# TRAJECTORY PLANNING AND CONTROL FOR AN AUTONOMOUS RACE VEHICLE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Nitin R. Kapania
February 2016

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(J. Christian Gerdes) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Allison Okamura)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Mykel Kochenderfer)

Approved for the University Committee on Graduate Studies

*To be written.*

# Abstract

Autonomous vehicle technologies offer potential to eliminate the number of traffic accidents that occur every year, not only saving numerous lives but mitigating the costly economic and social impact of automobile related accidents. The premise behind this dissertation is that autonomous cars of the near future can only achieve this ambitious goal by obtaining the capability to successfully maneuver in friction-limited situations. With automobile racing as an inspiration, this dissertation presents and experimentally validates three vital components for driving at the limits of tire friction. The first contribution is a feedback-feedforward steering algorithm that enables an autonomous vehicle to accurately follow a specified trajectory at the friction limits while preserving robust stability margins. The second contribution is a trajectory generation algorithm that leverages the computational speed of convex optimization to rapidly generate both a longitudinal speed profile and lateral curvature profile for the autonomous vehicle to follow. While the algorithm is applicable to a wide variety of driving objectives, the work presented is for the specific case of vehicle racing, and generating minimum-time profiles is therefore the chosen application. The final contribution is a set of iterative learning control and search algorithms that enable autonomous vehicles to drive more effectively by learning from previous driving maneuvers. These contributions enable an autonomous Audi TTS test vehicle to drive around a race circuit at a level of performance comparable to a professional human driver. The dissertation concludes with a discussion of how the algorithms presented can be translated into automotive safety systems in the near future.

# Acknowledgment

To be written.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Advancements in sensing, perception, and low-cost embedded computing have resulted in the rapid growth of autonomous vehicle technology over the last two decades. Once the subject of sci-fi imagination in world exhibitions and popular journalism, semi-autonomous driving features such as emergency braking, autonomous lane guidance and adaptive cruise control are now readily available. Furthermore, many automotive manufacturers and technology firms are developing automated vehicles requiring little or no human interaction [11][14][35][37][43][61].

The potential benefits of an automated vehicle ecosystem are significant. A comprehensive 2015 study by the consulting firm McKinsey and Company [2] estimates that widespread adoption of autonomous vehicle technology would reduce automobile accidents by over 90%, preventing thousands of fatalities, hundreds of thousands of hospitalizations, and many billions of dollars in property damage annually.

While a large portion of autonomous vehicle research and development is focused on handling routine driving situations, achieving the safety benefits of autonomous vehicles also requires a **focus on automated driving at the limits of tire friction**. The need for an automated vehicle to fully utilize its capability can arise when avoiding a collision with human-operated vehicles. This is crucial from an automotive safety standpoint as human error accounts for over 90% of automobile accidents [68], and there will likely be a significant period of time where autonomous vehicles must interact with human-operated vehicles [2]. Furthermore, successful handling at

the friction limits will be required where environmental factors are involved, such as unpredicted natural obstructions and poor tire friction caused by inclement weather (e.g. ice, rain). The potential for technology to assist in friction-limited situations has already been demonstrated by electronic stability control (ESC) systems, which reduced single-vehicle accidents by 36% in 2007 [13] and are now standard on all commercial vehicles.

## 1.1 Driving at the Handling Limits

Each of the four tires on an automobile contacts the road surface over a contact patch, an area roughly the size of a human hand (Fig. 1.1(b)). As shown in Fig. 1.1(a), these contact patches generate the friction forces between the tire and road that are necessary for both vehicle longitudinal acceleration (braking and acceleration) as well as lateral acceleration (turning). Because the available friction between the tire and road is limited, each of the four tires is limited in the turning, braking, and accelerating forces they can produce. This relationship is given for each tire by the commonly known "friction circle" equation:

$$\mu F_z \geq \sqrt{F_x^2 + F_y^2} \tag{1.1}$$

where $\mu$ is the friction coefficient between the tire and the road, $F_z$ is the normal force acting on the tire, and $F_x$ and $F_y$ are the lateral and longitudinal forces, respectively (Fig. 1.1(c)). One key insight from (1.1) is that the cornering and braking ability of the car is heavily determined by the amount of friction. On a dry, paved asphalt surface, values of $\mu$ are typically equal to 1.0. However, on wet or rainy asphalt, $\mu$ can decrease to 0.7, and in snow or ice, the value of $\mu$ can be as low as 0.2 [54]. Another insight from (1.1) is the coupled relationship between vehicle lateral and longitudinal forces. If the vehicle is braking (or accelerating) heavily, the value of $F_x^2$ will be large and there will be less friction force available for turning.

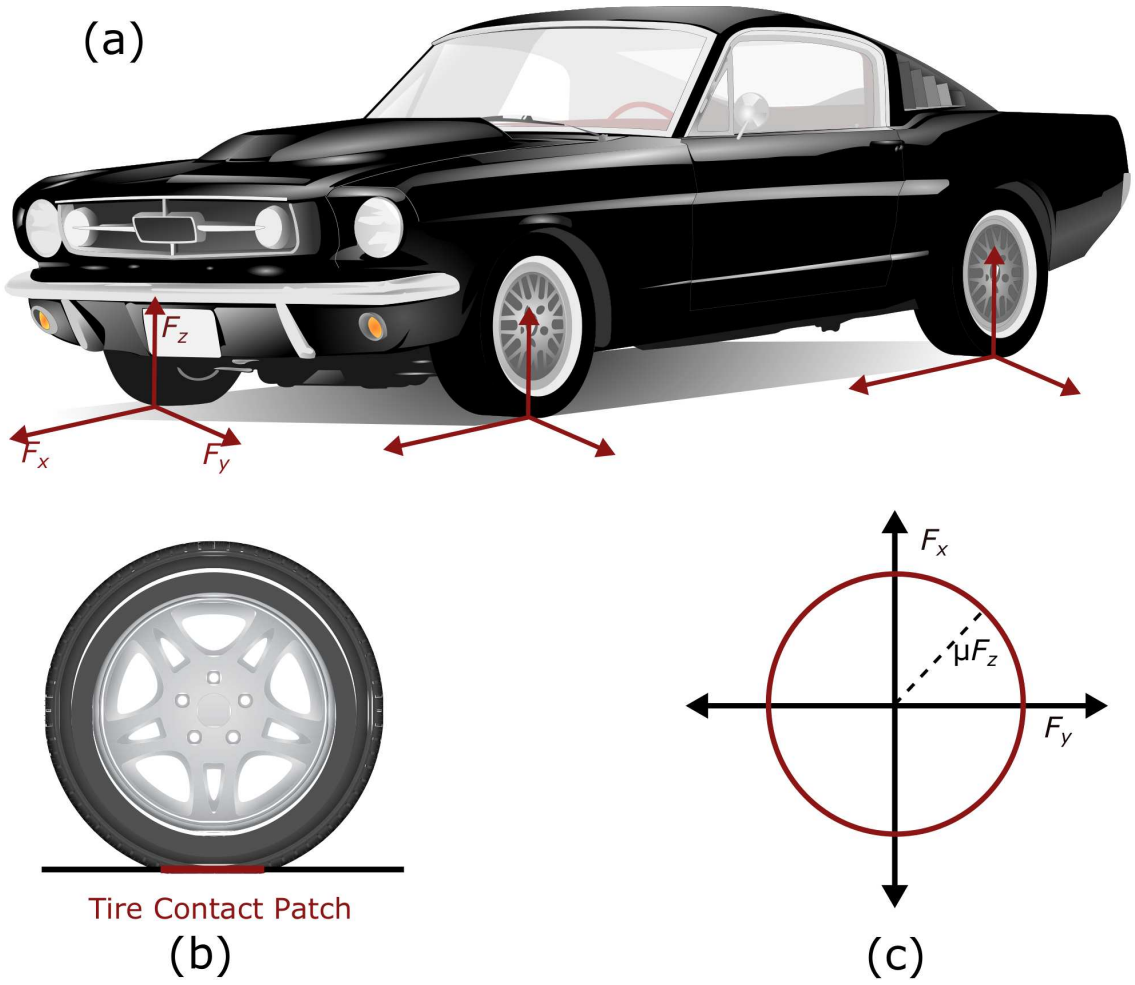


Figure 1.1: (a) Friction forces $F_x$ and $F_y$ generated in the contact patch allow for lateral and longitudinal vehicle acceleration. (b) Side view of tire contact patch. (c) Graph showing combined lateral and longitudinal force capability for a tire given the normal load and friction coefficient $\mu$.

### 1.1.1 Exceeding the Friction Limits: Understeer and Oversteer

In normal driving situations, the forces required for turning, braking, and accelerating will be much smaller than the available friction force. However, in rainy or icy conditions, accidents frequently occur when the driver enters a turn too fast or when the driver attempts to turn too quickly while already applying the brakes. In these situations, the friction limit of one or more of the tires are reached and the vehicle loses control.

There are two ways for an automobile to lose control. If the front two tires reach the friction capability before the rear tires, the vehicle will *understeer*, as illustrated in Fig. 1.2(a). In this condition, the vehicle is unable to rotate enough to complete a

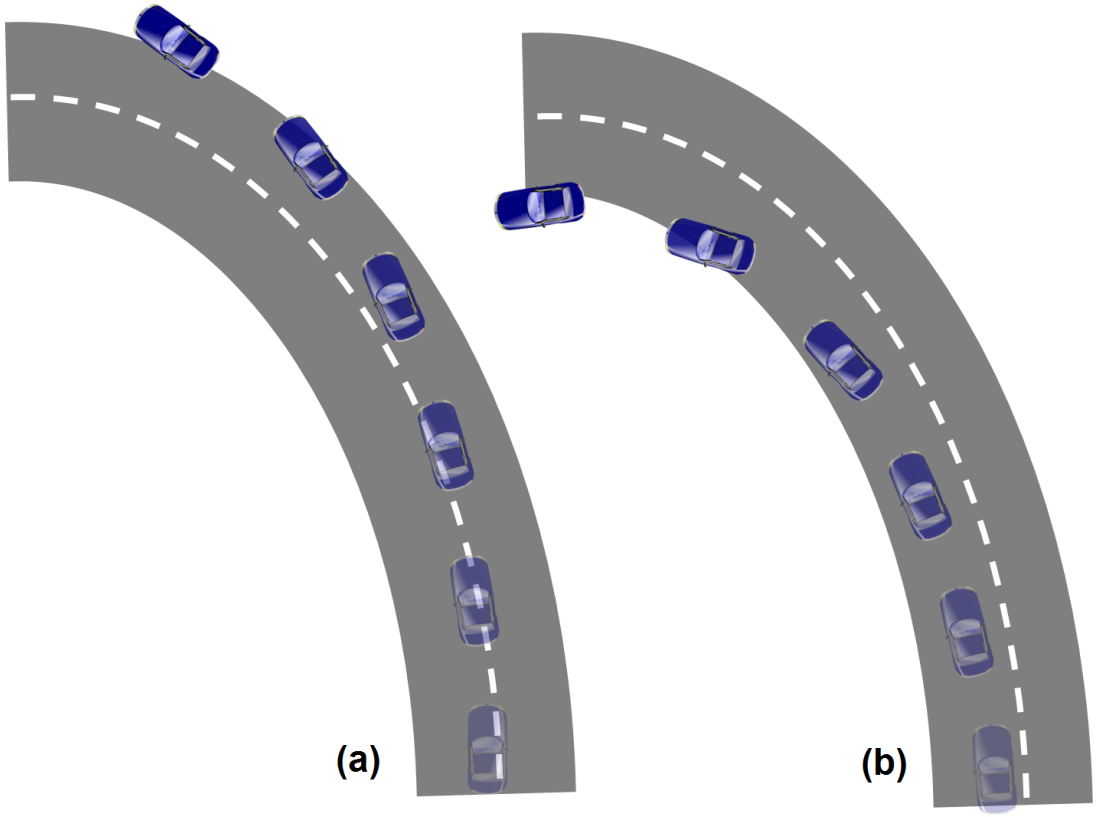


Figure 1.2: (a) Vehicle understeering at the limits of handling. (b) Vehicle oversteering at the limits of handling.

turn and typically skids off the road. For the converse scenario where the rear tires saturate first, the vehicle enters an *oversteer* condition, as illustrated in Fig. 1.2(b). In this situation, the vehicle over-rotates and goes into a spin. Both conditions are extremely dangerous as they can easily result in the vehicle colliding with a fixed obstacle such as a tree or with another vehicle. In large vehicles such as SUVs, understeer and oversteer can also result in vehicle rollover.

## 1.2 Race Car Driving as Inspiration for Autonomous Safety Systems

Automotive engineers today face the challenge of designing autonomous safety systems that can utilize the full capabilities of the vehicle's tires in emergency scenarios to avoid accidents and significant understeer or oversteer. While this is a difficult task, professional race driving provides a source of inspiration for designing autonomous safety systems.

In order to complete a race in minimum time, race car drivers routinely use 100% of the available friction between their vehicle's tires and the road. Professional drivers are extremely skilled at coordinating their brake, throttle, and steering inputs to maximize the speed of the vehicle through all corners of a race course while keeping the vehicle tires within the friction limits. Furthermore, they must achieve this while avoiding collisions with other competing drivers who are also driving extremely aggressively. Finally, race car drivers often exceed the friction limits temporarily while seeking the fastest lap time, and have the ability to re-stabilize the vehicle from an understeer or oversteer scenario.

**The primary focus of this dissertation is therefore to develop a set of control algorithms that allow an autonomous vehicle to drive at the handling limits with the same capability as a professional race driver.** In particular, these algorithms focus on autonomously completing three primary tasks that race car drivers demonstrate with proficiency:

1. **Vehicle Steering at the Limits of Handling**. A vital task of racing is steering an automobile through a race course at the handling limits. Given the high vehicle speeds and lateral accelerations required for racing, the risk of vehicle oversteer or understeer is very high. Good race car drivers have the ability to quickly and aggressively operate the steering wheel to complete a turn while maintaining vehicle stability.

2. **Finding a Time-Optimal "Racing Trajectory"**. Given a race track and race vehicle, another fundamental task of racing is determining the fastest trajectory, or "racing line" for the vehicle to follow. Race car drivers are skilled at driving though a race track along a path that enables them to take larger radius turns and accelerate aggressively on straight paths, increasing the permissible speed of the vehicle given tire friction constraints.

3. **Lap-to-Lap Learning**. Finally, most races require the driver to complete many laps around the same race track. Given this repetition, race car drivers have the opportunity to improve their lap times by slightly modifying their driving behavior on each lap to account for observations made during prior laps. The ability to learn from prior laps of driving also enables race car drivers to account for changing conditions (e.g. increasing temperatures, tire wear) over the course of a race.

While these tasks may seem specific to the niche field of race car driving, algorithms that enable a vehicle to autonomously drive like a race professional have enormous potential for vehicle safety systems. Algorithms that allow for steering at the limits of handling can be vital in piloting a vehicle through a sudden stretch of icy road during the winter. With a small modification to the objective function, an algorithm that maximizes the turning radius on a race course can be used to maximize the distance between a vehicle and oncoming traffic. Learning algorithms that allow more precise driving over a fixed race course can be used to assist drivers with their daily commute. Potential applications of the developed racing algorithms will be discussed further in the conclusion of this dissertation.

## 1.3 State of the Art

There has been significant prior work focused on autonomous steering control at the friction limits, time-optimal trajectory planning, and iteration-based learning control. This section provides a brief overview of prior work that is relevant to the research contributions presented in this dissertation.

### 1.3.1 Autonomous Race Vehicles

Given the highly visible marketing opportunity provided by racing, several automotive companies have made notable attempts at racing-inspired automated driving. In 2008, BMW introduced the "Track Trainer", which records race data collected from a professional driver. To "replay" the professional's driving autonomously, the vehicle tracks the pre-recorded speed and racing line with a proportional-derivative controller for throttle and brake and a dynamic programming algorithm for steering [81]. Using pre-recorded inputs allows the controller to naively account for nonlinear vehicle dynamics at the handling limits, although this approach limits the flexibility of the controller to respond to unpredicted events.

A second German luxury brand, Audi AG, also launched a collaborative research effort with Stanford University in 2008. The collaboration resulted in the development of "Shelley", an autonomous Audi TTS. Doctoral work by Stanford students Theodosis [73] and Kritayakirana [45] provided initial forays into racing line generation and trajectory-following algorithms. Notable early accomplishments include autonomous driving at speeds of 190 mph at the Salt Flats in Utah and an autonomous drive up the Pikes Peak International Hill Climb in 2009 [27][78]. More recently, Audi has incorporated results from the collaboration to build a second generation race vehicle, "Bobby" (Fig. 1.3), an autonomous RS7 which debuted at Germany's Hockenheimring [11].


Figure 1.3: "Bobby", Audi's autonomous RS7.

### 1.3.2 Automated Steering at the Limits of Handling

In the 1990's and early 2000's, a primary focus of autonomous driving research was designing control systems to follow a desired path *below* the limits of handling. Initial designs typically centered around linear feedback-feedforward controllers, using linear models of the vehicle dynamics to design the steering control laws [67]. An important development at this time was the idea of *lookahead* steering feedback, where the objective is to minimize the lateral tracking error at a certain point in front of the vehicle [31][62].

Given the success of linear controller designs for automated steering, early attempts at driving at the handling limits also made the assumption of linear vehicle dynamics. While the dynamics of an automobile become nonlinear at the handling limits due to tire saturation, assuming linear dynamics in the controller design resulted in respectable results in several studies [55][66][75]. To improve upon these results, more recent publications have proposed control systems that account for the nonlinear effect of tire saturation at the handling limits [19][46][83]. The most recent development has been the application of model-predictive control (MPC), which enables state-of-the-art steering controllers to track a path at the handling limits while trading off between competing objectives of obstacle avoidance and vehicle stabilization [8][21].

While there are a wide variety of published steering controllers with varying levels of complexity, there is no single experimentally validated controller that displays a well-understood combination of robust stability margins and low path tracking error at the limits of handling. Work by Rosseter [62] and Talvala [72] provides great analysis of the desirable stability properties of lookahead steering feedback, but no discussion of how path tracking behavior changes as the vehicle approaches the limits of handling. While stability of a control system is vital, a path-following controller with poor tracking performance has the risk of hitting an obstacle in an emergency avoidance maneuver. Work presented by Kritayakirana and Gerdes [46] was validated experimentally and has the desirable stability of lookahead steering, but exhibits relatively poor path tracking at the limits of handling and cannot be driven at accelerations above 8 $m/s^2$ without incurring significant yaw rate oscillations. Experimentally validated results using model-predictive control [8][21] demonstrate the ability to balance competing objectives of vehicle path tracking and stability, but must explicitly sacrifice one for the other at the handling limit. Furthermore, model-predictive controllers consist of complex optimization problems that make fundamental issues such as stability and tracking performance difficult to analyze mathematically.

### 1.3.3 Time-Optimal Trajectory Planning

The problem of calculating the minimum lap time trajectory for a given vehicle and race track has been studied over the last several decades in the control, optimization, and vehicle dynamics communities. The minimum lap time problem initially drew significant interest from professional racing teams trying to understand the effect of vehicle modifications on the optimal lap time. Casanova [9] therefore developed a method in 2000 (later refined by Kelly [42] in 2008) capable of simultaneously optimizing both the path and speed profile for a fully nonlinear vehicle model using nonlinear programming (NLP). The developed software was able to meet the needs of Formula One race teams by analyzing the effects of subtle changes in vehicle parameters, including tire thermodynamic properties and suspension designs.

More recently, the development of autonomous vehicle technology has led to research on optimal path planning algorithms that can be used for driverless cars. Theodosis and Gerdes published a nonlinear gradient descent approach for determining time-optimal racing lines [74], which has the rare distinction of being validated experimentally on an autonomous race vehicle.

However, a significant drawback of nonlinear programming solutions is high computational expense. Given the need for real-time trajectory planning in autonomous vehicles, there has been a recent interest in finding approximate methods that provide fast lap times with low computational expense. Published methods include formulating the minimum lap time problem into a model predictive control (MPC) problem [76] or solving a series of locally optimal optimization problems [24][80]. However, one potential drawback of the model predictive control approach is that an optimization problem must be reformulated at every time step, which can still be computationally expensive.

Experimental validation on an autonomous race vehicle has only been reported by Theodosis and Gerdes [74] and Gerdts et al. [24]. Furthermore, there has been little focus on developing a trajectory planning algorithm with a runtime close enough for real-time implementation on an experimental vehicle. While an autonomous vehicle can apply a closed-loop controller to follow a time-optimal vehicle trajectory computed offline, there are significant benefits to developing a fast trajectory generation algorithm that can approximate the globally optimal trajectory in real-time. If the algorithm runtime is small compared to the actual lap time, the algorithm can run as a real-time trajectory planner and find a fast racing line for the next several turns of the racing circuit. This would allow the trajectory planner to modify the desired path based on the motion of competing race vehicles and estimates of road friction, tire wear, engine/brake dynamics and other parameters learned over several laps of racing. Additionally, the fast trajectory algorithm can be used to provide a very good initial trajectory for a nonlinear optimization method.

### 1.3.4 Iteration-Based Learning

Developing algorithms that mimic a human's ability to adapt and learn over time has been a focus for researchers in a variety of fields. In the field of automated control, an interesting approach for adaptation is iterative learning control (ILC), based on the notion that the performance of a system that executes the *same task* multiple times can be improved by learning from previous executions [6]. Because iterative learning control works best when learning to follow the same reference trajectory under the same ambient conditions, the most common applications of ILC are in the field of automated manufacturing. Notable examples include CNC machining [44], industrial robotics [20][32], piezolectric stage positioning [34], motor control [51], and microdeposition [33]. However, the rise of automated systems outside factory environments has led to preliminary applications of ILC for ground and air robotics [10][60][71].

In the field of computer science, a technique widely used for training in automated systems is *reinforcement learning*. Reinforcement learning is similar to iterative learning control in that an automated system overcomes uncertainty in the world by gradually learning over multiple trials. However, iterative learning control algorithms typically assume the system is modeled by a discrete (often linear) dynamic system, with uncertainty in the form of an unknown but repeating disturbance. On the other hand, reinforcement learning algorithms act on systems modeled by Markov Decision Processes (MDPs), with the uncertainty typically in the form of unknown state transition probabilities and rewards. Furthermore, iterative learning algorithms attempt to gradually determine an input *control signal* to overcome the unknown disturbance and provide accurate tracking of a reference trajectory. Reinforcement learning algorithms are more general, and develop a *policy* that maps any state within the MDP to an optimal action.

Like recent ILC research, reinforcement learning has also been widely investigated for applications in ground and air robotics. In the field of UAV control, Ng et al. presented a reinforcement learning algorithm to learn a controller for autonomous inverted helicopter flight [57]. There have also been many publications in the area of robotic motion control. For example, in a modification of reinforcement learning

known as "apprenticeship learning" Lee et al. presented research where a robot was able to tie a knot after observing human-guided observations [64]. Finally, in the area of autonomous vehicles, Lauer presented a reinforcement learning approach to designing a steering controller for a 1:5 scale RC car [47].

In summary, iteration-based learning algorithms have a rich history of validation on manufacturing and robotic systems. Developing similar algorithms for an autonomous race vehicle could therefore yield significant benefits. Even with a well-designed trajectory planner and path-following controller, there will often be regions of the race track where transient vehicle dynamics and unmodeled disturbances result in poor tracking of the optimal trajectory. Furthermore, a major determinant of the optimal trajectory is the friction coefficient between the road and the tires. In reality, this is hard to know ahead of time beyond a reasonable estimate (e.g $0.95 \leq \mu \leq 1.0$). However, at the limits of handling, small differences in the amount of grip between the tires and road can result in significant lap time differences. Additionally, series of consecutive turns can be driven more aggressively, as there is less risk of an overall lap time increase if the vehicle understeers or oversteers slightly. On the other hand, turns before a long straight section must be driven more cautiously, as exceeding the friction limit can result in lower top speeds on the fastest part of the track, significantly increasing lap times. Human drivers understand this effect well, especially for front-heavy vehicles, and use the term "slow-in, fast-out" to describe their strategy on these more crucial turns. However, this strategy is hard to capture in a typical trajectory planning algorithm without a learning component.

## 1.4 Research Contributions and Outline

Section 1.3 provided a brief overview of the state of the art for the three primary tasks of trajectory planning, trajectory following and iteration-based learning. Opportunities for important further research in each task were articulated as well. This section outlines the primary contributions of this doctoral work for each of these racing-inspired research areas.

**Chapter 2: A Feedback-Feedforward Steering Controller for Accurate Path Tracking and Stability at the Limits of Handling**

Chapter 2 of this dissertation presents a feedback-feedforward steering controller that simultaneously maintains vehicle stability at the limits of handling while minimizing lateral path tracking deviatio  The design begins by considering the performance of a baseline controller with a lookahead feedback scheme and a feedforward algorithm based on a nonlinear vehicle handling diagram. While this initial design exhibits desirable stability properties at the limits of handling, the steady-state path deviation increases significantly at highway speeds. Results from both linear and nonlinear analyses indicate that lateral path tracking deviations are minimized when vehicle sideslip is held tangent to the desired path at all times. Analytical results show that directly incorporating this sideslip tangency condition into the steering feedback dramatically improves lateral path tracking, but at the expense of poor closed-loop stability margins. However, incorporating the desired sideslip behavior into the feedforward loop creates a robust steering controller capable of accurate path tracking and oversteer correction at the physical limits of tire friction. Experimental data collected from an Audi TTS test vehicle driving at the handling limits (up to 9.5 $m/s^2$) on a full length race circuit demonstrates the improved performance of the final controller design.

**Chapter 3: A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories**

Chapter 3 presents an iterative algorithm that divides the path generation task into two sequential subproblems that are significantly easier to solve than the fully nonlinear lap time optimization. Given an initial path through the race track, the algorithm runs a forward-backward integration scheme to determine the minimum-time longitudinal speed profile, subject to tire friction constraints. With this speed profile fixed, the algorithm updates the vehicle's path by solving a convex optimization problem that minimizes the curvature of the vehicle's driven path while staying within track boundaries and obeying affine, time-varying vehicle dynamics constraints. This two-step process is repeated iteratively until the predicted lap time no longer improves.

While providing no guarantees of convergence or a globally optimal solution, the approach performs very well when validated on the Thunderhill Raceway course in Willows, CA. The predicted lap time converges after four to five iterations, with each iteration over the full 4.5 km race course requiring only thirty seconds of computation time on a laptop computer. The resulting trajectory is experimentally driven at the race circuit with an autonomous Audi TTS test vehicle, and the resulting lap time and racing line is comparable to both a nonlinear gradient descent solution and a trajectory recorded from a professional racecar driver. The experimental results indicate that the proposed method is a viable option for online trajectory planning in the near future.

**Chapters 4 and 5: Iterative Learning Algorithms to Improve Autonomous Driving Performance**

This dissertation proposes two sets of learning algorithms that gradually refine the driving performance of the autonomous race car over time. Chapter 4 presents an iterative learning control (ILC) formulation to gradually determine the proper steering and throttle input for transient driving maneuvers along the race track. Racing is an ideal scenario for ILC because race cars drive the same sequence of turns while operating near the physical limits of tire-road friction. This creates a difficult to model, but repeatable, set of nonlinear vehicle dynamics and road conditions from lap to lap. Simulation results are used to design and test convergence of both a proportional-derivative (PD) and quadratically optimal (Q-ILC) iterative learning controller, and experimental results are presented at combined vehicle accelerations of up to 9 $m/s^2$.

Chapter 5 focuses on determining the best value of the friction coefficient $\mu$ for turn-by-turn trajectory planning on the track. Because the friction coefficient is directly linked to the peak accelerations of the speed profile, locally varying $\mu$ for each turn on the track is a way to tune the aggressiveness of the planned trajectory. Rather than directly encoding the "slow-in, fast-out" heuristic that human drivers typically employ, a learning approach is used to automatically determine the fastest strategy. A small but significant collection of data is gathered from the autonomous

vehicle driving different turns of the track with different values of $\mu$ assumed. From this data, an A* search algorithm is devised that searches through the data and finds the best value of $\mu$ for each portion of the track in order to globally minimize the resulting lap time. Key developments of this algorithm include designing an appropriate A* heuristic to minimize the needed computation time and designing the cost function to account for the physical difficulty of altering the vehicle's trajectory while understeering or oversteering.

# Chapter 2

# Feedforward-Feedback Steering Controller

A central control task in the operation of an autonomous vehicle is the ability to maneuver along a desired path, typically generated by a high-level path planner. As a result, a large research effort has been devoted to the subject of active steering control for autonomous or semi-autonomous vehicles. Steering systems based on feedback-feedforward (FB-FFW) control architectures have been a major focus of research. Early work by Shladover et al. [67] described a FB-FFW controller where the feedforward steering angle was determined from path curvature and longitudinal force inputs, and the feedback gains were selected from a frequency shaped linear quadratic regulator (FSLQR) designed to minimize path tracking error while maintaining good ride quality at different frequencies. Nagai et al. [56] also used LQR to design two feedback steering controllers for autonomous path following, with one controller using steer angle as the control input and the other using steering torque.

Another simple but effective approach to feedback-feedforward steering control is to design a controller with the objective of making the lateral tracking error zero at a certain "lookahead" point in front of the vehicle. Minimization of a lookahead objective was studied by Hingwe and Tomizuka [31]. A crucial result was the finding that internal yaw dynamics can be damped at all longitudinal velocities by making the lookahead point a quadratic function of the vehicle velocity. Rosseter [62] also

studied lookahead feedback systems extensively, and derived a similar controller by minimizing a quadratic potential function of the projected lookahead error.

While initial work was focused at driving well below the limits of tire friction, several authors obtained respectable results by applying similar methods for more aggresive maneuvers. Müller-Beßler applied a linear plant inversion method to determine feedforward commands for a Volkswagen Golf attempting an aggressive double lane change maneuver [55]. Additionally, Thommyppillai et al. [75] and Sharp et al. [66] used a linear preview controller to simulate path following of a virtual racecar driver. The tendency for linear controller designs to work reasonably well at the limits of handling was partially explained by Talvala et al. [72], who were able to find a Lyapunov function demonstrating stability of lookahead steering feedback even in the presence of significant front and rear tire saturation.

More recent work has focused on improving control performance by accounting for the nonlinear effect of tire saturation at the handling limits. Kritiyikarana and Gerdes [46] proposed a force-based FB-FFW controller with the feedforward steering force determined by eliminating the error dynamics about the vehicle's center of percussion. The effect of tire saturation was accounted for by inverting a nonlinear vehicle dynamics model to convert the desired steering force to a steer angle input. Most recently, the emerging trend of model-predictive control has resulted in steering controllers that attempt to track a path at the handling limits while also avoiding obstacles. In 2013, Carvalho et al. [8] demonstrated a model predictive controller (MPC) capable of steering an experimental passenger sedan around obstacles at high speeds on an icy road. The controller was based on a nonlinear Ackermann model that was iteratively linearized. A similar affine linearization MPC technique was employed by Funke [21] to demonstrate a student-built vehicle tracking an oval path at accelerations of 7 $m/s^2$ while avoiding rapidly emerging obstacles.

In summary, there are a wide variety of published steering controllers with varying levels of complexity. However, there is no single experimentally validated controller that displays a well-understood combination of robust stability margins and low path tracking error at the limits of handling. The stability properties of lookahead steering below and at the handling limits are demonstrated by [62] and [72], but with but no

discussion of how path tracking behavior changes as the vehicle approaches the limits of handling. A path-following controller with poor tracking performance at the handling limits has the risk of hitting an obstacle in an emergency avoidance maneuver. Work presented by Kritayakirana and Gerdes [46] was validated experimentally and has the desirable stability of lookahead steering, but exhibited poor path tracking at the limits of handling and could not be driven at accelerations above 8 $m/s^2$ without incurring significant yaw rate oscillation. Experimentally validated results using model-predictive control [21][8] demonstrate the ability to balance competing objectives of vehicle path tracking and stability, but must explicitly sacrifice one for the other in emergency situations. Furthermore, model-predictive controllers consist of complex optimization problems that make fundamental issues such as stability and tracking performance difficult to analyze mathematically.

This chapter therefore presents the design of a feedback-feedforward steering controller capable of combined stability and accurate path tracking at the limits of handling. A baseline controller with lookahead steering feedback and feedforward based on the nonlinear handling diagram is presented in §2.2. Section 2.3 uses steady-state simulation results to demonstrate the relatively poor path tracking performance of this baseline controller. In §2.4, we consider a modified steering feedback that aims to keep the vehicle sideslip tangent to the desired path. This approach results in a closed-loop steering response with zero steady-state lateral path deviation, but at the cost of poor stability margins. A better design approach, presented in §2.5, is to incorporate the desired sideslip behavior as a feedforward input, which significantly improves path tracking while maintaining robust stability margins. Section 2.6 provides experimental path tracking data collected on an Audi TTS test vehicle at Thunderhill Raceway in Willows, California, with combined lateral and longitudinal accelerations up to 9.5 $m/s^2$.

## 2.1 Path Description

The objective of the steering controller presented in this chapter is to follow a path generated by a separate high level controller. While there are several ways to mathematically represent the coordinates of a desired path, the controller design will assume the desired trajectory is defined as a series of curvilinear $(s, \kappa(s))$ coordinates, where $s$ is the distance along the path and $\kappa(s)$ is the instantaneous path curvature. This coordinate system is chosen because the curvature of a path is very intuitive to map into a desired lateral vehicle force, and ultimately a desired vehicle steering input. The chosen path description is illustrated for a simple path in Figure 2.1.

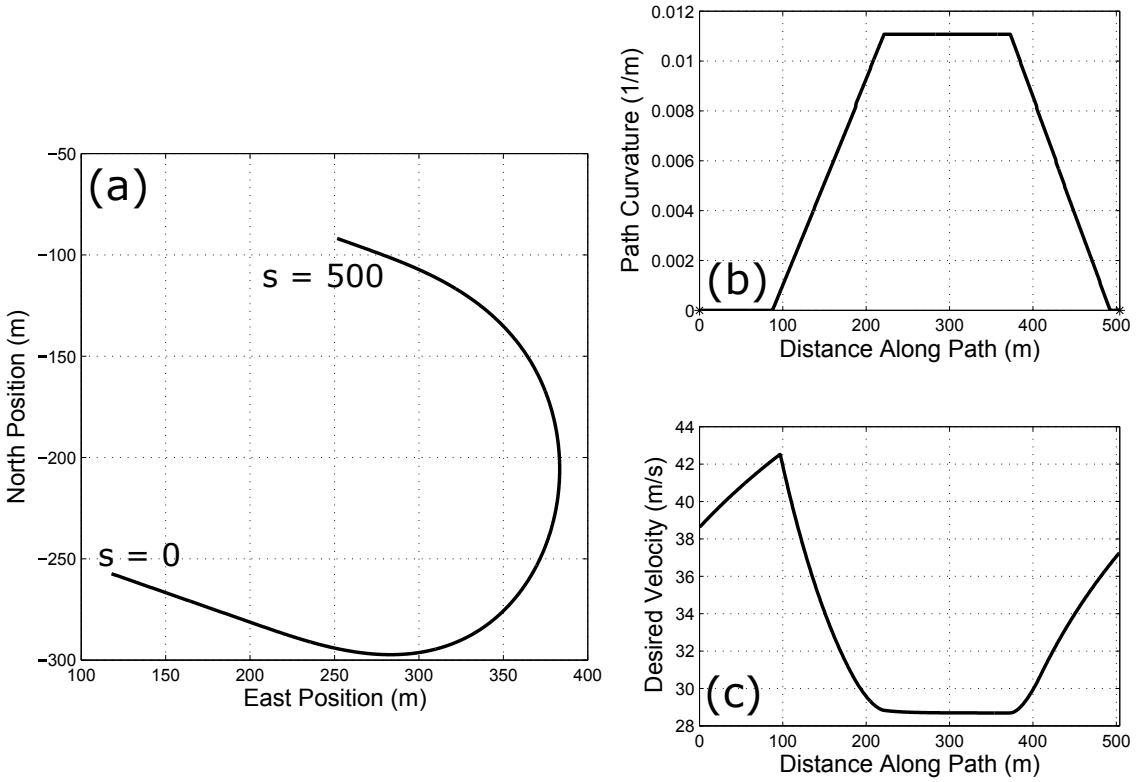


Figure 2.1: (a) A 500 meter path plotted in Cartesian coordinates. (b) Curvature profile $\kappa(s)$ as function of path length $s$ for associated path. (c) Example velocity profile $(s, U_x^{\text{des}}(s))$ for path.

Additionally, given the speed dependence of an automobile's steering dynamics, the steering controller also requires knowledge of the desired speed profile (Figure 2.1(c)), which must also come from the high level trajectory planner. The speed profile can be represented as a series of $(s, U_x^{\text{des}}(s))$ coordinates, where $U_x^{\text{des}}(s)$ is the desired speed at a given distance along the path.

## 2.2 Controller Architecture

A block diagram of a typical feedback-feedforward structure for steering control is shown in Fig. 2.2. Inputs to the feedforward steering angle $\delta_{\text{FFW}}$ are the current path curvature $\kappa$ and forward velocity $U_x$. Inputs to the feedback steering angle $\delta_{\text{FB}}$ are lateral path deviation $e$ and path heading error $\Delta\Psi$ (Fig. 2.3). The total steering command $\delta$ is the sum of the feedback and feedforward inputs.

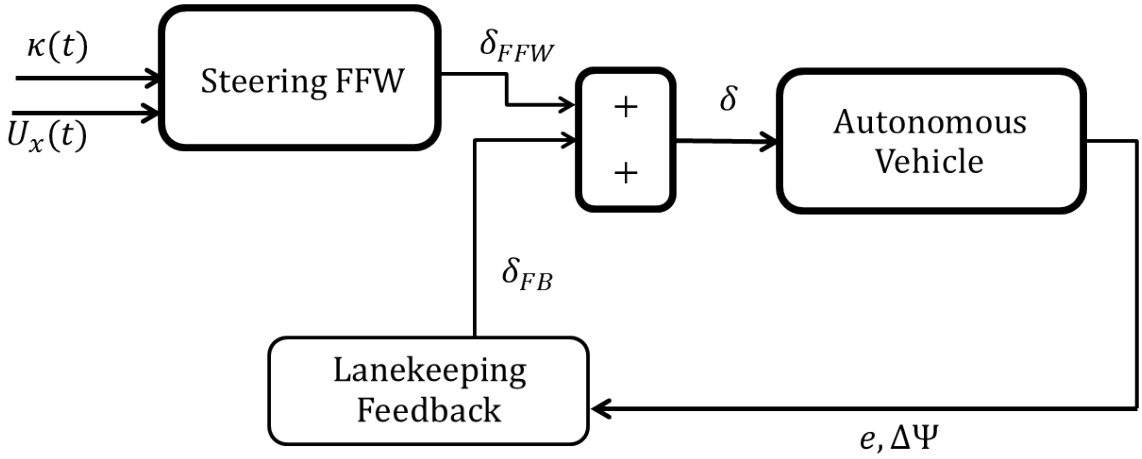


Figure 2.2: Block diagram of feedback-feedforward steering controller.

### 2.2.1 Feedforward Steering Design

The objective of the steering feedforward is to provide an estimate of the steer angle required to traverse a path with a known path curvature and velocity profile. This minimizes the level of compensation required by the steering feedback, reducing tracking errors and allowing for less overall control effort. To simplify the controller

structure, the feedforward steering angle should depend only on the desired trajectory and be independent of the actual vehicle states.

The proposed structure of the steering feedforward begins with the assumption that vehicle dynamics are given by the planar "bicycle" model, with relevant vehicle states and dimensions shown in Fig. 2.3 and described in Table 2.1. The planar bicycle model makes the key assumption that the left and right tires act to produce a single combined lateral force, resulting in just two lateral forces $F_{\mathrm{yf}}$ and $F_{\mathrm{yr}}$ acting at the front and rear. Actuation of steer angle $\delta$ at the front tire results in generation of the lateral tire forces through the tire slip angles $\alpha_{\mathrm{f}}$ and $\alpha_{\mathrm{r}}$. The two resulting states that evolve are vehicle yaw rate $r$, which describes the vehicle angular rotation, and sideslip $\beta$, which is the ratio of lateral velocity $U_y$ to longitudinal velocity $U_x$.

Table 2.1: Bicyle Model Definitions

| Parameter | Symbol | Units |
|---|---|---|
| Front axle to CG | $a$ | m |
| Rear axle to CG | $b$ | m |
| Front Lateral Force | $F_{\mathrm{yf}}$ | N |
| Front Tire Slip | $\alpha_{\mathrm{f}}$ | rad |
| Rear Lateral Force | $F_{\mathrm{yr}}$ | N |
| Rear Tire Slip | $\alpha_{\mathrm{r}}$ | rad |
| Steer Angle Input | $\delta$ | rad |
| Yaw Rate | $r$ | rad/s |
| Sideslip | $\beta$ | rad |
| Lateral Path Deviation | $e$ | m |
| Heading Deviation | $\Delta\Psi$ | rad |
| Longitudinal Velocity | $U_x$ | m/s |
| Lateral Velocity | $U_y$ | m/s |

In addition to the two vehicle states $\beta$ and $r$, two additional states are required to describe the vehicle's position relative to the desired path. These are also shown in Fig. 2.3. The lateral path deviation, or lateral error, $e$, is the distance from the vehicle center of gravity to the closest point on the desired path. The vehicle heading error $\Delta\Psi$ is defined as the angle between the vehicle's centerline and the tangent line drawn on the desired path at the closest point. Note that the longitudinal dynamics

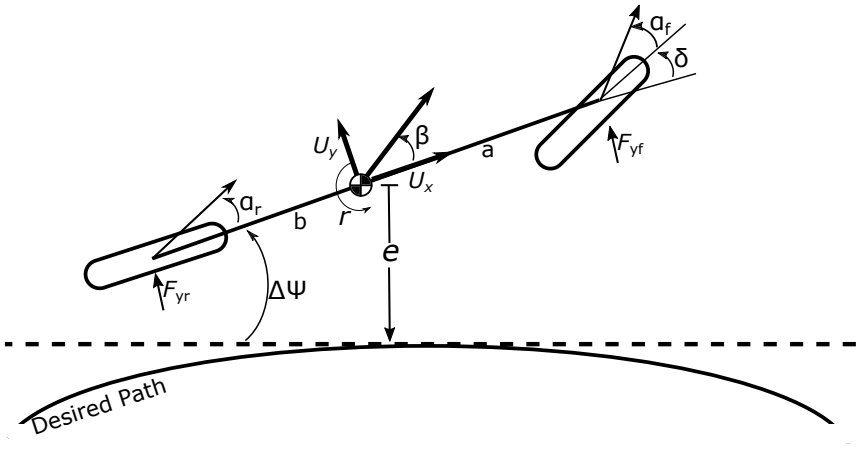


Figure 2.3: Schematic of planar bicycle model.

of the vehicle are not explicitly modeled in this formulation. Instead of keeping the longitudinal velocity $U_x$ as a vehicle state, the longitudinal velocity is treated as a time-varying parameter.

The problem of determining suitable feedforward lateral tire forces for autonomous path following was studied by Krisada and Gerdes [46]. The (linearized) equations of motion for the states shown in Fig. 2.3 are given by:

$$\dot{\beta} = \frac{F_{\mathrm{yf}} + F_{\mathrm{yr}}}{mU_x} - r \tag{2.1a}$$

$$\dot{r} = \frac{aF_{\mathrm{yf}} - bF_{\mathrm{yr}}}{I_z} \tag{2.1b}$$

$$\dot{e} = U_x(\beta + \Delta\Psi) \tag{2.1c}$$

$$\Delta\dot{\Psi} = r - \dot{s}\kappa \tag{2.1d}$$

where $m$ and $I_z$ are the vehicle mass and out-of-plane rotational inertia, and $s$ is the distance along the desired path. Taking time derivatives of $\dot{e}$ and $\Delta\dot{\Psi}$ and substituting

from (2.1a) and (2.1b) yields:

$$\ddot{e} = \frac{F_{\mathrm{yf}} + F_{\mathrm{yr}}}{m} - U_x \kappa \dot{s} \tag{2.2a}$$

$$\Delta\ddot{\Psi} = \frac{aF_{\mathrm{yf}} - bF_{\mathrm{yr}}}{I_z} - \kappa \ddot{s} - \dot{\kappa}\dot{s} \tag{2.2b}$$

In general, the values chosen for the feedforward front and rear tire forces $F_{\mathrm{yf}}$ and $F_{\mathrm{yr}}$ should bring $\ddot{e}$ and $\Delta\ddot{\Psi}$ to zero. However, for a typical front-steer vehicle, direct control is only available for the front steering force $F_{\mathrm{yf}}$ via command of the steering input $\delta$. The rear tire force depends indirectly on the steering angle via the build-up of rear tire slip $\alpha_{\mathrm{r}}$. It is therefore not possible to simultaneously eliminate both the lateral tracking error and heading angle error.

An alternative is to consider eliminating a weighted combination of the two error states by eliminating the lateral tracking error $e_p$ at a specified point $x_p$ in front of the vehicle, as shown in Fig. 2.4. The error dynamics at this projected point are given by:

$$e_p = e + x_p \Delta\Psi \tag{2.3a}$$

$$\ddot{e}_p = \frac{F_{\mathrm{yf}} + F_{\mathrm{yr}}}{m} - U_x \kappa \dot{s} + x_p \frac{aF_{\mathrm{yf}} - bF_{\mathrm{yr}}}{I_z} - x_p(\kappa \ddot{s} + \dot{\kappa}\dot{s}) \tag{2.3b}$$

Kritayakirana and Gerdes [46] proposed the center of percussion $x_{\mathrm{cop}} = \frac{I_z}{bm}$ as a convenient projection point for the feedforward steering. Substituting $x_p = x_{\mathrm{cop}}$ and $\ddot{e}_{\mathrm{cop}} = 0$ yields a simplified equation for the front tire force:

$$F_{\mathrm{yf}} = \frac{mb}{L}\left(U_x^2 \kappa + x_{\mathrm{cop}}(\kappa \ddot{s} + \dot{\kappa}\dot{s})\right) \tag{2.4}$$

The benefit of choosing the center of percussion becomes clear in (2.4). The error dynamics at the center of percussion are independent of the rear tire force, which can be highly transient when the vehicle is cornering near the limits of handling. This leaves the only control input as $F_{\mathrm{yf}}$, which can be directly manipulated by the front wheel steering.

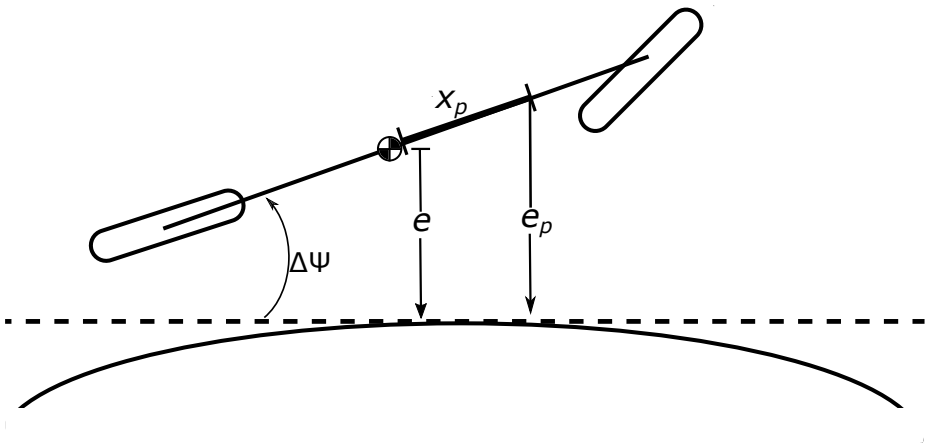


Figure 2.4: Projection of lateral error at distance $x_p$ in front of the center of gravity.

A feedforward steering approach based on eliminating tracking error at the center of percussion performed well experimentally at lateral accelerations up to 7-8 m/s$^2$ [46]. However, at higher lateral accelerations, the closed-loop steering response became underdamped, and the result was significant levels of yaw rate and steering wheel oscillation. This was due to the complex relationship between the desired feedforward front tire force $F_{\mathrm{yf}}^{\mathrm{FFW}}$ and the required steering angle $\delta_{\mathrm{FFW}}$ needed to achieve that desired tire force. In general, this relationship is dependent on the vehicle yaw rate $r$ and sideslip $\beta$:

$$\delta_{\mathrm{FFW}} = \frac{U_x\beta + ar}{U_x} - f^{-1}(F_{\mathrm{yf}}^{\mathrm{FFW}}) \tag{2.5}$$

where $f^{-1}(F_{\mathrm{y}})$ is an inverse tire model relating tire force to tire slip. Even though (2.4) does not explicitly depend on rear tire force, the feedforward steering command $\delta_{\mathrm{FFW}}$ implicitly depends on the transient rear tire dynamics through the yaw rate and sideslip states. At the limits of handling, these transient dynamics become underdamped, resulting in an oscillatory $\delta_{\mathrm{FFW}}$ command.

To eliminate yaw rate oscillation, we propose simplifying the feedforward tire forces by assuming steady-state cornering conditions. Setting $\dot{s} = U_x$, $\ddot{s} = \dot{\kappa} = 0$ in (2.4) and $\dot{r} = 0$ in (2.1) yields the following front and rear tire forces:

$$F_{\mathrm{yf}}^{\mathrm{FFW}} = \frac{mb}{L} U_x^2 \kappa \tag{2.6a}$$

$$F_{\mathrm{yr}}^{\mathrm{FFW}} = \frac{ma}{L} U_x^2 \kappa \tag{2.6b}$$

At steady-state conditions and assuming small angles, the feedforward steering angle of the vehicle relates to the front and rear lateral tire slip $\alpha_{\mathrm{f}}$ and $\alpha_{\mathrm{r}}$ and path curvature $\kappa$ by vehicle kinematics:

$$\delta_{\mathrm{FFW}} = L\kappa - \alpha_{\mathrm{f}}^{\mathrm{FFW}} + \alpha_{\mathrm{r}}^{\mathrm{FFW}} \tag{2.7}$$

where $\alpha_{\mathrm{f}}^{\mathrm{FFW}}$ and $\alpha_{\mathrm{r}}^{\mathrm{FFW}}$ are the lumped front and rear feedforward tire slip angles. The choice of feedforward tire slip angles is related to the tire forces in (2.6) via the inverted tire model $f^{-1}(F_{\mathrm{y}})$. To account for saturation of tire force with increasing tire slip magnitude, a single friction coefficient brush Fiala model [58] maps lateral tire slip angles into tire force as follows:

$$F_{\mathrm{y}\diamond} = \begin{cases} -C_\diamond \tan\alpha_\diamond + \frac{C_\diamond^2}{3\mu F_{\mathrm{z}\diamond}} |\tan\alpha_\diamond| \tan\alpha_\diamond \\ \qquad -\frac{C_\diamond^3}{27\mu^2 F_{\mathrm{z}\diamond}^2} \tan^3\alpha_\diamond, & |\alpha_\diamond| < \arctan\left(\frac{3\mu F_{\mathrm{z}\diamond}}{C_\diamond}\right) \\ -\mu F_{\mathrm{z}\diamond} \mathrm{sgn}\ \alpha_\diamond, & \text{otherwise} \end{cases} \tag{2.8}$$

where the symbol $\diamond \in [\mathrm{f}, \mathrm{r}]$ denotes the lumped front or rear tire, $\mu$ is the surface coefficient of friction, and $C_\diamond$ and $F_{\mathrm{z}\diamond}$ are the corresponding cornering stiffness and normal load parameters. The cornering stiffnesses and friction coefficient are determined from experimental data taken from a ramp-steer maneuver, as shown in Fig. 2.5.

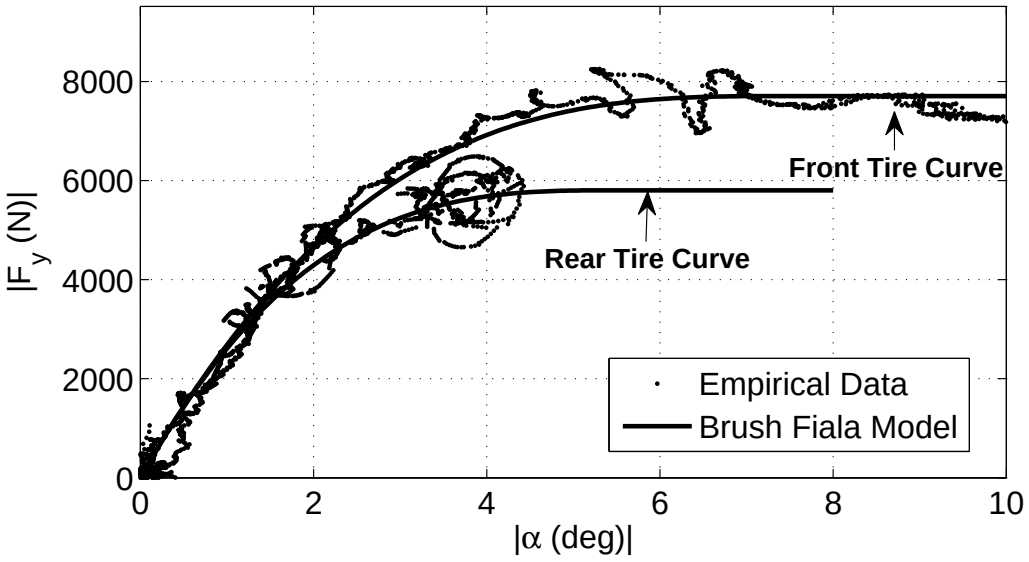


Figure 2.5: Nonlinear tire curves for FFW steering.

### 2.2.2 Feedback Steering Design

With the feedforward design complete, the remaining step is to design the feedback controller. The goal of the feedback controller is to minimize a *lookahead* error $e_{\mathrm{LA}}$, which is the vehicle tracking error projected a distance $x_{\mathrm{LA}}$ in front of the vehicle (Fig. 2.6).

The lookahead error and resulting feedback control law are given by:

$$e_{\mathrm{LA}} = e + x_{\mathrm{LA}} \Delta\Psi \tag{2.9a}$$

$$\delta_{\mathrm{FB}} = -k_{\mathrm{p}} e_{\mathrm{LA}} \tag{2.9b}$$

with proportional gain $k_{\mathrm{p}}$. The control law (2.9) is a natural extension of potential field lanekeeping, as described by Rossetter et al. in [62], which also provides heuristics for selecting $k_{\mathrm{p}}$ and $x_{\mathrm{LA}}$. Desirable stability properties of this feedback controller are demonstrated in [72].

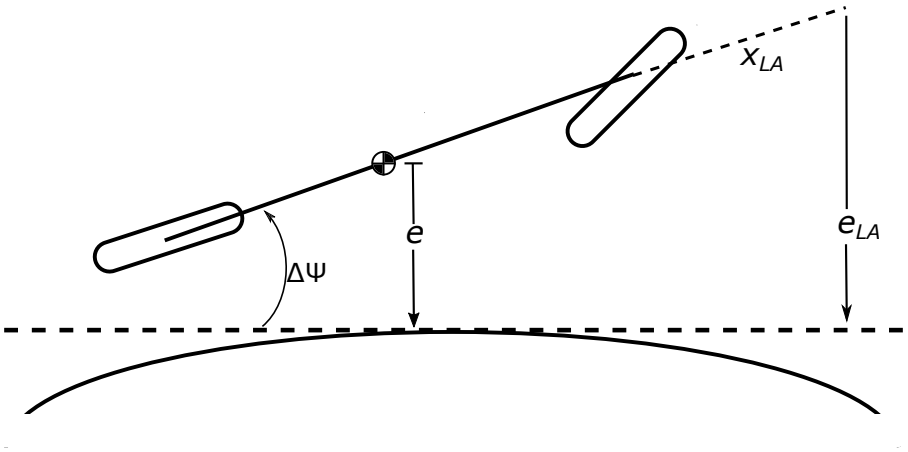


Figure 2.6: Schematic of planar bicycle model showing projected lookahead error.

## 2.3 Predicted Steady-State Path Tracking Error

Simulation results provide useful insight about the steady-state path tracking behavior of the baseline feedback-feedforward system. Linearized equations of motion for the vehicle and error states can be written in state-space form, with state variable $x$ and control input $\delta$ defined as follows:

$$x = [e \;\; \Delta\Psi \;\; r \;\; \beta]^T \tag{2.10a}$$

$$\delta = \delta_{\mathrm{FB}} + \delta_{\mathrm{FFW}} \tag{2.10b}$$

$$= [-k_{\mathrm{LK}} \;\; -k_{\mathrm{LK}} x_{\mathrm{LA}} \;\; 0 \;\; 0]\, x + \left( L + \frac{K_{\mathrm{ug}} U_x^2}{g} \right) \kappa \tag{2.10c}$$

where $K_{\mathrm{ug}}$ is the vehicle understeer gradient, a parameter related to the front/rear weight distribution of the vehicle. Note that $\delta_{\mathrm{FB}}$ in (2.10b) depends on the state variable, and $\delta_{\mathrm{FFW}}$ depends on the path curvature. Rewriting the vehicle state equations of motion using curvature as the input results in:

$$\dot{x} = Ax + B\kappa \tag{2.11a}$$

$$A = \begin{bmatrix} 0 & U_x & 0 & U_x \\ 0 & 0 & 1 & 0 \\ \frac{-ak_{\mathrm{p}}C_{\mathrm{f}}}{I_{\mathrm{z}}} & \frac{-ak_{\mathrm{p}}x_{\mathrm{LA}}C_{\mathrm{f}}}{I_{\mathrm{z}}} & \frac{-(a^2C_{\mathrm{f}}+b^2C_{\mathrm{r}})}{U_xI_{\mathrm{z}}} & \frac{bC_{\mathrm{r}}-aC_{\mathrm{f}}}{I_{\mathrm{z}}} \\ \frac{-k_{\mathrm{p}}C_{\mathrm{f}}}{mU_x} & \frac{-k_{\mathrm{p}}x_{\mathrm{LA}}C_{\mathrm{f}}}{mU_x} & \frac{bC_{\mathrm{r}}-aC_{\mathrm{f}}}{mU_x^2} - 1 & \frac{-(C_{\mathrm{f}}+C_{\mathrm{r}})}{mU_x} \end{bmatrix} \tag{2.11b}$$

$$B = \begin{bmatrix} 0 & -U_x & \dfrac{aC_{\mathrm{f}}G_{\mathrm{FFW}}}{I_{\mathrm{z}}} & \dfrac{C_{\mathrm{f}}G_{\mathrm{FFW}}}{mU_x} \end{bmatrix}^T \tag{2.11c}$$

where $G_{\mathrm{FFW}} = (L + \frac{K_{\mathrm{ug}}U_x^2}{g})$ and $C_{\mathrm{f}}$ and $C_{\mathrm{r}}$ are the lumped front and rear cornering stiffnesses. Fig. 2.7 shows results from using the linear model (2.11) to compute steady-state path tracking errors over a range of vehicle speeds, given a constant lateral acceleration of $a_y = 3$ m/s$^2$. Additionally, steady-state results from the nonlinear feedforward control law (2.7) are plotted for the case where $a_y = 7$ m/s$^2$. For this high lateral acceleration case, the equations of motion (2.1) and nonlinear Fiala tire model (2.8) are used to predict the steady-state results. In general, steady-state lateral dynamics are reached within one second for a constant lateral acceleration maneuver. At low lateral accelerations, when the vehicle dynamics are dictated by linear equations of motion, the accumulated steady-state tracking error is small ($<$ 0.05 m) at highway speeds. However, when steering at the limits of handling, Fig. 2.7 shows that the increased sideslip of the vehicle results in higher steady-state tracking error, creating challenges for accurate driving in safety-critical situations.

A qualitative explanation for this steady-state error is shown in Fig. 2.8. Since the steering controller acts to eliminate a weighted sum of heading error $\Delta\Psi$ and lateral path deviation $e$, the lookahead feedback is successful in minimizing the desired metric $e_{\mathrm{LA}}$ at the projected distance $x_{\mathrm{LA}}$ in front of the vehicle. However, this still allows for steady-state equilibria where values of $e$ and $\Delta\Psi$ themselves are nonzero.

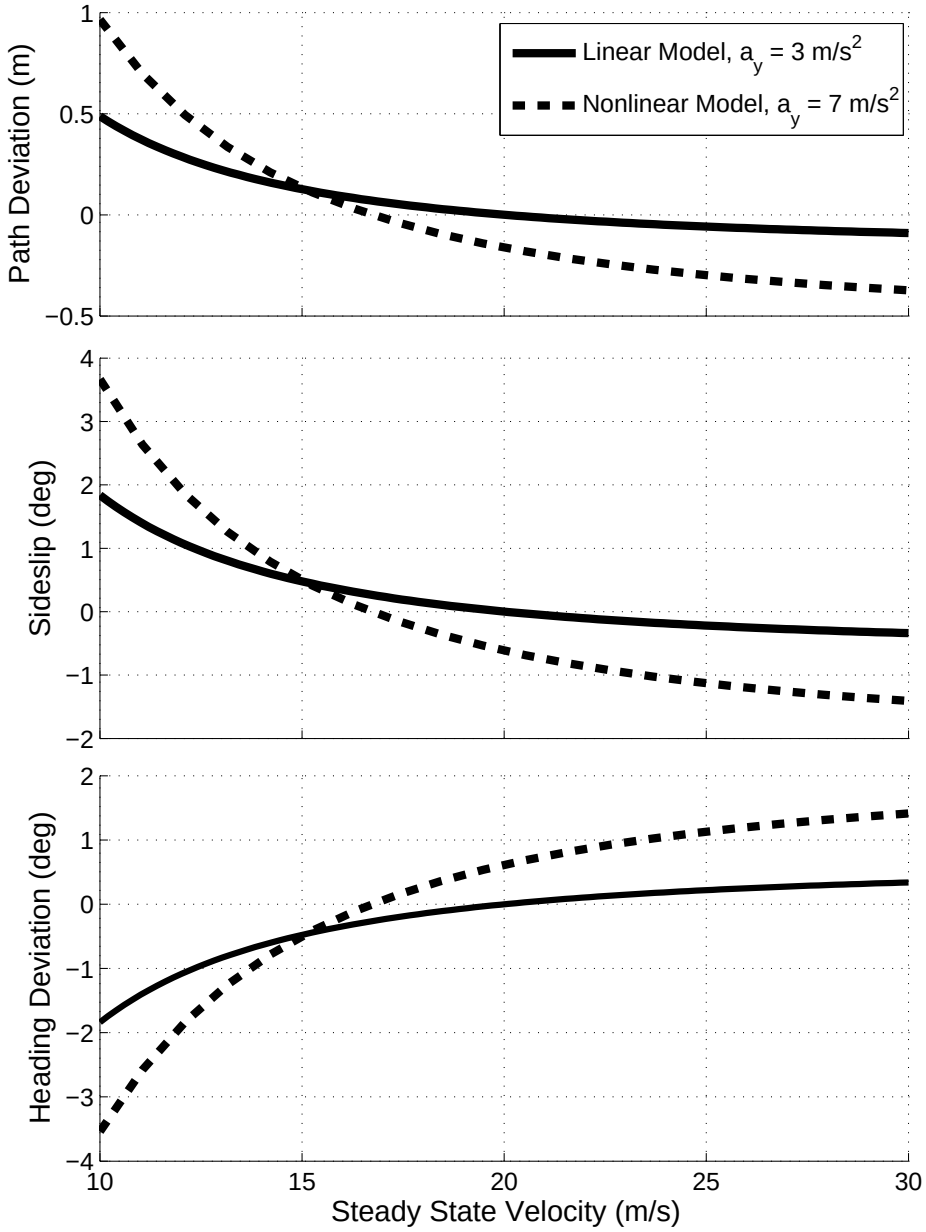


Figure 2.7: Steady-state path tracking error $e$, sideslip $\beta$ and heading deviation $\Delta\Psi$ as a function of vehicle speed. Results are plotted for the linear model, with fixed lateral acceleration $a_y = 3$ m/s$^2$, and for the nonlinear model, with fixed lateral acceleration $a_y = 7$ m/s$^2$.

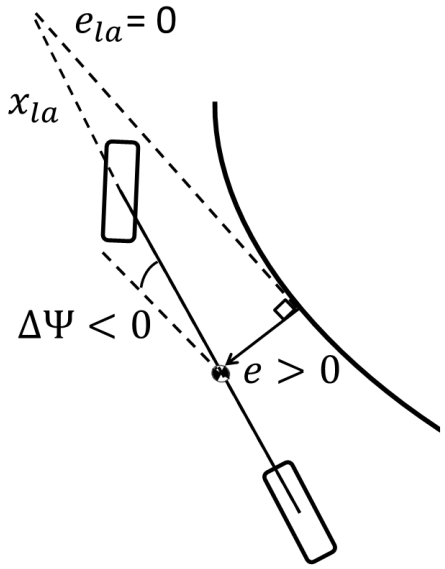


Figure 2.8: Steady-state cornering where vehicle has lateral error but no lookahead error.

## 2.4 Incorporating Sideslip-Path Tangency into Steering Feedback

An interesting observation from Fig. 2.7 is that the steady-state path deviation is zero at a vehicle speed of around $U_x$=20 m/s for the linear model and $U_x$=17 m/s for the nonlinear model. At these speeds, the steady-state vehicle sideslip is predicted to be zero as well, and the vehicle heading $\Psi$ naturally becomes tangent to the desired path heading.

This observation motivates a second form of lookahead feedback where the feedback objective is to maintain the vehicle velocity vector $U = \langle U_x \ U_y \rangle$ tangent to the desired path, as shown in Fig. 2.9. Since the direction of $U$ is given by $\angle U = \Psi + \beta$, the resulting control law is:

$$\delta_{\mathrm{FB}} = -k_{\mathrm{p}} \left( e + x_{\mathrm{LA}} (\angle U - \Psi_{\mathrm{r}}) \right) \tag{2.12a}$$

$$= -k_{\mathrm{p}} \left( e + x_{\mathrm{LA}} (\Psi + \beta - \Psi_{\mathrm{r}}) \right) \tag{2.12b}$$

$$= -k_{\mathrm{p}} \left( e + x_{LA} (\Delta\Psi + \beta) \right) \tag{2.12c}$$

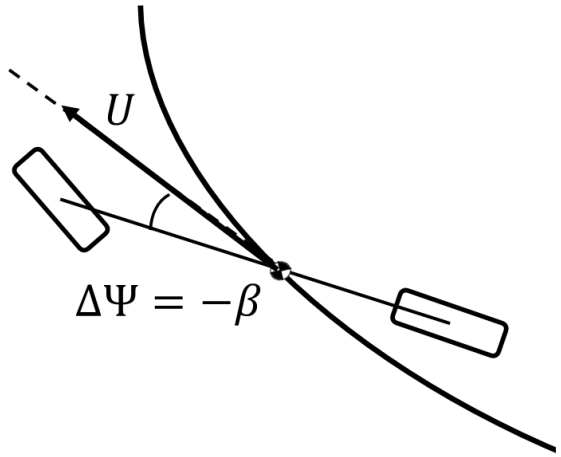


Figure 2.9: Zero steady-state lateral deviation requires vehicle velocity vector to be tangent to path.

where $\Psi_{\mathrm{r}}$ is the heading of the desired vehicle path at a given point. The modified control law can be modeled by reformulating the matrix $A$ in (2.11) as:

$$A = \begin{bmatrix} 0 & U_x & 0 & U_x \\ 0 & 0 & 1 & 0 \\ \frac{-ak_{\mathrm{p}}C_{\mathrm{f}}}{I_{\mathrm{z}}} & \frac{-ak_{\mathrm{p}}x_{\mathrm{LA}}C_{\mathrm{f}}}{I_{\mathrm{z}}} & \frac{-a^2C_{\mathrm{f}}-b^2C_{\mathrm{r}}}{U_xI_{\mathrm{z}}} & \frac{\mathbf{bC_r-aC_f(1-ak_px_{LA})}}{\mathbf{I_z}} \\ \frac{-k_{\mathrm{p}}C_{\mathrm{f}}}{mU_x} & \frac{-k_{\mathrm{p}}x_{\mathrm{LA}}C_{\mathrm{f}}}{mU_x} & \frac{-aC_{\mathrm{f}}-bC_{\mathrm{r}}}{mU_x^2}-1 & \frac{-(\mathbf{C_f(1+k_px_{LA})+C_r})}{\mathbf{mU_x}} \end{bmatrix} \tag{2.13}$$

Note that (2.13) is equal to (2.11b) with the exception of the last column, highlighted in bold. Fig. 2.10 shows the resulting steady-state behavior, and indicates that lateral error $e$ settles to zero for all velocities.

However, the disadvantage of directly adding vehicle sideslip into the feedback control is reduced stability margins. Closed-loop eigenvalues of (2.11b) and (2.13) are plotted in Fig. 2.11 as a function of increasing vehicle speed from 5 m/s to 25 m/s.

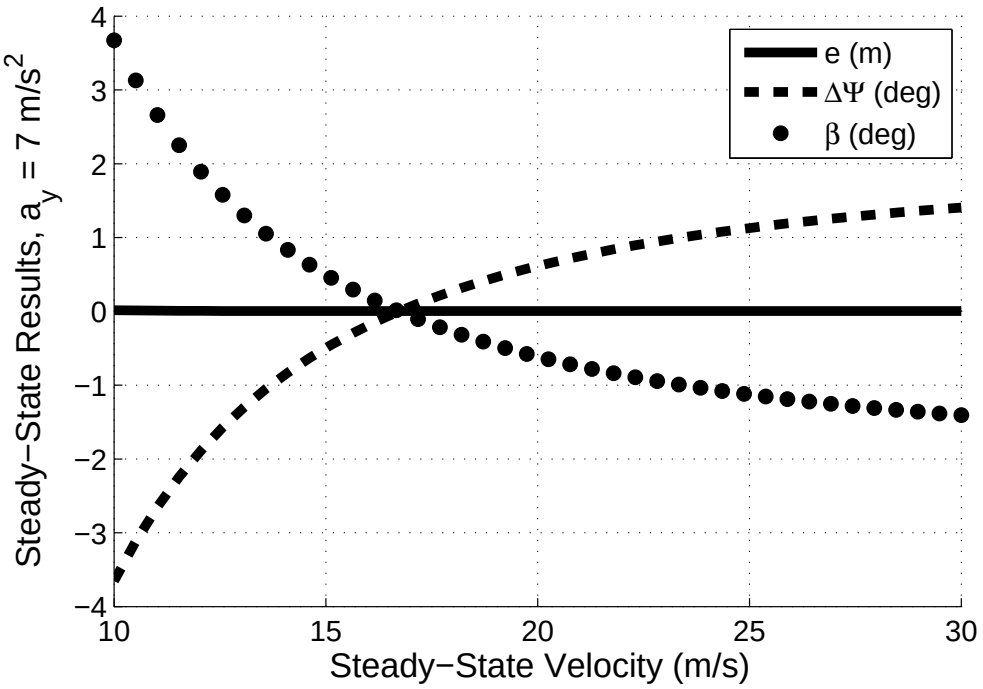


Figure 2.10: Steady-state simulation results with sideslip added to feedback control, using the nonlinear vehicle model with fixed lateral acceleration of 7 m/s$^2$.

The results indicate that the closed-loop steering response is well-damped ($\zeta = 0.9$ at a vehicle speed of 25 m/s) with the original lookahead feedback controller. However, when, the steering feedback acts to keep the vehicle sideslip tangent to the desired path via (2.12), the closed-loop steering response becomes highly underdamped ($\zeta = 0.2$ at $U_x = 25$ m/s).

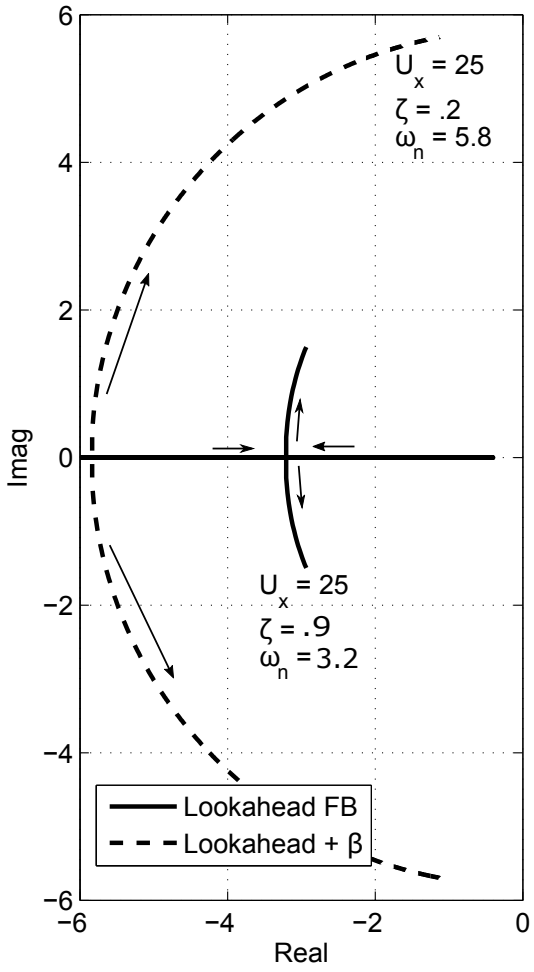


Figure 2.11: Closed-loop pole locations for steering system as vehicle speed is varied from 5 to 25 m/s. Damping ratio $\zeta$ and natural frequency $\omega_n$ are shown for $U_x = 25$. Root locus plots are shown for both the lookahead feedback controller (2.11) as well as the feedback controller with added sideslip (2.13). Root locus moves in direction of arrows as vehicle speed is increased.

Note that the results shown in Fig. 2.11 are for a single vehicle and controller parameterization (see Table 1). In general, the reduction in stability margin will vary significantly depending on the vehicle understeer gradient and steering controller gains, namely the lookahead distance. Fig. 2.12 shows the critical speed $V_{\mathrm{cr}}$, beyond which the closed-loop steering system becomes unstable, for neutral, understeering, and oversteering configurations as a function of $x_{\mathrm{LA}}$. For an understeering vehicle, lookahead feedback is always stable as long as the lookahead point $x_{\mathrm{LA}}$ is above a certain critical value (a conclusion derived in [62]). Even in situations where the

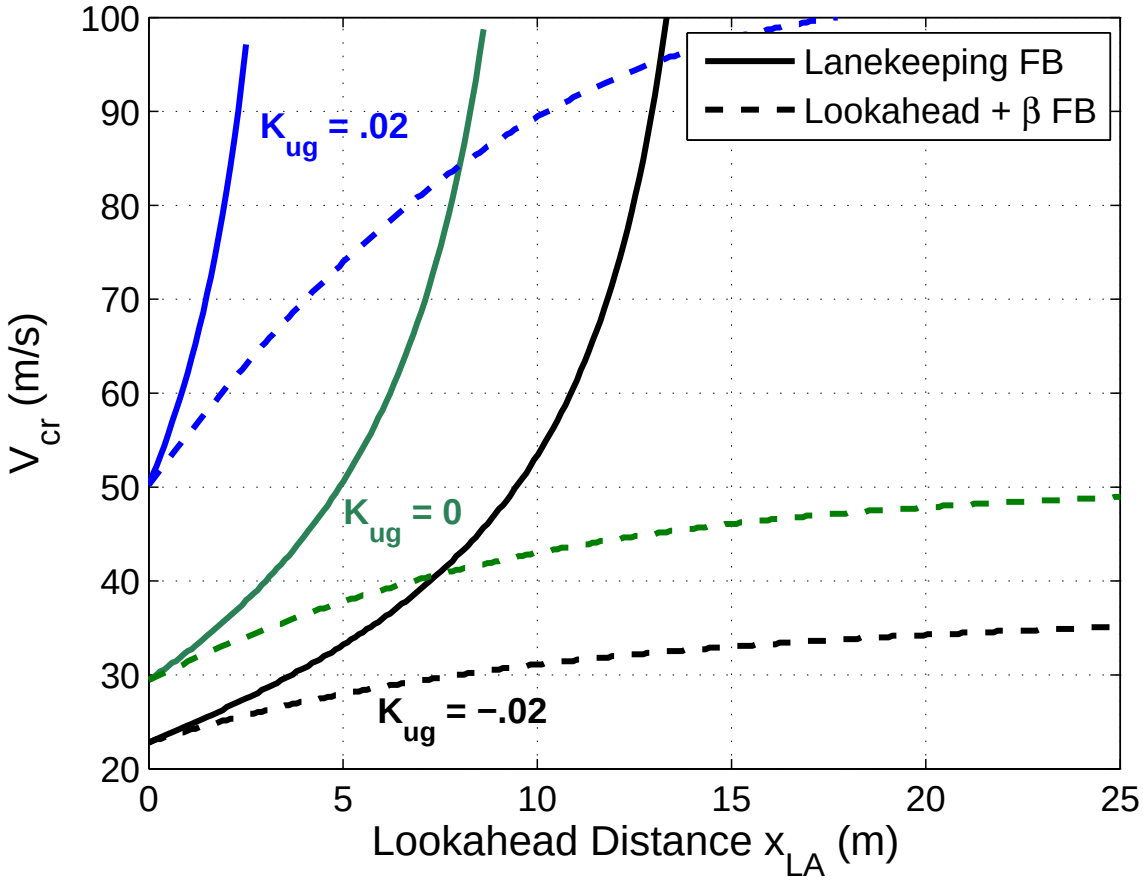


Figure 2.12: Maximum speed for closed-loop stability for the original lookahead feedback and the modified feedback with sideslip tracking. Results are based on eigenvalue computations of the $A$ matrix of the linear vehicle model.

vehicle is in a neutral steer or oversteer configuration, the critical speed for closed-loop stability increases rapidly with lookahead distance. A different trend is present when sideslip tracking is added to the steering feedback. Fig. 2.12 shows that the critical speeds for stability increase very slowly as a function of lookahead distance.

## 2.5 Incorporating Sideslip Information Into Steering Feedforward

Given the trade-off between path tracking and stability when sideslip-path alignment is enforced via feedback, a promising approach is to replace vehicle sideslip $\beta$ in (2.12) with the predicted steady-state sideslip value $\beta_{\text{ss}}$ for a given vehicle speed and curvature.

The rear tire slip for a fixed track vehicle, assuming small angles, is given by

$$\alpha_{\text{r}} = \beta - \frac{br}{U_x} \tag{2.14}$$

At steady-state, $\alpha_{\text{r}} = \alpha_{\text{r}}^{\text{FFW}}$ from (2.7) and $r = \kappa U_x$, yielding the following feedback control law:

$$\delta_{\text{FB}} = -k_{\text{P}} \left( e + x_{\text{LA}} (\Delta\Psi + \beta_{\text{ss}}) \right) \tag{2.15a}$$

$$\beta_{\text{ss}} = \alpha_{\text{r}}^{\text{FFW}} + b\kappa \tag{2.15b}$$

The effect of this change is to remove the steering controller's dependence on *real-time* sideslip information. The steering controller will now act to align the vehicle's predicted *steady-state* sideslip along the desired path. Since the controller no longer has feedback on vehicle sideslip, the state matrix equation $A$ for the closed-loop system dynamics is now once again given by (2.11b), which was shown to have desirable stability properties as a function of $K_{\text{ug}}$ and $x_{\text{LA}}$ (Fig. 2.12). The sideslip now affects the vehicle path tracking dynamics through the feedforward path, since the predicted steady-state sideslip $\beta_{ss}$ depends only on the desired speed $U_x$ and curvature $\kappa$ as well

as the feedforward tire model. The $B$ matrix in (2.11) now becomes:

$$B = \begin{bmatrix} 0 & -U_x & \dfrac{aC_{\mathrm{f}}(G_{\mathrm{FFW}} + G_\beta)}{I_{\mathrm{z}}} & \dfrac{C_{\mathrm{f}}(G_{\mathrm{FFW}} + G_\beta)}{mU_x} \end{bmatrix}^T \tag{2.16a}$$

$$G_\beta = U_x^2 \frac{ma}{L} \frac{k_{\mathrm{p}} x_{\mathrm{LA}}}{C_{\mathrm{r}}} \tag{2.16b}$$

Assuming perfect knowledge of the feedforward tire model, the resulting steady-state lateral path deviation will be zero at all vehicle speeds, as shown in Fig. 2.10. However, error in the feedforward tire model will result in steady-state lateral path

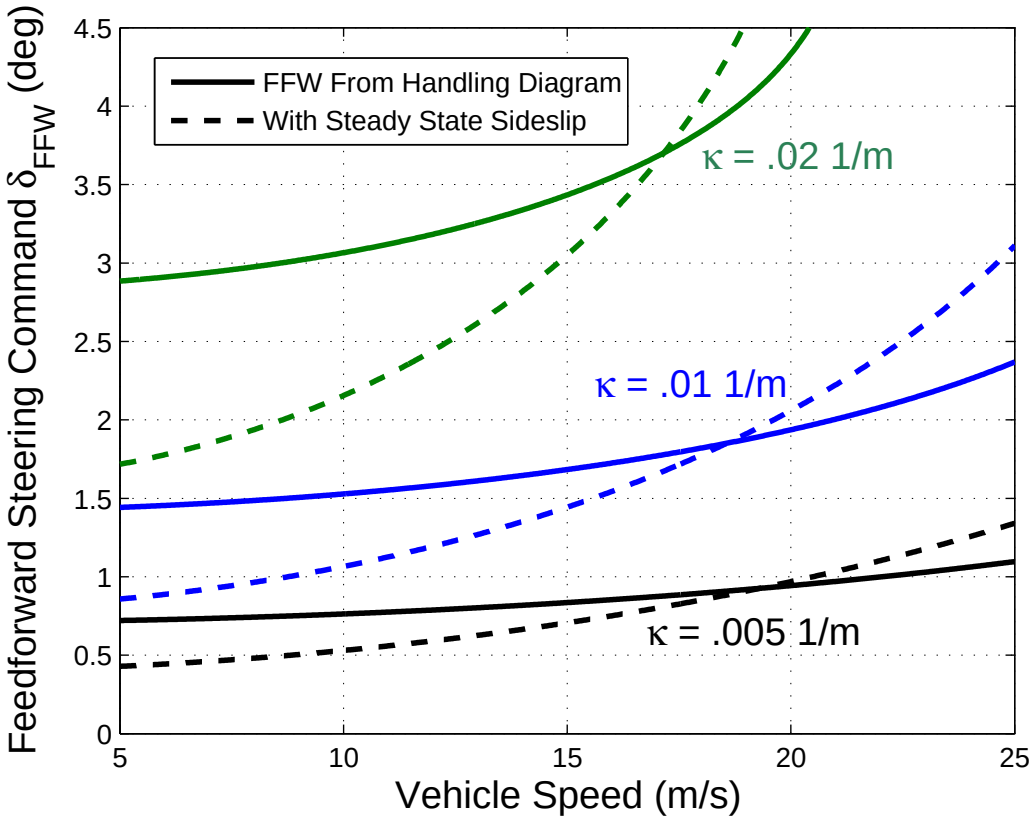


Figure 2.13: Effect of incorporating sideslip behavior into feedforward steering command $\delta_{\mathrm{FFW}}$, as a function of vehicle speed and desired path curvature.

deviation. The effect of incorporating steady-state sideslip into the feedforward control is shown in Fig. 2.13. The feedforward steering command $\delta_{\text{FFW}}$ as a function of path curvature and vehicle speed is plotted both for the original feedforward control law (2.11c) as well as the sideslip-incorporating feedforward command (2.16).

## 2.6 Experimental Results

### 2.6.1 Experimental Setup

Experimental data was collected on "Shelley", an Audi TTS equipped with an electronic power steering motor for autonomous steering and active brake booster and throttle by wire for longitudinal control (Fig. 2.14). The testbed is also equipped with an integrated Differential Global Positioning System (DGPS) and Inertial Measurement Unit (IMU) in order to obtain global vehicle states. Mentioned previously in §1.3.1, this is the same vehicle developed by Stanford and Audi to autonomously drive the Utah Salt Flats and Pikes Peak Hill Climb [78].


Figure 2.14: Audi TTS used for experimental validation.

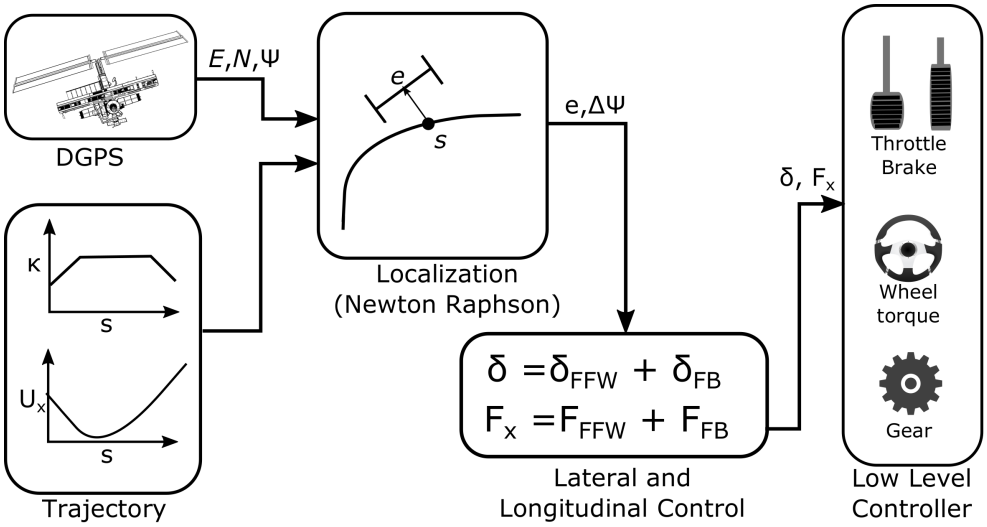


Figure 2.15: Diagram showing controller setup.

The overall controller setup is shown in Fig. 2.15. At each time step, the differential GPS unit obtains precise ( $< 2$ cm) measurements of the vehicle global East and North coordinates as well as the vehicle angular orientation $\Psi$. The controller also has prior knowledge of the desired high level trajectory, consisting of the clothoid curvature profile from Theodosis [74] and the speed profile from Kritayakirana [45]. Vehicle parameters for the Audi test vehicle and the lookahead feedback gains are presented in Table 2.2.

Table 2.2: Vehicle Parameters

| Parameter | Symbol | Value | Units |
|---|---|---|---|
| Vehicle mass | $m$ | 1500 | kg |
| Yaw moment of inertia | $I_z$ | 2250 | $\mathrm{kg \cdot m^2}$ |
| Front axle to CG | $a$ | 1.04 | m |
| Rear axle to CG | $b$ | 1.42 | m |
| Front cornering stiffness | $\mathrm{C_f}$ | 160 | $\mathrm{kN \cdot rad^{-1}}$ |
| Rear cornering stiffness | $\mathrm{C_r}$ | 180 | $\mathrm{kN \cdot rad^{-1}}$ |
| Lookahead Distance | $x_{\mathrm{LA}}$ | 14.2 | m |
| Lookahead Gain | $k_{\mathrm{p}}$ | 0.053 | rad/m |

The global East, North, and $\Psi$ measurements are fed into a Newton-Raphson

localization algorithm, which finds the closest point from the vehicle center of gravity to the path and outputs values of $e$ and $\Delta\Psi$. The resulting steering command is generated using the feedback-feedforward control laws developed in this chapter. In addition, to track the desired speed profile, a longitudinal force command is generated with a basic proportional control law presented by Kritayakirana [45]. Finally, a low level controller developed by Audi and Stanford maps the steer angle and force command into the resulting throttle, brake, steering wheel torque, and gear shift commands. The control loop operates at 200 Hz, with computations performed on a dSPACE MicroAutobox real-time embedded computer.

The site for data collection was a 3.1 mile paved racing circuit (friction coefficient $\mu \approx 1$) at Thunderhill Raceway Park in Willows, CA. Fig. 2.17 shows an overhead plot of the race track as well as the desired racing line. The desired racing line is parameterized as a curvature profile $\kappa(s)$ that varies with distance along the path. The curvature profile associated with the racing line is shown in Fig. 2.16 along with a typical longitudinal velocity profile $U_x$ from the path planner.

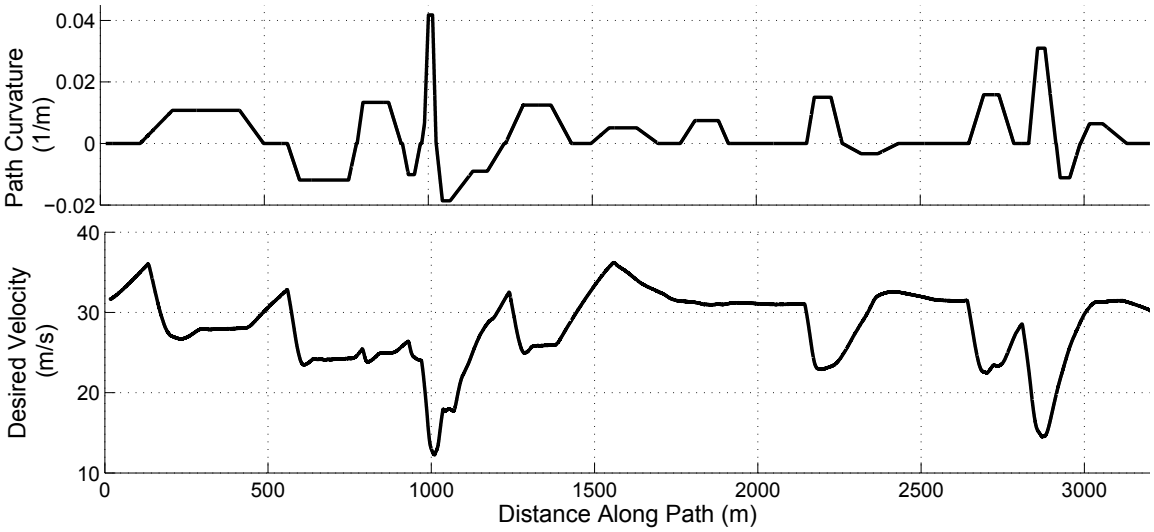


Figure 2.16: Curvature and velocity profile inputs for steering controller as a function of distance along racing line.

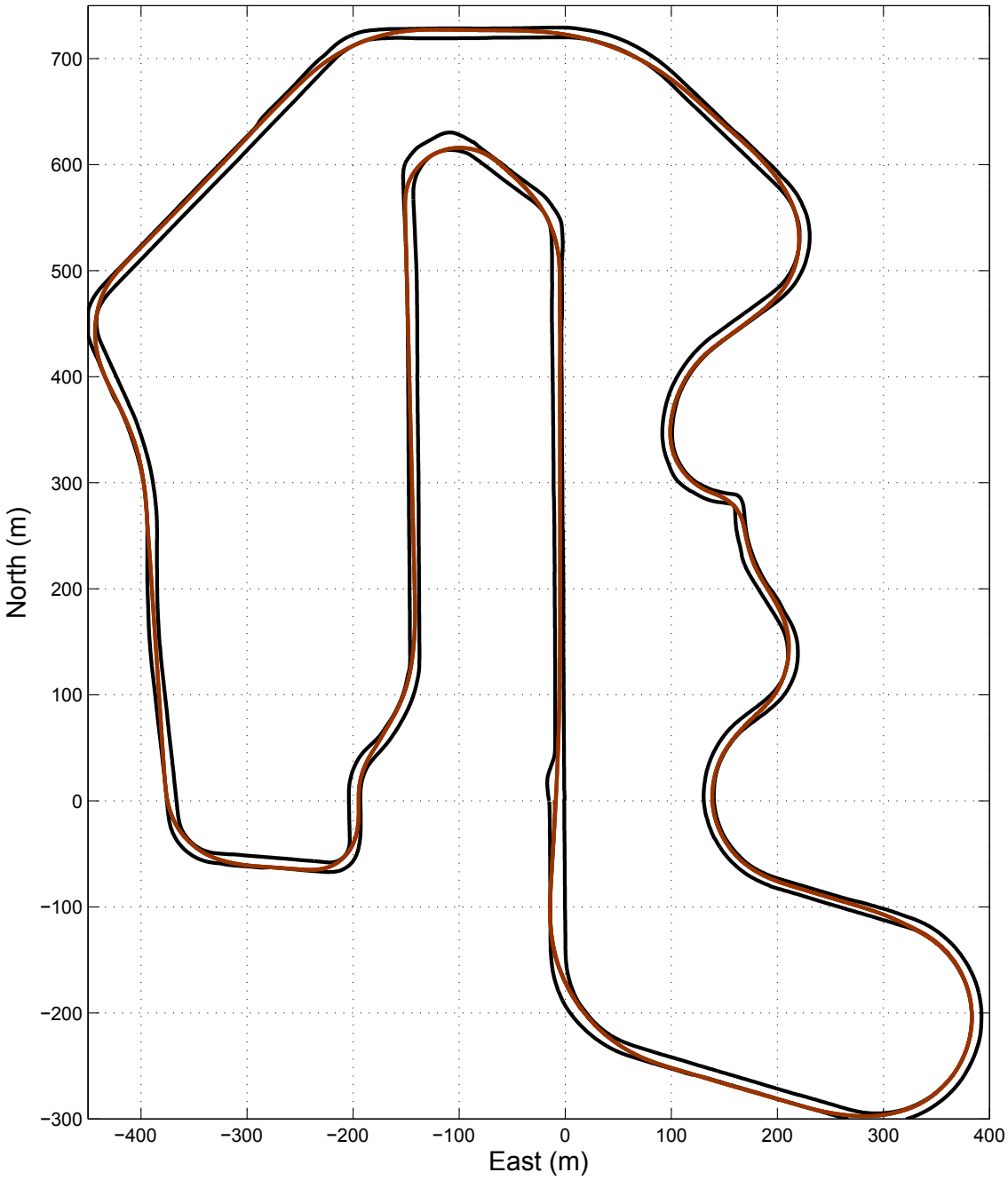


Figure 2.17: Overhead plot of path planner output for steering controller to follow.

## 2.7 Experimental Testing of Sideslip Feedback Controller

The steering feedback controller with sideslip-path tangency presented in Section 2.4 offers very low path tracking error, but at the expense of reduced stability margins. To test this controller safely at the limits of handling, experimental data was collected on a constant radius turn in an open parking lot at two constant speeds. The results of this test are shown in Fig. 2.18. As a baseline, the feedback controller with sideslip (2.12) from Section 2.4 is compared to the original lookahead feedback controller (2.9). The feedforward steering based on the nonlinear handling diagram (2.7) is used in both cases. For the case where the speed is 10 m/s, the resulting steady-state acceleration is 7 m/s$^2$, and both steering controllers maintain stability of the vehicle. In addition, incorporating sideslip-path tangency into the feedback control law results in significantly lower path tracking errors compared to the baseline lookahead feedback controller.

However, when the speed is increased to 13 m/s, the resulting steady-state lateral acceleration is 9 m/s$^2$, and the sideslip feedback controller becomes unstable, causing the vehicle to spin out spin, as demonstrated by the plots of yaw rate and sideslip. For the same conditions, the baseline lookahead controller remains stable. The issue with the sideslip feedback controller is shown in the plot of vehicle sideslip and heading error. As the vehicle heading error decreases and becomes unstable around $s = 170$ m, the feedback controller does not intervene because the vehicle's velocity vector remains tangent to the path (i.e. $\Delta\Psi + \beta = 0$). A counter steering action is finally provided at $s = 190$ m, but at this point the vehicle has already spun out.

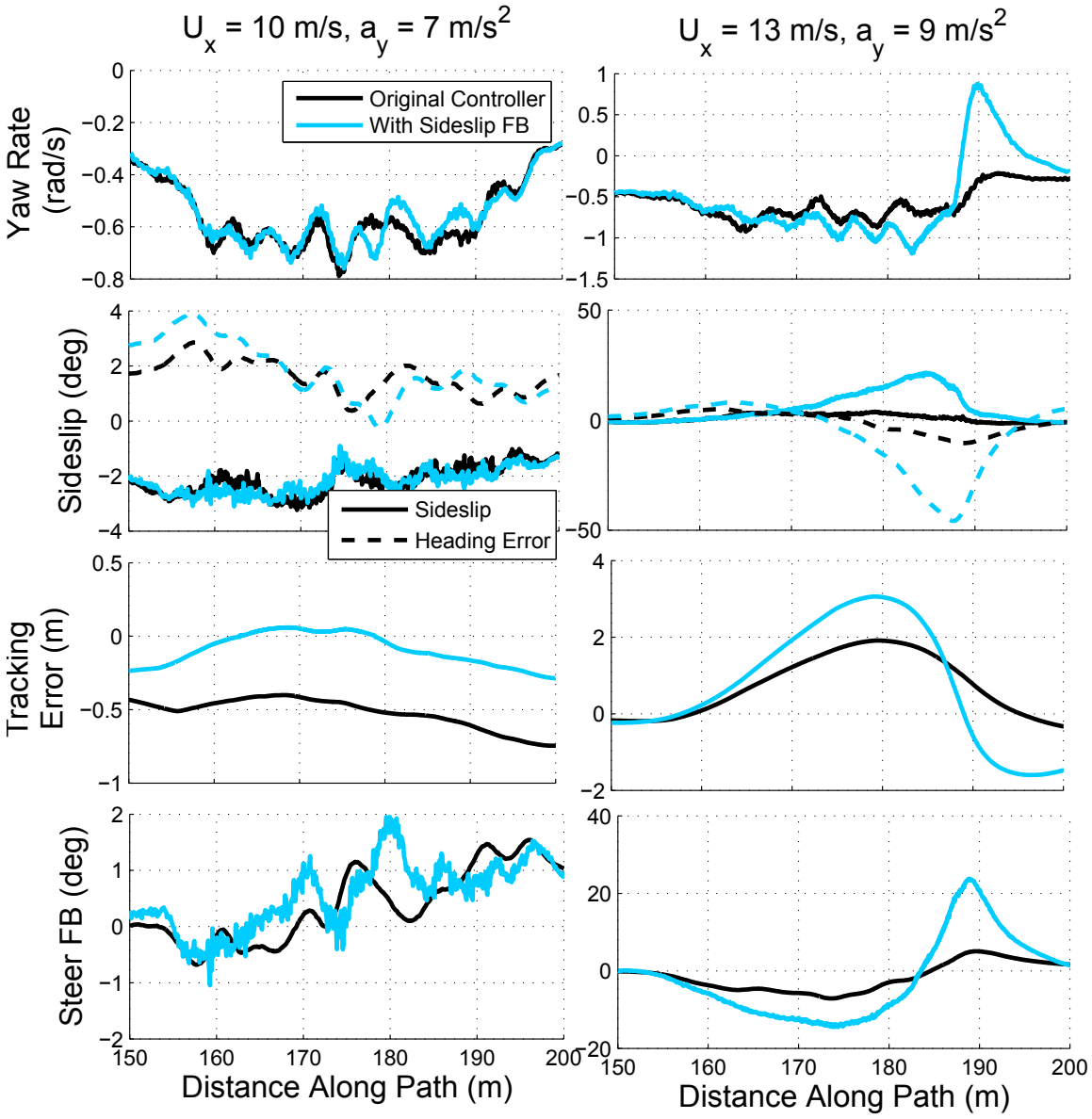


Figure 2.18: Parking lot test for constant radius turning at 10 m/s and 13 m/s. Resulting steady-state accelerations are 7 m/s$^2$ and 9 m/s$^2$. Steering feedback with sideslip is compared to original lookahead steering controller.

## 2.8 Experimental Data from Racetrack

Fig. 2.19 shows experimental data taken over a 3 km subset of the entire racetrack, excluding portions of the course where there is little vehicle cornering. The experimental data is plotted for two separate steering controllers. The first controller is

the baseline controller described in §2.2, with the steering feedback provided by the lookahead system and the feedforward steering given by the steady-state handling diagram. The second controller uses the same lookahead feedback, but incorporates the steady-state sideslip behavior into the feedforward steering to align the vehicle's velocity vector with the path heading (§2.5). Note that the controller where real-time sideslip was incorporated into the steering feedback (§2.4) was not tested, as there is a high likelihood of vehicle instability near the limits of handling given the results from the previous parking lot test. The same longitudinal controller is used for both cases in order to brake into the entry of turns and accelerate out of turn exits. For this lap, the peak longitudinal deceleration is -8 m/s$^2$ and peak lateral acceleration is 8 m/s$^2$.

The results in Fig. 2.19 show that lateral path deviation $e$ is reduced when incorporating steady-state sideslip in the feedforward control law. This confirms the expected results from the analysis performed in §2.3 and §2.5. Note that while the lateral path deviation is smaller, the levels of vehicle heading error $\Delta\Psi$ remain roughly the same. This matches the predicted result in Fig. 2.10. The steering controller will align the vehicle's velocity vector with the path, not the vehicle's heading, and due to a vehicle's tendency to develop a steady-state sideslip angle, a non-zero heading error $\Delta\Psi$ is in general necessary for zero steady-state lateral path deviation, as shown by the schematic in Fig. 2.8.

For a better comparison of the lateral path deviation, Fig. 2.20 shows histograms of the lateral tracking error $e$ for both controllers at three levels of peak lateral acceleration. The baseline controller keeps the vehicle within 0.5 meters on either side of the desired path, with a large amount of variation in the resulting histogram. An interesting observation is that the histogram is not symmetric. In general, the raceway has more left turns than right turns, and as Fig. 2.7 indicates, at high speeds the baseline controller will track toward the outside of the turn. This tendency is manifested experimentally in the asymmetric nature of the histograms. The histograms for the improved controller show a much tighter distribution on the lateral path tracking error. The path tracking error generally remains within 10-15 cm on either side of the lane boundary and contains less of a bias towards tracking on the outside of turns.

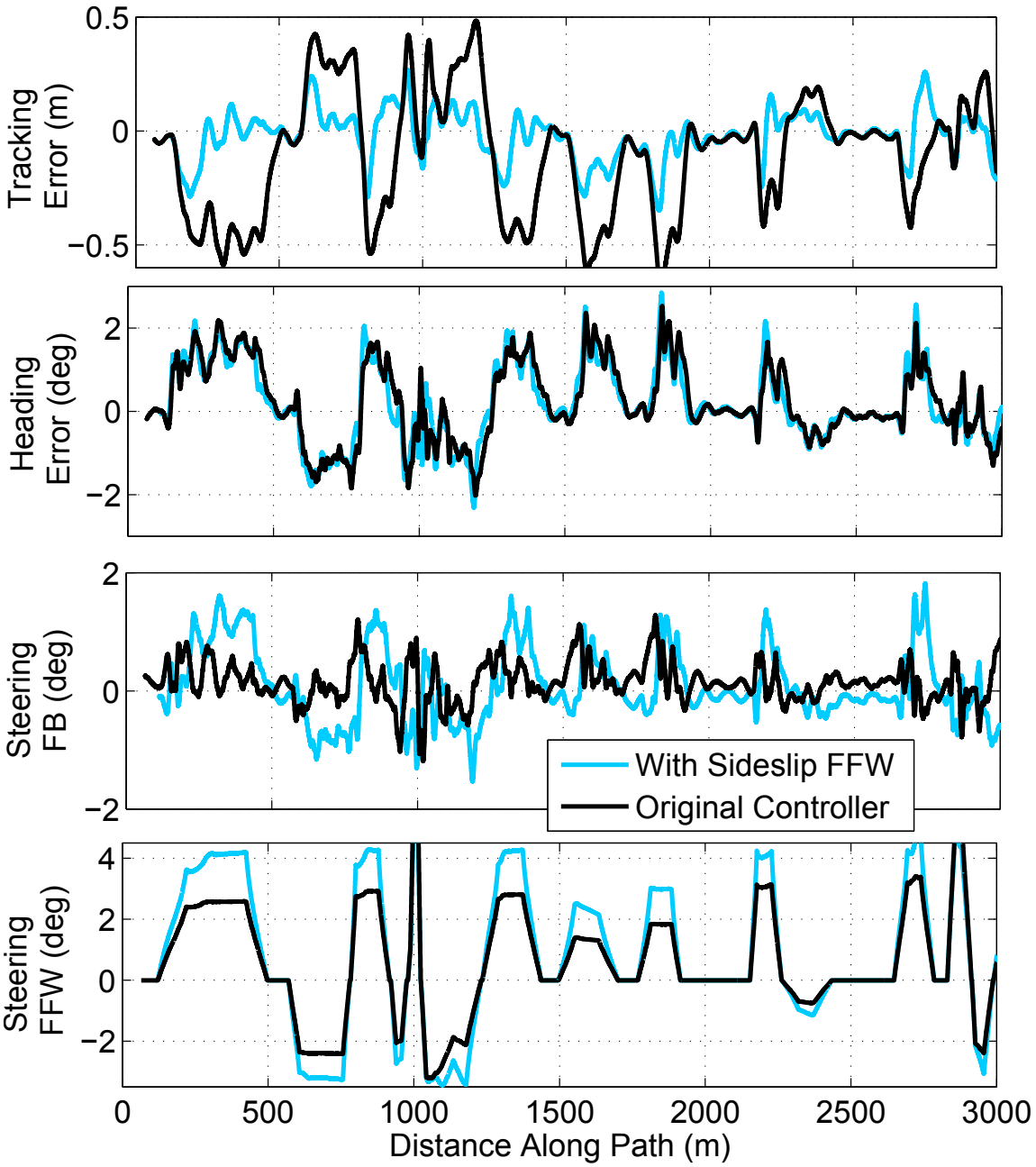


Figure 2.19: Experimental data with combined acceleration magnitude 8 $\mathrm{m/s^2}$ over a 3 km stretch of Thunderhill Raceway Park. Results are shown for both the baseline FB-FFW controller and the modified controller with sideslip tracking in the feedforward loop.

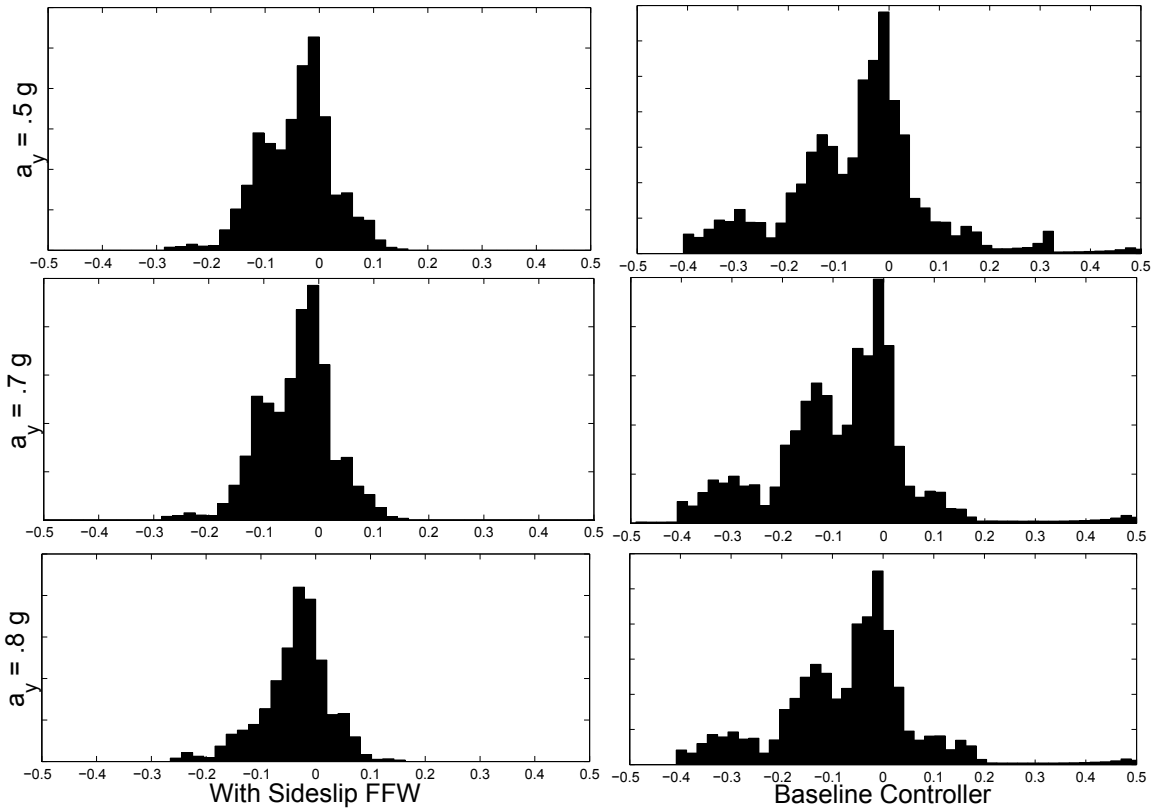


Figure 2.20: Histogram of path tracking error for six laps around the track. Left column represents performance of controller with feedforward sideslip tracking, and right column is baseline controller with feedforward from the steady-state handling diagram. Path tracking error is in meters.

As the lateral acceleration increases beyond 0.8 g and the vehicle approaches the handling limits, the steering controller remains stable and well-damped, although the tracking performance begins to degrade. Fig. 2.21 shows experimental data for a lap around Thunderhill Raceway park with peak lateral and longitudinal accelerations of 0.95 g. At several points along the track, the tracking error increases above 0.5 m, significantly higher than the levels of path deviation seen at 0.8 g of vehicle acceleration. The reasons for this are three-fold.

First, the sharp drop in front tire slip from $s = 300$ to 400 meters indicates that the lateral force demand for the front tires has exceeded the available friction at that region of the track. As a result, the vehicle understeers heavily and tracks well to the

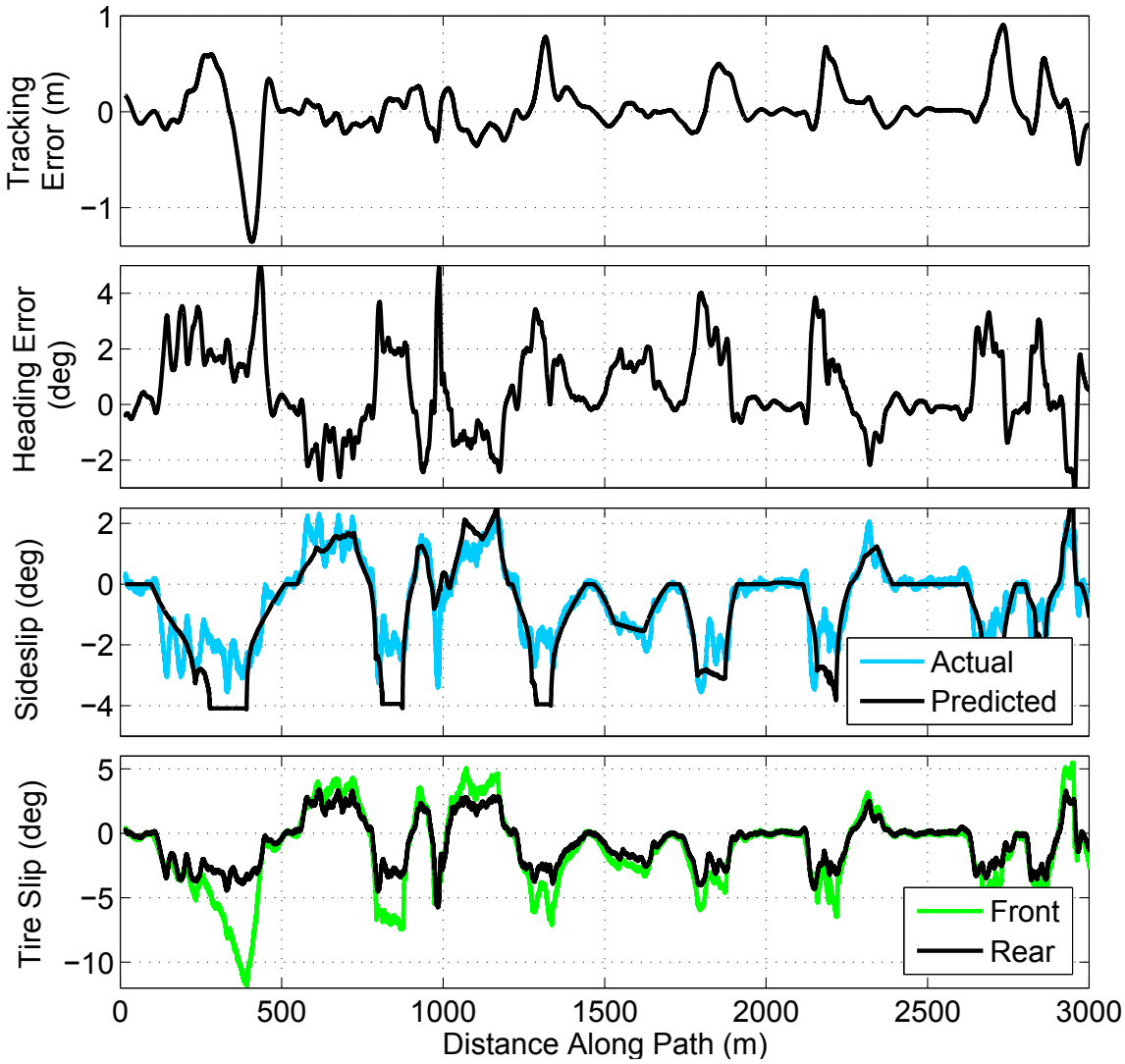


Figure 2.21: Experimental data with combined acceleration magnitude 9.5 m/s$^2$ over a 3 km stretch of Thunderhill Raceway Park.

outside of the left-hand turn, resulting in a large negative tracking error. At this point, there is nothing the steering controller can do to bring the vehicle back to the desired path, and the vehicle must alter its desired trajectory to become less aggressive, either by slowing down or reducing the turn curvature. Second, the steady-state feedforward controller requires an accurate estimate of the vehicle parameters in order to estimate the steady-state sideslip in (2.15a). From $s$ = 700-800, 1300-1400, 1800-1900, and

2200-2300 meters along the track, there are observable discrepancies between the predicted feedforward sideslip and the actual vehicle sideslip, as measured by the GPS-INS. This plant-model mismatch results in significant path tracking errors at those portions of the racing circuit. Finally, the feedforward model in (2.15a) assumes steady-state conditions. As the vehicle approaches the limits of handling, transient vehicle dynamics can result in larger path deviations as well.

The lap-to-lap learning algorithms presented in Chapter 4 focus on reducing the latter two sources of error by gradually learning a better feedforward model of the vehicle dynamics over time. For situations where a vehicle repeats a given trajectory multiple times, iterative learning control (ILC) is a promising technique for refining the feedforward input to improve the reference tracking performance of the controller. Additionally, online estimation approaches can also be used to gradually improve knowledge of difficult-to-measure parameters such as friction and vehicle cornering stiffness.

## 2.9 Conclusion

This chapter describes the design of a feedback-feedforward controller capable of path tracking at the limits of handling. Desirable path tracking behavior occurs when the vehicle sideslip is aligned with the desired path heading. However, directly incorporating this behavior into a feedback steering control law results in a closed-loop controller with poor stability margins. A better approach for combined path tracking and stability is to align the steady-state vehicle sideslip with the desired path heading through the feedforward steering command.

The benefit of the presented work is a controller design that provides low path tracking error over a large range of vehicle lateral accelerations. More importantly, the lateral path tracking improvement is achieved without sacrificing the robust stability properties of the lookahead steering feedback. Results from a histogram analysis quantitatively indicate that the improved feedforward command reduces lateral path deviation from the baseline controller by more than fifty percent. One potential drawback is that this feedforward approach is sensitive to vehicle model uncertainty,

especially at the physical limits of handling where transient dynamics become prevalent. Chapter 4 will present iterative learning control algorithms to improve the feedforward vehicle model and eliminate undesirable transient dynamics.

*Note: This chapter reuses material previously published by the author in [38].*

# Chapter 3

# Fast Generation Path Planning

Chapter 2 presented a steering controller capable of driving an aggressive trajectory from a high level trajectory planner. Since the steering controller only requires the curvature and velocity profile output, the details of the trajectory planner were not considered for the controller design. However, for the purpose of race car driving, the trajectory planning phase is just as important as the real-time path following. This chapter therefore provides a novel approach for planning the trajectory of an automated race vehicle. Because of the focus on racing, the primary consideration of the trajectory generation algorithm will be minimizing the vehicle's lap time.[1]

The problem of calculating the minimum lap time trajectory for a given vehicle and race track has been studied over the last several decades in the control, optimization, and vehicle dynamics communities. Early research by Hendrikx et al. [29] in 1996 used Pontryagin's minimum principle to derive coupled differential equations to solve for the minimum-time trajectory for a vehicle lane change maneuver. In 2000, Casanova [9] published a method to optimize both the path and speed profile for a fully nonlinear vehicle model using nonlinear programming (NLP). Kelly [42] further extended the results from Casanova by considering the physical effect of tire thermodynamics and applying more robust NLP solution methods such as Feasible Sequential Quadratic Programming. More recently, Perantoni and Limebeer

[1] Future modifications of the presented algorithm to passenger safety applications will be discussed in Chapter 6.

[59] showed that the computational expense could be significantly reduced by applying curvilinear track coordinates, non-stiff vehicle dynamics, and the use of smooth computer-generated analytic derivatives.

The primary focus of these NLP solutions was developing a simulation tool for Formula One race teams to analyze the lap time effects of subtle race car modifications. As a result, experimental validation was not considered, and high computation times were not a major issue. However, the development of autonomous vehicle technology has led to research on optimal path planning algorithms that can be validated on driverless cars. Theodosis and Gerdes published a gradient descent approach for determining time-optimal racing lines, with the racing line constrained to be composed of a fixed number of clothoid segments that are amenable for autonomous driving [74][2]. When driven autonomously using a closed-loop trajectory following controller [38][46], the resulting lap times were within one second of lap times from a professional race car driver. However, the gradient descent method, like other nonlinear programming techniques, took several hours of computation time to complete on a standard desktop computer.

Given the computational expense of performing nonlinear optimization, there has recently been an effort to find approximate methods that provide fast lap times. Timings and Cole [76] formulated the minimum lap time problem into a model predictive control (MPC) problem by linearizing the nonlinear vehicle dynamics at every time step and approximating the minimum-time objective by maximizing distance traveled along the path centerline. The resulting racing line for a 90 degree turn was simulated next to an NLP solution. Gerdts et al. [24] proposed a similar receding horizon approach, where distance along a reference path was maximized over a series of locally optimal optimization problems that were combined with continuity boundary conditions. One potential drawback of the model predictive control approach is that an optimization problem must be reformulated and solved at every time step, which can still be computationally expensive. For example, Timings and Cole reported a computation time of 900 milliseconds per 20 millisecond simulation step with the

[2]The curvature and speed profile used for the controller validation in Chapter 2 came from the racing trajectory generated by Theodosis and Gerdes.

CPLEX quadratic program solver on a desktop PC.

In summary, due to the primary objective of minimizing lap time while staying on the race track, constrained optimization is frequently used for planning a minimum-time trajectory. The most common method is nonlinear programming, which provides low lap-time trajectories, but at the expense of high computation times. The complex nature of the minimum-time vehicle optimization problem is two-fold. First, two sets of vehicle inputs, longitudinal and lateral, must be determined. Unfortunately, the lateral and longitudinal dynamics become highly coupled and nonlinear at the limits of handling. Second, directly minimizing lap time requires minimizing a non-convex cost function (§3.3). Not only are non-convex optimization problems more expensive to solve than their convex counterparts, but solution techniques are also only guaranteed to converge to a local minima. While computation time is not an issue for simulation tools, with the rapid progress in autonomous vehicle technology, there are significant benefits to a trajectory generation algorithm that can rapidly approximate the fastest racing trajectory for at least the next several turns of the race track (see §1.4).

This chapter therefore presents an experimentally validated algorithm that bypasses the complexity of minimum-time vehicle optimization in order to generate racing trajectories with low computational expense. To avoid the issue of coupled control inputs, the combined lateral/longitudinal optimal control problem is replaced by two sequential sub-problems that are solved iteratively. In the first sub-problem, the minimum-time longitudinal speed inputs are computed given a fixed vehicle path. In the second sub-problem, the vehicle path is updated given the fixed speed commands. To avoid minimizing the non-convex lap time cost function, the vehicle path is updated by solving a convex minimum curvature heuristic. The concept of solving a coupled, non-convex optimization via sequential approximations is not new, and the proposed approach is inspired by the methodology used in sequential convex programming (SCP) and the expectation/maximization (EM) algorithm [16][28].[3] The biggest potential drawback of these approaches is that the guarantee of convergence

[3]Sequential convex programming attempts to solve a nonconvex optimization problem by iteratively solving a convex approximation over a trust region that is modified after every iteration. Expectation/Maximization determines maximum likelihood estimates in statistical models with unobserved variables by repeatedly alternating between an expectation step and a maximization step.

to a globally optimal solution is lost, and the proposed method is therefore as sensitive to initial conditions as any nonlinear optimization.

The following section presents a mathematical framework for the trajectory generation problem and provides a linearized five-state model for the planar dynamics of a racecar following a set of speed and steering inputs on a fixed path. This model is identical to the model presented in Chapter 2, where the lateral vehicle dynamics are explicitly modeled but the longitudinal speed $U_x$ is treated as a time-varying parameter. Section 3.2 describes the method of finding the minimum-time speed inputs given a fixed path. While this sub-problem has been recently solved using convex optimization [48], a forward-backward integration scheme based on prior work [70] is used instead. Section 3.3 describes a method for updating the racing path given the fixed speed inputs using convex optimization, where the curvature norm of the driven path is explicitly minimized. The complete algorithm is outlined in §3.4, and a trajectory is generated for the Thunderhill Raceway circuit from Chapter 2. This trajectory is compared with a trajectory recorded from a professional human driver and the gradient descent trajectory from Theodosis [74]. In §3.5, the generated racing trajectory is validated experimentally in the autonomous Audi TTS testbed using the path-following controller from Chapter 2. The resulting lap time compares well with the lap times recorded for the gradient descent trajectory and the human driver. However, there are particular sections of the track where minimizing the driven curvature does not provide a fast trajectory. Section 3.7 therefore proposes a modified cost function for the path update step that also incorporates the benefit of reducing the length of the racing line. Section 3.8 concludes by discussing future implementation of the algorithm in a real-time path planner.

## 3.1 Path Description and Vehicle Model

Figure 3.1 describes the parameterization of the reference path that the vehicle will follow. The reference path is most intuitively described in Fig. 3.1(a) as a smooth curve of Cartesian East-North coordinates, with road boundaries represented by similar Cartesian curves. However, for the purposes of quickly generating a racing trajectory, it is more convenient to parameterize the reference path as a curvature profile $\kappa$ that is a function of distance along the path $s$ (Fig. 3.1(c)). Additionally, it is convenient to store the road boundary information as two functions $w_{\text{in}}(s)$ and $w_{\text{out}}(s)$, which correspond to the lateral distance from the path at $s$ to the inside and outside road boundaries, respectively (Fig. 3.1(b)). This maximum lateral distance representation will be useful when constraining the generated racing path to lie within the road boundaries. The transformation from curvilinear $s$, $\kappa$ coordinates to Cartesian coordinates $E$, $N$ are given by the Fresnel integrals:

$$E(s) = \int_0^s -\sin(\Psi_{\text{r}}(z))dz \tag{3.1a}$$

$$N(s) = \int_0^s \cos(\Psi_{\text{r}}(z))dz \tag{3.1b}$$

$$\Psi_{\text{r}}(s) = \int_0^s \kappa(z)dz \tag{3.1c}$$

where $\Psi_{\text{r}}(s)$ is the heading angle of the reference path and $z$ is a dummy variable.

With the reference path defined in terms of $s$ and $\kappa$, the next step is to define the dynamic model of the vehicle. For the purposes of trajectory generation, we assume the vehicle dynamics are given by the same planar bicycle model presented in Chapter 2, with yaw rate $r$ and sideslip $\beta$ states describing the lateral dynamics. Additionally, the vehicle's offset from the reference path are again given by the path lateral deviation state $e$ and path heading error state $\Delta\Psi$. Linearized equations of motion for all four states are given by (2.1). Recall that that while the vehicle longitudinal dynamics are not explicitly modeled, the bicycle model does allow for time-varying values of $U_x$. This is a reasonable approximation because the vehicle

model will be used for the lateral path update step, whereas the longitudinal dynamics will be treated separately in the velocity profile generation step.

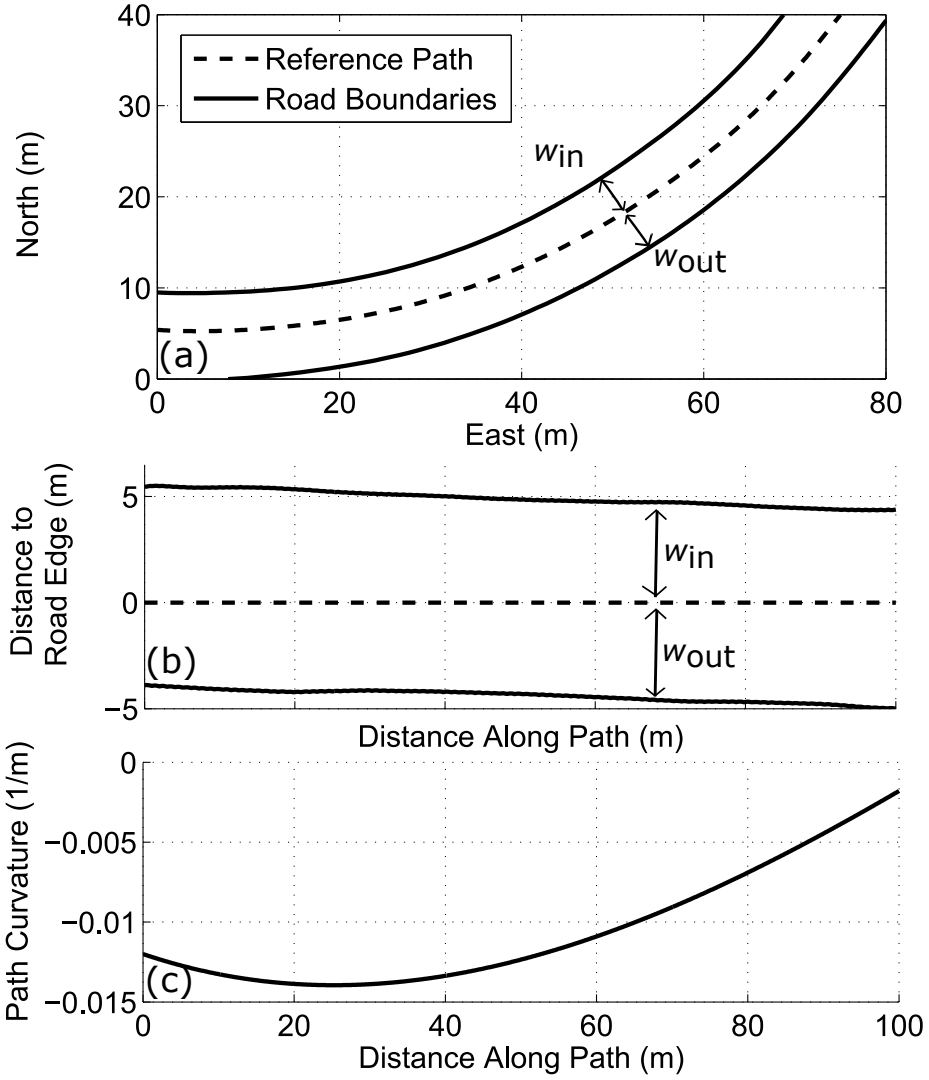


Figure 3.1: (a) View of a sample reference path and road boundaries, plotted in the East-North Cartesian frame (b) Lateral distance from path to inside road edge (positive) and outside road edge (negative) as a function of distance along path. (c) Curvature as a function of distance along path.

## 3.2 Velocity Profile Generation Given Fixed Reference Path

Given a fixed reference path described by $s$ and $\kappa$, the first algorithm step is to find the minimum-time speed profile the vehicle can achieve without exceeding the available tire-road friction. Finding the minimum-time speed profile for a fixed path has been explored extensively in the literature, and several viable solution methods exist. The algorithm presented in this chapter directly uses the "three-pass" approach described by Subosits and Gerdes [70], and originally inspired by work from Velenis and Tsiotras [79] and Griffiths [26]. Given the lumped front and rear tires from the bicycle model, the available longitudinal and lateral forces $F_{\mathrm{x}}$ and $F_{\mathrm{y}}$ at each wheel are constrained by the friction circle:

$$F_{\mathrm{xf}}^2 + F_{\mathrm{yf}}^2 \leq (\mu F_{\mathrm{zf}})^2 \tag{3.2a}$$

$$F_{\mathrm{xr}}^2 + F_{\mathrm{yr}}^2 \leq (\mu F_{\mathrm{zr}})^2 \tag{3.2b}$$

where $\mu$ is the tire-road friction coefficient and $F_{\mathrm{z}}$ is the available normal force. The first pass of the speed profile generation finds the maximum permissible vehicle speed given zero longitudinal force. For the simplified case where weight transfer and topography effects are neglected, this is given by:

$$U_x(s) = \sqrt{\frac{\mu g}{|\kappa(s)|}} \tag{3.3}$$

where the result in (3.3) is obtained by setting $F_{\mathrm{yf}} = \frac{mb}{a+b} U_x^2 \kappa$ and $F_{\mathrm{zf}} = \frac{mgb}{a+b}$. The results of this first pass for the sample curvature profile in Fig. 3.2(a) are shown in Fig. 3.2(b). The next step is a forward integration step, where the velocity of a given point is determined by the velocity of the previous point and the available longitudinal force $F_{x,\max}$ for acceleration. This available longitudinal force is calculated in [70] by accounting for the vehicle engine force limit and the lateral force demand on all tires

due to the road curvature:

$$U_x(s+\Delta s) = \sqrt{U_x^2(s) + 2\frac{F_{x,\text{accel,max}}}{m}\Delta s} \tag{3.4}$$

A key point of the forward integration step is that at every point, the value of $U_x(s)$ is compared to the corresponding value from (3.3), and the minimum value is taken. The result is shown graphically in Fig. 3.2(c). Finally, the backward integration step occurs, where the available longitudinal force for deceleration is again constrained by the lateral force demand on all tires:

$$U_x(s-\Delta s) = \sqrt{U_x^2(s) - 2\frac{F_{x,\text{decel,max}}}{m}\Delta s} \tag{3.5}$$

The value of $U_x(s)$ is then compared to the corresponding value from (3.4) for each point along the path, and the minimum value is chosen, resulting in the final velocity profile shown by the solid line in Fig. 3.2(d). While treatment of three-dimensional road effects are not described in this chapter, the method described in [70] and used for the experimental data collection determines the normal and lateral tire forces $F_{\text{z}}$ and $F_{\text{y}}$ at each point along the path by accounting for weight transfer and bank/grade of the road surface.

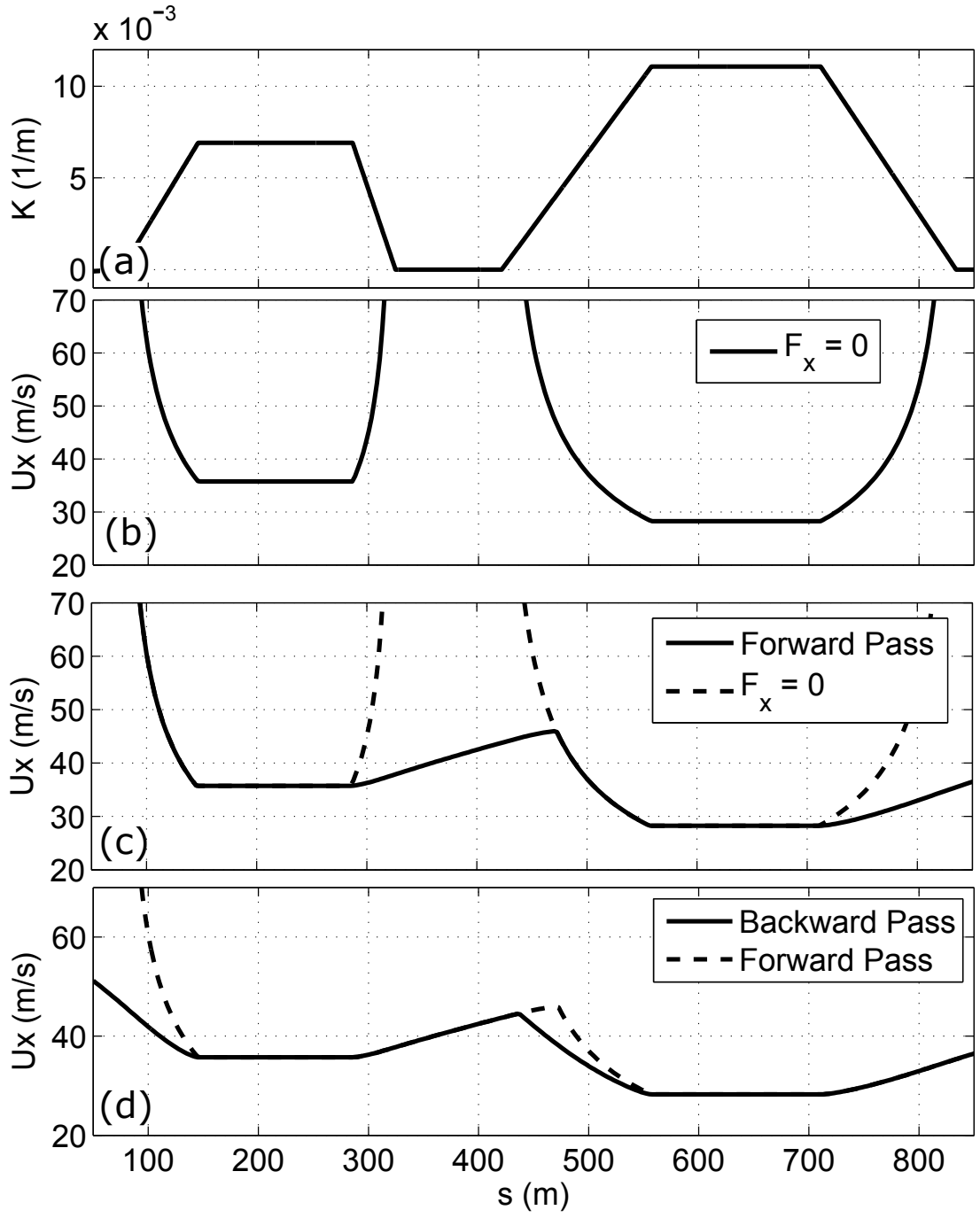


Figure 3.2: (a) Sample curvature profile. (b) Velocity profile given zero longitudinal force. (c) Velocity profile after forward pass. (d) Final velocity profile after backward pass.

## 3.3 Updating Path Given Fixed Velocity Profile

### 3.3.1 Overall Approach and Minimum Curvature Heuristic

The second step of the trajectory generation algorithm takes the original reference path $\kappa(s)$ and corresponding velocity profile $U_x(s)$ as inputs, and modifies the reference path to obtain a new, ideally faster, racing line. Sharp [65] suggests a general approach for modifying an initial path to obtain a faster lap time by taking the original path and velocity profile and incrementing the speed uniformly by a small, constant "learning rate." An optimization problem is then solved to find a new reference path and control inputs that allow the vehicle to drive at the higher speeds without driving off the road. If a crash is detected, the speed inputs are locally reduced around the crash site and the process is repeated.

However, one challenge with this approach is that it can take several hundred iterations of locally modifying the vehicle speed profile, detecting crashes, and modifying the reference path to converge to a fast lap time. An alternative approach is to modify the reference path in one step by solving a single optimization problem. The lap time $t$ for a given racing line is provided by the following equation:

$$t = \int_0^l \frac{ds}{U_x(s)} \tag{3.6}$$

Equation (3.6) implies that minimizing the vehicle lap time requires simultaneously minimizing the total path length $l$ while maximizing the vehicle's longitudinal velocity $U_x$. These are typically competing objectives, as lower curvature (i.e. higher radius) paths can result in longer path lengths but higher vehicle speeds when the lateral force capability of the tires is reached, as shown in (3.3). Since (3.6) is a nonconvex cost function in the optimization variables, time-intensive nonlinear programming is required to manage this curvature/distance trade-off and explicitly minimize the lap time.

The proposed approach is therefore to simplify the cost function by only minimizing the norm of the vehicle's driven curvature $\kappa(s)$ at each path modification step. Path curvature can be easily formulated as a convex function with respect to

the vehicle state vector $x$, enabling the path modification step to be easily solved by leveraging the computational speed of convex optimization.

However, minimizing curvature is not the same as minimizing lap time and provides no guarantee of finding the time-optimal solution. The proposed cost function relies on the hypothesis that a path with minimum curvature is a good approximation for the minimum-time racing line. Lowering the curvature of the racing line is more important than minimizing path length for most race courses, as the relatively narrow track width provides limited room to shorten the overall path length. Simulated and experimental results in §3.4 and §3.5 will validate this hypothesis by showing similar lap time performance when compared to a gradient descent method that directly minimizes lap time. However, a particular section of the race track where the minimum curvature solution shows poor performance will be discussed as well, and improved upon in §3.7.

### 3.3.2 Convex Problem Formulation

Formulating the path update step as a convex optimization problem requires an affine, discretized form of the bicycle model presented earlier. The equations of motion in (2.1) are already linearized, but the front and rear lateral tire forces become saturated as the vehicle drives near the limits of tire adhesion. The well-known brush tire model [58], also presented in Chapter 2, captures the effect of tire saturation.

$$F_{\mathrm{y}\diamond} = \begin{cases} -C_\diamond \tan\alpha_\diamond + \frac{C_\diamond^2}{3\mu F_{\mathrm{z}\diamond}} |\tan\alpha_\diamond| \tan\alpha_\diamond \\ \qquad -\frac{C_\diamond^3}{27\mu^2 F_{\mathrm{z}\diamond}^2} \tan^3\alpha_\diamond, & |\alpha_\diamond| < \arctan\left(\frac{3\mu F_{\mathrm{z}\diamond}}{C_\diamond}\right) \\ \\ -\mu F_{\mathrm{z}\diamond} \mathrm{sgn}\ \alpha_\diamond, & \text{otherwise} \end{cases} \tag{3.7}$$

where the symbol $\diamond \in [\mathrm{f}, \mathrm{r}]$ denotes the lumped front or rear tire, and $C_\diamond$ is the corresponding tire cornering stiffness. The linearized tire slip angles $\alpha_\mathrm{f}$ and $\alpha_\mathrm{r}$ are

functions of the vehicle lateral states and the steer angle input, $\delta$:

$$\alpha_{\mathrm{f}} = \beta + \frac{ar}{U_x} - \delta \tag{3.8a}$$

$$\alpha_{\mathrm{r}} = \beta - \frac{br}{U_x} \tag{3.8b}$$

The brush tire model in (3.7) can be linearized at every point along the reference path assuming steady state cornering conditions:

$$F_{\mathrm{y}\diamond} = \tilde{F}_{\mathrm{y}\diamond} - \tilde{C}_{\diamond}(\alpha_{\diamond} - \tilde{\alpha}_{\diamond}) \tag{3.9a}$$

$$\tilde{F}_{\mathrm{y}\diamond} = \frac{F_{z\diamond}}{g} U_x^2 \kappa \tag{3.9b}$$

with parameters $\tilde{F}_{\mathrm{y}}$, $\tilde{\alpha}$ and $\tilde{C}$ shown in Fig. 3.3.

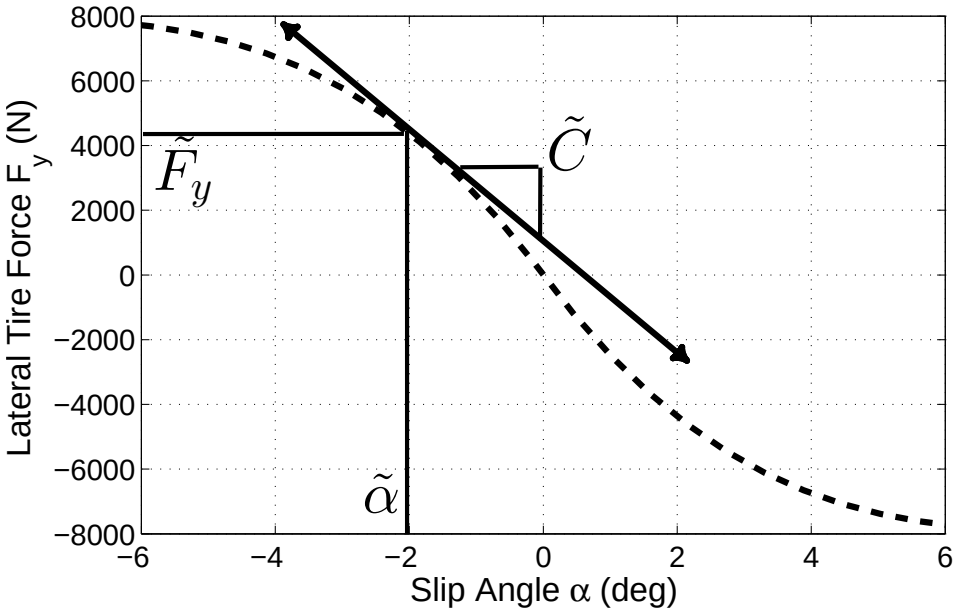


Figure 3.3: Nonlinear tire force curve given by Fiala model, along with affine tire model linearized at $\alpha = \tilde{\alpha}$.

The affine, continuous bicycle model with steering input $\delta$ is then written in state-space form as:

$$\dot{x}(t) = A(t)x + B(t)\delta + d(t) \tag{3.10a}$$

$$x = [e \;\; \Delta\Psi \;\; r \;\; \beta \;\; \Psi]^T \tag{3.10b}$$

where we have added a fifth state, vehicle heading angle $\Psi$, defined as the time integral of yaw rate $r$. This makes explicit computation of the minimum curvature path simpler. The state matrices $A(t)$, $B(t)$, and affine term $d(t)$ are given by:

$$A(t) = \begin{bmatrix} 0 & U_x(t) & 0 & U_x(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-a^2\tilde{C}_{\mathrm{f}}(t) - b^2\tilde{C}_{\mathrm{r}}(t)}{U_x(t)I_{\mathrm{z}}} & \frac{b\tilde{C}_{\mathrm{r}}(t) - a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & 0 \\ 0 & 0 & \frac{b\tilde{C}_{\mathrm{r}}(t) - a\tilde{C}_{\mathrm{f}}(t)}{mU_x^2(t)} - 1 & \frac{-\tilde{C}_{\mathrm{f}}(t) - \tilde{C}_{\mathrm{r}}(t)}{mU_x(t)} & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.11}$$

$$B(t) = [0 \;\; 0 \;\; \frac{a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} \;\; \frac{\tilde{C}_{\mathrm{f}}(t)}{mU_x(t)} \;\; 0]^T \tag{3.12}$$

$$d(t) = \begin{bmatrix} 0 \\ -\kappa(t)U_x(t) \\ \frac{a\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t) - b\tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t) + a\tilde{F}_{\mathrm{yf}}(t) - b\tilde{F}_{\mathrm{yr}}(t)}{I_z} \\ \frac{\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t) + \tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t) + \tilde{F}_{\mathrm{yf}}(t) + \tilde{F}_{\mathrm{yr}}(t)}{mU_x(t)} \\ 0 \end{bmatrix} \tag{3.13}$$

With the nonlinear model approximated as an affine, time-varying model, updating the path is accomplished by solving the following convex optimization problem:

$$\underset{\delta,\, x}{\text{minimize}} \quad \sum_k \left( \frac{\Psi_k - \Psi_{k-1}}{s_k - s_{k-1}} \right)^2 + \lambda(\delta_k - \delta_{k-1})^2 \tag{3.14a}$$

$$\text{subject to} \quad x_{k+1} = A_k x_k + B_k \delta_k + d_k \tag{3.14b}$$

$$w_k^{\text{out}} \leq e_k \leq w_k^{\text{in}} \tag{3.14c}$$

$$x_1 = x_T \tag{3.14d}$$

where $k = 1 \ldots T$ is the discretized time index, and $A_k$, $B_k$, and $d_k$ are discretized versions of the continuous state-space equations in (3.10). The objective function (3.14a) minimizes the curvature norm of the path driven by the vehicle, as path curvature is the derivative of the vehicle heading angle with respect to distance along the path $s$ (3.1c). To maintain convexity of the objective function, the term $s_k - s_{k-1}$ is a constant rather than a variable, and is updated for the next iteration after the optimization has been completed (see §3.4). Additionally, there is a regularization term with weight $\lambda$ added in the cost function to ensure a smooth steering profile for experimental implementation.

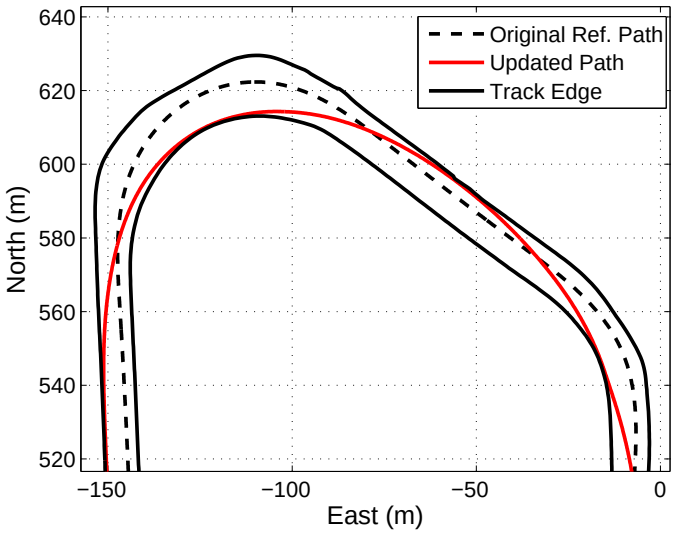


Figure 3.4: Path update for an example turn.

The equality constraint (3.14b) ensures the vehicle follows the affine lateral dynamics. The inequality constraint (3.14c) allows the vehicle to deviate laterally from the reference path to find a new path with lower curvature, but only up to the road edges. Finally, the equality constraint (3.14d) is required for complete racing circuits to ensure the generated racing line is a continuous loop. The results of running the optimization are shown for an example turn in Fig. 3.4. The reference path starts out at the road centerline, and the optimization finds a modified path that uses all the available width of the road to lower the path curvature. Note that the available road widths $w_{\text{in}}$ and $w_{\text{out}}$ have an offset built in to account for the width of the vehicle.

## 3.4 Algorithm Implementation and Simulated Results

### 3.4.1 Algorithm Implementation

The final algorithm for iteratively generating a vehicle racing trajectory is described in Fig. 3.5. The input to the algorithm is any initial path through the racing circuit, parameterized in terms of distance along the path $s$, path curvature $\kappa(s)$, and the lane edge distances $w_{\text{in}}(s)$ and $w_{\text{out}}(s)$ from Fig. 3.1.

1: **procedure** GENERATETRAJECTORY($s^0, \kappa^0, w_{\text{in}}^0, w_{\text{out}}^0$)
2: path $\leftarrow (s^0, \kappa^0, w_{\text{in}}^0, w_{\text{out}}^0)$
3: **while** $\Delta t^\star > \epsilon$ **do**
4: $U_x \leftarrow$ calculateSpeedProfile(path)
5: path $\leftarrow$ minimizeCurvature($U_x$, path)
6: $t^\star \leftarrow$ calculateLapTime($U_x$, path)
7: **end while**
8: **return** path, $U_x$
9: **end procedure**

Figure 3.5: Iterative algorithm for fast generation of vehicle trajectories. Each iteration consists of a sequential two-step approach where the velocity profile is generated given a fixed path and then the path is updated based on the solution from a convex optimization problem.

Given the initial path, the minimum-time speed profile $U_x(s)$ is calculated using the approach from §3.2. Next, the path is modified by solving the minimum curvature convex optimization problem (3.14).

The optimization only solves explicitly for the steering input $\delta^\star$ and resulting vehicle lateral states $x^\star$ at every time step. Included within $x^\star$ is the optimal vehicle heading $\Psi^\star$ and lateral deviation $e^\star$ from the initial path. To obtain the new path in terms of $s$ and $\kappa$, the East-North coordinates ($E_k$, $N_k$) of the updated vehicle path are updated as follows:

$$E_k \leftarrow E_k - e_k^\star \cos(\Psi_{\mathrm{r},k}) \tag{3.15a}$$

$$N_k \leftarrow N_k - e_k^\star \sin(\Psi_{\mathrm{r},k}) \tag{3.15b}$$

where $\Psi_\mathrm{r}$ is the path heading angle of the original path. Next, the new path is given by the following numerical approximation:

$$s_k = s_{k-1} + \sqrt{(E_k - E_{k-1})^2 + (N_k - N_{k-1})^2} \tag{3.16a}$$

$$\kappa_k = \frac{\Psi_k^\star - \Psi_{k-1}^\star}{s_k - s_{k-1}} \tag{3.16b}$$

Notice that (3.16) accounts for the change in the total path length that occurs when the vehicle deviates from the original path. In addition to $s$ and $\kappa$, the lateral distances to the track edges $w_\mathrm{in}$ and $w_\mathrm{out}$ are different for the new path as well, and are recomputed using the Cartesian coordinates for the inner and outer track edges and ($E_k$, $N_k$). The two-step procedure is iterated until the improvement in lap time $\Delta t^\star$ over the prior iteration is less than a small positive constant $\epsilon$.

### 3.4.2 Algorithm Validation

The proposed algorithm is tested by analyzing the lap time performance on the same racing circuit and Audi TTS experimental test vehicle described in Chapter 2. The

Table 3.1: Optimization Parameters

| Parameter | Symbol | Value | Units |
|---|---|---|---|
| Regularization Parameter | $\lambda$ | 1 | $1/\mathrm{m}^2$ |
| Stop Criterion | $\epsilon$ | 0.1 | s |
| Vehicle mass | $m$ | 1500 | kg |
| Yaw Inertia | $I_z$ | 2250 | $\mathrm{kg} \cdot \mathrm{m}^2$ |
| Front axle to CG | $a$ | 1.04 | m |
| Rear axle to CG | $b$ | 1.42 | m |
| Front cornering stiffness | $\mathrm{C_f}$ | 160 | $\mathrm{kN} \cdot \mathrm{rad}^{-1}$ |
| Rear cornering stiffness | $\mathrm{C_r}$ | 180 | $\mathrm{kN} \cdot \mathrm{rad}^{-1}$ |
| Friction Coefficient | $\mu$ | 0.95 | – |
| Path Discretization | $\Delta s$ | 2.75 | $m$ |
| Optimization Time Steps | $T$ | 1843 | - |
| Max Engine Force | - | 3750 | N |

vehicle parameters used for the lap time optimization are shown along with the optimization parameters in Table 3.1. The initial path is obtained by collecting GPS data of the inner and outer track edges and estimating the $(s, \kappa, w_{\mathrm{in}}, w_{\mathrm{out}})$ parametrization of the track centerline via a separate curvature estimation subproblem similar to the one proposed in [59]. The algorithm is implemented in MATLAB, with the minimum curvature optimization problem (3.14) solved using the CVX software package [25] and the speed profile generation problem solved using a library from Subosits and Gerdes [70].

### 3.4.3 Comparison with Other Methods

The generated racing path after five iterations is shown in Fig. 3.6. To validate the proposed algorithm, the racing line is compared with results from a nonlinear gradient descent algorithm implemented by Theodosis and Gerdes [74] and an experimental trajectory recorded from a professional racecar driver in the experimental testbed vehicle. While time-intensive to compute, experimental lap times from the gradient descent trajectory are within one second of lap times from professional racecar drivers.

To better visualize the differences between all three racing lines, Fig. 3.7 shows the lateral deviation from the track centerline as a function of distance along the centerline

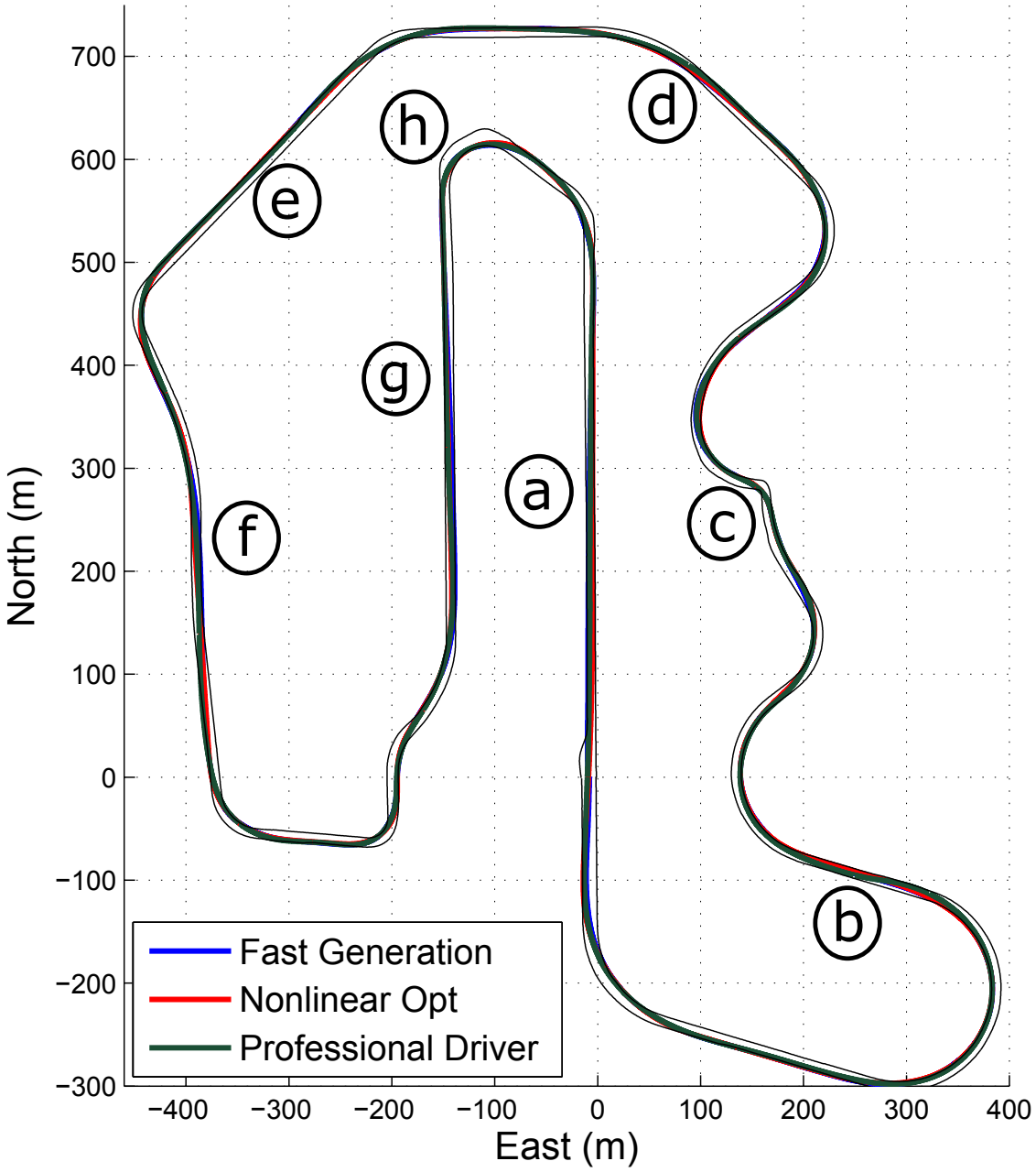


Figure 3.6: Overhead view of Thunderhill Raceway park along with generated path from algorithm. Car drives in alphabetical direction around the closed circuit. Labeled regions a-h are locations of discrepancies between the two-step algorithm solution and comparison solutions.

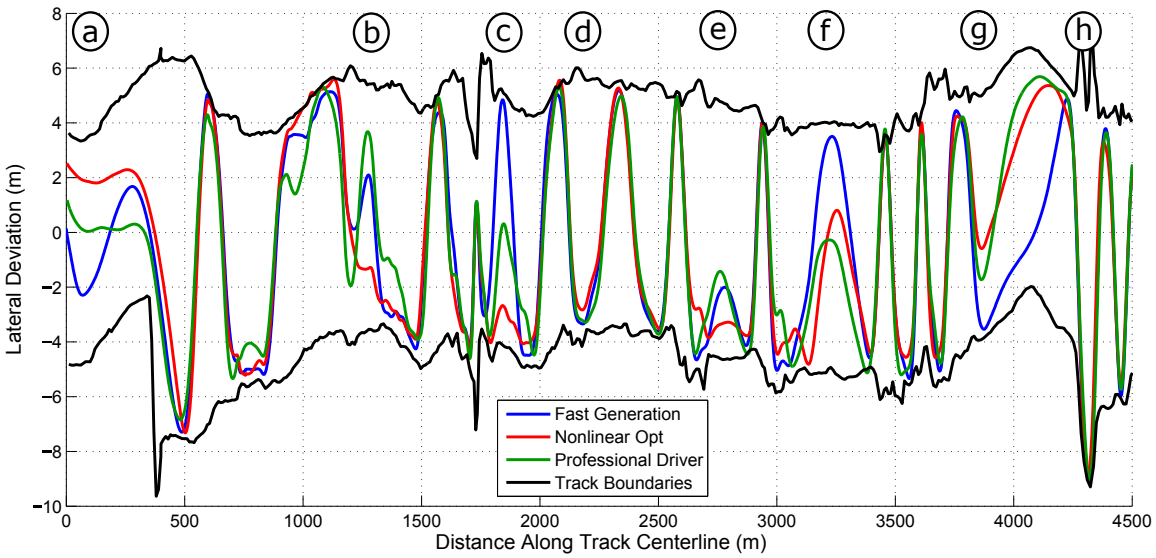


Figure 3.7: Lateral path deviation of racing line from track centerline as a function of distance along the centerline. Note that upper and lower bounds on $e$ are not always symmetric due to the initial centerline being a smooth approximation. Results are compared with racing line from a nonlinear gradient descent algorithm and experimental data recorded from a professional racecar driver.

for all three trajectories. The left and right track boundaries $w_{\text{in}}$ and $w_{\text{out}}$ are plotted as well. The two-step iterative algorithm provides a racing line that is qualitatively similar to the gradient descent and human driver racing lines. In particular, all three solutions succeed at utilizing the available track width whenever possible, and strike similar apex points for each of the circuit's 15 corners.

However, there are several locations on the track where there is a significant discrepancy (on the order of several meters) between the two-step algorithm's trajectory and the other comparison trajectories. These locations of interest are labeled (a) through (h) in Fig. 3.6. Note that sections (a), (e), (f), and (g) all occur on large, relatively straight portions of the racing circuit. In these straight sections, the path curvature is relatively low and differences in lateral deviation from the track centerline have a relatively small effect on the lap time performance.

Of more significant interest are the sections labeled (b), (c), (d), and (h), which all occur at turning regions of the track. These regions are plotted in Fig. 3.8 and

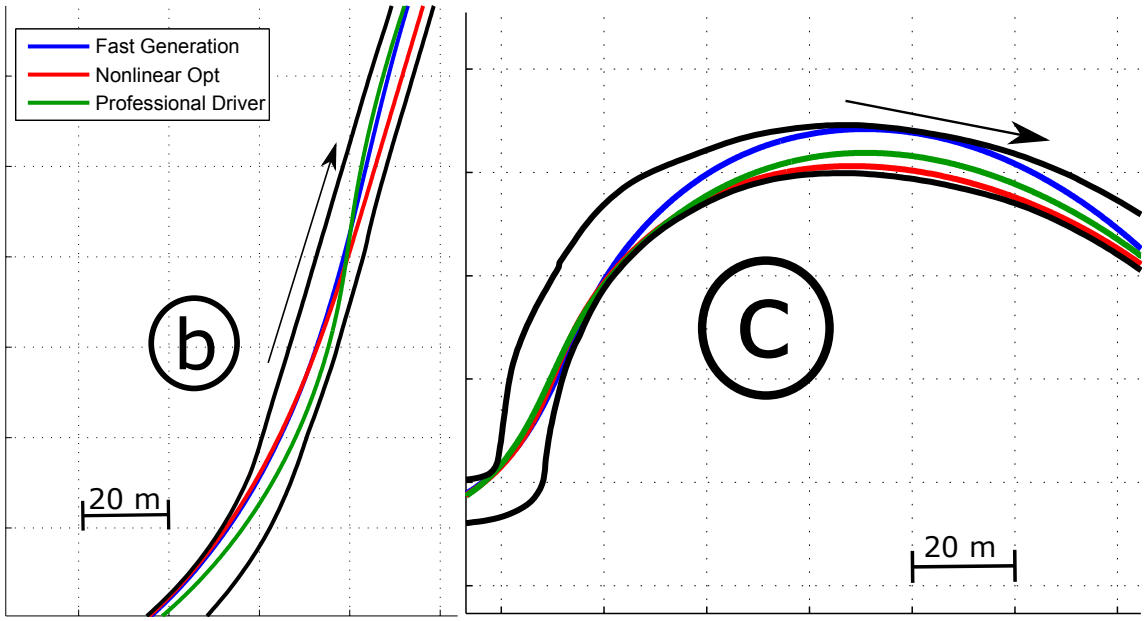


Figure 3.8: Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Car drives in direction of labeled arrow.

Fig. 3.9 for zoomed-in portions of the race track. While it is difficult to analyze a single turn of the track in isolation, discrepancies can arise between the two-step fast generation method and the gradient descent as the latter method trades off between minimizing curvature and distance traveled. As a result, the gradient descent method finds regions where it may be beneficial to use less of the available road width in order to reduce the total distance traveled. In region (b), for example, the fast generation algorithm exits the turn and gradually approaches the left side in order to create space for the upcoming right-handed corner. The nonlinear optimization, however, chooses a racing line that stays toward the right side of the track. In this case, the behavior of the human driver more closely matches that of the two-step fast generation algorithm.

The human driver also drives closer to the fast generation solution in (h), while the gradient descent algorithm picks a path that exits the corner with a larger radius. In section (c), the gradient descent algorithm again prefers a shorter racing line that remains close the the inside edge of the track, while the two-step algorithm drives

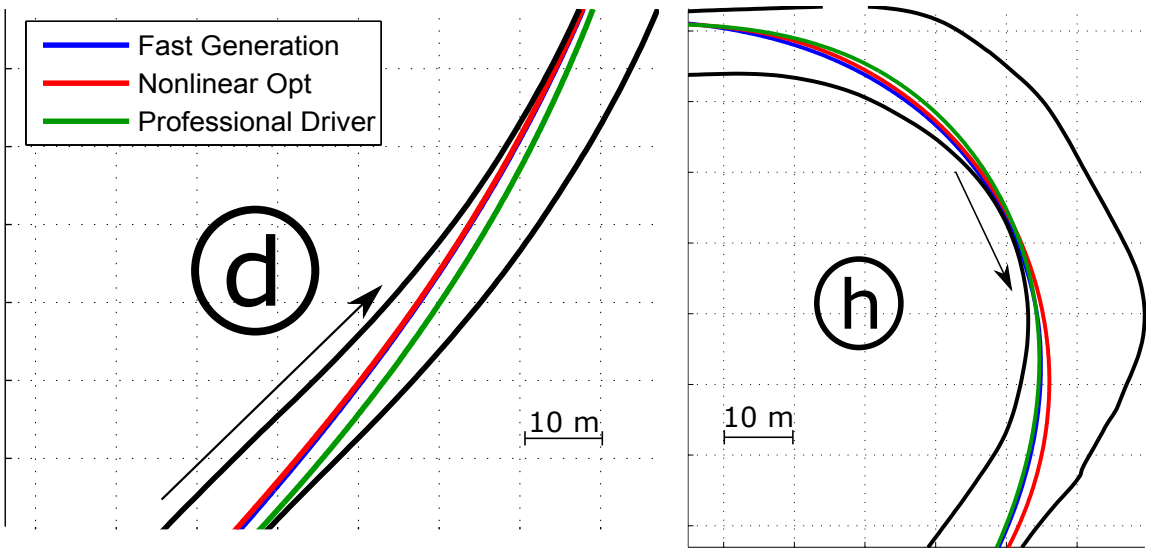


Figure 3.9: Racing lines from the two-step fast generation approach, nonlinear gradient descent algorithm, and experimental data taken from professional driver. Car drives in direction of labeled arrow.

all the way to the outside edge while making the right-handed turn. Interestingly, the human driver stays closer to the middle of the road, but more closely follows the behavior of the gradient descent algorithm. There are also regions of the track where the computational algorithms pick a similar path that differs from the human driver, such as region (d).

### 3.4.4 Lap Time Convergence and Predicted Lap Time

Fig. 3.10 shows the predicted lap time for each iteration of the fast generation algorithm, with step 0 corresponding to the initial race track centerline. The lap time was estimated after each iteration by numerically simulating a vehicle following the desired path and velocity profile using a closed-loop controller. The equations of motion for the simulation were the nonlinear versions of (2.1) with tire forces given by the brush tire model in (3.7).

Fig. 3.10 shows that the predicted lap time converges monotonically over four or five iterations, with significant improvements over the centerline trajectory occuring

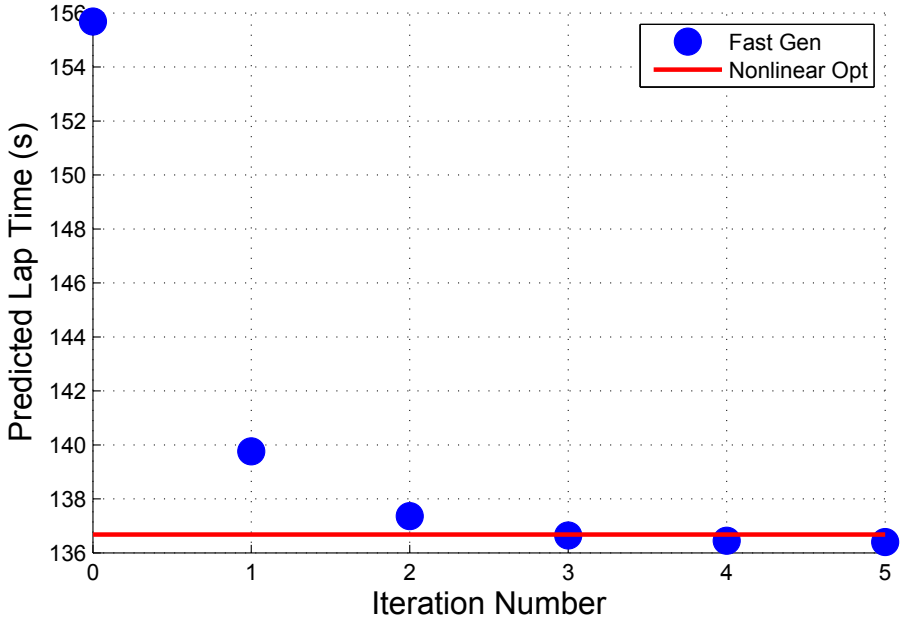


Figure 3.10: Lap time as a function of iteration number for the two-step fast trajectory generation method. Final lap time is comparable to that achieved with the nonlinear gradient descent approach. Iteration zero corresponds to the lap time for driving the track centerline.

over the first two iterations. The predicted minimum lap time of 136.4 seconds is similar to the predicted lap time of 136.7 seconds from the nonlinear gradient descent, although in reality, the experimental lap time will depend significantly on unmodeled effects such as powertrain dynamics.

The final curvature and velocity profile for the two-step method is compared with the equivalent profiles for the gradient descent algorithm in Fig. 3.11. Notice that the piecewise linear $\kappa(s)$ for the gradient descent is due to the clothoid constraint imposed by [74] for ease of autonomous path following.

In general, the curvature and velocity profiles are very similar, although the fast generation algorithm results in a velocity profile with slightly lower cornering speeds but slightly higher top speeds. The predicted time difference between a car driving both trajectories is shown in Fig. 3.11(a), with a negative value corresponding the two-step algorithm being ahead.

Notice that in region (c), the trajectory from the two-step algorithm performs poorly, losing almost a half second of time to the nonlinear optimization over just 150 meters. Referring back to Fig. 3.8, region (c) is a sweeping right-hand turn that comes after a very tight left-hand turn on the track, and both the human driver and nonlinear optimization prefer to take a shorter path and stay closer to the inside edge of the track. While this results in a higher curvature for the first turn, the shorter path on the second turn creates a net time advantage. As a result, the gradient descent optimization retains an overall time advantage from this turn on until losing ground in section (g), where the two-step method catches up and ultimately completes the lap with a 0.3 second time advantage.

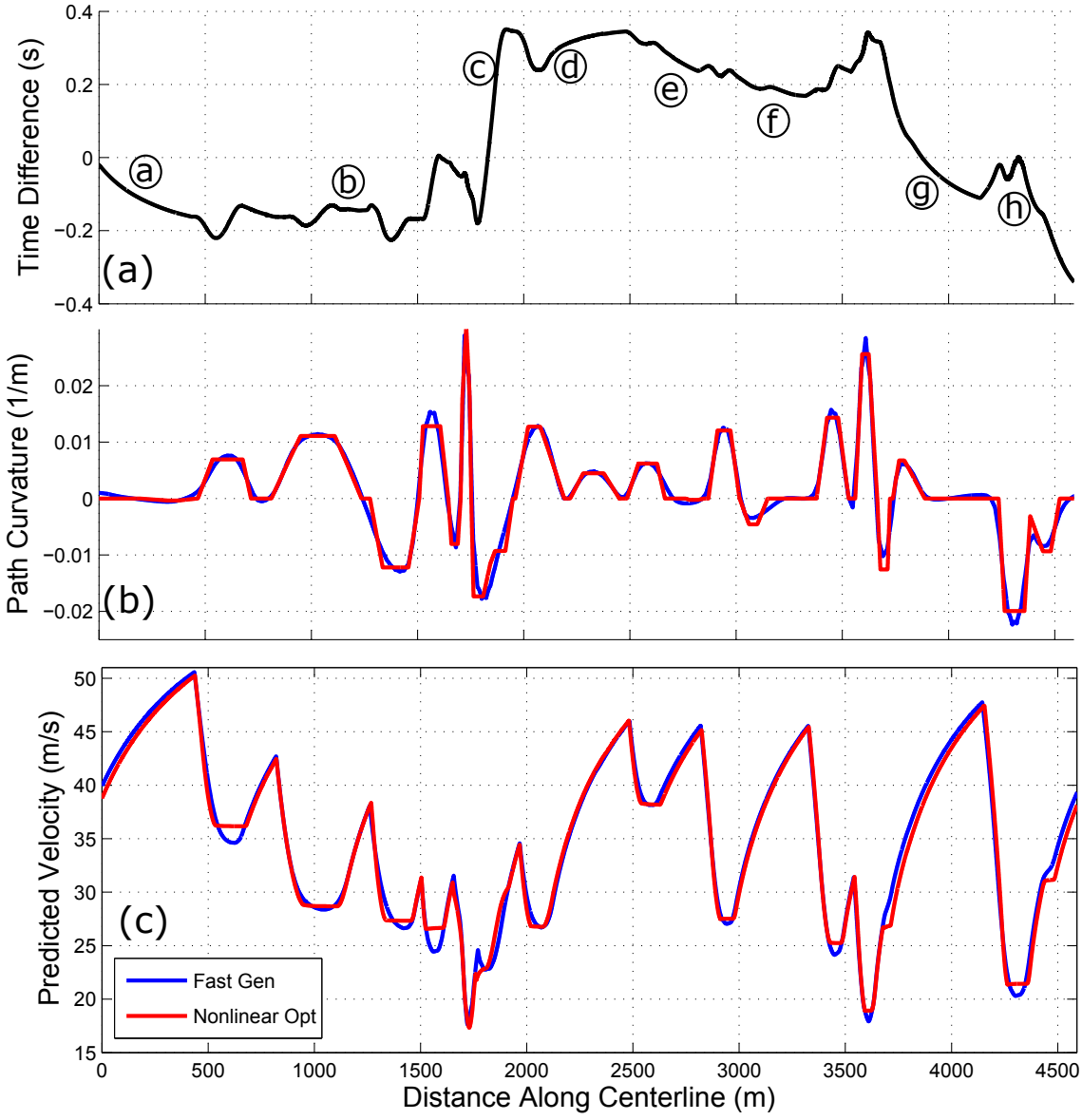


Figure 3.11: (a) Predicted time difference between a car driving both trajectories, with a negative value corresponding the two-step algorithm being ahead. (b) Curvature profile $\kappa(s)$ plotted vs. distance along the path $s$. (c) Velocity profile $U_x(s)$ plotted vs. distance along the path $s$ for the two-step method and nonlinear gradient descent method.

## 3.5 Experimental Setup

While the two-step algorithm works well in simulation, the most critical validation step is to have an autonomous race car drive the generated trajectory. This was accomplished by collecting experimental data on the Audi TTS.

The experimental controller setup is shown in Fig. 3.12 and is very similar to that presented in Chapter 2. The main two differences in the controller are highlighted in red. Instead of using the piecewise linear clothoid curvature profile from Theodosis and Gerdes [74], the trajectory from the presented algorithm is applied. This trajectory is represented mathematically as an array of discrete points. This point cloud is relatively dense, with each point spaced about 25 cm apart over the entire path.

Since the trajectory is now a set of discrete points rather than a piecewise linear $\kappa(s)$ function, the localization algorithm cannot rely on Newton-Raphson gradient descent. Instead, a simple search algorithm iterates through the point cloud and finds the closest two point to the vehicle's center of gravity. Bisection is applied to the find the closest distance between the vehicle and the line connecting these two points on the point cloud. To save the expense of searching the entire point cloud on every iteration, the localization starts the search algorithm where the last iteration terminated and searches only a small region of the entire map.

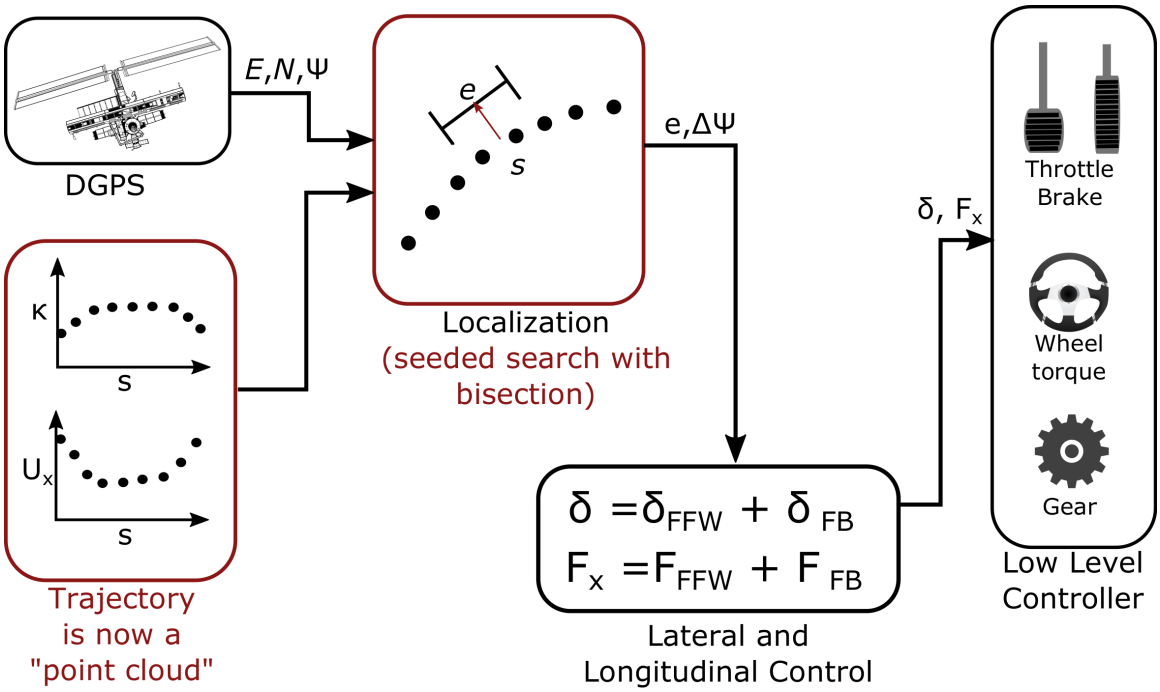


Figure 3.12: Diagram of controller setup.

## 3.6 Experimental Results

The resulting experimental lap time for the iterative two-step algorithm was 138.6 seconds, about 0.6 seconds faster than the experimental lap time for the gradient descent algorithm (139.2 seconds). For safety reasons, the trajectories were generated using a conservative peak road friction value of $\mu = 0.90$, resulting in peak lateral and longitudinal accelerations of $0.9g$. In reality, the true friction value of the road is closer to $\mu = 0.95$. As a result, both of these lap times are slightly slower than the fastest lap time recorded by a professional race car driver (137.7 seconds) as well as the predicted lap times from Section 3.4. A summary of all lap times is provided in Table 5.1.

Plots of the experimental data are shown in Fig. 3.13, with a negative time difference again corresponding to the two-step algorithm being ahead The experimental data generally matches the simulated results in Fig. 3.11. The simulation predicted the trajectory from the iterative two-step algorithm would be 0.3 seconds shorter

| | Simulation | Experiment |
|---|---|---|
| Fast Generation | 136.4 | 138.6 |
| Gradient Descent | 136.8 | 139.2 |
| Human Driver | N/A | 137.7 |

Table 3.2: Lap Times in Seconds

than that of the nonlinear algorithm, compared to the 0.6 second speed advantage observed experimentally. The simulation also predicted a relative time advantage for the two-step algorithm from sections (a) to (c) and from (e) to (h), a trend seen in the experimental data as well. Additionally, the two-step algorithm has relatively poor performance from sections (c) to (d) when compared to the nonlinear algorithm. This experimental result confirms that the minimum curvature heuristic works well for the majority of the track, but relatively poorly on particular "irregular" sequences of turns such as region (c). Section 3.7 will show the benefit of adding a term in the convex optimization cost function to consider distance traveled in addition to path curvature.

One reason for minor variations between the simulated and experimental time difference plots is variation in speed tracking. The speed tracking error for both racing lines is shown in Fig. 3.13(c). Interestingly, while the same speed tracking controller was used to test both racing lines, the controller has slightly better speed tracking performance when running the trajectory from the nonlinear optimization.

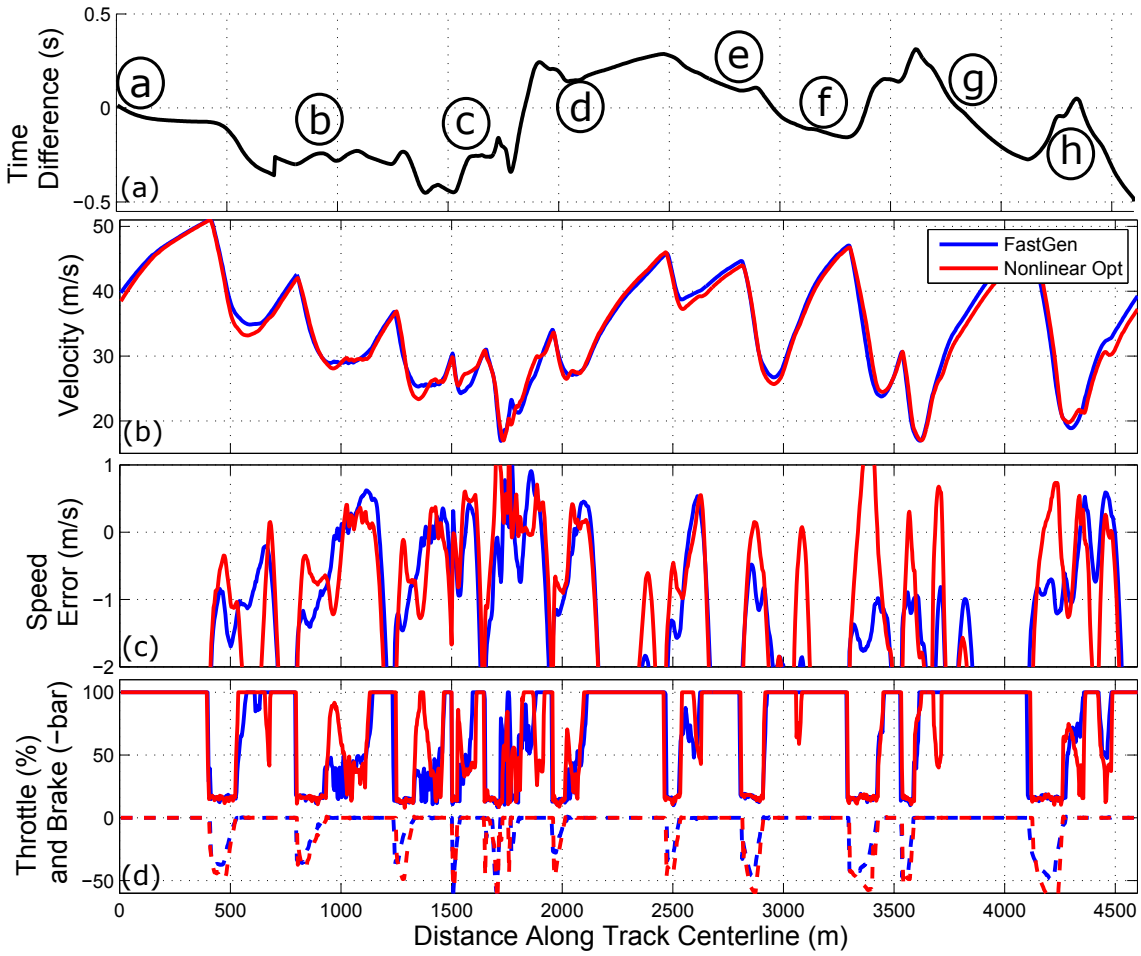


Figure 3.13: Experimental data for an autonomous vehicle driving the trajectories provided by the two-step fast generation and gradient descent algorithms.(a) Relative time difference between vehicle driving both trajectories, with a negative time difference corresponding to the two-step algorithm being ahead. (b) Actual recorded velocity of vehicle. (c) Difference between actual and desired speed. Large negative values outside plotting range occur on straight sections of the track where the vehicle is limited by engine power and speed tracking error is poorly defined. (d) Throttle percentage and brake pressure, with brake pressures shown as negative.

## 3.7 Incorporating the Effect of Distance Traveled

The performance of the presented trajectory generation approach can be further improved by modifying the cost function (3.14) of the path update step. Instead of only minimizing curvature, a new convex cost function is proposed that minimizes a weighted sum of the distance traveled and the path curvature. While this also does not directly minimize lap time, it does account for the incremental benefit provided by a shorter path, which may be helpful in improving the performance of the algorithm on particular turns such as region (c).

A convex term for the total distance traveled by the race vehicle is derived as follows. The instantaneous rate of progress of the vehicle along a fixed path is given by:

$$\dot{s} = \frac{U_x}{1 - ke} \cos(\Delta\Psi + \beta) \tag{3.17}$$

and the time to travel between two fixed points on the nominal path is given by:

$$t_k = \frac{s_k - s_{k-1}}{\dot{s}_k} \tag{3.18}$$

Since the path discretization $s_k - s_{k-1}$ and speed profile $U_x$ are fixed during the path update step , minimizing path length is equivalent to minimizing the sum over all $t_k$:

$$\sum_k \frac{s_k - s_{k-1}}{U_{xk}} \left( \frac{1 - \kappa_k e_k}{\cos(\Delta\Psi_k + \beta_k)} \right) \tag{3.19}$$

Taking the Taylor series expansion in the optimization variables $(e, \Delta\Psi, \beta)$ for the path update step yields a convex approximation for minimizing the distance traveled by the vehicle:

$$\sum_k \frac{\Delta s_k}{U_{xk}} \left( -\kappa_k e_k + (\Delta\Psi_k + \beta_k)^2 \right) \tag{3.20}$$

The first term $-\kappa_k e_k$ in (3.20) rewards moving to the inside of curved sections, and the second term $(\Delta\Psi_k + \beta_k)^2$ represents the additional distance traveled when driving at an angle to the original path.

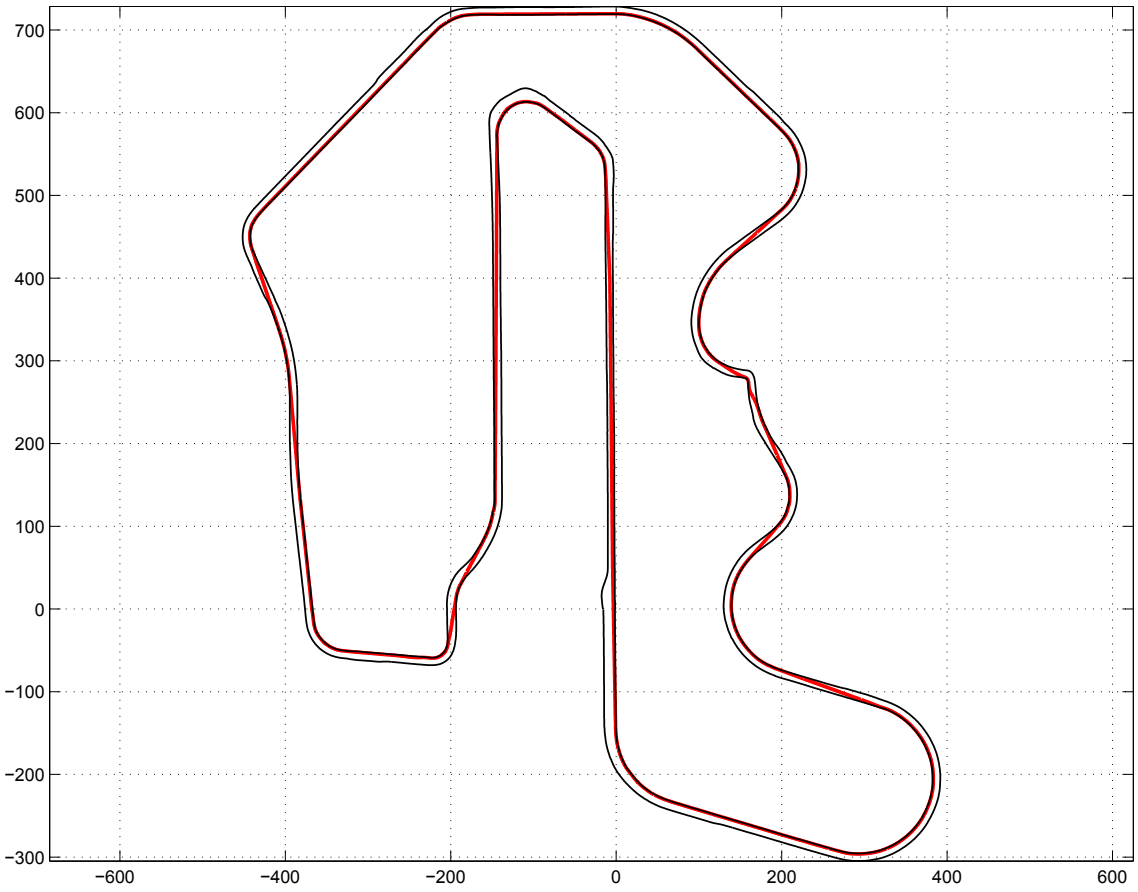


Figure 3.14: Minimum distance path around Thunder Hill

### 3.7.1 Balancing Minimum Distance and Curvature

Minimizing (3.20) subject to the vehicle dynamics and road boundary constraints from (3.14) results in the path shown in Fig. 3.14. As expected, the resulting path simply clings to the inner edge of the track wherever possible. A simple glance shows that entirely minimizing distance traveled generates an extremely poor racing line. In fact, the resulting simulated lap times are over ten seconds slower than the minimum distance solution!

There is clearly a need for a balance between minimizing distance and minimizing curvature, weighted more significantly towards the latter. There have been several prior attempts in the literature to perform this balance. Braghin [3] proposed finding

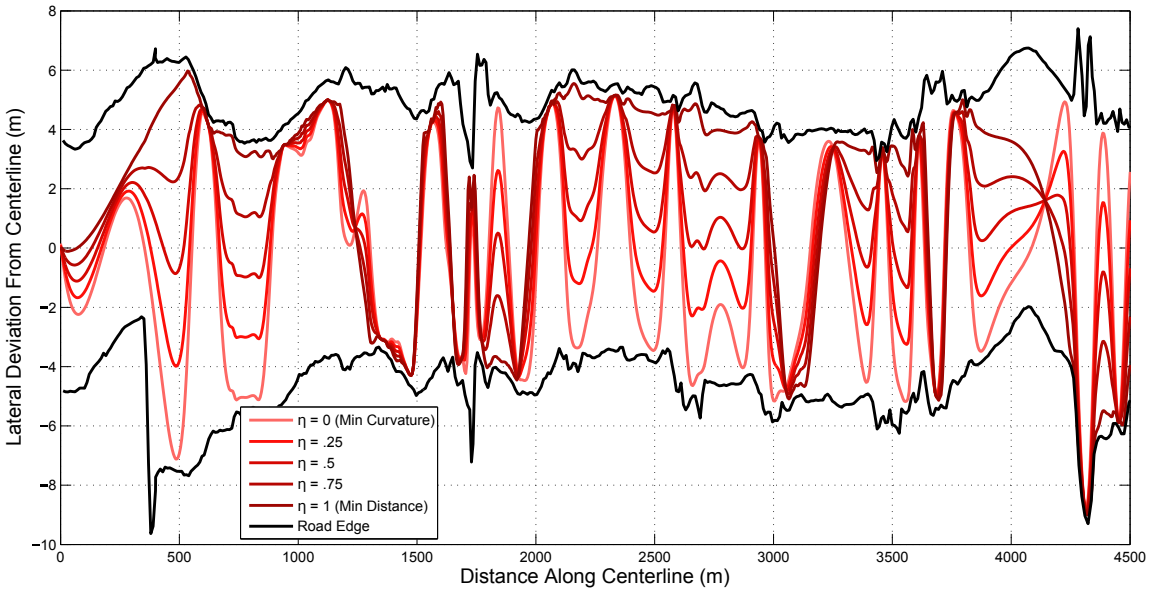


Figure 3.15: A family of racing lines generated from linear combinations of minimum distance and minimum curvature racing lines, with weighting parameter $\eta$.

the minimum distance and minimum curvature paths through a purely geometric optimization, with no vehicle dynamics considered. Weighted combinations of these basis paths were then generated and tested using a simple point mass model. For example, let $e_D(s)$ denote the lateral offsets from the track centerline corresponding to the minimum distance path. Then let $e_\kappa(s)$ be the corresponding offsets for the minimum curvature path. A proposed racing line is then defined by:

$$e = (1 - \eta)e_\kappa + \eta e_D \tag{3.21}$$

Where $0 \leq \eta \leq 1$ is the weighting parameter. Figure 3.15 demonstrates this concept for the Thunderhill Racing Circuit. Racing lines generated with $\eta$ close to 0 are very similar to the minimum curvature path, while candidate solutions with $\eta$ close to 1 approximate the minimum distance path.

The issue with this approach is that a single weighting parameter $\eta$ does not adequately balance the tradeoff between minimizing distance and minimizing curvature.

On most sections of a given track, the primary objective is simply to minimize curvature. However, there are typically a small minority of turns (for example, region ⓒ in our case) where minimizing distance traveled is relatively important. A better method is therefore to have the weighting parameter $\eta$ be a function $\eta(s)$ that varies along the track. This is exactly the approach suggested by Cardamone et al. [7], who analyzed Braghin's approach over a number of tracks and found that if only a single weighting factor was chosen, the optimal solution was frequently just $\eta = 0$ (i.e. the minimum curvature path).

Cardamone et al. suggested determining $\eta(s)$ by applying a genetic algorithm to choose a different weighting parameter between every intersection of the minimum curvature and minimum distance paths. Similar approaches were also presented by Gadola et al. [23] and Mühlmeier and Müller [53]. While this provided improved lap times over the minimum curvature solution, genetic algorithms are typically slow computationally, as every candidate solution $\eta(s)$ must be simulated. The computational benefit over nonlinear programming that directly minimizes lap time is therefore debatable.

### 3.7.2 Using Human Driver Data to Obtain Optimization Weights

Using professional driver data as a baseline offers a simpler method to determine parts of the track where minimizing distance is important. Fig. 3.16(a) shows ten laps of human driver data on the Thunderhill racetrack overlaid onto Figure 3.15. Fig. 3.16(b) shows the resulting weighting function $\eta(s)$ obtained by averaging the human data and finding the relative distance from the human centerline deviation $e_H(s)$ to the minimum curvature and minimum distance solutions:

$$\eta(s) = \frac{|e_H(s) - e_\kappa(s)|}{|e_H(s) - e_D(s)|} \tag{3.22}$$

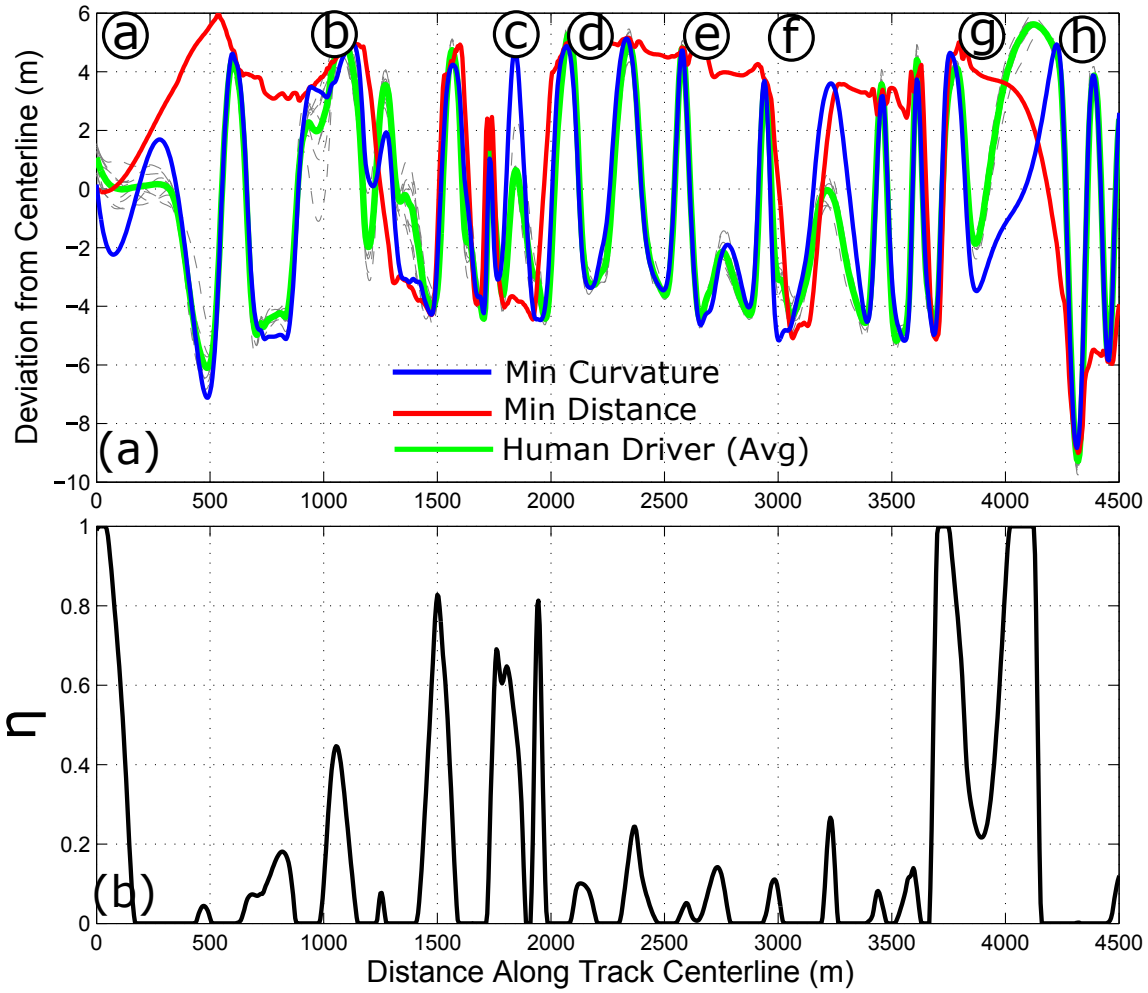


Figure 3.16: (a) Ten laps of professional human driver data overlaid over the minimum distance and minimum curvature solutions $e_D(s)$ and $e_\kappa(s)$. The average of the human driver data is shown in green, and the individual datasets are shown in light grey. (b) Values of $\eta(s)$ from human data, with low pass filter applied to eliminate rapid changes. Values are also limited to range from 0 to 1.

This definition is only relevant if the human racing line is bounded by the minimum distance and minimum curvature racing lines. Since this is frequently not the case in Fig. 3.16(a), $\eta(s)$ is set to 1 if $e_H(s) < e_D(s) < e_\kappa(s)$ or 0 if $e_H(s) < e_\kappa(s) < e_D(s)$. Furthermore, a low-pass filter is also applied to Fig. 3.16(a) to eliminate rapid changes in $\eta(s)$.

Fig. 3.16 shows that there are several regions of the track where the human drives closer to to the minimum distance racing line (i.e. locations where $\eta$ is significantly greater than 0). Most of these are trivial, occurring in locations where the minimum distance and minimum curvature racing lines are relatively close. However, in region (c), the professional human driver is on average about halfway between the minimum distance and minimum curvature solutions, an interesting result. Furthermore, on straight sections of the track such as (a) and (g), the human driver appears to be seeking a minimum distance path as well.

### 3.7.3 Combined Cost Function and Simulated Results

Using the information from Fig. 3.16, the combined cost function for the path update step in (3.14) is given by:

$$\underset{\delta,\, e,\, \Psi,\, \Delta\Psi,\, \beta}{\text{minimize}} \quad \sum_k \left( \frac{\Psi_k - \Psi_{k-1}}{s_k - s_{k-1}} \right)^2 + \lambda \sum_k \frac{\eta_k \Delta s_k}{U_{xk}} \left( -\kappa_k e_k + (\Delta\Psi_k + \beta_k)^2 \right) \tag{3.23}$$

The first summation in (3.23) is the same curvature minimization term, while the second summation represents the distance minimization term. The weights $\eta_k$ from Fig. 3.16(a) are used to determine how much of the minimum distance term to use at each point along the track. This approach is fundamentally different from the methods in [3] and [7]. The prior approaches search a space of solutions to find the best linear combination of the *pre-calculated* minimum distance and minimum curvature racing lines, with weights given by a constant $\eta$ or function $\eta(s)$. The presented approach performs a single optimization for each path update step, with the *optimization weights* given by $\eta_k$. As a result, there is an additional tunable parameter $\lambda$ in (3.23) with units of seconds, that ensures units of both summation terms are the same.

There are several benefits of the proposed approach. First, since we obtained $\eta(s)$ from human driver data, simply applying a linear combination with weights $\eta(s)$ would trivially give back the averaged professional driver's racing line. More

importantly, however, using pure linear combinations of two precomputed solutions provides excessive restrictions on the resulting racing lines. This is because the resulting solutions can never explore regions that are not contained within the minimum curvature and minimum distance solutions. Furthermore, there is no guarantee the resulting path will be experimentally drivable. Even if the minimum curvature and distance paths come from an optimization with vehicle dynamics constraints, if the weighting function $\eta(s)$ is not sufficiently smooth, there will be discontinuities in the resulting curvature profile. By using $\eta(s)$ to instead guide the optimization, the resulting curvature profile will always be smooth.

Fig. 3.17 shows simulation results when the fast generation method is run for five iterations. The simulation compares the racing lines generated by the curvature minimization cost function and the combined curvature-distance cost functions. Unsurprisingly, the primary difference occurs at region Ⓒ. With the combined cost function, the resulting racing line takes a slightly higher curvature turn on the initial left turn. While this initially loses time, the resulting solution can minimize distance traveled on the next turn, resulting in an overall time advantage of 0.2 seconds.

The minimum distance, minimum curvature, and combined racing lines at region Ⓒ are shown in Fig. 3.18. The combined solution follows the minimum curvature solution more closely for the initial left-hand turn, but the minimum distance solution more closely for the second right-hand turn. Additionally, the racing line from the dual cost function is not always inside the minimum distance and minimum curvature racing lines. This demonstrates the advantage of a weighted cost function as opposed to a linear combination of pre-calculated solutions.

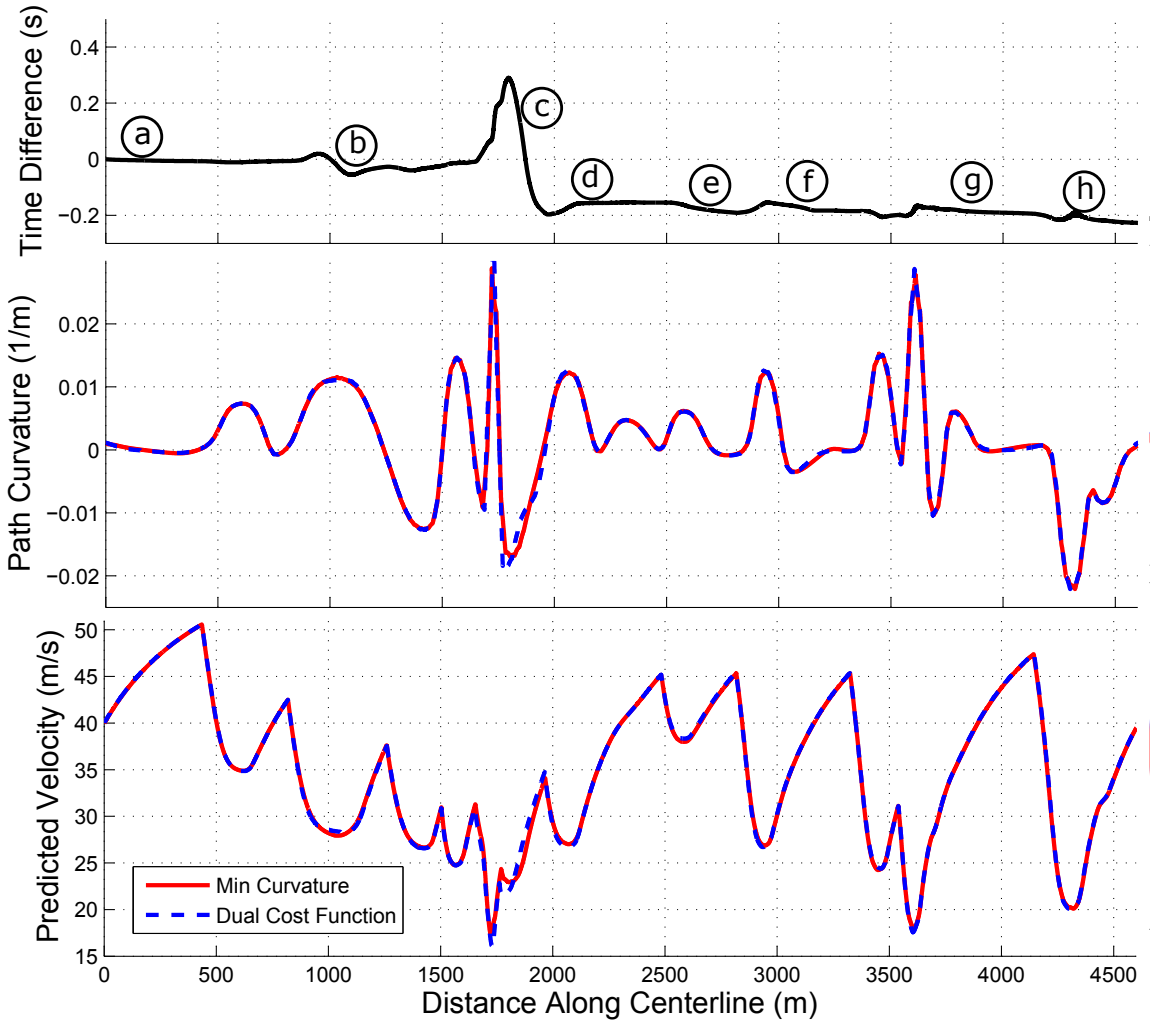


Figure 3.17: Simulation results comparing minimum curvature cost function with weighted distance/curvature cost function ($\lambda = 0.05$ sec). (a) Time difference between two solutions as a function of distance along centerline, with a negative time difference corresponding to the weighted optimization being ahead. (b) Path curvature. (c) Simulated velocities.

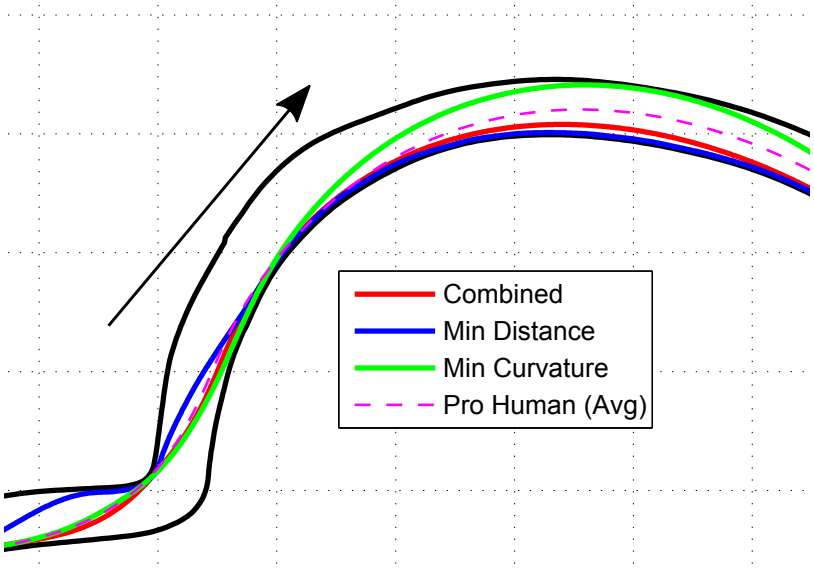


Figure 3.18: Racing lines for minimum curvature, minimum distance, and combined cost functions around region c. Notice that with the combined cost function, the resulting racing line is not bounded by the minimum curvature and minimum distance solutions. Averaged pro human racing line is shown as well.

## 3.8 Discussion and Future Work

The 0.6 second experimental difference between the fast generation and gradient descent lap times is less than 0.5% of the total lap time, making the two algorithms comparable in terms of performance, even without the combined cost function. The primary benefit of the proposed algorithm is not improved lap time performance over the nonlinear algorithm but rather a radical improvement in computational simplicity and speed. Each two-step iteration of the full course takes only 26 seconds on an Intel i7 processor, whereas the nonlinear algorithm from [74] typically runs over the course of several hours on the same machine. The most significant computational expense for the proposed algorithm is solving the convex curvature minimization problem for all 1843 discrete time steps $T$ over the 4.5 km racing circuit.

This computational efficiency will enable future work to incorporate the trajectory modification algorithm as an online "preview" path planner, which would provide the desired vehicle trajectory for an upcoming portion of the race track. Since the

Table 3.3: Iteration Computation Time

| Lookahead (m) | $T$ | Solve Time (s) |
|---|---|---|
| 450 | 184 | 5 |
| 900 | 369 | 6 |
| 1800 | 737 | 12 |
| 4500 | 1843 | 26 |

computation time of the algorithm is dependent on the preview distance, the high-level planner would not need to run at the same sample time as the vehicle controller. Instead, the planner would operate on a separate CPU and provide a velocity profile and racing line for only the next 1-2 kilometers of the race track every few seconds, or plan a path for the next several hundred meters within a second.

Table 3.3 shows problem solve times for a varying range of lookahead lengths with the same discretization $\Delta s$, and shows that the runtime scales roughly linearly with the lookahead distance. The above solve times are listed using the CVX convex optimization solver, which is designed for ease of use and is not optimized for embedded computing. Preliminary work has been successful in implementing the iterative two-step algorithm into C code using the CVXGEN software tool [50]. When written in optimized C code, the algorithm can solve the curvature minimization problem (3.14) in less than 0.005 seconds for a lookahead distance of 650 meters.

The possibility of real-time trajectory planning for race vehicles creates several fascinating areas of future research. An automobile's surroundings are subject to both rapid and gradual changes over time, and adapting to unpredictable events requires an approximate real-time trajectory planning algorithm. On a short time scale, the real-time trajectory planner could find a fast but stable recovery trajectory in the event of the race vehicle entering an understeer or oversteer situation. On an intermediate time scale, the fast executing two-step algorithm could continuously plan a racing line in the presence of other moving race vehicles by constraining the permissible driving areas to be collision-free convex "tubes" [17]. Finally, the algorithm could update the trajectory given estimates of the friction coefficient and other vehicle parameters learned gradually over time.

## 3.9 Conclusion

This chapter demonstrates an iterative algorithm for quickly generating vehicle racing trajectories, where each iteration is comprised of a sequential velocity update and path update step. Given an initial path through the race track, the velocity update step performs forward-backward integration to determine the minimum-time speed inputs. Holding this speed profile constant, the path geometry is updated by solving a convex optimization problem to minimize path curvature.

The primary benefit of the presented trajectory planner is computational speed. Experimental data on an autonomous race vehicle over a three mile race course confirms that the trajectory generated by the algorithm provide comparable lap times to that from a nonlinear gradient descent algorithm. However, the required computation time is at least two orders of magnitude faster. One drawback of the presented approach is that lap time is not explicitly minimized, resulting in sub-optimal performance on complex sequences of turns. A second analysis was therefore conducted in simulation to show the benefit of adding a distance minimizing term to the convex path update step. An exciting opportunity for future research is incorporating the trajectory modification algorithm into an online path planner to provide racing trajectories in real time.

*Note: This chapter reuses material previously published by the author in [40].*

# Chapter 4

# Iterative Learning Control

In Chapters 2 and 3, a steering controller and trajectory planning algorithm were presented for an autonomous race car. One of the goals from the introduction is to compare the resulting autonomous driving performance with that of a human driver. While there are several metrics that could be used as a comparison, by far the easiest to measure and most relevant for racing is lap time. Figure 4.1 shows lap times recorded on the Audi TTS, both autonomously and from two human drivers. One of the human drivers is a professional race driver, while the other is an expert amateur driver.

While the lap times from the autonomous driver are comparable to the amateur expert, they are about a second behind on average from the professional human driver. There are several ways to analyze why this difference arises, but a simple insight that makes the case for learning algorithms comes from viewing the trajectory tracking

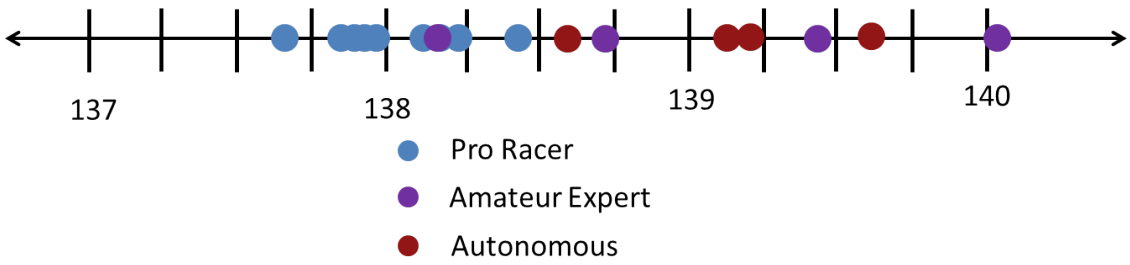


Figure 4.1: Experimentally recorded lap times (in seconds).

performance and friction utilization of the controller, shown in Fig. 4.2.

One of the issues shown in Fig. 4.2 is the relatively poor controller tracking on a few sections of the race track. While this is probably less significant for the lateral speed tracking (sections ②, ④ - ⑦), failing to drive as fast as the trajectory plans (sections ①, ②, ⑦) results directly in a loss of lap time. The speed and lateral path tracking will be improved in this chapter with the addition of iterative learning control (ILC) algorithms.

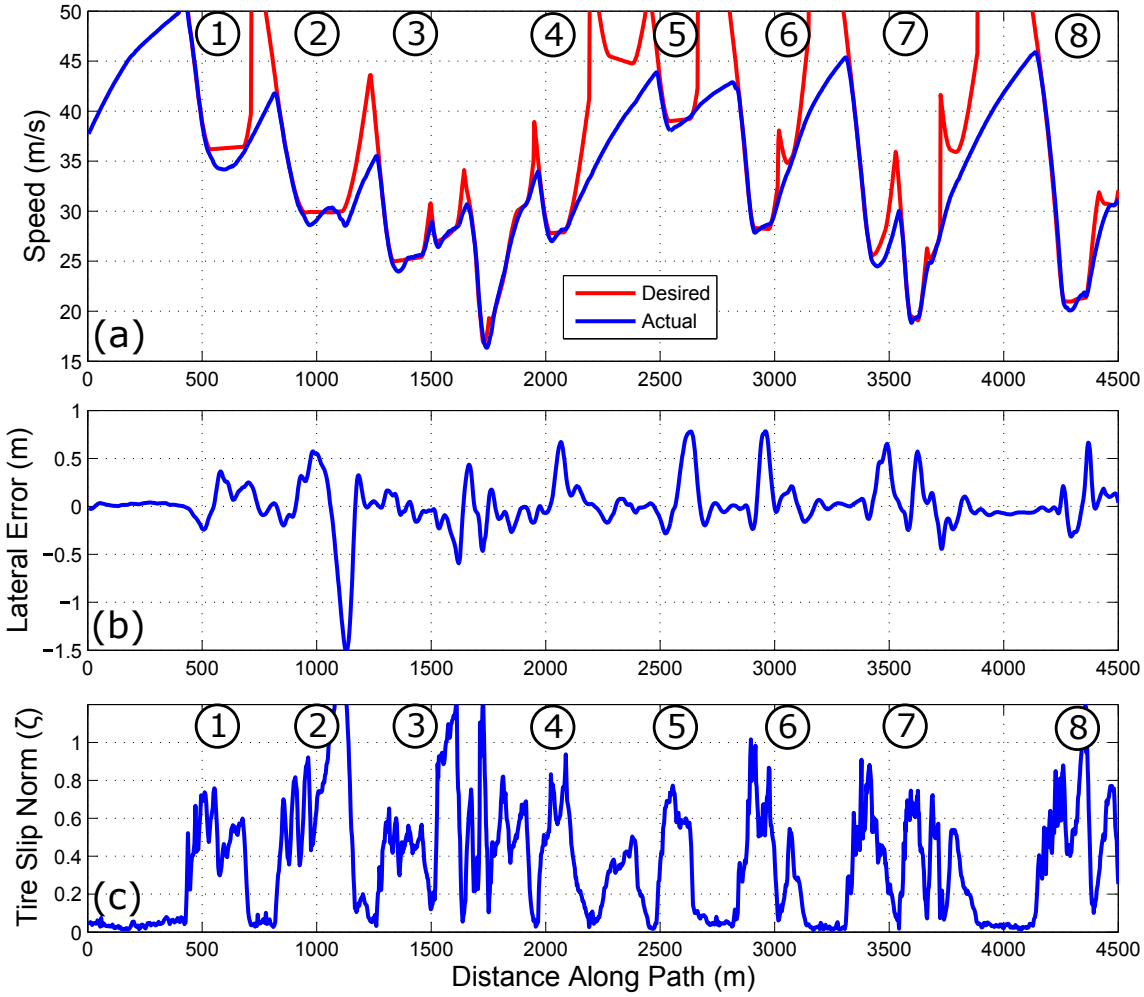


Figure 4.2: Controller tracking performance and tire slip norm on a test run at the limits of handling ($\mu = 0.95$).(a) Desired vs actual speed of the vehicle. (b) Lateral tracking error and (c) tire slip norm as a function of distance along the path.

The second issue shown in Fig. 4.2 is the inconsistent usage of the tire friction capacity, as judged by the tire slip norm metric. The tire slip norm, formalized in [45], is given by $\zeta$:

$$\zeta = \sqrt{\left(\frac{\alpha}{\alpha_p}\right)^2 + \left(\frac{\sigma}{\sigma_p}\right)^2} \tag{4.1}$$

Where $\sigma$ and $\alpha$ are the longitudinal and lateral tire slips for a given tire, and $\sigma_p$ and $\alpha_p$ are empirically determined *peak slip* values resulting in maximum longitudinal and lateral tire force generation. As a result, $\zeta < 1$ corresponds to the tires having excess force generation capacity, while $\zeta > 1$ corresponds to tire saturation. There is technically a $\zeta$ value for each tire, but the maximum $\zeta$ over all four tires is typically used for conservatism.

Fig. 4.2 shows inconsistent usage of tire friction across many turns. On some turns (sections (2) and (8)), the vehicle significantly exceeds the limits of handling, and the car's stability control systems kick in to regain control, slowing the car down in the process. On other turns (sections (3), (5), and (7)), the vehicle uses only a portion of the available tire force, indicating the vehicle can actually drive with higher acceleration on the next lap. Learning from prior runs to find the optimal acceleration (or $\mu$ parameter) for each part of the track will be accomplished in the next chapter via a search algorithm.

### Iterative Learning Control

Iterative learning control (ILC) is based on the notion that the performance of a system that executes the *same task* multiple times can be improved by learning from previous executions (trials, iterations, or in our case, laps of racing) [6]. On every iteration, a control signal is applied to a system in order to follow an ideal, unchanging "reference trajectory". The tracking error for that iteration is recorded, and a learning algorithm is applied to improve the control signal and achieve more accurate system performance on the next iteration. There are a variety of learning algorithms used, but most attempt to correct the tracking error by using a model of the system to determine the augmentation to apply to the prior control signal. This process is

repeated until the reference tracking performance becomes satisfactory. As noted by Bristow et al., this approach is analogous to how a basketball player shooting a free throw from a fixed position can improve her ability to score by practicing the shot repeatedly, making adjustments to her shooting motion based on observations of the ball's trajectory [6].

Inspired by a series of papers published in 1984 [1][12][41], iterative learning control has become increasingly widespread. Because iterative learning control works best when learning to follow the same reference trajectory under the same ambient conditions, the most common applications of ILC are in the field of automated manufacturing. Notable examples include CNC machining [44], industrial robotics [20][32], piezolectric stage positioning [34], motor control [51], and microdeposition [33]. However, the rise of automated systems outside factory environments has led to important applications of ILC for ground and air robotics. In 2006, Chen and Moore [10] proposed a simple iterative learning scheme in 2006 to improve path-following of a ground vehicle with omni-directional wheels. In 2011, Purwin and Andrea synthesized an iterative controller using least-squares methods to aggressively maneuver a quadrotor unmanned aerial vehicle (UAV) from one state to another [60]. As a novel step, the authors then generalized the experience learned from the iterative learning control in order to tune the parameters of the UAV model. In 2013, Sun et al. [71] proposed an iterative learning controller for speed regulation of high-speed trains. Given the high safety requirements of fast trains, the algorithm heavily penalized train overspeeding and enabled the trains to learn how to maintain a safe following distance.

Due to the nature of automotive racing, iterative learning control techniques are a promising method to gradually eliminate the trajectory tracking errors described in Fig. 4.2. For this application of ILC, the repetitive trials are laps of racing, and the reference trajectory is the optimal speed profile and curvature profile from Chapter 3. This chapter will present adaptations of two established ILC methods (proportional-derivative and quadratically optimal) for use in the Audi TTS racing system. Section 4.1 presents both coupled and decoupled models of the vehicle lateral and longitudinal dynamics. These models are converted from state-space representations to

*lifted domain* representations required for iterative learning control in §4.2, and the proportional-derivative and quadratically optimal ILC algorithms are presented in §4.3 and §4.4. Section 4.5 presents simulated results of the ILC algorithms for a sample vehicle trajectory, and finally, experimental results showing a gradual reduction of trajectory-following errors is presented in §4.6.

## 4.1 Dynamic System Model

The ILC algorithms we consider require the closed-loop system dynamics to be (a) stable to any disturbance input, and (b) expressible as an *affine* discrete dynamical system. In our case, we have two subsystems: the steering controller and the longitudinal speed control. Stability of the steering controller under lanekeeping feedback was shown in the linear case by [63] and in the saturated case by [72], and was also discussed in Chapter 2. Similar analyses can be considered to show the stability of the simple proportional speed-following controller.

The more difficult task is expressing the dynamics of the two subsystems using an affine model, given the tendency for the vehicle tires to saturate at the limits. Chapter 2 presented the lateral vehicle dynamics obtained by neglecting longitudinal forces. However, since we are modifying both the longitudinal force $F_x$ and steer input $\delta$ on each iteration, it may be important to account for the coupled lateral/longitudinal dynamics of the vehicle. Nonlinear, coupled equations of motion are provided in (4.2):

$$\frac{de}{dt} = \left(v + U_x^{\text{des}}(s)\right)(\beta + \Delta\Psi) \tag{4.2}$$

$$\frac{dv}{dt} = \left(v + U_x^{\text{des}}(s)\right)\beta r + \frac{F_x}{m} - \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x)\delta}{m} \tag{4.3}$$

$$\frac{d\beta}{dt} = \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x) + F_{\text{yr}}(\alpha_{\text{r}}, F_x)}{m\left(v + U_x^{\text{des}}(s)\right) - r} \tag{4.4}$$

$$\frac{dr}{dt} = \frac{aF_{\text{yf}}(\alpha_{\text{f}}, F_x) - bF_{\text{yr}}(\alpha_{\text{r}}, F_x)}{I_z} \tag{4.5}$$

$$\frac{d\Delta\Psi}{dt} = r \tag{4.6}$$

The system dynamics presented in (4.2) have five states, the original four from Chapter 2 and a new state $v$, the speed tracking error of the system defined by:

$$v = U_x - U_x^{\text{des}} \tag{4.7}$$

The two inputs to the nonlinear system are the steering angle $\delta$ and longitudinal force $F_x$. In reality, the true longitudinal input is a brake pressure or throttle input, but longitudinal force control is assumed for simplicity. The potential for coupling between the subsystems is apparent from (4.2), not only directly from the state equations but also due to the complex nature of tire force generation at the handling limits. As shown in Fig. 4.3, as longitudinal force $F_x$ is distributed across the tires, the available lateral force decreases. At the limits of handling, this *derating* of the lateral force may become significant. As a result, we now model $F_y$ as a function of both lateral tire slip $\alpha$ and the applied longitudinal force $F_x$, using a modified form of the Fiala equation presented in Chapter 2 [30]. Recall that the front and rear tire slips are themselves functions of the vehicle state and are given by:

$$\alpha_{\text{f}} = \beta + \frac{ar}{U_x} - \delta \tag{4.8a}$$

$$\alpha_{\text{r}} = \beta - \frac{br}{U_x} \tag{4.8b}$$

The next step is to break up the control inputs into the closed-loop feedback term and the *learned* component that is modified on every lap by the ILC algorithm:

$$F_x = F_x^{\text{FB}} + F_x^L \tag{4.9}$$

$$= -K_x v + F_x^L \tag{4.10}$$

$$\delta = \delta_{\text{FB}} + \delta_L \tag{4.11}$$

$$\delta = -k_p(e + x_{\text{LA}}\Delta\Psi) + \delta_L \tag{4.12}$$

where $K_x$ is the proportional speed tracking gain and $k_p$ and $x_{\text{LA}}$ are the lookahead

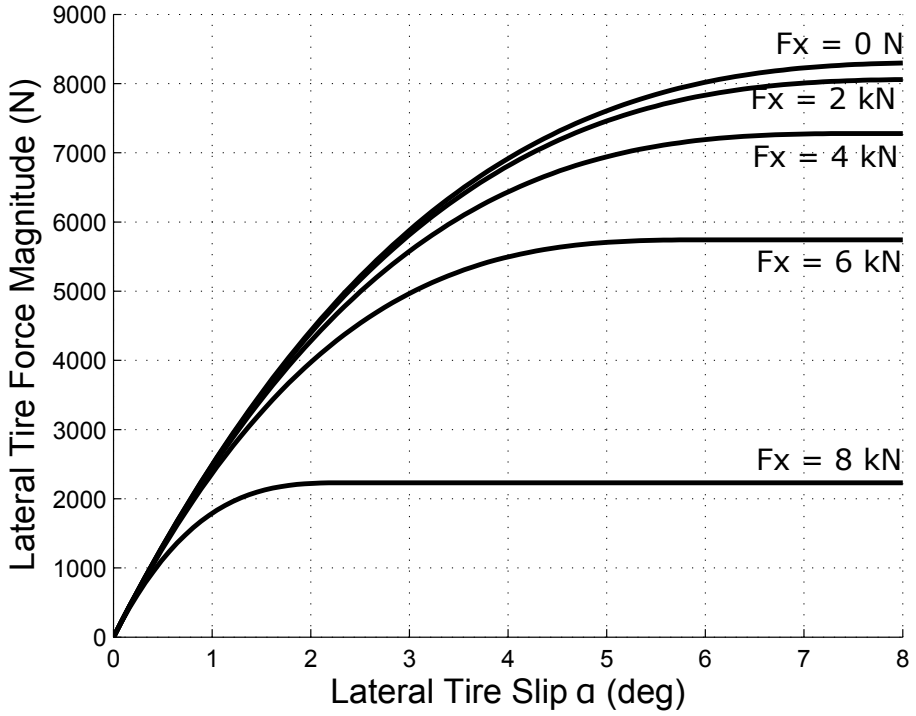


Figure 4.3: Lateral tire force curve as a function of longitudinal force $F_x$ and lateral tire slip $\alpha$.

gains discussed in Chapter 2. Note that (4.9) is similar to a feedback-feedforward control formulation. In fact, iterative learning control achieves near-perfect reference tracking by refining the *feedforward* control input to account for unmodeled dynamics and repeating disturbances that affect the closed-loop controller performance. The closed loop system dynamics are now given by:

$$\frac{de}{dt} = \left(v + U_x^{\text{des}}(s)\right)(\beta + \Delta\Psi) \tag{4.13}$$

$$\frac{dv}{dt} = \left(v + U_x^{\text{des}}(s)\right)\beta r + \frac{-K_x v + F_x^L}{m} - \frac{F_{\text{yf}}(\alpha_F, F_x) - k_p(e + x_{LA}\Delta\Psi) + \delta_L}{m} \tag{4.14}$$

$$\frac{d\beta}{dt} = \frac{F_{\text{yf}}(\alpha_{\text{f}}, F_x) + F_{\text{yr}}(\alpha_{\text{f}}, F_x)}{m\left(v + U_x^{\text{des}}(s)\right) - r} \tag{4.15}$$

$$\frac{dr}{dt} = \frac{aF_{\text{yf}}(\alpha_{\text{f}}, F_x) - bF_{\text{yr}}(\alpha_{\text{r}}, F_x)}{I_z} \tag{4.16}$$

$$\frac{d\Delta\Psi}{dt} = r \tag{4.17}$$

The nonlinear closed-loop dynamics must be converted into an affine, discrete-time dynamical system to apply conventional iterative learning control algorithms. In our case, we have two system outputs ($y = [e\ v]^T$) that are measured and two input signals to learn ($u = [\delta_L\ F_x^L]^T$). Since we run the iterative learning control algorithm after seeing a trial of data, we can approximate the dynamics in (4.13) by linearizing about the observed states and inputs $x_o, u_o$ from the first lap.

The affine model is therefore given by:

$$\begin{bmatrix} \dot{e} \\ \dot{\beta} \\ \dot{r} \\ \dot{\Delta\Psi} \\ \dot{v} \end{bmatrix} = \Big[A(t)\Big] \begin{bmatrix} e \\ \beta \\ r \\ \Delta\Psi \\ v \end{bmatrix} + \Big[B(t)\Big] \begin{bmatrix} \delta_L \\ F_x^L \end{bmatrix} + d(t) \tag{4.18}$$

$$\begin{bmatrix} \dot{e} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x \tag{4.19}$$

Where $A(t)$ and $B(t)$ are given by Jacobian linearizations of the closed loop nonlinear dynamics $f(x)$ (4.13) about the observed states and inputs $x_o(t), u_o(t)$ from

the last trial:

$$\Big[A(t)\Big] = \frac{\partial f}{\partial x}\Big|_{x_o(t)} \tag{4.20}$$

$$\Big[B(t)\Big] = \frac{\partial f}{\partial u}\Big|_{u_o(t)} \tag{4.21}$$

While this multiple-input, multiple-output (MIMO) model captures the coupled behavior of the longitudinal and lateral inputs, it is tedious to compute, either numerically or analytically. For the case where the longitudinal and lateral inputs are decoupled, we obtain the same Chapter 2 linear state equations for the lateral dynamics:

$$\dot{\beta} = \frac{F_{\text{yf}} + F_{\text{yr}}}{mU_x} - r \qquad \dot{r} = \frac{aF_{\text{yf}} - bF_{\text{yr}}}{I_z} \tag{4.22a}$$

$$\dot{e} = U_x(\beta + \Delta\Psi) \qquad \Delta\dot{\Psi} = r - U_x\kappa \tag{4.22b}$$

as well as the following first order equation for the longitudinal dynamics, assuming a simple point mass model with proportional speed tracking feedback:

$$\dot{v} = \frac{-K_x v + F_x^L}{m} \tag{4.23}$$

The resulting state matrices $A(t)$ and $B(t)$ for (4.18) are then simply block diagonal matrices consisting of the linearized lateral dynamics from Chapter 3 and the first order longitudinal dynamics:

$$A(t) = \begin{bmatrix} 0 & U_x(t) & 0 & U_x(t) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \frac{-ak_{\mathrm{p}}\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & \frac{-ak_{\mathrm{p}}x_{\mathrm{LA}}\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & \frac{-a^2\tilde{C}_{\mathrm{f}}(t)-b^2\tilde{C}_{\mathrm{r}}(t)}{U_x(t)I_{\mathrm{z}}} & \frac{b\tilde{C}_{\mathrm{r}}(t)-a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & 0 \\ \frac{-k_{\mathrm{p}}\tilde{C}_{\mathrm{f}}(t)}{mU_x(t)} & \frac{-k_{\mathrm{p}}x_{\mathrm{LA}}\tilde{C}_{\mathrm{f}}(t)}{mU_x(t)} & \frac{b\tilde{C}_{\mathrm{r}}(t)-a\tilde{C}_{\mathrm{f}}(t)}{mU_x(t)^2} - 1 & \frac{-\tilde{C}_{\mathrm{f}}(t)-\tilde{C}_{\mathrm{r}}(t)}{mU_x(t)} & 0 \\ 0 & 0 & 0 & 0 & -K_x \end{bmatrix} \tag{4.24}$$

$$B(t) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{a\tilde{C}_{\mathrm{f}}(t)}{I_{\mathrm{z}}} & 0 \\ \frac{\tilde{C}_{\mathrm{f}}(t)}{mU_x(t)} & 0 \\ 0 & 1 \end{bmatrix} \tag{4.25}$$

The affine term $d(t)$ is given by:

$$d(t) = \begin{bmatrix} 0 \\ -\kappa(t)U_x(t) \\ \frac{a\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t)-b\tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t)+a\tilde{F}_{\mathrm{yf}}(t)-b\tilde{F}_{\mathrm{yr}}(t)}{I_z} \\ \frac{\tilde{C}_{\mathrm{f}}(t)\tilde{\alpha}_{\mathrm{f}}(t)+\tilde{C}_{\mathrm{r}}(t)\tilde{\alpha}_{\mathrm{r}}(t)+\tilde{F}_{\mathrm{yf}}(t)+\tilde{F}_{\mathrm{yr}}(t)}{mU_x(t)} \\ 0 \\ 0 \end{bmatrix} \tag{4.26}$$

While (4.24) is written as a MIMO system for compactness, assuming decoupled lateral and longitudinal dynamics provides two single-input, single-output (SISO) systems.

## 4.2 Lifted Domain Representation and ILC Problem Statement

Whether the coupled (4.20) or decoupled (4.24) dynamics are assumed, the final modeling step is to apply standard discretization techniques to obtain dynamics in the following form:

$$x_{k+1} = A_k x_k + B_k u_k + d_k \tag{4.27}$$

$$y_k = C x_k \tag{4.28}$$

For a given lap of racing $j$, sensor measurements provide $N$ observations of both the lateral path deviation $e$ and longitudinal speed tracking error $v$. These measurements can be stacked into a $2N \times 1$ array:

$$\mathbf{e}_j = \begin{bmatrix} e_1 & \dots & e_N & v_1 & \dots & v_N \end{bmatrix}^T \tag{4.29}$$

These measurement errors are related to the learned control inputs $\delta^L$ and $F_x^L$ as follows:

$$\mathbf{e}_j = P\mathbf{u}_j^L + \mathbf{w} \tag{4.30}$$

$$\mathbf{u}_j^L = \begin{bmatrix} \delta_1^L & \dots & \delta_N^L & F_{x1}^L & \dots & F_{xN}^L \end{bmatrix}^T \tag{4.31}$$

The system dynamics modeled in the previous section are represented by the *lifted-domain* dynamics matrix $P$, which is $2N \times 2N$ and given by:

$$P = \left[\begin{array}{c|c} P_{e\delta} & P_{eF} \\ \hline P_{v\delta} & P_{vF} \end{array}\right] \tag{4.32}$$

Where each submatrix in (4.32) is $N \times N$ and represents the lifted-domain dynamics from a given input to a given output. Individual terms of the sub-matrices

are given by:

$$p_{lk} = \begin{cases} 0 & \text{if } l < k \\ C_y B_u(k) & \text{if } l = k \\ C_y A(l)A(l-1)\cdots A(k)B_u(k) & \text{if } l > k \end{cases} \tag{4.33}$$

Where $C_y$ is the row of $C$ in (4.27) corresponding to the desired output and $B_u$ the column of $B$ in (4.27) corresponding to the desired input. Note that for the case of uncoupled lateral and longitudinal dynamics, the off-diagonal sub-matrices of $P$ are [0] since we have two SISO systems. The term $\mathbf{w}$ in (4.30) is the unknown disturbance. Iterative learning control relies on the assumption that this disturbance is the underlying cause of the observed errors $\mathbf{e}_j$, and that the disturbance, while unknown, is constant from lap to lap.

Given the error signal $\mathbf{e}_j$ for a given lap $j$, the iterative learning problem is to find the inputs $\mathbf{u}_{j+1}$ that will cancel out the tracking error on the next lap. The learned inputs are then applied, the observed error $\mathbf{e}_{j+1}$ is recorded, and the process is repeated until the tracking error falls to a desired level. There is a wide body of literature on methods to determine $\mathbf{u}_{j+1}$ given $P$ and $\mathbf{e}_j$, but this dissertation will investigate the most common approach. We compute the ILC input for the next lap with the following formulation:

$$\mathbf{u}_{j+1}^L = Q(\mathbf{u}_j^L - L\mathbf{e}_j) \tag{4.34}$$

where $Q$ is the $2N \times 2N$ *filter* matrix, and $L$ is the $2N \times 2N$ *learning* matrix. In the following two sections, the matrices $Q$ and $L$ will be obtained by designing a proportional-derivative (PD) iterative learning controller as well as a quadratically optimal (Q-ILC) learning controller.

## 4.3 Proportional-Derivative Controller

The proportional-derivative ILC computes the steering addition $\delta^L$ for the current lap $j$ based on the error $e_{j-1}$ and error derivative at the same time index $k$ from the previous lap. A similar formulation is used for the force addition.

$$\delta_j^L(k) = \delta_{j-1}^L(k) - k_{p\delta} e_{j-1}(k) - k_{d\delta}(e_{j-1}(k) - e_{j-1}(k-1)) \tag{4.35}$$

$$F_j^L(k) = F_{j-1}^L(k) - k_{pF} v_{j-1}(k) - k_{dF}(v_{j-1}(k) - v_{j-1}(k-1)) \tag{4.36}$$

where $k_{p\delta}$ and $k_{pF}$ are proportional gains and $k_{d\delta}$ and $k_{dF}$ are derivative gains. In the lifted domain representation from (4.34), the resulting learning matrix $L$ is given by

$$L = \begin{bmatrix} -(k_{p\delta}+k_{d\delta}) & & 0 & 0 & \dots & 0 \\ k_{d\delta} & \ddots & & \vdots & \ddots & \vdots \\ 0 & k_{d\delta} & -(k_{p\delta}+k_{d\delta}) & 0 & \dots & 0 \\ 0 & \dots & 0 & -(k_{pF}+k_{dF}) & & 0 \\ \vdots & \ddots & \vdots & k_{dF} & \ddots & \\ 0 & \dots & 0 & 0 & k_{dF} & -(k_{pF}+k_{dF}) \end{bmatrix} \tag{4.37}$$

The PD equation (4.35) determines $\delta^L$ only using lateral path deviation $e$ and $F_x^L$ using only the speed tracking error $v$. Since we have formulated the problem as a MIMO system, it is possible to generalize and have both inputs depend on both outputs, but this is not considered for simplicity of gain selection. The filter matrix $Q$ is obtained by taking any filter transfer function and converting into the lifted domain via (4.33). An important design consideration in choosing the two $k_p$ and $k_d$ gains is avoiding a poor lap-to-lap "transient" response, where the path tracking error increases rapidly over the first several laps before eventually decreasing to a converged error response $e_\infty$. This is a commonly encountered design requirement for ILC systems, and can be solved by ensuring the following *monotonic convergence* condition is met [6]:

$$\gamma \triangleq \bar{\sigma}(PQ(I - LP)P^{-1}) < 1 \tag{4.38}$$

where $\bar{\sigma}$ is the maximum singular value. In this case, the value of $\gamma$ provides an upper bound on the change in the tracking error norm from lap to lap, i.e.

$$||\mathbf{e}_\infty - \mathbf{e}_{j+1}||_2 \leq \gamma||\mathbf{e}_\infty - \mathbf{e}_j||_2 \tag{4.39}$$

Fig. 4.4 shows values of $\gamma$ for both an unfiltered PD controller ($Q = I$), and for a PD controller with a 2 Hz, first order low pass filter. The $\gamma$ values are plotted as a contour map against the controller gains $k_{p\delta}$ and $k_{d\delta}$. Addition of the low-pass filter assists with monotonic stability by removing oscillations in the control input generated when trying to remove small reference tracking errors after several iterations. Since the filtering occurs when generating a control signal for the next lap, the filter $Q$ can be zero-phase. The plot shown in Fig. 4.4 is for the steering ILC design only, but the same analysis is possible for the longitudinal ILC design as well.

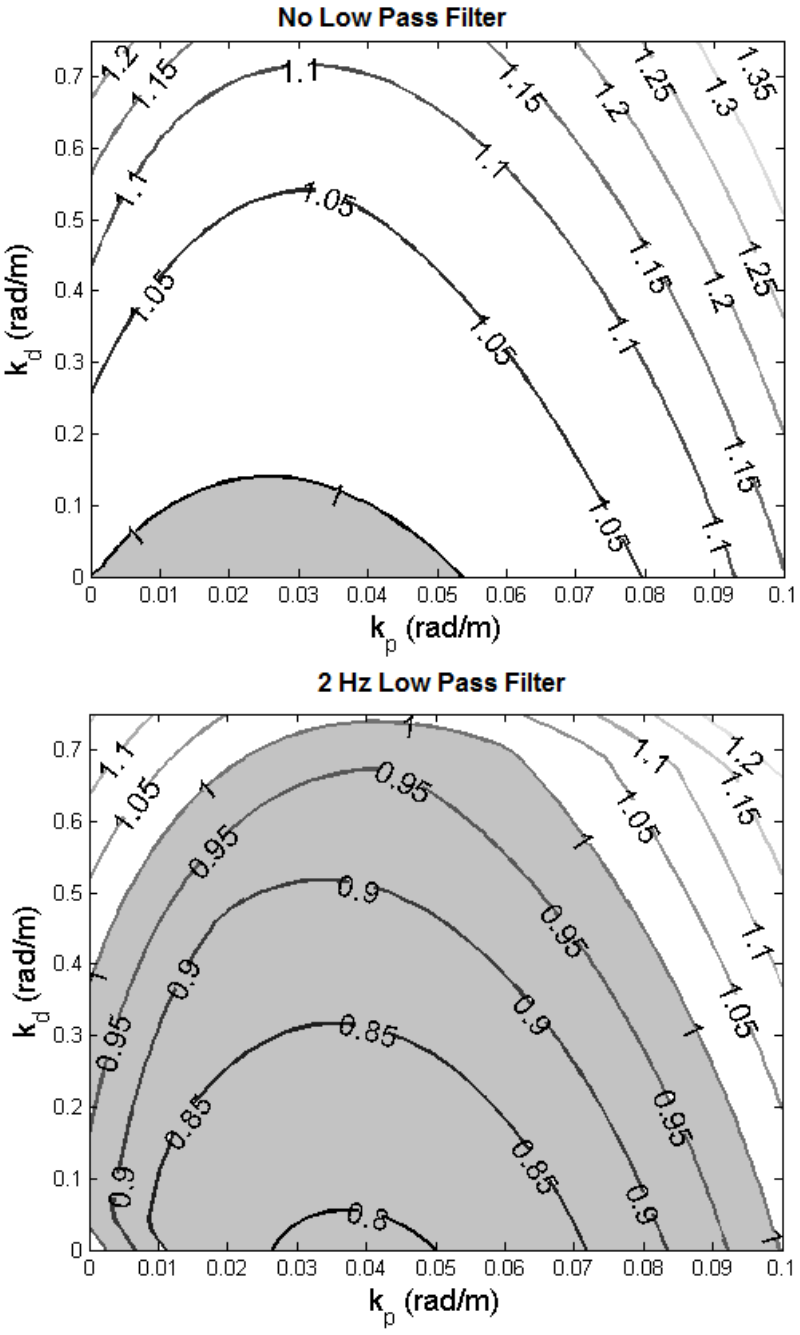


Figure 4.4: Values of convergence bound $\gamma$ vs. $k_{p\delta}$ and $k_{d\delta}$ for PD iterative learning controller with (top) no filtering and (bottom) with a 2 Hz low-pass filter. Lower values of $\gamma$ correspond to faster convergence. Shaded regions correspond to gains that result in system monotonic stability.

## 4.4 Quadratically Optimal Controller

An alternate approach to determining the learned steering and longitudinal force input is to minimize a quadratic cost function for the next lap:

$$J_{j+1} = \mathbf{e}_{j+1}^T T \mathbf{e}_{j+1} + \mathbf{u}_{j+1}^T R \mathbf{u}_{j+1}^T + \Delta_{j+1}^T S \Delta_{j+1} \tag{4.40}$$

where $\Delta_{j+1} = \mathbf{u}_{j+1}^L - \mathbf{u}_j^L$ and the $2N \times 2N$ matrices $T$, $R$, and $S$ are weighting matrices, each given by a scalar multiplied by the identity matrix for simplicity. This formulation allows the control designer to weight the competing objectives of minimizing the tracking errors $e$ and $v$, control effort $|\delta^L|$ and $|F_x^L|$, and change in the control signal from lap to lap. While constraints can be added to the optimization problem, the unconstrained problem in (4.40) can be solved analytically [5] to obtain desired controller and filter matrices:

$$Q = (P^T T P + R + S)^{-1}(P^T T P + S) \tag{4.41a}$$

$$L = (P^T T P + S)^{-1} P^T T P (T^{1/2} P)^{-1} T^{1/2} \tag{4.41b}$$

An advantage of the quadratically optimal control design over the simple PD controller is that the controller matrices $Q$ and $L$ take the linearized, time-varying dynamics $P$ into account. This allows the iterative learning algorithm to take into account changes in the steering dynamics due to changes in vehicle velocity. Furthermore, if the fully coupled dynamics (4.20) are used, the iterative learning algorithm also accounts for the second-order effect of steering on the longitudinal dynamics and longitudinal force application on the lateral dynamics. However, a disadvantage is that computing $\delta^L$ in (4.34) requires matrix multiplications with the typically dense matrices $Q$ and $L$ for every lap, which can be computationally expensive for fast sampling rates.

## 4.5 Simulated Results

To test the feasibility of the PD and Q-ILC learning algorithms, the vehicle tracking performance over multiple laps is simulated using the path curvature and speed profiles shown in Fig. 4.5. To test the performance of the controller at varying accelerations, four speed profiles are tested. Each profile is generated with a different level of peak combined longitudinal/lateral acceleration, ranging from 5 m/s$^2$ (below the limits) up to 9.5 m/s$^2$ (close to exceeding the limits). For accurate results, simulations were conducted using numerical integration with fully nonlinear equations of motion and coupled lateral/longitudinal dynamics.

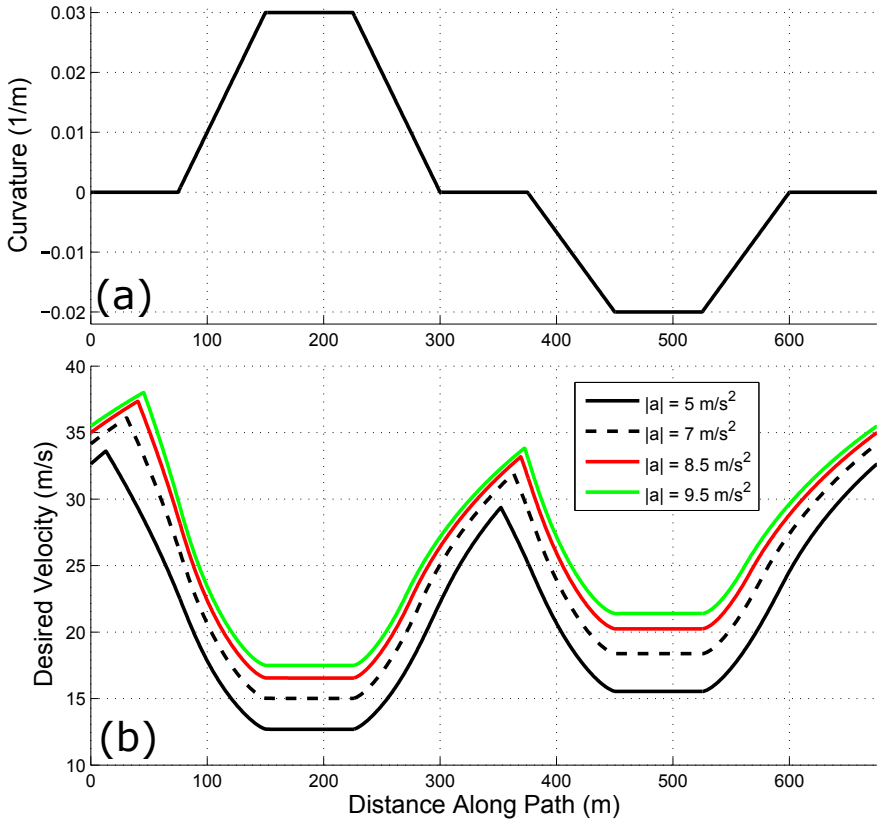


Figure 4.5: (a) Curvature profile used for ILC simulation. (b) Velocity profiles generated for four different levels of longitudinal/lateral acceleration.

Simulated results showing the root-mean-square (RMS) lateral path deviation is shown in Fig. 4.6. The results show the change in RMS error as the number of ILC iterations increase. Three different ILC controllers are tested. The first controller is the simple PD controller with low-pass filter (4.35), and the second controller is the quadratically optimal ILC algorithm (4.41) assuming fully coupled dynamics (4.20) in the plant matrix $P$ (i.e. the full MIMO system). The third controller is also the quadratically optimal ILC algorithm, but the $P$ matrix used in the optimization assumes decoupled dynamics (4.24) and therefore solves the lateral and longitudinal SISO problems separately.

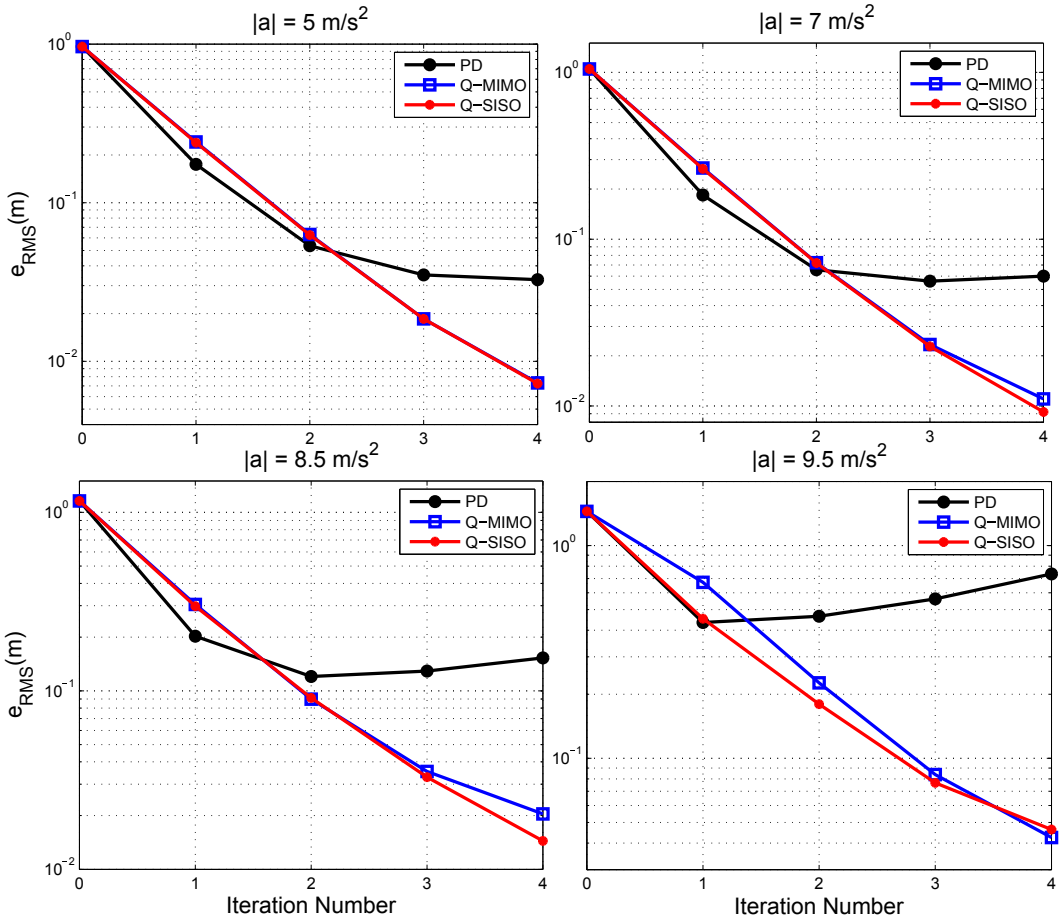


Figure 4.6: Simulated results for root-mean-square path tracking error at several values of vehicle acceleration, with $T = R = I$ and $S = 100I$. Results are plotted on a log scale.

Fig. 4.6 shows that both quadratically optimal ILC algorithms exponentially reduce the lateral path tracking error as the number of learning iterations is increased. Overall, the RMS lateral tracking performance is better at lower vehicle accelerations. This is unsurprising for two reasons. In Chapter 2, we discovered that lateral path deviation in general increases for the lookahead steering feedback at higher accelerations. Second, our estimate of the vehicle dynamics contained in $P$ is based on linearization, and the vehicle dynamics at lower accelerations are mostly linear.

Fig. 4.7 shows the same results as Fig. 4.6, but for the speed tracking performance. The overall trends are very similar. In both plots, there is very little difference between the coupled MIMO formulation and decoupled SISO formulation at low accelerations. This is expected, as the longitudinal and lateral dynamics are independent when the vehicle tires are not saturated. At higher accelerations, there are small differences, but the overall RMS errors are still quite similar. A reason for this is the nature of the speed profiles in Fig. 4.5. The vehicle spends the majority of time either fully braking/accelerating or turning at a constant velocity. There are only a few small transient regions where the vehicle needs significant amounts of both lateral and longitudinal acceleration. As a result, the need to account for the coupled dynamics may not be important in practice, especially given the larger computation time needed when $P$ is dense and not block-diagonal.

A final comment is that the proportional-derivative ILC algorithm performs relatively poorly. At low accelerations, the speed and path tracking performance both improve initially, but fail to improve after the second learning iteration. At high lateral and longitudinal accelerations, the tracking performance becomes even worse for the steering ILC in Fig. 4.6. This is unsurprising given that the linearized plant dynamics $P$ are not explicitly accounted for in the selection of the PD gains. While the point mass model for the longitudinal dynamics is a relatively simple first order model, the lateral dynamics are fourth order and highly speed dependent. A simple PD approach for learning control is likely insufficient at the limits of handling without a more sophisticated set of PD gains for different vehicle speeds.

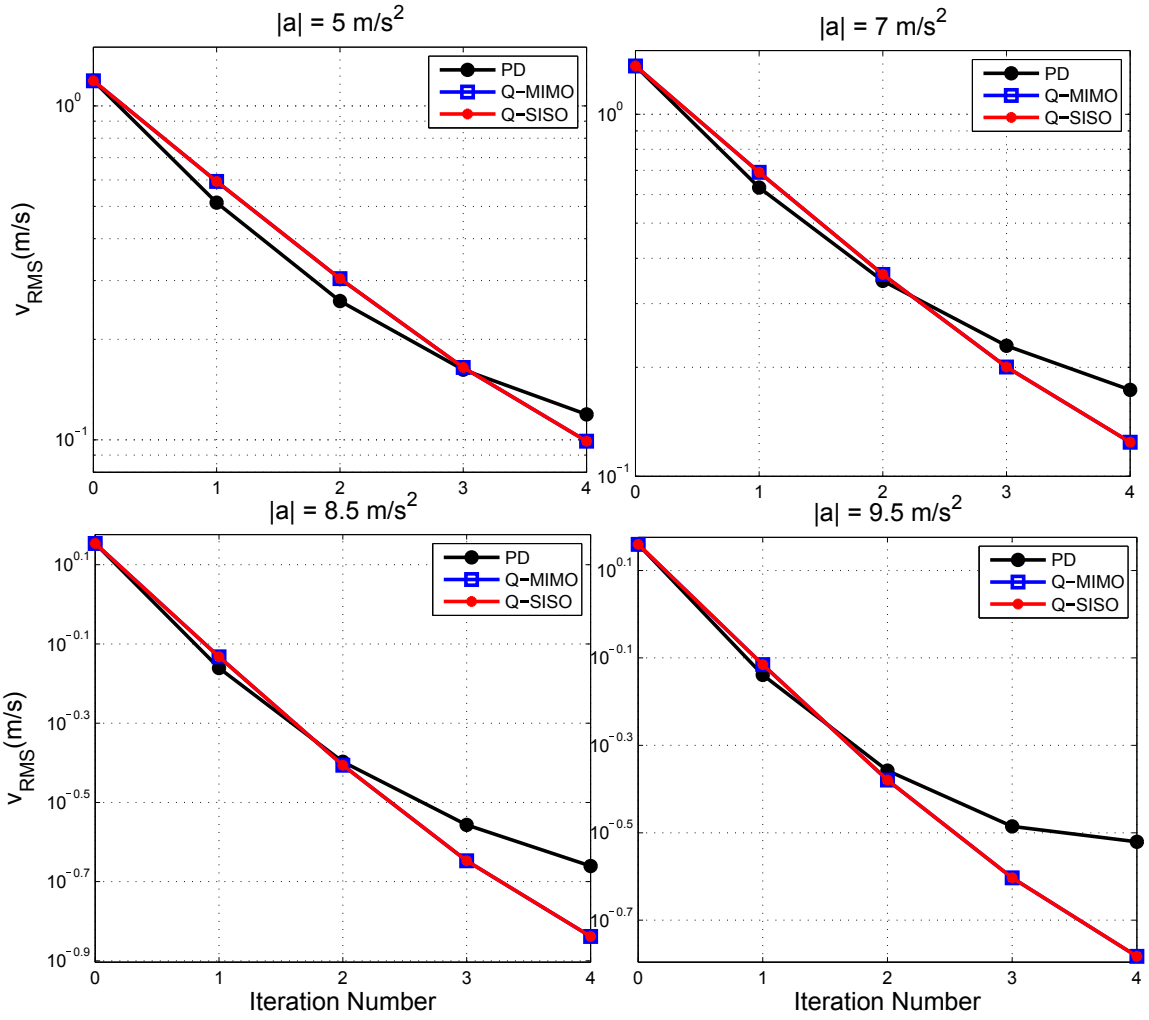


Figure 4.7: Simulated results for root-mean-square speed tracking error $v$ at several values of vehicle acceleration, with $T = I$, $R = 0$, and $S = 1\mathrm{e}-7I$. Results are plotted on a log scale.

## 4.6 Experimental Results

Experimental data for iterative learning control was collected over four laps at Thunderhill Raceway with the autonomous Audi TTS setup described in Chapters 2 and 3. The experimental controller setup is shown in Fig. 4.8, and controller parameters are shown in Table 4.1.

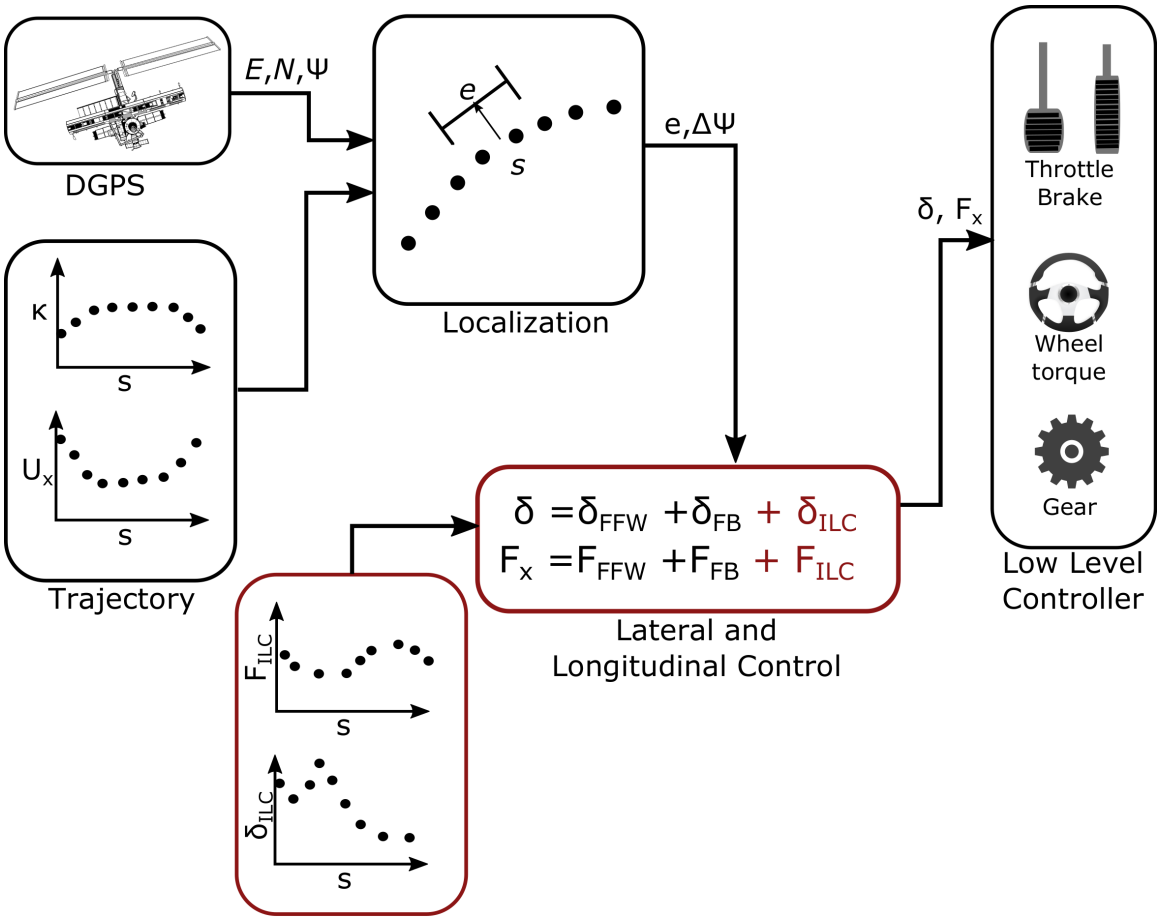


Figure 4.8: Controller setup for experimental testing of iterative learning control.

Table 4.1: Vehicle Parameters

| Parameter | Symbol | Value | Units |
|---|---|---|---|
| Lookahead Distance | $x_{\mathrm{LA}}$ | 15.2 | m |
| Lanekeeping Gain | $k_{\mathrm{LK}}$ | 0.053 | $\mathrm{rad\,m^{-1}}$ |
| Lanekeeping Sample Time | $t_s$ | 0.005 | s |
| ILC Sample Time | $T_s$ | 0.1 | s |
| Speed Tracking Gain | $K_x$ | 2500 | $\mathrm{Nsm^{-1}}$ |
| Q-ILC Matrix (Path) | $T$ and $R$ | $I$ | - |
| Q-ILC Matrix (Path) | $S$ | $100\,I$ | - |
| Q-ILC Matrix (Speed) | $T$ | $I$ | - |
| Q-ILC Matrix (Speed) | $R$ | 0 | - |
| Q-ILC Matrix (Speed) | $S$ | 1e-7 $I$ | - |

The key difference between the controller setup from Chapter 3 is the inclusion of the learning inputs $\delta^L$ and $F_x^L$. To save computation time, these are calculated at a 10 Hz update rate at the end of every race lap and stored as lookup tables in the controller. Since the the real-time control occurs at 200 Hz, at every time step the controller interpolates the lookup table and applies the correct force and steer angle correction. One key difference between the simulation and the experiment is that the simulation only applied the learning correction to the closed-loop path tracking controller. In the experiment, the steady-state feedforward control laws from Chapter 2 are also applied to keep the tracking error on the first lap below 1 m for safety.

Fig. 4.9 shows the applied iterative learning signals and resulting path tracking error over four laps using the SISO quadratically optimal learning algorithm. The car is driven aggressively at peak lateral/longitudinal accelerations of 8 $\mathrm{m/s^2}$. On the first lap, despite the incorporation of a feedforward-feedback controller operating at a high sampling rate, several transient spikes in tracking error are visible due to the underdamped tire dynamics near the limits of handling.

However, the iterative learning algorithm is able to significantly attenuate these transient spikes after just two or three laps. One of the most important features of the time series plot is that the learned steering corrections are applied slightly before a lateral path deviation was observed the prior lap (i.e. the steering corrections *lead*

the observed error). This is because the learning algorithm has knowledge of the system model and knows that a steering correction must be applied a few meters early to cancel a path deviation further down the road.

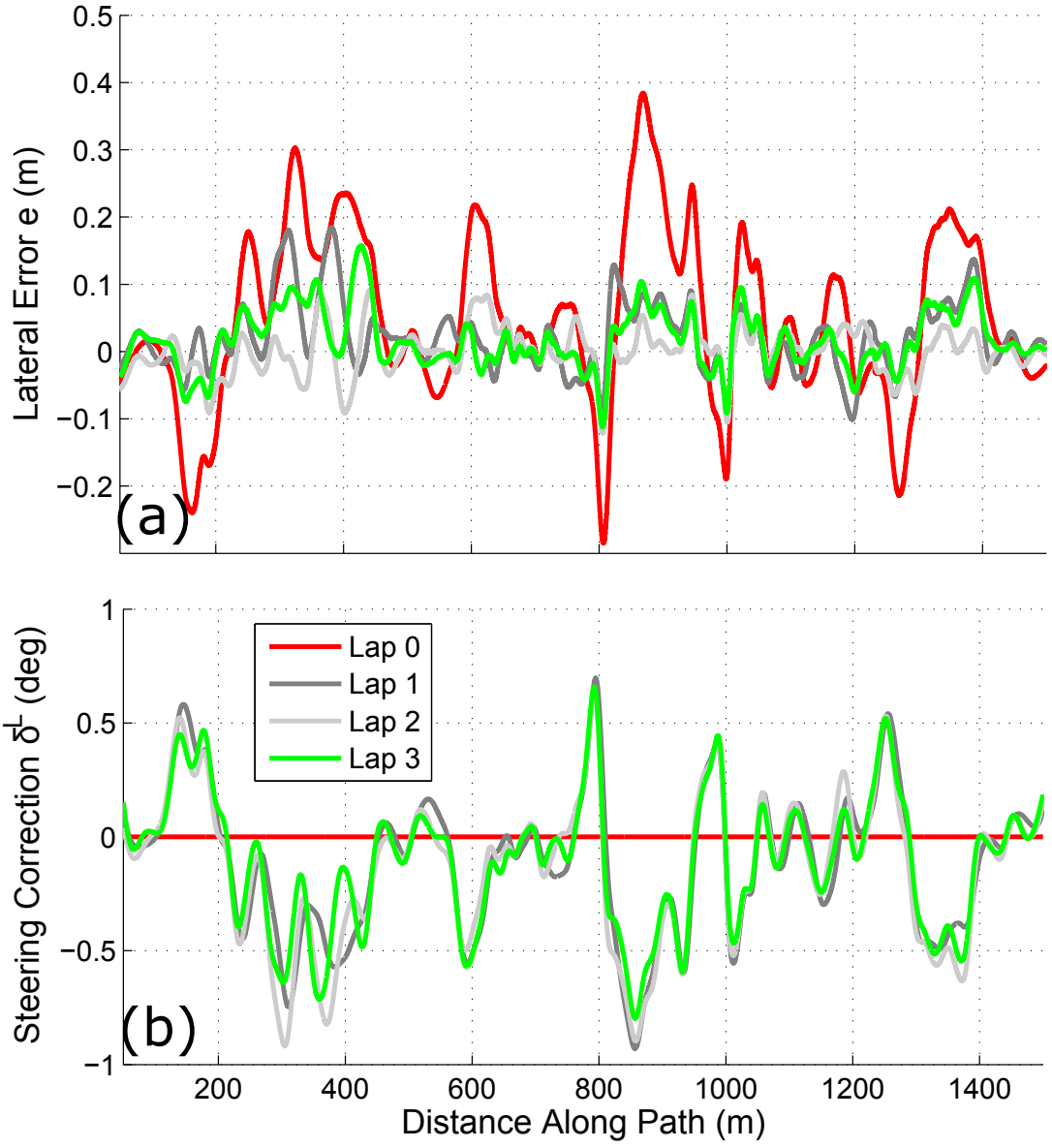


Figure 4.9: Experimental results for path tracking error with Q-SISO learning controller, at peak lateral accelerations of 8 m/s$^2$.

Fig. 4.10 shows the applied iterative learning signals for the longitudinal speed control at a slightly higher acceleration of 8.5 m/s$^2$. Again, in just two to three iterations of learning, significant lags in the speed tracking are attenuated from $s = 800 - 900$ meters and $s = 1200 - 1300$ meters. Additionally, the controller also acts to slow the car down when $v > 0$ and the vehicle exceeds the speed profile. This is also desirable from a racing perspective as it it prevents the vehicle from exceeding the friction limit.

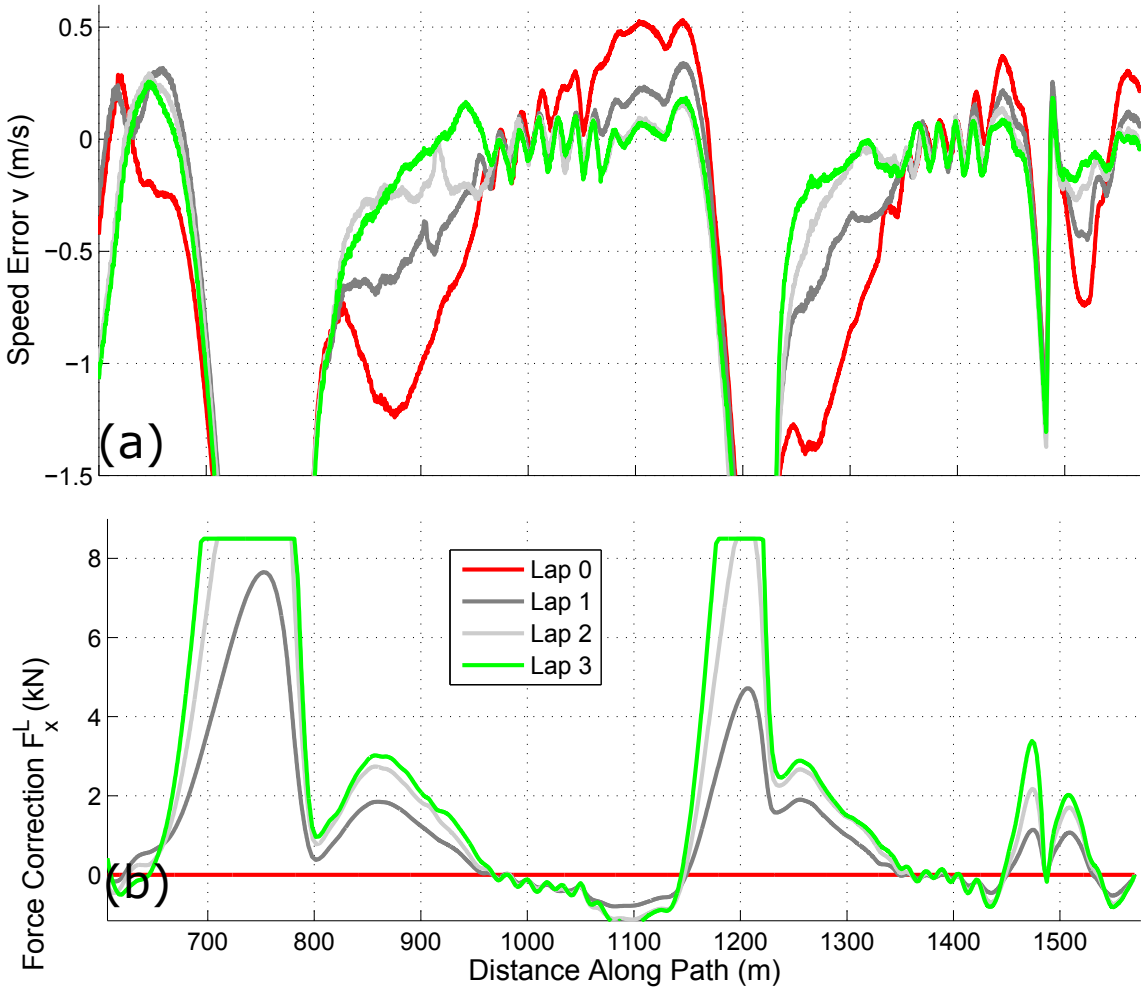


Figure 4.10: Experimental results for speed tracking error with Q-SISO learning controller, at peak lateral accelerations of 8.5 m/s$^2$.

Notice that Fig. 4.10 has several regions where the speed error $v << 0$. These are straight regions of the track where the vehicle can not match the planned longitudinal acceleration due to engine limitations. As a result, the ILC is programmed to saturate the longitudinal learning signal to 8000 Newtons.

In Fig. 4.11, root-mean-square tracking results are shown for a range of peak vehicle accelerations. The results show that at lower vehicle accelerations, the initial speed and lateral tracking errors (Iteration 0) are smaller, as the built-in feedback-feedforward controller performs better. However, as the speed profile becomes more aggressive, the path and speed tracking degrades in the presence of highly transient tire dynamics. Regardless of the initial error, application of iterative learning control reduces the trajectory tracking errors significantly over just 2 or 3 laps. At an acceleration of 8.5 m/s$^2$, for example, the RMS lateral tracking error is around 3 cm, on the order of the expected RMS error from the GPS position sensor! On some tests, the RMS tracking error occasionally increases slightly from Lap 2 to Lap 3, and for the case where vehicle acceleration is 9 m/s$^2$, the lateral tracking error is constant from Lap 1 to Lap 2 before decreasing further in Lap 3. While not predicted in simulation, this behavior likely occurs because the repeating disturbance from lap-to-lap is not exactly constant, especially as the vehicle approaches the handling limits. More refined tuning of the gain matrices may be able to prevent this RMS error increase, or the ILC algorithm can be stopped after several iterations once the tracking performance is acceptable.

Experimental results in this section were only given for the quadratically optimal controller with decoupled (SISO) dynamics. The PD iterative learning controller was not tested due to the relatively worse simulation performance, and the quadratically optimal controller with coupled dynamics provided no clear benefit in simulation but a much longer computation time.

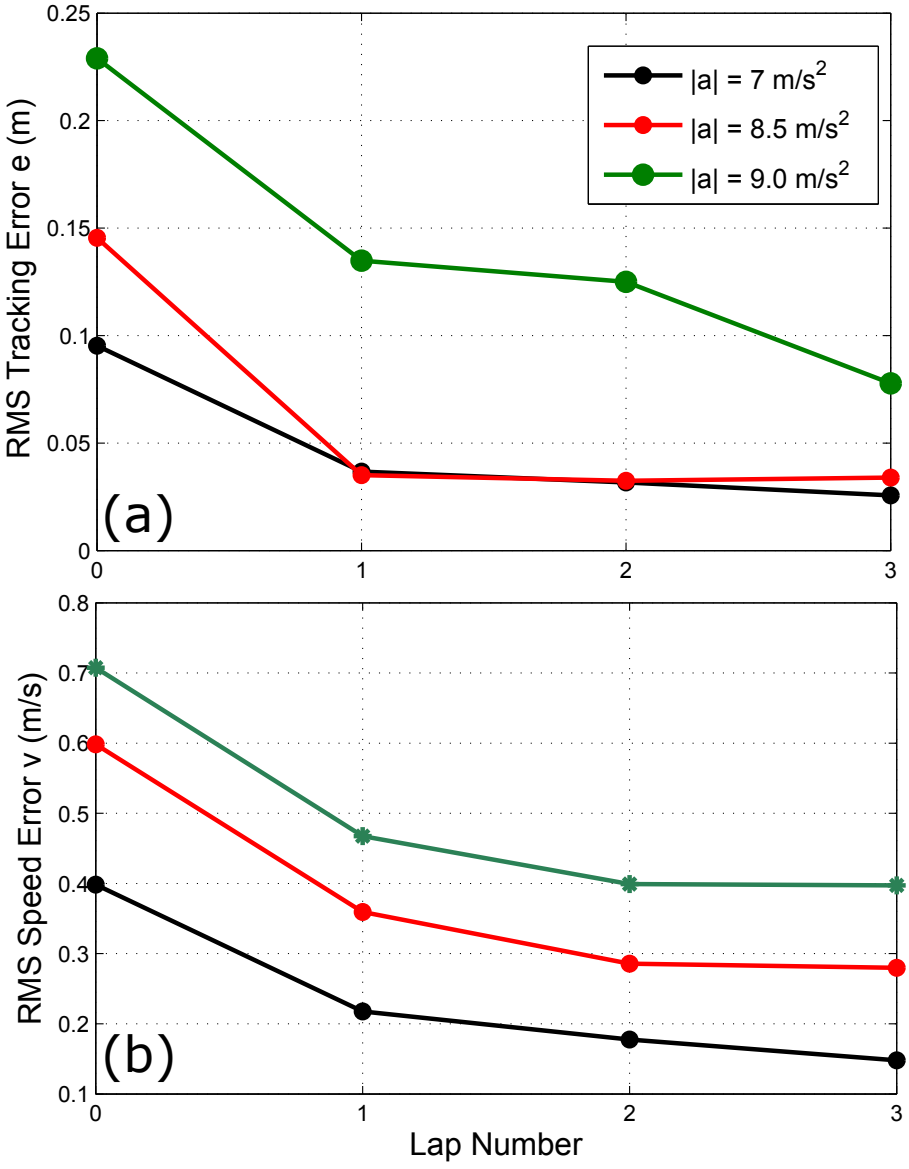


Figure 4.11: Experimental RMS tracking error for the Q-SISO learning controller at several levels of lateral acceleration. (a) RMS lateral tracking error as a function of lap number/iteration number. (b) RMS speed tracking error as a function of lap/iteration number. Note that lap 0 corresponds to the baseline case where iterative learning control is not applied.

## 4.7 Conclusion

This chapter demonstrated the application of iterative learning control (ILC) methods to achieve accurate trajectory following for an autonomous race car over multiple laps. Two different algorithms, proportional-derivative (PD) and quadratically optimal (Q-ILC) learning control are tested in simulation and then used to experimentally eliminate path tracking errors caused by the transient nature of the vehicle dynamics near the limits of friction.

The primary significance of this work is improved racing performance of the autonomous vehicle over time. Because the vehicle lateral and longitudinal dynamics become difficult to accurately model at the limits of handling, following a minimum-time speed and curvature profile is difficult to achieve over one lap with a standard feedback control system. However, because the desired trajectory and vehicle conditions are relatively unchanged on each subsequent lap, the presented ILC algorithms ensure accurate tracking of the minimum-time trajectory after just two or three laps of learning.

One drawback with iterative learning control is that applying a steering wheel input to eliminate lateral errors will work only if the vehicle is near the limits of handling, but has not fully saturated the available tire force and entered a limit understeer condition. In the next chapter, a separate learning algorithm is developed to learn the best velocity profile that minimizes lap time by maximizing the available tire friction on all turns of the track.

*Note: This chapter reuses material previously published by the author in [39].*

# Chapter 5

# Learning the Optimal Speed Profile

The iterative learning algorithms presented in Chapter 4 can help an autonomous vehicle follow a desired trajectory more precisely over several laps of driving. However, the ILC algorithms do not alter the trajectory itself, only the input signals that attempt to track the trajectory. This will not be sufficient at the limits of handling. Consider Fig. 4.2, reprinted below. The speed profile was generated assuming a tire-road friction value of $\mu = 0.94$. Since peak vehicle acceleration is given by $\mu g$, this corresponds to maximum lateral and longitudinal acceleration values of 9.4 m/s$^2$. While this is a reasonable assumption overall, there are several parts on the track where the vehicle exceeds the available friction and begins to understeer. Region (2) is one example. The tire slip norm $\zeta$ (4.1) climbs above one, and as a result, the vehicle begins to slide off the track, resulting in the large negative tracking error spike in Fig. 5.1(b). The vehicle's stability algorithms (not discussed in this thesis, see [45]) kick in and slow the vehicle down, which results in the speed tracking error seen in Fig. 5.1(a). In this situation, iterative learning control will be unable to achieve better speed tracking and lateral tracking performance. The tires are saturated and the car is slowly beginning to careen off the track. Simply steering more on the next lap will not achieve better tracking performance due to the saturation of the steering actuator.

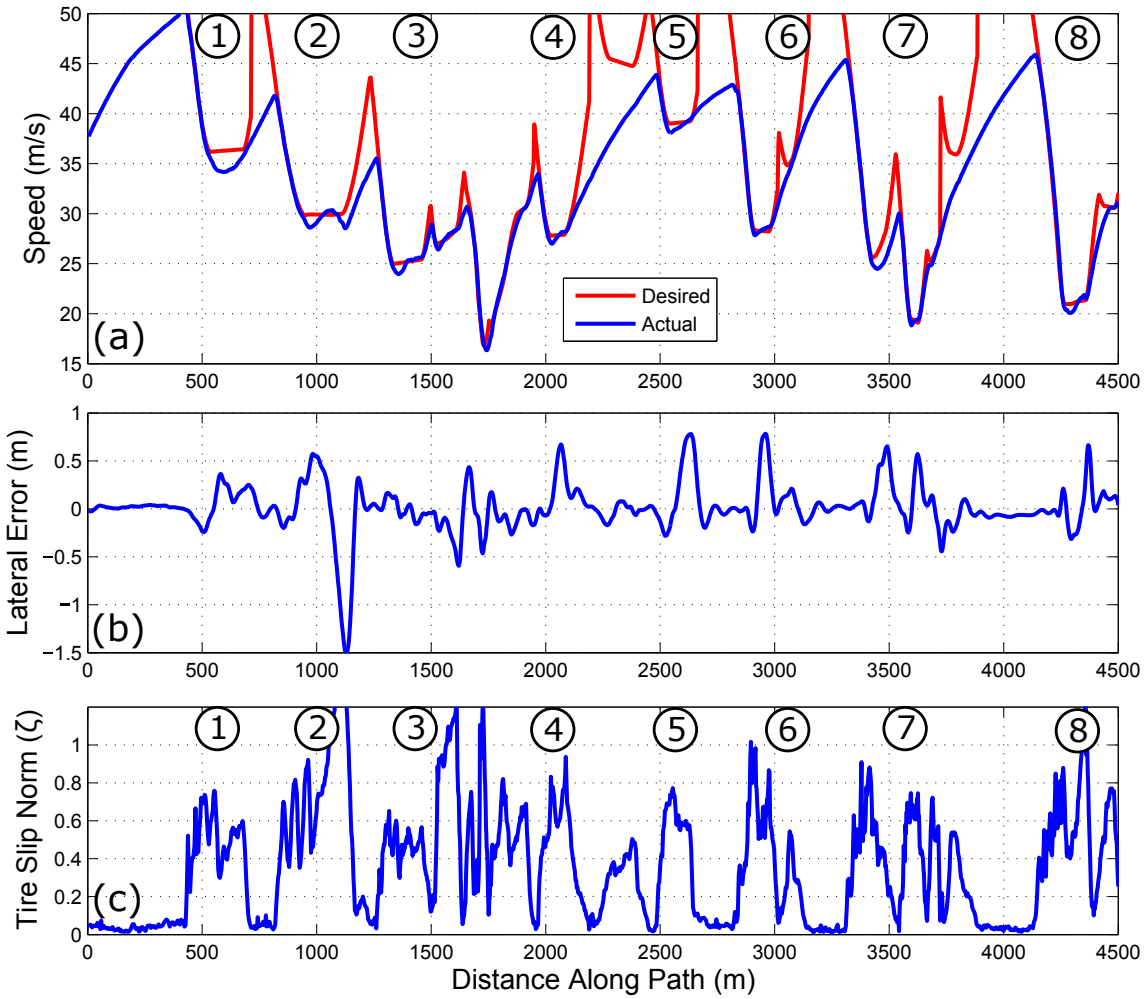


Figure 5.1: Reprint of Fig. 4.2. Controller tracking performance and tire slip norm on a test run at the limits of handling ($\mu = 0.95$).

As another situation, consider region (3). While the vehicle exceeds the friction limit on the prior section of the track, here the vehicle does not appear close to the limits at all, with $\zeta \approx 0.6$. A professional human driver would feel the vehicle being below the limits of handling and would increase her speed to decrease the time through the turn. However, ILC is only concerned with trajectory tracking, and would not raise the speed above the planned trajectory.

These two situations demonstrate the need for separate algorithms that learn from data in order to modify the desired *trajectory itself* as opposed to just the control

signals that attempt to track it. Trajectory modification algorithms previously investigated in the literature have focused on modifying the *lateral* trajectory by altering the curvature profile as the vehicle understeers or oversteers. For example, Theodosis presented an algorithm to gradually widen the radius of a turn in response to a detected understeer [73]. The algorithm was validated experimentally, but assumed sufficient availability of road width. Funke et al. [22] presented a model predictive control (MPC) approach that generally aimed to follow a planned vehicle trajectory at the limits. However, if the vehicle was at risk of understeering or oversteering, the MPC algorithm would deviate laterally from the planned trajectory in order to maintain stability of the vehicle without driving off the road.

This chapter presents an algorithm that takes a different approach. Instead of modifying the curvature profile, an A* search algorithm is presented that modifies portions of the *velocity* profile to be more conservative if a stability violation is encountered on a prior lap. This is accomplished by generalizing the speed profile such that each part of the track can be driven with a different assumed value of friction $\mu$ and therefore a different maximum acceleration. Since this dissertation is concerned with racing as well, the algorithm also modifies the velocity profile to be more aggressive if the tires are not being driven at the limits. Finally, instead of acting as a stability control algorithm that modifies the trajectory in real time, the presented algorithm relies on existing ESP algorithms for real-time stabilization and focuses on learning the time-optimal friction profile $\mu^\star(s)$ by searching through datasets obtained over multiple laps.

## 5.1 Effect of Tire Slip Norm on Lap Time

Fig. 5.2 shows the complicated effect of driving at different levels of lateral and longitudinal acceleration for region (2) in Fig. 5.1. Recall that the speed profile is generated by assuming a global value of tire friction $\mu$, which is directly related to the acceleration norm of the desired speed profile ($\sqrt{a_x^2 + a_y^2}$) by a factor of $g = 9.81$ m/s$^2$. Fig. 5.2(a) confirms that higher levels of $\mu$ result in strictly faster speed profiles for the same turn.

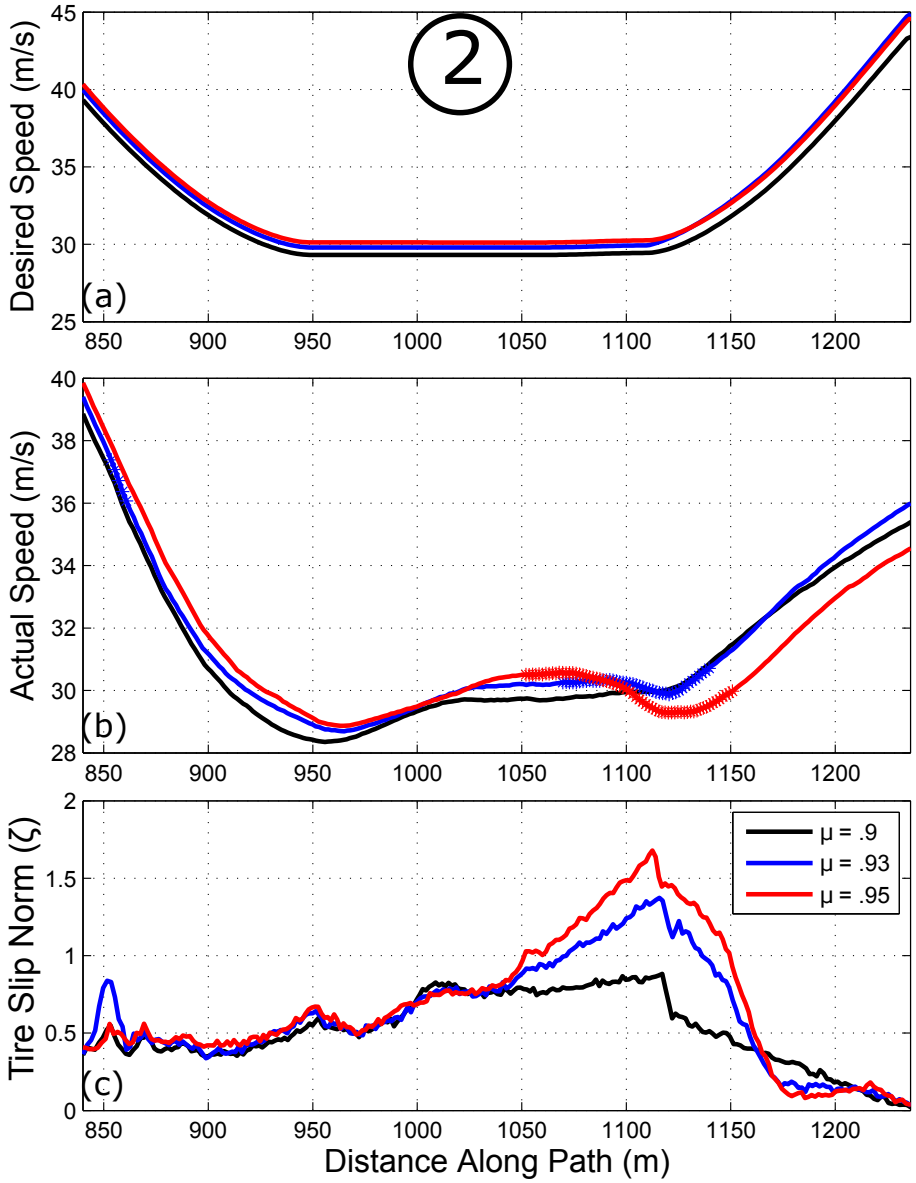


Figure 5.2: (a) Desired speed for varying levels of $\mu$ (b) Actual speed for varying levels of $\mu$. Asterisks correspond to regions where $\zeta > 1$. (c) Tire slip norm for varying levels of $\mu$.

However, selecting a more aggressive value of $\mu$ does not necessarily entail a faster experimental lap time. Consider the actual speeds of the vehicle in Fig. 5.2(b) when trying to experimentally follow three different speed profiles. For the case where $\mu = 0.9$, the vehicle completes the turn without fully utilizing the tire's capability and achieves relatively low velocities. For the extreme case where $\mu = 0.95$, the vehicle enters the turn at a high speed but then begins to slide as $\zeta = 1.6$. While not shown in the plot, the saturation is occurring primarily at the front tires, causing an understeer that can cause the car to skid off the track. Completing the lap therefore requires a stabilizing action (typically occurring through the vehicle's ESP system or similar black box algorithm) to slow the car down and regain control of the vehicle. As a result, when the vehicle accelerates at the end of the turn, the actual vehicle speed at $\mu = 0.95$ is significantly slower than the case where $\mu = 0.9$! A final "just right" value of $\mu = 0.93$ was also tested experimentally for this turn, and while the car does slide a bit at this level of driving (peak $\zeta = 1.3$), the needed stabilizing action is significantly smaller and the vehicle exits the turn with the highest speed.

Unfortunately, the best value of $\mu$ is not constant throughout the track. Fig. 5.3 shows the same data plotted for region (3) of the Thunderhill Raceway. In this case, even with an atypically high value of $\mu = 0.97$, the slip norm of the vehicle tires is relatively low, and the value of $\mu$ that optimizes the completion time for this turn could be even larger.

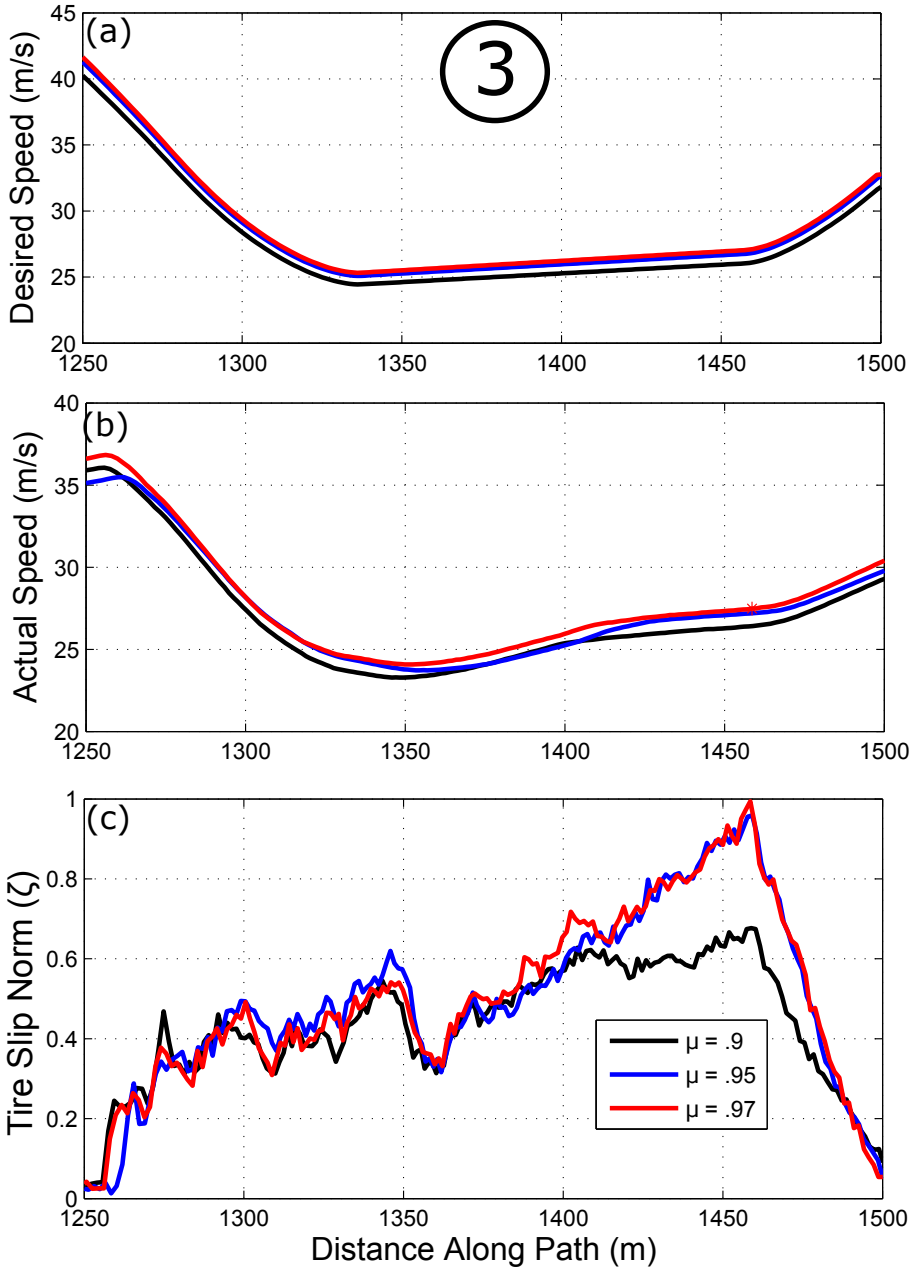


Figure 5.3: (a) Desired speed for varying levels of $\mu$. (b) Actual speed for varying levels of $\mu$. Asterisks correspond to regions where $\zeta > 1$. (c) Tire slip norm for varying levels of $\mu$.

## 5.2 Naive Method: Greedy Algorithm

Section 5.1 shows that to minimize the overall lap time, there is a need to generalize the speed profile such that different portions of the track can be driven with different values of $\mu$. In other words, **the problem is to learn the "friction profile" $\mu^\star(s)$ along the path that minimizes the experimental vehicle lap time.**

The simplest approach to finding $\mu(s)$ is a greedy algorithm where a set of experimental data is collected for a variety of different speed profiles, each corresponding to a different value of $\mu$ and therefore a different acceleration level. The greedy approach is then to discretize the track into a number of small sections and pick the value of $\mu$ for each section that corresponds to the highest observed experimental velocity. The final desired velocity profile is then generated using the numerical integration approach presented in §3.2. This method does not require a single value of friction across the whole track, and generates a smooth velocity profile even when the peak acceleration limits vary from point to point.

A plot showing the results of applying the greedy algorithm for section (2) is shown in Fig. 5.4. The flaw in simply selecting $\mu$ based on the highest speeds is apparent at $s = 1050$m. The greedy algorithm suggests the vehicle mostly drive at $\mu = 0.95$, but then switch to driving at $\mu = 0.93$ as soon as the vehicle begins to slide. Switching to a less aggressive velocity profile at this point is impossible to achieve in practice, because the vehicle is already fully sliding and has no control until stability is regained. As a result, the greedy algorithm fails to capture the hidden cost of a large understeer, which results in a period of time where the speed must inevitably drop.

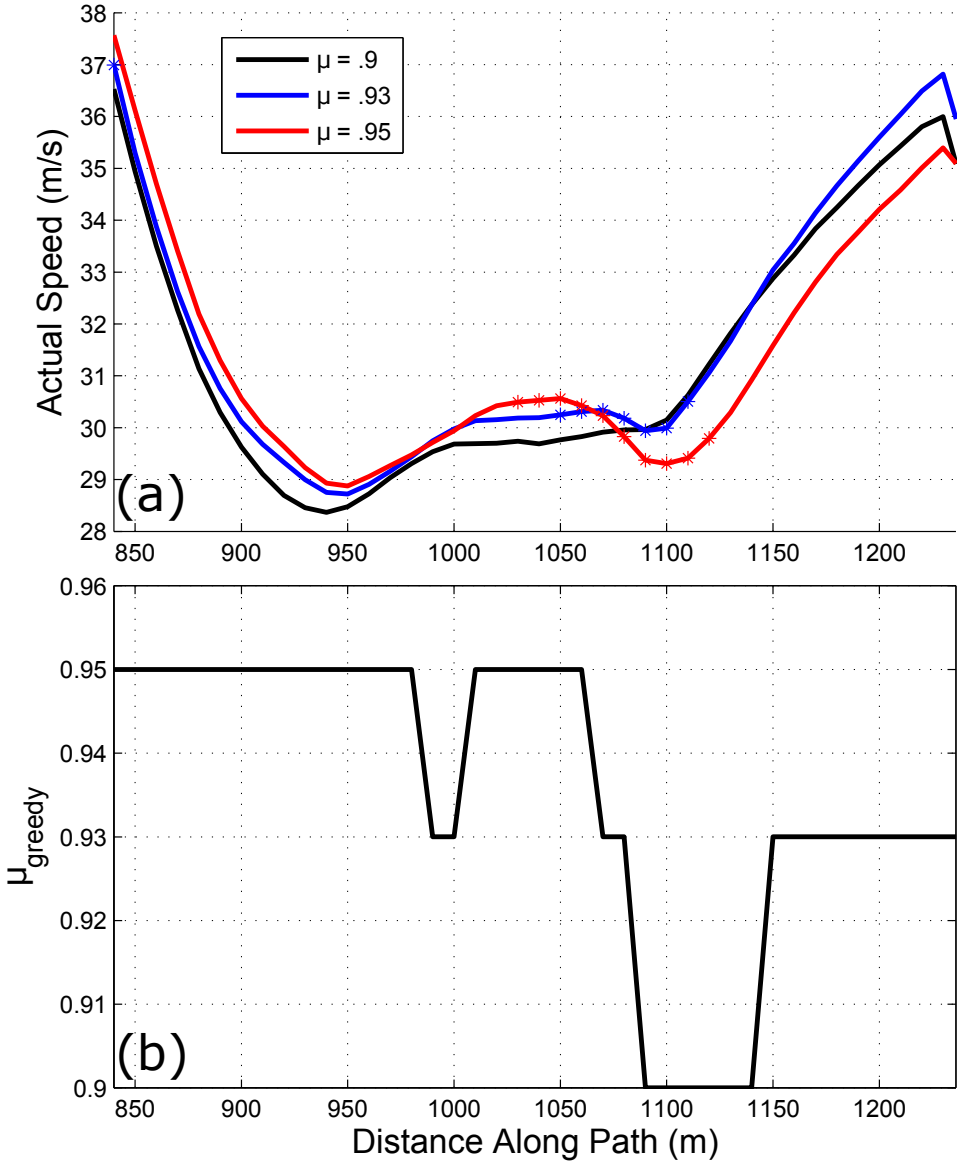


Figure 5.4: (a) Desired speed for varying levels of $\mu$ (b)"Greedy" value of $\mu$ as a function of distance along track. Asterisks denote region of track where vehicle is sliding.

## 5.3 Framing Trajectory Learning as Search Problem

Given the inadequacy of the greedy algorithm, a more sophisticated approach is necessary to learn $\mu^\star(s)$ from experimentally observed data. This section frames the desire to find the minimum time $\mu^\star(s)$ as a tree search problem. Consider discretizing the racing path into $N$ evenly spaced segments. For example, on the Thunderhill Raceway with $\Delta s = 5$m, $\mathbf{s} = [0\ \ 5 \ldots s_k \ldots 4495\ \ 4500]$ for $k = 0 \ldots N$, with $N = 901$. For each path distance $s_k$, there are $M_k$ velocity and $M_k$ tire slip observations from experimental data, each corresponding to a different $\mu$. For example, looking at Fig. 5.2, if $k = 191$, $s_k = 950$m, $M_k = 3$, the velocity and slip norm $\zeta$ observations $U_k(\mu)$ and $Z_k(\mu)$ are as follows:

Table 5.1: $U_k(\mu)$ and $Z_k(\mu)$ for $k = 191$

| $\mu$ | $U_x(m/s)$ | $\zeta$ |
|---|---|---|
| 0.9 | 28.42 | 0.55 |
| 0.93 | 28.91 | 0.63 |
| 0.95 | 29.14 | 0.66 |

$M_k$ is not necessarily the same for all $k$ to account for experimental trials that do not cover the full lap. For safety and time reasons, some parts of the track will only have experimental data collected at two different friction values, while others may have five or six.

Nodes of the search tree are then defined as a two-element state tuple, with the first state element being $s_k$ and the second element being the current friction coefficient $\mu_k$. Since the car must start from the beginning of the path, the first state has $s_0 = 0$. The $M_k$ edges from a given node correspond to actions that the car can take at $s = s_k$. In this case, the "action" is the next value of $\mu$, and the successor states are $(s_{k+1}, \mu_1)...(s_{k+1}, \mu_{M_k})$. A diagram of the search tree is shown in Fig. 5.5.

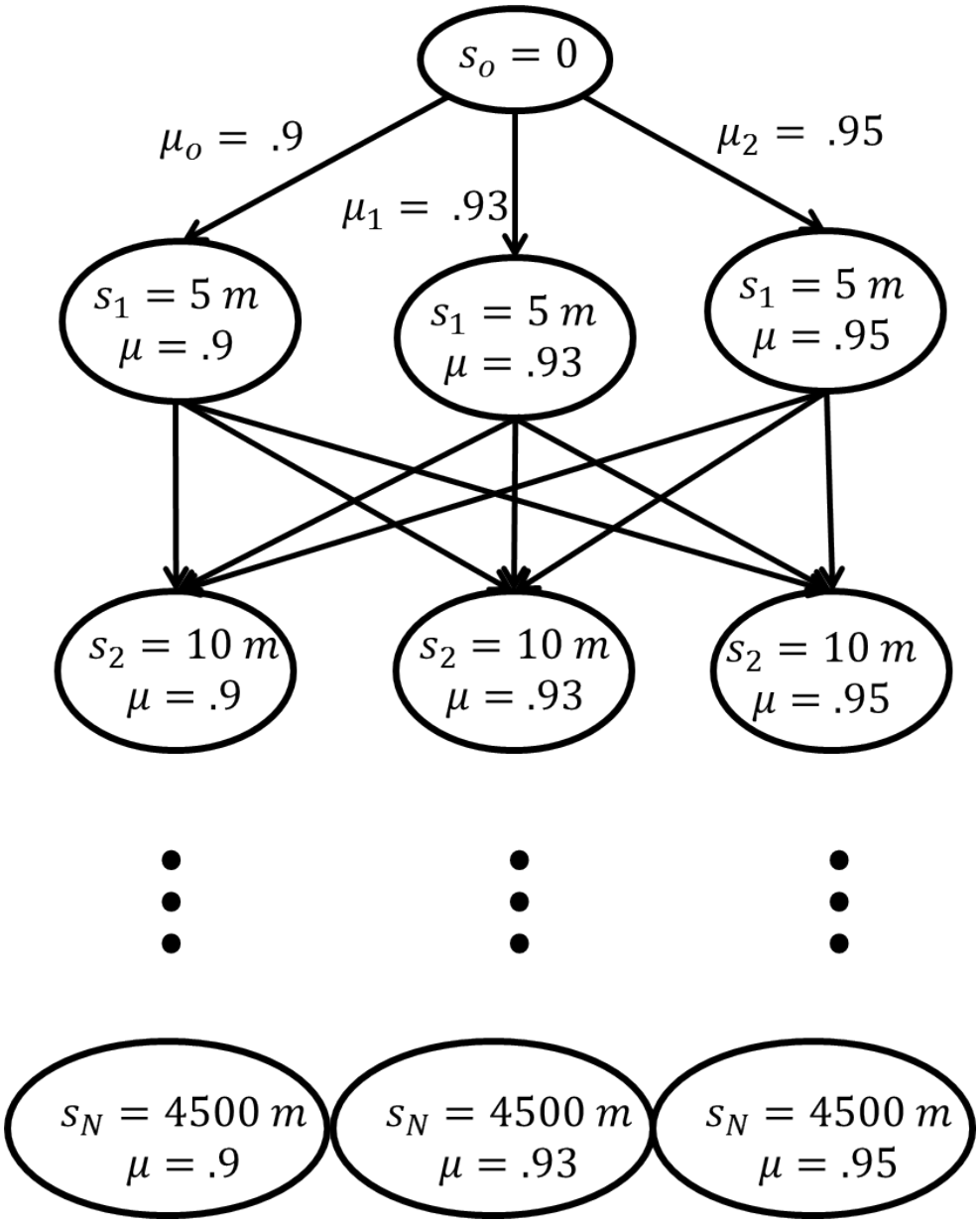


Figure 5.5: Sample search tree for Thunderhill race track where there are only 3 experimentally observed full laps at $\mu = 0.9, 0.93, 0.95$.

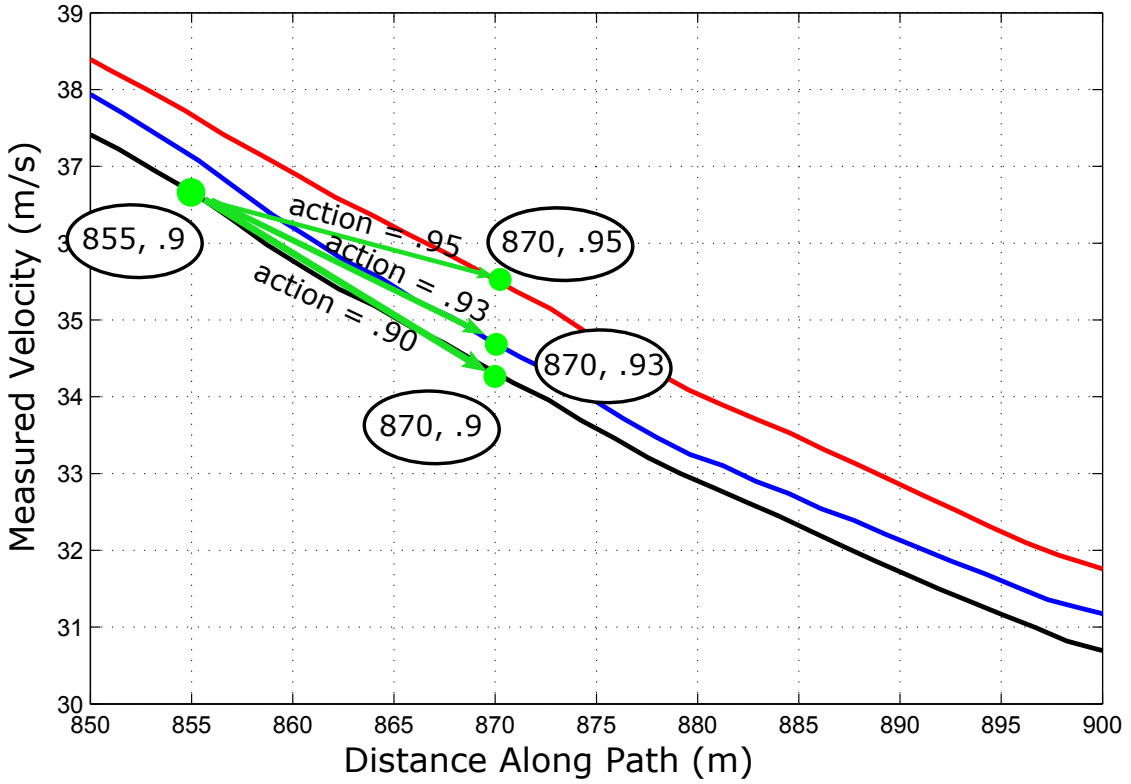


Figure 5.6: Illustration of travel cost. Current state is (855, 0.9). Assuming $\Delta s$ is 15 meters, cost is time to travel from this node to one of the three possible successor nodes, depending on the action taken.

Each edge is associated with a travel cost $c_t$. The travel cost for a given edge is the amount of time it takes to go from $s = s_k$ to $s = s_{k+1}$ driving at the friction coefficient $\mu$ associated with that edge. As illustrated in Fig. 5.6, the cost is estimated from the experimentally observed data assuming linear acceleration between points. The travel cost from node $(s_k, \mu_k)$ to $(s_{k+1}, \mu_{k+1})$ can therefore be expressed mathematically with trapezoidal integration of the straight-line velocity profile:

$$c_t(s_k, s_{k+1}, \mu_k, \mu_{k+1}) = \ln \frac{a_x(\Delta s) + U_k(\mu_k)}{a_x} - \ln \frac{U_k(\mu_k)}{a_x} \tag{5.1}$$

$$a_x = \frac{U_{k+1}(\mu_{k+1}) - U_k(\mu_k)}{\Delta s} \tag{5.2}$$

In addition to the travel cost, there is also a *switching* cost $c_s$ associated with

switching to a different value of $\mu$. These are determined by the observed tire slip norm measurements $Z_k(\mu)$. The switching cost is expressed mathematically as:

$$c_s(s_k, s_{k+1}, \mu_k, \mu_{k+1}) = \mathbb{1}\left(\mu_{k+1} \neq \mu_k\right)\left(\lambda + \infty \times \mathbb{1}\left(Z_k(\mu_k) > 1\right)\right) \tag{5.3}$$

Where $\mathbb{1}$ is the indicator function. The switching cost function (5.3) implies that an action will never incur a switching cost if the selected value of $\mu$ is unchanged from the previous selection. If the value of $\mu$ does change, there is a small switching penalty $\lambda$ to discourage the search algorithm from changing the friction profile to gain a trivial decrease in lap time. Additionally, there is a very large (infinite) switching penalty if the search algorithm attempts to change the friction profile while the vehicle was observed to be sliding ($\zeta > 1$). This reflects the physical inability of the car to control its trajectory while sliding and is what separates the search algorithm from the greedy algorithm. A diagram demonstrating (5.3) is shown in Fig. 5.7.

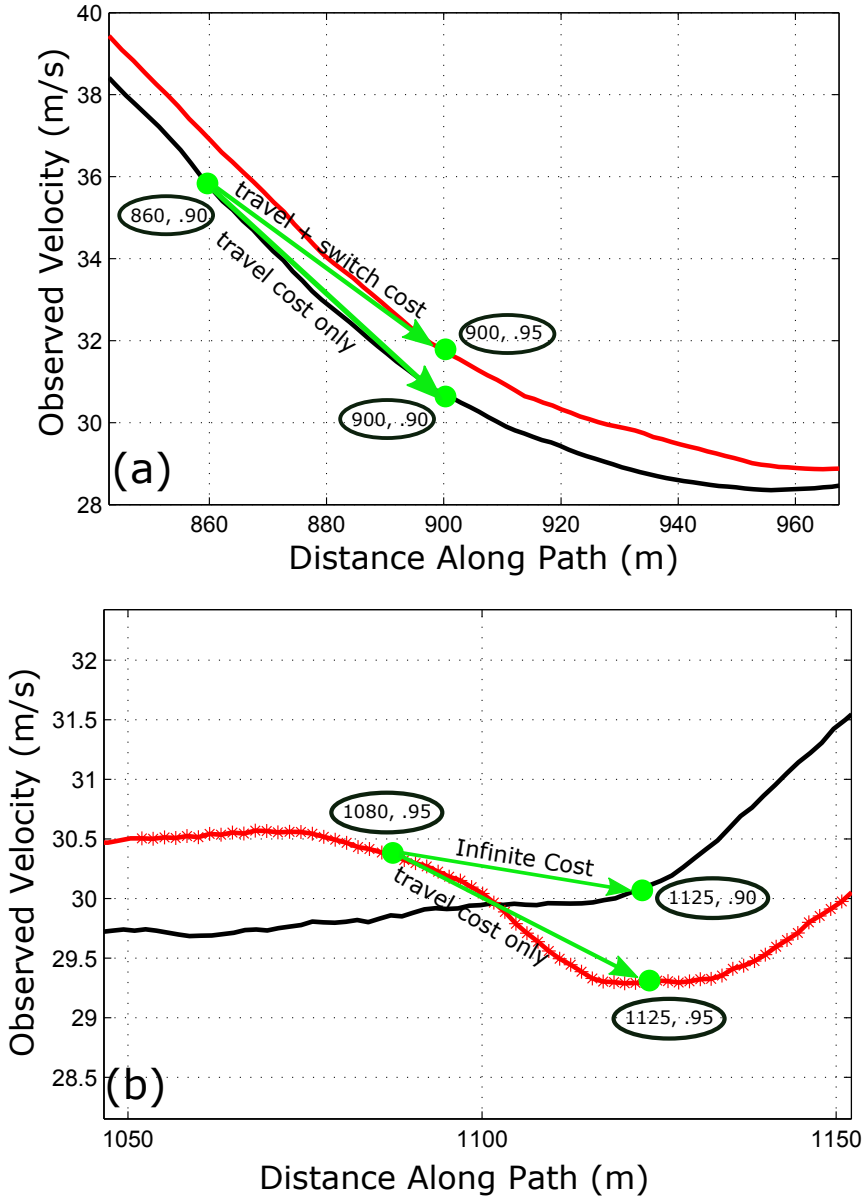


Figure 5.7: (a) Costs when vehicle currently is not sliding. The vehicle can switch to a more aggressive velocity profile, paying a small switching penalty, or can continue on the current profile. (b) Costs when vehicle is currently sliding. Vehicle has no choice but to continue on current trajectory. Asterisks denote regions where $\zeta > 1$ and the vehicle is sliding.

## 5.4 A* Search Algorithm and Heuristic

Once the tree is mathematically defined in terms of the start state, nodes, edges, and costs, the search problem is to find the sequence of actions from the start node to any terminal node that minimizes the total cost. In our case, the start state, nodes, edges and costs were defined in the previous section, and a terminal node is any node at the end of the path (i.e. $k = N$). The sequence of actions in our case is the friction profile $\mu = [\mu_1 \dots \mu_k \dots \mu_N]$, which determines how aggressively the vehicle will drive on every part of the track. The total cost is the sum of all individual travel costs $c_t$ and switching costs $c_s$, and has intuitive units of time.

Minimum-cost tree spanning algorithms (e.g. breadth-first search, Dijkstra's algorithm, etc.) are a well known subject and a thorough description can be found in [69]. For the purpose of solving this search problem, the A* search algorithm is used. Like breadth-first search, the A* algorithm is guaranteed to find the lowest cost path from a start node to a goal node, but uses a priority queue data structure to more efficiently explore the search tree. Frontier leaf nodes $n$ being considered for exploration are ranked according to the following modified cost:

$$f(n) = g(n) + h(n) \tag{5.4}$$

where $g(n)$ is the true cost to go from the start node to node $n$. The function $h(n)$ is a heuristic estimate of the cost to get from node $n$ to any goal state (i.e. to the end of the path). For A* to be guaranteed to find the shortest path, $h(n)$ must be *admissible*, meaning that $h(n)$ must underestimate the true cost of getting to the end of the path.

In our case, we have a very intuitive heuristic function $h(n)$ for the A* implementation. In Sec. 5.2, the greedy approach was discussed. Define
$\mathbf{U}_\mathrm{g} = [U_\mathrm{g}(1) \dots U_\mathrm{g}(k) \dots U_\mathrm{g}(N)]$ to be the highest observed experimental speed for each index $s_k$. Because we know tracking this greedy profile is physically impossible due to vehicle sliding, time estimates from this profile will always underestimate the true cost. We therefore define $h(n)$ as follows:

$$h(n) = \int_{s_n}^{s_N} \frac{1}{U_g(s)} ds \tag{5.5}$$

Where $s_n$ is the value of $s$ corresponding to node $n$ and $s_N$ is the total length of the track. While (5.5) is an integral equation, it can be solved for the discrete array $\mathbf{U}_{\mathrm{g}}$ via trapezoidal numerical integration.

## 5.5 A* Implementation and Results

Because the A* algorithm relies on experimental observations to learn $\mu^\star(s)$, experimental data was collected over several trials, with each trial consisting of a speed profile generated with a constant $\mu$ value over the track. The $\mu$ values chosen for experimental data collection were 0.85, 0.9, 0.92, 0.93, 0.94, 0.95, and 0.97. Ideally, each speed profile would be tested experimentally for a full lap at high speed. However, due to safety and time constraints associated with collecting high speed race data, only the speed profiles corresponding to $\mu = 0.92$ and $\mu = 0.94$ were observed over the whole track. The other speed profiles were only tested on sections of the track. Fig. 5.8 shows the experimental data coverage.

Table 5.2: Search Algorithm Information

| Parameter | Symbol | Value | Units |
|---|---|---|---|
| Track Length | $L$ | 4500 | m |
| Discretization Length | $\Delta s$ | 5 | m |
| Number of Points | $N$ | 901 | - |
| Switching Cost | $\lambda$ | 0.05 | s |
| CPU Solution Time | | 70 | s |
| Nodes Explored | | 6887 | - |

After collecting the experimental data, the A* algorithm was applied to learn the optimal friction profile $\mu^\star(s)$. Parameters for the algorithm are shown in Table 5.2. To interface more efficiently with experimental data files, the search algorithm was implemented in MATLAB using custom code instead of a standard search algorithm library. The entire search process took approximately 70 seconds on a core i7 laptop

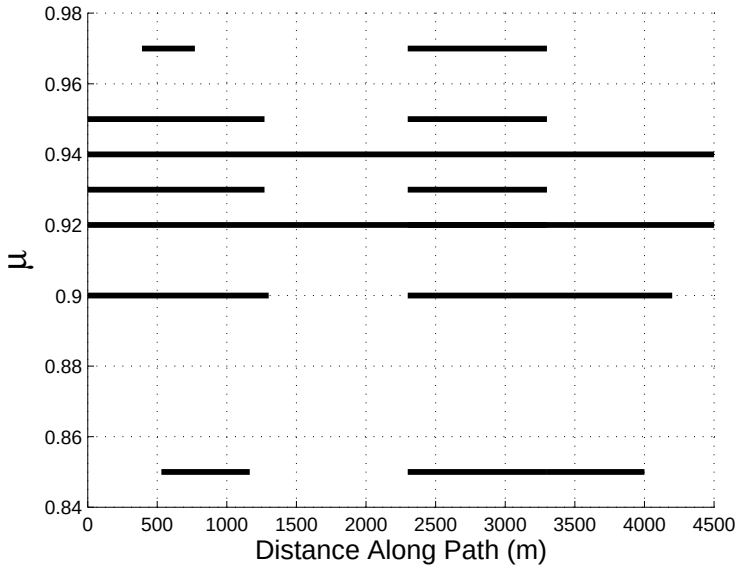


Figure 5.8: Coverage of experimental data. For speed profiles corresponding to $\mu = 0.92$ and $\mu = 0.94$, experimental data of the autonomous race vehicle was observed over the whole track. For safety and time constraints, the other speed profiles were only tested on sections of the track.

machine, exploring 6887 nodes in the process. However, since MATLAB is generally a poor software environment for tree-based search algorithms, a C++ implementation would likely be several orders of magnitude faster.

The resulting $\mu^\star(s)$ profile associated with the minimum lap time on Thunderhill Raceway is shown in Fig. 5.9. For comparison, the A* solution is plotted against the greedy algorithm solution. Because of the incorporation of switching costs, the A* algorithm switches $\mu$ values only when necessary to achieve a nontrivial increase in lap time, and only switches $\mu$ when the vehicle is not sliding. The same results are plotted on a map of the track in Fig. 5.10. A satisfying observation is that the optimal profile reduces $\mu$ to 0.93 in section (2) to be more conservative and increases $\mu$ to 0.97 in section (3) to be more aggressive. This matches our observations about tire slip norm $\zeta$ in Sec. 5.1.

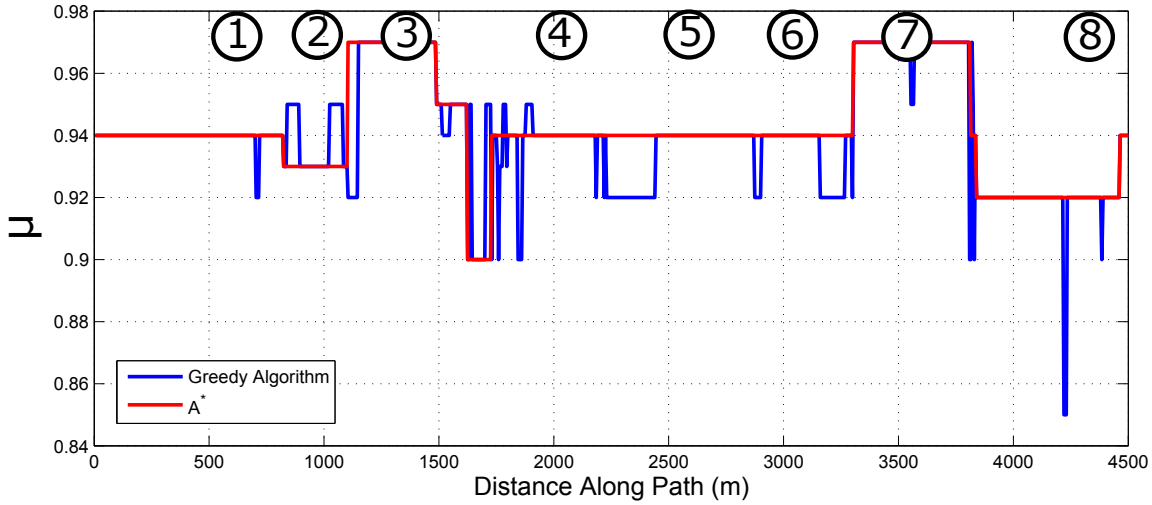


Figure 5.9: Minimum time $\mu(s)$ profile for Thunderhill, for both the A* solution and greedy algorithm solution.

Predicted lap time results are shown in Table 5.3. Notice that the A* predicted lap time is slightly slower than the greedy algorithm prediction, which is expected given the physical infeasability of the greedy algorithm assumptions. The results also indicate that a significant lap time improvement can be expected over a velocity profile generated with a constant $\mu$. In fact, experimentally driving at $\mu^{\star}(s)$ could even result in a lap time faster than a professional human driver.

Table 5.3: Lap Times

| Driver | Lap Time |
|---|---|
| Constant $\mu(s) = 0.94$ | 139.2 |
| Constant $\mu(s) = 0.92$ | 139.4 |
| Pro Driver (Best Lap) | 137.7 |
| Greedy $\mu_g(s)$ (Prediction) | 136.4 |
| A* $\mu^*(s)$ (Prediction) | 136.8 |

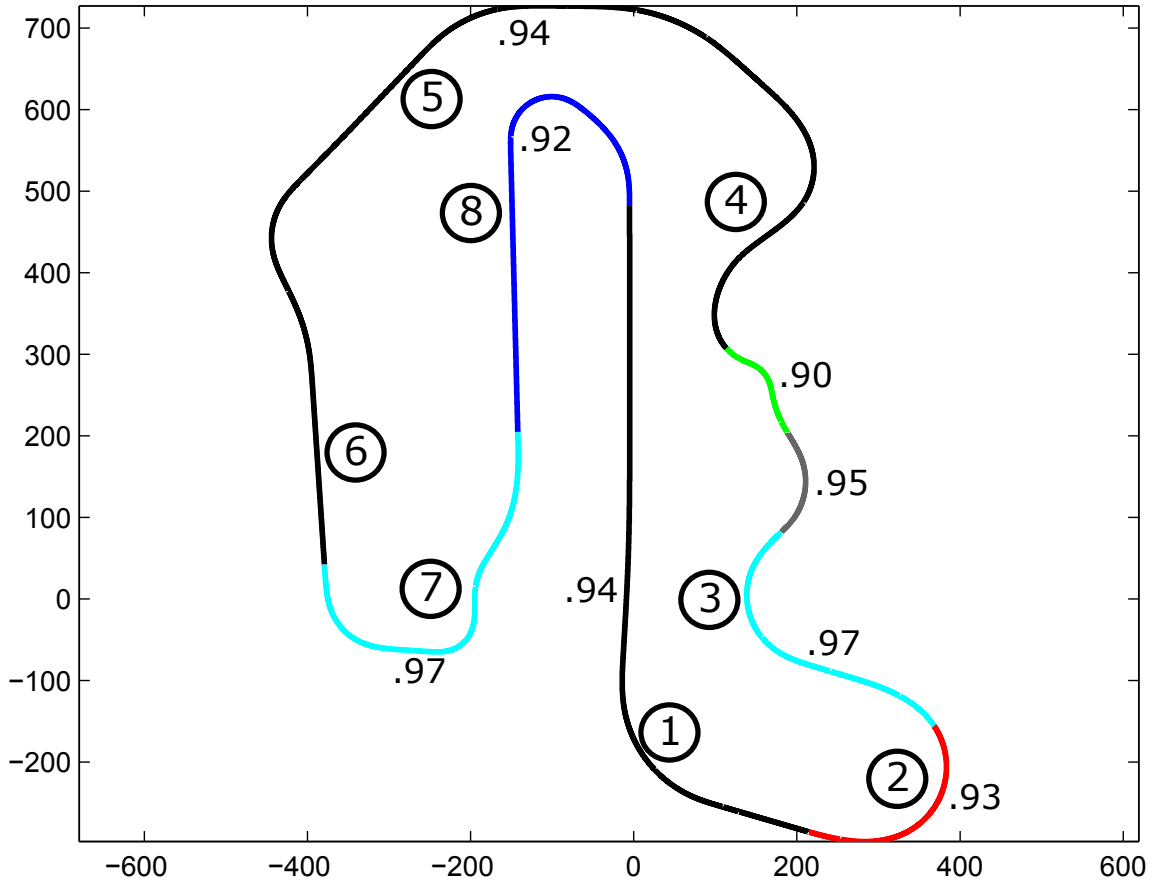


Figure 5.10: Minimum time $\mu^\star(s)$ profile from A* algorithm plotted on map of Thunderhill.

## 5.6 Experimental Validation

The best validation of the A* algorithm is to experimentally drive the optimal velocity profile $U_x^\star(s)$ generated from $\mu^\star(s)$. From Fig. 5.9, this velocity profile will have accelerations as low as 9.0 m/s$^2$ on some turns, and as high as 9.7 m/s on others. Fig. 5.11 shows autonomous experimental data from driving $\mu^\star(s)$ compared to two other experiments[1]. The first experiment is a full autonomous test with a constant $\mu =$

[1]There was a minor change between the optimal $\mu$ profile from Fig. 5.9 and what was tested experimentally. For section (2), the value of $\mu$ was set to 0.92 as opposed to 0.93. This was a safety measure taken based on preliminary testing.

0.94, and the second experiment is the fastest single lap recorded by the professional race car driver in Fig. 4.1.

The experimental results show solid performance of the learned friction profile $\mu^\star(s)$. From Fig. 5.11(a), the lap time using the learned friction profile is roughly 1.5 seconds faster than the lap time from the constant friction profile. The lap time is also comparable to the fastest recorded lap time of the pro driver. A significant part of this improved performance comes through more efficient friction usage. For example, in sections (2) and (8), learning to drive more cautiously enables the tire slip norm $\zeta$ to drop closer to 1, avoiding a costly understeer. In sections (3) and (7), the A* algorithm has learned that more aggressive driving is possible, increasing $\zeta$ closer to 1 and matching the tire slip norm of the professional driver.

However, there are several caveats associated with Fig. 5.11 to disclose. Due to time constraints, the three datasets were all taken on different dates, meaning that weather, tire conditions, etc. were different for each test. Second, the two autonomous datasets were taken nearly a year apart, and as a result, there were independent improvements made to the vehicle controllers that also contribute to the 1.5 second experimental lap time improvement. For example, the higher speed in section (1) (see Fig. 5.11(a)) was due to a more tightly tuned longitudinal controller and not a difference in the desired speed profile. Finally, because autonomous racing is performed with nobody in the vehicle, the human driver has the disadvantage of both his added mass and the added mass of a graduate student. Since the available engine force is limited, this decreases the available acceleration on straight sections of the track by roughly 10%. This gives a significant time boost for any autonomous lap over a human-driving lap, as seen by the higher top speeds in Fig. 5.11(b).

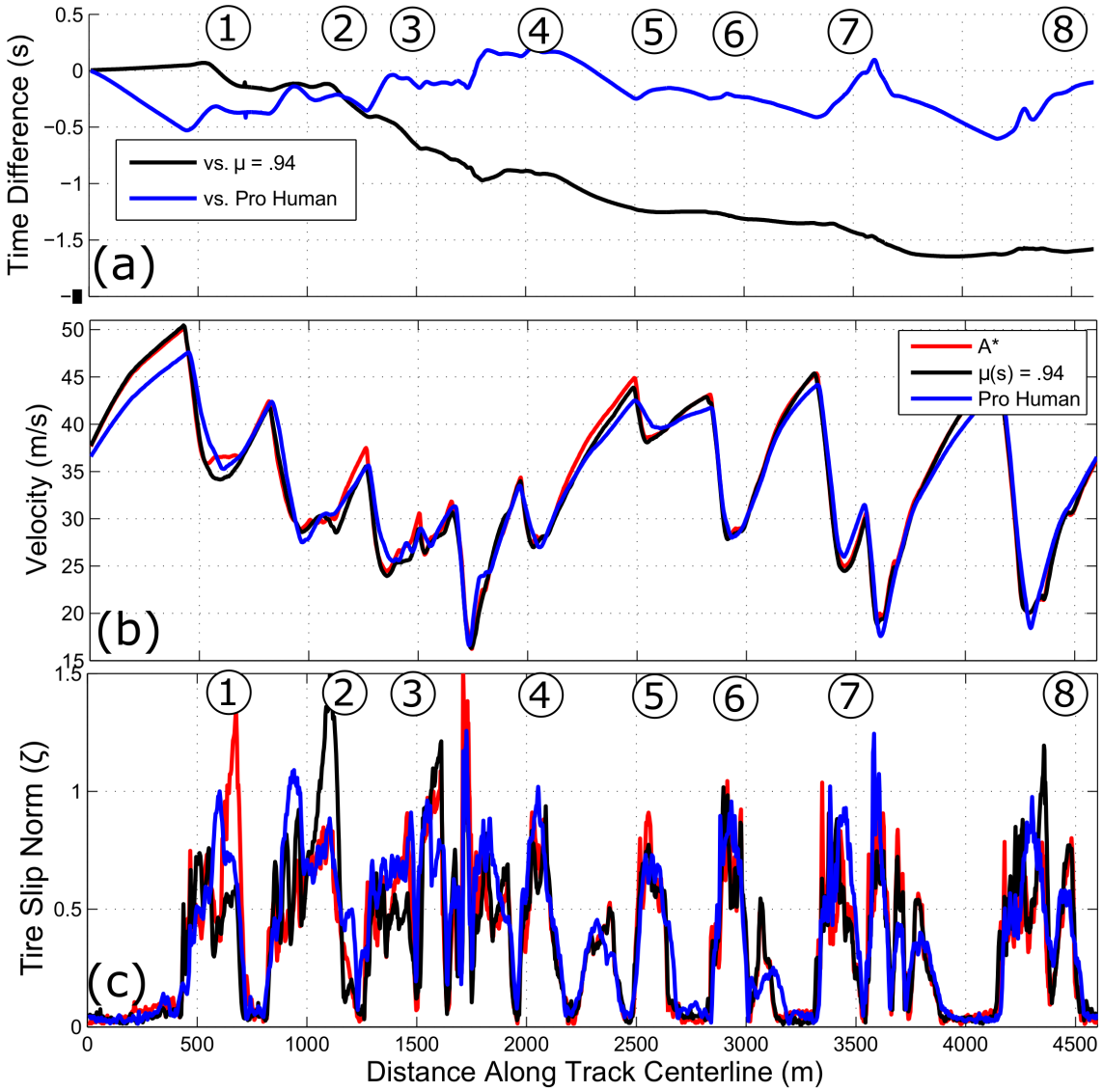


Figure 5.11: (a) Time difference between experimental dataset collected with A* result $\mu^*(s)$ and dataset from professional driver. Also included is comparison with constant friction velocity profile at $\mu = 0.94$. Negative time distance corresponds to A* result being ahead. (b) Experimental velocities between all three experimental datasets. (c) Tire slip norm measurements for all three datasets.

## 5.7 Future Work

There are a several next steps to improve the preliminary research presented in this chapter. First, the algorithm presented here assumes the availability of pre-existing data. For a track where existing data at the handling limits is unavailable, the algorithm should be modified so that laps are driven at a low assumed friction value (i.e. $\mu = 0.9$) and slowly ramped up, with the A* algorithm used after every lap to find portions on the track where the vehicle could benefit from a more aggressive acceleration profile.

Finally, the algorithm makes the key assumption that observations made on prior laps will hold for upcoming laps. However, when data is collected for the same velocity profile multiple times, the resulting observed speeds and tire slips will vary. Furthermore, the tire slip norm $\zeta$, defined in Chapter 4, is a noisy empirical estimate of whether the vehicle is actually sliding. Further work is necessary to add uncertainty to the model. For example, preliminary work is underway to treat experimental tire slip measurements as noisy indicators of whether the vehicle has exceeded the limits. This would complement the existing literature for real-time vehicle decision making under uncertainty, which has considered uncertainty in sensor noise, perception constraints, and the behavior of surrounding vehicles and pedestrians [4][77][82]. Accounting for state uncertainty by developing a Partially Observable Markov Decision Process (POMDP) is therefore a promising avenue for future work.

## 5.8 Conclusion

This chapter presented an algorithm to improve the experimental lap time of an autonomous race car by learning the optimal friction profile, and therefore the optimal desired speed and acceleration profiles. The approach works by searching through a tree built up from experimentally collected observations and finding the fastest speed profile via an A* implementation. Edge costs for this tree are given by travel time calculations and a switching cost that accounts for the difficulty of speed control while the vehicle is understeering or oversteering. The results compared well experimentally

to an autonomous dataset from a uniform $\mu$ profile and against an experimentally recorded dataset from a professional driver.

The significance of this work is that the autonomous race vehicle is no longer required to race with a single predetermined estimate of the tire-road friction. Experimental data indicates that in reality, each turn on the track has a slightly different acceleration limit that enables the autonomous vehicle to minimize travel time without sliding off the track. Instead of naively guessing an average friction/acceleration limit for the entire racing circuit, the presented algorithm allows the vehicle to search through data obtained from previous laps and find an optimal friction *profile* that varies along the track.

# Chapter 6

# Conclusion

Inspired by automobile racing, this dissertation documented several contributions for trajectory planning and control at the limits of handling. Chapter 2 presented a feedback-feedforward steering controller that simultaneously maintains vehicle stability at the limits of handling while minimizing lateral path deviation. Section 2.2.1 demonstrated the relatively poor path tracking performance of a baseline lookahead steering controller. In §2.4, analytical results revealed that perfect path tracking performance occurs only when the vehicle sideslip angle is held tangent to the desired path. This desired sideslip behavior was incorporated into the feedforward control loop to create a robust steering controller capable of accurate path tracking and oversteer correction at the physical limits of tire friction (§2.5). Experimental data collected from an Audi TTS test vehicle driving at the handling limits on a full length race circuit (§2.6) demonstrated the desirable steering performance of the final controller design.

Chapter 3 presented a fast algorithm for minimum-time path planning that divided the path generation task into two sequential lateral and longitudinal subproblems that were solved repeatedly. The longitudinal subproblem, described in §3.2, determined the minimum-time velocity profile given a fixed curvature profile. The lateral subproblem updated the path given the fixed speed profile by solving a convex optimization problem that minimized the curvature of the vehicle's driven path while staying within track boundaries and obeying discretized equations of motion (§3.3).

Experimental lap times and racing lines from the proposed method were shown to be comparable to both a nonlinear gradient descent solution and a trajectory recorded from a professional racecar driver (§3.6). The cost function for the path update subproblem was also improved by incorporating a distance minimization term in §3.7.

Finally, Chapters 4 and 5 presented two approaches to gradually refine the driving performance of the autonomous race car over time. Chapter 4 developed two iterative learning control (ILC) formulations that gradually determined the proper steering and throttle inputs to precisely track the desired racing trajectory. In §4.1, simulation and analytical results were used to design and test convergence of proportional-derivative (PD) and quadratically optimal (Q-ILC) iterative learning controllers. Experimental results at combined vehicle accelerations of 9 $m/s^2$ indicate that the proposed algorithm can rapidly attenuate trajectory following errors over just two or three laps of racing (§4.6). Chapter 5 presented a tree-search algorithm to minimize experimental lap times by learning different acceleration limits for each turn on the track. An A* search algorithm was devised in §5.3 to search through experimental data and find the best value of $\mu$ for each portion of the track in order to globally minimize the resulting lap time. Key developments of this algorithm include designing an appropriate A* heuristic (§5.4) to minimize the needed computation time and designing the cost function to account for the physical difficulty of altering the vehicle's trajectory while understeering or oversteering.

## 6.1 Future Work

The dissertation concludes with a discussion of both future work and applications of the research to commercial automotive safety systems.

#### Feedback-Feedforward Steering Controller

One drawback of the feedback-feedforward steering controller in Chapter 2 is the reliance on steady-state feedforward estimates of vehicle states. This will cause issues for tracking highly transient trajectories, such as those encountered in obstacle avoidance maneuvers [21]. Furthermore, because the sideslip dynamics are captured only at the

*feedforward* level to ensure robust stability margins, the steering controller becomes sensitive to modeling errors between the actual vehicle system and the steady-state model.

There are several avenues for future work to improve the path tracking performance of the steering controller. One possibility to improve robustness to plant modeling errors is to come back to the feedback controller that directly incorporates vehicle sideslip measurements in the feedback control law (2.12). This controller was shown to have excellent path tracking characteristics. However, the controller suffered from poor stability margins at the handling limits, and a compromise was ultimately selected where steady-state predictions of the vehicle sideslip were used instead. A promising solution is to use a blend of measured and predicted sideslip values. For example, a model-predictive controller could switch between measured or estimated vehicle sideslip in (2.12) depending on whether there is significant risk of the vehicle approaching the handling limits.

A second avenue for future work is to eliminate transient path tracking errors by tightening the lanekeeping controller gains. Funke [21] noted that transient dynamics become significant when avoiding obstacles at the limits of handling. An LQR approach for gain selection revealed that tighter path tracking is possible if the gain on heading error $\Delta\Psi$ is significantly shortened. Understandably, the drawback of this tighter path tracking is higher levels of steering input, typically resulting in high frequency twitches of the steering wheel. Again, an MPC controller could manage this tradeoff between smoother steering inputs and path tracking error. More of the standard lookahead controller could be used in normal steady-state driving situations, and tighter path tracking gains would be used in transient obstacle avoidance scenarios.

### Rapid Path Generation

The primary difficulty with the trajectory generation method from Chapter 3 is the need to balance minimizing path curvature and path length. Without the computational expense of directly minimizing lap time or applying a trial-and error method, it is difficult to determine the areas of the track where minimizing distance is important.

The presented method of learning optimization weights from human data provides a quick solution, but professional driver data is not always available for a given racing circuit. To manage the tradeoff, it may be beneficial to develop an *anytime* algorithm that starts by globally minimizing curvature with a cheap convex optimization step, and then uses the remaining computational time to refine specific turns where minimizing curvature is unlikely to be the best approach. This would most likely rely on general heuristics learned for racing in general, and not just for a specific track. For example, on sequences of alternating left/right turns, only minimizing curvature may not be the best approach.

The second area for future work is transferring the presented algorithm onto an embedded computer for real-time trajectory planning. This enables the controller to account for real-time changes such as competing race vehicles and updated estimates of tire friction. This requires two steps. First, the convex optimization code for the path update step must be written in a language such as CVXGEN [50] that is suitable for real-time computing. Second, given hardware restrictions on the size of optimization problems for embedded computing, the optimization algorithm must be modified into a "preview" controller that optimizes the next several turns instead of the entire track. The feasibility of this approach has been confirmed with a preliminary analysis, which showed that an optimization over 500 meters of track could be completed on the order of milliseconds using CVXGEN.

### Iterative Driving Improvement

Chapters 4 and 5 presented two interesting preliminary methods for an autonomous race car to learn how to drive better. Chapter 4 focused on improving tracking of a desired trajectory via iterative learning control, while Chapter 5 focused on modifying the *longitudinal* component of the planned trajectory based on experimental observations of tire utilization and vehicle speeds. These two approaches should be combined and tested together, so that the vehicle begins with a preliminary trajectory planned with a conservative assumed friction value, and then slowly learns how to track that trajectory while simultaneously making the trajectory faster on the turns where tire slips are lower than predicted.

A key prerequisite for achieving this is the ability to perform the learning algorithms in real time. Both learning algorithms currently operate offline after a lap (or several laps) have already been recorded, and typically take 30 - 60 seconds of computing time in MATLAB. C++ implementation of the algorithms, particularly the tree-search approach from Chapter 5, could provide a significant computational speedup. Improvements in algorithm efficiency and parallelization would enable a system where learning is continuously performed on a separate processor during the autonomous run itself. This would enable a futuristic system where the autonomous race vehicle could run uninterrupted for five or ten laps, improving the lap time each lap through iterative learning control and trajectory modification.

## 6.2 Applications for Future Automotive Safety Systems

Automobile racing is a fascinating subject, and the quest for an autonomous vehicle that can compete with the best human drivers is akin to to the search for chess algorithms in the 1970's and 80's that could defeat a grandmaster. While automobile racing typically occurs at speeds and accelerations much higher than those seen on commercial highways, trajectory planning and control algorithms for an autonomous race car have significant potential benefits for future autonomous passenger vehicles. In the same way that the race to beat the best human chess players inspired a new generation of broadly applicable artificial intelligence techniques, designing a fast race vehicle offers a new generation of technology for future commercial passenger safety systems. In fact, transfer of technology from the race car to the passenger automobile is nothing new, and everyday automotive technology ranging from direct-shift gearboxes to the modern disc brake can be traced to innovations in race technology [15].

### Steering Controller

The presented feedback-feedforward steering controller is immediately ready for application in commercial autonomous driving features. The required inputs of the algorithm are relatively simple to obtain. The controller requires knowledge of a desired speed and curvature profile, available from any high level trajectory planner that computes smooth driving profiles, such as the high level planner for Stanford's 2008 "Junior" DARPA Urban Challenge Vehicle [52]. Additionally, the controller requires knowledge of the vehicle error states, namely the deviation from the desired path and the heading error from the desired path. While the Audi TTS used for experimental validation obtains these precisely from DGPS technology, this technology is not viable for commercial driving. However, there has been a large research effort on vehicle localization relative to a known map via sensor fusion of commercially available sensors such as standard GPS/INS, LIDAR, and cameras, resulting in localization accuracy suitable for autonomous driving [36].

If incorporated in a passenger automobile, the feedback-feedforward algorithm would be able to achieve accurate and smooth driving in non-emergency situations. Common issues frequently reported on autonomous vehicles such as steering wheel twitches could be avoided along with significant lateral path deviation. Most importantly, in the event of an autonomous safety maneuver at the handling limits, the controller could follow an emergency trajectory without losing stability or deviating off the desired trajectory into an obstacle. Furthermore, the steering response would be smooth and non-oscillating, giving the human passengers confidence in the capability of their vehicle.

### Rapid Trajectory Planner

The rapid trajectory planner in Chapter 3 also offers potential for a commercial autonomous safety system. The algorithm could be reformulated as a high level trajectory planner for the next several hundred meters of open road instead of over an entire closed-circuit race track. Combined with LIDAR or other sensor information on

the presence of obstacles, the objective of the lateral planner could be to avoid an upcoming obstacle while staying on the road, a framework first proposed by Erlien et al. [18] for shared human/computer control. However, instead of a shared controller minimizing deviation from the driver's steering command, this trajectory planner would autonomously avoid obstacles while attempting to maintain a minimum curvature path. The benefit of minimum-curvature obstacle avoidance is increased safety margins. By maximizing the permissible collision-free speeds the car can safely drive at, the envelope of safe driving trajectories is increased. Furthermore, in non-emergency scenarios, the trajectory planner can plan driving paths below the limits by driving through desired waypoints on the road with minimum curvature for driver comfort.

### Lap-to-Lap Learning

The algorithms for iteratively improving autonomous driving performance will be more difficult to apply in real-world situations, simply because most passenger driving doesn't occur over the same closed-circuit race course. However, there is an emerging trend towards automation in all aspects of society, and several industrial companies have expressed interest in iterative learning algorithms for repetitive driving maneuvers. For example, manufacturers of agricultural equipment have sponsored iterative learning research for heavy vehicles that can precisely repeat the same driving pattern for applications such as fertilizer and seed deployment [49]. Beyond agriculture, similar applications for iterative learning could include autonomous tour vehicles or industrial vehicles for applications such as mining. Additionally, many drivers drive on similar roads every day for commuting or other routine trips. Advances in iterative learning could enable applications where an automated system detects a routine commute and learns from human driver data in order to replicate the driving style of a specific passenger over time. This would be a vital step in gaining acceptance of autonomous vehicles as passengers would be more comfortable with a self-driving car that matches their own driving style.

# Bibliography


[1] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[2] Michele Bertoncello and Dominik Wee. Ten ways autonomous driving could redefine the automotive world. http://www.mckinsey.com, June 2015.

[3] F Braghin, F Cheli, S Melzi, and E Sabbioni. Race driver model. *Computers & Structures*, 86(13):1503–1516, 2008.

[4] S. Brechtel, T. Gindele, and R. Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 392–399, Oct 2014.

[5] Douglas A Bristow and Brandon Hencey. A Q,L factorization of norm-optimal iterative learning control. In *47th IEEE Conference on Decision and Control (CDC)*, pages 2380–2384, 2008.

[6] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A survey of iterative learning control. *IEEE Control Systems*, 26(3):96–114, 2006.

[7] Luigi Cardamone, Daniele Loiacono, Pier Luca Lanzi, and Alessandro Pietro Bardelli. Searching for the optimal racing line using genetic algorithms. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 388–394, 2010.

[8] A Carvalho, Y Gao, A Gray, H Tseng, and F Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2335–2340, 2013.

[9] D Casanova. *On minimum time vehicle manoeuvring: The theoretical optimal lap*. PhD thesis, Cranfield University, 2000.

[10] Changfang Chen, Yingmin Jia, Junping Du, and Fashan Yu. Lane keeping control for autonomous 4WS4WD vehicles subject to wheel slip constraint. In *American Control Conference (ACC)*, pages 6515–6520, 2012.

[11] Jeff Cobb. Driverless Audi RS7 blazes around Hockenheim circuit. http://www.hybridcars.com/driverless-audi-rs7-blazes-around-hockenheim-race-circuit, October 2014.

[12] John J Craig. Adaptive control of manipulators through repeated trials. In *American Control Conference (ACC)*, pages 1566–1573, 1984.

[13] Jennifer N Dang. Statistical analysis of the effectiveness of electronic stability control (ESC) systems. Technical report, NHTSA, 2007.

[14] Alex Davies. GM has aggressive plans for self-driving cars. http://www.wired.com/2015/10/gm-has-aggressive-plans-for-self-driving-cars/, October 2015.

[15] Jamie Paige Deaton. Top 10 everyday car technologies that came from racing. http://auto.howstuffworks.com/under-the-hood/trends-innovations/top-10-car-tech-from-racing.htm, August 2008.

[16] Moritz Diehl, Francois Glineur, Elias Jarlebring, and Wim Michiels. *Recent Advances in Optimization and its Applications in Engineering*. Springer, 2010.

[17] Stephen Erlien, Susumu Fujita, and J Christian Gerdes. Safe driving envelopes for shared control of ground vehicles. In *Advances in Automotive Control*, volume 7, pages 831–836, 2013.

[18] Stephen M Erlien, Joseph Funke, and J Christian Gerdes. Incorporating nonlinear tire dynamics into a convex approach to shared steering control. In *American Control Conference (ACC)*, pages 3468–3473, 2014.

[19] Carlos Filho and Denis Wolf. Dynamic inversion-based control for front wheel drive autonomous ground vehicles near the limits of handling. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2138–2143, 2014.

[20] Chris T Freeman and Ying Tan. Iterative learning control with mixed constraints for point-to-point tracking. *IEEE Transactions on Control Systems Technology*, 21(3):604–616, 2013.

[21] Joe Funke. *Collision avoidance up to the handling limits for autonomous vehicles*. PhD thesis, Stanford University, 2015.

[22] Joseph Funke, Matthew Brown, Stephen M Erlien, and J Christian Gerdes. Prioritizing collision avoidance and vehicle stabilization for autonomous vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1134–1139, 2015.

[23] Marco Gadola, David Vetturi, Danilo Cambiaghi, and Luca Manzo. A tool for lap time simulation. Technical report, SAE Technical Paper, 1996.

[24] Matthias Gerdts, Simon Karrenberg, Bernhard Müller-Beßler, and Gregor Stock. Generating locally optimal trajectories for an automatically driven car. *Optimization and Engineering*, 10(4):439–463, 2009.

[25] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[26] R Griffiths. Minimum time simulation of a racing car. *School of Mechanical Engineering, Cranfield University, MSc thesis*, 1992.

[27] Mike Hanlon. Audi's autonomous Audi TT conquers Pikes Peak - how long before it betters a human driver? http://www.gizmag.com/audis-autonomous-audi-tt-conquers-pikes-peak-how-long-before-it-betters-a-human-driver/17001/, November 2010.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[29] JPM Hendrikx, TJJ Meijlink, and RFC Kriens. Application of optimal control theory to inverse simulation of car handling. *Vehicle System Dynamics*, 26(6):449–461, 1996.

[30] Rami Y Hindiyeh and J Christian Gerdes. A controller framework for autonomous drifting: Design, stability, and experimental validation. *Journal of Dynamic Systems, Measurement, and Control*, 136(5):051015, 2014.

[31] P Hingwe and M Tomizuka. A variable look-ahead controller for lateral guidance of four wheeled vehicles. In *American Control Conference (ACC)*, pages 31–35, 1998.

[32] Lukasz Hladowski, Krzysztof Galkowski, Zhonglun Cai, Eric Rogers, Chris T Freeman, and Paul L Lewin. Experimentally supported 2D systems based iterative learning control law design for error convergence and performance. *Control Engineering Practice*, 18(4):339–348, 2010.

[33] David J Hoelzle, Andrew G Alleyne, and Amy J Wagoner Johnson. Basis task approach to iterative learning control with applications to micro-robotic deposition. *IEEE Trans. on Control Systems Technology*, 19(5):1138–1148, 2011.

[34] Deqing Huang, Jian-Xin Xu, Venkatakrishnan Venkataramanan, and TCT Huynh. High-performance tracking of piezoelectric positioning stage using current-cycle iterative learning control with gain scheduling. *IEEE Trans. on Industrial Electronics*, 61(2):1085–1098, 2014.

[35] Philip Iglauer. Hyundai to develop fully autonomous cars by 2030. http://www.zdnet.com/article/hyundai-to-develop-fully-autonomous-cars-by-2030/, November 2015.

[36] Kichun Jo, Yongwoo Jo, Jae Kyu Suhr, Ho Gi Jung, and Myoungho Sunwoo. Precise localization of an autonomous car based on probabilistic noise models of road surface marker features using multiple cameras. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3377–3392, 2015.

[37] Curtis Franklin Jr. Ford's autonomous car: Under the hood. http://www.informationweek.com/iot/fords-autonomous-car-under-the-hood/a/d-id/1323920, January 2016.

[38] Nitin R. Kapania and J. Christian Gerdes. Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. *Vehicle System Dynamics*, 53(12):1687–1704, 2015.

[39] Nitin R Kapania and J Christian Gerdes. Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control. In *American Control Conference (ACC)*, pages 2753–2758, 2015.

[40] Nitin R Kapania, John Subosits, and J Christian Gerdes. A sequential two-step algorithm for fast generation of vehicle racing trajectories. In *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2015.

[41] Sadao Kawamura, Fumio Miyazaki, and Suguru Arimoto. Iterative learning control for robotic systems. In *Proc. of IECON*, volume 84, pages 393–398, 1984.

[42] Daniel Patrick Kelly. *Lap time simulation with transient vehicle and tyre dynamics*. PhD thesis, Cranfield University, 2008.

[43] Heather Kelly. Toyota to invest $1 billion into artificial intelligence. http://money.cnn.com/2015/11/05/technology/toyota-ai-research/, November 2015.

[44] Dong-II Kim and Sungkwun Kim. An iterative learning control method with application for CNC machine tools. *IEEE Transactions on Industry Applications*, 32(1):66–72, 1996.

[45] Krisada Kritayakirana. *Autonomous vehicle control at the limits of handling*. PhD thesis, Stanford University, 2012.

[46] Krisada Kritayakirana and J. Christian Gerdes. Using the center of percussion to design a steering control for an autonomous racecar. *Vehicle System Dynamics*, 50:33–51, 2012.

[47] Martin Lauer. A case study on learning a steering controller from scratch with reinforcement learning. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 260–265, 2011.

[48] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[49] Nanjun Liu and Andrew G Alleyne. Iterative learning identification for an automated off-highway vehicle. In *American Control Conference (ACC)*, pages 4299–4304, 2011.

[50] Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[51] Ali Mohammadpour, Sandipan Mishra, and Leila Parsa. Iterative learning control for fault-tolerance in multi-phase permanent-magnet machines. In *American Control Conference (ACC)*, pages 5929–5934, 2013.

[52] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[53] Martin Mühlmeier and Norbert Müller. Optimisation of the driving line on a race track. *Auto Technology*, 3(2):68–71, 2003.

[54] Steffen Muller, Michael Uchanski, and Karl Hedrick. Estimation of the maximum tire-road friction coefficient. *Journal of Dynamic Systems, Measurement, and Control*, 125(4):607–617, 2003.

[55] Dipl-Wirtsch-Ing Bernhard Müller-Beßler, Ing Roman Henze, and Ing Ferit Küçükay. Reproducible transverse dynamics vehicle evaluation in the double lane change. *ATZ worldwide*, 110(4):44–49, 2008.

[56] M Nagai, H Mouri, and P Raksincharoensak. Vehicle lane-tracking control with steering torque input. *Vehicle System Dynamics Supplement*, 37:267–278, 2002.

[57] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[58] Hans B Pacejka. *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 3rd edition, 2012.

[59] Giacomo Perantoni and David JN Limebeer. Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics*, 52(5):653–678, 2014.

[60] Oliver Purwin and Raffaello DAndrea. Performing and extending aggressive maneuvers using iterative learning control. *Robotics and Autonomous Systems*, 59(1):1–11, 2011.

[61] Mike Ramsey. Renault-Nissan gives details on autonomous vehicle plan. http://www.wsj.com/articles/renault-nissan-announces-autonomous-vehicle-plan-1452194640, January 2016.

[62] Eric J Rossetter. *A potential field framework for active vehicle lanekeeping assistance*. PhD thesis, Stanford University, 2003.

[63] Eric J Rossetter and J Christian Gerdes. A study of lateral vehicle control under a virtual force framework. In *Proceedings of the International Symposium on Advanced Vehicle Control*, 2002.

[64] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[65] RS Sharp. A method for predicting minimum-time capability of a motorcycle on a racing circuit. *Journal of Dynamic Systems, Measurement, and Control*, 136(4):041007, 2014.

[66] RS Sharp, Daniele Casanova, and P Symonds. A mathematical model for driver steering control, with design, tuning and performance results. *Vehicle System Dynamics*, 33(5):289–326, 2000.

[67] Steven E Shladover, Charles A Desoer, J Karl Hedrick, Masayoshi Tomizuka, Jean Walrand, Wei Zhang, Donn H McMahon, Huei Peng, Shahab Sheikholeslam, and Nick McKeown. Automated vehicle control developments in the PATH program. *IEEE Transactions on Vehicular Technology*, 40(1):114–130, 1991.

[68] Bryant Walker Smith. Human error as a cause of vehicle crashes. http://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes, December 2013.

[69] Russell Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[70] John K Subosits and John Christian Gerdes. Autonomous vehicle control for emergency maneuvers: The effect of topography. In *American Control Conference (ACC)*, pages 1405–1410, 2015.

[71] Heqing Sun, Zhongsheng Hou, and Dayou Li. Coordinated iterative learning control schemes for train trajectory tracking with overspeed protection. *IEEE Trans. on Automation Science and Engineering*, 10(2):323–333, 2013.

[72] Kirstin L Talvala and J Christian Gerdes. Lanekeeping at the limits of handling: Stability via Lyapunov functions and a comparison with stability control. In *Dynamic Systems and Control Conference*, pages 361–368, 2008.

[73] Paul A Theodosis. *Path planning for an automated vehicle using professional racecar driving techniques*. PhD thesis, Stanford University, 2014.

[74] Paul A Theodosis and John Christian Gerdes. Generating a racing line for an autonomous racecar using professional driving techniques. In *Dynamic Systems and Control Conference*, pages 853–860, 2011.

[75] M Thommyppillai, S Evangelou, and RS Sharp. Car driving at the limit by adaptive linear optimal preview control. *Vehicle System Dynamics*, 47(12):1535–1550, 2009.

[76] Julian P Timings and David J Cole. Minimum maneuver time calculation using convex optimization. *Journal of Dynamic Systems, Measurement, and Control*, 135(3):031015, 2013.

[77] S. Ulbrich and M. Maurer. Probabilistic online POMDP decision making for lane changes in fully automated driving. In *IEEE 16th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2063–2067, Oct 2013.

[78] Audi USA. Piloted to the peak: Audi TTS Pikes Peak. http://www.audi.com/, January 2016.

[79] Efstathios Velenis and Panagiotis Tsiotras. Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation. *Journal of Optimization Theory and Applications*, 138(2):275–296, 2008.

[80] Efstathios Velenis, Panagiotis Tsiotras, and Jianbo Lu. Optimality properties and driver input parameterization for trail-braking cornering. *European Journal of Control*, 14(4):308–320, 2008.

[81] Dr.-Ing. Peter Waldmann and Dipl.-Ing. Daniel Niehues. Der BMW Track-Trainer - automatisiertes fahren im Grenzbereich auf der Nürburgring Nordschleife. Technical report, Tagung aktive Sicherheit, Lehrstuhl für Fahrzeugtechnik, Technische Universität München, 2010.

[82] Junqing Wei, J.M. Dolan, J.M. Snider, and B. Litkouhi. A point-based MDP for robust single-lane autonomous driving behavior under uncertainties. In *IEEE International Conference onRobotics and Automation (ICRA)*, pages 2586–2592, May 2011.

[83] Derong Yang, Bengt Jacobson, Mats Jonasson, and Tim J Gordon. Closed–loop controller for post–impact vehicle dynamics using individual wheel braking and front axle steering. *International Journal of Vehicle Autonomous Systems*, 12(2):158–179, 2014.