and racing driver. By comparing the results in Figure 3.19, one observes that the control signal (T_{dr}) of the racing driver has a smaller number of singularities. This result indicates that the steering wheel torque command of the racing driver is smoother than the experienced driver. The distribution of the

Lipschitz exponent α of the racing driver shows a smaller minimal, maximal and mean value than the experienced driver. This statistical result indicates that the singularities of the steering wheel torque command of the racing driver are likely to be more irregular and impulsive than the experienced driver. The racing driver perhaps tends to sacrifice smoothness locally (smaller Lipschitz exponents) by making good use of the double-lane road, such that (s)he could obtain overall better smoothness (fewer singularities) than the experienced driver. The experienced driver, who was only allowed to drive within a single lane of the road, behaved less aggressively since the mean and minimal value of the Lipschitz exponents are larger than the racing driver.

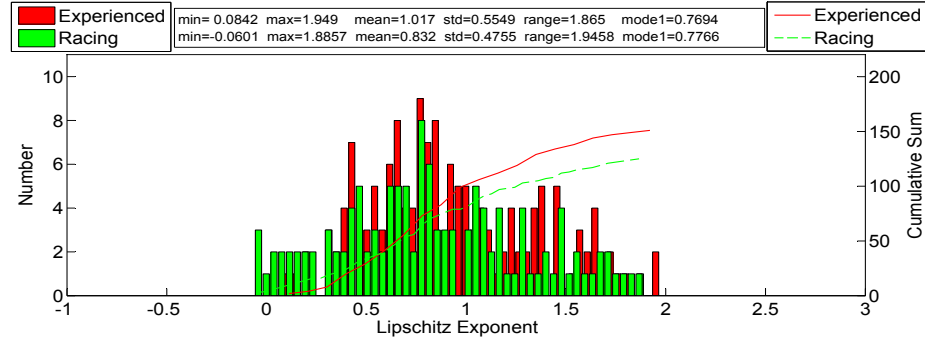


Figure 3.19: The histogram of the Lipschitz exponents α for the experienced and racing driver.

Figure 3.20 illustrates the histogram showing the distribution of the Lipschitz exponents corresponding to the novice driver and the experienced driver. By comparing the results in Figure 3.20, one observes that the control signal (T_{dr}) of the novice driver has a larger number of singularities, and the Lipschitz exponent α shows a larger range from -0.1074 to 2.204 , with a larger standard deviation of 0.6876 . This result implies that the steering wheel torque command of the novice driver is more noisy than the experienced driver (see Figure 3.16). Furthermore, the distribution of α exhibits multiple modes, and the first mode on the left is smaller than that of the experienced driver. These features observed from the distribution of the Lipschitz exponents may be used to distinguish the control signals of different drivers and hence classify drivers into different groups.

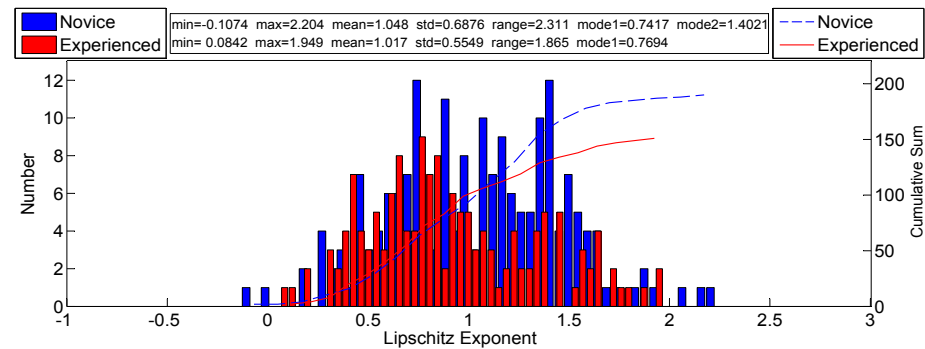


Figure 3.20: The histogram of the Lipschitz exponents α for the novice and experienced driver.

3.6 Conclusion

This chapter adopts the parameterized two-point visual driver model to characterize the steering behavior of a driver, and conducts a series of field tests to investigate the validity of this model to predict human driver behavior and driving style. We have implemented four nonlinear filters, namely, the joint EKF, the joint UKF, the dual EKF and the dual UKF to estimate the parameters based on field test data conducted at Ford's Dearborn Development Center (DDC) test facility. The validation results agree well with the data. The UKF is considered to be more accurate than the EKF in propagating the Gaussian random variables, but the difference is not obvious in this work. The results of our investigation indicate that some of the driver parameters are not exactly constant, but rather vary slowly during a driving task more than a few minutes long. This observation suggests that similar parameterized driver models need to incorporate this effect to faithfully represent reality.

The main difficulty to use the two-point visual driver model may be the inconvenience of measuring the two visual angles θ_{near} and θ_{far} . This dissertation calculates the road curvature ρ_{ref} using the GPS data and adopts a linear estimator in (3.2) to obtain the values of θ_{near} and θ_{far} . One needs to estimate ρ_{ref} if it is unavailable [119].

The parameters of the driver model provide some interesting features to better understand the steering behavior of different types of drivers. An experienced driver is likely to pay more attention to both the anticipatory control (θ_{far}) and the compensatory control (θ_{near}) than a novice driver. The model parameters may also depend on the driving task [23]. The wavelet transform provides insights into the driver's control signal, in terms of the number and the location of the singularities of the signal and the distribution characteristics of the associated Lipschitz exponents. These can be used to characterize the control signal into different levels of smoothness. Our analysis showed that the steering wheel torque of an experienced driver has fewer singularities, and the Lipschitz exponents seem to follow a comparatively more concentrated distribution. Based on this work, a potential next step would be to use machine learning ideas to distinguish the behaviors of the drivers, and to classify the drivers into distinct categories based on features arising from the wavelet transform.

CHAPTER 4

VEHICLE MODELING AND PARAMETER ESTIMATION

4.1 INTRODUCTION

Several vehicle models can be used to represent a vehicle's behavior and help with controller implementation and testing. Such vehicle models are important for the design of advanced control algorithms related to vehicle's active safety. In the past, many models have been proposed for studying the vehicle dynamics and for investigating stability and handling characteristics of vehicles. Examples include the single track model [84, 136], the double track model [136, 137], and the full vehicle model [138, 139]. Commercial software such as CarSim use high-fidelity vehicle models for simulation and animation [123].

The values of the parameters of each vehicle model are critical for accurately predicting the behavior of the vehicle under various operating regimes. We apply the ALM-UKF to estimate the vehicle model parameters for three different vehicle models using both simulation data and experimental data, and we compare its performance against a UKF tuned manually by a human expert.

This chapter is organized as follows: Section 4.2 introduces the three mathematical vehicle models used in this work. Section 4.3 presents the experimental platform we used to collect data. Section 4.4 shows and validates the parameter estimation results. Finally, Section 4.5 summarizes the results of this study.

4.2 VEHICLE MODELING

In this section, we describe the three vehicle models used in this work, namely, the single-track model, the double-track model and a full 11-dof vehicle model [138, 84, 136, 137, 140, 139]. The single-track model incorporates the effect of load transfer arising from the longitudinal acceleration or deceleration of the vehicle, while the double-track model takes into account the effect of load transfer arising from the lateral acceleration of the vehicle. Besides the acceleration caused by the load transfer, the full vehicle model takes also into consideration the suspension dynamics and hence also the additional load transfer arising from the rolling and pitching motions of the vehicle sprung mass.

4.2.1 Single-Track Model

The single-track model takes into consideration the longitudinal and lateral displacement, as well as the yaw motion of the vehicle, as shown in Figure 4.1.

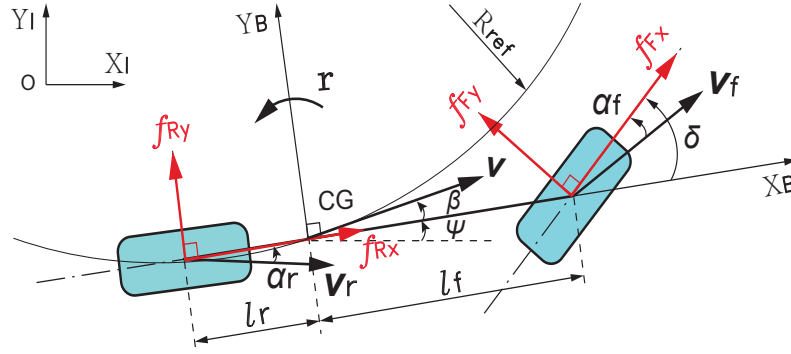


Figure 4.1: Single-track vehicle model.

In this figure, $X_I - O - Y_I$ and $X_B - CG - Y_B$ denote the inertial frame fixed on the ground and the body frame fixed on the vehicle, respectively. V_f , V_r and V denote the velocities at the front and rear wheels and the vehicle's center of gravity (CG), and α_f , α_r and β denote the sideslip angles of the front and the rear wheel and CG, respectively. The parameters ℓ_f and ℓ_r denote the distances of the CG to the front and rear axles, f_{ij} ($i = F, R$ and $j = x, y$) denote the longitudinal and lateral friction forces at the front and rear wheels, and ψ and r denote the yaw angle and the yaw rate of the vehicle, respectively. Finally, δ is the steering angle of the front wheel. The equations of motion of the model can be expressed in a body-fixed frame with the origin at CG as follows [84]:

$$\dot{V}_x = (f_{Fx} \cos \delta - f_{Fy} \sin \delta + f_{Rx}) / m + V_y \dot{\psi}, \quad (4.1a)$$

$$\dot{V}_y = (f_{Fx} \sin \delta + f_{Fy} \cos \delta + f_{Ry}) / m - V_x \dot{\psi}, \quad (4.1b)$$

$$\dot{r} = ((f_{Fy} \cos \delta + f_{Fx} \sin \delta) \ell_f - f_{Ry} \ell_r) / I_z, \quad (4.1c)$$

where V_x and V_y are the components of V along the X_B and Y_B directions, respectively; m is the total mass, and I_z is the moment of inertia of the vehicle about the vertical axis. If we include the vehicle position and the orientation into the state vector x such that $x = [V_x, V_y, r, x_I, y_I, \psi]^T$, we then obtain the following kinematic equations:

$$\dot{x}_I = V_x \cos \psi - V_y \sin \psi, \quad (4.2a)$$

$$\dot{y}_I = V_x \sin \psi + V_y \cos \psi, \quad (4.2b)$$

$$\dot{\psi} = r, \quad (4.2c)$$

where x_I and y_I are the coordinates of CG in the inertial frame.

4.2.2 Double-Track Model

The double-track model takes into consideration the longitudinal, lateral and yaw motion of the vehicle, but considers the load difference between the left and right wheels arising from the lateral load transfer. The velocity and the tire friction forces corresponding to each wheel must be calculated separately. The double-track model is shown in

Figure 4.2.

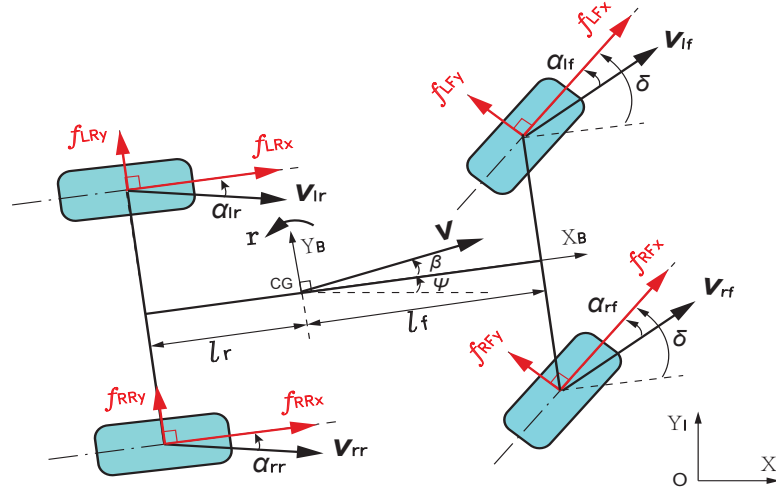


Figure 4.2: Double-track vehicle model.

In this figure, $f_{i,j,k}$ ($i = L, R$, $j = L, R$ and $k = x, y$) denote the longitudinal or lateral friction force for each wheel, respectively. The remaining notation is similar to Figure 4.1 and hence is omitted. The vehicle's equations of motion in the body-fixed frame are given by:

$$\dot{V}_x = ((f_{LFx} + f_{RFx}) \cos \delta - (f_{LFy} + f_{RFy}) \sin \delta + f_{LRx} + f_{RRx}) / m + V_y \dot{\psi}, \quad (4.3a)$$

$$\dot{V}_y = ((f_{LFx} + f_{RFx}) \sin \delta + (f_{LFy} + f_{RFy}) \cos \delta + f_{LRy} + f_{RRy}) / m - V_x \dot{\psi}, \quad (4.3b)$$

$$\dot{r} = (((f_{LFy} + f_{RFy}) \cos \delta + (f_{LFx} + f_{RFx}) \sin \delta) \ell_f - (f_{LRy} + f_{RRy}) \ell_r) / I_z, \quad (4.3c)$$

4.2.3 Full Vehicle Model

The full vehicle model considers the dynamics of the sprung and unsprung mass of the vehicle separately. The equations of motion for the total mass are the same as (4.3a)-(4.3c) for the double-track model. To make the model more accurate, we take the air resistance into account and modify (4.3a) as follows

$$\begin{aligned} \dot{V}_x = & ((f_{LFx} + f_{RFx}) \cos \delta - (f_{LFy} + f_{RFy}) \sin \delta + f_{LRx} \\ & + f_{RRx}) / m + V_y \dot{\psi} - C_D \rho_{air} A V_x^2 / 2, \end{aligned} \quad (4.4)$$

where C_D is the air resistance coefficient, ρ_{air} is the air density, and A is the frontal area of the vehicle. The vertical translation is accounted for by a riding model as shown in Figure 4.3. The rolling and pitching model are given in Figure 4.4.

In Figure 4.3, K_i and C_i ($i = f, r$) denote the spring stiffness and the damping coefficient of the suspension system related to each wheel, $m_{i,tire}$ ($i = f, r$) denotes the mass of the front and rear tire, respectively, m^s is the sprung mass, and $\dot{\phi}$ and $\dot{\theta}$ are the rolling and pitching rate about X_B and Y_B axis, respectively.

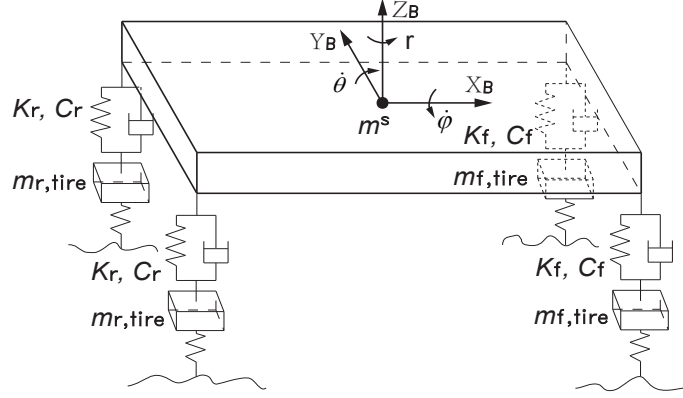


Figure 4.3: Riding model.

Figure 4.4(a) shows the rolling motion arising from the lateral acceleration and the gravity center offset from the rolling center. The parameters h^s and h^c are the heights of the sprung mass center and the rolling center (CR), respectively. Figure 4.4(b) shows the pitching motion arising from the longitudinal acceleration and the gravity center offset from the pitching center (CP) that is assumed to be on the ground. The dynamical equations of the vertical, rolling and pitching motion of the sprung mass are given by

$$\begin{aligned} \dot{V}_z^s = & \left(-2(K_f + K_r)\theta - 2(C_f + C_r)V_z^s + 2(\ell_f K_f - \ell_r K_r)\phi \right. \\ & \left. + 2(\ell_f C_f - \ell_r C_r)\dot{\theta} \right) / m^s, \end{aligned} \quad (4.5a)$$

$$\begin{aligned} \ddot{\theta} = & \left(2(\ell_f K_f - \ell_r K_r)z^s + 2(\ell_f C_f - \ell_r C_r)V_z^s - 2(\ell_f^2 K_f + \ell_r^2 K_r)\theta \right. \\ & \left. - 2(\ell_f^2 C_f + \ell_r^2 C_r)\dot{\theta} + m^s g h^s \sin \theta + m^s a_x^s h_s \cos \theta \right) / I_y^p, \end{aligned} \quad (4.5b)$$

$$\begin{aligned} \ddot{\phi} = & \left(-w_f^2 K_f \phi / 2 - w_f^2 C_f \dot{\phi} / 2 - w_r^2 K_r \phi / 2 - w_r^2 C_r \dot{\phi} / 2 \right. \\ & \left. + m^s g (h^s - h^c) \sin \phi + m^s a_y^s (h^s - h^c) \cos \phi \right) / I_x^R, \end{aligned} \quad (4.5c)$$

where w_i ($i = f, r$) denote the front and rear track, respectively; a_x^s and a_y^s are the longitudinal and lateral acceleration of the sprung mass center in the body-fixed frame, and I_x^R and I_y^p are the moments of inertia of the sprung mass about the rolling axis and the pitching axis, respectively.

4.2.4 Tire Force Model

The tire slip is defined by the non-dimensional relative velocity of each tire with respect to the road, which is given by

$$s_{ijx} = \frac{V_{ijx} - \omega_{ijx} R_j}{\omega_{ijx} R_j}, \quad s_{ijy} = \frac{V_{ijy}}{\omega_{ijx} R_j}, \quad (4.6)$$

where $i = L, R$ and $j = F, R$. V_{ijk} ($k = x, y$) is the tire frame component of the vehicle velocity of each tire. The total slip of each tire is defined by $s_{ij} = \sqrt{s_{ijx}^2 + s_{ijy}^2}$. The total

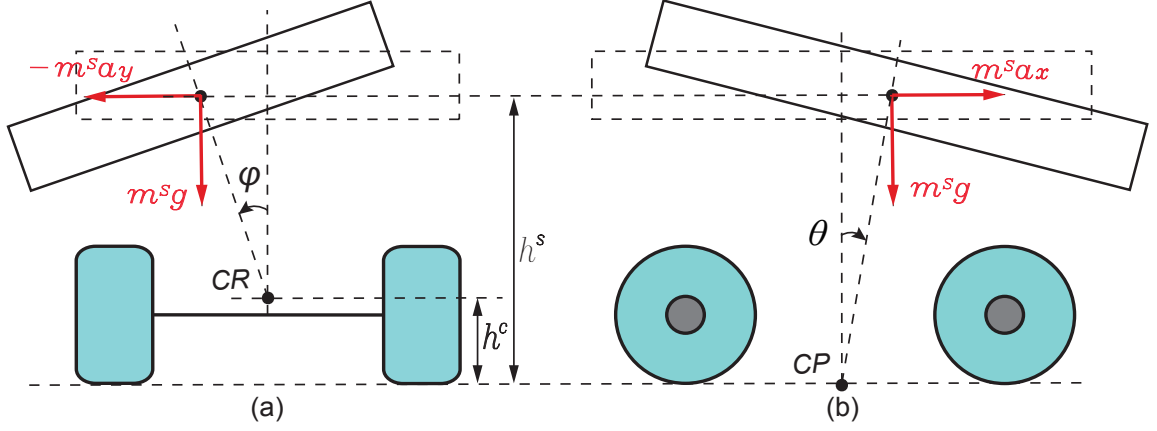


Figure 4.4: Rolling and pitching model.

friction coefficient related to each tire is calculated using Pacejka's "magic formula" (MF) as follows [141, 84, 137]:

$$\mu_{ij} = D \sin \left(\text{Catan} \left(B S_E - E (B S_E - \text{atan} S_E) \right) \right) + S_v \quad (4.7)$$

where B, C, D, E are the stiffness, shape, peak and curvature factors, respectively; $S_E = s_{ij} - S_h$, where S_h is the horizontal shift. S_v is the vertical shift. The MF is sketched in Figure 4.5.

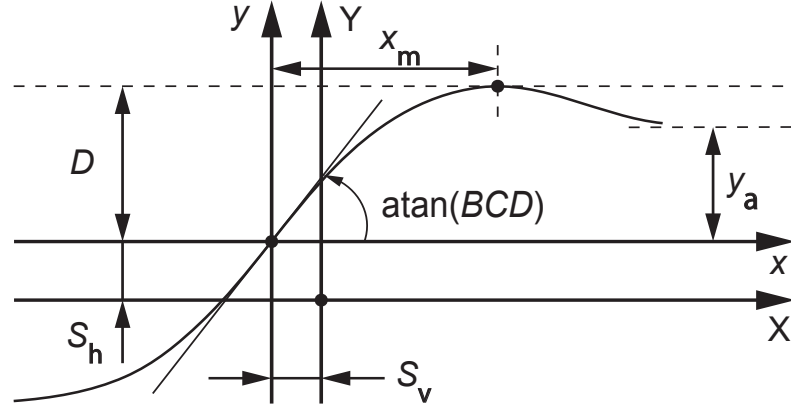


Figure 4.5: The magic formula.

The tire friction force components are given by

$$f_{ijk} = -\frac{s_{ijk}}{s_{ij}} \mu_{ij} f_{ijz}, \quad i = L, R; \quad j = F, R; \quad k = x, y. \quad (4.8)$$

where f_{ijz} is the normal load on the corresponding tire and can be calculated following [84, 140]. We do not show the details on the calculation for the normal load f_{ijz} due to lack of space.

4.3 Parameter Estimation

In the following, we use data collected from a fifth-scale Auto-Rally vehicle (see Figure 4.6) and estimate the unknown parameters of all three of the vehicle models. Table 4.1 (bottom) summarizes the unknown parameters to be estimated for the three different vehicle models.



Figure 4.6: The test track and the Auto-Rally vehicle model.

The fifth-scale Auto-Rally vehicle is driven by two rear wheels, and the top speed can reach up to 27 [m/s]. The size of the whole model measures about $1 \times 0.6 \times 0.4$ [m³]. The other known parameters of the Auto-Rally vehicle model are given in Table 4.1 (top). The perimeter of the centerline of the test track is about 63 [m], and the width is about 3.4 [m]. More detailed descriptions about the vehicle model and the test track can be found in [142].

We use a joint-state UKF algorithm to estimate the unknown parameters of the system. For the system given in (2.1a)-(2.1b), we introduce the following dynamics for the parameter vector p ,

$$p_{k+1} = p_k + w_k^p, \quad (4.9)$$

where $w_k^p \sim N(q^p, Q^p)$ is Gaussian process noise. We define the augmented state $x^a = [x^T, p^T]^T$. It then follows from (2.1a)-(2.1b) and (4.9) that

$$x_{k+1}^a = F(x_k^a, u_k) + w_k^a, \quad (4.10a)$$

$$y_k = H(x_k^a, u_k) + v_k, \quad (4.10b)$$

where $w_k^a = [w_k^T, (w_k^p)^T]^T$.

It is noticed that \hat{R} and \hat{Q} in (2.13) and (2.20) may become negative definite during the process of the implementation (this is also mentioned in [95]). This dissertation calculates the nearest positive definite matrices of \hat{R} or \hat{Q} when negative eigenvalues of \hat{R} or \hat{Q} are observed, such that a symmetric positive definite matrix nearest to \hat{R} or \hat{Q} in terms of the Frobenius norm can be obtained [143, 144].

It is worth mentioning that the artificial Gaussian process noise w_k^p in (4.9) is used to change the parameter p when the UKF is working. However, if the value of w_k^p is large,

the parameter p will be changed by a large amount at each time step. This condition may further cause the filter to diverge, since the parameterized vehicle models in Section 4.2 are sensitive to p and may thus become unstable for unreasonable values of p . We addressed this problem by rescaling the diagonal entries of Q^p to be some small positive values at each time step. Other discussions on the numerical instability problems of the UKF can be found in [95, 145, 144].

4.4 Results and Discussion

In this section we implement the proposed filter, show and validate the results from the parameter estimations using Algorithms I and II, respectively.

4.4.1 Standard UKF

We first implemented the standard joint-state UKF using the three different vehicle models in Section 4.2, respectively. The hyperparameters of the filter are critical for the filter design, especially the process noise covariance Q [124, 125]. In this section, we tune the diagonal elements of these matrices recursively, until the parameterized vehicle model shows satisfactory simulation results that have good agreement with data.

We selected 113 seconds of experimental data of the Auto-Rally vehicle [142]. The first 100 seconds data were used to tune the hyperparameters and estimate the vehicle parameters, and the remaining 13 seconds (a complete cycle around the testing track) were used to validate the results. Figure 4.7 shows the estimates for several selected states of the system for the single-track model. It can be seen that the estimates of the states agree well with the data. The results for the other vehicle models were similar and hence are omitted.

Next, we validated the estimated parameters in simulation. This was done in order to ensure that the parameters we obtained were able to satisfactorily reproduce the data, hence accurately predicting the vehicle's motion in practical applications. Figure 4.8 shows the simulated trajectories for different vehicle models configured with the estimated parameters. The results in Figure 4.8 indicate that the larger the number of degrees of freedom (DoF) of the model, the more accurate the results and the better the

Table 4.1: Known / Unknown Vehicle model parameters.

$m[kg]$	21.5	total mass	$m^s[kg]$	18.03	sprung mass
$m_f[kg]$	0.84	front wheel mass	$m_r[kg]$	0.89	rear wheel mass
$w_f[m]$	0.44	front track	$w_r[m]$	0.46	rear track
$L[m]$	0.57	wheel base	$R[m]$	0.095	wheel radius

Tire forces model	B, C, D, E, S_h, S_v
Single/Double-track model	I_z, ℓ_f, h, g_s^*
Full vehicle model	$I_z, \ell_f, h, g_s^*, C_D, I_x^R, I_y^P, K_f, K_r, C_f, C_r, h^c$

* g_s is the gear ratio defined by the steering command divided by δ

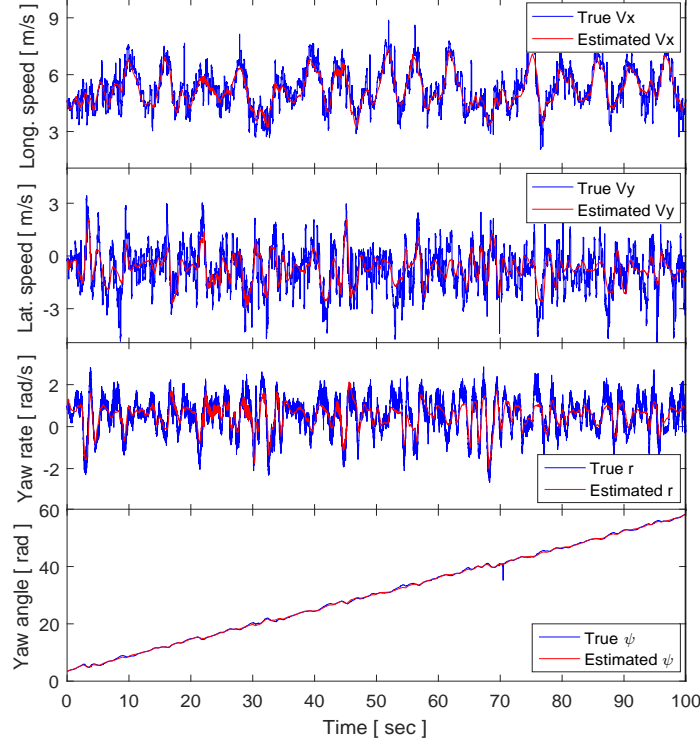


Figure 4.7: State estimation for the single-track model using JUKF.

agreement with the experimental data. This should be expected, although it typically takes more time to estimate the parameters and perform the prediction than the lower dof vehicle models.

The process of manually tuning the hyperparameters of the UKF one-by-one until we achieve good performance is time consuming, and can be done only off-line.

4.4.2 Adaptive Limited Memory UKF

Instead of tuning the noise, we implemented the ALM-UKF to find the suboptimal estimation of the noise statistics on-line, during which the augmented-state and the noise are estimated simultaneously. Both simulation data collected using CarSim and experimental data collected with the Auto-Rally vehicle were used to validate Algorithm II. The noise samples at each time step k are from the estimation based on the last 10 seconds of data (defined by N and M in Algorithm II).

The estimation for only the states (i.e., the velocities, yaw angle and positions) is not difficult. We only show the estimation results of the unknown vehicle parameters. We implemented the adaptive limited memory joint-state UKF (ALM-JUKF) to estimate a full vehicle model's parameters using the Auto-Rally experimental data. Figure 4.9 shows the time trajectories of several parameters (see Table 4.1) during the estimation process, where all the parameters converge fast and get stabilized after about 20 seconds.

Since CarSim provides a complete full-scale vehicle model and data from the sim-

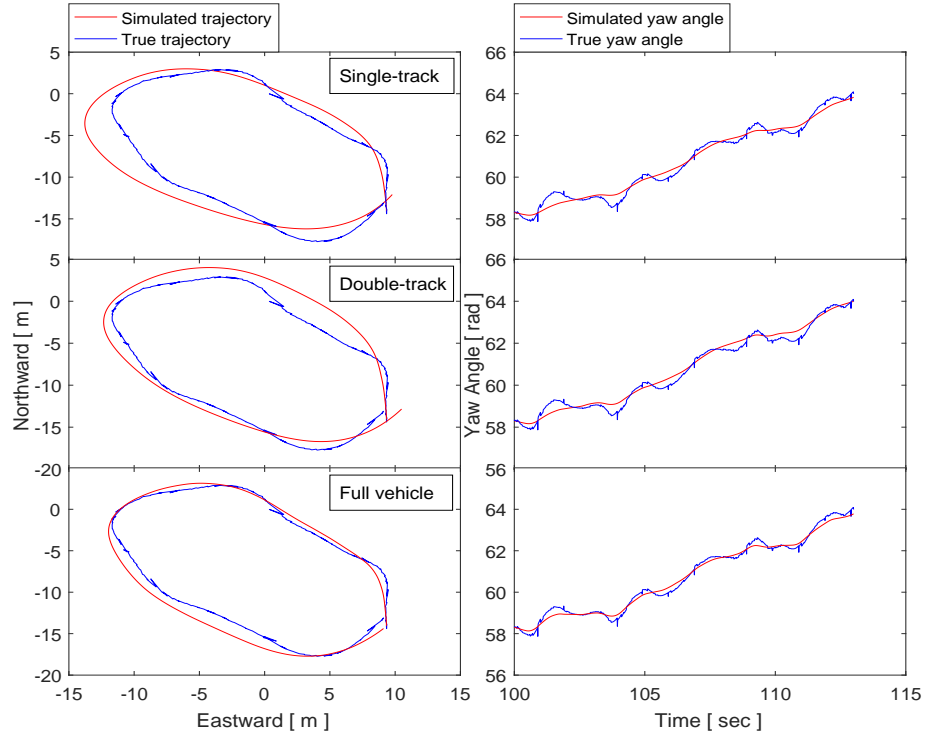


Figure 4.8: Simulation results of the estimated vehicle models using standard UKF.

ulation using CarSim show little irregular noise, the ALM-JUKF was also implemented using simulation data for validation purposes. In Figure 4.10, we show the estimated vehicle parameters corresponding to the CarSim vehicle model and the Auto-Rally vehicle model from simulations. We simulated a full vehicle model using the estimated parameters and compared the model output with data. It can be seen that the identified vehicle model can satisfactorily reproduce the data, especially when the system uses simulated data, as expected. The experimental data collected using the Auto-Rally vehicle show obvious non-Gaussian noise which may have some effect on the estimation process. Compared with the results in Figure 4.8, the simulated trajectories of the

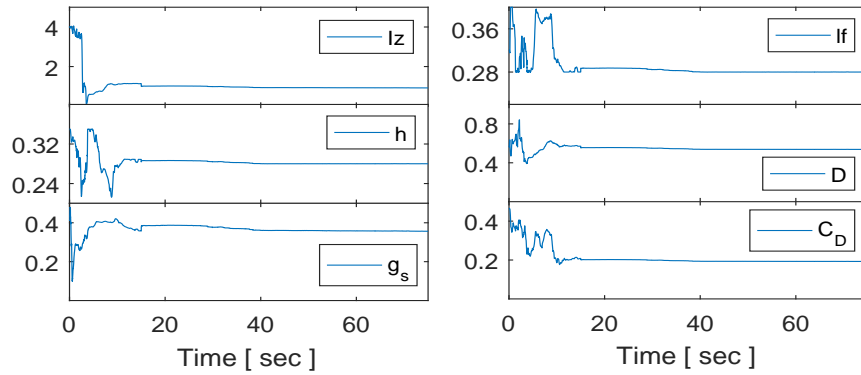


Figure 4.9: Convergence of the vehicle parameters along with the estimation process.

Auto-Rally vehicle in Figure 4.10 show larger deviation from the data. The reason may

be that Algorithm I was intended to tune the estimation of the noise statistics to be optimal (in some degree), but Algorithm II was using a suboptimal estimator for the noise statistics. Algorithm I may obtain a better estimation of the noise statistics, as long as it keeps tuning the noise. The advantages of Algorithm II are, of course, that it is more efficient and can work on-line. We also expect that Algorithm II is especially useful for time-varying parameters estimation problems (i.e., identification of a linear parameter varying (LPV) driver model), since the algorithms that work only off-line cannot capture the real-time change of the parameters due to its low implementation speed.

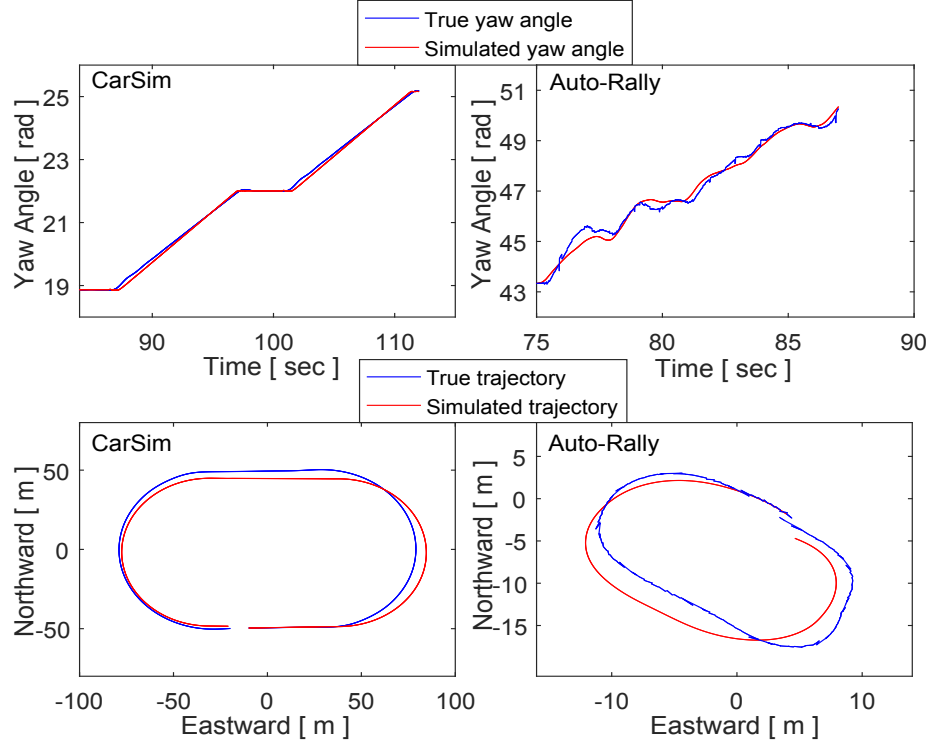


Figure 4.10: Simulation results of the estimated vehicle models using ALM-JUKE.

4.4.3 Experiments

Next, we compare the performance of the above vehicle models, namely, the single-track model, the double-track model and the full vehicle model, with a neural-network (NN) model in a vehicle behavior prediction task. To this end, we implement the model predictive path integral (MPPI) controller [142] using different models. We show only the results corresponding to the single-track model and the NN model. The double-track model and the full vehicle model have been shown to be able to provide better prediction result than the single track model (see Figure 4.8). The description of the test track and the Auto-Rally vehicle platform are found in Section 7.8.2.

Figure 4.11 shows the color map of the vehicle's speed profile along the trajectories. The maximum speed corresponding to the NN model is about 7.7 [m/s], while the maximum speed corresponding the single-track model is about 7.8 [m/s], which are very

close. The maximum speeds achieved using both the two models are below 8 [m/s], while the target speed was setup to 12 [m/s] for both cases in the experiments. The Auto-Rally vehicle cannot reach the target speed in the experiments. The reason may be that the friction condition of the track is not good enough during the experiments. The performance of the two models are close.

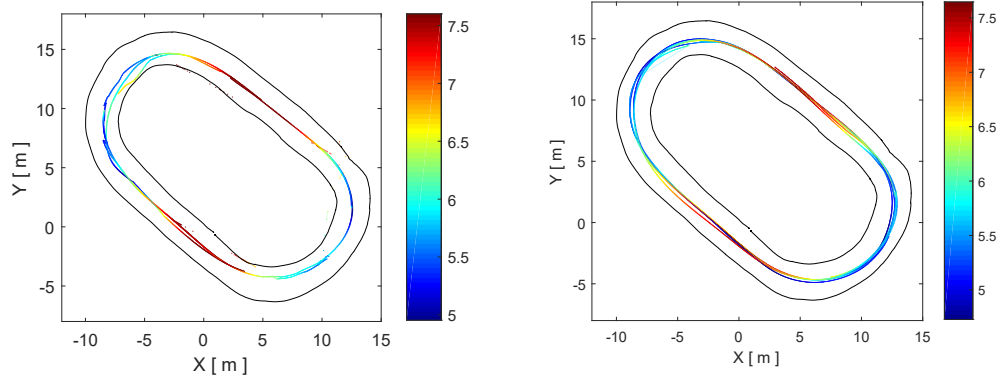


Figure 4.11: MPPI implementation using a neural-network model (left) and a single-track model (right).

We also plot the slip angle in a typical round for both the two cases, as shown in Figure 4.12. Sometimes the physical model is able to predict the vehicle's behavior more accurately than an NN model, since the performance of an NN model mainly depends on its learning set that was used to train the model parameters.

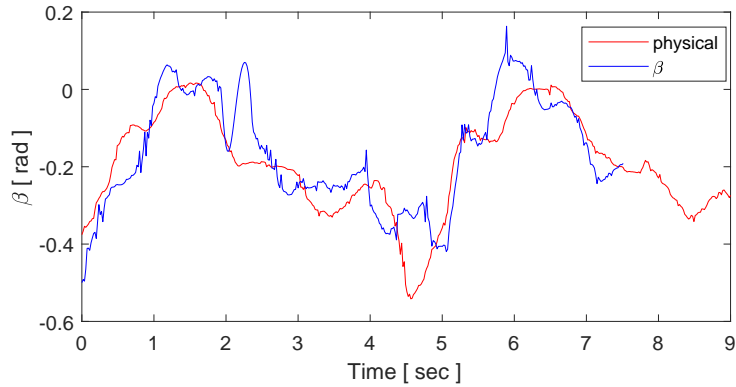


Figure 4.12: Sideslip angle of Auto-Rally.

4.5 Conclusion

In this chapter we introduced three vehicle models, namely, a single-track model, a double-track model and a full vehicle model, and we estimated the model parameters using the joint-state UKF algorithms based on both simulation and experimental data. By tuning the noise statistics of the standard joint-state UKF, satisfactory estimates of the model parameters can be obtained, but the tuning process is time consuming and

hence can only be implemented off-line. In contrast, we implement an adaptive limited memory joint-state UKF algorithm (ALM-JUKF), which estimates the system state, model parameters and the Kalman filter hyperparameters related to the noise simultaneously, hence making possible to provide on-line estimates of the model parameters. The algorithm was validated with both CarSim simulation data and experimental data from a fifth-scale Auto-Rally vehicle.

CHAPTER 5

HIGHWAY TRAFFIC MODELING AND OPTIMAL DECISION MAKING

5.1 Introduction

Autonomous vehicles promise to improve traffic safety while, at the same time, increase fuel efficiency and reduce congestion. They represent the main trend in future intelligent transportation systems. In order to determine the optimal driving strategy for autonomous vehicles in traffic, different techniques have been developed for optimal decision making and path planning [66, 48, 73, 36, 67, 46]. Application of reinforcement learning requires knowledge of the reward function, which needs to be carefully designed. An alternative is to learn the optimal driving strategy using demonstrations of the desired driving behaviors.

Abbeel and Ng [77] used a driving simulator to collect two minutes of driving data from an expert driver, and assumed that the reward function of this expert driver is a linear combination of a number of known features. In order to recover the reward function and the expert driving policy, they proposed a max-margin algorithm along with a projection algorithm to solve the inverse reinforcement learning problem. Although one can approximately recover expert driving behaviors using this approach, the matching between the optimal policy/reward and the features is ambiguous, as indicated by Ziebart and his colleagues [78]. In order to address this ambiguity, Ziebart introduced the maximum entropy principle (MEP) to uniquely match the rewards with the features. He proposed the maximum entropy inverse reinforcement learning (MaxEnt IRL) algorithm [78, 79], which was shown to be computationally efficient in [78]. It was implemented on a routing problem (mission planning). The researchers in [80, 81, 82, 83] developed different versions of the MaxEnt IRL algorithm based on [78, 79]. Among these, the authors of [82] formulated the maximum entropy inverse reinforcement learning problem using a deep neural network (DNN) to represent the unknown reward function. All the above formulations of the MaxEnt IRL problem require complete knowledge of the environment dynamics, and they all employ a state reward instead of a state-action reward so that they cannot learn a complicated driving behavior showing preference on certain actions. This chapter focuses on the problem of planning for autonomous vehicles in traffic. Specifically, we wish to reproduce the decision making of an expert driver, that is, we wish to duplicate the optimal driving strategy involving several typical driver actions such as lane-shifting, lane and speed maintaining, accelerating and braking, by also considering the stochastic driving behaviors of the environmental vehicles in traffic.

The contribution of this chapter is summarized as follows: 1) We propose a new MDP model to represent the stochastic behaviors of the environmental vehicles in highway traffic. This model differs from previous similar MDP models [146, 147, 148, 149] in the sense that we take the road geometry into consideration in order to compare and ana-

lyze different driving strategies during cornering. Another advantage is that the model is easily scalable to have more vehicles and more lanes in traffic. Unlike the MDP models [148, 149] that need to discretize the velocity of each vehicle in traffic, which, consequently, tend to make the problem have a large state space, we remove the velocities of the vehicles from the MDP model and consider them either in the perception layer or in the control layer. The optimal control policy for the proposed MDP is solved using both reinforcement learning (RL) and inverse reinforcement learning (IRL). 2) We generalize the formulation of the MaxEnt IRL by using a reward function in the form of a linear combination of the parameterized features, and we show, for the first time, that the reward function in the MaxEnt IRL formulation can take any nonlinear form. Previous results of MaxEnt IRL either assumed that the reward function can be represented using a linear combination of fixed features [78, 79, 80, 81, 83], or directly used a deep neural network (DNN) to represent the reward function in the MaxEnt IRL formulation without clarifying the relation between the DNN and the parameterized feature functions [82]. 3) We propose three new MaxEnt deep IRL algorithms to solve the model-free MDP problem. Although several researchers have proposed different versions of MaxEnt IRL algorithms [78, 79, 80, 81, 82, 83], these algorithms cannot be used to solve a model-free problem. Specifically, we use a deep neural network to approximate the state-action reward, instead of the state reward, as in most of existing MaxEnt IRL formulations.

The rest of the chapter is organized as follows: Section 5.2 introduces the traffic model using a stochastic MDP. Section 5.3 designs the reward function and solves the MDP problem using Q-learning. Section 5.4 introduces the maximum entropy principle and formulates the inverse optimal control problem. Section 5.5 summarizes and refines the MaxEnt deep IRL algorithms, and Section 5.6 implements both RL and IRL algorithms, and analyzes the results. Finally, Section 5.7 summarizes the results of this study.

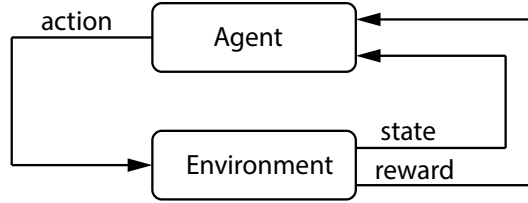


Figure 5.1: The agent-environment interaction.

5.2 Traffic Modeling

In this section we first introduce a Markov decision process (MDP) to model the interaction between the autonomous vehicle and the surrounding vehicles in traffic. In the following sections we use both RL and IRL techniques to solve the MDP problem for the optimal policy that achieves the desired driving behaviors. We start with a brief summary of MDPs.

5.2.1 Markov Decision Process

Markov decision processes (MDPs) are used in a wide area of applications such as robotics, economics, manufacturing and automatic control. An MDP is a mathematical framework that probabilistically models the interaction between an agent and the environment, pioneered by the work of Bellman [150]. The agent is assumed to be a learner or decision maker, who interacts with the environment[76]. It receives a reward and a representation of the environment's state at each time step, and exerts an action on the environment that may change its future state. This interaction between the agent and the environment is shown in Figure 5.1.

A typical MDP is represented using a 6-tuple $(S, A, \mathcal{T}, \gamma, D, R)$, where S is a (finite) set of possible states that represent a dynamic environment, A is a (finite) set of available actions that the agent can select at a certain state¹, \mathcal{T} is the state transition probability matrix that provides the probability of the system transition between every pair of the states, $\gamma \in [0, 1)$ is the discount rate that guarantees the convergence of total returns, D is the initial-state distribution, and R is the reward function that specifies the reward gained at a specific state by taking a certain action.

MDPs assume that the effect of taking an action at a given state only depends on the present state-action pair, and not on the previous states and actions, that is,

$$\mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = \mathbb{P}(s_{t+1}|s_t, a_t). \quad (5.1)$$

Equation (5.1) is called Markov property[151].

The core problem of an MDP is to find a policy π for the agent, where the policy $\pi : S \rightarrow A$ specifies the action to take at the current state s_t . The goal is to find the optimal policy π^* that maximizes the cumulative discounted reward over an infinite horizon:

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \right], \quad (5.2)$$

where the term $R(s_t, \pi(s_t))$ represents the reward the agent receives by taking an action determined by policy π at the present state s_t . Given a policy π , the MDP in (5.1) is reduced to a Markov chain with transition probabilities \mathbb{P}^{π} , given by

$$\mathbb{P}^{\pi}(s_{t+1}|s_t) = \mathbb{P}(s_{t+1}|s_t, \pi(s_t)). \quad (5.3)$$

5.2.2 System Modeling

The MDP to be used to model the traffic is based on the following observations. Consider a typical scenario of traffic on a multi-lane road as shown in Figure 5.2. Each vehicle moves in the middle of each lane with the average speed of the traffic flow.

Let us now consider the driving behavior of the blue vehicle in the middle of the red rectangular shown in Figure 5.2. There are several actions the blue vehicle can take.

¹WLOG we will assume that all actions are available in each state.

For instance, it can maintain its current speed, accelerate or brake to occupy the vacant positions ahead of it or behind it, or move to the left or to the right lane if there is no chance for a collision. Assuming that each driver intends to maximize a certain reward function, if one can obtain the reward function of an “expert” driver by either constructing it manually or from observing real driving data, one should be able to reproduce this expert driver’s behaviors using reinforcement learning techniques [76].

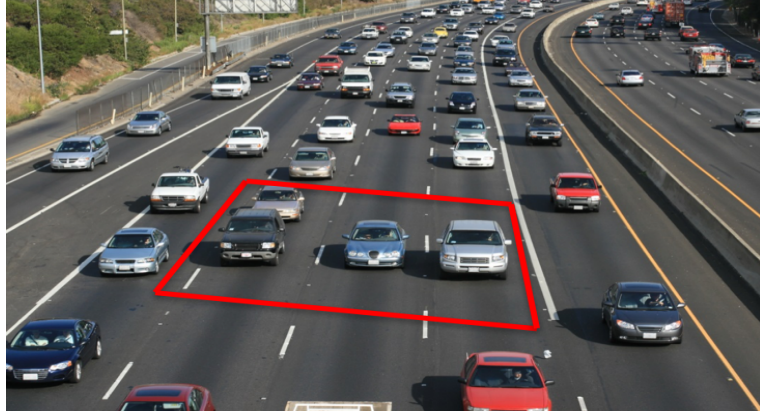


Figure 5.2: The traffic on multi-lane road.

In the following, we designate the vehicle we want to control as the host vehicle (HV), and all remaining vehicles in traffic as the environmental vehicles (EVs). We assume that the drivers of different vehicles do not communicate with one another, and also that the vehicles do not share data with each other. Hence, the MDP system has only a single actively controlled agent. The available action set for each vehicle in traffic is given by $A \triangleq \{\text{“maintain”, “accelerate”, “brake”, “left-turn”, “right-turn”}\}$.

State Definition

By considering the positions of the HV, and the number and positions of the EVs around the HV, we define the state of the MDP as shown in Figure 5.3.

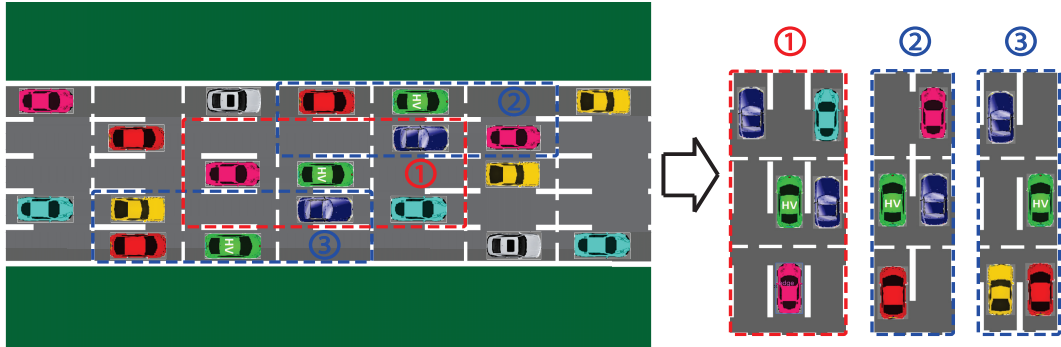


Figure 5.3: The cells and the definition of the state: ① 9-cell internal-lane state, ② 6-cell left-boundary state and ③ 6-cell right-boundary state

In Figure 5.3, we use the white dashed lines to divide the road into small cells and use the green vehicle to denote the HV. The states of the MDP represent either of the three conditions shown in Figure 5.3: 1) the HV is in the middle lane of the road, where we use nine cells to represent the state, and 2) and 3) where the HV is next to the road boundaries and we use six cells to represent the current state. Taking all possible combinations into account, the number of the internal-lane states is $2^8 = 256$, and the number of the left(right)-boundary states is $2^5 = 32$. Hence the total number of the states of the MDP is $256 + 2 \times 32 = 320$. Note that the approach can be easily extended to highways with any number of lanes and vehicles ².



Figure 5.4: Overtaking during cornering.

Figure 5.4 shows a possible overtaking behavior of the HV (green car) during a left-turn corner. The HV driver may prefer overtaking the pink car in front from the left rather than from the right. In order to investigate the effect of the road geometry on the observed driving behaviors of different drivers, in this work we take the road curvature into account and consider three kinds of roads, namely, left-turn, right-turn, and straight roads. The total number of the states is therefore $320 \times 3 = 960$. It is worth mentioning that, although we only consider three kinds of road geometries, one can, similarly, divide the road characteristics into more classes, as needed. For instance, one could also take into account different slopes of the roads, such as downhill, uphill, flat roads etc.

State Transitions

We want to model the state transition process by mimicking the traffic in real world scenarios. To this end, we make the following assumptions: 1) the number of lanes n is free and greater than equal to two ($n \geq 2$), 2) the number of EVs N is free but no larger than eight, given the cell geometry of Figure 5.3 ($0 \leq N \leq 8$), 3) the EVs have their

²The traffic model is also possible to be used for modeling urban traffic by adjusting the state definition. For instance, the cell size may be defined to change with the size of each EV and its velocity relative to the HV.

own policies that may be different from the HV, 4) the EVs take a random action, 5) no collision arises from the actions of the EVs, and 6) each vehicle takes a single action at each time step.

The state transition procedure from the current state s_t to the next state s_{t+1} is given in two steps: First, the HV observes the current state s_t and selects an action $\pi(s_t)$ following its current policy. Second, the EVs respond to the action of the HV, and take an action following their own policies in a random sequence. The new positions of the HV and the EVs around the HV define the next state s_{t+1} . This state transition process is demonstrated in Figure 5.5.

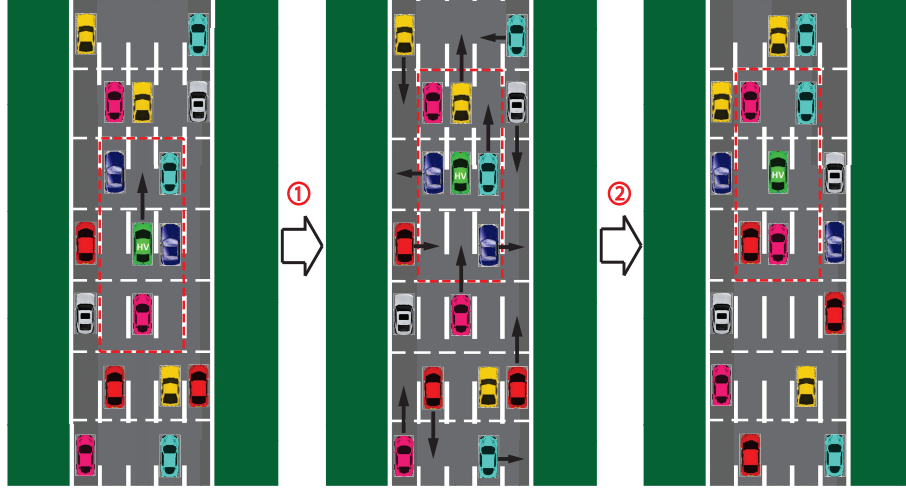


Figure 5.5: State transition process.

The current state s_t is defined using the nine cells in the red rectangular on the left graph in Figure 5.5. Based on s_t , the HV may brake or switch to the right lane but these actions will result in an collision. The available safe actions of the HV are maintaining, accelerating and switching to the left lane. For instance, suppose that the HV accelerates and occupies the cell in front of it. As a consequence, the red rectangular also moves since the EVs surrounding the HV change. Next, all EVs respond to the action of the HV and take an action following a certain policy in a random order (see Algorithm 7). The next state s_{t+1} is obtained after all vehicles complete their actions (see the red rectangular on the right graph in Figure 5.5).

5.2.3 Dynamic Cell

The traffic model in Section 5.2.2 does not consider different vehicle sizes (i.e., truck, sedan etc.), and the vehicle velocities in real traffic are changing. Instead of discretizing the velocity of each vehicle in traffic [148, 149], which, consequentially, tends to make the problem have a large state space, we introduce a new layer, namely, the “dynamic cell” layer, into the control architecture.

We assume that each vehicle intends to keep a safe distance from the front vehicle

depending on its current longitudinal velocity. The length of the dynamic cell for the host vehicle is therefore defined by

$$L_{HV} = \Delta T \times V_{HV} + \ell_{HV} \quad (5.4)$$

where ΔT is the time constant that defines the minimum distance one wants to keep from the front vehicle, and ℓ_{HV} is the chassis length of the host vehicle. When the host vehicle is static, L_{HV} equals to the length of the vehicle itself. The 9-cell state of the traffic MDP model is therefore defined with a specified size as shown in the first picture in Fig. 5.6.

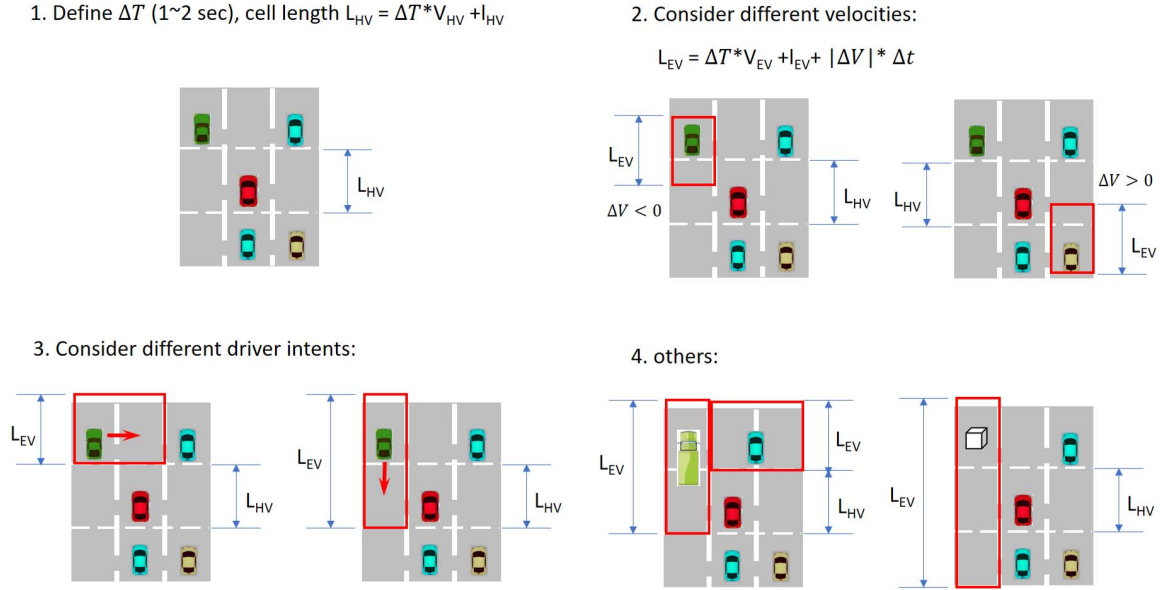


Figure 5.6: Dynamic cells.

Similar to (5.4), the definition of the cell length for the TVs requires a modification term depending on the relative velocity of the TV to the HV (longitudinal velocity difference),

$$L_{TV} = \Delta T \times V_{TV} + \ell_{TV} + |\Delta V| \Delta t, \quad (5.5)$$

where V_{TV} and ℓ_{TV} are the longitudinal velocity and the chassis length of the TV, respectively and Δt is the time constant that determines how much the TV approaches the HV in the next step. We can look at the second picture in Fig. 5.6 for instance. If the green TV is slower than the HV, the cell this TV occupies will move backward with a distance $|\Delta V| \Delta t$ due to the relative velocity ΔV , and hence it overlaps on the cell on the left of the HV. As a consequence, the left cell of the HV is not available for the lane-switching action. Some special cases, such as when there is a truck or some static obstacle in traffic, can also be handled by dynamically changing the cells. One can see Fig. 5.6 for the graphical explanation.

The cell width for either the HV or the TVs is naturally defined using the lane width.

The signal light indicates the driver's intent and the HV is able to predict the motion of a TV by its signal light (i.e., the left/right turn signals and the braking light) and avoid taking dangerous actions. One can change the cell width of a TV according to its signal lights to indicate the area this TV occupies.

5.3 Reinforcement Learning

In order to obtain the desired driving policy for the HV, in this section we design the reward function and use reinforcement learning techniques to solve the MDP problem formulated in Section 5.2.

5.3.1 Reinforcement Learning Algorithms

The main methods used in reinforcement learning can be classified into two categories, namely, tabular solution methods and function approximation methods. The tabular solution methods are suitable for solving MDP problems with a finite (small) number of states and actions. Such methods mainly include dynamic programming, Monte Carlo and temporal-difference learning [76].

Classical dynamic programming algorithms use the value function to organize the search for the optimal policies. Such algorithms include value iteration and policy iteration methods. These algorithms require perfect knowledge of the environment, and cannot be easily applied to problems having continuous state and action spaces. Unlike dynamic programming, Monte Carlo does not require complete knowledge of the environment, but only the agent's experience, namely, the sample sequences of the states, actions and rewards from actual or simulated interactions of the agent with the environment. Monte Carlo methods are based on averaging sample returns in an episode-by-episode manner, which means that the learning of the values and the corresponding policies is performed only upon completion of each episode. Hence, Monte Carlo methods cannot update the values and policies in an on-line fashion.

Temporal-difference learning is a combination of the ideas from dynamic programming and Monte Carlo methods. The most obvious advantage of temporal-difference learning over Monte Carlo is that it can be naturally implemented in an on-line fashion. Unlike Monte Carlo methods, one does not need to wait till the completion of every episode to receive the return. The most obvious advantage of temporal-difference learning over dynamic programming is that it does not require full knowledge of the environment and hence it can be implemented without a model. The two main temporal-difference learning algorithms are Sarsa and Q-learning[152, 76].

The main difference between Sarsa and Q-learning is the future (state-action) values they refer to in order to update the current (state-action) values. Sarsa stands for "state-action-reward-state-action" and uses the value of the real action the agent takes in the next step following the current control policy in order to update the value of the action at the present state. Q-learning explores the maximum possible action value the agent can have in the next step, and uses this value to update the value of the action at the

present state. Hence, Sarsa is an on-policy algorithm, while Q-learning is an off-policy algorithm. For many problems, both Sarsa and Q-learning are able to learn a good policy with good performance. Q-learning can potentially provide a better policy where death can be easily caused in each episode using the ϵ -greedy policy in an on-policy method [152]. The term “death” indicates the termination of an episode caused by the agent arriving at certain states (i.e., the goal state). However, Q-learning is also known to diverge in certain cases where function approximation is used [152, 153]. More details on the tabular solution methods can be found in Chapters 4-6 of [76].

Function approximation methods are used to address large or continuous state space problems, where one may use a series of (nonlinear) functions to represent the values, policies and rewards. Theoretically, all methods used in the area of supervised learning are possible to use in reinforcement learning as function approximators, such as artificial neural network [154], naive Bayes [155], Gaussian processes [156], or support vector machines [157].

Since the MDP model in Section 5.2 has a finite number of states and actions and assumes that the agent cannot predict the behavior of the EVs, we prefer to use a tabular, model-free method to solve the corresponding optimal control problem in (5.2). In the following section, we first define the reward function, and then use Q-learning to learn the optimal policy that maximizes the cumulative discounted future rewards.

5.3.2 Reward Function

The design of the reward function is a difficult task, since the driver behavior is hard to characterize and the real reward function is unknown. The reward function also differs from driver to driver and it may be even change with time. A widely used approach to design the reward function is to represent it as a function of some manually chosen features. These features depend on the action of the agent and the state of the environment. We use a linear combination of the features to represent the reward function [77, 78, 79, 158, 80]:

$$R(s, a) = w^T \Phi(s, a), \quad (5.6)$$

where w is the weight vector, and $\Phi(s, a)$ is the feature vector with each component representing a single feature point in the state-action space. Possible choices of feature points may be the binary values indicating whether a certain argument is true or not. In this work we define the features in $\Phi(s, a)$ as follows:

- 1) Action features. The driver may prefer taking certain actions than others if he receives a higher reward from these actions.
- 2) Position of the HV. It indicates if the HV is driving next to the road boundaries. The driver may prefer to drive in different lanes, depending on the road geometry.
- 3) Overtaking strategy. This feature is used to achieve different overtaking behaviors of drivers during cornering. The driver may have a different preference in regards to overtaking the car in front either from the left or from the right.

4) Tailgating. The value of this feature is “true” if the HV is behind an EV and “false” otherwise.

5) Collision incident. Collision occurs if the HV and a EV appear in the same cell.

One can design the weight vector w to encourage or penalize certain features using the given reward function, and then use reinforcement learning to learn the corresponding optimal policy by maximizing the total reward. Another idea is to design the reward function using a parameterized function approximator such as a Gaussian process [158, 80] or a DNN [82]. The parameters of the function approximator are hard to design manually since they may not be directly related to features that have clear physical meaning, and hence they can only be learned from data. This approach will be discussed in Section 5.5.

5.3.3 Q-Learning

Q-learning was first introduced in Watkins’s PhD thesis in 1989 [159]. In 1992 Watkins and Dayan proved that Q-learning converges to the optimum action values with probability 1 if all actions are repeatedly sampled in all states [160]. Different variants of Q-learning were developed to solve various reinforcement learning problems, such as double Q-learning that reduces overestimation [161], deep Q-learning that uses a deep neural-network to represent the value function for large state space problems [162], fuzzy Q-learning that uses fuzzy logic rules to interpret and refine the imprecise environment knowledge [163], and minimax-Q [164], Nash-Q [165], correlated-Q [166] and friend-or-foe-Q [167] that solve multi-agent reinforcement learning problems.

Next, we briefly introduce the basic Q-learning algorithm. To this end, we need to introduce two important concepts used extensively in the reinforcement learning literature, namely, the state value and the state-action value (also referred to as the action value). The value of a state s_t under policy π is denoted as $V^\pi(s_t)$, which indicates the expected discounted cumulative reward starting at state s_t and then following policy π , that is,

$$V^\pi(s_t) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| s_t, \pi \right], \quad (5.7)$$

where γ is the discount rate, and

$$R_t \triangleq R(s_t, a_t) = \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1} | s_t, a_t) R(s_t, a_t, s_{t+1}), \quad (5.8)$$

stands for the immediate reward the agent receives by taking action a_t at present state s_t , and where the term $R'(s_t, a_t, s_{t+1})$ represents the reward the agent receives by taking action a_t at present state s_t to obtain the next state s_{t+1} (see Section 5.5.1 for more discussion).

The values of two sequential states of the MDP are related and satisfy the following

equation,

$$\begin{aligned} V^\pi(s_t) &= \mathbb{E} \left[R_t + \gamma V^\pi(s_{t+1}) \middle| s_t, \pi \right] \\ &= \sum_{a_t \in A} \pi(s_t, a_t) \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1} | s_t, a_t) \left(R'(s_t, a_t, s_{t+1}) + \gamma V^\pi(s_{t+1}) \right). \end{aligned} \quad (5.9)$$

Equation (5.9) is called the Bellman evaluation equation. The optimal policy π^* maximizes the associated value function at each state, which, mathematically, can be determined by solving the following problem,

$$\pi^* = \arg \max_{\pi} V^\pi(s), \quad \forall s \in S. \quad (5.10)$$

The associated value function V^* corresponding to the optimal policy π^* satisfies the Bellman optimality equation [76],

$$\begin{aligned} V^*(s_t) &= \max_{a_t \in A} \mathbb{E} \left[R_{t+1} + \gamma V^*(s_{t+1}) \middle| s_t, \pi \right] \\ &= \max_{a_t \in A} \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1} | s_t, a_t) \left(R(s_t, a_t, s_{t+1}) + \gamma V^*(s_{t+1}) \right). \end{aligned} \quad (5.11)$$

The state-action value is denoted as $Q^\pi(s_t, a_t)$. In contrast to the state value V^π , the state-action value Q^π emphasizes the value of the choice of the first action starting at the current state. $Q^\pi(s_t, a_t)$ indicates the expected discounted cumulative reward starting at state s_t , taking action a_t and then following policy π , afterwards

$$Q^\pi(s_t, a_t) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| s_t, a_t, \pi \right]. \quad (5.12)$$

Similarly with (5.9), the evaluation equation for the state-action value Q^π is derived as follows,

$$\begin{aligned} Q^\pi(s_t, a_t) &= \mathbb{E} \left[R_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) \middle| s_t, a_t, \pi \right] \\ &= \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1} | s_t, a_t) \left(R'(s_t, a_t, s_{t+1}) + \gamma \sum_{a_{t+1} \in A} \pi(s_{t+1}, a_{t+1}) Q^\pi(s_{t+1}, a_{t+1}) \right). \end{aligned} \quad (5.13)$$

The state value V^π in (5.7) and the state-action value Q^π in (5.12) are related and they satisfy the Bellman equation as follows,

$$V^\pi(s_t) = \sum_{a_t \in A} \pi(s_t, a_t) Q^\pi(s_t, a_t). \quad (5.14)$$

The optimal policy π^* satisfying (5.10) also satisfies the following equation, which means that one can determine π^* by either solving (5.10) or, equivalently, by solving

(5.15),

$$\pi^* = \operatorname{argmax}_{\pi} Q^{\pi}(s, a), \quad \forall (s, a) \in S \times A. \quad (5.15)$$

The optimal state-action value Q^* corresponding to the optimal policy π^* satisfies the following Bellman optimality equation,

$$\begin{aligned} Q^*(s_t, a_t) &= \mathbb{E} \left[R_t + \gamma \max_{a_{t+1} \in A} Q^*(s_{t+1}, a_{t+1}) \middle| s_t, a_t, \pi \right] \\ &= \sum_{s_{t+1} \in S} \mathbb{P}(s_{t+1} | s_t, a_t) \left(R'(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1} \in A} Q^*(s_{t+1}, a_{t+1}) \right). \end{aligned} \quad (5.16)$$

Algorithm 1 Q-Learning Algorithm

Input: $S, A, \alpha, \gamma, \epsilon, R$

Output: Q^*, π^*

```

1:  $Q \leftarrow Q_0$ 
2:  $Q(s_{\text{final}}, \cdot) \leftarrow 0$ 
3:  $\text{Converge} \leftarrow \text{False}$ 
4: while not Converge do
5:    $s \leftarrow s_0$ 
6:   EpisodeOver  $\leftarrow \text{False}$ 
7:   while not EpisodeOver do
8:      $a \leftarrow \max_{a \in A} Q(s, a)$  (i.e.,  $\epsilon$ -greedy)
9:      $s' \leftarrow$  state after taking action  $a$ 
10:    if  $s' \in s_{\text{final}}$  then
11:      EpisodeOver  $\leftarrow \text{True}$ 
12:    else
13:       $Q(s, a) \leftarrow Q(s, a) + \alpha \left( R(s, a) + \gamma \max_{a' \in A} Q(s', a') - Q(s, a) \right)$ 
14:       $s \leftarrow s'$ 
15:    if  $Q$  converges then
16:      Converge  $\leftarrow \text{True}$ 
17:  $Q^* \leftarrow Q$ 
18:  $\pi^*(s) = \max_{a \in A} Q^*(s, a)$ 

```

The above definitions and equations in (5.7)-(5.16) are the basis for understanding most reinforcement learning algorithms such as value iteration, policy iteration, Sarsa and Q-learning. The Q-learning algorithm works directly on the Q values. The update law of the Q values can be expressed as follows [159, 76],

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right), \quad (5.17)$$

where $\alpha \in [0, 1]$ is the learning rate (step size), which determines how much the newly

acquired information overrides the current Q values. If $\alpha = 0$ one learns nothing since $Q(s_t, a_t)$ remains the same. If $\alpha = 1$ one abandons the old Q value and keeps only the newly learned value $R_t + \gamma \max_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1})$. In particular, if $\alpha = \alpha(t)$ is time-varying and equals to $1/(t+2)$, where $t+2$ represents the total number of visits to the state-action pair (s_t, a_t) , one obtains the sample-average result for all observed Q values. The well-known conditions on α to guarantee convergence of the Q values with probability 1 are given as follows [76],

$$\sum_{t=0}^{\infty} \alpha(t) = \infty, \quad (5.18)$$

$$\sum_{t=0}^{\infty} \alpha^2(t) < \infty, \quad (5.19)$$

where the first condition in (5.18) is used to overcome the random fluctuations, and the second condition in (5.18) guarantees convergence as $t \rightarrow \infty$. However, the conditions in (5.18) are seldom used in practice. For instance, a constant step size $\alpha \neq 0$ does not satisfy (5.18), but it has been shown to perform well in many problems. In this dissertation we also use a constant step size (i.e., $\alpha = 0.75$) in all the examples.

The discount rate $\gamma \in [0, 1)$ in (5.17) describes the importance of future rewards for the agent. Specifically, $\gamma = 0$ indicates that the agent only considers the immediate reward after taking action a_t , and hence the agent is “myopic”. The agent becomes more “far-sighted” as γ approaches 1, since more cumulative future rewards are taken into account to update the Q values. A value of $\gamma \geq 1$ may lead to divergence.

We summarize the Q -learning algorithm in Algorithm 1. In Section 5.6 we design the weight vector w in the reward function (5.6), and the values of the parameters α , γ and ϵ , in Algorithm 1 to learn the optimal policy π^* .

5.4 Maximum Entropy Principle

In Section 5.3 the driver’s reward was designed and we learned the desired driving policy using reinforcement learning. Nonetheless, in some cases one may have little knowledge about the reward function, and it is hard to design the required reward function to achieve the desired driving behavior. The approach in Section 5.3 is not convenient to use if the prior knowledge of the reward function is not sufficient. However, one can avoid designing the reward function and directly learn the optimal driving policy from demonstrations performed by an expert driver. This type of problem is called inverse reinforcement learning or inverse optimal control [168, 80].

Before proceeding with the discussion on inverse reinforcement learning, we introduce the maximum-entropy principle, and explain how this principle can be used to recover the unknown reward function, which can then be used to learn the driving policy using the given demonstrations.

5.4.1 Maximum Entropy Principle

The maximum entropy principle was first introduced by Jaynes [169], and since then it has been used in many areas of computer science and statistical learning. In the basic maximum entropy formulation, one is given a set of samples from a target distribution and a set of constraints on this distribution, and then one estimates this distribution using the maximum entropy distribution that satisfies these constraints [170]. Mathematically, the idea can be demonstrated as the following theorem.

Theorem 5.4.1 *Suppose $x_i \in \mathcal{X}$, $i = 1, \dots, n$ are independent and identically distributed (i.i.d) samples from a certain distribution $x_i \sim p^*$. Let*

$$\hat{\mu}_j = \frac{1}{n} \sum_{i=1}^n f_j(x_i), \quad j = 1, \dots, m, \quad (5.20)$$

where $f_j : \mathcal{X} \rightarrow \mathbb{R}$ are real-valued functions and $\hat{\mu}_j$ are the empirical expectations of f_j . The maximum entropy estimate \hat{p} of the distribution p^* satisfies

$$\hat{p} = \arg \max_p \int_{\mathcal{X}} -p(x) \log p(x) v(dx), \quad (5.21a)$$

$$\text{subject to } \mathbb{E}[f_j(x)] = \int_{\mathcal{X}} p(x) f_j(x) v(dx) = \hat{\mu}_j, \quad (5.21b)$$

$$\int_{\mathcal{X}} p(x) v(dx) = 1, \quad j = 1, \dots, m, \quad (5.21c)$$

where v is a base measure. The solution \hat{p} of (5.21) is given by

$$\hat{p}(x) = \frac{1}{Z(\theta)} e^{\sum_{j=1}^m \theta_j f_j(x)}, \quad (5.22)$$

where $Z(\theta)$ is the partition function having the following form

$$Z(\theta) = \int_{\mathcal{X}} e^{\sum_{j=1}^m \theta_j f_j(x)} v(dx), \quad (5.23)$$

and where $\theta_j \in \mathbb{R}$ are parameters satisfying the following equations,

$$\int_{\mathcal{X}} \frac{f_j(x)}{Z(\theta)} e^{\sum_{k=1}^m \theta_k f_k(x)} v(dx) = \hat{\mu}_j. \quad (5.24)$$

The maximum entropy principle finds a distribution satisfying the constraints with the largest remaining uncertainty, so that one does not introduce any additional assumptions or biases into the computation of \hat{p} .

In inverse reinforcement learning problems, one is given a number of time histories of the agent's behaviors consisting the past states and actions. These past states and actions are usually called demonstrations. Ziebart [78] first applied the maximum entropy principle to solve inverse reinforcement learning problems, for cases where the

reward function depends only on the current state, and it was represented via a linear combination of feature functions, namely,

$$R(s) = \sum_i w_i \phi_i(s) = w^\top \Phi(s), \quad (5.25)$$

where w and $\Phi(s)$ are the weight and feature vectors, respectively. Note that in [78] the feature vector $\Phi(s)$ is a function of state s only, and the actions were not considered. The probability of a demonstration $\zeta \triangleq \{s_0, a_0, \dots, s_T, a_T\}$ over all paths of duration T is calculated following Theorem 5.4.1 [78],

$$\mathbb{P}(\zeta|w) = \frac{1}{Z(w)} e^{\sum_{s \in \zeta} w^\top \Phi(s)}, \quad (5.26a)$$

$$\mathbb{P}(\zeta|w) = \frac{1}{Z(w)} e^{\sum_{s \in \zeta} w^\top \Phi(s)} \prod_{(s,a,s') \in \zeta} \mathbb{P}(s'|s, a), \quad (5.26b)$$

where the partition function $Z(w)$ is a normalization constant, and (5.26a) and (5.26b) provide the solutions corresponding to a deterministic MDP, where the future state can be uniquely determined with the given action at the present state, and a stochastic MDP, where the future state is unpredictable with the given action at the present state, respectively.

Note that in order to simplify the expressions, in the following we use the notations $s \in \zeta$, $(s, a) \in \zeta$ and $(s, a, s') \in \zeta$ to denote the cases when the state s , the state-action pair (s, a) and the state-action-state triple (s, a, s') are demonstrated in ζ , respectively. Either equation in (5.26) presents a distribution over paths (demonstrations) and indicates that the probability of a path is proportional to the exponential of its total reward, which implies that the paths having higher rewards are more preferable by the agent.

The goal of an inverse reinforcement learning problem is to find the optimal weight w^* , such that the likelihood of the observed demonstrations is maximal under the distribution in (5.26). In the following, we formulate the inverse reinforcement learning problem using different reward structures. The necessary derivations are provided for IRL problems satisfying the following requirements: 1) Instead of using the state reward $R(s)$ and the state feature $\Phi(s)$ as in [78, 82], we use $R(s, a)$ and $\Phi(s, a)$, and 2) We focus only on the stochastic MDP. The demonstrations are required to start from the same state s_0 and are observed over the same time horizon ranging from $t = 0$ to $t = T$. We use the following notation: \mathcal{D} denotes the set of demonstrations, N denotes the number of demonstrations in \mathcal{D} , $\Omega \supseteq \mathcal{D}$ denotes the complete path space, and Φ_ζ denotes the feature counts along the path $\zeta \in \mathcal{D}$ which is given by $\Phi_\zeta = \sum_{(s,a) \in \zeta} \Phi(s, a)$.

5.4.2 Nonparameterized Features

The (nonparameterized) features $\Phi(s, a)$ are functions of only the states and actions. One may then consider reward functions as a linear combination of features in the fol-

lowing form

$$R(w; s, a) = w^\top \Phi(s, a). \quad (5.27)$$

In order to explicitly show the dependency of the reward function on the unknown weight vector w , we use the notation $R(w; s, a)$ instead of $R(s, a)$ in (5.27). It follows from [169] that maximizing the entropy of the distribution over Ω subject to the feature constraints from observations \mathcal{D} implies the maximization of the likelihood of \mathcal{D} under the maximum entropy distribution in (5.26b), that is,

$$\begin{aligned} w^* &= \arg\max_w \mathcal{L}_D(w) = \arg\max_w \frac{1}{N} \sum_{\zeta \in \mathcal{D}} \log \mathbb{P}(\zeta|w) \\ &= \arg\max_w \frac{1}{N} \left(\sum_{\zeta \in \mathcal{D}} \left(w^\top \Phi_\zeta + \sum_{(s,a,s') \in \zeta} \mathbb{P}(s'|s, a) \right) \right) - \log Z(w). \end{aligned} \quad (5.28)$$

In order to use gradient-based optimization methods to solve the problem in (5.28), we take the partial derivative of \mathcal{L}_D with respect to the partial derivative of w , to obtain

$$\begin{aligned} \frac{\partial \mathcal{L}_D}{\partial w} &= \frac{1}{N} \sum_{\zeta \in \mathcal{D}} \Phi_\zeta - \frac{1}{Z(w)} \sum_{\zeta \in \Omega} \Phi_\zeta e^{w^\top \Phi_\zeta} \prod_{(s,a,s') \in \zeta} \mathbb{P}(s'|s, a) = \tilde{\Phi} - \sum_{\zeta \in \Omega} \mathbb{P}(\zeta|w) \Phi_\zeta \\ &= \tilde{\Phi} - \mathbb{E}[\Phi_\zeta], \end{aligned} \quad (5.29)$$

where $\tilde{\Phi} \triangleq \frac{1}{N} \sum_{\zeta \in \mathcal{D}} \Phi_\zeta$ is the expected empirical feature count, and the expectation of the feature count Φ_ζ can be calculated by

$$\mathbb{E}[\Phi_\zeta] = \sum_{s \in S} \sum_{a \in A} \mathbb{E}[\mu(s, a)] \Phi(s, a), \quad (5.30)$$

and where the term $\mathbb{E}[\mu(s, a)]$ denotes the expected state-action pair visitation counts. One may refer to Algorithm 3 in Section 5.5.2 for the calculation of $\mathbb{E}[\mu(s, a)]$.

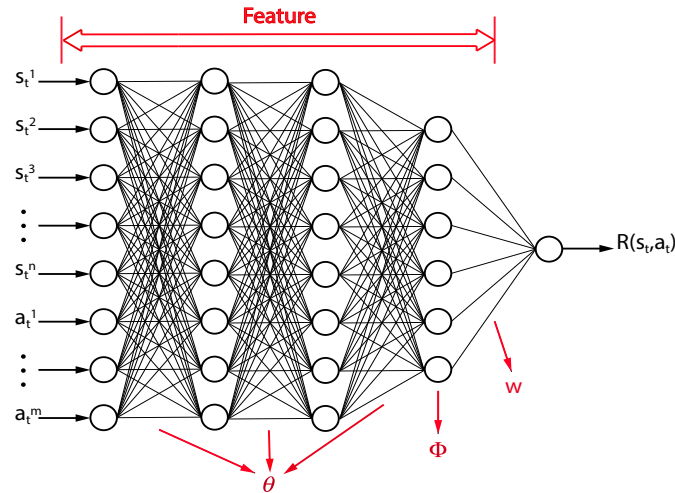


Figure 5.7: Deep neural-network feature function and reward.

5.4.3 Parameterized Features

The use of nonparameterized features in Section 5.4.2 requires one to design the features manually, which may be a difficult task, in general, since it may not always be possible to approximate a certain unknown reward function having a complicated form. Hence, we consider the use of parameterized features instead of nonparameterized features, so that one can refine the feature design by optimizing the parameters of the features.

In order to formulate the maximum entropy IRL problem, we still consider a reward function given as a linear combination of the features, but the features now depend explicitly on a parameter vector θ ,

$$R(w, \theta; s, a) = w^\top \Phi(\theta; s, a). \quad (5.31)$$

Instead of tuning only the weight vector w in (5.28), we also tune the vector θ associated with w to maximize the likelihood \mathcal{L}_D as follows,

$$\begin{aligned} w^*, \theta^* &= \arg \max_{w, \theta} \mathcal{L}_D(w, \theta) = \arg \max_{w, \theta} \frac{1}{N} \sum_{\zeta \in \mathcal{D}} \log \mathbb{P}(\zeta | w, \theta) \\ &= \arg \max_{w, \theta} \frac{1}{N} \left(\sum_{\zeta \in \mathcal{D}} \left(w^\top \Phi_\zeta(\theta) + \sum_{(s, a, s') \in \zeta} \mathbb{P}(s' | s, a) \right) \right) - \log Z(w, \theta). \end{aligned} \quad (5.32)$$

The derivations of $\partial \mathcal{L}_D / \partial w$ are similar with (5.29)-(5.30) and hence are omitted. Thus, we only show the derivation of $\partial \mathcal{L}_D / \partial \theta$, which following the chain rule yields

$$\frac{\partial \mathcal{L}_D(w, \theta)}{\partial \theta} = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{\partial \mathcal{L}_D(w, \theta)}{\partial R(w, \theta; s, a)} \frac{\partial R(w, \theta; s, a)}{\partial \theta}, \quad (5.33)$$

where

$$\begin{aligned} \frac{\partial \mathcal{L}_D(w, \theta)}{\partial R(w, \theta; s, a)} &= \frac{\frac{1}{N} \sum_{\zeta \in \mathcal{D}} \sum_{(\hat{s}, \hat{a}) \in \zeta} R(w, \theta; \hat{s}, \hat{a})}{\partial R(w, \theta; s, a)} - \frac{1}{Z(w, \theta)} \sum_{\zeta \in \Omega} e^{w^\top \Phi_\zeta} \prod_{(\hat{s}, \hat{a}, \hat{s}') \in \zeta} \mathbb{P}(\hat{s}' | \hat{s}, \hat{a}) \\ \frac{\partial \sum_{(\hat{s}, \hat{a}) \in \zeta} R(w, \theta; \hat{s}, \hat{a})}{\partial R(w, \theta; s, a)} &= \mu_D(s, a) - \sum_{\zeta \in \Omega} \mathbb{P}(\zeta | w, \theta) \frac{\partial \sum_{(\hat{s}, \hat{a}) \in \zeta} R(w, \theta; \hat{s}, \hat{a})}{\partial R(w, \theta; s, a)} = \mu_D(s, a) - \mathbb{E}[\mu(s, a)], \end{aligned} \quad (5.34)$$

and where $\mu_D(s, a)$ is the expected empirical state-action pair visitation counts over the demonstrations \mathcal{D} . The expression $\partial R(w, \theta; s, a) / \partial \theta$ in (5.33) is given by

$$\frac{\partial R(w, \theta; s, a)}{\partial \theta} = \frac{\partial w^\top \Phi(\theta; s, a)}{\partial \theta} = w^\top \frac{\partial \Phi(\theta; s, a)}{\partial \theta}, \quad (5.35)$$

where the (i, j) entry of the matrix $\partial \Phi(\theta; s, a) / \partial \theta$ is defined as follows

$$\left[\frac{\partial \Phi(\theta; s, a)}{\partial \theta} \right]_{i, j} = \frac{\partial \phi_i(\theta; s, a)}{\partial \theta_j}, \quad (5.36)$$

and where $\phi_i(\theta; s, a)$ is the i th element of $\Phi(\theta; s, a)$.

Equations (5.33)-(5.36) provide the necessary ingredients to improve the design of the feature functions by tuning their parameters. One interesting application of these results is to use a DNN having multiple outputs to represent the features (see Figure 5.7). In Figure 5.7 the reward $R(s_t, a_t)$ is given by a linear combination of features represented by a DNN, where the inputs s_t^i and a_t^j of the DNN represent the i th and j th elements of the n -dimension state vector and the m -dimension action vector, respectively. One can then calculate $\partial R(w, \theta; s, a) / \partial \theta$ by back propagating the network following the delta rule[171].

It is worth mentioning that one can let $w = 1$ and directly use a single DNN feature to represent the reward function. In this case the maximum entropy distribution over the path space Ω is obtained by tuning only the parameters θ in the DNN. For instance, Wulfmeier [82] used a DNN to express the state reward $R(s)$ and determined the maximum entropy distribution of the path ζ by training a DNN.

5.5 Inverse Reinforcement Learning

Based on the theoretical derivation in Section 5.4, we next summarize the inverse reinforcement learning algorithm, which was used in this work to learn the optimal policy from driving demonstrations.

One can use a DNN as a parameterized reward function in order to apply the maximum entropy principle to solve the inverse reinforcement learning problem. In the following, we first discuss the structure of the DNN reward function, and next, we introduce two new IRL algorithms to learn the unknown parameters of the DNN.

5.5.1 Reward Approximator

In order to recover the unknown reward function from demonstrations, we use a universal approximator that has the ability to represent any function, such as a DNN with multiple layers [172, 173].

We consider three definitions of the reward functions from the literature, namely, the state reward $R : S \rightarrow \mathbb{R}$ [77, 78, 81, 82], the state-action reward $R : S \times A \rightarrow \mathbb{R}$ [76, 47, 80] and the state-action-state reward $R : S \times A \times S \rightarrow \mathbb{R}$ [76]. We denote the corresponding reward functions as $R(s)$, $R(s, a)$ and $R(s, a, s')$, respectively. $R(s)$ is used when the agent wants to reach a goal state or avoid certain dangerous states by taking any action. This definition indicates that the agent has no specific preference over the existing actions. In contrast, $R(s, a)$ takes the action into consideration, so that it can be used to show the agent's preference on a specific action. The last definition $R(s, a, s')$ takes into consideration also the resulting state s' after the agent takes action a at the present state s . Nevertheless, since the resulting state s' depends on the response of the environment after the agent takes action a , the agent can only make a decision according to the expected reward of taking action a without knowing the future state s' . Thus $R(s, a, s')$ and

$R(s, a)$ are expected to be equivalent in terms of learning the same policy. See also equation (5.8). According to the above discussion, we will use the state-action reward $R(s, a)$ to reproduce driving behaviors having different preference on the available actions.

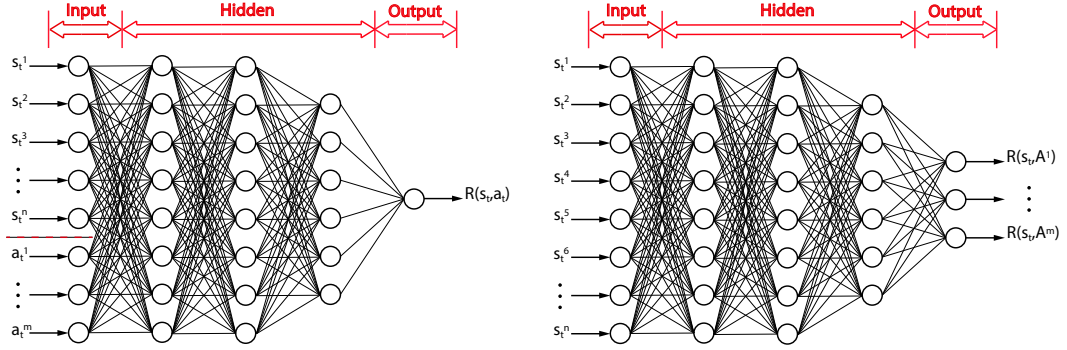


Figure 5.8: Structures of the deep neural-network reward functions.

The two structures of the DNNs we have in mind are shown in Figure 5.8. In the first structure we use both the state s_t and the action a_t as the input, where s_t^i and a_t^j represent the i th and j th elements of the n -dimensional state vector and the m -dimensional action vector at time step t , respectively. In the second structure we use only the state s_t as the input to the network, and use different channels of the output to represent the reward corresponding to different actions in the action set A . The output $R(s_t, A^j)$, $j = 1, \dots, 5$ represents the reward received by the agent by taking action A^j at present state s_t , where A^j represents the j th action in the action set A . The present action a_t is not an explicit input for the second structure. Both of these two DNN structures in Figure 5.8 can be used as an approximator for the state-action reward function, and one can take either form that is most convenient for the learning task at hand.

5.5.2 MaxEnt Deep IRL Algorithm

We summarize the IRL algorithm used to learn the unknown parameters of the DNN in Figure 5.8, using the maximum entropy principle based on the results in Section 5.4. Since the calculation of the expected state-action visitation number requires knowledge of the model, we first discuss the model learning.

Model Learning

We consider a totally model-free case, where no knowledge about the state transition model $\mathbb{P}(s'|s, a)$ is available. The idea of model learning is that one can analyze the visitation count of each state-action-state triple and calculate the probability for each possible result of the state transitions, which is given by

$$\mathbb{P}(s'|s, a) = \frac{v(s, a, s')}{\sum_{s' \in S} v(s, a, s')}, \quad (5.37)$$

where $v(s, a, s')$ is the total number of the state transition from s to s' by taking action a . The probability $\mathbb{P}(s'|s, a)$ approaches its actual value as the state visitation count $v(s, a, s')$ approaches infinity. Model learning can be implemented along with the Q-learning (see Algorithm 1). The algorithm for Q-learning with model learning is summarized in Algorithm 2.

IRL Algorithm

In this section we summarize the MaxEnt Deep IRL algorithm according to (5.32)-(5.36). To this end, we first introduce the following algorithm to calculate the expected state-action visitation counts $\mathbb{E}[\mu(s, a)]$ in (5.34), using the model learning result $\mathbb{P}(s'|s, a)$ from Algorithm 2.

Algorithm 2 Q-Learning with Model Learning

Input: $S, A, \alpha, \gamma, \epsilon, R, v_0$

Output: $Q^*, \pi^*, v, \mathbb{P}$

```

1:  $v \leftarrow v_0$ 
2:  $Q(s_0, a_0) \leftarrow Q_0$ 
3:  $Q(s_{\text{final}}, \cdot) \leftarrow 0$ 
4: Converge  $\leftarrow$  False
5: while not Converge do
6:    $s \leftarrow s_0$ 
7:   EpisodeOver  $\leftarrow$  False
8:   while not EpisodeOver do
9:      $a \leftarrow \max_{a \in A} Q(s, a)$  (i.e.,  $\epsilon$ -greedy)
10:     $s' \leftarrow$  state after taking action  $a$ 
11:     $v(s, a, s') \leftarrow v(s, a, s') + 1$ 
12:    if  $s' \in s_{\text{final}}$  then
13:      EpisodeOver  $\leftarrow$  True
14:    else
15:       $Q(s, a) \leftarrow Q(s, a) + \alpha \left( R(s, a) + \gamma \max_{a \in A} Q(s', a) - Q(s, a) \right)$ 
16:       $s \leftarrow s'$ 
17:    if  $Q$  converges then
18:      Converge  $\leftarrow$  True
19:  $Q^* \leftarrow Q$ 
20:  $\pi^*(s) = \max_{a \in A} Q^*(s, a)$ 
21:  $\mathbb{P}(s'|s, a) \leftarrow \frac{v(s, a, s')}{\sum_{s' \in S} v(s, a, s')}$ 

```

In Algorithm 3 the state s_{final} denotes the terminal state or the goal state, after which the state of the system will not change, meaning that no future state transitions can occur at s_{final} . Next, one calculates the gradient $\partial \mathcal{L}_D / \partial \theta$ using (5.41), and updates the

parameters θ of the DNN using gradient decent. The expression of $\Delta\theta$ can be calculated as follows,

$$\Delta\theta = \lambda \frac{\partial \mathcal{L}_D}{\partial \theta}, \quad (5.38)$$

where λ is the learning rate (time step). One can also introduce a weight decay term as a model regularizer into $\Delta\theta$, such as an L_1 regularizer [78], an L_2 regularizer [82], and other forms of regularizers [174, 83]. The regularizers are used to convexify the problem, mitigate overfitting, or introduce other properties to the optimization problem such as monotonicity by adding a self-defined cost term.

Algorithm 3 Expected State-Action Visitation Counts

Input: $T, S, A, \pi(s, a), \mathbb{P}(s|s', a)$

Output: $\mathbb{E}[\mu(s, a)]$

- 1: **Calculate expected state/state-action visitation counts:**
 - 2: $\mathbb{E}[\mu(s_0)] \leftarrow 1$
 - 3: $\mathbb{E}[\mu(s_0, a)] \leftarrow \pi(s_0, a)$
 - 4: **for** $i = 1 : T$ **do**
 - 5: $\mathbb{E}_i[\mu(s_{\text{final}})] \leftarrow 0$
 - 6: $\mathbb{E}_i[\mu(s_{\text{final}}, a)] \leftarrow 0$
 - 7: $\mathbb{E}_{i+1}[\mu(s)] \leftarrow \sum_{s' \in S} \sum_{a \in A} \mathbb{P}(s|s', a) \pi(s', a) \mathbb{E}_i[\mu(s')]$
 - 8: $\mathbb{E}_{i+1}[\mu(s, a)] \leftarrow \pi(s, a) \mathbb{E}_{i+1}[\mu(s)]$
 - 9: $\mathbb{E}[\mu(s, a)] \leftarrow \sum_{i=1}^T \mathbb{E}_i[\mu(s, a)]$
-

The proposed MaxEnt Deep IRL algorithm is summarized in Algorithm 4.

5.5.3 IRL Algorithm Refinement

Learning of the model may not yield good results before one has a large number of visitations for each state-action pair. The error of the state transition probability $\mathbb{P}(s'|s, a)$ may lead to errors in calculating $\partial \mathcal{L}^D / \partial R(s, a)$ in Algorithm 4, which leads to further errors in calculating the gradients $\partial \mathcal{L}^D / \partial \theta$ that are used to update the parameters θ in the neural-network. To address this issue, one can pre-learn the model until it converges before using it in Algorithm 4. Nevertheless, the demonstrations in \mathcal{D} may not be enough to represent the environment's random behavior, especially in the case where the system is complicated and the demonstrations are required to represent the long-term behavior of the stochastic system. One can then either split the demonstrations into small pieces to avoid a large error in predicting the long-term behavior of the system, or, alternatively, try to avoid using the state transition terms in the calculation of the gradients in (5.29) and (5.34).

In this work we regenerate a number of new sets of the demonstrations $\mathcal{D}_\tau = \{\zeta_\tau^i, i = 1, \dots, N_\tau\}$ using the original demonstrations \mathcal{D} . The element ζ_τ^i satisfies the following

conditions: 1) ζ_τ^i starts at the state $\tau \in S$, 2) The length of ζ_τ^i is constant ΔT for all $\tau \in S$, and 3) There exists a path $\zeta \in \mathcal{D}$ such that $\zeta_\tau^i \subseteq \zeta$. The corresponding path space for ζ_τ^i is denoted as Ω_τ . We then maximize the entropy of the joint distribution over all Ω_τ subject to the constraints from the demonstrations \mathcal{D}_τ ,

$$\begin{aligned} \theta^* &= \arg \max_{\theta} \mathcal{L}_D(\theta) = \arg \max_{\theta} \sum_{\tau \in S} \frac{1}{N_\tau} \sum_{\zeta \in \mathcal{D}_\tau} \log \mathbb{P}(\zeta|\theta) \\ &= \arg \max_{\theta} \sum_{\tau \in S} \left(\frac{1}{N_\tau} \left(\sum_{\zeta \in \mathcal{D}_\tau} \left(\sum_{(s,a) \in \zeta} R(\theta; s, a) + \sum_{(s,a,s') \in \zeta} \mathbb{P}(s'|s, a) \right) \right) - \log Z_\tau(\theta) \right), \end{aligned} \quad (5.39)$$

where the partition function Z_τ is given by

$$Z_\tau(\theta) = \sum_{\zeta \in \Omega_\tau} e^{\sum_{(s,a) \in \zeta} R(\theta; s, a)}. \quad (5.40)$$

The partial derivative of \mathcal{L}_D with respect to the partial derivative of θ is given by

$$\frac{\partial \mathcal{L}_D(\theta)}{\partial \theta} = \sum_{s \in S} \sum_{a \in A} \frac{\partial \mathcal{L}_D(\theta)}{\partial R(\theta; s, a)} \frac{\partial R(\theta; s, a)}{\partial \theta}, \quad (5.41)$$

where

$$\begin{aligned} \frac{\partial \mathcal{L}_D(\theta)}{\partial R(\theta; s, a)} &= \sum_{\tau \in S} \left(\frac{\frac{1}{N_\tau} \sum_{\zeta \in \mathcal{D}_\tau} \sum_{(\hat{s}, \hat{a}) \in \zeta} R(\theta; \hat{s}, \hat{a})}{\partial R(\theta; s, a)} - \frac{1}{Z_\tau(\theta)} \sum_{\zeta \in \Omega_\tau} e^{\sum_{(\hat{s}, \hat{a}) \in \zeta} R(\theta; \hat{s}, \hat{a})} \prod_{(\hat{s}, \hat{a}, \hat{s}') \in \zeta} \mathbb{P}(\hat{s}'|\hat{a}, \hat{s}) \right. \\ &\quad \left. \frac{\partial \sum_{(\hat{s}, \hat{a}) \in \zeta} R(\theta; \hat{s}, \hat{a})}{\partial R(\theta; s, a)} \right) = \sum_{\tau \in S} \left(\mu_{D_\tau}(s, a) - \sum_{\zeta \in \Omega_\tau} \mathbb{P}(\zeta|\theta) \frac{\partial \sum_{(\hat{s}, \hat{a}) \in \zeta} R(\theta; \hat{s}, \hat{a})}{\partial R(\theta; s, a)} \right) \\ &= \sum_{\tau \in S} (\mu_{D_\tau}(s, a) - \mathbb{E}[\mu_\tau(s, a)]), \end{aligned} \quad (5.42)$$

and where μ_{D_τ} is the expected empirical state-action pair visitation counts over the demonstrations \mathcal{D}_τ . The term $\partial R/\partial \theta$ in (5.41) can be obtained by backward propagating the DNN.

The expected state-action pair visitation counts $\mathbb{E}[\mu_\tau(s, a)]$ is calculated over ΔT steps. Specifically, we let $\Delta T = 1$ and consider only the one-step action case, such that we avoid using the unknown model transition probabilities to calculate $\partial \mathcal{L}_D(\theta)/\partial R(\theta; s, a)$ in (5.42), which is given by

$$\begin{aligned} \frac{\partial \mathcal{L}_D(\theta)}{\partial R(\theta; s, a)} &= \sum_{\tau \in S} \left(\frac{\frac{1}{N_\tau} \sum_{(\hat{s}, \hat{a}) \in \mathcal{D}_\tau} R(\theta; \hat{s}, \hat{a})}{\partial R(\theta; s, a)} - \frac{1}{Z_\tau(\theta)} \sum_{(\hat{s}, \hat{a}) \in \Omega_\tau} e^{R(\theta; \hat{s}, \hat{a})} \frac{\partial R(\theta; \hat{s}, \hat{a})}{\partial R(\theta; s, a)} \right) \\ &= \mu_{D_s}(s, a) - \mathbb{P}(s, a|\theta) \triangleq \pi_D(s, a) - \pi(s, a), \end{aligned} \quad (5.43)$$

where μ_{D_s} is the expected empirical state-action pair visitation counts over the demonstrations \mathcal{D}_s , which, can be defined as the expected empirical policy $\pi_D(s, a)$. The result

of (5.43) indicates that, by using the demonstrations with $\Delta T = 1$, the maximum entropy IRL formulation in (5.39) learns a reward function $R(\theta; s, a)$ such that the learned policy equals the expected empirical policy, namely, $\pi(s, a) = \pi_D(s, a)$.

Algorithm 4 MaxEnt Deep IRL Algorithm

Input: $\mu_D(s, a)$, T , S , A , α , β , γ , λ , ϵ , v_0

Output: π^* , θ^* , R^* , \mathbb{P}

```

1:  $\theta \leftarrow \theta_0$ 
2:  $v \leftarrow v_0$ 
3: Converge  $\leftarrow$  False
4: while not Converge do
5:   Update reward function:
6:    $R \leftarrow NN(\theta)$ 
7:   Update policy:
8:    $\pi, v, \mathbb{P} \leftarrow$  Q-learning with model learning( $S, A, \alpha, \gamma, \epsilon, R, v$ ) from Algorithm 2
9:   Calculate expected state/state-action visitation counts:
10:   $\mathbb{E}[\mu(s_0)] \leftarrow 1$ 
11:   $\mathbb{E}[\mu(s_0, a)] \leftarrow \pi(s_0, a)$ 
12:  for  $i = 1 : T$  do
13:     $\mathbb{E}_i[\mu(s_{\text{final}})] \leftarrow 0$ 
14:     $\mathbb{E}_i[\mu(s_{\text{final}}, a)] \leftarrow 0$ 
15:     $\mathbb{E}_{i+1}[\mu(s)] \leftarrow \sum_{s' \in S} \sum_{a \in A} \mathbb{P}(s|s', a) \pi(s', a) \mathbb{E}_i[\mu(s')]$ 
16:     $\mathbb{E}_{i+1}[\mu(s, a)] \leftarrow \pi(s, a) \mathbb{E}_{i+1}[\mu(s)]$ 
17:     $\mathbb{E}[\mu(s, a)] \leftarrow \sum_{i=1}^T \mathbb{E}_i[\mu(s, a)]$ 
18:   Determine Maximum-Entropy gradients:
19:    $\frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \leftarrow \mu_D(s, a) - \mathbb{E}[\mu(s, a)]$ 
20:   Update neural-network weights:
21:    $\frac{\partial R(\theta; s, a)}{\partial \theta} \leftarrow$  backward propagating neural-network
22:    $\frac{\partial \mathcal{L}_D}{\partial \theta} \leftarrow \frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \cdot \frac{\partial R(\theta; s, a)}{\partial \theta}$ 
23:    $\theta \leftarrow \theta + \lambda \frac{\partial \mathcal{L}_D}{\partial \theta} + \beta \theta$  (in case of using an  $L_2$  regularizer)
24:   if  $\theta$  converges then
25:     Converge  $\leftarrow$  True
26:      $\theta^* \leftarrow \theta$ 
27:    $R^* \leftarrow NN(\theta^*)$ 
28:  $\pi^* \leftarrow$  Q-learning( $S, A, \alpha, \gamma, \epsilon, R^*, v$ ) from Algorithm 2

```

We summarize the refined algorithms using (5.42) and (5.43), respectively, which are given by Algorithms 5 and 6. As a special case when the data length is $\Delta T = 1$, the single-step IRL algorithm in Algorithm 5 totally avoids calculating the expected state-action visitation counts, and hence the policy is learned without any knowledge of the model. The selection among the three IRL algorithms proposed in this dissertation depends on the data and the MDP model one has for the problem. For instance, if the behavior of

the MDP model is not difficult to predict (i.e., deterministic MDP) and one has collected a long set of data starting from the same initial state, one can use Algorithm 4 to recover the reward function and learn the policy. If the data have different lengths and were collected with different initial states, one may have to reorganize the data and consider using Algorithms 5 and 6, especially for problems having complicated stochastic behavior.

Algorithm 5 Single-Step Joint Maximum-Entropy Deep IRL Algorithm

Input: $\pi_D(s, a)$, S , A , α , β , γ , λ , ϵ , v_0

Output: π^* , θ^* , R^* , \mathbb{P}

```

1:  $\theta \leftarrow \theta_0$ 
2:  $v \leftarrow v_0$ 
3: Converge  $\leftarrow$  False
4: while not Converge do
5:   Update reward function:
6:    $R \leftarrow NN(\theta)$ 
7:   Update policy:
8:    $\pi, v, \mathbb{P} \leftarrow$  Q-learning with model learning( $S, A, \alpha, \gamma, \epsilon, R, v$ ) from Algorithm 2
9:   Determine Maximum-Entropy gradients:
10:   $\frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \leftarrow \pi_D(s, a) - \pi(s, a)$ 
11:  Update neural-network weights:
12:   $\frac{\partial R(\theta; s, a)}{\partial \theta} \leftarrow$  backward propagating neural-network
13:   $\frac{\partial \mathcal{L}_D}{\partial \theta} \leftarrow \frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \cdot \frac{\partial R(\theta; s, a)}{\partial \theta}$ 
14:   $\theta \leftarrow \theta + \lambda \frac{\partial \mathcal{L}_D}{\partial \theta} + \beta \theta$ 
15:  if  $\theta$  converges then
16:    Converge  $\leftarrow$  True
17:     $\theta^* \leftarrow \theta$ 
18:   $R^* \leftarrow NN(\theta^*)$ 
19:   $\pi^*, \mathbb{P} \leftarrow$  Q-learning( $S, A, \alpha, \gamma, \epsilon, R^*, v$ ) from Algorithm 2

```

5.6 Results and Analysis

In this section we implement the previous RL and IRL algorithms for the traffic model of Section 5.2 and analyze the results.

5.6.1 Driving Behavior from Reinforcement Learning

We show two different driving behaviors using RL, namely, overtaking and tailgating. To this end, we use the features defined in Section 5.3.2 and design the weights w_1 and w_2 to achieve the two desired driving behaviors, respectively. The weights are provided in Table 5.1.

Algorithm 6 Multiple-Step Joint Maximum-Entropy Deep IRL Algorithm

Input: $\mu_{D_\tau}(s, a), \Delta T, S, A, \alpha, \beta, \gamma, \lambda, \epsilon, \nu_0$ **Output:** $\pi^*, \theta^*, R^*, \mathbb{P}$

```
1:  $\theta \leftarrow \theta_0$ 
2:  $\nu \leftarrow \nu_0$ 
3: Converge  $\leftarrow$  False
4: while not Converge do
5:   Update reward function:
6:    $R \leftarrow NN(\theta)$ 
7:   Update policy:
8:    $\pi, \nu, \mathbb{P} \leftarrow$  Q-learning with model learning( $S, A, \alpha, \gamma, \epsilon, R, \nu$ ) from Algorithm 2
9:   Calculate expected state/state-action visitation counts:
10:  for  $\tau$  in  $S$  do
11:     $\mathbb{E}[\mu_\tau(s_0 = \tau)] \leftarrow 1$ 
12:     $\mathbb{E}[\mu_\tau(s_0 = \tau, a)] \leftarrow \pi(s_0, a)$ 
13:    for  $i = 1 : \Delta T$  do
14:       $\mathbb{E}_i[\mu_\tau(s_{\text{final}})] \leftarrow 0$ 
15:       $\mathbb{E}_i[\mu_\tau(s_{\text{final}}, a)] \leftarrow 0$ 
16:       $\mathbb{E}_{i+1}[\mu_\tau(s)] \leftarrow \sum_{s' \in S} \sum_{a \in A} \mathbb{P}(s|a, s') \pi(s', a) \mathbb{E}_i[\mu_\tau(s')]$ 
17:       $\mathbb{E}_{i+1}[\mu_\tau(s, a)] \leftarrow \pi(s, a) \mathbb{E}_{i+1}[\mu_\tau(s)]$ 
18:       $\mathbb{E}[\mu_\tau(s, a)] \leftarrow \sum_{i=1}^{\Delta T} \mathbb{E}_i[\mu_\tau(s, a)]$ 
19:  Determine Maximum-Entropy gradients:
20:   $\frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \leftarrow \sum_{\tau \in S} (\mu_{D_\tau}(s, a) - \mathbb{E}[\mu_\tau(s, a)])$ 
21:  Update neural-network weights:
22:   $\frac{\partial R(\theta; s, a)}{\partial \theta} \leftarrow$  backward propagating neural-network
23:   $\frac{\partial \mathcal{L}_D}{\partial \theta} \leftarrow \frac{\partial \mathcal{L}_D}{\partial R(\theta; s, a)} \cdot \frac{\partial R(\theta; s, a)}{\partial \theta}$ 
24:   $\theta \leftarrow \theta + \lambda \frac{\partial \mathcal{L}_D}{\partial \theta} + \beta \theta$ 
25:  if  $\theta$  converges then
26:    Converge  $\leftarrow$  True
27:     $\theta^* \leftarrow \theta$ 
28:   $R^* \leftarrow NN(\theta^*)$ 
29:   $\pi^* \leftarrow$  Q-learning( $S, A, \alpha, \gamma, \epsilon, R^*, \nu$ ) from Algorithm 2
```

The desired driving behavior by designing w_1 is to show overtaking, which can be described as follows: 1) The HV accelerates to occupy the front cell if it is available; 2) The HV maintains its velocity if there is an EV in front of it and no overtaking is possible; 3) The HV overtakes the front EV if only one side is available for overtaking, by lane-shifting first and then accelerating and maintaining constant speed; 4) The HV overtakes the front EV from the inner side of the corner if both the left and right sides are available for overtaking; 5) The HV does not change lane unless for overtaking; 6) The HV does not brake to occupy the rear cell. 7) No collision is allowed.

Table 5.1: The selected features and the weights for reinforcement learning.

$\Phi(s, a)$	w_1	Interpretation	w_2	Interpretation
maintain	0	NA	0	NA
accelerate	0.075	Prefer accelerating	0.05	Prefer accelerating
brake	-0.625	Avoid braking	-0.5	Avoid braking
left-turn	-0.05	Reduce lane-shifting	-0.025	Reduce lane-shifting
right-turn	-0.05	Reduce lane-shifting	-0.025	Reduce lane-shifting
HV position	0	NA	0	NA
overtake	0.05	Prefer inner overtaking	0.025	Prefer inner tailgating
tailgate	0	NA	0.225	Prefer tailgating
collision	-0.15	Avoid collision	-0.15	Avoid collision

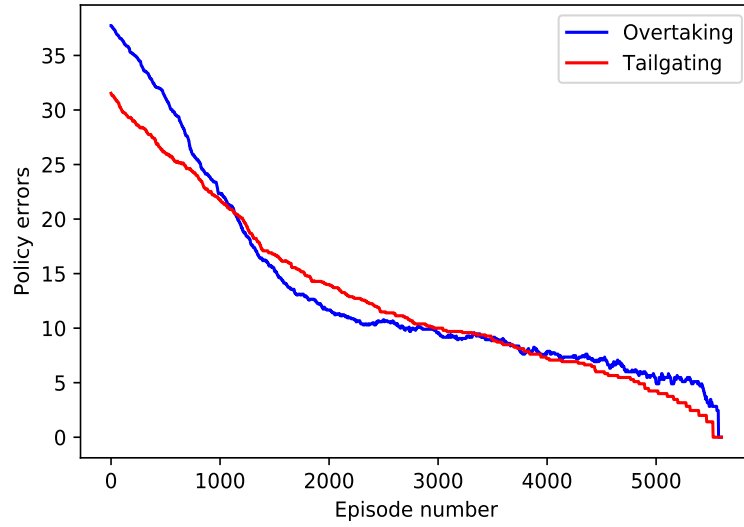


Figure 5.9: The convergence performance of the policy π in the learning process.

The desired driving behavior by designing w_2 is to demonstrate tailgating, which can be described as follows: 1) The HV maintains its velocity if there is an EV in front of it; 2) The HV accelerates to occupy the front cell if it is available and no tailgating will occur by changing lanes; 3) The HV changes lane to tailgate an EV if there is no EV in front of it; 4) The HV prefers to tailgate the vehicle in the lane closer to the inner curb of the road in a corner; 5) The HV does not change lanes unless for tailgating; 6) The HV does not brake to occupy the rear cell; 7) No collision is allowed.

We implemented the Q-learning algorithm (Algorithm 1 or 2) to learn the optimal policies using both w_1 and w_2 , with learning rate $\alpha = 0.75$, discount rate $\gamma = 0.5$, and $\epsilon = 8e^{-2}$ for the ϵ -greedy principle. Figure 5.9 shows the convergence behaviors of the policies π_1^* and π_2^* corresponding to w_1 and w_2 , respectively. One sees that, after 5,000 to 6,000 episodes the policies π_1 and π_2 get stabilized with the current setup of α , γ and ϵ . In both cases, it takes less than five minutes to obtain the results shown in Figure 5.9 on a dual-core 2.27 GHz Intel Xeon processor running 64-bit Windows 10 Enterprise operating system and programmed using Python 3.6.

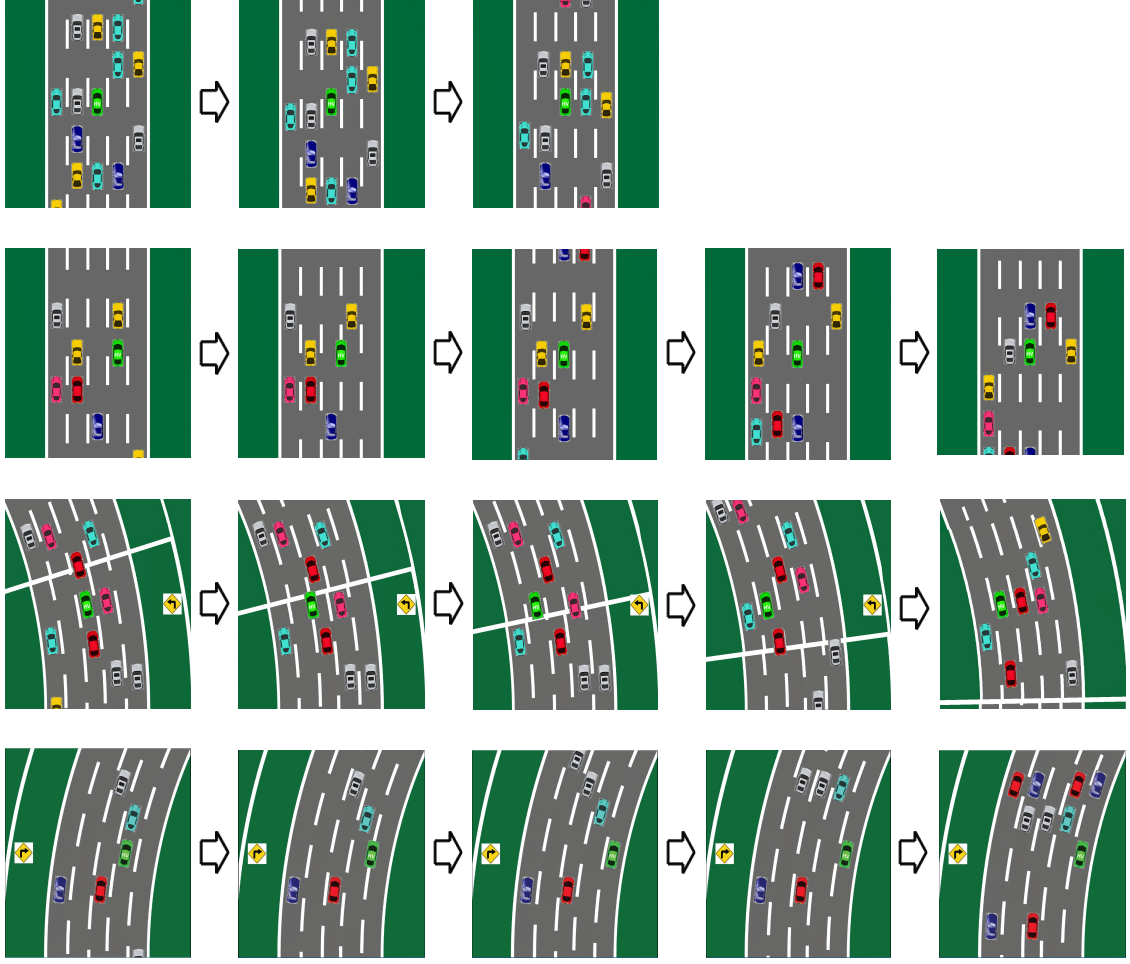


Figure 5.10: Overtaking scenarios in simulation by implementing π_1^* .

Next, we implemented the policies π_1^* and π_2^* in simulation. We show only the simulated result from the implementation of π_1^* (overtaking). The tailgating results by implementing the IRL algorithms are given in the next section. Figure 5.10 shows four different driving scenarios (each single row of pictures).

The first row of Figure 5.10 shows a scenario where there is a vacant space in front of the HV (green). The HV accelerates to occupy the front space and then maintains its distance behind the yellow vehicle. The second row shows a scenario where there is one vehicle in front of the HV and both the left and right lanes are available for the HV to overtake the front vehicle. Since the road is straight, the HV is free to use either the left or the right lane to complete the overtaking task. One sees from this figure that the HV first switches to the left lane, and then accelerates to overtake the front yellow vehicle, which switches to the right lane, until meeting the blue vehicle in the front. The third scenario shows one vehicle (red) in front of the HV but the right lane of the HV is occupied by another vehicle (pink). The HV can only use the left lane to overtake the front vehicle. The last scenario shows another driving scenario during cornering, which has one vehicle (cyan) in front of the HV and both the left and right lanes of the HV are

available to use for overtaking. The HV first switches to the right lane, which is closer to the inner curb of the corner, and then tries to overtake the cyan vehicle by accelerating. Overtaking is not completed since the cyan vehicle also moves forward. All these driving behaviors in the simulation using π_1^* agree with the desired behaviors, which validates the design of the reward function and the approach to find the optimal policy.

5.6.2 Driving Behavior from Inverse Reinforcement Learning

In this section we use the simulated data by implementing π_1^* and π_2^* to learn the reward functions represented using DNNs, and obtain the estimated policies $\hat{\pi}_1^*$ and $\hat{\pi}_2^*$ for π_1^* and π_2^* , respectively, using the maximum entropy principle.

We first select a structure of the DNN as shown in Figure 5.8 to represent the reward function. We use the second structure due to its convenience, since it uses only the state s_t as the input of the DNN and provides the rewards for taking every available action in the action set A . One then just needs to select the right output channel according to the current action a_t . The state vector s_t contains the information of the positions of nine vehicles (one HV and eight EVs) and the type of the road, hence, the DNN requires ten input channels to receive the ten-dimensional state s_t . The DNN requires five output channels since there are five actions in A . We define a DNN with the numbers of the neurons in each layer given by $[10, 20, 20, 20, 5]$, which has three hidden layers and each hidden layer has twenty neurons. We use the hyperbolic tangent activation function (tanh) instead of the sigmoid activation function, since the tanh function is unbiased at the origin, and it has a larger range $[-1, 1]$ than the sigmoid function which has a range $[0, 1]$.

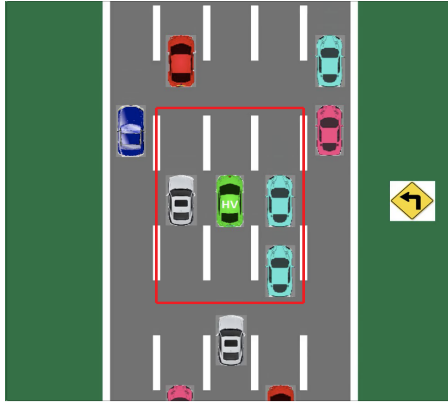


Figure 5.11: The initial setup for simulation.

Next, we collect simulated data by implementing the π_1^* and π_2^* learned in Section 5.6.1. The initial state s_0 is shown with the rectangular zone in Figure 5.11, where the HV is located in the middle lane of a five-lane road surrounding by three EVs.

Each EV in the simulator implements a random policy. For each EV, we define a nominal state s^{EV} using all the vehicles (HV and EVs) around it and find the safe actions for

this EV not leading to a collision. We then generate a random policy for this EV by assigning a random probability to each action in the safe action set. Algorithm 7 shows the procedures to generate a random policy π^{EV} for an EV in the simulation. For each EV, we only implement Lines 1-3 once, and we update the policy using Lines 4-8 when the safe action set and the state change. For either π_1^* and π_2^* , we collected 500 demonstrations with a fixed simulation period $T = 1,500$.

Algorithm 7 Generate EV policy

Input: s^{EV}

Output: π^{EV}

- 1: **Generate five random number between [0,1]:**
 - 2: $\pi_0^{\text{EV}} \leftarrow \text{random}(5)$
 - 3: $\pi^{\text{EV}} \leftarrow \text{sort}(\pi_0^{\text{EV}})$
 - 4: **Check availability of each action:**
 - 5: **for** $a \in A$ **do**
 - 6: **if** a is **not** available for s_t^{EV} **then**
 - 7: $\pi^{\text{EV}}(a|s^{\text{EV}}) \leftarrow 0$
 - 8: $\pi^{\text{EV}} \leftarrow \text{normalize}(\pi^{\text{EV}})$ **over** available actions
-

We then implemented the three MaxEnt IRL algorithms in Algorithms 4-6, respectively, to learn the policy using the simulated data. The parameters to setup the algorithms are as follows: learning rate for Q-learning $\alpha = 0.75$, discount rate $\gamma = 0.5$, DNN learning rate $\lambda = 5e^{-3}$, regularizer coefficient $\beta = -1e^{-4}$, and $\epsilon = 8e^{-2}$ in the ϵ -greedy principle. A summary of the results is shown in Table 5.2.

Table 5.2: IRL results summary.

	data length	Convergence	Time	Policy recovery
Algorithm 4	$T = 1500$	No	NA	NA
Algorithm 5	$\Delta T = 1$	Yes	1 hour	$\geq 99\%$
Algorithm 6	$\Delta T = 5$	Yes	1-3 hours	$\geq 99\%$

One notices from Table 5.2 that Algorithm 4 does not converge, due to the large data length and the stochastic system behavior. The proposed refined algorithms (Algorithm 5 and 6) provide better convergence performance and learn the policy effectively. In contrast to Algorithm 6, Algorithm 5 saves some running time since one skips the learning of the model and avoids calculating the expected state-action visitation counts.

We also implemented the learned policies $\hat{\pi}_1^*$ and $\hat{\pi}_2^*$ in simulation³. We show only the tailgating behavior by implementing $\hat{\pi}_2^*$ (see Figure 5.12), since the result by implementing $\hat{\pi}_1^*$ is similar with π_1^* , which has already been shown in Figure 5.10.

³Movies for both overtaking and tailgating by implementing $\hat{\pi}_1^*$ and $\hat{\pi}_2^*$ are available at: <https://www.youtube.com/watch?v=I3ecd9DXmBQ> and <https://www.youtube.com/watch?v=1VZcRR-Q2PE>

Figure 5.12 shows four driving scenarios. The first row shows a driving scenario having a vacant space in front of the HV and the HV cannot tailgate any EV by changing lanes. The HV accelerates to occupy the space behind the gray vehicle in front of it. The second driving scenario shows that the HV changes the lane to the left to tailgate the red vehicle. The third driving scenario is similar with the second, but it shows a driving behavior during cornering. The last driving scenario shows that there are two EVs in front of the HV in the neighboring lanes, and the HV can change to either the left or the right lane to tailgate an EV. By designing w_2 we have constructed a policy π_2^* to tailgate the EV closer to the inner curb of the road in a corner. One sees from this simulation that the HV changes to the left lane to tailgate the gray vehicle in a left-turn corner. All these driving behaviors using $\hat{\pi}_2$ agree with the desired tailgating behaviors we want to achieve using π_2^* , which validates the effectiveness of the IRL algorithms proposed in this dissertation.

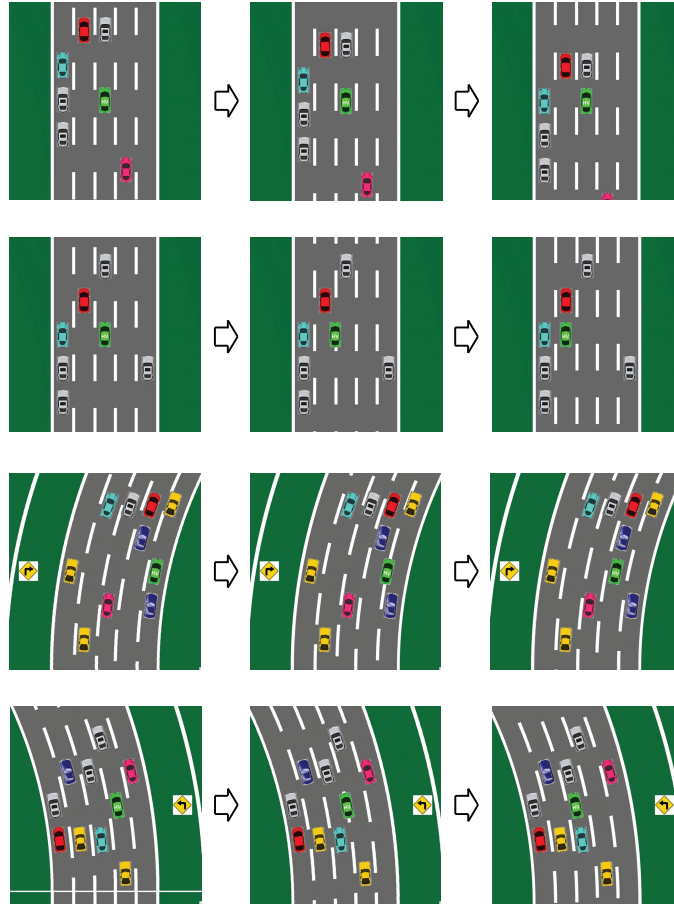


Figure 5.12: The tailgating in simulation by implementing $\hat{\pi}_2$.

5.7 Conclusion

We use a stochastic Markov decision process to model the traffic, and achieve desired driving behaviors using both reinforcement learning and inverse reinforcement learning. The definition of the state and the MDP traffic model are flexible and can be used

to model traffic with any number of lanes and any number of EVs. We also take the road geometry into consideration such that the driving policy may change depending on the road curvature.

By designing the driver’s reward function, we are able to show typical driving behaviors such as overtaking and tailgating, using the Q-learning algorithm to learn the corresponding optimal policies. We have demonstrated these policies using a road with five lanes and with each EV implementing a random policy. In order to be able to recover the policy and the reward function from data, we propose three new model-free inverse reinforcement learning algorithms based on the maximum entropy principle. Instead of using a state reward as in most of existing literature [77, 78, 82], we use a state-action reward, which is capable for the design of more diverse driving behaviors. This is the first work to generalize the formulation of the maximum entropy inverse reinforcement learning problem with any parameterized, continuously differentiable function approximators (i.e., a DNN). In order to refine the inverse reinforcement learning algorithm, we show that long demonstrations are hard to use for this problem if one has limited knowledge of the (stochastic) system behavior. The error stems from two factors: First, the capacity of the demonstrated data may not be enough to represent the stochastic behavior of the system. Second, the prediction error for a stochastic system is accumulated and becomes large for long term prediction horizons in a model-free problem. We refine our inverse reinforcement learning algorithm by maximizing the entropy of the joint distribution over short data pieces. The proposed algorithms are validated in simulation.

Future work will focus on designing the necessary controls to achieve the driving behavior in a high-fidelity simulation or a real driving task. Other possible extensions introduce multiple agents incorporated into the MDP traffic model to coordinate multiple vehicles simultaneously to better control the traffic flow (i.e., traffic congestion mitigation).

CHAPTER 6

PATH PLANNING AND CONTROL: HIGHWAY OVERTAKING

6.1 Introduction

In Chapter 5 we solve the high level optimal decision making problem for autonomous vehicles in traffic. In order to realize the optimal action, one still needs to solve low level path planning problem and develop the necessary control system. Numerous algorithms have been developed over the past decade for real-time path planning for autonomous vehicles, which can be categorized into three groups according to the methodology used to develop them, namely, sampling-based [48, 175], graph-search methods [176, 61], and geometry-based path planning [62, 63, 64].

Cimurs et al. used Dijkstra's algorithm to find the shortest viable path by connecting the Voronoi vertices, such that the path keeps a safe distance from all the obstacles in the environment [61]. They then used the Bézier curves to smooth the path with respect to the maximum curvature constraint by selecting and aligning the control points. The authors in [48] proposed a real-time path planning algorithm based on Rapidly-exploring Random Trees (RRTs). This algorithm was implemented on an autonomous vehicle which completed a 60 mile simulated military supply mission in the 2007 DARPA Urban Challenge. The path planning approaches using RRTs can efficiently explore the space to handle obstacle avoidance problems. Nevertheless, since the tree is built incrementally from the direction of the samples randomly from the search space, an additional smoother may be required to smooth the path. In order to generate a smooth path for an autonomous vehicle without using an additional smoother, Choi et al. [62, 63] presented a series of path planning algorithms based on the Bézier curves. The planned paths have continuous curvature and satisfy the road boundary constraints. In [64] Shim et al. used a parameterized 6th-order polynomial to represent a smooth path, and planned a feasible path for the autonomous vehicle satisfying both the initial/final conditions and the constraint conditions. They implemented their path planning algorithm in static/moving obstacle avoidance tasks. A more extensive survey on path planning for autonomous vehicles can be found in [73, 36].

Built on the work in Chapter 5, this chapter focuses on developing low level path planning and control algorithms for autonomous vehicles in highway traffic. The work of this chapter is summarized as follows: 1) We propose two path planning algorithms using both joint quadratic Bézier curves and fourth order Bézier curves. The optimal choices of the control points are provided in terms of the desired maximum curvature, hence a path satisfying the given curvature constraints can be generated in real-time. The fourth order Bézier curves are C^2 continuous and show better tracking performance when an output regulation tracking controller is implemented. 2) We design controller for path tracking/lane-keeping using three different ideas: a) driver parameters optimization, b) single output regulation controller and c) driver-based ADAS controller.

The rest of this chapter is organized as follows: Section 6.2 generates the smooth paths using Bézier curves. Section 6.3 and 6.4 design the low-level controllers for speed maintaining and lane keeping/switching. Section 6.5 implements the controllers on a simulator. Finally, Section 6.6 summarizes the results of this study.

6.2 Path Planning

Regarding to path planning for lane-switching, one important design objective is to limit the maximum curvature of the path, which depends on the road friction conditions and the velocity of the vehicle. We also expect the curvature to be continuous in order to have better smoothness and riding comfort. In this work we use Bézier curves for path planning during lane-switching. Such designed path is smooth and can be implemented efficiently for real-time planning.

6.2.1 Preliminaries

Bézier curves were introduced by Pierre Bézier in 1960 [177]. Since then Bézier curves become widely used in computer graphics and related fields. A Bézier curve of degree n with control points P_0, \dots, P_n is expressed in terms of Bernstein polynomials, which are given by

$$\gamma(t) = \sum_{i=0}^n B_i^n(t) P_i, \quad t \in [0, 1], \quad (6.1)$$

where

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad (6.2)$$

and where the binomial coefficients are

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n, \\ 0 & \text{otherwise.} \end{cases} \quad (6.3)$$

The curve $\gamma(t)$ in (6.1) is completely contained in the convex hull of its control points P_0, \dots, P_n . It is worth mentioning that the curve is tangent to the segments P_0P_1 and $P_{n-1}P_n$ at both the two ends. By designing the control points, one is able to reshape the curve to have the desired features.

6.2.2 Joint Quadratic Bézier curves

Built on the result of [178], we use piecewise quadratic Bézier curves to plan the path for lane switching.

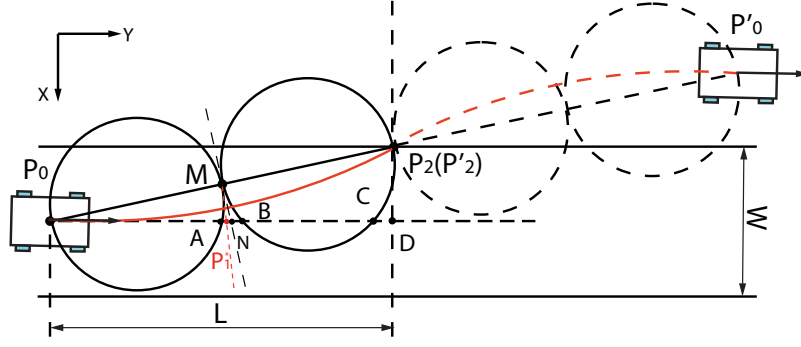


Figure 6.1: Path planning for the single lane change.

Let us assume that we have the following path planning problem to solve (see Fig. 6.1). The lane width is denoted by W . Without loss of generality, we can assume the trajectory is symmetric with respect to the point P_2 , which is located at a distance L in front of the vehicle. The quadratic Bézier curve γ is given by

$$\gamma(P_1, t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2, \quad t \in [0, 1]. \quad (6.4)$$

Mathematically, we want to solve the following problem,

$$P_1^* = \arg \min_{P_1 \in \overline{P_0 D}} \max_{t \in [0, 1]} \kappa(P_1, t) = \frac{|\gamma'(P_1, t) \times \gamma''(P_1, t)|}{\|\gamma'(P_1, t)\|^3}, \quad (6.5)$$

where κ denotes the curvature. Let M denote the midpoint of segment $\overline{P_0 P_2}$. We provide Theorem 6.2.1 to determine the optimal solution P_1^* .

Theorem 6.2.1 *Let N be the intersection of $\overline{P_0 D}$ and the perpendicular bisector to the segment $\overline{P_0 P_2}$ (see Fig. 6.1). The optimal choice of P_1^* is on the segment \overline{AN} , which satisfies $\angle NMP_1^* = \tan^{-1} \frac{\sqrt{8 \tan^2 \angle P_2 P_0 D + 9} - 3}{2 \tan \angle P_2 P_0 D}$. The minimal maximum curvature $\bar{\kappa}^*$ is given by*

$$\bar{\kappa}^*(P_1^*) = \frac{3 \tan \angle NMP_1^*}{\|P_0 M\| \cos \angle NMP_1^*}. \quad (6.6)$$

The maximum curvature $\bar{\kappa}(P_1)$ decreases monotonically as P_1 moves from P_0 to P_1^ , and then increases monotonically as P_1 moves from P_1^* to D .*

Proof: 1) P_1 is on $\overline{P_0 A}$: Since P_1 is located inside the disc of diameter $\|P_0 M\|$, and $\|P_0 P_1\| < \|P_1 P_2\|$, the maximum curvature is determined by

$$\bar{\kappa}(P_1) = \frac{\mathcal{A}}{\|P_0 P_1\|^3} = \frac{W}{2\|P_0 P_1\|^2}. \quad (6.7)$$

The result in (6.7) indicates that $\bar{\kappa}(P_1)$ decreases when P_1 approaches A from the direction of P_0 , and the minimum value is given by $\bar{\kappa}(A) = 2W/L^2$.

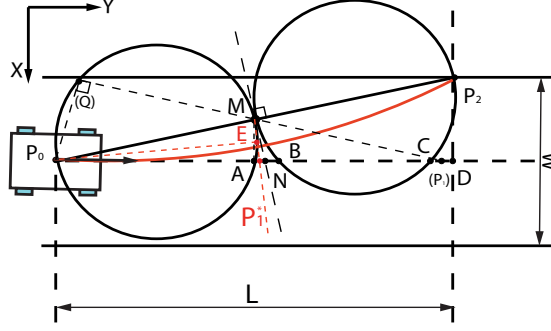


Figure 6.2: Path planning for the single lane change.

2) P_1 is on \overline{BC} : Since P_1 is located inside the disc of diameter $\|P_2M\|$ and $\|P_0P_1\| > \|P_1P_2\|$, the maximum curvature is determined by

$$\bar{\kappa}(P_1) = \frac{\mathcal{A}}{\|P_1P_2\|^3} = \frac{W\|P_0P_1\|}{2\|P_1P_2\|^3}. \quad (6.8)$$

When P_1 moves from B to C , $\|P_0P_1\|$ increases monotonically while $\|P_1P_2\|$ decreases monotonically. Hence $\bar{\kappa}(P_1)$ increases monotonically and the minimum value of $\bar{\kappa}(P_1)$ is obtained at B .

3) P_1 is on \overline{CD} : Since P_1 is outside of the two discs, the maximum curvature is determined by

$$\bar{\kappa}(P_1) = \frac{\|P_1M\|^3}{\mathcal{A}^2} = \frac{\|P_1M\|}{(\|P_0P_1\| \sin \angle MP_1P_0)^2} = \frac{\|P_1M\|}{\|P_0Q\|^2}, \quad (6.9)$$

where Q is the intersection of line \overline{CM} and the first disc (see Fig. 6.2). One can observe that $\|P_0Q\|$ reduces monotonically as P_1 moves from C to D (this holds even if P_1 is on the right of D till infinity). The minimum value of $\bar{\kappa}(P_1)$ is obtained at C .

4) P_1 is on \overline{NB} : The proof is similar with 3) and is hence omitted. In this case, $\bar{\kappa}(P_1)$ increases monotonically as P_1 approaches B from N . The minimum value of $\bar{\kappa}(P_1)$ is obtained at N .

The results from 2) to 4) imply that $\bar{\kappa}(N) < \bar{\kappa}(B) < \bar{\kappa}(C) < \bar{\kappa}(D)$, and hence $\bar{\kappa}$ increases monotonically as P_1 moves from N to D .

5) P_1 is on \overline{AN} : We connect P_1 and M and denote the intersection of $\overline{P_1M}$ and the circle as E . We then connect P_0E . In order to simplify the notation, we use α to denote the angle $\angle MP_0E$ ($\angle NME$), and use θ to denote the angle $\angle MP_0D$. The maximum curvature is determined by

$$\begin{aligned} \bar{\kappa}(\alpha) &= \frac{\|P_1M\|^3}{\mathcal{A}^2} = \frac{\|P_1M\|}{\|P_0E\|^2} = \frac{\tan \alpha + \tan(\theta - \alpha)}{\|P_0M\| \cos \alpha} \\ &= \frac{\sin \theta}{\|P_0M\| \cos^2 \alpha \cos(\theta - \alpha)}. \end{aligned} \quad (6.10)$$

By taking the partial derivative of $\bar{\kappa}$ with respect to partial derivative of α , one obtains

$$\frac{\partial \bar{\kappa}(\alpha)}{\partial \alpha} = \frac{\sin \theta}{\|P_0 M\|} \frac{2 \sin \alpha \cos(\theta - \alpha) - \cos \alpha \sin(\theta - \alpha)}{\cos^3 \alpha \cos^2(\theta - \alpha)}. \quad (6.11)$$

By letting $\partial \bar{\kappa}(\alpha) / \partial \alpha = 0$, we obtain the following result:

$$\tan(\theta - \alpha^*) = 2 \tan \alpha^*. \quad (6.12)$$

Equation (6.12) implies that $\|P_1^* E\| = 2\|ME\|$. We can solve (6.12) for α^* , which yields

$$\alpha^* = \tan^{-1} \frac{\sqrt{8 \tan^2 \theta + 9} - 3}{2 \tan \theta}. \quad (6.13)$$

Equations (6.11)-(6.13) imply that $\bar{\kappa}(P_1^*)$ is the minimum on \overline{AN} , and hence it is the unique minimum on $\overline{P_0 D}$. \square

The path planning algorithm is summarized in Algorithm 8.

Algorithm 8 Path Generation Using Joint Quadratic Bézier Curves

Input: $W, P_0, \bar{\kappa}_{\max}$

Output: $L, \gamma(t)$

1: $\angle P_2 P_0 D, \angle N M P_1^* \leftarrow$ by solving:

$$\begin{cases} \angle N M P_1^* = \tan^{-1} \frac{\sqrt{8 \tan^2 \angle P_2 P_0 D + 9} - 3}{2 \tan \angle P_2 P_0 D}, \\ \frac{3 \tan \angle N M P_1^*}{\|P_0 M\| \cos \angle N M P_1^*} - \bar{\kappa}_{\max} = 0, \\ \|P_0 M\| = \frac{W}{2 \sin \angle P_2 P_0 D}. \end{cases}$$

2: $L \leftarrow W / \tan \angle P_2 P_0 D, \angle A M P_1^* \leftarrow \angle P_2 P_0 D - \angle N M P_1^*$

3: $P_2 \leftarrow (\pm W/2, L), P_1 \leftarrow P_0 + (0, L/2 + W \tan \angle A M P_1^* / 4)$

4: $\zeta(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2, \quad t \in [0, 1]$

5: Curve: $\gamma(\tau) = \begin{cases} \zeta(2\tau) & \tau \in [0, 0.5), \\ (\pm W, 2L) - \zeta(2-2\tau) & \tau \in [0.5, 1]. \end{cases}$

6.2.3 Fourth Order Bézier Curves

The joint quadratic Bézier curves are easy to implement for path planning. Nevertheless, this approach only guarantees C^1 continuity and the curvature is not continuous at both P_0 and P_2 . In this section, we use fourth order Bézier curves to generate paths with continuous curvature. A typical fourth order Bézier curve is constructed using five control points, namely, P_0, \dots, P_4 , and is represented by

$$\gamma(t) = \sum_{i=0}^4 B_i^4(t) P_i, \quad t \in [0, 1]. \quad (6.14)$$

In order to generate a symmetric path (see Fig. 6.3), we let $P_2 = (P_0 + P_4)/2$, and $\vec{\Delta} = P_1 - P_0 = P_4 - P_3$. Equation (6.14) is simplified as

$$\gamma(t) = (1-t)^2(1+2t)P_0 + 4t(1-t)(1-2t)\vec{\Delta} + t^2(3-2t)P_4. \quad (6.15)$$

To calculate the curvature of $\gamma(t)$, we first calculate $\gamma'(t)$ and $\gamma''(t)$,

$$\gamma'(t) = 4 \sum_{i=0}^3 B_i^3(t)(P_{i+1} - P_i) = 4\vec{\Delta} + 12t(1-t)\vec{\Gamma}, \quad (6.16a)$$

$$\gamma''(t) = 12 \sum_{i=0}^2 B_i^2(t)(P_{i+2} - 2P_{i+1} + P_i) = 12(1-2t)\vec{\Gamma}, \quad (6.16b)$$

where $\vec{\Gamma} \triangleq (P_4 - P_0)/2 - 2\vec{\Delta}$. We then calculate the curvature as follows,

$$\kappa(t) = \frac{|\gamma'(t) \times \gamma''(t)|}{\|\gamma'(t)\|^3} = \frac{24(1-2t)|\vec{\Delta} \times (P_4 - P_0)|}{\|\gamma'(t)\|^3}. \quad (6.17)$$

From (6.17) one sees that $\kappa(0.5) = 0$ and $\kappa(0) = -\kappa(1) = 3|\vec{\Delta} \times (P_4 - P_0)|/8\|\vec{\Delta}\|^3$ (property of symmetry). Hence, we can just analyze the first half of the curve by letting $t \in [0, 0.5]$ since the curve is symmetric. Taking the partial derivative of $\kappa(t)$ with respect to t yields

$$\kappa'(t) = 24|\vec{\Delta} \times (P_4 - P_0)| \frac{F(t)}{\|\gamma'(t)\|^5}, \quad t \in [0, 0.5]. \quad (6.18)$$

where $F(t) = -2\|\gamma'(t)\|^2 - 3(1-2t)\gamma'(t) \cdot \gamma''(t)$. It is easy to see that the sign of $\kappa'(t)$ is the same as $F(t)$. We then simplify this equation, to obtain

$$F(x(t)) = 16 \left(90\|\vec{\Gamma}\|^2 x(t)^2 - (27\|\vec{\Gamma}\|^2 - 24\vec{\Delta} \cdot \vec{\Gamma})x(t) - 2\|\vec{\Delta}\|^2 - 9\vec{\Delta} \cdot \vec{\Gamma} \right). \quad (6.19)$$

where $x(t) = t(1-t) \in [0, 0.25]$ for $t \in [0, 0.5]$ and $\vec{\Gamma} \triangleq (P_4 - P_0)/2 - 2\vec{\Delta}$.

Theorem 6.2.2 *The curvature of the fourth order Bézier curve $\kappa(t)$ is monotonically decreasing from $t = 0$ to $t = 1$ if and only if $\|\vec{\Delta}\| \leq 9L/32$. The maximum curvature is given by $\kappa(0) = 3|\vec{\Delta} \times (P_4 - P_0)|/8\|\vec{\Delta}\|^3$.*

Proof: We just need to show the monotonicity on $t \in [0, 0.5]$ due to symmetry. To this end, we can equivalently show that $F(x(t)) \leq 0$ for $x(t) \in [0, 0.25]$ and $\|\vec{\Delta}\| \leq 9L/32$. Since $F(x(t))$ is parabolic and the coefficient of the second order term is positive, we need to ensure that $F(x(t)) \leq 0$ at the two endpoints of the interval of $x(t)$. Hence,

$$F(x = 0.25) = -\|3\sqrt{2}\vec{\Gamma} + 4\sqrt{2}\vec{\Delta}\|^2 \leq 0, \quad (6.20a)$$

$$F(x = 0) = -8 \left(9L\|\vec{\Delta}\| - 32\|\vec{\Delta}\|^2 \right) \leq 0, \quad \forall \|\vec{\Delta}\| \leq 9L/32. \quad (6.20b)$$

and the result follows. \square

For the case that $\|\vec{\Delta}\| > 9L/32$, the maximum curvature is obtained at $\gamma'(t^*) = F'(x^*(t^*)) = 0$, where

$$x^*(t^*) = t^*(1 - t^*) = \frac{9\|\vec{\Gamma}\|^2 - 8\vec{\Delta} \cdot \vec{\Gamma} - \sqrt{81\|\vec{\Gamma}\|^4 + 64(\vec{\Delta} \cdot \vec{\Gamma})^2 + 216\|\vec{\Gamma}\|^2(\vec{\Delta} \cdot \vec{\Gamma}) + 80\|\vec{\Delta}\|^2\|\vec{\Gamma}\|^2}}{60\|\vec{\Gamma}\|^2}. \quad (6.21)$$

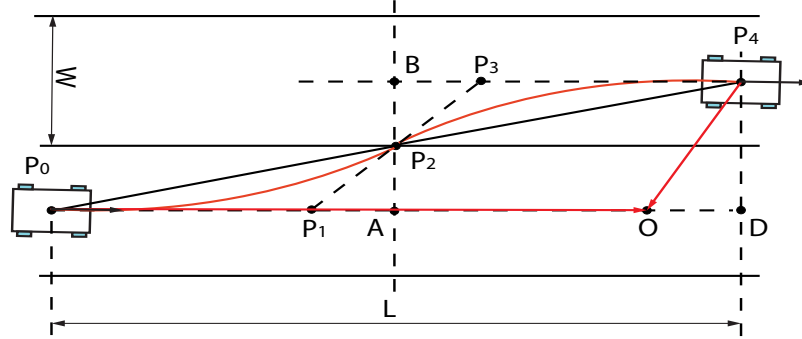


Figure 6.3: A symmetric fourth order Bézier curve.

Theorem 6.2.3 *The fourth order Bézier curve $\kappa(t)$ has the minimal jerk energy when $\|\vec{\Delta}^*\| = L/4$, where $\vec{\Delta}^*$ mathematically solves the following minimization problem*

$$\vec{\Delta}^* = \operatorname{argmin}_{\vec{\Delta}} E(\gamma) = \int_0^1 \|\gamma'''(\vec{\Delta}, t)\|^2 dt. \quad (6.22)$$

Proof: It follows from the equation (3.2) in [177] that, the jerk energy of the fourth order Bézier curve generated using control points P_0, \dots, P_4 can be represented by

$$E(\gamma) = 3350\|Q_0\|^2 + 1440\|Q_1\|^2, \quad (6.23)$$

where $Q_0 = P_0 - 4P_1 + 6P_2 - 4P_3 + P_4$ and $Q_1 = -P_0 + 2P_1 - 2P_3 + P_4$. Recall that $P_2 = (P_1 + P_4)/2$, $P_1 = P_0 + \vec{\Delta}$ and $P_3 = P_4 - \vec{\Delta}$, then one can simplify the expressions for Q_0 and Q_1 , which are given by $Q_0 = (0, 0)$, $Q_1 = 4\vec{\Delta} - (P_4 - P_0)$. Hence, in order to minimize $E(\gamma)$ we just need to minimize $\|Q_1\|$. The minimal $\|Q_1\|$ ($|P_4O|$) is obtained under the condition that $\vec{P_4O} \perp \vec{\Delta}$ (see Fig. 6.3), which represents the minimal distance from P_4 to P_0D . This result indicates $\|\vec{\Delta}^*\| = L/4$. \square

We notice that the curvature of the fourth order Bézier curve is not zero at both the two endpoints P_0 and P_4 (see Fig. 6.3). Consequentially, the transition between the Bézier curve and the straight line is not smooth. One may consider to use clothoids to smooth the transition. However, clothoid computation takes time. This dissertation proposes another method by extending the fourth order Bézier curve in order to obtain zero curvature at both the two endpoints.

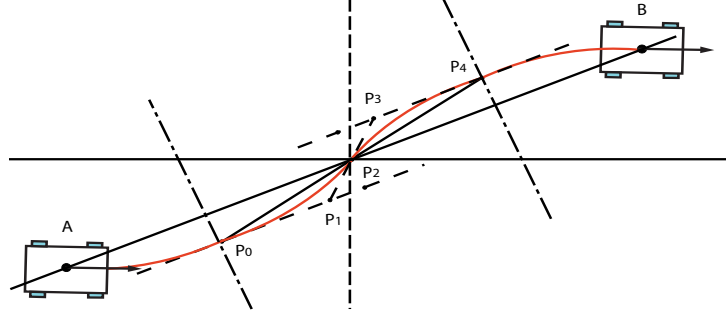


Figure 6.4: Bézier curve reconstruction for smooth transition at endpoints.

Algorithm 9 Path Generation Using 4th Order Bézier Curves

Input: $W, A, \bar{\kappa}_{\max}$

Output: $B, L, \gamma(t)$

- 1: $K \leftarrow \sqrt{\left(\sqrt{1 + \frac{144}{W^2 \bar{\kappa}_{\max}^2}} - 1\right)/2}$
 - 2: $L \leftarrow KW, d \leftarrow \frac{\sqrt{1+K^2}}{4K} W, \ell \leftarrow 2Kd, B \leftarrow A + (\pm W, L)$
 - 3: $\theta \leftarrow \arctan \frac{\ell}{2d}, \alpha \leftarrow 2\theta - \pi/2$
 - 4: $\vec{\Delta} \leftarrow (0, \ell/4)$ (Minimal jerky solution)
 - 5: $P_0 \leftarrow (0, 0), P_4 \leftarrow (\pm d, \ell), P_2 \leftarrow (P_0 + P_4)/2, P_1 \leftarrow P_0 + \vec{\Delta}, P_3 \leftarrow P_4 - \vec{\Delta}$
 - 6: $\hat{\gamma}(t) = \sum_{i=0}^4 B_i^4(t) P_i, t \in [0, 1]$
 - 7: $\vec{i} \leftarrow (1, 0)$
 - 8:
$$\zeta(\tau) = \begin{cases} 2(\hat{\gamma}(0.5 - 2\tau) \cdot \vec{i}, 0) - \hat{\gamma}(0.5 - 2\tau) & \tau \in [0, 0.25], \\ \hat{\gamma}(2\tau - 0.5) & \tau \in [0.25, 0.75], \\ 2(\hat{\gamma}(2.5 - 2\tau) \cdot \vec{i}, \ell) - \hat{\gamma}(2.5 - 2\tau) & \tau \in (0.75, 1]. \end{cases}$$
 - 9: Translation: $\zeta(\tau) \leftarrow \zeta(\tau) + A + (\mp d/2, \ell/2)$
 - 10: Rotation: $\gamma(t) \leftarrow \zeta(\tau)$ rotated about A by $\mp(\pi/2 - \theta)$
-

One notices that the curvature at the midpoint of a symmetric fourth order Bézier curve is always zero. The curvature of the curve between P_0 and P_2 changes gradually from $\kappa(0)$ to zero (similar as a clothoid). Based on this fact, one can use half of the Bézier curve to smooth the path, as shown in Fig. 6.4. We reflect the curve segments $\widehat{P_0 P_2}$ and $\widehat{P_2 P_4}$ about the axes passing through P_0 and P_4 , respectively. Such designed path from A to B is everywhere C^2 continuous, with the curvature $\kappa(A) = \kappa(B) = 0$ at the endpoints. We summarize the path planning algorithm using fourth order Bézier curves subject to the given maximum curvature constraints $\bar{\kappa}_{\max}$ in Algorithm 9.

6.3 Speed Control

Except lane-switching, the other actions such as maintaining, accelerating and braking are taking place in a single lane. These actions can be completed by maintaining the

longitudinal speed of the vehicle, associated with implementing certain lane tracking controllers (i.e., waypoint follower, two-point visual driver model, etc.). Here, we design the speed control and lane-switching control separately.

Speed control is mainly designed to achieve the maintaining, accelerating and braking actions. Based on this fact, one may assume that the vehicle behind will always yield to the vehicle in front. Consequentially, we only need to check the speed of the vehicle in front in order to decide the maximum allowed speed of the vehicle we want to control.

Maintaining: The design objective for maintaining control is to maintain certain constant speed or certain constant distance from the vehicle in front.

Accelerating: The design objective for accelerating control is to speed up the vehicle and maintain certain high speed afterwards if there is speed limit. The target speed of the vehicle in steady state should not exceed the speed of the front vehicle.

Braking: The design objective for braking control is to slow down the vehicle and maintain certain low speed. Under the assumption that every vehicle in traffic will yield to the vehicle in front, there is no need to consider the speed of the vehicle behind before taking braking action. Nevertheless, we still have to maintain the minimum safe distance from the front vehicle during braking.

We only show the controller design for constant distance maintaining from the front vehicle. We assume that ΔL is the desired distance between vehicle A and B .

We let $e_1 = Y_A - Y_B - \Delta L$, $e_2 = V_A - V_B$. The error dynamics are given by

$$\dot{e}_1 = e_2, \quad \dot{e}_2 = \dot{V}_A - \dot{V}_B = \lambda_1 e_1 + \lambda_2 e_2, \quad (6.24)$$

By designing λ_1 and λ_2 , one can drive the errors $e_1 \rightarrow 0$ and $e_2 \rightarrow 0$ as time $t \rightarrow \infty$. Now let us assume that the front wheel steer angle is small and that the vehicle's lateral motion can be neglected during the distance maintaining task. For the vehicles having a rear wheel drive differential type, the longitudinal dynamics can be simplified to

$$m_B \dot{V}_B = f_{Rx}. \quad (6.25)$$

Let $\dot{V}_B = \dot{V}_A - \lambda_1 e_1 - \lambda_2 e_2$. Since the tire force must be bounded due to the friction condition, we define the following saturation function,

$$\text{sat}(\dot{V}_B) = \begin{cases} \bar{f}_{Rx}/m_B, & f_{Rx} \geq \bar{f}_{Rx}, \\ \dot{V}_A - \lambda_1 e_1 - \lambda_2 e_2, & \underline{f}_{Rx} < f_{Rx} < \bar{f}_{Rx}, \\ \underline{f}_{Rx}/m_B, & f_{Rx} \leq \underline{f}_{Rx}, \end{cases} \quad (6.26)$$

where \bar{f}_{Rx} and \underline{f}_{Rx} denote the upper and lower bounds for f_{Rx} (to be discussed later). The wheel dynamics is given as follows,

$$I_w \dot{\omega} = T_R - f_{Rx} R_w, \quad (6.27)$$

where T_R is the propulsion torque on the rear wheel, and I_w and R_w are the rotary

inertia and the radius of the rear wheel, respectively. Under the assumption for the zero-slip rolling condition, the equation $V_B = \omega R_w$ holds. The control torque is determined as follows,

$$T_{\text{R}} = \text{sat}(\dot{V}_{\text{B}}) R_{\text{W}} \left(m_{\text{B}} + I_{\text{W}} / R_{\text{W}}^2 \right). \quad (6.28)$$

Next, we discuss how to determine \bar{f}_{Rx} and $\underline{f}_{\text{Rx}}$ if the lateral velocity V_y and the longitudinal velocity V_x (approximated by V_B) are given. The longitudinal load transfer arising from the longitudinal acceleration indicates the following equation,

$$f_{\text{Rx}} = \mu_{\text{Rx}} f_{\text{Rz}}, \quad f_{\text{Rz}} = \frac{f_{\text{Rx}} h + m g \ell_{\text{f}}}{\ell_{\text{f}} + \ell_{\text{r}}}, \quad (6.29)$$

where h is the height of the mass center and $\mu_{\text{Rx}} = -\mu_{\text{R}} s_{\text{Rx}}/s_{\text{R}}$. It follows from (6.29) that

$$f_{\text{Rx}} = \frac{mg\ell_{\text{f}}}{(\ell_{\text{f}} + \ell_{\text{r}})/\mu_{\text{Rx}} - h}. \quad (6.30)$$

Based on the fact that the wheel base $\ell_f + \ell_r$ is larger than the height of the mass center h , the maximum longitudinal tire force \bar{f}_{Rx} ($\underline{f}_{\text{Rx}}$) is obtained when μ_{Rx} reaches its maximum value $\bar{\mu}_{\text{Rx}}$ ($\underline{\mu}_{\text{Rx}}$).

Furthermore, we assume that the lateral sideslip is small, and for the sake of simplicity, we assume that the total slip of the rear tire does not exceed s_R^* , where s_R^* corresponds to the peak of μ_{R_x} . This assumption indicates that μ_R increases monotonically with $|s_R|$. Based on the definition of the slip ratio in (4.6), we can derive the following equation

$$s_{\text{Ry}} = (1 + s_{\text{Rx}}) \frac{V_{\text{Ry}}}{V_{\text{Rx}}} \approx (1 + s_{\text{Rx}}) \tan \beta. \quad (6.31)$$

In order to find $\bar{\mu}_{\text{Rx}}$ ($\underline{\mu}_{\text{Rx}}$), we plot the slip ratio circle in Figure 6.5. The norm of the red

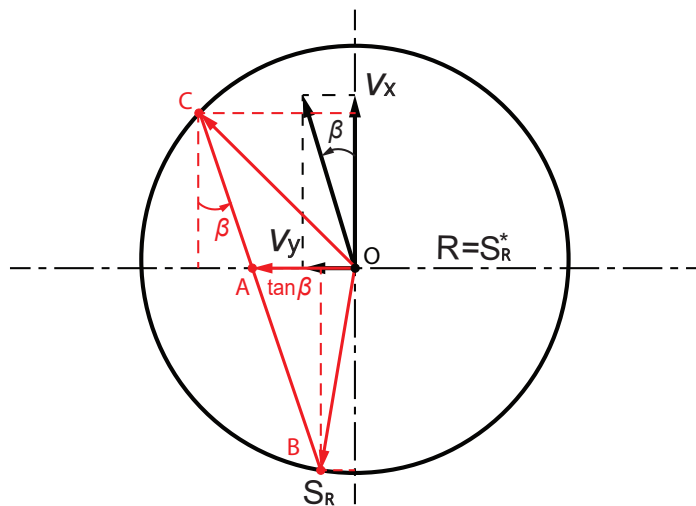


Figure 6.5: Slip ratio circle.

arrow starting from O denotes $|s_R|$. The equation in (6.31) indicates that, when the value of s_{Rx} changes, the arrow representing s_R moves along the segment \overline{BC} . When $s_{Rx} = 0$, $|s_R| = |s_{Ry}| = \tan \beta$ (for $\beta > 0$).

Recall that $\mu_{Rx} = -\mu_R s_{Rx} / s_R$, since both $-s_{Rx} / s_R$ and μ_R reaches their maximum values at B , the upper bound $\bar{\mu}_{Rx}$ is therefore obtained at B (maximal acceleration force). Similarly, the lower bound $\underline{\mu}_{Rx}$ is obtained at C (maximal braking force). The results are summarized as follows

$$\bar{f}_{Rx} = \frac{mg\ell_f\mu_R^*\left(\sqrt{\sec^2\beta s_R^{*2} - \tan^2\beta} + \tan^2\beta\right)}{(\ell_f + \ell_r)s_R^*\sec^2\beta - h\mu_R^*\left(\sqrt{\sec^2\beta s_R^{*2} - \tan^2\beta} + \tan^2\beta\right)}, \quad (6.32a)$$

$$\underline{f}_{Rx} = -\frac{mg\ell_f\mu_R^*\left(\sqrt{\sec^2\beta s_R^{*2} - \tan^2\beta} - \tan^2\beta\right)}{(\ell_f + \ell_r)s_R^*\sec^2\beta + h\mu_R^*\left(\sqrt{\sec^2\beta s_R^{*2} - \tan^2\beta} - \tan^2\beta\right)}, \quad (6.32b)$$

where μ_R^* is the peak friction coefficient, namely, the magnitude D of the magic formula (4.7), and $s_R^* = (1/B)\tan(\pi/2C)$ (let $S_h = S_v = E = 0$).

It is worth mentioning that the results in (6.32a)-(6.32b) only guarantee suboptimality. In order to find the optimal solution, one can take partial derivative of μ_{Rx} with respect to s_{Rx} ,

$$\frac{\partial \mu_{Rx}}{\partial s_{Rx}} = -\frac{\mu_R}{s_R} + \left(\frac{s_{Rx}}{s_R^2}\mu_R - \frac{s_{Rx}}{s_R}\frac{\partial \mu_R}{\partial s_R}\right)\frac{\partial s_R}{\partial s_{Rx}} = 0, \quad (6.33)$$

which can be solved numerically. The results in (6.32a)-(6.32b) are sufficiently accurate for this work.

6.4 Lane-Switching Control

We generate a smooth path for lane switching using Algorithm 8 or 9. We still need to design the tracking controller to follow the Bézier curve. This section designs tracking controls using an optimal two-point visual driver model and the output regulation theory.

6.4.1 Optimal Driver Model

We simplify the driver model in Figure 3.1 by removing of the kinesthetic perception feedback since it is not detrimental to the model accuracy and can be neglected [59], which is shown in Figure 6.6. The mathematical model of the driver subsystem is then formulated as

$$\dot{x}_d = A_d x_d + B_d u_d, \quad (6.34)$$

$$y_d = C_d x_d + D_d u_d, \quad (6.35)$$

where the state is $x_d = (x_{d1}, x_{d2}, T_{dr})^T$, the input is $u_d = (\theta_{near}, \theta_{far})^T$ and the output is $y_d = T_{dr}$. The system matrices in (6.34) are given explicitly by

$$A_d = \begin{bmatrix} -\frac{1}{T_L} & 0 & 0 \\ \frac{4}{t_p} & \frac{2}{T_L} & 0 \\ -\frac{1}{T_N} & \frac{1}{T_N} & -\frac{1}{T_N} \end{bmatrix}, \quad B_d = \begin{bmatrix} K_c \frac{T_L - T_L}{T_L^2} & 0 \\ 4 \frac{K_c T_L}{t_p T_L} & 4 \frac{K_a}{t_p} \\ -\frac{K_c T_L}{T_N T_L} & -\frac{K_a}{T_N} \end{bmatrix},$$

$$C_d = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, \quad D_d = \begin{bmatrix} 0 & 0 \end{bmatrix},$$

where K_a , K_c , T_L , T_I , T_N and t_p are the six tunable parameters of this driver model. The model of the overall vehicle-driver-perception system in Figure 3.1 is summarized by

$$\dot{x} = Ax + Bu, \quad (6.36)$$

$$y = Cx, \quad (6.37)$$

where $x = (\omega_s, \delta_s, \beta, r, \phi, \Delta y, x_{d1}, x_{d2}, T_{dr})^T$, $u = \rho_{ref}$ and $y = \Delta y$.

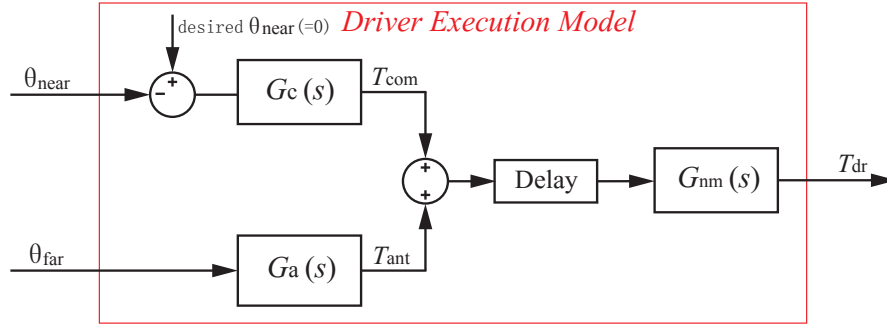


Figure 6.6: Two-point visual steering control driver model.

Next, we optimize the driver parameters in (6.36)-(6.37) by minimizing the H_2 norm of the closed loop system. For notational simplicity, let $p_1 = K_a$, $p_2 = K_c$, $p_3 = T_L$, $p_4 = T_I$, $p_5 = T_N$, $p_6 = t_p$. In addition to the parameters p_1, \dots, p_6 , the driver's near field look-ahead distance ℓ_s is also an important feature of the driver steering characteristics. We therefore consider ℓ_s as an additional tunable parameter and let $p_7 = \ell_s$. Considering the physical limits of the human driver, each model parameter is restricted to lie within some compact interval, $p_i \in [\underline{p}_i, \overline{p}_i]$, $i = 1, 2, \dots, 7$. Let $p = (p_1, p_2, \dots, p_7) \in \mathbb{R}^7$, and let $\mathcal{P} = [\underline{p}_1, \overline{p}_1] \times [\underline{p}_2, \overline{p}_2] \times \dots \times [\underline{p}_7, \overline{p}_7] \subset \mathbb{R}^7$. Note that the matrices A and B in (3.13)-(3.14) depend on p . In the following, we will use the notation $A(p)$ and $B(p)$ to emphasize this explicit dependence on p , when needed, for clarity. The partial derivative of a matrix-valued function $Z : \mathbb{R}^m \rightarrow \mathbb{R}^{n \times n}$ with respect to the vector $x \in \mathbb{R}^m$ takes the following form

$$\frac{\partial Z(x)}{\partial x} \triangleq \left[\frac{\partial Z(x)}{\partial x_1}, \frac{\partial Z(x)}{\partial x_2}, \dots, \frac{\partial Z(x)}{\partial x_{m-1}}, \frac{\partial Z(x)}{\partial x_m} \right] \in \mathbb{R}^{n \times mn},$$

where x_k denotes the k th element of the vector x , where

$$\frac{\partial Z(x)}{\partial x_k} = \begin{bmatrix} \frac{\partial Z_{11}(x)}{\partial x_k} & \frac{\partial Z_{12}(x)}{\partial x_k} & \cdots & \frac{\partial Z_{1n}(x)}{\partial x_k} \\ \frac{\partial Z_{21}(x)}{\partial x_k} & \frac{\partial Z_{22}(x)}{\partial x_k} & \cdots & \frac{\partial Z_{2n}(x)}{\partial x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Z_{n1}(x)}{\partial x_k} & \frac{\partial Z_{n2}(x)}{\partial x_k} & \cdots & \frac{\partial Z_{nn}(x)}{\partial x_k} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

and where $Z_{ij}(x)$ is the (i, j) entry of $Z(x)$, with $k = 1, 2, \dots, m$ and $i, j = 1, 2, \dots, n$.

First, note that system (6.36)-(6.37) is strictly proper. We need to keep the state matrix $A(p)$ in (6.36) Hurwitz by choosing appropriate driver parameter vector $p \in \mathcal{P}$. This condition imposes a set of nonlinear constraints on the parameter vector p . Furthermore, the H_2 -norm of the system from $u = \rho_{\text{ref}}$ to $y = \Delta y$ (when $T_{\text{con}} = 0$) can be computed from [179]

$$\|G(s, p)\|_2^2 = \text{tr}[CW_c(p)C^T], \quad (6.38)$$

where $G(s, p) = C(sI - A(p))^{-1}B(p)$. The matrix $W_c(p)$ satisfies the Lyapunov equation

$$W_c(p)A^T(p) + A(p)W_c(p) + B(p)B^T(p) = 0, \quad p \in \mathcal{P}. \quad (6.39)$$

Assuming the pair $(A(p), B(p))$ is controllable for all $p \in \mathcal{P}$, it follows (6.39) that $W_c(p) > 0$ for all $p \in \mathcal{P}$. The optimal parameters of the driver should minimize (6.38), that is, we want to solve the following problem

$$\min_{p \in \mathcal{P}} \text{tr}[CW_c(p)C^T], \quad (6.40a)$$

$$\text{s.t.} \quad W_c(p)A^T(p) + A(p)W_c(p) + B(p)B^T(p) = 0, \quad \forall p \in \mathcal{P}, \quad (6.40b)$$

$$W_c(p) > 0, \quad \forall p \in \mathcal{P}. \quad (6.40c)$$

Note that the parameter vector p enters nonlinearly in the expressions for $A(p)$ and $B(p)$. We use a gradient descent algorithm to find the minimum value of the constrained nonlinear problem (6.40). To this end, let $J(p) = \text{tr}[CW_c(p)C^T]$. The partial derivative of $J(p)$ with respect to the vector p is computed as follows

$$\frac{\partial J(p)}{\partial p} = \frac{\partial}{\partial p} \text{tr}[CW_c(p)C^T] = C \frac{\partial W_c(p)}{\partial p} (I_m \otimes C^T), \quad p \in \mathcal{P}, \quad (6.41)$$

where $W_c(p)$ satisfies the Lyapunov equation in (6.39), and m is the length of p . Note that the solution of (6.39) can be written as [179]

$$W_c(p) = -\text{vec}_1^{-1} \left([A(p) \oplus A(p)]^{-1} \text{vec}_1[B(p)B^T(p)] \right), \quad (6.42)$$

where $\text{vec}_1 : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{nm}$ is the vectorization operator. The matrix $A(p) \oplus A(p)$ in (6.42) is always invertible for all $p \in \mathcal{P}$ if the matrix $A(p)$ is Hurwitz. By taking the partial

derivative of (6.42) with respect to the parameter vector p , one obtains

$$\begin{aligned} \frac{\partial W_c(p)}{\partial p} = & \text{vec}_1^{-1} \left(\text{vec}_2 \left([A(p) \oplus A(p)]^{-1} \frac{\partial [A(p) \oplus A(p)]}{\partial p} \left(I_m \otimes [A(p) \oplus A(p)]^{-1} \right) \left(I_m \otimes \right. \right. \right. \\ & \left. \left. \left. \text{vec}_1[B(p)B^T(p)] \right) - [A(p) \oplus A(p)]^{-1} \text{vec}_2^{-1} \text{vec}_1 \left(\frac{\partial B(p)}{\partial p} \left(I_m \otimes B^T(p) \right) + B(p) \frac{\partial B^T(p)}{\partial p} \right) \right) \right), \end{aligned} \quad (6.43)$$

where $\text{vec}_2 : \mathbb{R}^{n^2 \times m} \rightarrow \mathbb{R}^{n^2 m}$ is the vectorization operator. The partial derivative of $A(p) \oplus A(p)$ with respect to p in (6.43) is given by

$$\frac{\partial [A(p) \oplus A(p)]}{\partial p} = \frac{\partial [A(p) \otimes I_n]}{\partial p} + \frac{\partial [I_n \otimes A(p)]}{\partial p} = \frac{\partial A(p)}{\partial p} \otimes I_n + \left(I_n \otimes \frac{\partial A(p)}{\partial p} \right) \Psi_{m,n}, \quad (6.44)$$

where $\Psi_{m,n}$ has the form

$$\Psi_{m,n} = \begin{bmatrix} I_m \otimes e_1 \otimes I_n \\ I_m \otimes e_2 \otimes I_n \\ \vdots \\ I_m \otimes e_n \otimes I_n \end{bmatrix},$$

where e_k denotes the n -dimensional unit row vector, having only the k th element equal to 1 and all other elements are 0. By substituting (6.43) into (6.41), one obtains the partial derivative of $J(p)$ with respect to the parameter vector p .

An indirect, alternative method to determine the gradient of $J(p)$ is given as follows. By taking the partial derivative of the Lyapunov equation in (6.42) with respect to p , one obtains

$$\begin{aligned} \frac{\partial W_c(p)}{\partial p} \left(I_m \otimes A^T(p) \right) + W_c(p) \frac{\partial A^T(p)}{\partial p} + \frac{\partial A(p)}{\partial p} \left(I_m \otimes W_c(p) \right) + A(p) \frac{\partial W_c(p)}{\partial p} \\ + \frac{\partial [B(p)B^T(p)]}{\partial p} = 0. \end{aligned} \quad (6.45)$$

One can then solve (6.45) for the partial derivative of $W_c(p)$ with respect to p as follows

$$\begin{aligned} \frac{\partial W_c(p)}{\partial p} = & -\text{vec}_1^{-1} \left(\left([I_m \otimes A(p)] \oplus A(p) \right)^{-1} \text{vec}_1 \left(\frac{\partial A(p)}{\partial p} \left(I_m \otimes W_c(p) \right) + W_c(p) \frac{\partial A^T(p)}{\partial p} \right. \right. \\ & \left. \left. + \frac{\partial B(p)}{\partial p} \left(I_m \otimes B^T(p) \right) + B(p) \frac{\partial B^T(p)}{\partial p} \right) \right), \quad p \in \mathcal{P}. \end{aligned} \quad (6.46)$$

The solutions for $\partial W_c(p)/\partial p$ given by (6.43) and (6.46) are the same (the proof to show this conclusion is not provided due to lack of space). Given $A(p)$ and $B(p)$, we can calculate $\partial A(p)/\partial p$, $\partial A^T(p)/\partial p$, $\partial B(p)/\partial p$ and $\partial B^T(p)/\partial p$ and further obtain $\partial W_c(p)/\partial p$ with (6.43) or (6.46). Then the partial derivative of $J(p)$ with respect to p follows directly

from (6.41)

$$\begin{aligned} \frac{\partial J(p)}{\partial p} = & -C \left(\text{vec}_1^{-1} \left(\left([I_m \otimes A(p)] \oplus A(p) \right)^{-1} \text{vec}_1 \left(\frac{\partial A(p)}{\partial p} [I_m \otimes W_c(p)] + W_c(p) \frac{\partial A^T(p)}{\partial p} \right. \right. \right. \\ & \left. \left. \left. + \frac{\partial B(p)}{\partial p} [I_m \otimes B^T(p)] + B(p) \frac{\partial B^T(p)}{\partial p} \right) \right) \right) (I_m \otimes C^T), \quad p \in \mathcal{P}, \end{aligned} \quad (6.47)$$

where $W_c(p)$ is given by (6.42).

6.4.2 Output Regulation

Instead of optimizing the driver parameters, we can develop a tracking controller for lane keeping/switching, such as an MPC controller or an output regulator [59]. This chapter designs tracking control using the output regulation theory (ORT). To this end, we use Fig. 6.7 to calculate the heading error $\Delta\psi$ and the lateral error Δy [60].

In Fig. 6.7, the red curve denotes the reference path we want to track, ℓ_s denotes the preview distance along the vehicle's heading direction, ψ_t denotes the angle between the tangent direction of the reference curve at current location (M) and the X_I axis. The lateral error Δy denotes the distance between the preview point A and the reference point B on the target path. The heading error is given by $\Delta\psi = \psi_t - \psi$, where ψ denotes the vehicle's yaw angle. The dynamics equations for Δy and $\Delta\psi$ can be approximatedly

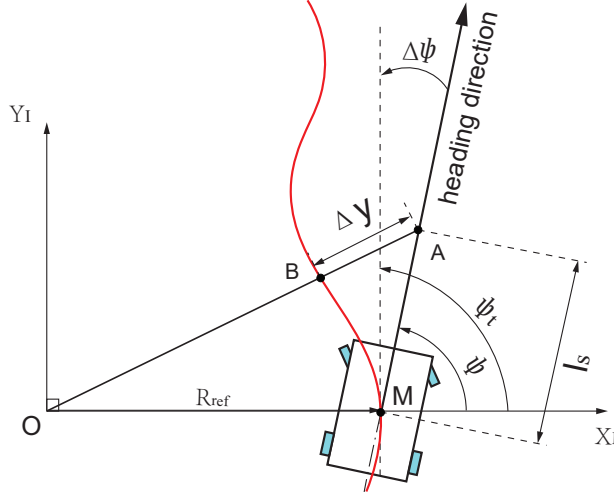


Figure 6.7: Path tracking error.

given by

$$\Delta \dot{y} = -V_x(\beta - \Delta\psi) - \ell_s r + V_x \ell_s \rho_{\text{ref}}, \quad (6.48a)$$

$$\Delta \dot{\psi} = V_x \rho_{\text{ref}} - r. \quad (6.48b)$$

where $\rho_{\text{ref}} = 1/R_{\text{ref}}$ denotes the reference road curvature. We then linearize the vehicle model in (4.1a)-(4.1c) and combine the vehicle model and the perception model in

(6.48a)-(6.48b). We treat the road curvature ρ_{ref} as the noise term and write the dynamics equation in the form of $\dot{x} = Ax + Bu + Ew$, where the state is $x = [\beta, r, \Delta y, \Delta \psi]^T$, the control is $u = \delta$ and the noise is $w = \rho_{\text{ref}}$.

We design the path tracking control following the approach of [180], with the aim of eliminating the tracking error at the near preview point, namely, $\lim_{t \rightarrow \infty} \Delta y(t) = 0$. Such a controller is referred to as the Output Regulation Theory (ORT) controller in that paper. The control input δ of the ORT controller is a linear combination of a feedback term and a feedforward term as follows,

$$\delta = Gx + H\rho_{\text{ref}}, \quad (6.49)$$

where the matrix G is chosen such that the matrix $A + BG$ is Hurwitz. Then H is determined by solving the following equations, for some matrices Γ and Π

$$H = \Gamma - G\Pi, \quad (6.50a)$$

$$A\Pi + B\Gamma + E = 0, \quad (6.50b)$$

$$C\Pi = 0. \quad (6.50c)$$

The unknown G and H can also be determined by solving a series of linear matrix inequalities (see [60]). We can also take the driver into consideration and design a driver-based ORT controller for the vehicle-driver-perception system in (6.36)-(6.37), following the equations in (6.49)-(6.50) similarly. More discussion can be found in [60].

6.5 Results and Analysis

In this section we first show the optimal driver parameters and implement the optimal driver model for lane keeping in simulations using CarSim [123]. We then demonstrate the overtaking policy learned in Chapter 5 using a traffic simulator along with implementing the path planning algorithms in Section 6.2 and the low level controllers designed in Section 6.3-6.4.

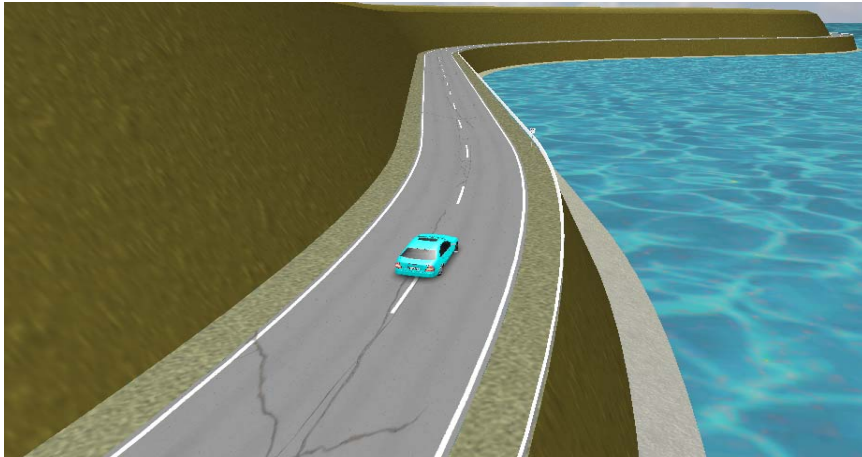


Figure 6.8: Road and vehicle used in Carsim.

6.5.1 Optimal driver parameters

The vehicle model is configured with Carsim 8.0 and is initialized with a constant speed of 15 [m/sec](54 [km/h]). Other constants can be found in Table 6.1. In addition, we assume a high-adhesion asphalt pavement with a constant friction factor of 0.89 in the simulations. The length and the width of the road are configured as 1000 [m] and 12 [m], respectively. Figure 6.8 gives a view of the vehicle and the road used in the numerical simulations. The road curvature is obtained through a sensor provided by CarSim.

Table 6.1: Constant parameters of the vehicle.

m	Mass of vehicle	1653	kg
V_x	Longitudinal velocity of vehicle	15	m/sec
ℓ_f	Distance from center of gravity to front axis	1.402	m
ℓ_r	Distance from center of gravity to rear axis	1.646	m
L_s	Distance from center of gravity to far field visual point	15	m
I_z	Moment of inertia of the vehicle	2765	kgm ²
J_s	Moment of inertia of the steering column	0.11	kgm ²
C_f	Front tires cornering power	42000	N/rad
C_r	Rear tires cornering power	81000	N/rad
b_s	Steering gear friction coefficient	0.57	Nm/rad/sec
g_s	Steering gear ratio	16	–
K_{aln}	Equivalent alignment torque coefficient	359.1	Nm/rad

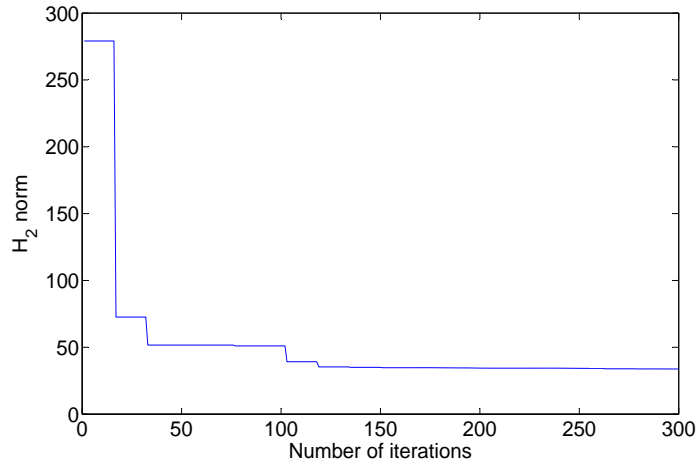


Figure 6.9: Trajectory of H_2 vs. the number of iterations.

We choose the parameters of Driver1 as the initial driver parameters, and perform the optimization following the steps outlined in equations (3.17)-(3.24). Table 6.2 provides the upper and lower bounds of each parameter and shows the optimal driver parameters (Driver4). The H_2 -norm initially is about 275.83, which is decreased to 33.78

as a result of the optimization process. Figure 6.9 illustrates the value of the H_2 -norm during the optimization process.

Table 6.2: Driver model parameters.

Parameter	Driver1	Driver2	Driver3	Driver4	Upper bound	Lower bound
K_a	22	30	45	70	70	10
K_c	14	20	27	50	50	5
T_L [sec]	1.6	2.4	3.5	3.03	5	0
T_I [sec]	0.35	0.2	0.1	0.02	0.5	0
T_N [sec]	0.12	0.12	0.12	0.01	0.2	0.01
t_p [sec]	0.1	0.06	0.04	0.01	0.3	0.01
ℓ_s [m]	5	5	5	3	15	3

To test the results, the other driver parameters (Driver2 and Driver3) were set as initial conditions as well, but the optimal parameters turned out to be quite close. By observing the optimal driver parameters we can note some interesting features. Except for the lead time constant T_L , all optimal parameters are close to their respective boundaries. The optimal static gain constants K_a and K_c are at their upper boundaries, while the time constants T_I , T_N and t_p are at their lower boundaries. These results indicate that the optimal driver spares no effort in the anticipatory/compensatory control paths, with negligibly small time delay.

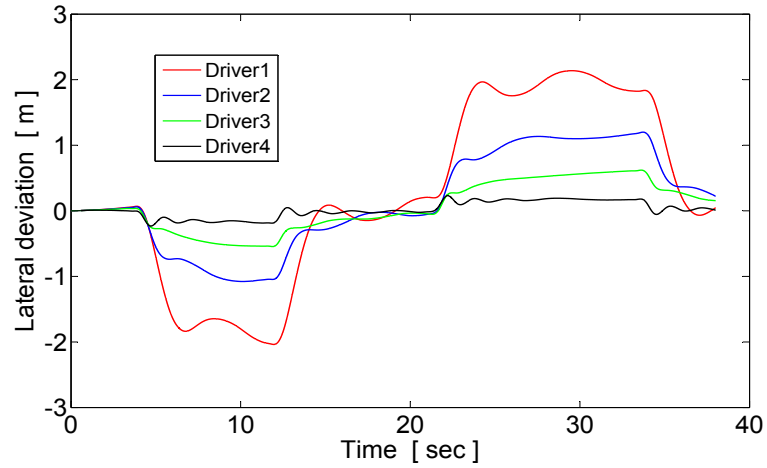


Figure 6.10: Comparison of tracking errors for all four drivers.

Figure 6.10 comparatively depicts the simulation results from all four drivers. In this figure, one sees that Driver3 behaves better than Driver1 and Driver2 because the parameters of Driver3 correspond to a higher driving skill; Also, as expected, Driver4 corresponding to the optimal parameters has the overall best performance. These results are validated using the H_2 -norm of the closed-loop system. Table 6.3 shows the H_2 -norms

of the systems with different drivers, and shows that the smaller H_2 -norm, the smaller the tracking error. Figure 6.10 and Table 6.3 agree well with each other.

Table 6.3: H_2 -norms regarding to various drivers.

	Driver1	Driver2	Driver3	Driver4
H_2	275.83	145.18	108.90	33.78

6.5.2 Path Planning

We implemented both Algorithms 8 and 9 to plan paths for lane switching. The maximum curvature of each curve is assigned with different values, namely, $\bar{\kappa}_{\max} = 0.05, 0.1, 0.15, 0.2$ and 0.25 . The width of the lane is $W = 4$ [m]. We plot the paths in Fig. 6.11.

Given the different requirements on the maximum curvature, the longitudinal lane-

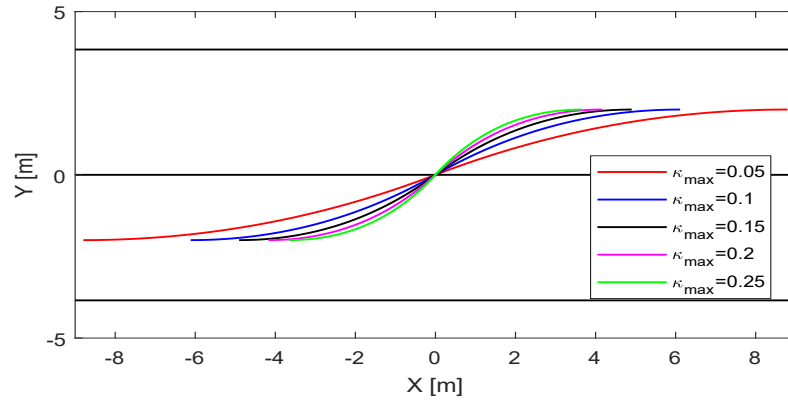


Figure 6.11: Quadratic Bézier curves for lane switching.

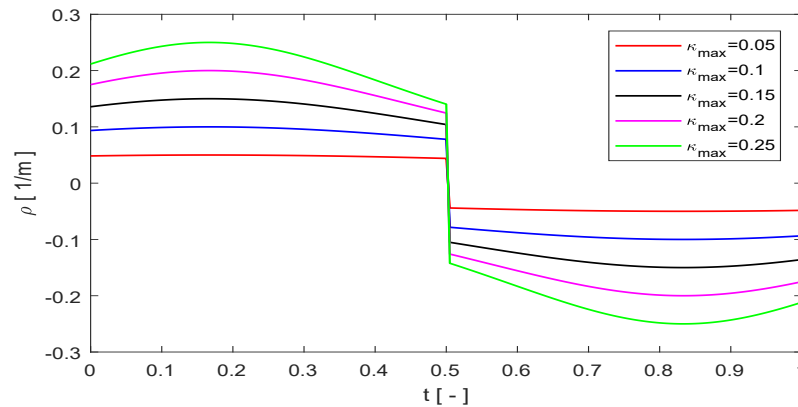


Figure 6.12: The curvature of the quadratic Bézier curves.

switching distance is also different. We then plot the curvature profile for each path, as shown in Fig. 6.12.

Fig. 6.12 shows that the maximum curvature for each path satisfies the design requirement. This result validates the effectiveness of Algorithm 8. Nevertheless, we notice that such designed Bézier curves are only C^1 continuous, since the curvature changes sign at the joint point of the two quadratic Bézier curves. In order to generate paths with better smoothness properties, we use Algorithm 9. Next, we implement Algorithm 9 to

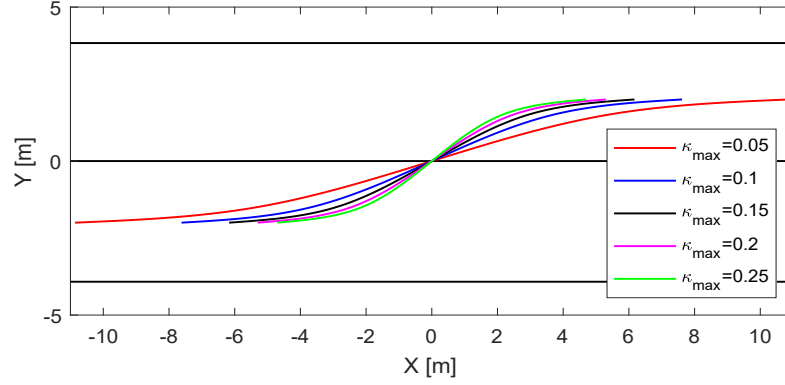


Figure 6.13: Fourth order Bézier curves for lane switching.

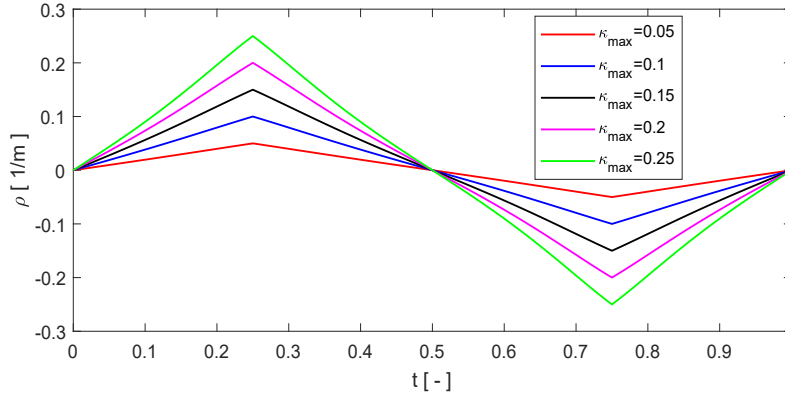


Figure 6.14: The curvature of the fourth order Bézier curves.

construct the paths using the fourth order Bézier curves. We keep the same setup for the road width and the maximum allowed curvatures. The fourth order Bézier curve paths are shown in Fig. 6.13. We also plot the curvature profile for each path in Fig. 6.14.

The results in Fig. 6.14 and Fig. 6.12 show that the maximum curvature for each path satisfies the design requirement. This result validates the effectiveness of both Algorithm 8 and 9. Nevertheless, we notice that the quadratic Bézier curves are only C^1 continuous, since the curvature changes sign at the joint point of the two Bézier curves. The fourth order Bézier curves are C^2 continuous and the curvature is continuous everywhere. Since the paths have zero curvature at both the two endpoints, Algorithm 9 provides much better smoothness than Algorithm 8.

6.5.3 Path Tracking Control

We then implemented the tracking controller in Section 6.4 to follow the Bézier curves. Since the optimal driver model has been validated using simulation in a lane-keeping task in Section 6.5.1, we only show the result by implementing the ORT controller for lane switching. The vehicle model parameters are summarized in Table 6.4.

Table 6.4: Vehicle model parameters.

$m[kg]$	850	total mass	$I_z[kgm^2]$	1401	rotational inertia
$I_{wr}[kgm^2]$	0.6	wheel rotational inertia	$\ell_f[m]$	1.5	distance to front axle
$h[m]$	0.5	height of mass center	$\ell_s[m]$	1	preview distance
$L[m]$	2.4	wheel base	$R[m]$	0.311	wheel radius
Tire force model parameters		$B = 3.9$	$C = 5.4$	$D = 0.7$	$E = S_h = S_v \approx 0$

We first generate several reference paths using both the joint quadratic Bézier curves and the fourth order Bézier curves, respectively. Next, the tracking controller in 6.4 is implemented to track the reference paths for lane switching. Since the fourth order Bézier curves have better smoothness, we show only the result for tracking the fourth order Bézier curves to conserve the space (see Figure 6.15).

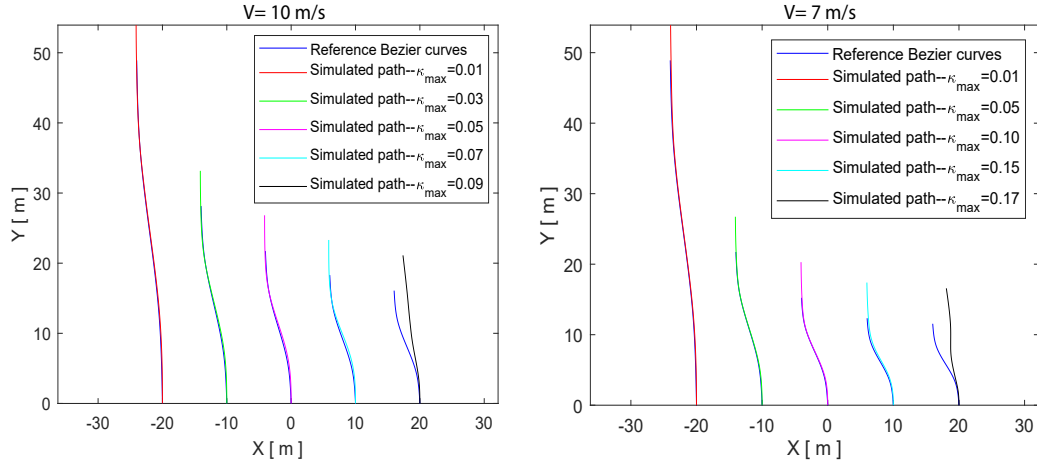


Figure 6.15: Tracking control for fourth order Bézier curves.

Fig. 6.15 (left) indicates that, given the vehicle velocity approximatedly fixed at 10 [m/s], the controller is not able to track the reference path when the maximum curvature is larger than 0.09 [1/m]. Tracking errors for small curvature paths are satisfactory. Next, we reduce the velocity of the vehicle to 7 [m/s] and implement the controller again. The results are plotted in Fig. 6.15 (right). One sees from Fig. 6.15 that, a larger maximum curvature may be allowed for tracking control if the velocity of the vehicle is lower.

We recall that the maximum friction coefficient between the tire and the ground is defined as $\mu_{\max} = D = 0.7$ in Table 6.4. One can evaluate the maximum curvature for path tracking using $\bar{\kappa}_{\max} = \mu_{\max}g/V^2$, which further indicates that $\bar{\kappa}_{\max} = 0.07$ [1/m] and 0.14 [1/m] for $V = 10$ [m/s] and 7 [m/s], respectively. This result agrees with 6.15.

The fourth order Bézier curves have better smoothness, and the controller designed in Section 6.4 is able to track the Bézier curves even if κ_{\max} is close to $\bar{\kappa}_{\max}$.

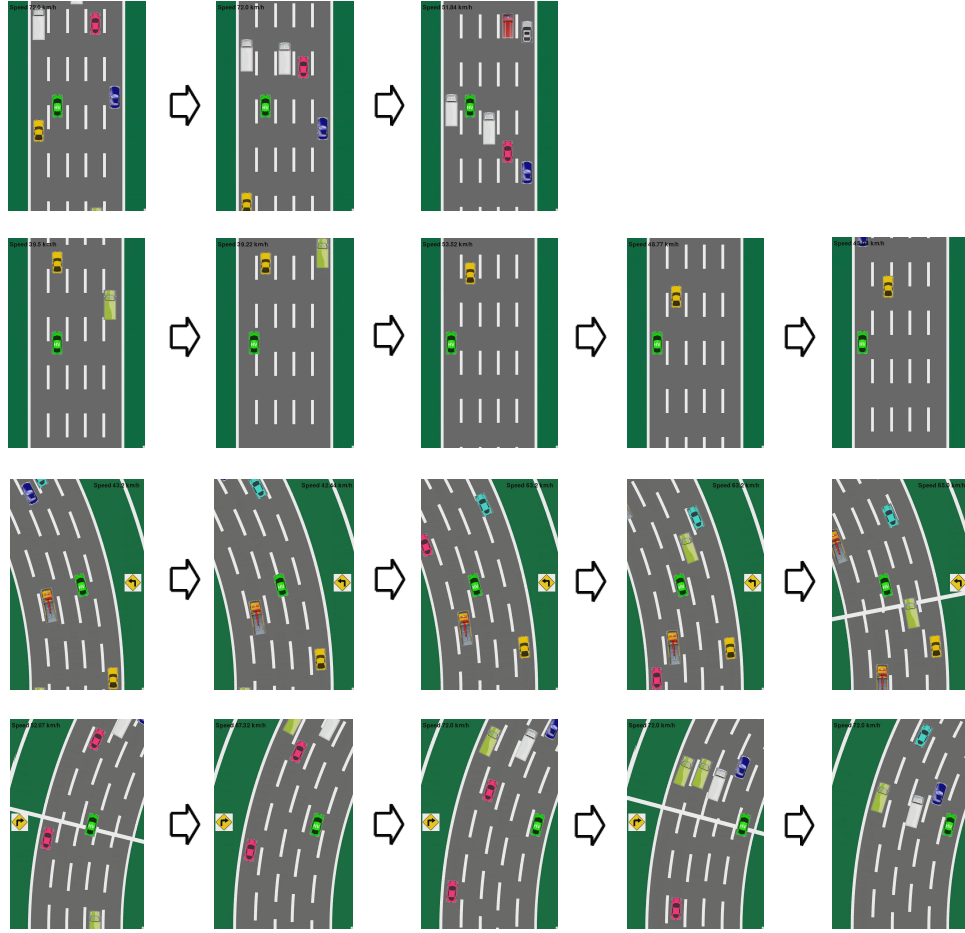


Figure 6.16: Overtaking scenarios in simulation by implementing π_1^* .

6.5.4 Overtaking Behavior

We implemented the optimal overtaking policies π_1^* from Chapter 5 and the controllers designed in Section 6.3-6.4 in simulation. Fig. 6.16 shows four different driving scenarios (each single row of pictures).

The first row of Fig. 6.16 shows a scenario where there is a vacant space in front of the HV (green). The HV accelerates to overtake the two trucks in the front on the neighbor lanes and then maintains certain high speed. The second row shows a scenario where there is one vehicle in front of the HV and both the left and right lanes are available for the HV to overtake the front vehicle. Since the road is straight, the HV is free to use either the left or the right lane to complete the overtaking task. One sees from this figure that the HV first switches to the left lane, and then accelerates to overtake the front yellow vehicle. The HV cannot successfully overtake the yellow vehicle since it detects another car (dark blue) in the front and has to brake to maintain a minimal distance. The third

scenario shows one vehicle (blue) in front of the HV but the left lane of the blue TV is occupied by another vehicle (dark blue). Nevertheless, the HV finds that the dark blue TV has a higher speed than the blue TV and hence it is possible to overtake the blue TV from the left lane. The HV then switch to the left lane. The dark blue TV disappears from the screen since it is running away. The HV then accelerates and maintains a high speed since there is no vehicle in its front. The last scenario shows another driving scenario during cornering, which has one vehicle (truck) in front of the HV and only the right lane of the HV are available to use for overtaking. The HV first switches to the right lane, which is closer to the inner curb of the corner, and then tries to overtake the truck by accelerating. However, the HV detects a new vehicle (dark blue) in the front and has to switch to its right lane again until overtaking is possible. These driving behaviors using π_1^* agree with the desired behaviors, which validates the design of the reward function and the approach of using “dynamic cells” to deal with complicated traffic information¹ (i.e., different vehicle velocities, sizes and signals).

6.6 Conclusion

We use a stochastic Markov decision process to model the traffic, and achieve desired driving behaviors using reinforcement learning. The “dynamic cell” approach is proposed to address the different vehicle velocities, vehicle sizes and driver intents in traffic. We also take the road geometry into consideration such that the driving policy may change depending on the road curvature.

By designing the driver’s reward function, we are able to show typical driving behaviors such as overtaking and tailgating, using the Q-learning algorithm to learn the corresponding optimal policies. We have demonstrated these policies using a road with five lanes and with each TV implementing a random policy on a self-developed simulator based on Pygame. In order to complete lane-switching, we separate the task into a path planning task and a tracking control task. We then formulate two different algorithms to generate the smooth paths using both the joint quadratic Bézier curves and the fourth-order Bézier curves subject to certain maximum curvature constraint. The joint quadratic Bézier curves use less space to create a path. Nevertheless, the path is only C^1 continuous and it is more difficult to track than the fourth-order Bézier curves, which are C^2 continuous and have better smoothness. We design the tracking control based on the output regulation theory. Simulation results validate the effectiveness of both the path planning algorithms and the design of the controller.

Future work will focus on improving the work to incorporate pedestrians, traffic signals and more road intersections.

¹The videos are available on the DCSL youtube channel: <https://www.youtube.com/watch?v=maUt8Cac2WU> and <https://www.youtube.com/watch?v=393MJA6Kp3L>.

CHAPTER 7

PATH PLANNING AND CONTROL: OFF-ROAD RALLY RACING

7.1 Introduction

High-speed cornering is a technique used especially in rally racing, during which the vehicle is driving at high sideslip angles while cornering to shave off excess speed. Advanced control system design for (semi-) autonomous vehicles requires understanding of such aggressive driving maneuvers in order to be able to take advantage of the full handling capacity of the vehicle, resulting to enhanced stability and safety vehicle characteristics.

Velenis in [181, 182, 183] modeled high-speed cornering (so called trail braking) and showed that high-speed cornering can be generated as the solution of a minimum-time cornering problem subject to appropriate boundary conditions. Tavernini [184] investigated minimum-time cornering strategies for a vehicle with different transmission layouts (front-wheel-drive, rear-wheel-drive and all-wheel-drive) using different road surfaces, and showed that the minimum-time driving strategy under low-friction conditions turned out to be an aggressive high-drift cornering maneuver. Hindiyeh [185] analyzed the stability of the vehicle under high sideslip drifting conditions and revealed the existence of unstable equilibria corresponding to a steady-state cornering maneuver. The unstable equilibria during steady-state cornering were also shown by Yi [186], who proposed a hybrid tire/road model and analyzed the effect of the longitudinal slip on the lateral stability.

The above papers indicate that high-speed cornering may be approximately modeled using a steady-state cornering process. Nevertheless, the evidence is not clear and limited work exists on trajectory planning and motion control to generate high-speed cornering maneuver in real time. To better understand high-speed cornering, this dissertation first generates a series of demonstrations of high-speed cornering trajectories by solving the minimum-time cornering problem subject to several different initial conditions [182].

Based on these demonstrations, next we learn a primitive trajectory using an iterative EM algorithm, by utilizing an unscented Kalman filter (UKF) along with a dynamic time warping (DTW) algorithm to align the time indexing of all the demonstrations. The result is a primitive trajectory, which can be used as the unique prototype to follow during high-speed cornering.

This primitive high-speed cornering maneuver indicates the existence of a segment of sustained steady-state cornering. This observation leads to a decomposition of high-speed cornering into three stages, namely, guiding, sliding and exiting. Subsequently, we design a hybrid-mode control strategy for different stages separately, using a combination of linear and nonlinear control techniques such as the linear quadratic regulator

theory (LQR) [187] and differential flatness [188, 189, 190, 191].

This chapter is organized as follows: Section 7.2 formulates the optimization problem and solves a series of demonstrations of high-speed cornering trajectories. Section 7.3 obtains a primitive trajectory from these demonstrations using trajectory learning. Section 7.4 introduces differential flatness, shows the flatness property of the vehicle model, and explains how differential flatness can be used for trajectory generation during the entry stage for this problem. Section 7.5 introduces high-speed cornering segmentation and plans the trajectory. Sections 7.6 and 7.7 design and implement the controller in simulations. Section 7.8 generates real-time high-speed cornering using auto-rally experimental platform. Finally, Section 7.9 summarizes the results of this study.

7.2 High-Speed Cornering Trajectories

As a demonstration of high-speed cornering, a high-speed high-sideslip cornering trajectory can be obtained by solving either a minimum-time or a maximum-exit velocity path optimization problem [181, 192].

The minimum-time path compromises between shortest distance to travel and highest average velocity through the corner. The vehicle decelerates until the point of minimum radius and accelerates afterwards. In contrast, the maximum-exit velocity path does not penalize time of travel, but instead maximizes the available time for accelerating and hence obtains a larger exit corner velocity. To this end, the maximum-velocity path compromises between the maximum lowest-velocity during cornering and the largest available time for accelerating after the vehicle reaches the lowest-velocity. Oversteering combined with intense braking may be required to get the vehicle ready for accelerating as soon as possible. A detail analysis can be found in [192].

By comparing the minimum-time and maximum-exit velocity solutions, one sees that the maximum-exit velocity path has a larger side-slip during cornering, but the average velocity is smaller and hence the time of travel is longer than the minimum-time path. This dissertation takes the minimum-time solution as the demonstration of high speed cornering, since it shows better the high-speed high sideslip cornering ability of the vehicle.

It is worth mentioning that one distinctive feature of a high-speed cornering trajectory generated by an expert rally driver is the “late apex”, in which the vehicle exits the corner close to the inner edge of the road [182]. This feature can be observed from the maximum-exit velocity path, as well as the minimum-time path subject to the fixed final positioning of the vehicle after the corner. The vehicle is likely to have to enter the corner from the outer edge of the road in order to observe a trajectory that exhibits the “late apex” feature[192].

In this section we formulate the optimization problem. We then obtain high-speed cornering trajectories by solving the minimum-time cornering problem using numerical optimization techniques.

7.2.1 Problem Formulation

For the sake of convenience, we use V , β and r to represent the state of the vehicle and rewrite the dynamics equations in (4.1) as follows

$$\dot{V} = \frac{1}{m} \left(f_{Fy} \sin(\beta - \delta) + f_{Fx} \cos(\beta - \delta) + f_{Ry} \sin \beta + f_{Rx} \cos \beta \right), \quad (7.1a)$$

$$\dot{\beta} = -r + \frac{1}{mV} \left(f_{Fy} \cos(\beta - \delta) - f_{Fx} \sin(\beta - \delta) + f_{Ry} \cos \beta - f_{Rx} \sin \beta \right), \quad (7.1b)$$

$$\dot{r} = \frac{1}{I_z} \left((f_{Fy} \cos \delta + f_{Fx} \sin \delta) \ell_f - f_{Ry} \ell_r \right), \quad (7.1c)$$

where the control is chosen as $u = [\delta, f_{Fx}, f_{Rx}]^T$. The lateral tire forces f_{Fy} and f_{Ry} are calculated by

$$f_{iy} = D_i \sin(C_i \operatorname{atan}(B_i \alpha_i)), \quad i = F, R, \quad (7.2)$$

where D_i, C_i and B_i are constants, and the tire sideslip angles are given by

$$\alpha_F = \delta - \operatorname{atan}\left(\frac{V \sin \beta + \ell_f r}{V \cos \beta}\right), \quad (7.3a)$$

$$\alpha_R = -\operatorname{atan}\left(\frac{V \sin \beta - \ell_r r}{V \cos \beta}\right). \quad (7.3b)$$

For a 90deg cornering the road geometry is plotted in Figure 7.1. In this figure S_1 and S_2 denote the lengths of the two straight road segments before and after the corner, and R_{ref} and O denote the radius and the center of the centerline of the corner, respectively. The vehicle enters from Point A and exits from Point D with certain initial and final velocities.

We momentarily assume that the general case of a high-speed cornering maneuver does not have to show the “late apex” feature (but later we still generate high-speed cornering with “late apex”), and hence we relax the fixed positioning condition of the vehicle after the corner. The minimum-time cornering optimal control problem is then formulated as follows

$$\min_u t_f, \quad (7.4)$$

subject to dynamics (4.1a) and (4.1c) and boundary conditions

$$X(t_0) = X_A, \quad Y(t_0) = Y_A, \quad \psi(t_0) = \frac{\pi}{2},$$

$$V(t_0) = V_0, \quad \beta(t_0) = 0, \quad r(t_0) = 0,$$

$$X(t_f) = X_D, \quad \psi(t_f) = \pi,$$

$$\text{and the constraints} \quad -\frac{d}{2} < \Delta s < \frac{d}{2} \quad \text{for } \forall t \in [t_0, t_f],$$

where t_0 and t_f are the initial and final time, respectively; (X_A, Y_A) are the coordinates of

the initial position A , X_D is the horizontal coordinate of final position D , and V_0 is the initial velocity. Δs is the deviation of the vehicle's mass center from the centerline of the road, which is calculated by

$$\Delta s = \begin{cases} X - R_{\text{ref}}, & \text{if } \text{atan2}(Y, X) < 0, \\ \sqrt{X^2 + Y^2} - R_{\text{ref}}, & \text{if } 0 \leq \text{atan2}(Y, X) \leq \frac{\pi}{2}, \\ Y - R_{\text{ref}}, & \text{if } \text{atan2}(Y, X) > \frac{\pi}{2}, \end{cases} \quad (7.5)$$

where $\text{atan2}(\cdot, \cdot)$ is a function that returns the four quadrant arctangent of the arguments.

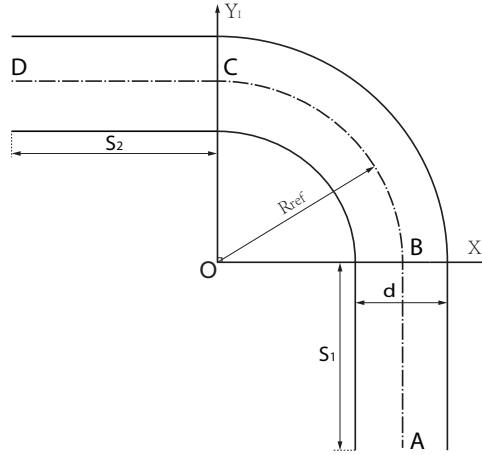


Figure 7.1: Road geometry.

7.2.2 Optimal Trajectories

In this section, we specify the geometry of the road and solve for the optimal trajectories of the vehicle subject to different initial conditions. For instance, by assigning the road geometry parameters $S_1 = S_2 = 5$ [m], $R_{\text{ref}} = 10$ [m], $d = 2$ [m] (see Figure 7.1), and the fifteen different initial positions and velocities (see Table 7.1), the minimization problem in (7.4) can be solved numerically. The vehicle model parameters we use to solve (7.4) are summarized in Table 4.1, which are from an actual fifth-scale auto-rally platform (see Section 7.8.2 for more descriptions).

Table 7.1: Initial conditions.

Initial Position A [(m,m)]	(9.5, -5), (10, -5), (10.5, -5)
Initial Velocity V_0 [m/s]	6, 7, 8, 9, 10

The optimal control solver GPOPS II [193] was used to obtain numerically the solution. The results, for all fifteen trajectories are shown in Figure 7.2(left).

Based on the results in Figure 7.2(left), one sees that the minimum-time cornering solutions tend to pass through the corner close to the inner boundary of the road. We

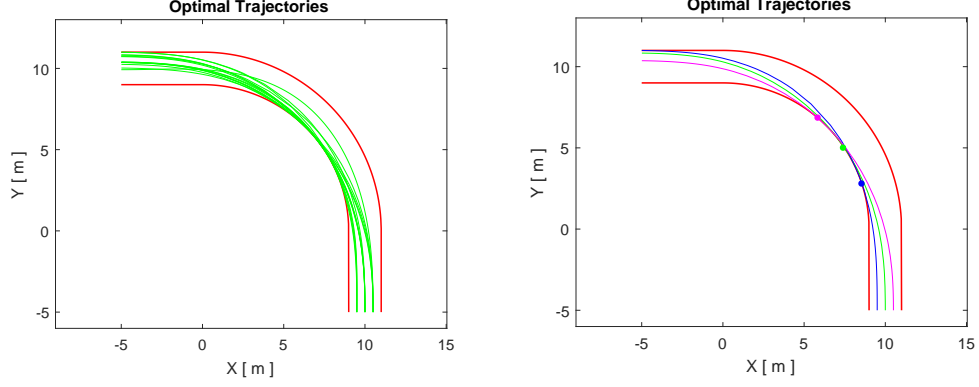


Figure 7.2: Optimal trajectories for different initial positions and velocities.

can also see how “late apex” is effected by different initial positions. Figure 7.2(right) keeps three of the trajectories and shows their tangential points to the inner edge of the road. All trajectories in Figure 7.2(right) are generated using the same initial velocity (6 [m/s]). “Late apex” is more obvious if the initial position is closer to the outer edge of the road. This result indicates that, in order to generate high-speed cornering with “late apex”, one may require to steer the vehicle to the outer boundary before entering the corner, which is typically how expert human rally drivers initiate a high-speed cornering maneuver [ref].

7.3 Trajectory Learning

In this section we present an algorithm to learn a primitive high-speed cornering trajectory from the demonstrations shown in Figure 7.2. The algorithm is based on the approach initially proposed in [194], which assumes that each demonstration is an independent, noisy observation of some (unknown) primitive trajectory, along with a possible time reparameterization.

7.3.1 Generative Model

To proceed with the analysis, we suppose that we are given M representative demonstrations of length N_k for $k = 0, 1, \dots, M - 1$. Each trajectory is assumed to be a discrete sequence of states x_j^k and controls u_j^k , which are composed into the augmented state:

$$y_j^k = \begin{bmatrix} x_j^k \\ u_j^k \end{bmatrix}, \quad j = 0, 1, \dots, N_k - 1, \quad k = 0, 1, \dots, M - 1. \quad (7.6)$$

We then define the “hidden” target trajectory z^* of length T , which is denoted by

$$z_t^* = \begin{bmatrix} x_t^* \\ u_t^* \end{bmatrix}, \quad t = 0, 1, \dots, T - 1. \quad (7.7)$$

The hidden trajectory in (7.7) must satisfy the system dynamics in (7.1)-(??). Assuming the control in the hidden trajectory does not change fast with time, the hidden trajectory satisfies the following equations,

$$\dot{x}^* = f(x^*, u^*), \quad (7.8a)$$

$$\dot{u}^* = \eta, \quad (7.8b)$$

$$\dot{\eta} = 0, \quad (7.8c)$$

After discretizing (7.8) and assuming some external noise, yields

$$z_{t+1}^* = f^*(z_t^*) + w_t^{(z)}, \quad (7.9a)$$

$$\eta_{t+1} = \eta_t + w_t^{(\eta)}, \quad (7.9b)$$

where $w_t^{(z)} \sim \mathcal{N}(0, \Sigma^{(z)})$ and $w_t^{(\eta)} \sim \mathcal{N}(0, \Sigma^{(\eta)})$ are Gaussian process noises. The value of $\Sigma^{(\eta)}$ determines the smoothness of u^* for the hidden trajectory. The function f^* is given by

$$f^*(z_t^*) = \begin{bmatrix} f(x_t^*, u_t^*) \, dt + x_t^* \\ \eta_t \, dt + u_t^* \end{bmatrix}, \quad t = 0, 1, \dots, T-1. \quad (7.10)$$

The demonstrations are independently observed from (7.9). The observations are therefore given by

$$y_j^k = z_{\tau_j^k}^* + w_j^{(y)}, \quad (7.11)$$

where $w_j^{(y)} \sim \mathcal{N}(0, \Sigma^{(y)})$ is Gaussian observation noise. Here τ_j^k is the time index in the hidden trajectory to which the observation y_j^k is mapped. The observation model is shown graphically in Figure 7.3(left). Since τ_j^k are not observed, we assume the following distribution with parameters d_i^k

$$\mathbb{P}(\tau_{j+1}^k | \tau_j^k) = \begin{cases} d_1^k & \text{if } \tau_{j+1}^k - \tau_j^k = 1, \\ d_2^k & \text{if } \tau_{j+1}^k - \tau_j^k = 2, \\ d_3^k & \text{if } \tau_{j+1}^k - \tau_j^k = 3, \\ 0 & \text{otherwise,} \end{cases} \quad (7.12)$$

with $\tau_0^k = 0$, where

$$\sum_{i=1}^3 d_i^k = 1, \quad d_i^k \geq 0. \quad (7.13)$$

For the sake of simplicity, in this work we apply inter/extrapolation to each demonstration with the given time index τ_j^k , such that the demonstrations have the same length

as the hidden trajectory. This idea is shown in Figure 7.3(right).

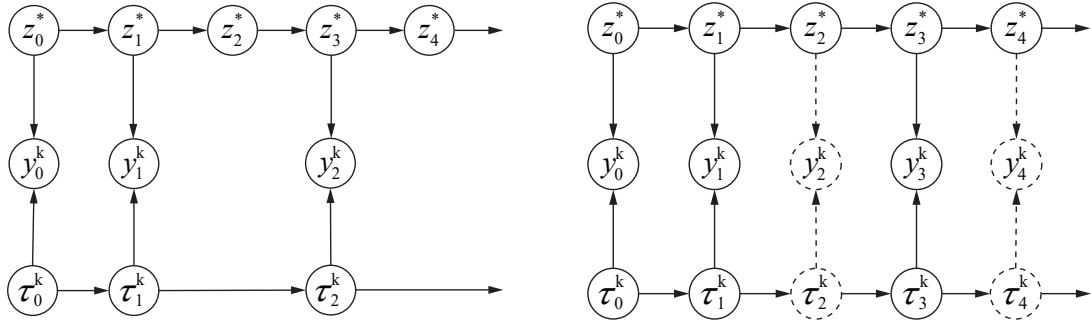


Figure 7.3: Graphical observation model with time indexing τ_j^k .

By comparing the two plots in Figure 7.3, one sees that the unobserved states z_2^* and z_4^* in Figure 7.3(left) are observed in Figure 7.3(right). The new observation in Figure 7.3(right) is considered to be reasonably accurate if the sampling interval is small and the demonstrations y_j^k in (7.6) are smooth. Since the demonstrations now have the same length as the hidden trajectory, we can rewrite Equation (7.11) as follows

$$\begin{bmatrix} y_j^0 \\ y_j^1 \\ y_j^2 \\ \vdots \\ y_j^{M-1} \end{bmatrix} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} z_j^* + \begin{bmatrix} {}^0w_j^{(y)} \\ {}^1w_j^{(y)} \\ \vdots \\ {}^{M-1}w_j^{(y)} \end{bmatrix}, \quad (7.14)$$

where the M Gaussian observation noises ${}^0w_j^{(y)}, \dots, {}^{M-1}w_j^{(y)}$ are assumed to be independent and identically distributed. The most likely hidden trajectory is then obtained by solving the following maximization problem

$$\max_{\tau, d, \Sigma^{(\cdot)}} \log \mathbb{P}(y; \tau, d, \Sigma^{(\cdot)}), \quad (7.15)$$

where \mathbb{P} is the joint likelihood of the observed trajectories y for the learned parameters $\tau, d, \Sigma^{(\cdot)}$.

To optimize the function in (7.15) we alternatively optimize over τ, d and $\Sigma^{(\cdot)}$. The procedure of optimization is described as follows. Given the initial values of τ, d and $\Sigma^{(\cdot)}$, one can implement an UKF to obtain the estimates for the distribution of the “hidden” target trajectory z^* . We denote the result as $\bar{z} \sim \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$. For the current estimate \bar{z} , the measurements y , and the initial values of τ and d , one updates the covariances $\Sigma^{(\cdot)}$ in the M-step using the standard EM algorithm. Next, for the updated covariances $\Sigma^{(\cdot)}$, we optimize the time indexing τ that maximizes the joint probability $\mathbb{P}(\bar{z}, y, \tau)$ using a dynamic time warping program. We optimize d in the last step. The algorithm is summarized in Algorithm 10.

Algorithm 10 Trajectory Learning Algorithm

Input: y

Output: $z^*, \tau^*, d^*, * \Sigma^{(\cdot)}$

- 1: $\Sigma^{(\cdot)} \leftarrow I, d_j^k \leftarrow \frac{1}{3}, \tau_j^k \leftarrow j \frac{T-1}{N^k-1}$
 - 2: Converge \leftarrow False
 - 3: **while not** Converge **do**
 - 4: E-step: run UKF to find the distributions $z_t \sim \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$ using observations y
 - 5: M-step: update the covariances $\Sigma^{(\cdot)}$ using the standard EM update
 - 6: E-step: run dynamic time warping to find τ that maximizes $\mathbb{P}(\bar{z}, y, \tau)$, where $\bar{z} = \mu_{t|t-1}$ (or $\mu_{t|T-1}$)
 - 7: M-step: estimate d from τ directly
 - 8: **if** z converges **then**
 - 9: Converge \leftarrow True
 - 10: $z^* \leftarrow z, \tau^* \leftarrow \tau, d^* \leftarrow d, * \Sigma^{(\cdot)} \leftarrow \Sigma^{(\cdot)}$
-

In the trajectory learning algorithm, Step 4) and Step 5) are standard. We show more detail about Step 6), in which we use a dynamic time warping algorithm to find τ to maximize $\mathbb{P}(\bar{z}, y, \tau)$, where \bar{z} is the mode of the distribution of the latent state computed using UKF. Mathematically, we want to solve

$$\begin{aligned}
 \bar{\tau} &= \arg \max_{\tau} \log \mathbb{P}(\bar{z}, y, \tau) \\
 &= \arg \max_{\tau} \log \mathbb{P}(y|\bar{z}, \tau) \mathbb{P}(\bar{z}) \mathbb{P}(\tau) \\
 &= \arg \max_{\tau} \log \mathbb{P}(y|\bar{z}, \tau) \mathbb{P}(\tau),
 \end{aligned} \tag{7.16}$$

where \bar{z} is fixed to $\mu_{t|t-1}$. Using $\ell(\cdot)$ to denote the log-likelihood, then $\bar{\tau}$ in (7.16) is computed by

$$\bar{\tau} = \arg \max_{\tau} \sum_{k=0}^{M-1} \left(\sum_{j=0}^{N_k-1} \ell(y_j^k | \bar{z}_{\tau_j^k}, \tau_j^k) + \sum_{j=1}^{N_k-1} \ell(\tau_j^k | \tau_{j-1}^k) \right), \tag{7.17}$$

where $\ell(y_j^k | \bar{z}_{\tau_j^k}, \tau_j^k)$ is given by

$$\ell(y_j^k | \bar{z}_{\tau_j^k}, \tau_j^k) = \ell(y_j^k | \mu_{\tau_j^k}, \tau_j^k) = (2\pi)^{-\frac{L}{2}} \|\Sigma_{\tau_j^k}^{-1}\|^{\frac{1}{2}} e^{-\frac{1}{2}(y_j^k - \mu_{\tau_j^k})^T \Sigma_{\tau_j^k}^{-1} (y_j^k - \mu_{\tau_j^k})}, \tag{7.18}$$

where L is the length of the vector $\bar{z}_{\tau_j^k}$, $\mu_{\tau_j^k} = \mu_{\tau_j^k | \tau_{j-1}^k}$ and $\Sigma_{\tau_j^k} = \Sigma_{\tau_j^k | \tau_{j-1}^k}$ are calculated in Step 2) of the Trajectory Learning Algorithm. Since the M observations are independent, the likelihoods for the M observations in (7.17) can be maximized separately. The maximization problem in (7.17) can be solved using dynamic programming.

7.3.2 Primitive High-Speed Cornering Trajectory

After implementing the previous Trajectory Learning Algorithm we obtain the result shown in Figure 7.4.

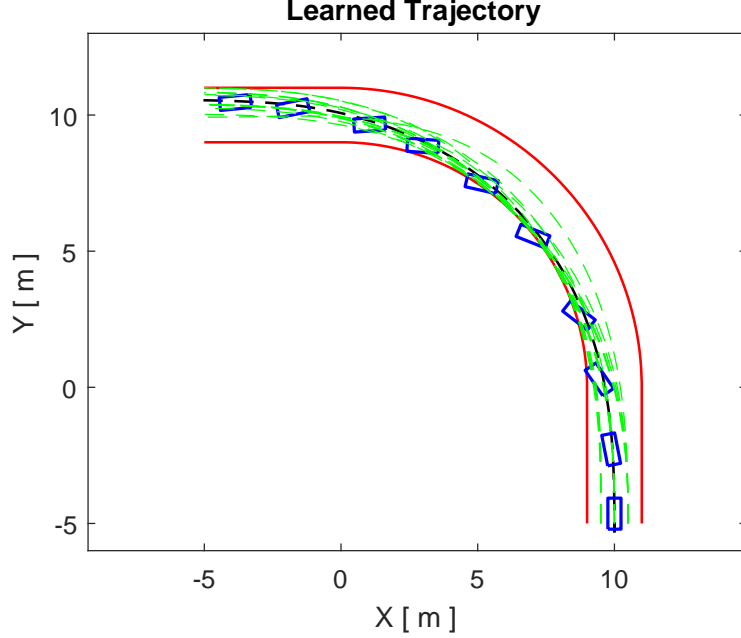


Figure 7.4: Multiple demonstrations and the learned primitive trajectory.

The primitive trajectory in Figure 7.4 is likely to share a segment of a circle that is tangent to the inner road boundary, and the vehicle seems to keep a constant side-slip angle β during the cornering. This observation leads to the conjecture that there is a segment of steady-state cornering in the primitive trajectory.

Steady-state cornering is characterized by a trajectory of constant radius R , negotiated at a constant speed V , constant yaw rate, and constant side-slip angle [84], as follows

$$R = R^{ss} = \text{const.}, \quad V = V^{ss} = \text{const.}, \quad \beta = \beta^{ss} = \text{const.}, \quad r = r^{ss} = \frac{V^{ss}}{R^{ss}} = \text{const.}, \quad (7.19)$$

where the steady-state triplet $(V^{ss}, \beta^{ss}, r^{ss})$ are obtained using (7.1) by letting $\dot{V} = \dot{\beta} = \dot{r} = 0$ as follows

$$0 = \frac{1}{m} \left(f_{Fy} \sin(\beta^{ss} - \delta^{ss}) + f_{Fx}^{ss} \cos(\beta^{ss} - \delta^{ss}) + f_{Ry} \sin \beta^{ss} + f_{Rx}^{ss} \cos \beta^{ss} \right), \quad (7.20a)$$

$$0 = -r^{ss} + \frac{1}{mV^{ss}} \left(f_{Fy} \cos(\beta^{ss} - \delta^{ss}) - f_{Fx}^{ss} \sin(\beta^{ss} - \delta^{ss}) + f_{Ry} \cos \beta^{ss} - f_{Rx}^{ss} \sin \beta^{ss} \right), \quad (7.20b)$$

$$0 = \frac{1}{I_z} \left((f_{Fy} \cos \delta^{ss} + f_{Fx}^{ss} \sin \delta^{ss}) \ell_f - f_{Ry} \ell_r \right), \quad (7.20c)$$

where the control components δ^{ss} , f_{Fx}^{ss} and f_{Rx}^{ss} are the steering angle and longitudinal

nal tire forces during steady-state cornering. By denoting the constant vector $u^{ss} = [\delta^{ss}, f_{Fx}^{ss}, f_{Rx}^{ss}]^T$, the equilibrium $(V^{ss}, \beta^{ss}, r^{ss})$ with the corresponding control u^{ss} are calculated following the equations in Section 3 of [84].

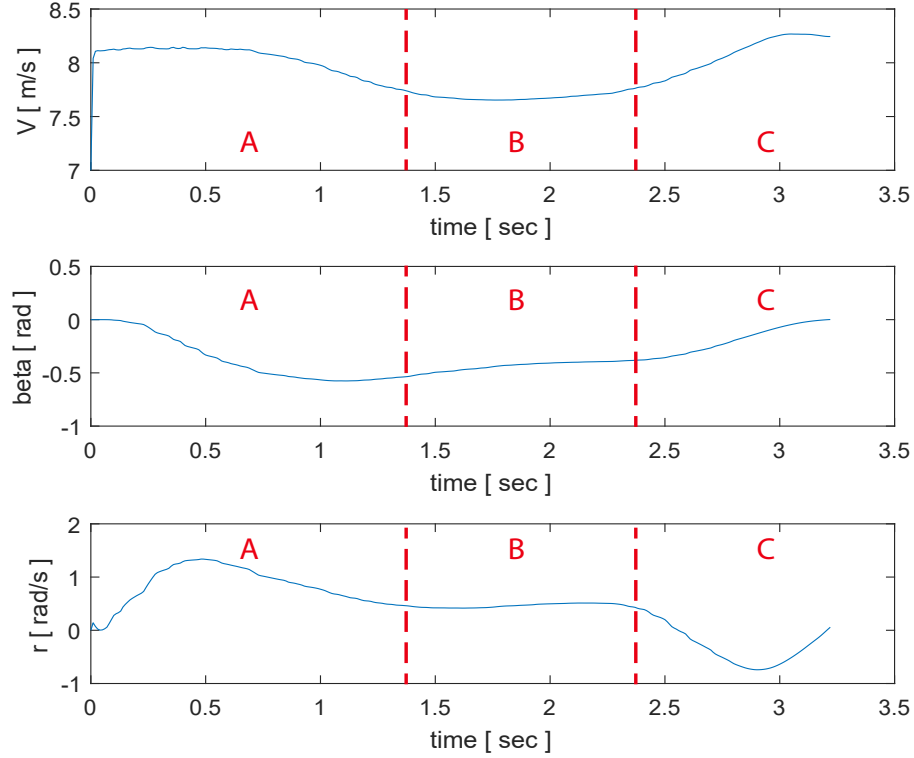


Figure 7.5: The velocity, side-slip and yaw motion of the learned primitive.

In order to see the existence of steady-state cornering in the primitive trajectory, we plot the velocity, side-slip angle and yaw rate of the primitive maneuver in Figure 7.5. From Figure 7.5 one observes that V , β and r are quite close to constants from 1.4 second to 2.4 second, which is typical of steady-state cornering. One also notices that, steady-state cornering begins after the vehicle has entered the corner for a while and terminates before the vehicle has left the corner. The large values of V and β indicate that the primitive maneuver indeed performs a high-speed, high-sideslip cornering maneuver.

Based on the result in Figure 7.5, we may divide a high-speed cornering trajectory into three segments. The first segment allows the vehicle to complete the transition from straight-line driving to steady-state cornering. Before the vehicle enters the corner, the vehicle gradually changes its velocity using certain steering and accelerating/braking operations, until it reaches the target steady state near the entry of the corner. The vehicle then maintains steady-state cornering during the second segment, until it gets (close) to the exit of the corner. Finally in the last segment, the vehicle leaves steady-state cornering and is steered back to straight-line driving after the vehicle exits the corner.

One may generate a high-speed, high-sideslip high-speed cornering maneuver using this idea. To this end, one first needs to calculate the equilibrium for steady-state cornering. The main problem occurs after one obtains the target steady state that, one

has to guide the vehicle from certain initial condition into the target steady state when the vehicle enters the corner. This task is challenging, since we need to find a feasible trajectory for the vehicle to follow such that the vehicle reaches the target steady-state exactly when the vehicle enters the corner. In order to solve this problem, in this dissertation we design the vehicle's trajectory and the corresponding tracking control using differential flatness.

7.4 Differentially Flatness Trajectory Generation

In this section we briefly introduce the concept of differential flatness and show that vehicle model (7.1) is differentially flat. This property of the vehicle will allow us to plan the vehicle's trajectory and build the corresponding control for the vehicle by using the desired output of the system, which is, indeed, the main theoretical base we use to generate the target high-speed cornering maneuvers in the following sections of this chapter.

7.4.1 Differential Flatness

Differential flatness theory and flatness-based control were introduced in the late 1980s by Michel Fliess and his colleagues. It provides an efficient solution to several nonlinear control and state estimation problems [188, 189]. A differentially flat system can be linearized and controlled through linear control methods [189, 195].

A nonlinear system $\dot{x} = f(x, u)$ is differentially flat if there exists an output variable $y \in \mathbb{R}^m$ in the following form

$$y = h(x, u, \dot{u}, \ddot{u}, \dots, u^{(r)}), \quad (7.21)$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$, such that

$$x = \Psi(y, \dot{y}, \ddot{y}, \dots, y^{(r)}), \quad (7.22a)$$

$$u = \Phi(y, \dot{y}, \ddot{y}, \dots, y^{(r)}), \quad (7.22b)$$

where f is a smooth vector field and h, Ψ and Φ are smooth functions. Equations (7.22) imply that the state x and the control u can be recovered using the m algebraic output variables $y_i, i = 1, 2, \dots, m$. The output y given in (7.21) is the flat output of the system.

The authors in [195, 196, 197, 198] designed a tracking control based on the differential flatness property of the single-track vehicle model, but only [198] used this for planning trajectories. Although [198] generated the reference trajectories for different cases from geometric path constraints, the assumptions were strong and the tire was required to work within its linear region. This dissertation plans the motion of a vehicle that involves a desired steady-state cornering, and analyzes the conditions on the trajectory in order to be recovered using the differential flatness theory, taking into account of the road condition and the steering capacity of the vehicle.

7.4.2 Differential Flatness of Vehicle Model

In this section we show that the equations (7.1) of the vehicle model are differentially flat with respect to a particular output. This property is stated in Theorem 7.4.1.

Theorem 7.4.1 *The vehicle model in (7.1) is differentially flat with respect to the following output [196]*

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} V \cos \beta \\ V \sin \beta - \left(I_z / m \ell_f \right) r \end{bmatrix}. \quad (7.23)$$

Proof: To show Theorem 7.4.1, we just need to find the expressions of the state x and control u in the form of (7.22). To this end, we recall from (7.2) that the lateral tire forces f_{iy} ($i = F, R$) are smooth functions of the sideslip angles α_i at the front and rear wheels, respectively. We can then derive the state of the system in (7.1) in terms of the flat output in (7.23) as follows

$$V = \sqrt{y_1^2 + \left(y_2 + \frac{I_z r}{m \ell_f} \right)^2}, \quad (7.24)$$

$$\beta = \text{atan} \left(\frac{m \ell_f y_2 + I_z r}{m \ell_f y_1} \right), \quad (7.25)$$

where $r = r(y_1, y_2, \dot{y}_2)$ is given by solving the following implicit equation,

$$D_R \sin \left(C_R \text{atan} \left(B_R \text{atan} \left(\frac{(m \ell_f \ell_r - I_z) r - m \ell_f y_2}{m \ell_f y_1} \right) \right) \right) - \frac{m \ell_f (\dot{y}_2 + y_1 r)}{\ell_f + \ell_r} \triangleq g(y_1, y_2, \dot{y}_2, r) = 0. \quad (7.26)$$

Equations (7.24)-(7.26) give the expressions of the state x in terms of the flat output in (7.23). The control $u = [\delta, f_{Fx}, f_{Rx}]^T$ can be recovered from (7.23) by solving the following equations

$$f_{Fy}(\delta, y_1, y_2, \dot{y}_2) \cos \delta \ell_f + f_{Fx} \sin \delta \ell_f - I_z \dot{r}(y_1, \dot{y}_1, y_2, \dot{y}_2, \ddot{y}_2) - f_{Ry}(y_1, y_2, \dot{y}_2) \ell_r = 0, \quad (7.27a)$$

$$m \left(\dot{y}_1 - y_2 r(y_1, y_2, \dot{y}_2) - \frac{I_z}{m \ell_f} r^2(y_1, y_2, \dot{y}_2) \right) - f_{Rx} + f_{Fy}(\delta, y_1, y_2, \dot{y}_2) \sin \delta - f_{Fx} \cos \delta = 0, \quad (7.27b)$$

$$\Gamma(f_{Fx}, f_{Rx}) = 0, \quad (7.27c)$$

where the rear wheel lateral tire force is given by

$$f_{Ry}(y_1, y_2, \dot{y}_2) = \frac{\dot{y}_2 m \ell_f + y_1 r(y_1, y_2, \dot{y}_2) m \ell_f}{\ell_f + \ell_r}, \quad (7.28)$$

and where $\Gamma(f_{Fx}, f_{Rx})$ in (7.27c) is a force distribution function related to the specific drive type of the vehicle. Proper choices of $\Gamma(f_{Fx}, f_{Rx})$ for different drive types are given

as follows:

a) All-Wheel-Drive (AWD): All four wheels provide accelerating and braking forces,

$$\Gamma(f_{Fx}, f_{Rx}) = \kappa f_{Fx} - (1 - \kappa) f_{Rx}, \quad (7.29)$$

where $\kappa \in [0, 1]$ is a constant that need to be specified.

b) Front-Wheel-Drive (FWD): All four wheels provide braking force but only the front wheels provide accelerating force,

$$\Gamma(f_{Fx}, f_{Rx}) = \begin{cases} f_{Rx}, & \text{if } f_{Fx} \geq 0, \\ \kappa f_{Fx} - (1 - \kappa) f_{Rx}, & \text{if } f_{Fx} < 0. \end{cases} \quad (7.30)$$

c) Rear-Wheel-Drive (RWD): All four wheels provide braking force but only the rear wheels provide accelerating force,

$$\Gamma(f_{Fx}, f_{Rx}) = \begin{cases} f_{Fx}, & \text{if } f_{Rx} \geq 0, \\ \kappa f_{Fx} - (1 - \kappa) f_{Rx}, & \text{if } f_{Rx} < 0. \end{cases} \quad (7.31)$$

Specifically, the auto-rally vehicle model introduced in Section 7.2 uses only the rear wheels for accelerating and braking and hence $f_{Fx} = 0$. We can then simplify (7.27) and determine the remaining control variables δ and f_{Rx} by solving the following equations

$$f_{Fy}(\delta, y_1, y_2, \dot{y}_2) \cos \delta \ell_f - I_z \dot{r}(y_1, \dot{y}_1, y_2, \dot{y}_2, \ddot{y}_2) - f_{Ry}(y_1, y_2, \dot{y}_2) \ell_r = 0, \quad (7.32a)$$

$$m \left(\dot{y}_1 - y_2 r(y_1, y_2, \dot{y}_2) - \frac{I_z}{m \ell_f} r^2(y_1, y_2, \dot{y}_2) \right) - f_{Rx} + f_{Fy}(\delta, y_1, y_2, \dot{y}_2) \sin \delta = 0. \quad (7.32b)$$

□

Equations(7.23)-(7.32) show the differential flatness of the vehicle model in (7.1). In the following sections, we design a tracking controller based on this differential flatness property of the single-track vehicle model in order to generate the high-speed cornering maneuver.

7.5 High-Speed Cornering Trajectory Planning

As mentioned earlier, based on the trajectory learning result in Section 7.3, we assume that a high-speed cornering maneuver consists of three stages: 1) An entry stage before the vehicle enters the corner; 2) a sliding stage where the vehicle passes through the corner at a steady-state; and 3) an exiting stage after the vehicle leaves the corner.

We consider two different scenarios of high-speed, high-slip cornering maneuvers, as shown in Figure 7.6. In the first scenario, we assume that the vehicle is required to pass through the corner next to the road centerline. Figure 7.6(left) shows the entire cornering process for this scenario. S_1 and S_2 denote the lengths of the straight road segments, R_{ref} and O denote the radius and the center of the centerline of the corner, R^{ss} and O^{ss} denote the radius and the center of the arc BPC , and D denotes the road width.

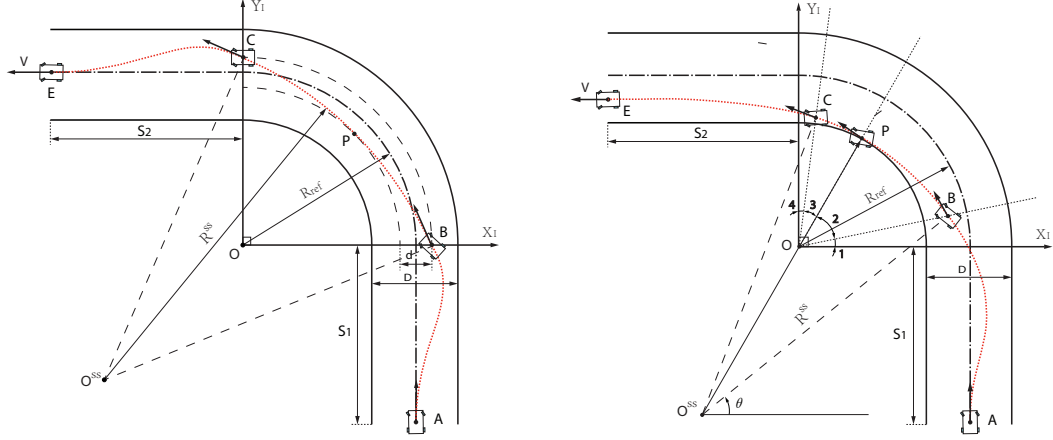


Figure 7.6: Road geometry and high-speed cornering trajectory.

d denotes the distance between the inner and outer constraint lines on the vehicle's trajectory in the corner, such that the vehicle keeps close to the road centerline. The points B and C are located on the outer constraint line and indicate the entering and exiting position of the corner, and P is a tangent point on the inner constraint curve. The vehicle starts at position A and enters the corner from position B . The trajectory of the vehicle is divided into three parts: the arc trajectory connecting BC for the steady-state cornering; and the two pieces of the trajectories connecting AB and CE where the vehicle enters/leaves the corner.

The second scenario shows the feature of “late apex”, where the vehicle exits the corner next to the inner edge of the road. This scenario does not require the vehicle to be close to the road centerline. Figure 7.6(right) shows this maneuver, which postpones the tangent point P to the second half of the corner and removes the inner/outer constraint lines. Similarly with the first scenario, the steady-state cornering begins from B , passes through P and ends up at C . We introduce $\angle 1, \dots, \angle 4$ to describe the position and length of the arc BPC . The nonnegative angles $\angle 1$ and $\angle 4$ denote the vehicle's late entering and early exiting positions of the steady-state cornering.

7.5.1 Sliding Trajectory

With the given road geometry shown in Figure 7.6, one can first determine the radius R^{ss} of the arc BPC for steady state cornering. By assigning appropriate values of $\angle 1, \dots, \angle 4$, one can change the position of the tangent point P and the length of the arc BPC , and hence obtain a trajectory with the desired “late apex” feature.

We denote the steady-state of the vehicle using the triplet $x^{ss} = [V^{ss}, \beta^{ss}, r^{ss}]^T$. As mentioned before, we can determine the equilibrium x^{ss} with the desired cornering speed V^{ss} , or the desired sideslip angle β^{ss} , following (7.20) and the equations in Section 3 of [84].

In this dissertation we calculate the equilibrium of the steady-state cornering and design the sliding mode trajectory offline.

7.5.2 Guiding Trajectory

The steady-state triplet $x^{ss} = [V^{ss}, \beta^{ss}, r^{ss}]^T$ obtained in Section 7.5.1 defines the state of the vehicle at position B . We assume that the initial state of the vehicle is known at A and we use the notations in Table 7.2 to represent the boundary conditions.

Table 7.2: Boundary conditions

	Speed	Slip angle	Yaw rate	X position	Y position	Yaw angle
Position A	V_A	β_A	r_A	X_A	Y_A	ψ_A
Position B	V^{ss}	β^{ss}	r^{ss}	X_B	Y_B	ψ_B

In order to determine a feasible trajectory of the vehicle from A to B , we claim that the trajectory must be designed to satisfy equation (7.26). To clarify the idea, let us assume that one has designed a trajectory $x(t) = [V(t), \beta(t), r(t)]^T$, $t \in [t_0, t_f]$, and has calculated the flat outputs $y_1(t)$ and $y_2(t)$ using the trajectory $x(t)$ following (7.23). However, it is not possible to recover $x(t) = [V(t), \beta(t), r(t)]^T$ using the flat outputs $y_1(t)$ and $y_2(t)$, unless $x(t)$ is designed to satisfy equation (7.26). We summarize this condition in Proposition 7.5.1.

Proposition 7.5.1 *A trajectory defined by $x(t) = [V(t), \beta(t), r(t)]^T$, $t \in [t_0, t_f]$, can be recovered from the output in (7.23) if and only if $y_1(t)$, $y_2(t)$ and $r(t)$ satisfy equation (7.26).*

Proof: a) To show necessity, We need to show that for given $x(t) = [V(t), \beta(t), r(t)]^T$ the flat outputs $y_1(t)$ and $y_2(t)$ satisfy the constraint equation (7.26). Recall from (7.23) that $y_2 = V \sin \beta - (I_z / m \ell_f) r$. Then the Lie derivative of y_2 along the vector field in (7.1) is given by

$$\dot{y}_2 = \mathcal{L}_f y_2(x, u) = \frac{\ell_f + \ell_r}{m \ell_f} f_{Ry} - V r \cos \beta, \quad (7.33)$$

where, following equations (7.2) and (7.3b), f_{Ry} is given by

$$f_{Ry} = -D_R \sin \left(C_{Ratan} \left(B_{Ratan} \left(\frac{V \sin \beta - \ell_r r}{V \cos \beta} \right) \right) \right). \quad (7.34)$$

Substituting (7.34) into (7.33), and replacing $V \cos \beta$ and $V \sin \beta$ with y_1 and $y_2 + I_z r / m \ell_f$, respectively, we then obtain the equation

$$D_R \sin \left(C_{Ratan} \left(B_{Ratan} \left(\frac{(m \ell_f \ell_r - I_z) r - m \ell_f y_2}{m \ell_f y_1} \right) \right) \right) \frac{\ell_f + \ell_r}{m \ell_f} - \dot{y}_2 - y_1 r = 0. \quad (7.35)$$

Equation (7.26) is implied by (7.35).

b) To show sufficiency, we need to prove that for given flat outputs $y_1(t)$, $y_2(t)$, one can solve for the trajectory $x^*(t) = [V^*(t), \beta^*(t), r^*(t)]^T$ that generates $y_1(t)$ and $y_2(t)$. This is, indeed, guaranteed by the property of differential flatness of the system. The

implicit function theorem [199, 200] guarantees the local solvability of (7.26). The conditions are summarized in Theorem 7.5.2, from which one can recover the yaw rate $r^*(t)$. The states $V^*(t)$ and $\beta^*(t)$ are then calculated directly using the equations (7.24) and (7.25) in Section 7.4.2. \square

Theorem 7.5.2 *Suppose there exists an open set $\Omega \subseteq \mathbb{R}^4$ and a point $P^* = (y_1^*, y_2^*, \dot{y}_2^*, r^*) \in \Omega$, such that $g(P^*) = 0$ and*

$$\left. \frac{\partial g}{\partial r} \right|_{P^*} = D_R C_R B_R \frac{\cos \left(C_R \text{atan} \left(B_R \text{atan} \left(\Pi(y_1^*, y_2^*, r^*) \right) \right) \right) (m\ell_f \ell_r - I_z)}{\left(1 + B_R^2 \text{atan}^2 \left(\Pi(y_1^*, y_2^*, r^*) \right) \right) \left(1 + \Pi^2(y_1^*, y_2^*, r^*) \right)} - \frac{m\ell_f y_1^*}{\ell_f + \ell_r} \neq 0, \quad (7.36)$$

where

$$\Pi(y_1^*, y_2^*, r^*) = \frac{(m\ell_f \ell_r - I_z)r^* - m\ell_f y_2^*}{m\ell_f y_1^*}, \quad (7.37)$$

then there must be some neighborhood $\mathcal{N}(P^*) \subseteq \Omega$ about P^* and a continuous function $G: \mathbb{R}^3 \mapsto \mathbb{R}$ such that $r = G(y_1, y_2, \dot{y}_2)$ for all $(y_1, y_2, \dot{y}_2, r) \in \mathcal{J}$, where $\mathcal{J} \triangleq \{(y_1, y_2, \dot{y}_2, r) \in \mathcal{N}(P^*) | g(y_1, y_2, \dot{y}_2, r) = 0\}$.

For details about the implicit function theorem one can refer to [200]. Based on the idea in Proposition 7.5.1, a feasible trajectory for the vehicle from A to B that satisfies all boundary conditions in Table 7.2 can be designed as follows.

1) **Path design:** We need to generate a smooth path connecting A and B in Figure 7.6, which is denoted as \widehat{AB} . The tangent lines to \widehat{AB} at A and B must be parallel with the direction of the vehicle's velocity (see Figure 7.7). We use the cubic Bézier curve to generate \widehat{AB} [62], which is constructed using four control points, namely, P_0, \dots, P_3 , and is represented by

$$\gamma(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, \quad t \in [0, 1], \quad (7.38)$$

where $P_0 = A$, $P_3 = B$, and the points P_1, P_2 are unknown and need to be designed.

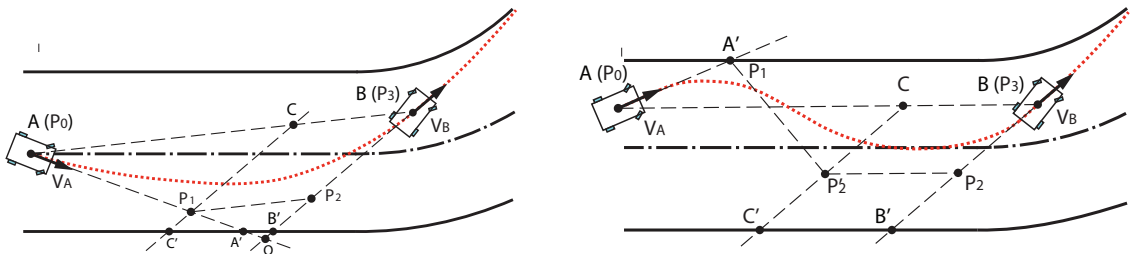


Figure 7.7: Path planning for guiding control.

In this work the optimal choices for the control points P_1, P_2 are determined by minimizing the jerk energy of the Bézier curve \widehat{AB} , namely, we want to solve the following

optimization problem,

$$P_1^*, P_2^* = \arg \min_{P_1, P_2} E(\gamma) = \int_0^1 \|\gamma'''(P_1, P_2, t)\|^2 dt. \quad (7.39)$$

By substituting (7.38) into the jerk energy functional $E(\gamma)$ in (7.39), one obtains the following equation,

$$E(\gamma) = 36\| -P_0 + 3P_1 - 3P_2 + P_3 \|^2. \quad (7.40)$$

We recall that a Bézier curve lies inside the convex hull defined by its control points. The path \overline{AB} must satisfy the road boundary constraints if the control points P_1 and P_2 are inside the lane. Hence, if one can find two points P_1, P_2 inner the lane such that $\overline{P_1P_2} \parallel \overline{P_0P_3}$ and the length $\|P_1P_2\| = \|P_0P_3\|/3$ (see Figure 7.7(left)), then such P_1, P_2 are the optimal solution and the optimal jerk energy is $E^*(\gamma) = 0$. Nevertheless, in some cases the optimal jerk energy $E^*(\gamma)$ cannot be zero and the corresponding optimal control points P_1^* and P_2^* may not be obtained inner the lane. For instance, in Figure 7.7(right) the optimal control point P_1^* is selected on the road boundary in order to minimize the jerk energy, which is given by $E^*(\gamma) = 324\|P_1P_2'\|^2$ in this case. We summarize the path planning algorithm in Algorithm 11.

Algorithm 11 Path Generation Using Cubic Bézier Curves

Input: A, B, V_A, V_B

Output: $P_1^*, P_2^*, \gamma(\tau)$

- 1: $P_0 \leftarrow A, P_3 \leftarrow B$
 - 2: Determine C on segment \overline{AB} : $\|BC\| = \|AB\|/3$
 - 3: Determine B' on the outer boundary where $\overline{BB'} \parallel V_B$
 - 4: Determine C' on the outer boundary where $\overline{CC'} \parallel \overline{BB'}$
 - 5: Determine A' on the inner/outer boundary where $\overline{AA'} \parallel V_A$
 - 6: **if** $A' \in$ outer boundary **then**
 - 7: **if** $\overline{AA'}$ crosses $\overline{CC'}$ (ideal case: $E^*(\gamma) = 0$) **then**
 - 8: $P_1^* \leftarrow$ intersection of $\overline{AA'}$ and $\overline{CC'}$
 - 9: Determine P_2^* on $\overline{BB'}$ where $\overline{P_1^*P_2^*} \parallel \overline{AB}$
 - 10: **else**
 - 11: $P_1^* \leftarrow A'$
 - 12: Determine P_2^* on $\overline{BB'}$ where $\overline{C'P_2^*} \parallel \overline{AB}$
 - 13: **else**
 - 14: $P_1^*, P_2^* \leftarrow \arg \min_{P_1 \in \overline{AA'}, P_2 \in \overline{CC'}} \|P_1P_2'\|$ (see [201])
 - 15: Determine P_2^* on $\overline{BB'}$ where $\overline{P_2^*P_2^*} \parallel \overline{AB}$
 - 16: $\gamma(\tau) = (1-\tau)^3P_0 + 3(1-\tau)^2\tau P_1^* + 3(1-\tau)\tau^2P_2^* + \tau^3P_3, \quad \tau \in [0, 1]$
-

2) **Speed profile design:** In order to achieve maximum speed, this work refers to [202] to design the speed profile along the trajectory generated using Algorithm 11, with the

extension that we use a single track model instead of a single mass model as in [202]. This allows to take into account the longitudinal load transfer arising from the accelerating/braking of the vehicle.

We formulate an optimal control problem to find the optimal speed profile. Let \hat{f}_{ij} ($i = F, R$ and $j = x, y$) denote the longitudinal and lateral friction forces at the front and rear wheels represented in the frame $X_V - C.M. - Y_V$ (fixed on the chassis, see Figure 1.3). The components \hat{f}_{ij} should not be confused with the components f_{ij} in (4.1a)-(4.1c). In the case of the vehicle having rear wheel drive of a differential type, we assume that: a) The lateral tire force of the front wheel \hat{f}_{Fy} is always able to balance the moment arising from the lateral tire force of the rear wheel \hat{f}_{Ry} , and b) The time derivative of the yaw rate along the trajectory is small, namely, $\dot{r}(s) \approx 0$, where s denotes the length of the path. Hence, one can approximate the front wheel tire forces from

$$\hat{f}_{Fy} = \hat{f}_{Ry} \ell_r / \ell_f, \quad (7.41a)$$

$$\hat{f}_{Fx} = -\hat{f}_{Fy} \tan(\delta - \alpha_F) = -\hat{f}_{Ry} \ell_r (\ell_f + \ell_r) / \ell_f R(s), \quad (7.41b)$$

where $R(s)$ denotes the radius. We use $\bar{\mu}$ to denote the peak friction coefficient and assume that the velocity $V(s)$ satisfies $V^2(s) - \bar{\mu} g R(s) \leq 0$. Then the limit of the acceleration a can be determined by solving (7.41a)-(7.41b) and the equations (7.42a)-(7.42d),

$$\hat{f}_{Ry} = m V^2(s) \ell_f / (\ell_f + \ell_r) R(s), \quad (7.42a)$$

$$ma = \hat{f}_{Rx} + \hat{f}_{Fx}, \quad (7.42b)$$

$$\bar{\mu} f_{Rz} = \sqrt{\hat{f}_{Ry}^2 + \hat{f}_{Rx}^2}, \quad (7.42c)$$

$$f_{Rz}(\ell_f + \ell_r) = mah + mg \ell_f, \quad (7.42d)$$

The result is given by

$$a_{\min} = -\frac{\mathcal{B} - \sqrt{\Delta}}{2\mathcal{A}}, \quad (7.43)$$

$$a_{\max} = -\frac{\mathcal{B} + \sqrt{\Delta}}{2\mathcal{A}}, \quad (7.44)$$

where $\mathcal{A} = \bar{\mu}^2 h^2 / (\ell_f + \ell_r)^2 - 1$, $\mathcal{B} = 2\bar{\mu}^2 g h \ell_f / (\ell_f + \ell_r)^2 - 2V^2 \ell_r / R^2$, and $\Delta = \mathcal{B}^2 - 4\mathcal{A}\mathcal{C}$, and where $\mathcal{C} = \bar{\mu}^2 g^2 \ell_f^2 / (\ell_f - \ell_r)^2 - V^4 \ell_r^2 / R^4 - V^4 \ell_f^2 / (\ell_f - \ell_r)^2 R^2$. If we define the new states $z_1 \triangleq s$, $z_2 \triangleq ds/dt = V$, the state equations may be written as

$$\dot{z}_1 = z_2, \quad (7.45a)$$

$$\dot{z}_2 = -\frac{\mathcal{B}(z_1, z_2) + u \sqrt{\Delta(z_1, z_2)}}{2\mathcal{A}}, \quad u \in [-1, 1]. \quad (7.45b)$$

Next, we derive the feasibility condition for (7.45). Since it is not possible to speed up the vehicle when the total tire force of the driving wheel reaches its peak value, it follows

that $m\dot{z}_2 = \hat{f}_{Rx} + \hat{f}_{Fx} = 0$, while (7.42c) holds. The feasibility region for (7.45) is given by

$$S \triangleq \{(z_1, z_2) : C_0(z_1, z_2) \triangleq z_2^2 - \bar{\mu}gR(z_1) / \sqrt{1 + \ell_r^2(\ell_f + \ell_r)^2 / \ell_f^2 R^2(z_1)} \leq 0\}. \quad (7.46)$$

We want to determine the optimal control u that drives (7.45) along a given trajectory in minimum time t_f subject to (7.46). Mathematically, we want to solve the optimization problem

$$\min_u t_f, \quad (7.47)$$

subject to (7.46) and the following boundary conditions

$$z_1(t_0) = z_{10}, \quad z_1(t_f) = z_{1f}, \quad z_2(t_0) = z_{20}, \quad z_2(t_f) = z_{2f}.$$

Next, we solve problem (7.47). The Hamiltonian for this problem is given by

$$H(z, \kappa, u) = 1 + \kappa_1 z_2 - \kappa_2 \frac{\mathcal{B}(z_1, z_2) + u\sqrt{\Delta(z_1, z_2)}}{2\mathcal{A}} + \kappa_3 C_0(z_1, z_2). \quad (7.48)$$

The dynamics equations for the adjoint states are $\dot{\kappa}_1 = -\partial H / \partial z_1$ and $\dot{\kappa}_2 = -\partial H / \partial z_2$, which are given by

$$\begin{aligned} \dot{\kappa}_1 = & \frac{\kappa_2}{2\mathcal{A}} \frac{\partial \mathcal{B}(z_1, z_2)}{\partial R(z_1)} R'(z_1) + \frac{\kappa_2 u}{4\mathcal{A}\sqrt{\Delta(z_1, z_2)}} \frac{\partial \Delta(z_1, z_2)}{\partial R(z_1)} R'(z_1) \\ & + \kappa_3 \bar{\mu}g \frac{1 + 2\ell_r^2(\ell_f + \ell_r)^2 / \ell_f^2 R^2(z_1)}{(1 + \ell_r^2(\ell_f + \ell_r)^2 / \ell_f^2 R^2(z_1))^{3/2}} R'(z_1), \end{aligned} \quad (7.49a)$$

$$\dot{\kappa}_2 = -\kappa_1 + \frac{\kappa_2}{2\mathcal{A}} \frac{\partial \mathcal{B}(z_1, z_2)}{\partial z_2} + \frac{\kappa_2 u}{4\mathcal{A}\sqrt{\Delta(z_1, z_2)}} \frac{\partial \Delta(z_1, z_2)}{\partial z_2} + \bar{\mu}gR'(z_1) - 2\kappa_3 z_2, \quad (7.49b)$$

Lemma 7.5.3 Assume $R'(z_1) \neq 0$ and $R'(z_1) \neq -\Gamma(R(z_1)) / \Pi(R(z_1))$ for any $z_1 \in (z_1(t_0), z_1(t_f))$, where

$$\Gamma(R(z_1)) = \left(2\bar{\mu}g\ell_f\ell_r / \sqrt{\ell_f^2 R^2(z_1) + \ell_r^2(\ell_f + \ell_r)^2} - \bar{\mu}^2 gh\ell_f / (\ell_f + \ell_r) \right) / (1 + \bar{\mu}^2 h^2), \quad (7.50)$$

$$\Pi(R(z_1)) = \bar{\mu}g \left(1 + 2\ell_r^2(\ell_f + \ell_r)^2 / \ell_f^2 R^2(z_1) \right) / 2 \left(1 + \ell_r^2(\ell_f + \ell_r)^2 / \ell_f^2 R^2(z_1) \right)^{3/2}. \quad (7.51)$$

Then the manifold $\partial S \triangleq \{(z_1, z_2) : C_0(z_1, z_2) = 0\}$ is not invariant for the system (7.45) for any control u .

Proof: Invariance of ∂S implies the following equation,

$$z_2 \dot{z}_2 - \Pi(R(z_1)) R'(z_1) \dot{z}_1 = 0, \quad (7.52)$$

Since $\dot{z}_1 = z_2 > 0$, (7.52) is equivalent to

$$\dot{z}_2 - \Pi(R(z_1))R'(z_1) = 0. \quad (7.53)$$

Given $C_0(z_1, z_2) = 0$, one is able to derive the following condition using (7.41) and (7.42),

$$\dot{z}_2^2 + \dot{z}_2 \Gamma(R(z_1)) = 0. \quad (7.54)$$

Equations (7.53) and (7.54) imply either $\dot{z}_2 = R'(z_1) = 0$ or $R'(z_1) = -\Gamma(R(z_1))/\Pi(R(z_1))$, which contradicts the given conditions in Lemma 7.5.3. \square

The paths satisfying $R'(z_1) = -\Gamma(R(z_1))/\Pi(R(z_1))$ are not common to see. An interesting result of Lemma 7.5.3 is that ∂S may be invariant under (7.45) only for the paths having constant curvature.

Proposition 7.5.4 *Assuming that $C_0(z_1, z_2) < 0$ holds throughout the optimal trajectory, then there can be no singular subarcs.*

Proof: The Kuhn-Tucker condition implies that $\kappa_3 C_0(z_1, z_2) = 0$ and the transversality condition implies $H(t_f) = 0$. Since the Hamiltonian does not depend explicitly on time, it follows that

$$H(t) = 0, \quad \forall t \in [t_0, t_f]. \quad (7.55)$$

Assume that the constraint is inactive, namely, $C_0(z_1, z_2) < 0$, hence one obtains $\kappa_3 = 0$ (Kuhn-Tucker condition). It follows from Pontryagin's Maximum Principle that the optimal control is given by

$$u^* = \arg \min_{u \in [-1, 1]} H(z, \kappa, u) = \begin{cases} -1, & \text{for } \kappa_2 > 0, \\ 1, & \text{for } \kappa_2 < 0, \end{cases} \quad (7.56)$$

which further implies that $u^* = -\text{sgn } \kappa_2(t)$ where κ_2 is the switching function. Suppose there exists a singular control interval $(t_1, t_2) \subset [t_0, t_f]$ such that $\kappa_2(t) = 0$ for $\forall t \in (t_1, t_2)$. $\kappa_2(t) = 0$ also implies that $\dot{\kappa}_2(t) = 0$. Then it follows from (7.49b) that $\kappa_1 = 0$ and hence $H(t) = 1$ for $t \in (t_1, t_2)$, which contradicts condition (7.55). \square

Proposition 7.5.5 *Assuming that $C_0(z_1, z_2) < 0$ holds throughout the optimal trajectory and the curvature of the trajectory is constant or monotonically decreasing/increasing, then there can be at most one switching in the control from $u = 1$ to $u = -1$.*

Proof: Suppose there exists a switching time $t_1 \in (t_0, t_f)$ such that $\kappa_2(t_1) = 0$. Then the transversality condition (7.55) implies $\kappa_1(t_1) = -1/z_2(t_1)$. Based on the fact that the velocity $z_2 > 0$, one obtains $\kappa_1(t_1) < 0$ and hence $\dot{\kappa}_2(t_1) > 0$. Therefore κ_2 changes sign from negative to positive at t_1 , and the control switches from $u = 1$ to $u = -1$.

Similarly, one can see that any other switching point in (t_0, t_f) has to be from $u = 1$ to $u = -1$. Since there can be no two consecutive switching points from $u = 1$ to $u = -1$

without a switching point from $u = -1$ to $u = 1$ inbetween, t_1 can be the only switching point in the interval (t_0, t_f) \square

Proposition 7.5.5 indicates that for a given path from A to B with constant or monotonically decreasing/increasing curvature, one can construct the speed profile by forward integration of (7.45) from (z_1^A, z_2^A) with $u = 1$ and by backward integration of (7.45) from (z_1^B, z_2^B) with $u = -1$. This result is enough for this work since we want to generate the speed profile for the vehicle driving on a segment of straight road, where the path curvature can be approximately treated as 0. For more general cases where the path is composed of a finite number of segments of constant curvature and segments of monotonically decreasing/increasing curvature, the optimal control solutions for a single mass vehicle model can be found in [202].

3) **Yaw motion design:** According to Proposition 7.5.1, the yaw motion of the vehicle $\psi(t)$ must be carefully designed with respect to the following boundary conditions and the nonlinear constraint in (7.57),

$$\begin{aligned} \psi(t=0) &= \psi_A, & \psi(t=t_f) &= \psi_B, & \dot{\psi}(t=0) &= r_A, \\ \dot{\psi}(t=t_f) &= r^{ss}, & g(\dot{\psi}(t)) &= 0, \end{aligned} \quad (7.57)$$

where the constraint function g was defined in (7.26).

4) **Feasibility analysis:** The trajectory designed following equations (7.38)-(7.57) does not take into account the steering range of the front wheel, and hence the trajectory is required to satisfy the following condition,

$$\min_{\delta \in [\underline{\delta}, \bar{\delta}]} f_{\text{Fy}}(\delta) \cos(\delta) \leq \Lambda(t) \leq \max_{\delta \in [\underline{\delta}, \bar{\delta}]} f_{\text{Fy}}(\delta) \cos(\delta), \quad (7.58)$$

where $\underline{\delta}$ and $\bar{\delta}$ denote the allowed minimal and maximal steering angles of the front wheel. The term $\Lambda(t)$ in (7.58) has the following expression,

$$\Lambda = \frac{m y_1 r \ell_r + m \dot{y}_2 \ell_r - I_z \dot{r} (1 + \ell_r / \ell_f)}{\ell_f + \ell_r}. \quad (7.59)$$

The inequality (7.58) provides the necessary condition a feasible trajectory has to satisfy. One can design a path $\gamma(\tau)$ following Algorithm 11, and then one can design $V(t)$ and $r(t)$ subject to the constraint in (7.58).

7.6 Control Design

A switching control is designed to achieve the tasks for the three different stages. In the guiding stage, a tracking controller is designed such that the vehicle is driven to reach the desired steady-state at the entry of the corner. Afterwards, a stabilizing controller is designed such that the vehicle follows the steady-state along the desired path. In the exiting stage, the vehicle leaves the corner and switches to a new control mode depending on the specific task. In this chapter we design a state feedback controller that aligns the

posture of the vehicle to be parallel with the road in the exiting stage.

Next, we design the tracking controller that drives the vehicle from A to B following the trajectory designed in Section 7.5.2 and also design a stabilizing controller that keeps the vehicle along BPC in steady state using LQR theory.

7.6.1 Tracking Controller

We denote the desired flat output as $y^d(t)$ and denote the current output from the plant as $y(t)$. Then the tracking error is $e(t) = y(t) - y^d(t)$ and the accumulated tracking error is $\zeta(t) = \int_0^t e(\tau) d\tau + \zeta_0$. The following design drives $\zeta(t), e(t) \rightarrow 0$ as $t \rightarrow \infty$ with proper choices of λ and ν ,

$$\dot{e}_1 = -\lambda_1 e_1 - \lambda_2 \zeta_1, \quad (7.60a)$$

$$\ddot{e}_2 = -\nu_1 \dot{e}_2 - \nu_2 e_2 - \nu_3 \zeta_2. \quad (7.60b)$$

To design λ and ν , we further let $\mathcal{E} = [\zeta_1, e_1, \zeta_2, e_2, \dot{e}_2]^T$. Then the dynamics of the error \mathcal{E} is given by $\dot{\mathcal{E}} = A\mathcal{E}$, where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -\lambda_2 - \lambda_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -\nu_3 - \nu_2 - \nu_1 & 0 & 0 \end{bmatrix}. \quad (7.61)$$

If we denote the set of the eigenvalues of A as $\text{mspec}(A) = \{s_1, s_2, s_3, s_4, s_5\}$, then it can be shown that

$$\lambda_1 = -s_1 - s_2, \quad (7.62a)$$

$$\lambda_2 = s_1 s_2, \quad (7.62b)$$

$$\nu_1 = s_3 s_4 + s_3 s_5 + s_4 s_5, \quad (7.62c)$$

$$\nu_2 = -s_3 - s_4 - s_5, \quad (7.62d)$$

$$\nu_3 = -s_3 s_4 s_5. \quad (7.62e)$$

Hence λ and ν can be determined by assigning the eigenvalues s_1, s_2, \dots, s_5 with appropriate negative real parts. It then follows from (7.60) that

$$\dot{y}_1 = \dot{y}_1^d - \lambda_1 e_1 - \lambda_2 \zeta_1, \quad (7.63a)$$

$$\ddot{y}_2 = \ddot{y}_2^d - \nu_1 \dot{e}_2 - \nu_2 e_2 - \nu_3 \zeta_2, \quad (7.63b)$$

and the control u is obtained by solving the following equations:

$$\mathcal{L}_f y_1(x, u) = \dot{y}_1, \quad \mathcal{L}_f^2 y_2(x, u) = \ddot{y}_2, \quad (7.64)$$

where $\mathcal{L}_f y_1(x, u)$ and $\mathcal{L}_f^2 y_2^2(x, u)$ are given by

$$\mathcal{L}_f y_1(x, u) = -\frac{\sin \delta}{m} f_{Fy} + \frac{\cos \delta}{m} f_{Fx} + \frac{1}{m} f_{Rx} + Vr \sin \beta, \quad (7.65a)$$

$$\begin{aligned} \mathcal{L}_f^2 y_2^2(x, u) = & \left(\frac{r}{m} \sin \delta - \frac{\ell_f V}{I_z} \cos \delta \cos \beta \right) f_{Fy} + \frac{\ell_r V}{I_z} \cos \beta f_{Ry} - \left(\frac{r}{m} \cos \delta + \frac{\ell_f V}{I_z} \sin \delta \sin \beta \right) f_{Fx} \\ & - \frac{r}{m} f_{Rx} - Vr^2 \sin \beta + \frac{\ell_f + \ell_r}{m \ell_f} \frac{\partial f_{Ry}}{\partial \alpha_R} \left(\frac{\partial \alpha_R}{\partial V} \dot{V} + \frac{\partial \alpha_R}{\partial \beta} \dot{\beta} + \frac{\partial \alpha_R}{\partial r} \dot{r} \right), \end{aligned} \quad (7.65b)$$

and where

$$\frac{\partial f_{Ry}}{\partial \alpha_R} = \frac{D_R \cos(C_R \text{atan}(B_R \alpha_R)) C_R B_R}{1 + (B_R \alpha_R)^2}, \quad (7.66a)$$

$$\frac{\partial \alpha_R}{\partial V} = -\frac{\ell_r r \cos \beta}{V^2 \cos^2 \beta + (V \sin \beta - \ell_r r)^2}, \quad (7.66b)$$

$$\frac{\partial \alpha_R}{\partial \beta} = -\frac{V^2 - V \ell_r r \sin \beta}{V^2 \cos^2 \beta + (V \sin \beta - \ell_r r)^2}, \quad (7.66c)$$

$$\frac{\partial \alpha_R}{\partial r} = \frac{V \ell_r \cos \beta}{V^2 \cos^2 \beta + (V \sin \beta - \ell_r r)^2}. \quad (7.66d)$$

The (local) solvability of equations in (7.64) is guaranteed following the implicit function theorem if and only if the following conditions are satisfied

$$\frac{I_z V \sin \beta (\partial f_{Ry} / \partial V) + I_z \cos \beta (\partial f_{Ry} / \partial \beta) + m V \ell_f (\partial f_{Ry} / \partial r)}{m^2 \ell_f^2} - \frac{V^2 \cos \beta}{\ell_f + \ell_r} \neq 0, \quad (7.67)$$

$$f_{Fy} \sin \delta - \frac{\partial f_{Fy}}{\partial \delta} \cos \delta \neq 0. \quad (7.68)$$

where $\frac{\partial f_{Ry}}{\partial V}$, $\frac{\partial f_{Ry}}{\partial \beta}$ and $\frac{\partial f_{Ry}}{\partial r}$ are calculated following (7.66) using the chain rule, and $\frac{\partial f_{Fy}}{\partial \delta}$ is given by

$$\frac{\partial f_{Fy}}{\partial \delta} = \frac{D_F \cos(C_F \text{atan}(B_F \alpha_F)) C_F B_F}{1 + (B_F \alpha_F)^2}. \quad (7.69)$$

One can refer to [196] for more analysis on the solvability of (7.64). The scheme of the controller design can be shown in Figure 7.8.

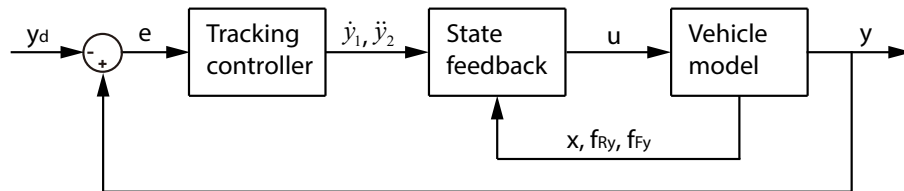


Figure 7.8: Scheme of flatness-based vehicle dynamics control.

7.6.2 Sliding Controller

After the vehicle reaches position B in Figure 1.3, the controller switches from the tracking control mode to the stabilizing control mode. To this end, we first linearize the vehicle model about the steady state $x^{ss} = [V^{ss}, \beta^{ss}, r^{ss}]^T$ [84] as follows,

$$\dot{x} = A^{ss}x + B^{ss}u, \quad (7.70a)$$

$$y = Cx, \quad (7.70b)$$

where

$$x = \begin{bmatrix} V - V^{ss} \\ \beta - \beta^{ss} \\ r - r^{ss} \end{bmatrix}, \quad u = \begin{bmatrix} \delta - \delta^{ss} \\ f_{Fx} - f_{Fx}^{ss} \\ f_{Rx} - f_{Rx}^{ss} \end{bmatrix}. \quad (7.71)$$

Then the optimal control that minimizes the following performance index

$$\min_u J = \int_0^{t_f} (x^T Q x + u^T R u) dt, \quad (7.72)$$

is given by $u = -R^{-1}(B^{ss})^T P x$, where $P > 0$ is the solution of the algebraic Riccati equation

$$(A^{ss})^T P + P A^{ss} - P B^{ss} R^{-1} (B^{ss})^T P + C^T Q C = 0. \quad (7.73)$$

It is worth mentioning that, the controller designed using (7.70)-(7.72) only maintains the state of the vehicle in the velocity level, which does not fix the tracking errors about certain predesigned trajectory. In order to improve the performance of the vehicle for successfully passing through the corner, we consider the following methods: 1) We calculate a couple of available steady states that have similar V^{ss} but the cornering radii R^{ss} are different. The controller switches to different steady states to change the cornering radius such that it is able to avoid hitting onto the road boundaries. 2) We could redefine the nominal trajectory by including the desired position/orientation into the reference state vector, such that the controller is able to maintain the distance of the vehicle from certain predesigned cornering center (fix the cornering radius).

Let us consider the polar coordinates (R, θ) of the vehicle in the corner (see Figure 7.6(right)), where R denotes the distance of the vehicle from the cornering center O^{ss} and θ denotes the angle between the position vector and the X_I axis. The kinematics equations can be formulated as follows:

$$\dot{R} = V \cos(\beta + \psi - \theta), \quad (7.74a)$$

$$\dot{\theta} = V \sin(\beta + \psi - \theta) / R, \quad (7.74b)$$

We then linearize the vehicle model about the new nominal state $x^{ref} = [V^{ss}, \beta^{ss}, r^{ss}, R^{ss}, \theta, \psi]^T$ at each time step, and redesign the state feedback stabilizing controller following (7.70)-(7.72), similarly.

7.6.3 Exiting Controller

After the vehicle exits steady-state cornering, we want to steer the vehicle to run along the straight line. We consider the steady state for straight-line driving $\hat{x}^{ss} = [\hat{V}^{ss}, \hat{\beta}^{ss}, \hat{r}^{ss}]^T = [V^{ss}, 0, 0]^T$, where V^{ss} takes the value of the exit velocity of the vehicle after the corner. In order to align the vehicle to be parallel with the road, we include a target yaw angle ψ^{ss} into the state vector, that is, $x^{ss} = [V^{ss}, 0, 0, \psi^{ss}]^T$. The corresponding control for steady-state straight-line driving is $\hat{u}^{ss} = [\hat{\delta}^{ss}, \hat{f}_{Fx}^{ss}, \hat{f}_{Rx}^{ss}]^T = [0, 0, 0]^T$. We linearize the vehicle model about the new equilibria \hat{x}^{ss} and \hat{u}^{ss}

$$\begin{aligned}\dot{\hat{x}} &= \hat{A}^{ss} \hat{x} + \hat{B}^{ss} \hat{u}, \\ \hat{y} &= \hat{C} \hat{x},\end{aligned}\tag{7.75a}$$

where

$$\hat{x} = \begin{bmatrix} V - \hat{V}^{ss} \\ \beta - \hat{\beta}^{ss} \\ r - \hat{r}^{ss} \\ \psi - \hat{\psi}^{ss} \end{bmatrix}, \quad \hat{u} = \begin{bmatrix} \delta - \hat{\delta}^{ss} \\ f_{Fx} - \hat{f}_{Fx}^{ss} \\ f_{Rx} - \hat{f}_{Rx}^{ss} \end{bmatrix}.\tag{7.76}$$

In order to drive $\hat{x} \rightarrow 0$ as $t \rightarrow \infty$, we consider the state feedback control law $\hat{u} = \hat{K} \hat{x}$, where \hat{K} is taken such that the matrix $\hat{A}^{ss} + \hat{B}^{ss} \hat{K}$ is Hurwitz. To this end, we solve the following linear matrix inequalities (LMIs) for the matrices Q and S ,

$$\hat{A}Q + BS + Q\hat{A}^T + S^T \hat{B}^T < 0,\tag{7.77a}$$

$$Q > 0.\tag{7.77b}$$

It follows from (7.77) that $\hat{K} = SQ^{-1}$. The state feedback gain \hat{K} can also be designed using an LQR solution (see Section 7.6.2).

We then apply acceleration control to speed up the vehicle after it enters steady-state straight-line driving, using the following control law

$$f_{Fx} = \text{sat}(f_{Fx}), \quad f_{Rx} = \text{sat}(f_{Rx}),\tag{7.78}$$

where sat is the saturation function.

7.7 Numerical Simulations

In this section we implement and validate the proposed control architecture in simulations and analyze the results. We plan the high-speed cornering trajectories for different corner angles and implement the tracking controller, the stabilizing controller and the exit feedback controller, to generate the entry, sliding and exiting stages of the high-speed cornering maneuvers respectively.

7.7.1 Trajectory Design

We first calculate the equilibrium for the steady-state cornering using the vehicle model introduced in Section 7.2. Table 7.3 shows the equilibria for steady state-cornering with different speeds.

Table 7.3: Equilibrium for steady-state cornering.

	1	2	3	4	5	6	7	8	9
V^{ss} [m/s]	5	8.5	9.4	10.8	11.9	12.1	13.9	15	6
β^{ss} [deg]	-2.8	-30	-20	-20	-28	-20	-23	-32	-14.3
r^{ss} [deg/s]	11.3	48.7	35.9	30.9	34.1	27.7	26.5	28	57.3
R^{ss} [m]	25.4	10	15	20	20	25	30	30.7	6

A feasible trajectory that includes the target steady-state cornering process is designed following (7.38)-(7.58). In order to demonstrate the proposed trajectory planning and controller design, we take the first equilibrium in Table 7.3 and use the segmentation in Figure 7.6(left) for instance. The radius R_{ref} of the corner is 10 [m], and the straight line segments have lengths $S_1 = 20$ [m] and $S_2 = 15$ [m]. The distance d between the inner and outer constraints is 2 [m].

Table 7.4: Boundary conditions

	t [s]	V [m/s]	β [deg]	r [deg/s]	X [m]	Y [m]	ψ [deg]
Position A	0	9	0	-9.6	0	0	78
Position B	4	5	-2.8	11.3	1	20	122

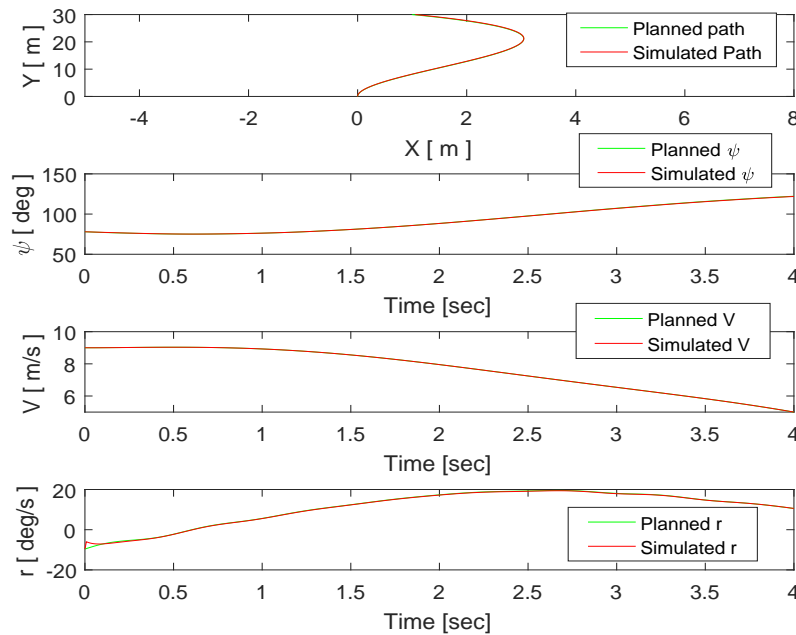


Figure 7.9: The desired and simulated trajectories.

For the boundary conditions and the designed time t given in Table 7.4, we plan the trajectory and the motion of the vehicle and plot them using green curves in Figure 7.9. The designed trajectory in Figure 7.9 must satisfy the feasibility condition in (7.58). We assume that the maximal steering angle is $\bar{\delta} = -\underline{\delta} = 0.5$ [rad] (28.6 [deg]), and the peak friction coefficient of the road surface is 0.5. Figure 7.10 shows the required steering and traction commands with their allowed maximal/minimal values due to the steering limit of the vehicle and the road condition. The dashed lines indicate the upper/lower bounds for δ and f_{Rx} , respectively. Since the green lines in Figure 7.10 are within the upper/lower bounds, this trajectory should be possible to be tracked with the current vehicle model.

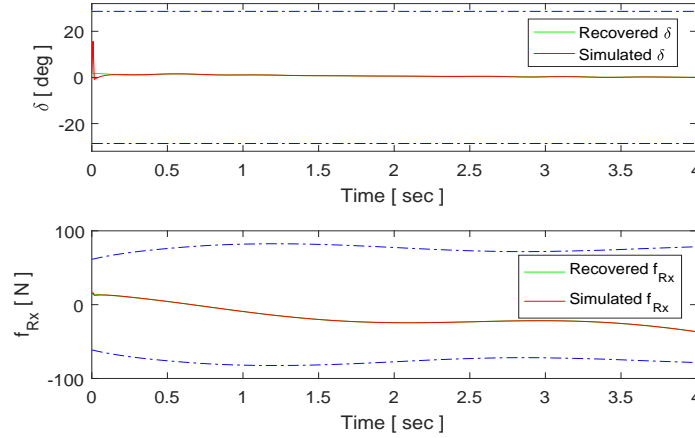


Figure 7.10: The desired and simulated controls.

7.7.2 Tracking Control

We assign the eigenvalues of the matrix A to be $s_1 = s_2 = s_3 = -10$ and $s_4 = s_5 = -5$ such that the matrix A is Hurwitz. It follows from (7.62) that the corresponding parameters for the tracking controller are

$$\lambda_1 = 20, \quad \lambda_2 = 100, \quad \nu_1 = 125, \quad \nu_2 = 20, \quad \nu_3 = 250. \quad (7.79)$$

We can then implement the tracking controller using numerical simulations. The plots in Figure 7.11 show that the desired flat outputs y_1 and y_2 and the simulated results agree well with each other.

We compare the simulated trajectory with the trajectory we designed in Section 7.7.1. The results in Figure 7.9 and Figure 7.11 show that the flatness-based tracking controller is able to track the trajectory designed following the procedure in (7.38)-(7.58).

Since the system is differentially flat, the control $u = [\delta, f_{Rx}]^T$ ($f_{Fx} = 0$ and hence omitted) to achieve the desired trajectory can be recovered from the designed flat output by solving equations (7.32). We plot the calculated control and the simulated result in Figure 7.10. The results in Figure 7.10 also verify the feasibility conditions provided in (7.58)-(7.58). Finally, we recover the lateral tire forces f_{iy} ($i = F, R$) using (7.2), which are

plotted against the simulation results in Figure 7.12.

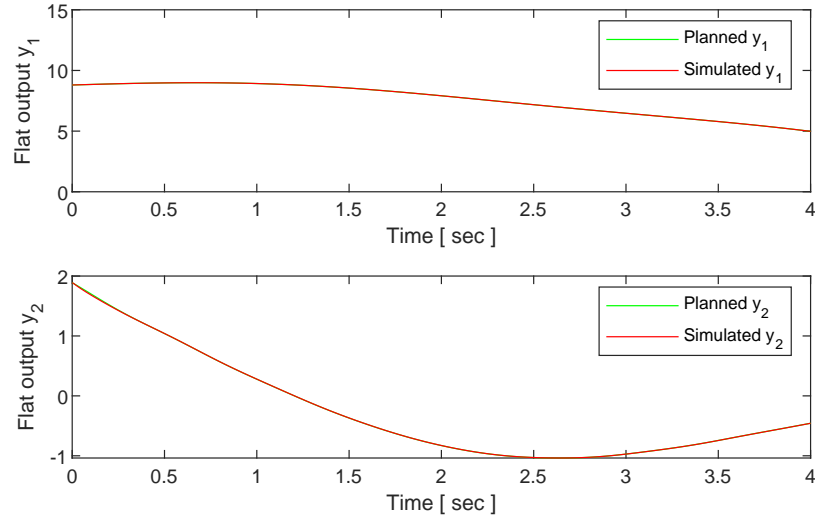


Figure 7.11: The desired and simulated output.

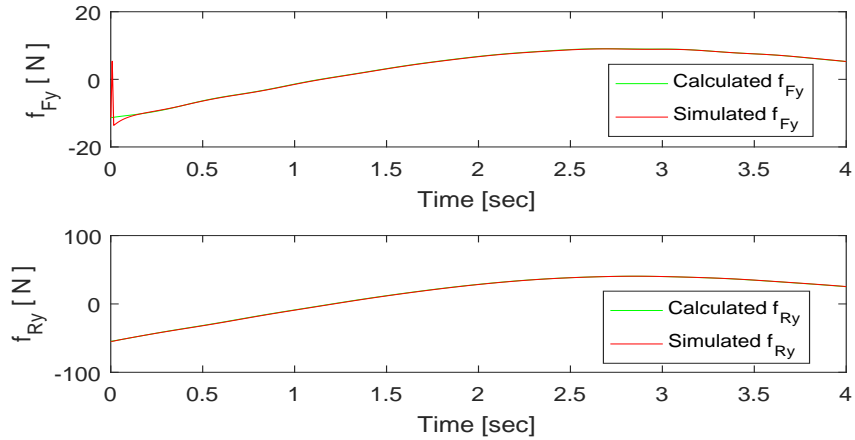


Figure 7.12: The desired and simulated lateral tire forces.

The above results show that if the reference trajectory and the tracking controller are properly designed, the entire system can be accurately recovered using the history of the flat outputs, including the state, the control and the other internal variables such as the forces.

We then implement the LQR controller after the vehicle reaches position B and implement the state feedback controller after the vehicle exits the corner. The complete trajectory is plotted in Figure 7.13. The desired high-speed cornering maneuver of the vehicle can be achieved using the hybrid-mode control that switches between the flatness-based controller, the LQR controller and the state feedback controller for the exiting stage.

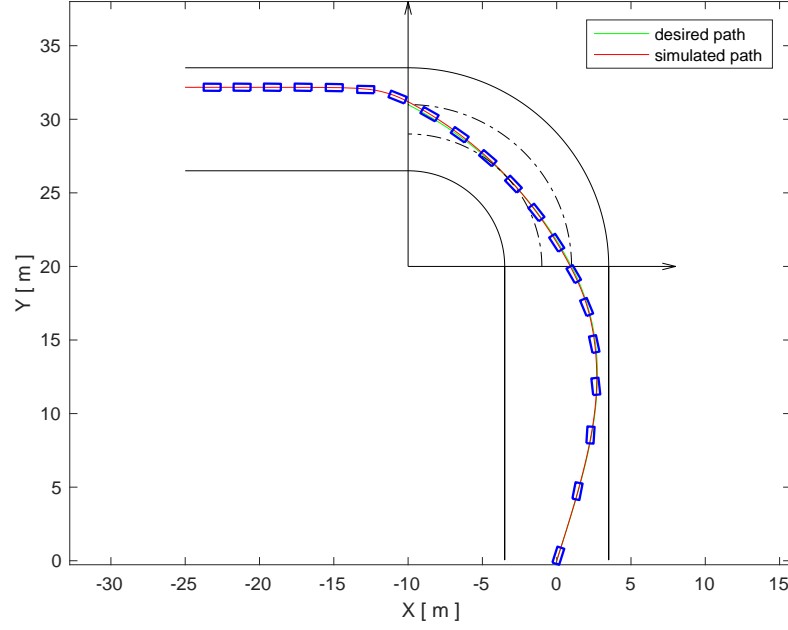


Figure 7.13: Switching-mode control for steady-state cornering.

7.7.3 Late-Apex High-Speed Cornering

In this section we generate high-speed cornering trajectories having “late apex” as shown in Figure 7.6(right), following the similar control design procedures in Section 7.6.1-7.6.3. We use the second and the third equilibria in Table 7.3 for steady-state cornering to achieve a high-speed, high-slip sliding maneuver.

Table 7.5: Road geometry and high-speed cornering trajectory setup.

Road geometry	S_1 [m]	S_2 [m]	R_{ref} [m]	D [m]	Corner angle [deg]		
	15	15	10	5	60/90/120/180		
Initial condition	V [m/s]	β [deg]	r [deg/s]	(X, Y) [(m,m)]	ψ [deg]		
	15	0	0	$(R_{\text{ref}}, -S_1)$	90		
	V^{ss} [m/s]	β^{ss} [deg]	r^{ss} [deg/s]	$\angle 1$ [deg]	$\angle 2$ [deg]	$\angle 3$ [deg]	$\angle 4$ [deg]
60 deg	9.4	-20	35.9	6	30	18	6
90 deg	9.4	-20	35.9	9	54	18	9
120 deg	8.5	-30	48.7	12	84	12	12
180 deg	8.5	-30	48.7	18	126	18	18

The road geometry is defined using the parameters in Table 7.5. It is worth mentioning that the other equilibria in Table 7.3 have steady-state cornering radius $R^{\text{ss}} > 20$ [m], which seems too large and is not convenient to use for the road geometry in Table 7.5, especially when the corner angle is larger than 90 deg. For high-speed cornering with different corner angles, Table 7.5 provides appropriate choices of the equilibria and the angles $\angle 1, \dots, \angle 4$ that determine the geometry of different high-speed cornering trajectories.

We plan high-speed cornering trajectories for different corner angles and implement the proposed switch-mode controller accordingly. The planned and simulated trajectories are plotted in Figure 7.14.

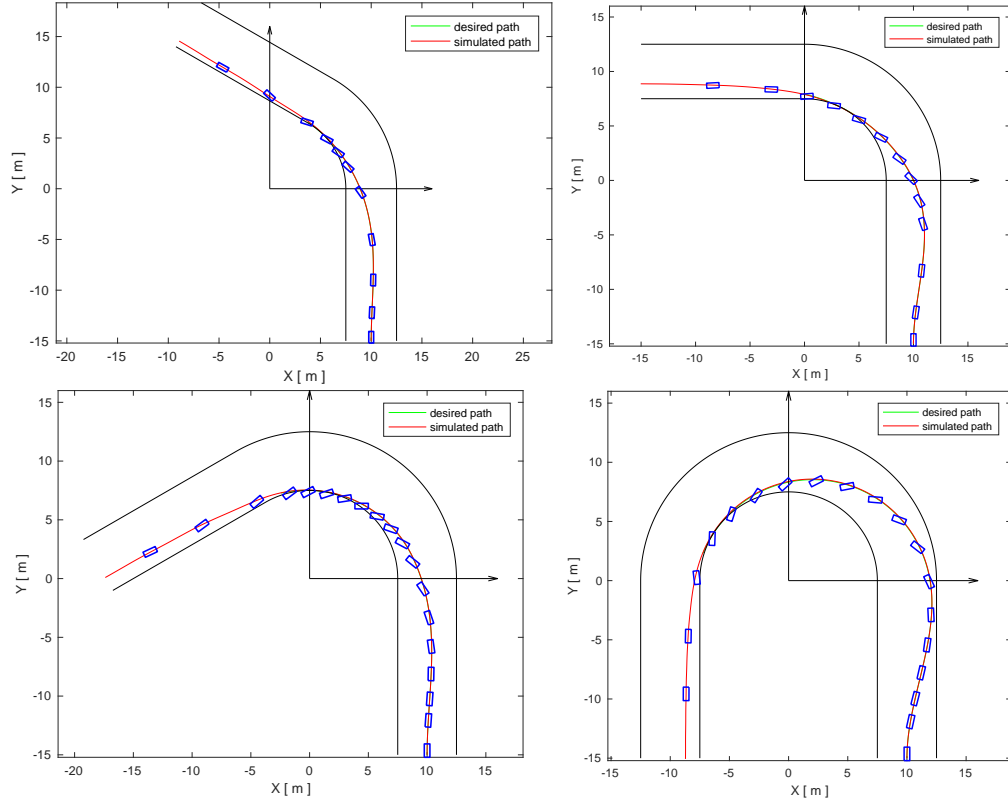


Figure 7.14: Trail braking maneuver generation for different road geometries.

Figure 7.14 shows the computed high-speed cornering trajectories for 60 deg, 90 deg, 120 deg and 180 deg cornering, respectively. In each figure, the simulated trajectory agrees well with the planned trajectory. By designing the angles $\angle 1, \dots, \angle 4$, one achieves high-speed cornering with different “late apex” features.

The design of the angles $\angle 1, \dots, \angle 4$ must take into account the target equilibrium of the steady-state cornering. If the radius of the trajectory is very large (i.e., more than two times of the corner radius), high-speed cornering with “late apex” will be hard to generate since the target path may cut the outer boundary of the road. One can use another equilibrium with a smaller steady-state cornering radius or change the angles $\angle 1, \dots, \angle 4$ properly (i.e., to reduce $\angle 2$ to move the tangent point P farther from the exit of the corner). Recommended values of $\angle 1, \dots, \angle 4$ are given in Table 7.6:

Table 7.6: “Late apex” specification.

Corner angle	$\angle 1$	$\angle 2$	$\angle 3$	$\angle 4$
Ω	0.1Ω	$(0.5 \sim 0.7)\Omega$	$(0.1 \sim 0.3)\Omega$	0.1Ω

7.8 Experimental Validation

In this section we generate high-speed cornering maneuvers by implementing the switch-mode controller on a closed track, using both the high-fidelity CarSim vehicle model and the auto-rally platform [203].

7.8.1 CarSim Simulation

We first show the simulated high-speed cornering trajectories using CarSim/Simulink software [123]. The vehicle model parameters are given by Table 6.4.

The geometry of the track is shown in Figure 7.15, where the width of the track is 16 [m], and the inner and the outer corner radii are 5[m] and 21 [m], respectively. The distance between the two corner centers is 60 [m]. The friction coefficient of the road is set to be 0.6.



Figure 7.15: Closed track in CarSim.

We then calculate the steady state of the vehicle model using the parameters in Table 6.4. In this work we generate high-speed cornering using the initial condition and the steady state in Table 7.7.

Table 7.7: Boundary conditions

	Speed [km/h]	Sideslip [deg]	Yaw rate [rad/s]
Initial condition	65	0	0
Target SS	29.44	-50	0.68

The switch-mode controller is implemented in the following orders: 1) Given any current state of the vehicle is in the straight road, the tracking controller calculates the entry point B of the corner based on the given map of the track. 2) Next, the tracking controller plans a smooth path connecting vehicle's current position and point B following Algorithm 11. We implement the tracking control for the next 0.2 seconds. The control

commands (i.e., steering angle, throttle and braking wheel cylinder pressures) are published at 100 Hz. 3) Repeat step 2 until the vehicle reaches the desired steady-state at the target position B. 4) Switch to LQR controller after the vehicle enters the corner for SS sliding (100 Hz). 5) Switch to exiting controller to align the vehicle to follow the straight road after the vehicle is about to leave the corner (100 Hz). 6) Speed up the vehicle after it exits the corner to the maximum allowable velocity. 7) Repeat steps 1-6 to finish a couple of circles along the track. Figure 7.16 shows the velocity, sideslip angle and yaw rate of the vehicle for a complete circle on the track. The vehicle slides through the corners at the two ends in the desired steady state from $t = 5.5$ [sec] to $t = 7.5$ [sec] and from $t = 17$ [sec] to $t = 19$ [sec], respectively. The animation is available on the DCSL Youtube channel ¹.

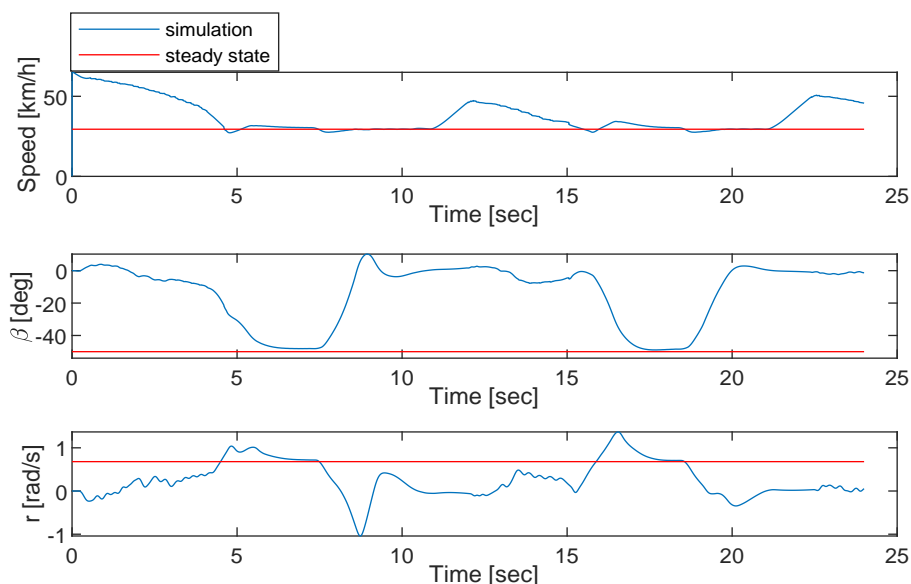


Figure 7.16: Simulated trajectories.

7.8.2 Auto-Rally Experiments

The simulation results using CarSim validates the control design by applying it on a high-fidelity vehicle model. Nevertheless, these simulations do not consider the performance of hardware and assume there is no computational delay, which is not realistic for real-time maneuvers. We therefore validated the control design on an fifth-scale Auto-Rally platform developed at Georgia Tech. The Auto-Rally platform and the test track are shown in Figure 7.17.

The Auto-Rally platform is driven by two rear wheels, and its top speed can reach up to 27 [m/s]. The size of the vehicle measures about 1[m] \times 0.6[m] \times 0.4[m]. The perimeter of the centerline of the test track is about 63 [m], and the width is about 3.4 [m]. A more detailed description of the vehicle model and the test track can be found in [203].

¹<https://www.youtube.com/watch?v=VN9Tpr8D1tk>



Figure 7.17: The test track and the Auto-Rally vehicle platform.

The vehicle and tire model parameters in Table 7.8 are estimated using the approach in [45, 204] with the data collected from the field test.

Table 7.8: Vehicle/tire model parameters.

m [kg]	21.5	total mass	m^s [kg]	18.03	sprung mass												
m_f [kg]	0.84	front wheel mass	m_r [kg]	0.89	rear wheel mass												
w_f [m]	0.44	front track	w_r [m]	0.46	rear track												
L [m]	0.57	wheel base	R [m]	0.095	wheel radius												
I_z [kgm ²]	1.02	rotationary inertia	ℓ_f [m]	0.46	distance of CM to front axle												
<table> <tr> <td>B_F</td><td>C_F</td><td>D_F [N]</td><td>B_R</td><td>C_R</td><td>D_R [N]</td></tr> <tr> <td>4.03</td><td>0.02</td><td>1192.16</td><td>2.45</td><td>0.02</td><td>5414.88</td></tr> </table>						B_F	C_F	D_F [N]	B_R	C_R	D_R [N]	4.03	0.02	1192.16	2.45	0.02	5414.88
B_F	C_F	D_F [N]	B_R	C_R	D_R [N]												
4.03	0.02	1192.16	2.45	0.02	5414.88												

Figure. 7.18 shows the estimated tire force calculated using the expression in (7.2) with the identified tire parameters from Table 7.8. The lateral tire force f_{iy} ($i = F, R$) can be approximated using (7.2) with satisfactory accuracy.

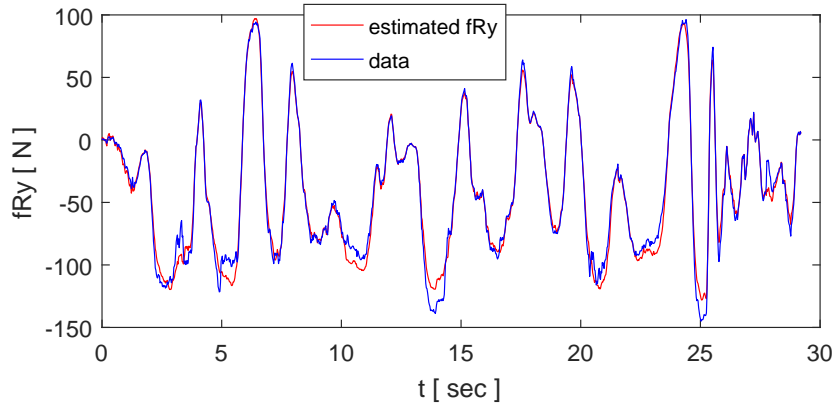


Figure 7.18: The estimated lateral tire force.

We developed the controller using the ROS software framework. The controller serves as a ROS node and publishes a throttle command and a steering command at 50 Hz. In order to validate the control design before running a test on the real track, we implemented the controller on the Gazebo simulator (see [203]) and the trajectory is shown in Figure 7.19. The target steady state is given by the last column of Table 7.3.

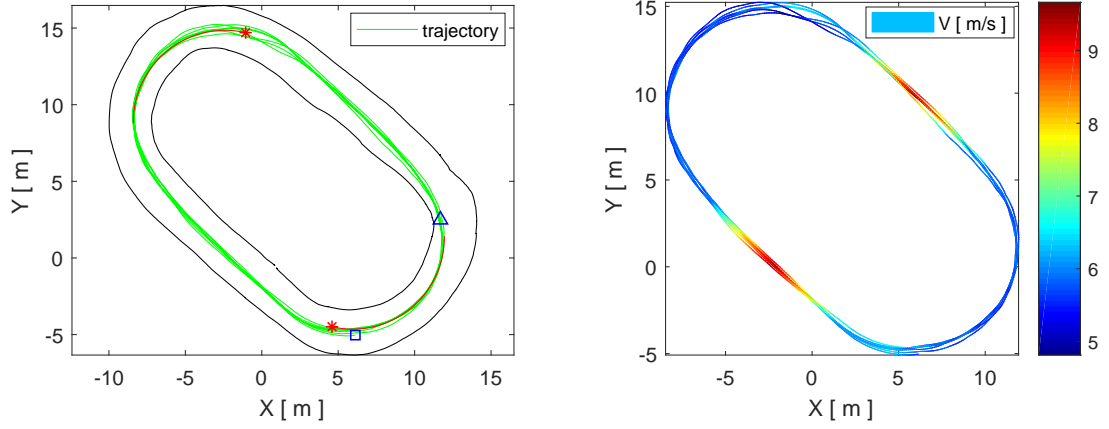


Figure 7.19: The trajectories of Auto-Rally (counter clockwise).

The start and end points of the trajectory are marked with a triangle and a square signs, respectively. The red stars indicate the target points where the vehicle starts steady state cornering (point B in Figure 7.6), and the red arcs indicate the desired steady state cornering trajectories. We measure the boundaries of the track and show them in black curves. The result shows that the flatness-based tracking controller is able to steer the vehicle into the corner with the desired speed, and the fixed-radius stabilizing controller is able to maintain the high sideslip sliding process of the vehicle following the desired trajectories. The speed of the vehicle is shown in Figure 7.19 using a color map. The minimum speed is about 6 [m/s] in the corner and the maximum speed is about 9.6 [m/s] near the midpoint of the straight road segments.

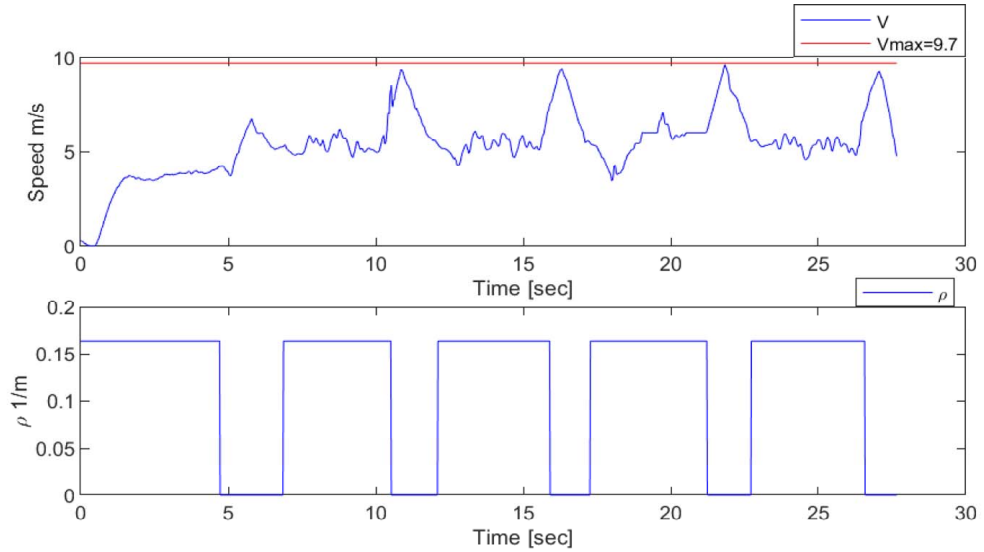


Figure 7.20: The speed profile of Auto-Rally.

Next, we implemented the controller using the Auto-Rally platform on the real track. Figure 7.20 shows the speed profile of the vehicle for a couple of rounds in the tests. The maximum speed is 9.7 [m/s], which is similar to the highest speed achieved in previ-

ous work [142]. The online path replanning process during the guiding stage is demonstrated by the green curves in Figure 7.21, where a new Bézier curve is generated every 0.1 seconds. We also compared the speed profile generated using the single track

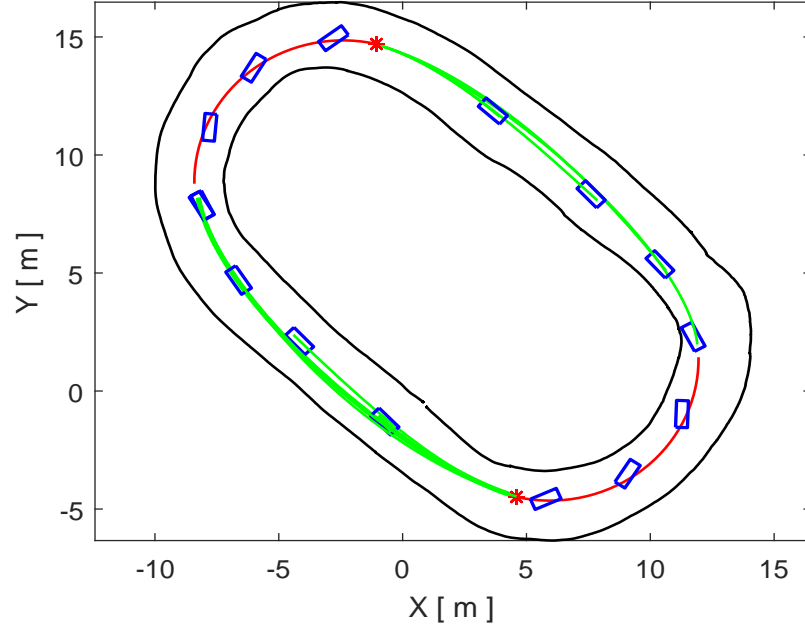


Figure 7.21: Online path replanning.

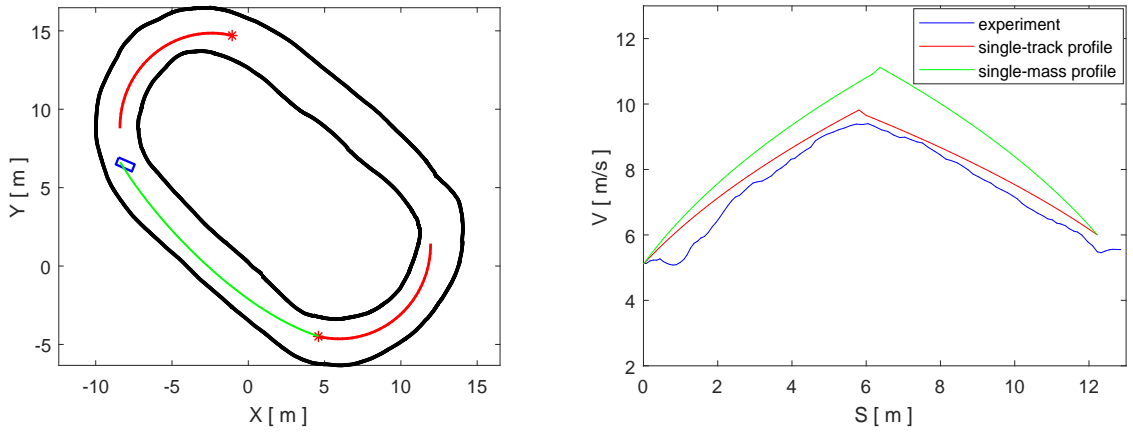


Figure 7.22: Online speed profile generating.

model and the speed profile generated using the single mass model [202]. We show the result in Figure 7.22 (right), which corresponds to the green trajectory in Figure 7.22 (left). The maximum tire friction coefficient in the experiments is about 0.8. One sees from Figure 7.22 that the speed profile generated using the single track model agrees better with the experimental result than the speed profile generated using the single mass model. One also notices that the speed profile generated using the single mass model is symmetric about the axis through the peak. The reason is that the single mass

model does not take into account the longitudinal load transfer arising from the accelerating/braking operations, and assumes that the maximum longitudinal tire forces for accelerating and braking are equal in magnitude. Moreover, the single mass model uses all wheels for the longitudinal control of the vehicle, which is not true for the Auto-Rally that has a rear wheel drive differential type, and consequentially, overestimates the maximum speed the vehicle is able to achieve.

Figure 7.23 shows a typical trajectory of the Auto-Rally vehicle in a complete round of experiment, where the vehicle enters the corner from the desired position (the red stars) and slides through the corner at high speed (5.5-6.5 [m/s]) and high sideslip (10-20 deg). The video demonstrating this high-speed cornering maneuver is available on the DCSL Youtube channel².

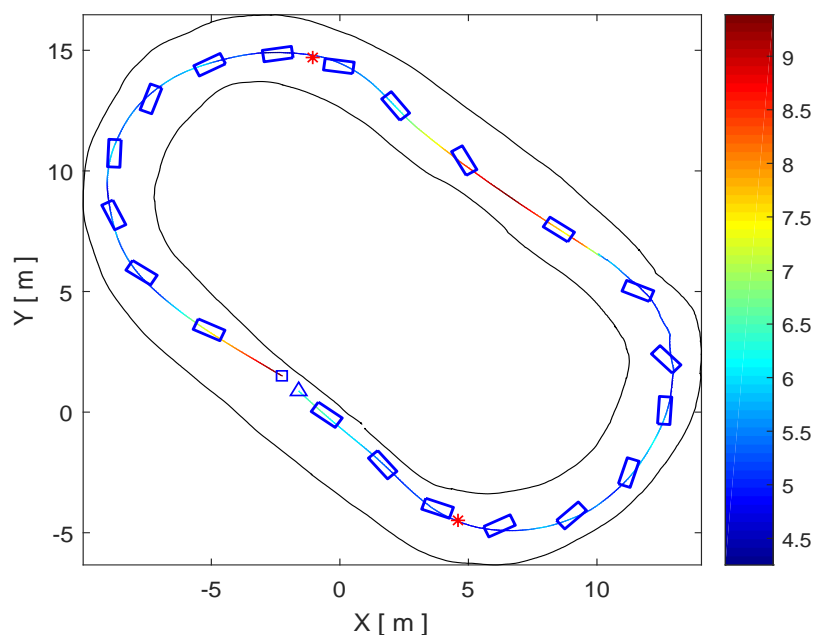


Figure 7.23: The posture of Auto-Rally in a typical round.

7.9 Conclusion

High-speed cornering is a technique often used by expert rally racing drivers. This dissertation provides a methodology to generate high-speed cornering maneuvers in semi-analytic form. We first use a trajectory learning technique to find a primitive trajectory that captures the “essence” of a high-speed cornering maneuver, using a series of high-speed cornering demonstrations. Based on this primitive trajectory we approximately divide the high-speed cornering trajectory into three stages, namely, the entry, sliding and exiting stages, and we show that the middle the sliding stage includes a segment of steady-state cornering. We then develop a switching-mode control for each stage using different control techniques to generate high-speed cornering.

²<https://youtu.be/3vJDDw6BEiY>

Based on the differential flatness property of the vehicle, we provide a method to plan the trajectory and design a tracking control. By changing the position of the tangential point of the vehicle's trajectory and the inner boundary of the road, we can successfully generate 60 deg, 90 deg, 120 deg and 180 deg high-speed cornering maneuvers at will with different "late apex" features. We demonstrate high-speed cornering maneuvers by implementing the switch-mode controller either in simulations using CarSim/Simulink software or on an Auto-Rally experimental platform. Since our approach plans the nominal trajectory based on geometry, one does not need to solve optimal control problems on-the-fly, and hence the computation effort is sufficiently low for real-time high-speed cornering maneuvers generation.

CHAPTER 8

CONCLUSIONS & FUTURE RESEARCH DIRECTIONS

8.1 Conclusions

Autonomous vehicles represent the main trend in future intelligent transportation systems, which promise to improve traffic safety while, at the same time, increase fuel efficiency and reduce congestion. In order to develop the control techniques for autonomous vehicles, this dissertation intends to understand and mimic the behavior of the expert human driver. We successfully generate two typical expert (aggressive) driving maneuvers in two different scenarios, namely, highway overtaking and off-road autonomous rally racing.

The parameterized two-point visual driver model is utilized to characterize the steering behavior of a human driver. In order to validate this driver model, we conducted a series of field tests at Ford's Dearborn Development Center (DDC) test facility. We show that the two-point visual driver model is able to satisfactorily predict human driver behavior and driving style. Moreover, the results of our investigation indicate that some of the driver parameters are not exactly constant, but rather vary slowly during a driving task. Reliable design of an ADAS controller should take into account this effect in order to consistently provide good performance.

The wavelet transform provides insights into the driver's control signal, in terms of the number and the location of the singularities of the signal and the distribution characteristics of the associated Lipschitz exponents. Our analysis shows that the steering wheel torque of an experienced driver has fewer singularities, and the Lipschitz exponents seem to follow a comparatively more concentrated distribution. These can be used to characterize the control signal into different levels of smoothness.

Instead of using the standard EKF/UFK for nonlinear parameter estimation, we propose an adaptive limited memory UKF algorithm (ALM-UKF) that is able to estimate the system state, model parameters and the Kalman filter hyperparameters related to the noise simultaneously, hence making possible to provide on-line estimates of the model parameters. We validate the ALM-UKF with both CarSim simulation data and experimental data from a fifth-scale Auto-Rally vehicle.

In order to reproduce the expert driving style in highway traffic, we use a stochastic Markov decision process to model the behaviors of the surrounding vehicles in traffic, and achieve desired driving maneuvers using both reinforcement learning (RL) and inverse reinforcement learning (IRL). We propose and validate the new concept of "dynamic cell" on an in-house developed traffic simulator. Since the "dynamic cell" is able to dynamically extract the essential state of the traffic according to different vehicle velocities, driver intents (i.e., lane-switching, braking, etc.) and the sizes of the surrounding vehicles (i.e., truck, sedan, etc.), the dimensionality of the state space can be main-

tained in a manageable level, and hence the reinforcement learning problem is easily scalable.

The driver's reward function can also be recovered from driving data using IRL. We generalize the formulation of the MaxEnt IRL by using a reward function in the form of a linear combination of the parameterized features, and we show, for the first time, that the reward function in the MaxEnt IRL formulation can take any nonlinear form. Particularly, we propose and validate three new MaxEnt deep IRL algorithms to solve the model-free MDP problem.

The lane-switching task can be separated into a path planning task and a tracking control task. We formulate two different algorithms for real-time path planning using both the joint quadratic Bézier curves and the fourth-order Bézier curves subject to certain curvature constraint. The fourth-order Bézier curves are everywhere C^2 continuous, and hence it is smooth and easy for tracking control.

Highway overtaking is an expert driving technique where the autonomous vehicle needs to be able to safely interact with the surrounding traffic vehicles. High-speed cornering is another expert driving maneuver we want to generate. This technique is often used by experienced drivers in off-road rally racing.

Based on the trajectory learning results, we are the first to show the existence of a segment of sustained steady-state cornering. Hence, we approximately divide the high-speed cornering trajectory into three stages, namely, the entry, sliding and exiting stages. We design a hybrid-mode control strategy for different stages separately, using a combination of linear and nonlinear control techniques. We propose a path planning algorithm that utilizes cubic Bézier curves to minimize the jerk energy, which leads to smooth paths with gradually changing curvature.

In order to generate minimum-time-travel speed profiles for high-speed cornering, we extend the approach proposed in [202] by using a single track vehicle model, instead of a point mass model. The new speed profile generation algorithm is able to consider the longitudinal load transfer arising from the accelerating/braking process, which better represents reality. Since our approach avoids solving optimal control problems on-the-fly, this approach requires low computation effort, while guaranteeing good racing performance in terms of the highest speed the vehicle achieved in off-road racing.

8.2 Future Work

Future work may consider the following directions to extend this research:

- 1) Based on driver parameter estimation, a potential next step would be to use machine learning ideas to distinguish the behaviors of the drivers, and to classify the drivers into distinct categories using the features arising from the wavelet transform (i.e., number of singularities, Lipschitz exponents, etc.).

- 2) This dissertation assumes there is only one agent (intelligent vehicle) in traffic. A possible direction for extension may introduce multiple agents incorporated into the MDP traffic model to coordinate multiple vehicles simultaneously, such that one can

better control the traffic flow (i.e., traffic congestion mitigation).

3) Other possible directions for extension may consider more driving scenarios by incorporating traffic merging/splitting, pedestrians, traffic signals and more road intersections.

4) The path planning and control algorithms for highway overtaking has been validated on a traffic simulator. Future work may implement and validate these algorithms in a real-world traffic system.

5) The Auto-Rally experimental platform uses only the rear wheels for longitudinal control. Future work may consider to generate high-speed cornering using the platforms having different differential types (i.e., FWD, AWD, etc.).

REFERENCES

- [1] NHTSA(2015), “Traffic safety facts 2014: A compilation of motor vehicle crash data from the fatality analysis reporting system and the general estimates system,” Department of Transportation, National Highway Traffic Safety Administration, Washington, DC, USA, Tech. Rep. DOT HS 812 261.
- [2] NHTSA *et al.*, “2015 motor vehicle crashes: Overview,” *Traffic safety facts research note*, vol. 2016, pp. 1–9, 2016.
- [3] D. Hendricks, J. Fell, and M Freedman, “The relative frequency of unsafe driving acts in serious traffic crashes,” *Report no: DOT-HS-809-206*, 2001.
- [4] W. G. Najm, M. D. Stearns, H. Howarth, J. Koopmann, and J. Hitz, “Evaluation of an automotive rear-end collision avoidance system,” Department of Transportation, National Highway Traffic Safety Administration, Washington, DC, USA, Tech. Rep. Technical Report DOT HS 810 569, 2006.
- [5] G. Li, S. E. Li, and B Cheng, “Field operational test of advanced driver assistance systems in typical chinese road conditions: The influence of driver gender, age and aggression,” *International Journal of Automotive Technology*, vol. 16, no. 5, pp. 739–750, 2015.
- [6] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, no. 1, p. 6, 2017.
- [7] D. H. Weir and D. T. McRuer, “Measurement and interpretation of driver steering behavior and performance,” SAE Technical Paper, Tech. Rep., 1973.
- [8] C. C. MacAdam, “An optimal preview control for linear systems,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 3, pp. 188–190, 1980.
- [9] —, “Application of an optimal preview control for simulation of closed-loop automobile driving,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 393–399, 1981.
- [10] R. Hess and A Modjtahedzadeh, “A control theoretic model of driver steering behavior,” *IEEE Control Systems Magazine*, vol. 10, no. 5, pp. 3–8, 1990.
- [11] I Kageyama and H. Pacejka, “On a new driver model with fuzzy control,” *Vehicle System Dynamics*, vol. 20, no. sup1, pp. 314–324, 1992.

- [12] A. Burgett and R. Miller, "Using parameter optimization to characterize driver's performance in rear end driving scenarios," in *Proceedings: International Technical Conference on the Enhanced Safety of Vehicles*, National Highway Traffic Safety Administration, vol. 2003, Nagoya, Japan, 2003, p. 21.
- [13] Y Lin, P Tang, W. Zhang, and Q. Yu, "Artificial neural network modelling of driver handling behaviour in a driver-vehicle-environment system," *International Journal of Vehicle Design*, vol. 37, no. 1, pp. 24–45, 2005.
- [14] C. Cacciabue, *Modelling Driver Behaviour in Automotive Environments: Critical Issues in Driver Interactions with Intelligent Transport Systems*. Springer, 2007.
- [15] M. Flad, C. Trautmann, G. Diehm, and S. Hohmann, "Individual driver modeling via optimal selection of steering primitives," in *World Congress*, vol. 19, Cape Town, South Africa, 2014, pp. 6276–6282.
- [16] A Modjtahedzadeh and R. Hess, "A model of driver steering control behavior for use in assessing vehicle handling qualities," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, no. 3, pp. 456–464, 1993.
- [17] C. Chatzikomis and K. Spentzas, "A path-following driver model with longitudinal and lateral control of vehicle's motion," *Forschung im Ingenieurwesen*, vol. 73, no. 4, p. 257, 2009.
- [18] S. D. Keen and D. J. Cole, "Application of time-variant predictive control to modelling driver steering skill," *Vehicle System Dynamics*, vol. 49, no. 4, pp. 527–559, 2011.
- [19] R. Hamada, T. Kubo, K. Ikeda, Z. Zhang, T. Shibata, T. Bando, K. Hitomi, and M. Egawa, "Modeling and prediction of driving behaviors using a nonparametric Bayesian method with AR models," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 131–138, 2016.
- [20] C. W. De Silva, *Modeling and control of engineering systems*. CRC Press, 2009.
- [21] D. L. Wilson and R. A. Scott, "Parameter determination for a Crossover driver model," Department of Mechanical Engineering and Applied Mechanics, Tech. Rep. UM-MEAM-83-17, 1983.
- [22] L. Saleh, P. Chevrel, F. Claveau, J.-F. Lafay, and F. Mars, "Shared steering control between a driver and an automation: Stability in the presence of driver behavior uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 974–983, 2013.

- [23] C. Sentouh, P. Chevrel, F. Mars, and F. Claveau, "A sensorimotor driver model for steering control," in *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, 2009, pp. 2462–2467.
- [24] D. D. Salvucci and R. Gray, "A two-point visual control model of steering," *Perception*, vol. 33, no. 10, pp. 1233–1248, 2004.
- [25] J. Steen, H. J. Damveld, R. Happee, M. M. van Paassen, and M. Mulder, "A review of visual driver models for system identification purposes," in *IEEE International Conference on Systems, Man, and Cybernetics*, Anchorage, AK, 2011, pp. 2093–2100.
- [26] J. V. Beck and K. J. Arnold, *Parameter estimation in engineering and science*. New York: John Wiley and Sons, 1977.
- [27] Z. Zhang, "Parameter estimation techniques: A tutorial with application to conic fitting," *Image and Vision Computing*, vol. 15, no. 1, pp. 59–76, 1997.
- [28] F. R. Hampel, "Robust estimation: A condensed partial survey," *Probability Theory and Related Fields*, vol. 27, no. 2, pp. 87–104, 1973.
- [29] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005, vol. 571.
- [30] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, Alberta, Canada, 2000, pp. 153–158.
- [31] M. Wu and A. W. Smyth, "Application of the unscented Kalman filter for real-time nonlinear structural system identification," *Structural Control and Health Monitoring*, vol. 14, no. 7, pp. 971–990, 2007.
- [32] S. Hong, T. Smith, F. Borrelli, and J. K. Hedrick, "Vehicle inertial parameter identification using extended and unscented Kalman filters," in *IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 1436–1441.
- [33] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.
- [34] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the Carnegie-Mellon NAVLAB," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.

- [35] T. Jochem, D. Pomerleau, B. Kumar, and J. Armstrong, "PANS: A portable navigation platform," in *Proceedings of the Intelligent Vehicles' 95 Symposium*, Detroit, MI, 1995, pp. 107–112.
- [36] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
- [37] A. J. Hawkins. (2017). Google's new self-driving minivans will be hitting the road at the end of January 2017, (visited on 01/08/2017).
- [38] J. Berr. (2016). Uber's audacious plan to replace human drivers, (visited on 08/25/2016).
- [39] C. Thompson. (2016). Tesla just revealed new cars and Model 3 will have fully self-driving hardware, (visited on 10/19/2016).
- [40] D. Lee. (2016). Ford's self-driving car 'coming in 2021', (visited on 08/17/2016).
- [41] V. Carlström. (2017). Volvo just launched the world's most ambitious autonomous driving trial in Gothenburg, (visited on 01/10/2017).
- [42] Auto Tech. (2017). 44 corporations working on autonomous vehicles, (visited on 05/18/2017).
- [43] A Farina, B Ristic, and D Benvenuti, "Tracking a ballistic target: Comparison of several nonlinear filters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 854–867, 2002.
- [44] G. Chowdhary and R. Jategaonkar, "Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter," *Aerospace Science and Technology*, vol. 14, no. 2, pp. 106–117, 2010.
- [45] C. You and P. Tsiotras, "Vehicle modeling and parameter estimation using adaptive limited memory joint-state UKF," in *American Control Conference*, Seattle, WA, 2017, pp. 322–327.
- [46] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *arXiv preprint arXiv:1602.01580*, 2016.
- [47] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, 2011, pp. 1537–1542.

- [48] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.
- [49] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [50] K. Yi, T. Chung, J. Kim, and S. Yi, "An investigation into differential braking strategies for vehicle stability control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 217, no. 12, pp. 1081–1093, 2003.
- [51] S. Di Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1236–1248, 2013.
- [52] L. De Novellis, A. Sorniotti, P. Gruber, and A. Pennycott, "Comparison of feedback control techniques for torque-vectoring control of fully electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 3612–3623, 2014.
- [53] L. De Novellis, A. Sorniotti, P. Gruber, L. Shead, V. Ivanov, and K. Hoepping, "Torque vectoring for electric vehicles with individually controlled motors: State-of-the-art and future developments," in *26th Electric Vehicle Symposium*, Los Angeles, CA, 2012.
- [54] J. Ackermann, T. Bunte, and D. Odenthal, "Advantages of active steering for vehicle dynamics control," in *Proceedings of the 32nd International Symposium on Automotive Technology and Automation*, Vienna, Austria, 1999, pp. 263–270.
- [55] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.
- [56] Y. A. Ghoneim, W. C. Lin, D. M. Sidlosky, H. H. Chen, and Y.-K. Chin, "Integrated chassis control system to enhance vehicle stability," *International Journal of Vehicle Design*, vol. 23, no. 1-2, pp. 124–144, 2000.
- [57] Y. Kou, "Development and evaluation of integrated chassis control systems," PhD thesis, The University of Michigan, 2010.
- [58] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 20–37, 2006.

- [59] S. Zafeiropoulos and P. Tsiotras, "Design of a lane-tracking driver steering assist system and its interaction with a two-point visual driver model," in *American Control Conference*, Portland, OR, 2014, pp. 3911–3917.
- [60] C. You, J. Lu, and P. Tsiotras, "Driver parameter estimation using joint E-/UKF and dual E-/UKF under nonlinear state inequality constraints," in *IEEE International Conference on Systems, Man, and Cybernetics*, Budapest, Hungary, 2016.
- [61] R. Cimurs, J. Hwang, and I. H. Suh, "Bezier curve-based smoothing for path planner with curvature constraint," in *IEEE International Conference on Robotic Computing*, Taichung, Taiwan, 2017, pp. 241–248.
- [62] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science.*, IEEE, San Francisco, CA, 2008, pp. 158–166.
- [63] J.-w. Choi, R. E. Curry, and G. H. Elkaim, "Continuous curvature path generation based on Bézier curves for autonomous vehicles.," *International Journal of Applied Mathematics*, vol. 40, no. 2, 2010.
- [64] T. Shim, G. Adireddy, and H. Yuan, "Autonomous vehicle collision avoidance system using path planning and model-predictive-control-based active front steering and wheel torque control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering*, vol. 226, no. 6, pp. 767–778, 2012.
- [65] M. A. Mousavi, Z. Heshmati, and B. Moshiri, "LTV-MPC based path planning of an autonomous vehicle via convex optimization," in *21st Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, Iran, 2013, pp. 1–7.
- [66] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems*, Hague, Netherlands, 2013, pp. 2063–2067.
- [67] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *International Joint Conference on Neural Networks (IJCNN)*, Brisbane, QLD, Australia, 2012, pp. 1–8.
- [68] W. Chee and M. Tomizuka, "Vehicle lane change maneuver in automated highway systems," 1994.
- [69] T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

- [70] N Montés and J Tomero, “Lane changing using s-series clothoidal approximation and dual-rate based on bezier points to controlling vehicle.,” *WSEAS Transactions on Circuits and Systems*, vol. 3, no. 10, pp. 2285–2290, 2004.
- [71] J. Chen, P. Zhao, T. Mei, and H. Liang, “Lane change path planning based on piecewise bezier curve for autonomous vehicle,” in *Vehicular Electronics and Safety (ICVES), 2013 IEEE International Conference on*, 2013, pp. 17–22.
- [72] D Korzeniowski and G Ślaski, “Method of planning a reference trajectory of a single lane change manoeuver with bezier curve,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 148, 2016, p. 012 012.
- [73] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, “Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions,” *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.
- [74] T. Hastie, R. Tibshirani, and J. Friedman, “Overview of supervised learning,” in *The Elements of Statistical Learning*, Springer, 2009, pp. 9–41.
- [75] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [76] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1. MIT Press Cambridge, 1998, vol. 1.
- [77] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004, p. 1.
- [78] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning.,” in *AAAI*, Chicago, IL, vol. 8, 2008, pp. 1433–1438.
- [79] —, “Human behavior modeling with maximum entropy inverse optimal control.,” in *AAAI Spring Symposium: Human Behavior Modeling*, 2009, p. 92.
- [80] S. Levine and V. Koltun, “Continuous inverse optimal control with locally optimal examples,” *arXiv preprint arXiv:1206.4617*, 2012.
- [81] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, “Activity forecasting,” in *European Conference on Computer Vision*, Florence, Italy, 2012, pp. 201–214.
- [82] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.

- [83] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, New York, NY, 2016, pp. 49–58.
- [84] E. Velenis, E. Frazzoli, and P. Tsiotras, "Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2-4, pp. 217–241, 2010.
- [85] I. Chakraborty, P. Tsiotras, and J. Lu, "Vehicle posture control through aggressive maneuvering for mitigation of t-bone collisions," in *50th IEEE Conference on Decision and Control and European Control Conference*, Orlando, FL, 2011, pp. 3264–3269.
- [86] A. R. Hauber, "The social psychology of driving behaviour and the traffic environment: Research on aggressive behaviour in traffic," *Applied Psychology*, vol. 29, no. 4, pp. 461–474, 1980.
- [87] L. Mizell, M. Joint, and D. Connell, "Aggressive driving: Three studies," *AAA Foundation for Traffic Safety*, pp. 1–13, 1997.
- [88] D. Shinar, "Aggressive driving: The contribution of the drivers and the situation," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 1, no. 2, pp. 137–160, 1998.
- [89] L. Tasca, *A review of the literature on aggressive driving research*. Ontario, Canada: Ontario Advisory Group on Safe Driving Secretariat, Road User Safety Branch, Ontario Ministry of Transportation, 2000.
- [90] M. Abou-Zeid, I. Kaysi, and H. Al-Naghi, "Measuring aggressive driving behavior using a driving simulator: An exploratory study," in *Third International Conference on Road Safety and Simulation*, Indianapolis, 2011.
- [91] J. Han, Q. Song, and Y. He, *Adaptive Unscented Kalman filter and its applications in nonlinear control*. INTECH Open Access Publisher, 2009.
- [92] C. G. Hilborn and D. G. Lainiotis, "Optimal estimation in the presence of unknown parameters," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 1, pp. 38–43, 1969.
- [93] D. Alspach, "A parallel filtering algorithm for linear systems with unknown time varying noise statistics," *IEEE Transactions on Automatic Control*, vol. 19, no. 5, pp. 552–556, 1974.
- [94] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Courier Corporation, 2007.

- [95] K. Myers and B Tapley, "Adaptive sequential estimation with unknown noise statistics," *IEEE Transactions on Automatic Control*, vol. 21, no. 4, pp. 520–523, 1976.
- [96] G. Marafioti, S. Tebbani, D. Beauvois, G. Becerra-Celis, A. Isambert, and M. Hovd, "Unscented Kalman filter state and parameter estimation in a photobioreactor for microalgae production," in *International Symposium on Advanced Control of Chemical Processes*, Istanbul, Turkey, 2009.
- [97] D. Simon and T. L. Chia, "Kalman filtering with state equality constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 1, pp. 128–136, 2002.
- [98] N. Gupta and R. Hauser, "Kalman filtering with equality and inequality state constraints," *arXiv preprint arXiv:0709.2791*, 2007.
- [99] D. Simon, "Kalman filtering with state constraints: A survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.
- [100] W Wen and H. F. Durrant-Whyte, "Model-based multi-sensor data fusion," in *IEEE International Conference on Robotics and Automation*, Nice, 1992, pp. 1720–1726.
- [101] L.-S. Wang, Y.-T. Chiang, and F.-R. Chang, "Filtering method for nonlinear systems with constraints," *IEEE Proceedings-Control Theory and Applications*, vol. 149, no. 6, pp. 525–531, 2002.
- [102] S. Ko and R. R. Bitmead, "State estimation for linear systems with state equality constraints," *Automatica*, vol. 43, no. 8, pp. 1363–1368, 2007.
- [103] M. Tahk and J. L. Speyer, "Target tracking problems subject to kinematic constraints," *IEEE Transactions on Automatic Control*, vol. 35, no. 3, pp. 324–326, 1990.
- [104] C. Yang and E. Blasch, "Kalman filtering with nonlinear state constraints," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 1, pp. 70–84, 2009.
- [105] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decréton, "A smoothly constrained Kalman filter," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1171–1177, 1997.
- [106] H. Michalska and D. Q. Mayne, "Moving horizon observers and observer-based control," *IEEE Transactions on Automatic Control*, vol. 40, no. 6, pp. 995–1006, 1995.

- [107] B. O. S. Teixeira, L. A. Tôrres, L. A. Aguirre, D. S. Bernstein, *et al.*, “Unscented filtering for interval-constrained nonlinear systems,” in *Proceedings of the IEEE Conference on Decision and Control*, Canxun, Mexico, 2008, pp. 5116–5121.
- [108] D. Crisan and A. Doucet, “A survey of convergence results on particle filtering methods for practitioners,” *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, 2002.
- [109] S. Boyd and L. Vandenberghe, *Convex Optimization*. England: Cambridge University Press, 2004.
- [110] E. Donges, “A two-level model of driver steering behavior,” *Human Factors*, vol. 20, no. 6, pp. 691–707, 1978.
- [111] M. F. Land and D. N. Lee, “Where do we look when we steer,” *Nature*, vol. 369, no. 6483, pp. 742–744, 1994.
- [112] M. F. Land, “The visual control of steering,” *Vision and Action*, pp. 163–180, 1998.
- [113] H. Neumann and B. Deml, “The two-point visual control model of steering – new empirical evidence,” in *Digital Human Modeling*, Springer, 2011, pp. 493–502.
- [114] G. Markkula, O. Benderius, and M. Wahde, “Comparing and validating models of driver steering behaviour in collision avoidance and vehicle stabilisation,” *Vehicle System Dynamics*, vol. 52, no. 12, pp. 1658–1680, 2014.
- [115] L. Saleh, P. Chevrel, F. Mars, J.-F. Lafay, F. Claveau, *et al.*, “Human-like cybernetic driver model for lane keeping,” in *Proceedings of the 18th World Congress of the International Federation of Automatic Control*, Milano, Italy, 2011, pp. 4368–4373.
- [116] F. Mars, L. Saleh, P. Chevrel, F. Claveau, and J.-F. Lafay, “Modeling the visual and motor control of steering with an eye to shared-control automation,” in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 55, Las Vegas, 2011, pp. 1422–1426.
- [117] S. Noth, I. Rañó, and G. Schöner, “Investigating human lane keeping through a simulated driver,” in *Proceedings of the Driver Simulation Conference*, Paris, France, 2012.
- [118] I. Rano, H. Edelbrunner, and G. Schöner, “Naturalistic lane-keeping based on human driver data,” in *Intelligent Vehicles Symposium (IV)*, Gold Coast, Queensland, 2013, pp. 340–345.

- [119] C. Sentouh, B. Soualmi, J. C. Popieul, and S. Debernard, "Cooperative steering assist control system," in *IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, 2013, pp. 941–946.
- [120] D. T. McRuer, "Human pilot dynamics in compensatory systems," DTIC Document, Wright Air Development Center, Air Research Development Command, United States Air Force, Wright/Patterson Air Force Base, Dayton, OH, Tech. Rep., 1965.
- [121] S. P. Drake, "Converting GPS coordinates $[\phi, \lambda, h]$ to navigation coordinates (ENU)," Tech. Rep. DSTO-TN-0432, 2002.
- [122] H. Zhao and H. Chen, "Estimation of vehicle yaw rate and side slip angle using moving horizon strategy," in *The Sixth World Congress on Intelligent Control and Automation*, vol. 1, Dalian, 2006, pp. 1828–1832.
- [123] VERSION 4.5, This manual describes the CarSim Educational software., Mechanical Simulation Corporation, 709 W. Huron, Ann Arbor, MI 48103, Jan. 2000.
- [124] M. Saha, B. Goswami, and R. Ghosh, "Two novel costs for determining the tuning parameters of the Kalman filter," *arXiv preprint arXiv:1110.3895*, 2011.
- [125] M Ananthasayanam, "Kalman filter design by tuning its statistics or gains?" In *International Conference on Data Assimilation*, Mumbai, India, 2011.
- [126] D. Simon, *Optimal State Estimation: Kalman, H_∞ , and Nonlinear Approaches*. John Wiley & Sons, 2006.
- [127] Y. L. Murphey, R. Milton, and L. Kiliaris, "Driver's style classification using jerk analysis," in *IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, 2009, pp. 23–28.
- [128] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.
- [129] S. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 617–643, 1992.
- [130] H. Damveld and R Happee, "Identifying driver behaviour in steering: Effects of preview distance," in *Proceedings of the Measuring Behavior*, Utrecht, The Netherlands, 2012, pp. 44–46.

- [131] Y. Hassan and T. Sayed, "Effect of driver and road characteristics on required preview sight distance," *Canadian Journal of Civil Engineering*, vol. 29, no. 2, pp. 276–288, 2002.
- [132] A Grossmann, R. Kronland-Martinet, and J Morlet, "Reading and understanding continuous wavelet transforms," in *Wavelets*, Springer Berlin Heidelberg, 1989, pp. 2–20.
- [133] B. Jawerth and W. Sweldens, "An overview of wavelet based multiresolution analyses," *SIAM Review*, vol. 36, no. 3, pp. 377–412, 1994.
- [134] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bulletin of the American Meteorological Society*, vol. 79, no. 1, pp. 61–78, 1998.
- [135] Y.-U. Zhou and J.-Q. Cheng, "Wavelet transformation and its applications," *Acta Physica Sinica*, vol. 37, pp. 24–32, 2008.
- [136] K. Lundahl, J. Åslund, and L. Nielsen, "Investigating vehicle model detail for close to limit maneuvers aiming at optimal control," in *22nd International Symposium of Dynamic on Vehicle*, Manchester Metropolitan University, Manchester, UK, 2011.
- [137] M. S. Burhaumudin, P. M. Samin, H. Jamaluddin, R. A. Rahman, S. Sulaiman, *et al.*, "Integration of magic formula tire model with vehicle handling model," *International Journal of Research in Engineering and Technology*, vol. 1, no. 3, pp. 139–145, 2012.
- [138] J. D. Setiawan, M. Safarudin, and A. Singh, "Modeling, simulation and validation of 14 dof full vehicle model," in *International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*, 2009, pp. 1–6.
- [139] F Hunaini, I Robandi, and N Sutantra, "The optimal steering control system using imperialist competitive algorithm on vehicles with steer-by-wire system," *Iranian Journal of Electrical & Electronic Engineering*, vol. 11, no. 1, p. 25, 2015.
- [140] R. N. Jazar, *Vehicle Dynamics: Theory and Application*. Springer Science & Business Media, 2013.
- [141] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," SAE Technical Paper, Tech. Rep., 1987.
- [142] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *International Conference*

- on Robotics and Automation (ICRA)*, IEEE, Stockholm, Sweden, 2016, pp. 1433–1440.
- [143] N. J. Higham, “Computing a nearest symmetric positive semidefinite matrix,” *Linear Algebra and its Applications*, vol. 103, pp. 103–118, 1988.
 - [144] J. Qi, K. Sun, J. Wang, and H. Liu, “Dynamic state estimation for multi-machine power system by unscented kalman filter with enhanced numerical stability,” *arXiv preprint arXiv:1509.07394*, 2015.
 - [145] L. Perea, J. How, L. Breger, and P. Elosegui, “Nonlinearity in sensor fusion: Divergence issues in EKF, modified truncated SOF, and UKF,” in *ALAA Guidance, Navigation and Control Conference and Exhibit*, vol. 6514, Hilton Head, South Carolina, 2007.
 - [146] M. Ardel, P. Waldmann, F. Himm, and N. Kaempchen, “Strategic decision-making process in advanced driver assistance systems,” *IFAC Proceedings Volumes*, vol. 43, no. 7, pp. 566–571, 2010.
 - [147] R. Zheng, C. Liu, and Q. Guo, “A decision-making method for autonomous vehicles based on simulation and reinforcement learning,” in *International Conference on Machine Learning and Cybernetics*, vol. 1, Tianjian, China, 2013, pp. 362–369.
 - [148] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, “Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems,” in *IEEE 55th Conference on Decision and Control*, Las Vegas, NV, 2016, pp. 727–733.
 - [149] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky, “A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development,” in *American Control Conference (ACC), 2016*, Boston, MA, 2016, pp. 1705–1710.
 - [150] R. Bellman, “A Markovian decision process,” *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.
 - [151] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer Science & Business Media, 2013, vol. 31.
 - [152] A. Defazio and T. Graepel, “A comparison of learning algorithms on the arcade learning environment,” *arXiv preprint arXiv:1410.8620*, 2014.

- [153] L. Baird *et al.*, “Residual algorithms: Reinforcement learning with function approximation,” in *Proceedings of the 12th International Conference on Machine Learning*, Miami, Florida, 1995, pp. 30–37.
- [154] S.-C. Wang, “Artificial neural network,” in *Interdisciplinary Computing in Java Programming*, Springer, 2003, pp. 81–100.
- [155] J. M. Bernardo and A. F. Smith, *Bayesian Theory*. John Wiley & Sons Canada, 2001.
- [156] J. Q. Shi and T. Choi, *Gaussian Process Regression Analysis for Functional Data*. CRC Press, 2011.
- [157] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
- [158] S. Levine, Z. Popovic, and V. Koltun, “Nonlinear inverse reinforcement learning with Gaussian processes,” in *Advances in Neural Information Processing Systems*, 2011, pp. 19–27.
- [159] C. J.C. H. Watkins, “Learning from delayed rewards,” PhD thesis, King’s College, Cambridge, 1989.
- [160] C. J.C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [161] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *AAAI*, 2016, pp. 2094–2100.
- [162] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [163] H. R. Berenji, “Fuzzy Q-learning: A new approach for fuzzy dynamic programming,” in *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, Orlando, FL, 1994, pp. 486–491.
- [164] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *Proceedings of the 11th International Conference on Machine Learning*, vol. 157, New Brunswick, NJ, 1994, pp. 157–163.
- [165] J. Hu and M. P. Wellman, “Nash Q-learning for general-sum stochastic games,” *Journal of Machine Learning Research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

- [166] A. Greenwald, K. Hall, and R. Serrano, “Correlated Q-learning,” in *Proceedings of the 12th International Conference on Machine Learning*, vol. 3, Washington, DC, 2003, pp. 242–249.
- [167] M. L. Littman, “Friend-or-foe Q-learning in general-sum games,” in *Proceedings of the 18th International Conference on Machine Learning*, vol. 1, Williamstown, MA, 2001, pp. 322–328.
- [168] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA, 2000, pp. 663–670.
- [169] E. T. Jaynes, “Information theory and statistical mechanics,” *Physical Review*, vol. 106, no. 4, pp. 620–630, 1957.
- [170] M. Dudík and R. E. Schapire, “Maximum entropy distribution estimation with generalized regularization,” in *International Conference on Computational Learning Theory*, San Diego, CA, 2006, pp. 123–138.
- [171] R. Hecht-Nielsen *et al.*, “Theory of the backpropagation neural network,” *Neural Networks*, vol. 1, no. Supplement-1, pp. 445–448, 1988.
- [172] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [173] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [174] J. Audiffren, M. Valko, A. Lazaric, and M. Ghavamzadeh, “Maximum entropy semi-supervised inverse reinforcement learning,” in *International Joint Conferences on Artificial Intelligence*, Buenos Aires, Argentina, 2015, pp. 3315–3321.
- [175] L. Jaillet, J. Cortés, and T. Siméon, “Sampling-based path planning on configuration-space costmaps,” *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.
- [176] M. Garcia, A. Viguria, and A. Ollero, “Dynamic graph-search algorithm for global path planning in presence of hazardous weather,” *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 285–295, 2013.
- [177] H. Erişkin and A. Yücesan, “Bézier curve with a minimal jerk energy,” *Mathematical Sciences and Applications E-Notes*, vol. 4, no. 2, pp. 139–148,
- [178] H. Deddi, H. Everett, and S. Lazard, “Interpolation with curvature constraints,” PhD thesis, INRIA, 2000.

- [179] J. B. Hoagg, W. M. Haddad, and D. S. Bernstein, *Linear-Quadratic Control: Theory and Methods for Continuous-Time Systems*. Princeton University Press on Preparation.
- [180] B. A. Francis, “The linear multivariable regulator problem,” *SIAM Journal on Control and Optimization*, vol. 15, no. 3, pp. 486–505, 1977.
- [181] E. Velenis, P. Tsiotras, and J. Lu, “Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn,” in *European Control Conference*, Kos, Greece, 2007, pp. 1233–1240.
- [182] —, “Trail-braking driver input parameterization for general corner geometry,” SAE Technical Paper, Tech. Rep., 2008.
- [183] —, “Optimality properties and driver input parameterization for trail-braking cornering,” *European Journal of Control*, vol. 14, no. 4, pp. 308–320, 2008.
- [184] D. Tavernini, M. Massaro, E. Velenis, D. I. Katzourakis, and R. Lot, “Minimum time cornering: The effect of road surface and car transmission layout,” *Vehicle System Dynamics*, vol. 51, no. 10, pp. 1533–1547, 2013.
- [185] R. Y. Hindiyeh and J. C. Gerdes, “Equilibrium analysis of drifting vehicles for control design,” in *ASME 2009 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, Hollywood, CA, 2009, pp. 181–188.
- [186] J. Yi and E. H. Tseng, “Nonlinear stability analysis of vehicle lateral motion with a hybrid physical/dynamic tire/road friction model,” in *ASME Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, Hollywood, CA, 2009, pp. 509–516.
- [187] K. Zhou, J. C. Doyle, K. Glover, *et al.*, *Robust and Optimal Control*. Prentice hall New Jersey, 1996, vol. 40.
- [188] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, “Differential flatness and defect: An overview,” in *Workshop on Geometry in Nonlinear Control*, Banach Center Publications, Warsaw, 1993.
- [189] —, “Flatness and defect of non-linear systems: Introductory theory and examples,” *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.
- [190] M Fliess, J Lévine, P. Martin, F Ollivier, and P Rouchon, “Controlling nonlinear systems by flatness,” in *Systems and Control in the Twenty-first Century*, Springer, 1997, pp. 137–154.

- [191] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, “A Lie-Backlund approach to equivalence and flatness of nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 44, no. 5, pp. 922–937, 1999.
- [192] E. Velenis, “Analysis and control of high-speed wheeled vehicles,” PhD thesis, Georgia Institute of Technology, 2006.
- [193] M. A. Patterson and A. V. Rao, “GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming,” *ACM Transactions on Mathematical Software*, vol. 41, no. 1, p. 1, 2014.
- [194] A. Coates, P. Abbeel, and A. Y. Ng, “Learning for control from multiple demonstrations,” in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 144–151.
- [195] Y. Wang, S. Shi, and L. Li, “Flatness-based vehicle coupled control for steering stability and path tracking,” in *Proceedings of SAE-China Congress 2015: Selected Papers*, Springer, 2016, pp. 49–60.
- [196] S. Fuchshumer, K. Schlacher, and T. Rittenschober, “Nonlinear vehicle dynamics control—a flatness based approach,” in *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Spain, 2005, pp. 6492–6497.
- [197] H. Souilem and N. Derbel, “Vehicle control by flatness,” *Proceedings Engineering & Technology-Vol*, vol. 4, pp. 37–42, 2013.
- [198] J. Villagra, B. d’Andrea Novel, H. Mounier, and M. Pengov, “Flatness-based vehicle steering control strategy with sdre feedback gains tuned via a sensitivity approach,” *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 554–565, 2007.
- [199] G. M. Scarpello and D. Ritelli, “A historical outline of the theorem of implicit functions,” *Divulgaciones Matemáticas*, vol. 10, no. 2, pp. 171–180, 2002.
- [200] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.
- [201] P. Bourke. (1988). Points, lines, and planes.
- [202] E. Velenis and P. Tsiotras, “Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation,” *Journal of Optimization Theory and Applications*, vol. 138, no. 2, pp. 275–296, 2008.

- [203] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velez, P. Tsiotras, and J. M. Rehg, "Aurally: An open platform for aggressive autonomous driving," *arXiv preprint arXiv:1806.00678*, 2018.
- [204] C. You, J. Lu, and P. Tsiotras, "Nonlinear driver parameter estimation and driver steering behavior analysis for ADAS using field test data," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 686–699, 2017.

VITA

Changxi You received his B.S. and M.S. degrees from the Department of Automotive Engineering, Tsinghua University of China, and M.S. degree from the Department of Automotive Engineering, RWTH-Aachen University of Germany. He joined the DCSL lab since August 2014 and became a Ph.D. student under the supervision of Prof. Panagiotis Tsiotras at the School of Aerospace Engineering, Georgia Institute of Technology. His research interests are in system identification, aggressive driving, path planning and control of (semi)autonomous vehicles.