Planning and Control of Drift Maneuvers with the Berkeley Autonomous Race Car

by

Jon Matthew Gonzales

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Francesco Borrelli, Chair
Professor Mark Mueller
Professor Murat Arcak

Summer 2018

Planning and Control of Drift Maneuvers with the Berkeley Autonomous Race Car

Abstract

Planning and Control of Drift Maneuvers with the Berkeley Autonomous Race Car

by

Jon Matthew Gonzales

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

In competitive sporting events, drivers operate vehicles at the limits of handling, with near full rear tire saturation. Expert drivers intentionally drift their vehicles around corners to turn the vehicle quickly. These drivers operate their vehicles in a way that is contrary to the way safety systems in automotive electronic control units are designed. By investigating and understanding the physics and operating principles of these maneuvers, it may be possible to enhance safety features in automotive control systems for collision avoidance, as well as enable sports cars to autonomously perform drift within the context of racing. The main focus of this dissertation is on planning and control of drift maneuvers, in particular, steady state drift, drift parking, and drift cornering.

Secondly, with the growth of research and engineering in the domain of autonomous vehicles, the dissertation also focuses on the design and development of a robotic platform called the Berkeley Autonomous Race Car (BARC). The platform is based on a 1/10-scale remote control vehicle equipped with computing hardware and a suite of sensors that make it suitable for research and instruction. The project aims to provide a low-cost, open-source testbed option for researchers and instructors interested in autonomous vehicles. The methods and algorithms provided in the first part of the dissertation are experimentally validated on the BARC platform.

To my family for all their love and support, and all others who have helped me throughout this journey

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor, Francesco Borrelli, for his solid support and guidance through my graduate studies at UC Berkeley. I am indebted to him for all his advice and expertise, from academics to research projects to professional development. I am also very grateful to the late Professor Karl Hedrick for his support and encouragement in my academic endeavors. I would also like to thank Professor Mark Mueller and Professor Murat Arcak for serving on my dissertation committee. I would also like to acknowledge Professor O'Reilly's taking me in during my first year at UC Berkeley.

The Model Predictive Control lab represents my academic family. I cherish the personal and professional ties that I formed with the many members in the lab over the past several years. I'm grateful to Ashwin Carvalho, Sarah Koehler, Frank Chuang and Theresa Lin for welcoming me into the lab after I joined. I appreciate Tony Kelman for offering a ton of advice on numerical optimization routines and on the Julia programming language. I'd like to thank Ugo Rosolia, Charlott Vallon, Yeojun Kim, Jacopo Guanetti, Xiaojing (George) Zhang, Monimoy Bujarbaruah, Andreas Hansen, Yi-Wen (Grace) Liao and the many others of the MPC / VDL lab that have helped me along the way.

I would like to thank the leadership and engineers at Hyundai Motor Company for the opportunity to consult on their project in autonomous driving and for providing funding to support my graduate studies. Their engineers taught me a lot of the practical challenges of instrumenting commercial vehicles with sensors and computing hardware, as well as implementing advanced control and estimation techniques on OEM hardware.

I would like to acknowledge the support of UC Berkeley for the Chancellor's fellowship and GEM National Consortium for the GEM fellowship; any findings and conclusions herein do not necessarily reflect their views.

I want to thank my friends from the International Graduate Student Ministry for providing an environment to grow spiritually and preserve through the difficult times during graduate school.

Finally, I would like to thank my parents and sibling for all their love, support and encouragement. My parents are exemplary role models and they continue to inspire me.

Chapter 1

Introduction

1.1 Motivation and Background

Planning and Control of drift maneuvers

Drift represents one of the most advanced, dangerous, and captivating maneuvers a skilled driver can perform on a vehicle. It occurs when a driver intentionally oversteers a vehicle and maintains control of it amid loss of traction in the wheels. Drifting is easily recognized by three characteristic features: large side slip angles, rear tire saturation, and counter-steer. Visually, a large side slip angle corresponds to lateral motion, or how much the vehicle is moving 'side ways'. Rear tire saturation means that the tires are generating the maximum amount of force available from friction to propel the car forward. At this point, the vehicle is said to be operating at the limits of handling. With saturated tires, pushing further down on the throttle only causes the tires to slip more and smoke to come up from the tires' rubbing against the road. Lastly, counter-steer refers to the effect of steering the wheels in a direction opposite to the rotation of the vehicle.

Drift occurs in competitive sporting contexts, like rally racing and performance demonstrations. In rally racing, expert drivers skillfully execute a drift maneuver to turn sharp 90 deg to 180 deg corners quickly. For example, Ken Block, one of the world's top rally racers, masterfully drifts through the racetrack at l'Autodrome de Linas in France for a promotional event in [35]. In performance demos, drivers showcase their handling abilities by sliding a vehicle into a tight parking spot or performing a continuous drift. Han Yue set the world record for the tightest parallel parking by sliding his vehicle into a space 15 cm longer than the length of his vehicle [32]. These drivers operate their vehicles in a way that is contrary to the way the safety system in automotive electronic control units are designed. By investigating and understanding the physics and operating principles of these maneuvers, it may be possible to enhance safety features in automotive control systems for collision avoidance, as well as enable sports cars to autonomously perform drift within the context of racing.

While drift represents a maneuver that is beyond the handling capabilities of the ordinary driver, in some contexts, like drift parking and drift cornering, the actions an expert driver

Figure 1.1: Snapshot of drift maneuver at 2016 Midwest Drift Union event in Saint, Louis. Image from [22] under Creative Common license

executes to drift can often be described through simple, yet precisely timed control actions. Secondly, over short time durations, vehicle systems that operate in open-loop (i.e. apply a fixed sequence of input commands) from similar initial conditions behave deterministically, and produce repeatable, predictable responses, even if the dynamics are complex and difficult to model. A big challenge for model-based control design of drift maneuvers lies in using a sufficiently good mathematical model to capture the drift dynamics. The previous observations suggest a way to incorporate open-loop operation as a part of a larger control strategy for drift. The dissertation focuses on the ideas of sample-based path planning and mixed open-loop, closed-loop control as a complete framework to perform autonomous drift maneuvers. These ideas are verified from experimental results on drift parking and drift cornering. A separate, related work on the control of steady state drift is included as well.

Robotic platform for autonomous driving

With the rise of autonomous vehicles, it is becoming more and more important to instruct engineering students not only on how to develop control algorithms, but also to validate them on a physical testbed through experimentation. While full-scale vehicles are the ideal testbed to validate control algorithms for autonomous driving, they present challenges in terms of safety and liability, as they can cause property damage and potential loss of human life if the algorithms malfunction. Additionally, even if the proper safety mechanism in software and hardware are designed, instrumented full-scale vehicles are not accessible to many due to cost. While costs are declining for high-end sensors (e.g. LiDAR, RGB cameras,

RADAR, differential GPS, etc.), equipping a vehicle with the appropriate sensors, computing hardware, cabling and acquiring legal authorization to test on roads represents a high barrier to entry. As an alternative, small scale autonomous vehicles afford a much more practical means to develop and test control algorithms.

Unfortunately, to the best of the author's knowledge, on the market, there are no commercial low-cost, small-scale platforms for research and development for self-driving cars. Many of the existing products like Turtlebot [42], Magni [26], ROSbot [33], are all strong platforms for mobile robotic development, but do not have a chassis construction or actuation system that resembles that of a full-scale vehicle. A few research groups have developed open-source platforms [27] [20] [8], but cost or difficulty of assembly present a hurdle for others interested in replicating the platform.

To help bridge the gap for students and researchers between theory and practice, we have developed an open-source, low-cost robotic platform for autonomous driving called the Berkeley Autonomous Race Car (BARC). The platform is based on a 1/10-scale remote control vehicle equipped with computing hardware and a suite of sensors that make it suitable for research and instruction. The project aims to provide a low-cost, open-source testbed option for researchers and instructors interested in autonomous vehicles. The second focus of this dissertation is on the development of the BARC platform for research and instruction.

1.2 Outline and Contributions

This dissertation is organized thematically into three main chapters.

Mathematical models lie at the center of any model-based control algorithm. The first part of Chapter Two provides an overview of vehicle and tire models for control. The focus is on the underlying assumptions of each model and the trade-off between accuracy and computational complexity, starting with kinematic models and progressing to four-wheel vehicle models with tire models that account for both the slip angle and slip ratio. The next segment of Chapter Two then discusses model identification using optimization programs. These programs find the optimal vehicle parameters and tire model coefficients to fit the mathematical models to the data. Lastly, Chapter Two concludes with an analysis of drift at state steady. This analysis gives insight into the interplay between side slip angle, yaw rate, and steering during drift.

Chapter Three gives an introduction to a 1/10-scale robotic platform the author developed called the Berkeley Autonomous Race Car (BARC). The platform aims to bring a low-cost, open-source testbed to students, instructors, and researchers for testing control algorithms for self-driving cars. The focus of the chapter is on the mechanical, electrical, and software design decisions and development of the platform. The mechanical portions touches on the chassis selection and the fabrication methods (e.g. water jet, fused deposition modeling 3D printing) . The electrical segment discusses the sensor suite (e.g. encoders, inertial measurement unit, camera, sonar, etc.) and the computing hardware. Other considerations like power distribution and cabling are discussed as they impact the ease of assembly

and protection of onbaord electronic devices. The software section discusses the use of the Robotic Operating System (ROS), open-source tools, and cloud services with BARC. Chapter Three concludes by discussing how the platform has been integrated into an engineering college curriculum to enrich the education of students by complementing theory with experimentation.

Chapter Four discusses planning and control for three specific drifting maneuvers: steady state drift, drift parking and drift cornering. This chapter ties in themes from Chapters Two and Three by discussing planning and control techniques, and then applying them to the BARC platform. The results from the equilibrium analysis in Chapter Two shape the design of the control policy for steady state drift. The experimental implementation aims to use only on-board sensors, including camera, encoders, and the inertia measurement unit, and onboard computation for the BARC platform to achieve drift. For the topics of drift parking and drift cornering, we discuss a path planning strategy and then a mixed open-loop, closed-loop control scheme for conducting transient, complex drift maneuvers. The motivation for mixed control stems from the observation that vehicle systems behave deterministically over a short duration when operating in open-loop (i.e. fixed sequence of input commands) from approximately the same initial condition, even if the dynamics are complex and difficult to model.

The contributions of this dissertation are the following:

- Development of a Berkeley Autonomous Race Car (BARC) platform - 1/10 scale robot for research and instruction for autonomous driving
- Co-development of curriculum for integrating BARC platform into college curriculum in vehicle dynamics and control
- Sample-based path planning strategy for drift maneuvers
- LQR-based control design for steady state drifting
- Mixed open-loop, closed-loop control scheme for reference tracking with heuristic implementation strategies for drift parking and drift cornering

The results presented in this dissertation have appeared in the following list of publications coauthored by the author of the dissertation:

– J. Gonzales, F. Zhang, K. Li, F. Borrelli. "Autonomous Drifting with Onboard Sensors". In: *13th International Symposium on Advanced Vehicle Control, Munich*. 2016

– F. Zhang, J. Gonzales, K. Li, F. Borrelli. "Autonomous Drift Cornering with Mixed Open-loop and Closed-loop Control" *20th World Congress of the International Federation of Automatic Control*. 2017

– E. Jelavic, J. Gonzales, F. Borrelli. "Autonomous Drift Parking using a Switched

Control Strategy with Onboard Sensors". In: *20th World Congress of the International Federation of Automatic Control*. 2017

The author has discussed the BARC platform in the following workshops and talks:

– J. Gonzales, C. Vallon, T. Zheng, F. Borrelli. "Autonomous Vehicles: An Open Platform for Learning and Teaching" In: *American Control Conference*. Milwaukee, USA, June 26, 2018

– J. Gonzales. "Autonomous Driving for RC Cars with ROS and Julia" In: *JuliaCon*. Boston, USA, June 23, 2016

Chapter 2

Vehicle Models

2.1 Introduction

At the heart of any model-based control algorithm lies the mathematical model of the system. The purpose of the model is to capture the underlying physics and mathematically describe how the system inputs effect the system outputs. For our purposes, these models take the shape of differential equations or difference equations for the continuous and discrete system, respectively. In the space of vehicle dynamics, these models serve primarily to capture system states such as position, orientation and velocity. Vehicle models vary in terms of complexity and fidelity, but at a high level, these models typically fall into one of the following three categories:

- *Point-mass models*
- *Kinematic models*
- *Dynamic models*

This chapter provides an overview of the vehicle models widely used for Advanced Driving Assistance Systems (ADAS) and chassis control systems. We will begin by describing the basic assumptions and principles underlying each of the models, and then transform the models into a state space representation.

Before moving forward, we first define the coordinate system used in each of the following models. We define a fixed global coordinate system with mutually orthogonal axes $\{\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3\}$. These axes form a right handed coordinate system, where $\mathbf{E}_1, \mathbf{E}_2$ lie on the road surface and $\mathbf{E}_3$ points upward. We define the vehicle-fixed coordinate system $\{\mathbf{e}_1^v, \mathbf{e}_2^v, \mathbf{e}_3^v\}$, where the $\mathbf{e}_1^v$ axis points from the vehicle's center of gravity (CoG) to the front of the vehicle, the $\mathbf{e}_2^v$ axis points from the CoG to the left of the vehicle, and $\mathbf{e}_3^v$ points upward. Lastly, we define the tire-fixed coordinate systems $\{\mathbf{e}_1^{t_{*,\Delta}}, \mathbf{e}_2^{t_{*,\Delta}}, \mathbf{e}_3^{t_{*,\Delta}}\}$, where $* = \{1 = \text{front}, 2 = \text{rear}\}$ and $\Delta = \{r = \text{right}, l = \text{left}\}$.

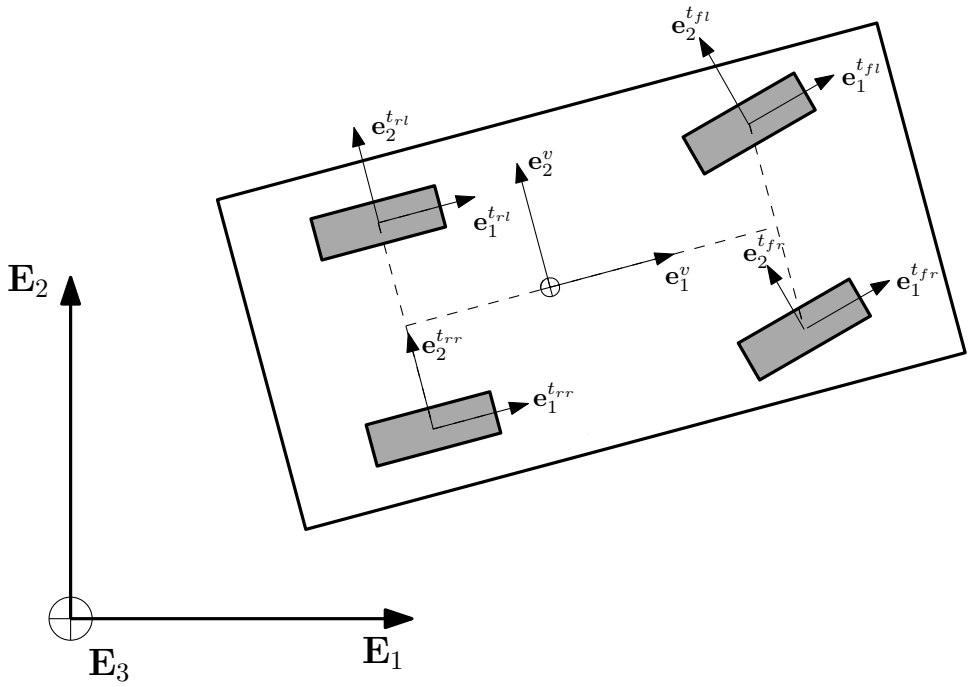


Figure 2.1: Coordinate system for vehicle motion.

The $\mathbf{e}_1^{t_*,\Delta}$ axis pointing from the tire contact patch along the direction of the tire, the $\mathbf{e}_2^{t_*,\Delta}$ axis pointing to the left of the tire and $\mathbf{e}_3^{t_*,\Delta}$ pointing upward.

As we review various vehicle models, we describe kinematic quantities like velocity and position in terms of either the global coordinate frame or the vehicle-fixed frame. Transforming the vector quantities from one coordinate frame to another is done by multiplying it by a rotation matrix.

2.2 Point-mass Model

We begin our survey of mathematical models with a simple point-mass description of the vehicle. Point-mass models are the most basic system models that only consider kinematic definitions of motion. The type of point mass model described here neglects angular dynamics. The vehicle state is $\mathbf{z} = \begin{bmatrix} x & v \end{bmatrix}^\top$, where x is the position coordinate and v is the speed, and the input is $u = a$, where a is the acceleration. The point-mass model only captures unidirectional motion, meaning the vehicle motion is only along a straight path (i.e. no rotation).

Starting from the definition of acceleration $a = \frac{dv}{dt}$ and velocity $v = \frac{dx}{dt}$, we arrange the equations into the following state space representation

$$\frac{d}{dt}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a \tag{2.1}$$

The explicit equation for position as a function of time simply integrates the acceleration twice. If constant acceleration is assumed, we arrive at the following expression for position

$$x = x_0 + vt + \frac{1}{2}at^2 \tag{2.2}$$

For a few applications, like longitudinal control in ADAS systems, the basic point mass model works sufficiently well, but does not work well for lateral controllers due to lateral errors [2].

For the remainder of the chapter, we focus on other vehicle models, and we simply present the point-mass model for completeness.

2.3 Kinematic Model

Next, we consider the kinematic vehicle model. Similar to the previous model, this model only considers kinematic relations that govern the motion of the vehicle. Forces acting on the vehicle are not considered. This model assumes the velocity vectors at the front and rear wheels are in the direction of the front and rear tires, respectively. An equivalent statement is that the tire side slip angles are zero. Such assumptions are reasonable for vehicles at low speeds on high friction surfaces [31].

The kinematic motion only accounts for planar motion. The vehicle state and input are:

$$z = \begin{bmatrix} X & Y & \psi & a \end{bmatrix}^\top$$
$$u = \begin{bmatrix} \delta & a \end{bmatrix}^\top$$

where (x, y) are the position coordinates of the vehicle in global coordinate frame, ψ is the yaw angle (i.e. angle between vehicle heading and global X axis), δ is the steering angle (define with respect to the body fixed coordinate system), and a is the acceleration.

The equations of motion for the kinematic model can be derived by taking into account the instantaneous center of rotation and applying standard trigonometric identities [31]. The kinematic model is described by following set of differential equations in the global reference frame:

$$\dot{X} = v\cos(\psi + \beta) \tag{2.3a}$$
$$\dot{Y} = v\sin(\psi + \beta) \tag{2.3b}$$
$$\dot{\psi} = \frac{v}{L_r} \tag{2.3c}$$
$$\dot{v} = a \tag{2.3d}$$
$$\beta = \tan^{-1}\left(\frac{L_r}{L_f + L_r}\tan(\delta_f)\right) \tag{2.3e}$$

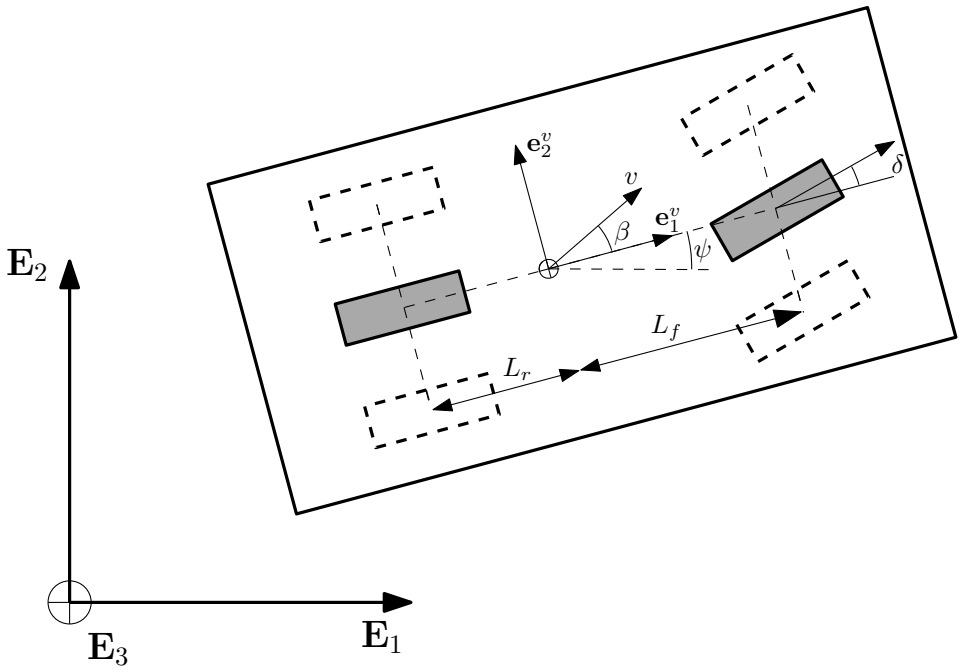


Figure 2.2: Kinematic Bicycle Model.

where β is called the slip angle, L_f is the distance from the CoG to the front wheel axle and L_r is the distance from the CoG to the rear wheel axle. We remark on two points. First, we assume the rear tires are aligned with the $\mathbf{e}_1^v$ axis. This simplification is not necessary, but for this work, we only model vehicles with front wheel steering. Second, the kinematic bicycle model, as the name suggests, simplifies the 4-wheeled vehicle into a bicycle by lumping the front two tires into one and lumping the rear two tires into one. This assumes the front and rear wheels point in the same direction and rotate at the same speed. Strictly speaking, this assumption is not correct whenever the vehicle rotates, since the radius path of each wheel is different. Nonetheless, the quantities are roughly equal, and thus, the lumped-tire bicycle kinematic model describes the motion accurately enough for autonomous driving control [24].

2.4 Dynamic Models

Models for vehicle dynamics incorporate two major components. First is the rigid body model of the vehicle itself, and the forces acting on the contact patches. Second, and just as important, is the tire model. For extreme maneuvers like drift, it is necessary to accurately capture the effects of the tire forces in order to model the motion of the vehicle. We first begin by examining a couple of common tire models in the literature, and then transition into an overview of various vehicle model. Many of the vehicle models require an accurate tire model, so it is better to treat the tire model first.

Tire Models

Tire models are a major subject of vehicle dynamics. Many works in vehicle dynamics have been devoted into characterizing tire forces [28] [30] [11], but tire models still represent a challenge because of the myriad of factors that affect the forces generated from them. Tires vary in material composition and tread design, which affects elastic and thermal properties. Additionally, tire wear and pressure inflation also influence how the tire behaves [15].

Many mathematical models have been developed for tire forces. They fall into either physics-based or empirical-based models. Physics based tire models treat the tire as a spring that generates a force due to material deformation. Empirical-based models, on the other hand, simply fit a function with parameters to data. Both approaches have their merits. Physics-based models provide insight for how physical parameters directly influence the tire forces. Empirical-based models tend to predict the tire forces more accurately, but at the expense of physical intuition. In related disciplines, these types of models are classified as white, black, or gray-box models, depending on the level of clarity in the theoretical structure. These models also vary in fidelity, from basic linear equations to complex, highly nonlinear expressions with several parameters. Selecting an appropriate tire model depends on the application.

In the following section, we will give an overview of common tire models for vehicle dynamics and control.

Slip angle and slip ratio

In practice, the forces generated at each tire depend on several variables, including slip angle α, slip ratio ω, the normal force F_z, the coefficient of friction μ, and the temperature of the tire. The complexity of these high fidelity models is not conducive to system analysis and control, so we will consider simplified tire models, where the lateral force F_y depends on the side slip angle α and the longitudinal force F_x depends on the slip ratio σ.

We begin with modeling lateral force. All of the tire models estimating lateral force depend on the slip angle α, which is defined as the angle between the lateral component and longitudinal component of the velocity vector of the tire, as illustrated in Figure 2.3. The axes $\mathbf{e}_1^t, \mathbf{e}_2^t$ indicate the velocity components under consideration are in the tire frame, with magnitudes v_x^t and v_y^t. In physical terms, this captures the angle between the wheel's rolling direction and the actual direction of movement. This is analogous to the vehicle slip angle β, which captures the difference between the vehicle's direction of motion and the direction it is facing. Any non-zero slip angle α will generate a lateral force at the tire that is perpendicular to the direction of travel. All of the tire models discussed in this section depend the magnitude of the slip angle.

For a four-wheel vehicle with front steering angle δ, we can define the tire slip angle of each wheel by considering the kinematic relation between the linear velocity and angular velocity at the center of mass, and the velocity components at each of the tires. From Figure 2.4, v_x, v_y indicate the velocity at the center of mass with angular velocity ω, $v_{x'}^{\Delta,*}, v_{y'}^{*,\Delta}$, with

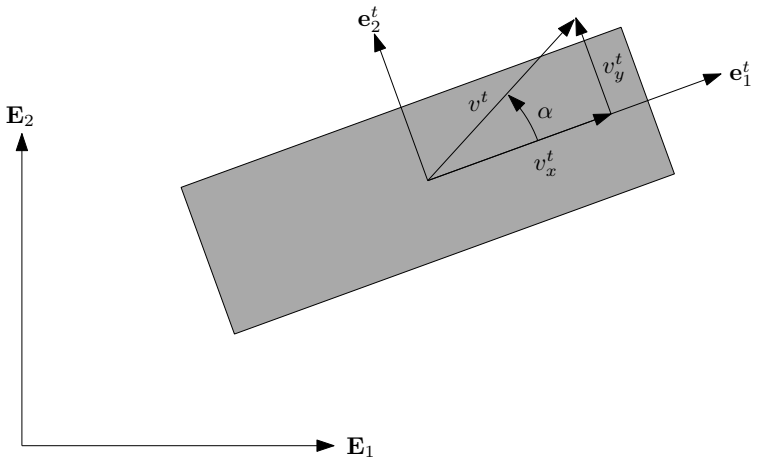


Figure 2.3: The tire side slip measures the angle between the lateral and longitudinal component of the tire velocity vector.

$* = \{1 = \text{front}, 2 = \text{rear}\}$, $\Delta = \{r = \text{right}, l = \text{left}\}$ indicate the velocity components of each tire in the tire frame, and $v_x^{\Delta,*}, v_y^{*,\Delta}$ indicate the velocity components of the tire projected onto the vehicle frame.

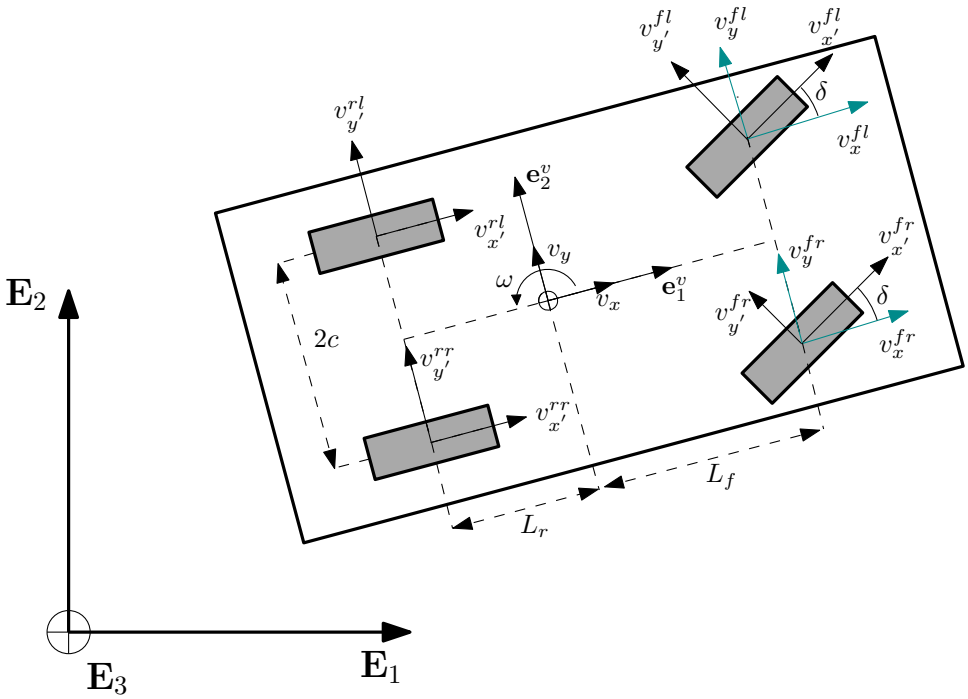


Figure 2.4: Longitudinal and lateral components of tire velocity in a four-wheel vehicle model.

The velocity of each tire in each frame is computed based on the kinematic relation:

$$\mathbf{v}^t = \mathbf{v}^v + \boldsymbol{\omega} \times \mathbf{r}_{v \to t} \tag{2.4}$$

where $\mathbf{v}^v$ is the velocity vector at the center of mass in the vehicle frame, $\mathbf{v}^t$ is the velocity vector of the tire in the vehicle frame, $\boldsymbol{w}$ is the angular velocity vector of the vehicle and $\mathbf{r}_{v \to t}$ is the displacement vector from the center of mass to the center of the tire contact patch. Since we focus on planar motion, the angular velocity vector only has one component in the z-direction of value ω_z. Later in this section, by convention, we will refer to the yaw rate with the symbol $r = \omega_z$. The longitudinal and lateral components of the velocity at each of the tires are given in the following set of equations:

$$v_x^{fr} = v_x + \omega_z c \tag{2.5a}$$
$$v_x^{fl} = v_x - \omega_z c \tag{2.5b}$$
$$v_x^{rr} = v_x + \omega_z c \tag{2.5c}$$
$$v_x^{rl} = v_x - \omega_z c \tag{2.5d}$$
$$v_y^{fr} = v_y + \omega_z L_f \tag{2.5e}$$
$$v_y^{fl} = v_y + \omega_z L_f \tag{2.5f}$$
$$v_y^{rr} = v_y - \omega_z L_r \tag{2.5g}$$
$$v_y^{rl} = v_y - \omega_z L_r \tag{2.5h}$$

The tire models for lateral force depend on velocity at each tire in the tire frame, so the velocities are projected into the tire frame from the velocity frame through a rotation of δ (i.e. the front steering angle) about the $\mathbf{e}_3^v$ axis:

$$v_{x'}^{f*} = v_x^{f*} \cos\delta + v_y^{f*} \sin\delta \tag{2.6a}$$
$$v_{y'}^{f*} = -v_x^{f*} \sin\delta + v_y^{f*} \cos\delta \tag{2.6b}$$
$$v_{x'}^{r*} = v_x^{r*} \tag{2.6c}$$
$$v_{y'}^{r*} = v_y^{r*} \tag{2.6d}$$

For each tire, the side slip angle is the angle between the lateral and longitudinal velocity components in the tire frame, as defined by the following equations:

$$\alpha^{fl} = \tan^{-1} \frac{v_{y'}^{fl}}{v_{x'}^{fl}} \approx \frac{v_{y'}^{fl}}{v_{x'}^{fl}} \tag{2.7a}$$

$$\alpha^{fr} = \tan^{-1} \frac{v_{y'}^{fr}}{v_{x'}^{fr}} \approx \frac{v_{y'}^{fr}}{v_{x'}^{fr}} \tag{2.7b}$$

$$\alpha^{rl} = \tan^{-1} \frac{v_{y'}^{rl}}{v_{x'}^{rl}} \approx \frac{v_{y'}^{rl}}{v_{x'}^{rl}} \tag{2.7c}$$

$$\alpha^{rr} = \tan^{-1} \frac{v_{y'}^{rr}}{v_{x'}^{rr}} \approx \frac{v_{y'}^{rr}}{v_{x'}^{rr}} \tag{2.7d}$$

Note that the side slip angle expression in equation (2.7) is often simplified to just the ratio of the lateral to longitudinal velocity using the small angle approximation. For reference, we also remark that the tire slip angles can be computed using the following set of equations, where the expressions avoid the explicit rotation of the velocities into the tire frame:

$$\alpha^{fl} = \tan^{-1} \left(\frac{v_y^{fl}}{v_x^{fl}} \right) - \delta \tag{2.8a}$$

$$\alpha^{fr} = \tan^{-1} \left(\frac{v_y^{fr}}{v_x^{fr}} \right) - \delta \tag{2.8b}$$

$$\alpha^{rl} = \tan^{-1} \left(\frac{v_y^{rl}}{v_x^{rl}} \right) \tag{2.8c}$$

$$\alpha^{rr} = \tan^{-1} \left(\frac{v_y^{rr}}{v_x^{rr}} \right) \tag{2.8d}$$

In additional to lateral deformation, the tire also deforms longitudinally, which propels the vehicle forward. We quantify the longitudinal deformation using two related quantities, the 'theoretical' slip ratio, σ, and the 'practical' slip ratio, κ. Both quantities provide a measure of the slip (i.e. in contrast to pure rolling), but these quantities differ in terms of the normalizing factor in the denominator. The theoretical slip ratio normalizes with respect to the rolling speed of the tire, while the practical ratio normalizes with respect to the longitudinal velocity at the center of the tire. The theoretical and practical slip ratios are defined as follows [28]:

$$\sigma = -\frac{v_x - \omega R}{\omega R} = \frac{k}{1+k} \tag{2.9}$$

$$\kappa = -\frac{v_x - \omega R}{v_x} \tag{2.10}$$

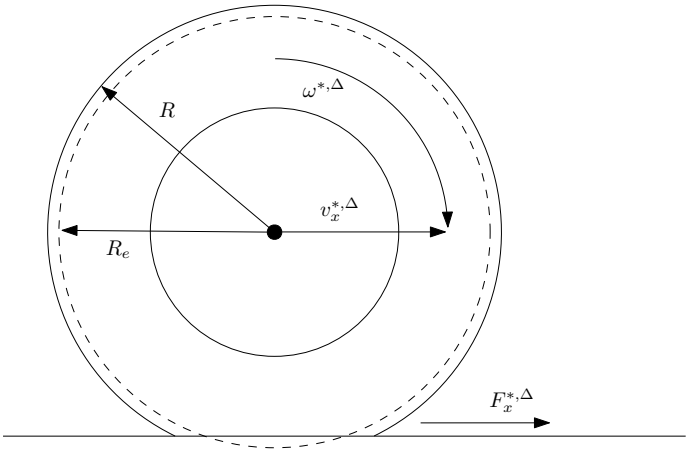


Figure 2.5: The slip-ratio measures the amount of deformation along the rolling direction of the tire.

Braking occurs when $\omega, \kappa > 0$. The use case for each slip ratio definition varies. For example, during hard braking, $\omega R \approx 0$ since the wheels lock, but $v_x > 0$, so using the definition of the theoretical slip ratio would yield numerical values that grow very large, thus the practical slip ratio would provide a better measure of slip.

Figure 2.5 provides an illustration of the quantities involved in the calculation of the slip ratio, where R denotes the radius of the tire and R_e denotes the effective radius. As the wheel rolls, it deforms in the vertical direction, so the effective radius R_e of the rotational motion of the wheel is smaller than the actual physical radius of the wheel R.

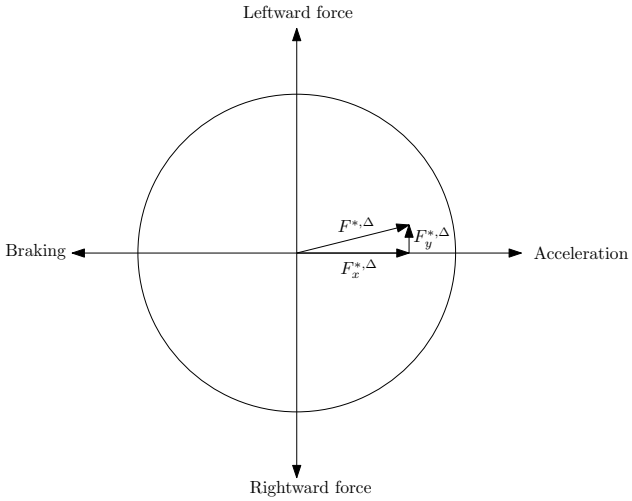
The slip ratio is important for estimating the longitudinal force generated from the tire. The vehicle models used in the control design section of this thesis do not consider the effect of the slip ratio on the vehicle dynamics, but we include it in this section for completeness.

Lastly, while both the tire side slip and slip ratio directly relate to magnitude of the lateral and longitudinal force, the largest possible lateral and longitudinal forces are constrained by the amount of friction available between the tire patch and the road surface. The resultant force of each tire must be less than the total friction available, given below as:

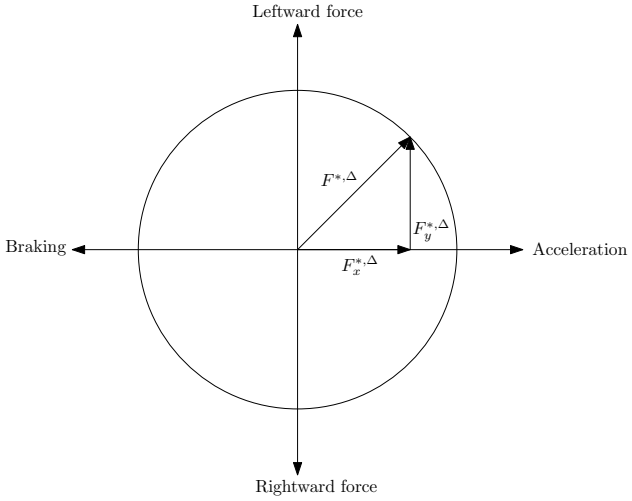
$$F_x^{*,\Delta} + F_y^{*,\Delta} \leq \mu F_z^{*,\Delta} \tag{2.11}$$

where μ is the coefficient of friction and F_z is the normal force of the tire.

Friction heavily depends on the interaction between the road surface and tire. The friction between a dry surface and rubber tires provides much more friction than between a wet surface and rubber tires. We visualize the tire force constraint through a diagram called the friction circle, in which the magnitude of the resultant force from the longitudinal and lateral components of each tire are constrained to lie within the circle. The radius of the circle is equal to the maximum force available from friction.



(a) During normal cornering, the vector sum of longitudinal and lateral components of the tire force lies within the limits of the friction circle.



(b) During tire saturation, the vector sum of longitudinal and lateral components of the tire force reach the limits of the friction circle.

Figure 2.6: During typical cornering, the total tire force, $F^{*,\Delta}$, uses only a portion of the available friction from the friction circle, while during extreme maneuvers, like drifting, the tire force uses all the available friction.

We show the friction circle for a tire in Figure 2.6. The x-axis corresponds to the longitudinal force and the y-axis corresponds to the lateral force. The direction of the lateral force determines the rotational direction of the vehicle. If we consider the forces at the front wheels, a positive lateral force corresponds to a left turn and a negative one corresponds to

a right turn. In Figure 2.6a, the resultant tire force lies within the friction force, whereas in Figure 2.6a, the resultant force lies at the limits. Tires operating at the limits of the friction circle are called saturated. We remark that tire saturation is important in drift dynamics, which we will discuss in detail later.

In the following subsections, we discuss a few physics-based and empirical-based tire models that estimate the lateral force. The models all consist of some functional form with parameters that either correspond to physical parameters or a non-physical property of the function curve. Longitudinal models often take a similar functional form, which the reader can refer to [28] for more details, but we will primarily focus on lateral models, since the lateral force, and in particular, the side slip angle β, play an important role in the dynamics of drifting maneuvers.

Fiala Tire Model

We begin with a discussion of physics-based models, collectively called "brushed" tire models. The most important element of this models are the "brushes" at the bottom of the tire that come into contact with the road. The brushes act as springs as the tire deforms during acceleration. The magnitude of the force along the $\mathbf{e}_1^t$ and $\mathbf{e}_2^t$ axis of the tires depends on two factors: the amount of deformation and the maximum force available from friction.

The Fiala Tire Model describes analytically the relationship between the tire side slip angles and the lateral force, based on the following physical parameters:

$$\begin{aligned} \alpha &= \text{ Slip angle} \\ C_\alpha &= \text{ Tire cornering stiffness} \\ \mu_s &= \text{ Sliding friction coefficient} \\ \mu_p &= \text{ Peak friction coefficient} \\ F_z &= \text{ Normal load applied to the tire} \end{aligned}$$

The tire model takes in the tire slip angle α as input; all other terms are fixed parameters. The Fiala model is given by the following non-linear piecewise function:

$$F_y(\alpha) = \begin{cases} -C_\alpha \tan\alpha - \frac{C_\alpha^3(1-\frac{2\mu_s}{3\mu_p})}{9\mu_p^2 F_z^2}\tan^3\alpha + \frac{C_\alpha^2(2-\frac{\mu_s}{\mu_p})}{3\mu_p F_z}|\tan\alpha|\tan\alpha & : |\alpha| < \alpha_{s1} \\ -\mu_s F_z \text{ sgn } \alpha & : |\alpha| \geq \alpha_{s1} \end{cases} \tag{2.12}$$

The Fiala model strives for accuracy and physical intuition in describing tire forces. All the parameters in the model correspond to physical properties of the tire or the surface, which helps build intuition of how changing the tire or the road surface would affect the forces generated. The model, however, is not smooth due to the piece-wise construction of the model, which complicates its use in optimization solvers.

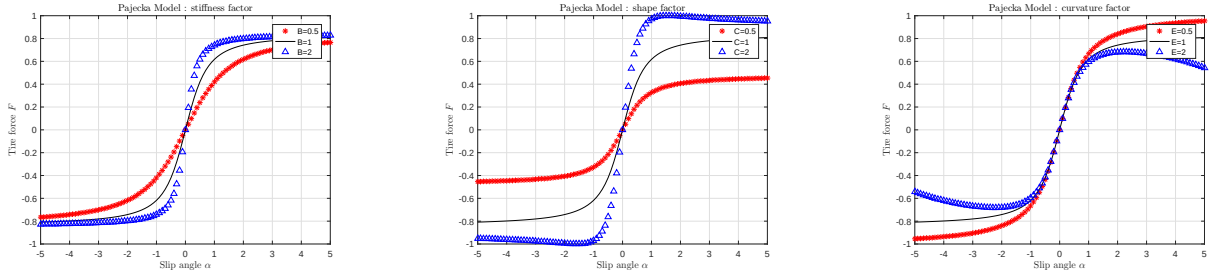
Pacejka Tire Model

The Pacejka model, as referred to as the 'magic tire model', is categorized as a semi-empirical model, because the parameters within the model are estimated from data and do not correspond to any physical terms, like the parameters in the Fiala tire model. The mathematical structure of the Pacejka model, however, has similarities to the physical based models like the Fiala and Duff's model. The analytical expression for the Pacejka model is given as:

$$F_y(\alpha) = D\sin(C\tan^{-1}(B(1-E)\alpha + E\tan^{-1}(B\alpha))) \tag{2.13}$$

where B is the stiffness factor, C is the shape factor, D is the peak value, and E is the curvature value. The stiffness factor, B, directly affects the slope of the model at small slip angles. The plots in Figure 2.7 illustrate the way the curve changes as these factors vary. All factors have a nominal value of 1.

From Figure 2.7a, the stiffness factor, B, affects the slope of the curve at small slip angles, such that a large coefficient makes the slope steep and a small one makes the slope more flat. From Figure 2.7b, the shape factor, C, similarly affects the slope of the curve, but also affects the amplitude of the curve at small slip angles. From Figure 2.7c, the curvature factor, E, affects the tail-end behavior of the curve. For high values of E, the curve takes on more of an 'S' shape.



(a) Stiffness factor (b) Shape Factor (c) Curvature Factor

Figure 2.7: The Pacejka tire model is characterized by coefficients that manipulate different properties of the force curve, such as amplitude, curvature, and tail behavior.

In many applications, the Pacejka model is simplified by removing the curvature factor, giving the following simplified model:

$$F_y(\alpha) = D\sin(C\tan^{-1}(B\alpha)) \tag{2.14}$$

The Pacejka model remains one of the most popular models in the vehicle dynamics community for its ability to describe a wide spectrum of tires. Since the model factors are determined from experimental data, the Pacejka model yields accurate predictions for tire

force. Through out this work, the Pacejka model is used to describe the tire forces because the model parameters can be optimized to fit experimental data well, and because the model is smooth and continuous, unlike the Fiala tire model. Smooth models are also easier to use with non-linear solvers for the equilibrium analysis discussed later.

Combined Slip Models

Combined side slip models take both the slip angle and the slip ratio into account when computing the lateral and longitudinal forces emerging from the tire. The main difference is that the lateral force and longitudinal force are coupled and are constrained by the limits of the friction circle. The tire slip quantities are computed according to the definitions given earlier, except:

$$s_{x'} = \begin{cases} \kappa = \frac{\omega R - v_x}{v_x} & v_{x'} \geq \omega R \\ \\ \sigma = \frac{\omega R - v_x}{\omega R} & v_{x'} < \omega R \end{cases} \tag{2.15a}$$

$$s_{y'} = \frac{v_{y'}}{wR} \tag{2.15b}$$

$$s = \sqrt{s_{x'}^2 + s_{y'}^2} \tag{2.15c}$$

where $s_{x'}$ represents the tire slip ratio and $s_{y'}$ represents the tire side slip angle, with the tick superscript indicating that the quantities are with respect to the tire frame. By using the small angle approximation, note that $\beta = \tan^{-1} \frac{v_{y'}}{v_{x'}} \approx \frac{v_{y'}}{v_{x'}} \approx \frac{v_{y'}}{\omega R} = s_{y'}$.

The longitudinal and lateral force in the tire frame at each tire are then computed using the following expression:

$$F_{x'} = -\mu F_z \frac{s_{x'}}{s} D \sin\left(C \tan^{-1}(Bs)\right) \tag{2.16a}$$

$$F_{y'} = -\mu F_z \frac{s_{y'}}{s} D \sin\left(C \tan^{-1}(Bs)\right) \tag{2.16b}$$

where μ is the sliding friction coefficient, F_z is the normal force of the tire, and B, C, D are Pacejka model parameters. The combined slip quantity, s, comes into play in the calculation of the friction forces. It enters inside the fractional term, $\frac{s_{i'}}{s}$, which divides the total amount of friction available for between the longitudinal and lateral directions.

Dynamic Vehicle Models

Planar Two-Wheel Model

We now discuss dynamic vehicle models and start with the most basic dynamic model, the planar dynamic bicycle model. This model only considers translation in the x and y direction

and rotation about the $\mathbf{E}_3$ axis. As in the previous section, the vehicle model lumps the front and rear tires to simplify the model by reducing the number of forces acting on the rigid body. For the equation of motion, we assume the vehicle is rear-wheel drive only, meaning the input force from the engine appears only through the term F_x^r, so the front tire longitudinal force is zero (i.e. $F_x^f = 0$).

The state vector is $z = \begin{bmatrix} v_x & v_y & r \end{bmatrix}^\top$, where v_x and v_y are the longitudinal and lateral velocity of the vehicle in the vehicle body-fixed frame, and $r = \dot{\psi}$ is the angular velocity along the $\mathbf{e}_3^v$ axis. The input vector is $u = \begin{bmatrix} \delta & F_{xr} \end{bmatrix}^\top$. The equation of motion in state space form are given below:

$$\dot{v}_x = v_y r + \frac{1}{m}(F_x^r - F_y^r \sin\delta) \tag{2.17a}$$

$$\dot{v}_y = -v_x r + \frac{1}{m}(F_y^r + F_y^f \cos\delta) \tag{2.17b}$$

$$\dot{r} = \frac{1}{I_z}(L_f F_y^f \cos\delta - L_r F_y^r) \tag{2.17c}$$

where F_y^f and F_y^r and the lateral components of the tire force in the front and rear wheel, respectively. The terms $\sin\delta$ and $\cos\delta$ come from projecting the forces in the tire frame into the vehicle frame.

Even with a lumped tire model assumption and neglect of other external forces, the dynamic bicycle model serves as an appropriate model for steering control [9] and lateral control, [29], among many other applications.

Two-Wheel Model Load Transfer

For extreme maneuvers like drift that excite roll and pitch dynamics, it is important to account for the effects of load transfer. Load transfer refers to the change in distribution of normal forces among the wheels during acceleration. Load transfer is a physical phenomenon that occurs in any vehicle just by the principles of Newtonian mechanics. Aside from the rotational dynamics on the vehicle, load transfer impacts how fast a vehicle can accelerate. For example, when the vehicle moves forward and accelerates, more of the normal force is distributed to the rear wheels, meaning the friction circle on the rear wheels gets larger so the vehicle can utilize more of that force to accelerate forward. At the same time, the normal on the front wheels decrease, so they do not have as much capability to propel the vehicle forward as the rear.

For planar models, we can capture the dynamics of load transfer by writing out the equations for the balance of linear and angular momentum. We write the balance of forces in the $\mathbf{e}_3$ direction by equating the normal forces from the tires with the weight of the vehicle

$$mg = F_z^f + F_z^r \tag{2.18}$$

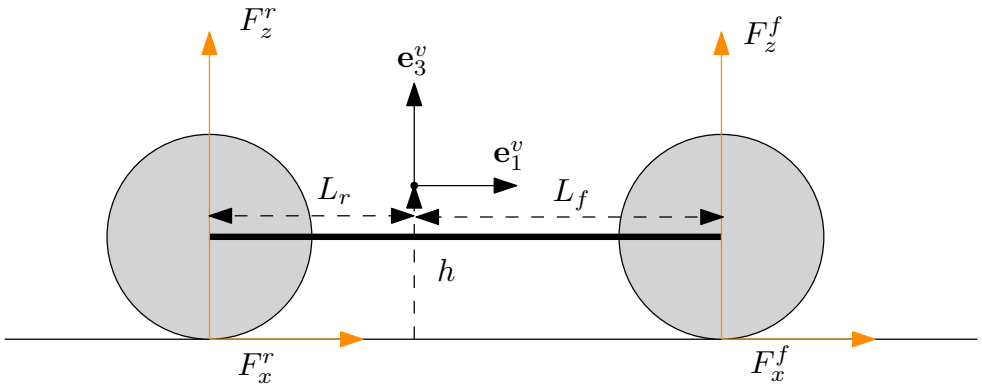


Figure 2.8: The longitudinal and vertical components of the tire forces acting within the $\mathbf{e}_1^v$-$\mathbf{e}_3^v$ plane generate a moment about the $\mathbf{e}_2^v$ axis. The positive $\mathbf{e}_2^v$ axis points into the page.

We also write the balance of angular momentum equation about the $\mathbf{e}_2$ axis:

$$L_f(F_z^f) + h(F_x^f + F_x^r) = L_r(F_z^r) \tag{2.19}$$

From the balance equations in (2.18) and (2.19), and the combined slip tire model in (2.15) and (2.16), we can solve for an analytical expression for the normal force on each tire as:

$$F_z^f = \frac{gm\left(L_r - h\mu\mu_{x'}^r\right)}{L_f + L_r - h\mu\mu_{x'}^r + h\mu\left(\mu_{x'}^f\cos\left(\delta\right) - \mu_{y'}^f\sin\left(\delta\right)\right)} \tag{2.20}$$

$$F_z^r = \frac{gm\left(L_f + h\mu\left(\mu_{x'}^f\cos\left(\delta\right) - \mu_{y'}^f\sin\left(\delta\right)\right)\right)}{L_f + L_r - h\mu\mu_{x'}^r + h\mu\left(\mu_{x'}^f\cos\left(\delta\right) - \mu_{y'}^f\sin\left(\delta\right)\right)} \tag{2.21}$$

where the terms $\mu_{i'}^f, \mu_{i'}^r, i' \in \{x', y'\}$ come from the combined slip tire model as follows:

$$\mu_{i'}^f = \frac{s_{i'}}{s} D^f \sin\left(C^f \tan^{-1}(B^f s)\right) \tag{2.22}$$

$$\mu_{i'}^r = \frac{s_{i'}}{s} D^r \sin\left(C^r \tan^{-1}(B^r s)\right) \tag{2.23}$$

The terms B, C, D come from the Pacejka model and the superscripts indicate which tire the parameter corresponds to, either front or rear.

Planar Four-Wheel Dynamic Model

The four-wheel vehicle model more accurately captures the dynamics since it accounts for the forces generated at each of the wheels.

As with the previous subsection, the state vector is $z = \begin{bmatrix} v_x & v_y & r \end{bmatrix}^\top$, but the input vector is now $u = \begin{bmatrix} \delta & F_x^{rr} & F_x^{rl} \end{bmatrix}^\top$, where F_x^{rr} and F_x^{rl} are the magnitudes of the forces acting on the back-right and back-left wheel, respectively:

$$\dot{v}_x = v_y r + \frac{1}{m}(F_x^{fl} + F_x^{fr} + F_x^{rl} + F_x^{rr}) \tag{2.24a}$$

$$\dot{v}_y = -v_x r + \frac{1}{m}(F_y^{fl} + F_y^{fr} + F_y^{rl} + F_y^{rr}) \tag{2.24b}$$

$$\dot{r} = \frac{1}{I_z}(L_f(F_y^{fl} + F_y^{fr}) - L_r(F_y^{rl} + F_y^{rr}) + c(F_x^{rr} - F_x^{rl})) \tag{2.24c}$$

where we introduce the coordinate transformations from the tire frame, denoted by a tick superscript (e.g. x', y'), to the vehicle frame. Note that the rear wheels are not steered, so only the forces from the front tires need to be projected into the vehicle frame:

$$F_x^{f*} = -F_{y'}^{f*} \sin\delta \tag{2.25a}$$

$$F_y^{f*} = F_{y'}^{f*} \cos\delta \tag{2.25b}$$

where $* = \{r = \text{right}, l = \text{left}\}$ and $\Delta = \{x, y\}$, and c is half the width of the vehicle.

Again, we assume rear wheel drive only, which means that longitudinal forces acting on the front tire are all zero, i.e. $F_x^{fl} = F_x^{fr} = 0$. We also remark that in this formulation, we assume the front wheels point in the same direction, and that the rear wheels also point in the same direction.

By comparing the expression from the lumped tire approximation in the bicycle model, we note that bicycle model assumes $F_*^f = F_*^{fr} + F_*^{fl}$ and $F_*^r = F_*^{rr} + F_*^{rl}$, where $* = \{x, y\}$. In the four-wheel tire model, the forces generated at each tire patch are not assumed to be equal. Furthermore, the rotational dynamics account for the moment generated by the longitudinal tire force at the rear wheels, since the wheels are not acting on the $\mathbf{e}_1^v$ axis, as in bicycle model.

Planar Four-Wheel Model with Wheel Dynamics

The previous model serves well in many applications by accounting for the forces generated at each of the tire patches. A more accurate model, however, accounts for the rotational dynamics of each of the wheels. The wheels turn due to the torque generated by the engine.

The state vector is now $z = \begin{bmatrix} v_x & v_y & r & \omega^{fl} & \omega^{fr} & \omega^{rl} & \omega^{rr} \end{bmatrix}^\top$, where $\omega^{\{*,\Delta\}}$ is the angular velocity of each one of the four wheels, $* = \{f = \text{front}, r = \text{rear}\}$, $\Delta = \{r = \text{right}, l = \text{left}\}$. The input vector is $u = \begin{bmatrix} \delta & T^{rr} & F^{rl} \end{bmatrix}^\top$, where T^{rr} is the torque acting on the back-right wheel and T^{rl} is the torque acting on the back-left wheel.

Note that each of the forces at the wheel patches $F_x^{*,\Delta}$ and $F_y^{*,\Delta}$ are no longer inputs we directly control, but forces generated by deformation of the tires. There are various

functional forms of these forces, but we will defer that to a later section. Also note that we now account for the longitudinal force generated along the front tires F_x^{fl} and F_x^{fr}, but torque inputs do not directly act on those terms, so the magnitudes will be relatively small. Again, we are assuming a rear-wheel drive vehicle:

$$\dot{v}_x = v_y r + \frac{1}{m}(F_x^{fl} + F_x^{fr} + F_x^{rl} + F_x^{rr}) \tag{2.26a}$$

$$\dot{v}_y = -v_x r + \frac{1}{m}(F_y^{fl} + F_y^{fr} + F_y^{rl} + F_y^{rr}) \tag{2.26b}$$

$$\dot{r} = \frac{1}{I_z}(L_f(F_y^{fl} + F_y^{fr}) - L_r(F_y^{rl} + F_y^{rr}) + c(F_x^{fr} + F_x^{rr} - F_x^{fl} - F_x^{rl})) \tag{2.26c}$$

$$\dot{\omega}^{fl} = \frac{1}{I_w}(-F_{x'}^{fl} R) \tag{2.26d}$$

$$\dot{\omega}^{fr} = \frac{1}{I_w}(-F_{x'}^{fr} R) \tag{2.26e}$$

$$\dot{\omega}^{rl} = \frac{1}{I_w}(T^{rl} - F_{x'}^{rl} R) \tag{2.26f}$$

$$\dot{\omega}^{rr} = \frac{1}{I_w}(T^{rr} - F_{x'}^{rr} R) \tag{2.26g}$$

Again, since we now consider the longitudinal force at the front wheels, we introduce the coordinate transformations:

$$F_x^{f*} = F_{x'}^{f*}\cos\delta - F_{y'}^{f*}\sin\delta \tag{2.27a}$$

$$F_y^{f*} = F_{x'}^{f*}\sin\delta + F_{y'}^{f*}\cos\delta \tag{2.27b}$$

$$F_\Delta^{r*} = F_{\Delta'}^{r*} \tag{2.27c}$$

where I_w is the moment of inertia of the wheel and R is the radius of the wheel. More details on the derivation of this system model are given in [31].

Four-Wheel Model Load Transfer

As with the two-wheel model, we can account for the effect of weight transfer by writing out the balance equations for the linear and angular momentum, so that we capture how the normal force is distributed among each of the four wheels. The physical effect of weight transfer is visible to observers when a vehicle aggressively turns left around a corner, the tires on the right side absorb some of the normal force from tires on the left wheels and the vehicle roll angle tilts heavily to the right.

To compute the total normal force on each of the four wheels, $F_z^{i,j}$, where $i \in \{$ f = front, r = rear $\}$ and $j \in \{$ r = right, l = left $\}$, we need four balance equations. The total weight of the vehicle is equal to the sum of the normal forces produced by the tires:

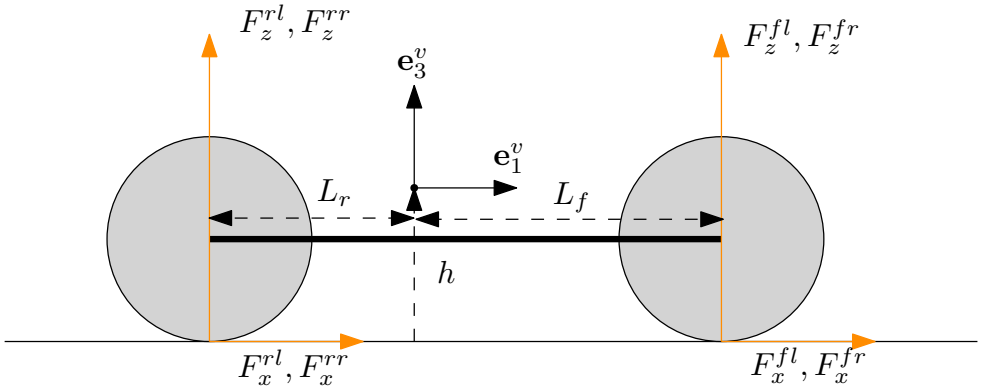


Figure 2.9: The longitudinal and vertical components of the tire forces acting within the $\mathbf{e}_1^v$-$\mathbf{e}_3^v$ plane generate a moment about the $\mathbf{e}_2^v$ axis. The positive $\mathbf{e}_2^v$ axis points into the page

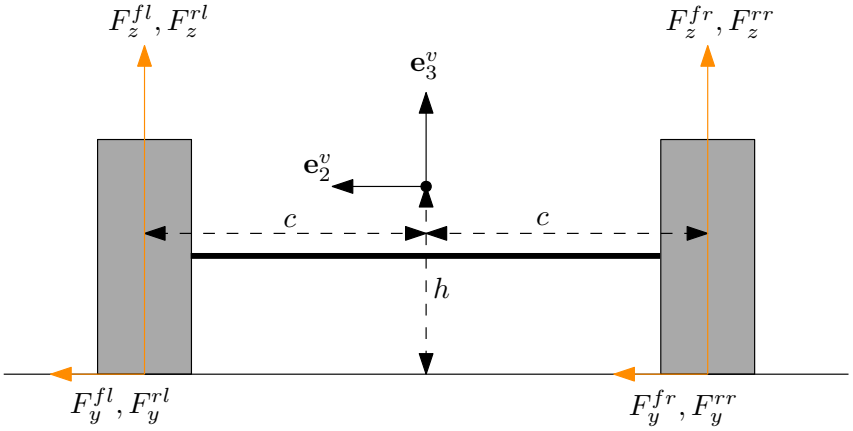


Figure 2.10: The lateral and vertical components of the tire forces acting within the $\mathbf{e}_2^v$-$\mathbf{e}_3^v$ plane generate a moment about the $\mathbf{e}_1^v$ axis. The positive $\mathbf{e}_1^v$ axis points into the page (i.e., we are facing the rear side of the vehicle).

$$mg = F_z^{fl} + F_z^{fr} + F_z^{rl} + F_z^{rr} \tag{2.28}$$

For balance of angular momentum about the $\mathbf{e}_y$ axis, using the diagram in Figure 2.9, we have the following equation:

$$(F_z^{fl} + F_z^{fr})L_f + (F_x^{fl} + F_x^{fr} + F_x^{rl} + F_x^{rr})h = (F_z^{rl} + F_z^{rr})L_r \tag{2.29}$$

For balance of angular momentum about the $\mathbf{e}_x$ axis, using the diagram in Figure 2.10, we have the following equation:

$$(F_z^{fl} + F_z^{rl})c + (F_y^{fl} + F_y^{fr} + F_y^{rl} + F_y^{rr})h = (F_z^{fr} + F_z^{rr})c \tag{2.30}$$

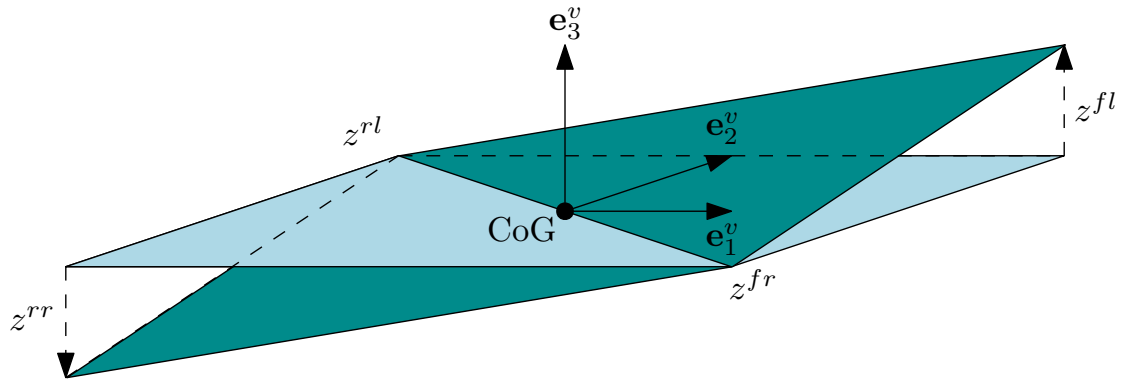


Figure 2.11: The vertical deformation of the tires are balanced in the diagonal direction.

Lastly, since we consider only planar motion, we constrain the vehicle wheels that are diagonal from each other to deform vertically in equal, but opposite, magnitude. This deformation constraint is illustrated in Figure 2.11. We write the constraint as follows:

$$z_{fl} + z_{rr} = z_{fr} + z_{rl} \tag{2.31}$$
$$F_z^{fl} + F_z^{rr} = F_z^{fr} + F_z^{rl} \tag{2.32}$$

The equations for balance of linear and angular momentum, as well as the vertical deformation constraint, constitute a system of linear equations with which we can solve for the normal force on each of the tires. After solving, we have the following set of equations:

$$F_z^{fl} = \frac{1}{4c\left(L_f + L_r\right)}\left(2L_r cgm - 2chF_X - h\left(L_f + L_r\right)F_Y\right) \tag{2.33a}$$
$$F_z^{fr} = \frac{1}{4c\left(L_f + L_r\right)}\left(2L_r cgm - 2chF_X + h\left(L_f + L_r\right)F_Y\right) \tag{2.33b}$$
$$F_z^{rl} = \frac{1}{4c\left(L_f + L_r\right)}\left(2L_f cgm + 2chF_X - h\left(L_f + L_r\right)F_Y\right) \tag{2.33c}$$
$$F_z^{rr} = \frac{1}{4c\left(L_f + L_r\right)}\left(2L_f cgm + 2chF_X + h\left(L_f + L_r\right)F_Y\right) \tag{2.33d}$$
$$F_X = F_x^{fl} + F_x^{fr} + F_x^{rl} + F_x^{rr} \tag{2.33e}$$
$$F_Y = F_y^{fl} + F_y^{fr} + F_y^{rl} + F_y^{rr} \tag{2.33f}$$

The normal forces F_z^{ij} come into play when calculating the longitudinal $F_{x'}^{ij}$ and lateral forces $F_{y'}^{ij}$ in the tire frame from the combined slip model in (2.15) and (2.16). The set of equations in (2.33) are not in closed form, since F_x^{ij} and F_y^{ij} depend on F_z^{ij} from equations (2.15) and (2.16), but F_z^{ij} depends on F_x^{ij} and F_y^{ij} from the set of equations in (2.33). This

issue can be resolved either by using an algebraic engine to solve for a closed-form solution for each tire force F_z^{ij}, or treating all the force variables as dynamic terms, by setting them with initial conditions and then updating their values in simulation at each iteration using the equations in (2.15), (2.16), and (2.33).

2.5 Model Fidelity

The previous sections focused on tire models and planar vehicle models to capture the motion of the vehicle. Model selection directly impacts the performance of model-based control and state estimation. For example, using a kinematic model for a cornering maneuver will yield predictions that diverge significantly from the true path. Even with dynamic models, if the tire model is inaccurate, then the entire system model is compromised, and will not accurately predict the motion of the vehicle. In this section, we briefly discuss the accuracy of two of the dynamic models to model a high slip maneuver.

To assess the accuracy of the system model, we simulate a high cornering maneuver using a two-wheel and four wheel bicycle model with a combined slip tire model, and benchmark the results against the results from CarSim, a high fidelity, commercial vehicle dynamics simulator. CarSim sets the standard for high-fidelity simulated dynamics and has a significant user-base in both the academic and automotive communities.[1]. CarSim does not provide the analytical models used for simulation, but the vehicle state includes hundreds of terms, since subsystems like suspension, power steering, and engine dynamics are all modeled.

We configure CarSim to use the parameters in the table below. The two-wheel and four-wheel models are given the same system parameters and initial state as CarSim.

Table 2.1: CarSim parameters

Parameter	Value
m [kg]	1270
I_z [kg $\cdot$ m^2]	1535
L_f [m]	1.015
L_r [m]	1.895

The tire model parameters from CarSim are used in the combined slip models for the two-wheel and four-wheel vehicle models. The two-wheel and four-wheel models include wheel dynamics and are simulated forward using Euler discretization with a time step of $\Delta t = 10$ ms, over a horizon of $T = 6$ s. We remark that in CarSim, the user can specify steering commands and torque commands to all four wheels. As such, the steering angle of the rear two wheels are set to zero, and the input torques to the front two wheels are set to

[1]For an overview of automotive companies that use CarSim, visit: https://www.carsim.com/company/customers/index.php

zero. We feed the same open-loop sequence of torque and steering commands to CarSim for the four-wheel model and the two-wheel model.

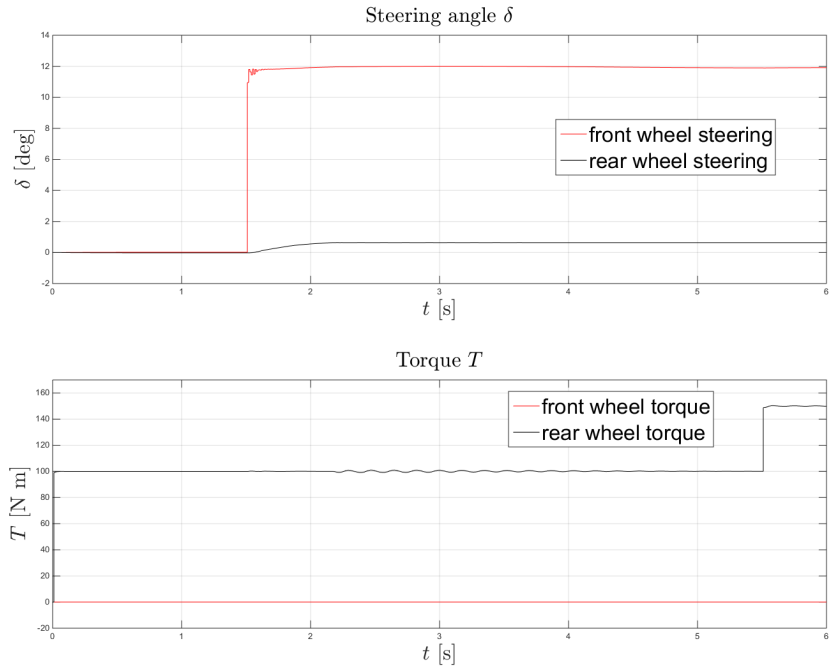


Figure 2.12: All system models initially drive straight and then turn to the left and apply a torque command to the motor.

Figure 2.12 shows the form of the input command we feed into each system model. The inputs are fed into each system over the simulation period, and then the position outputs from each model are recorded.

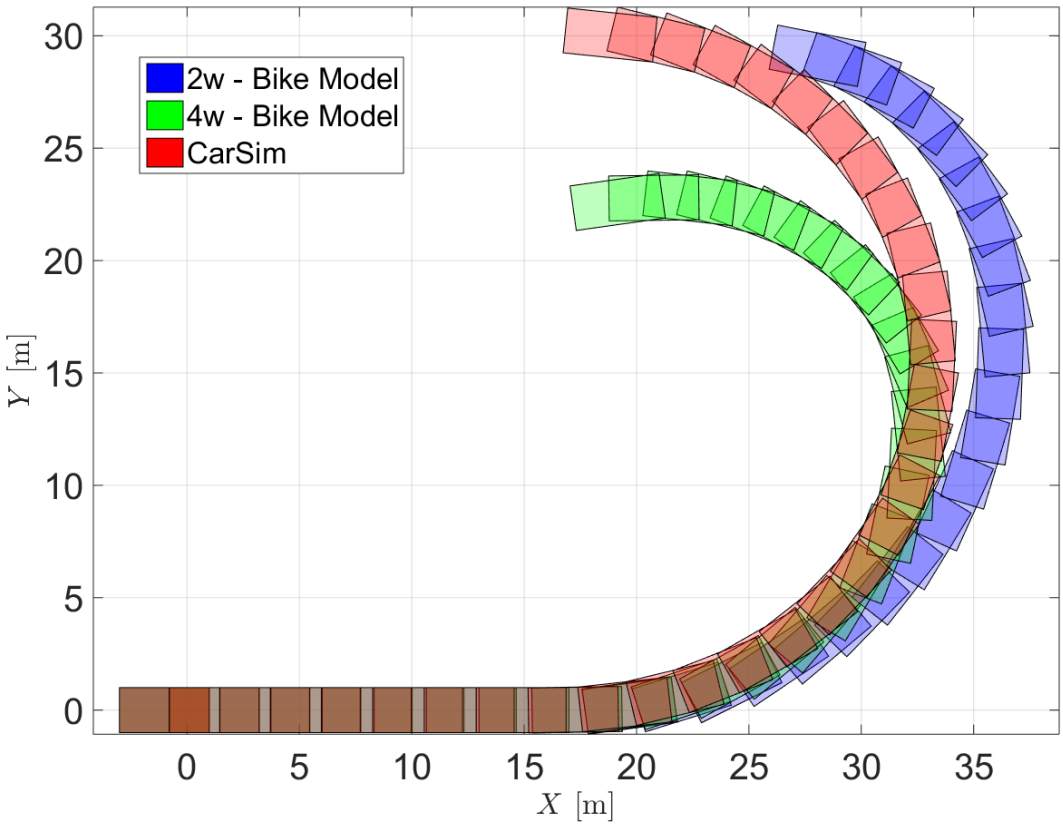


Figure 2.13: The trajectories from the two-wheel (blue) and four-wheel (green) models both diverge from the trajectory of the CarSim model (red) as the vehicle enters into the drift maneuver.

Figure 2.13 illustrates the trajectories using each of the system models in the global frame. The blue and green trajectories represent the position output coming from the two-wheel and four-wheel models, respectively. The red trajectory comes from CarSim and sets the benchmark for comparison. We see that the trajectory of the two-wheel model deviates significantly from the CarSim one, with a difference in final position of over 10 meters. The four-wheel model initial predicts the motion much better, but still diverges midway through the turn, resulting in a terminal position that is several meters off from CarSim.

Figure 2.14 shows the the longitudinal velocity, lateral velocity, and yaw rate over time. The bold curve represents CarSim, the thinner solid curve represents the four-wheel model and the dash thin curve represents the two-wheel model. These curves more clearly illustrate that the yaw rate and lateral velocity, in particular, diverge from the CarSim benchmark as soon as the turning maneuver starts.

From the figures above, both the two-wheel and four-wheel models output the same

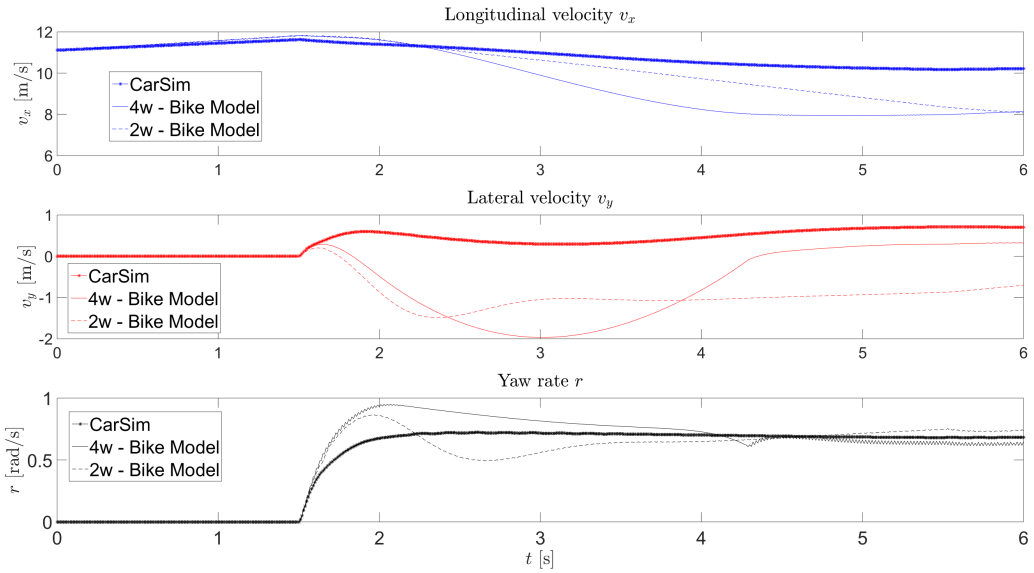


Figure 2.14: The longitudinal velocity (blue), lateral velocity (red), and yaw rate (black) begin to diverge significantly from the CarSim benchmark values as the vehicle initiates the turning maneuver at $t = 5$ s.

vehicle position as CarSim for the straight portion of the trajectory, but then diverge when the vehicle starts cornering. Even with a four-wheel model with a combined slip model and identical parameters to CarSim, the simulated results diverge. The divergence stems primarily from tire model mismatch. While the Pacejka model estimates the forces well at low slip angles, the model loses accuracy at high slip angles when the tire forces are at or near saturation. CarSim uses a more complex tire model that accounts for many more coupled effects between the lateral and longitudinal forces, which become significant at high slip angles. These models include dozens of equations to compute each force component, as in [28].

One of the contributions of this thesis is showing that over a short time duration in an unchanging environment (e.g. race track), systems that operate in open loop behave deterministically, with small variance, yielding very predictable system responses, even when the dynamics are complex and difficult to model. For example, even though the two-wheel and four-wheel models do not accurately predict the position of the vehicle during the high-steer turn, if we perform the maneuver again, from approximately the same initial position and velocity, the resulting trajectory would be nearly the same. This effect makes intuitive sense for many physical systems. If the system evolves again from the same initial state using the same control input over a short time scale, then we would expect the resulting trajectory to be similar to the original one (red path). We demonstrate this later in the dissertation through a mixed open-loop, closed-loop strategy for drift parking and drift cornering.

2.6 Model Identification

In order to apply model-based control techniques, the vehicle and tire model parameters must first be identified for a specific vehicle. In this dissertation, experiments are only conducted on a 1/10-scale remote control (RC) vehicle called BARC, and are explained more in the next chapter.

Mass and Moment of Inertia

For the BARC platform, we measure the mass, m, directly using a digital hanging luggage scale from Etekcity. We estimate the moment of inertia, I_z, by suspending the vehicle using two wires and rotating it, then measuring the frequency of oscillation and then using that information to infer the moment of inertia.

From [21], we have the dynamics of a bifiliar pendulum given as follows:

$$\ddot{\theta} + \left(\frac{K_D}{I}\dot{\theta}|\theta| + \frac{C}{I}\theta\right) + \frac{mgD^2}{4hI}\frac{\sin\theta}{1 - 0.5(\frac{D}{h})^2(1-\cos\theta)} = 0 \tag{2.34}$$

where θ is the angular displacement, h is the vertical displacement from the ceiling, D is the distance between the two wires at the suspension point on the ceiling, g is acceleration due to gravity, and K_D and C are empirically determined aerodynamic and viscous damping parameters, respectively. This equation can be simplified into a linear form by neglecting the damping terms and using a small angle approximation to yield the following:

$$\ddot{\theta} + \frac{mgD^2}{4hI}\theta = 0 \tag{2.35}$$

We can solve for the moment of inertia by substituting a solution form $\theta = A\sin(\omega_n t)$, where A is a constant and ω_n is the frequency of oscillation. After substituting, we have the following:

$$I = \frac{mgD^2}{4h\omega_n^2} \tag{2.36}$$

Equation (2.36) gives a way to estimate inertia through the bifilar pendulum experiment by only measuring the length of suspension wires and timing the period of oscillation.

Tire Model Parameters

The Pacejka and Fiala tire models discussed previously both have sets of parameters that need to be identified in order to estimate the lateral forces. In order to obtain them, we first estimate the lateral forces using a dynamic bicycle model. We consider only the lateral and

rotational dynamics of the vehicle:

$$\dot{v}_y = -v_x r + \frac{1}{m}(F_y^r + F_y^f \cos\delta)$$
$$\dot{r} = \frac{1}{I_z}(L_f F_y^f \cos\delta - L_r F_y^r)$$

From those dynamic equations, we can solve for the lateral force for both the front and rear wheels, giving the following expressions:

$$F_y^f = \frac{I_z\dot{r} + L_r m(\dot{v}_y + r v_x)}{(L_f + L_r)\cos(\delta)} \tag{2.38a}$$
$$F_y^r = \frac{1}{L_f + L_r}\left(-I_z\dot{r} + L_f m\left(\dot{v}_y + r v_x\right)\right) \tag{2.38b}$$

During an experiment, we can measure all the quantities on the right hand side of the equation (2.38) using sensor measurements (e.g. encoders, inertial measurement unit (IMU), camera and GPS). Assuming the IMU is mounted near the center of gravity, we remark that the terms $a_x = \dot{v}_x - r v_y$ and $a_y = \dot{v}_y + r v_x$ come directly from IMU linear acceleration measurements, as shown in the following kinematic equations:

$$\mathbf{v} = v_x \mathbf{e}_1^v + v_y \mathbf{e}_1^v \tag{2.39a}$$
$$\mathbf{a} = \dot{\mathbf{v}} \tag{2.39b}$$
$$= \dot{v}_x \mathbf{e}_1^v + v_x \dot{\mathbf{e}}_1^v + \dot{v}_y \mathbf{e}_1^v + v_y \dot{\mathbf{e}}_2^v \tag{2.39c}$$
$$= (\dot{v}_x - \dot{\theta} v_y)\mathbf{e}_1^v + (\dot{v}_y + \dot{\theta} v_x)\mathbf{e}_2^v \tag{2.39d}$$
$$\mathbf{a} = a_x \mathbf{e}_1^v + a_y \mathbf{e}_2^v \tag{2.39e}$$

where $\mathbf{v}$ and $\mathbf{a}$ are the velocity and acceleration vectors, respectively, in the vehicle body frame. Note that for the yaw rate, we define $r = \dot{\theta}$.

From the equations above, we can estimate the lateral forces at each time step and then solve an optimization program to fit a tire model to the data. We solve for the front tire model with the following optimization program:

$$\min_{B^f, C^f, D^f} \quad \sum_{k=0}^{T} \| F_y^f[k] - F_{\text{mdl}}[k] \|^2 \tag{2.40a}$$
$$s.t \quad \forall k \in \{0, 1, ...T\}$$
$$\alpha^f[k] = \tan^{-1}\left(\beta[k] + \frac{r[k] L_f}{v_x[k]}\right) - \delta[k] \tag{2.40b}$$
$$F_{\text{mdl}}[k] = D^f \sin(C^f \tan^{-1}(B^f \alpha^f[k])) \tag{2.40c}$$

and similarly, for the rear wheel, using the following optimization program:

$$\min_{B^r,C^r,D^r} \quad \sum_{k=0}^{T} \|F_y^r[k] - F_{\text{mdl}}[k]\|^2 \tag{2.41a}$$

$$s.t \quad \forall k \in \{0, 1, ...T\}$$

$$\alpha^r[k] = \tan^{-1}\left(\beta[k] - \frac{r[k]L_r}{v_x[k]}\right) \tag{2.41b}$$

$$F_{\text{mdl}}[k] = D^r \sin(C^r \tan^{-1}(B^r \alpha^r[k])) \tag{2.41c}$$

where B^i, C^i, and D^i are the parameters we want to identify and $\alpha^i[k]$ is the slip angle at time step k. Both programs fit the model to the data by minimizing the sum of residuals of the lateral force over the entire experiment. The constraints reflect the definition of the slip angle for the front and rear tires, as well as define the functional form of the tire model, in this case, the Pacejka tire model.

2.7 Equilibrium Analysis

In this section, we characterize drift by conducting an analysis of the system during steady state drift. This analysis helps build intuition of what is meant by drift in terms of state and input values. Works from [15] and [7] also conduct this analysis as the basis for designing our control policies.

Drift is an advanced driving maneuver that is characterized by rear tire saturation, high side slip angle, and counter-steering. Intentional drifting is beyond the skill set of the average driver, but it is common maneuver in sporting events, like rally racing. From the perspective of an observer, high side-slip corresponds to 'sideways' (i.e. lateral) motion of the vehicle, counter-steering means that the direction of rotation is opposite that of the steering angle, and tire saturation, as discussed earlier, means the tires cannot generate any more force from the friction available. During a turning maneuver with tire saturation, the tires often produce smoke at the tire contact patches and emit an audible high pitch sound.

We now study the dynamics of drift at steady state. Recall that for any system $\dot{z} = f(z, u)$, with z as the state vector and u as the input vector, the equilibrium of the system occur at values z^{eq} and u^{eq}, such that

$$f(z^{eq}, u^{eq}) = 0 \tag{2.42}$$

The equilibria state z^{eq} and input u^{eq} set all of the system derivatives to zero. The steady state definition in (2.42) does not mean that our system state is zero, but that the state and input do not change (i.e. the slip angle, velocity and yaw rate may be non-zero, but they do not change over time). The problem statement is to find values of z^{eq} and u^{eq} such that equation 2.42 holds true.

In the following subsections, we consider two models, the two-state dynamic bicycle model and the three-state dynamic bicycle model. Each model provides some insight into how the state variables interact during drifting conditions.

Two state equilibrium analysis

Recall the dynamic equations for a bicycle model:

$$\begin{aligned}\dot{v}_x &= v_y r + \frac{1}{m}(F_x^r - F_y^f \sin\delta)\\ \dot{v}_y &= -v_x r + \frac{1}{m}(F_y^r + F_y^f \cos\delta)\\ \dot{r} &= \frac{1}{I_z}(L_f F_y^f \cos\delta - L_r F_y^r)\end{aligned}$$

Note that the system dynamics reduce from a set of differential equations to a system of n nonlinear equations with n unknowns, where n is the order of the state.

As we study the dynamics of drift, we remark that steady state implies a continuous drift with an unchanging state $z = \begin{bmatrix} v_x & v_y & r \end{bmatrix}^\top$. For our system, to simplify the analysis, we assume a constant longitudinal velocity, v_x. We treat v_x as a parameter, reducing the system state to $z = \begin{bmatrix} v_y & r \end{bmatrix}^\top$. Assuming $v_y \approx \beta v_x$, for constant longitudinal velocity, we transform the lateral dynamics to side slip dynamics, since drift is characterized by the side slip angle, which encodes information about both the lateral and longitudinal velocity. From the dynamics equations above, we solve the following reduced set of equations:

$$0 = -r^{eq} + \frac{1}{m v_x}(F_y^{f,eq}\cos\delta^{eq} + F_y^{r,eq}) \tag{2.43a}$$

$$0 = \frac{1}{I_z}(L_f F_y^{f,eq}\cos\delta - L_r F_y^{r,eq}) \tag{2.43b}$$

where we treat the longitudinal velocity v_x as a fixed parameter. From equation (2.43), we have two equations but three unknowns, the steering angle, δ, the slip angle, β, and the yaw rate, r. We have an underdetermined system of equations since we have more unknown variables than equations. We resolve this by gridding the value of the steering angle along a range of values, and then iteratively fixing the value of steering angle, then solving the system of equations. In the end, we find equilibrium values β^{eq}, r^{eq} for each equilibrium steering angle δ^{eq}. We can also compute the equilibrium lateral forces $F_y^{f,eq}$ and $F_y^{r,eq}$ since

they are functions of the equilibrium states and inputs.

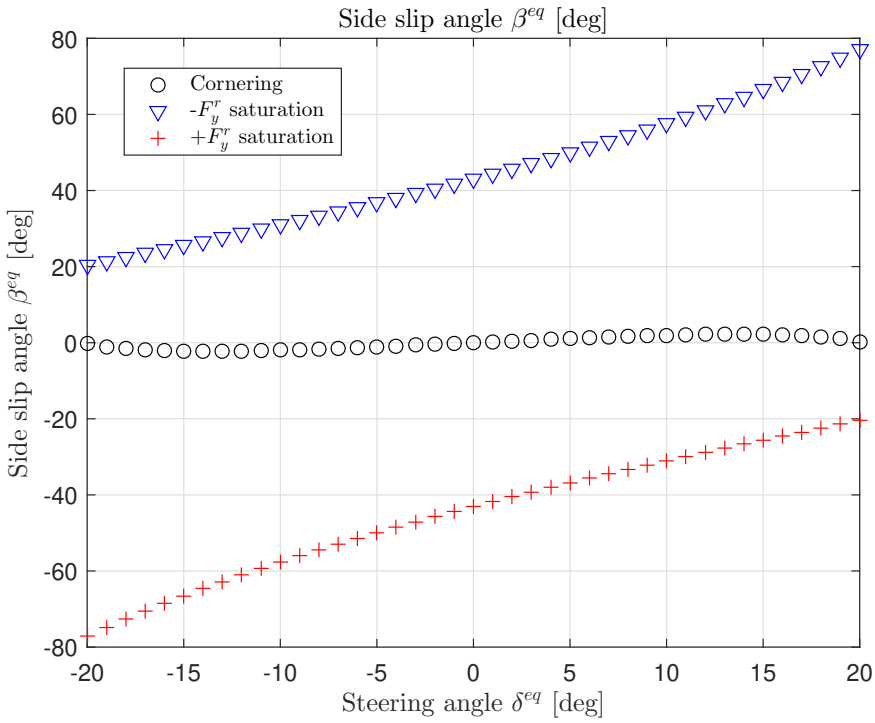


Figure 2.15: The side slip angle β does not grow beyond a couple of degrees under normal cornering conditions (black circled curve). Under rear tire saturation (blue and red curves), the slip angles grow to large values.

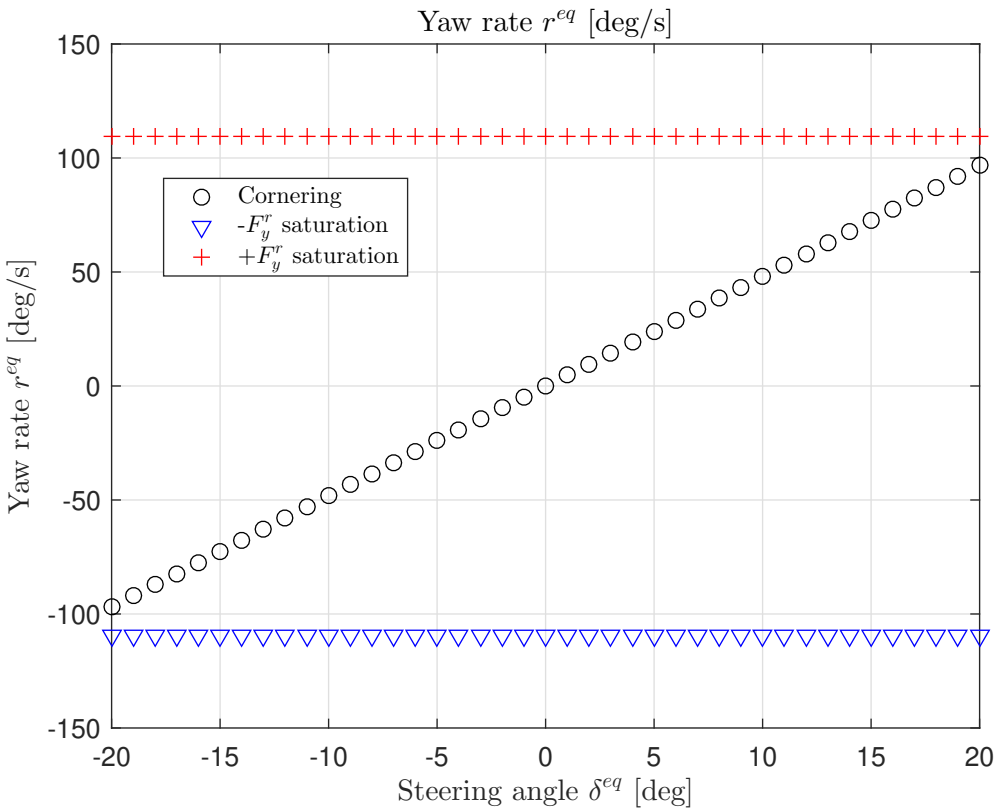


Figure 2.16: The yaw rate r scales linearly with the magnitude of the steering angle under typical cornering conditions (black circled curve). Under rear tire saturation (blue and red curves), the yaw rate maintains a large value.

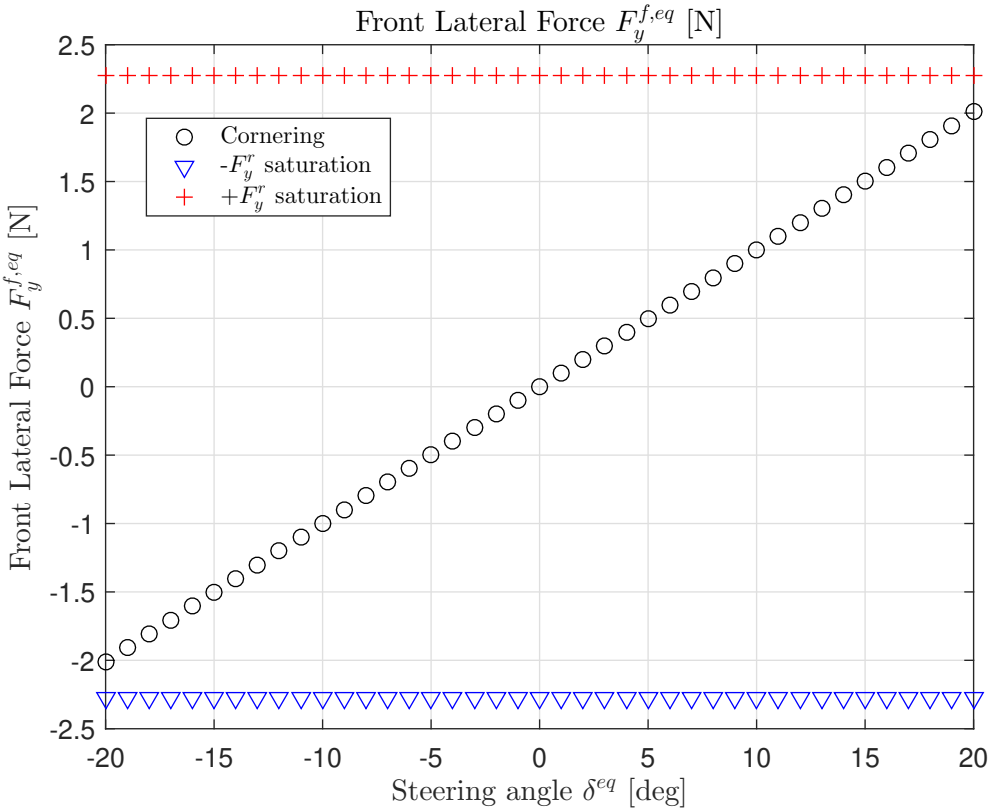


Figure 2.17: The front lateral force $F_y^{f,eq}$ scales linearly with the magnitude of the steering angle under typical cornering conditions (black circled curve). Under drift conditions (blue and red curves), the rear lateral force is saturated and the front lateral force maintains a large value.

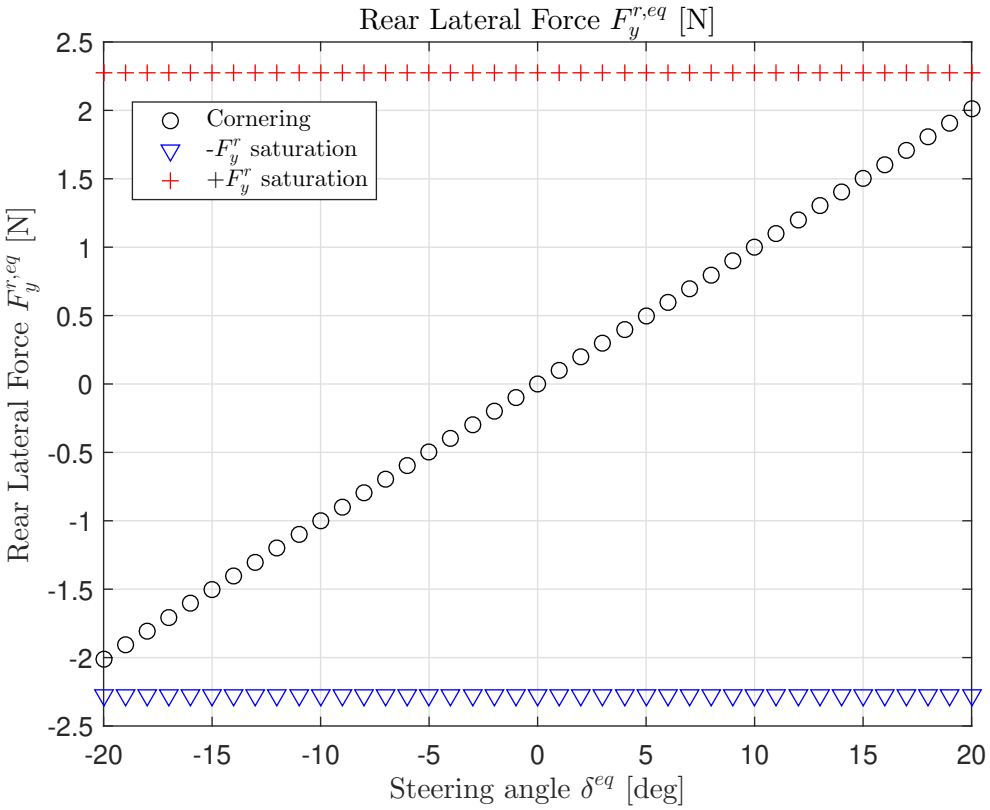


Figure 2.18: The rear lateral force $F_y^{r,eq}$ scales linearly with the magnitude of the steering angle under typical cornering conditions (black circled curve). Under drift conditions (blue and red curves), the rear lateral force is saturated.

We grid the input δ, and repeatedly solve equation (2.43) for the equilibrium states under three conditions. In the first condition, we set the rear lateral force to equal the value predicted by the Pacejka model, so that $F_y^r = D^r \sin(C^r \tan^{-1}(B^r \alpha^r))$. In the second and third condition, we set the rear lateral force to the maximum value given by the friction circle (i.e. $F_y^r = \pm D^r$) to enforce tire saturation in either the clockwise or counter-clockwise direction. We use parameter values from the BARC platform to conduct the equilibrium analysis, given in the table below:

Table 2.2: RC-car parameters

Parameter	Value
m [kg]	1.98
I_z [kg $\cdot$ m^2]	0.24
L_f [m]	0.125
L_r [m]	0.125
B^f, B^r	7.4
C^f, C^r	1.2
D^f, D^r	-2.27

The front and rear wheels have identical tires and approximately the same estimated values for Pacejka model coefficients. We then use the MATLAB nonlinear solver `vpasolve` to obtain a numerical solution.

After solving (2.43) for the equilibrium states and input, we obtain the plots shown in Figures 2.16 through 2.17 . These plots show equilibrium with three sets of markers for each of the three conditions mentioned above that reflect normal cornering and drift (clock wise, and counter-clockwise). The black open circles represent normal cornering conditions, and those with the red pluses and blue triangles represent drifting. The distinguishing characteristic between the two regions is the lateral force. For drifting to occur, the rear tire lateral force must be at or near saturation, which coincides with high slip angle and yaw rate values.

From Figure 2.16, we observe that the slip angle β does not grow beyond a few degrees in the case of typical cornering, when the rear tire lateral force operates in the linear region. When the rear lateral force is set to lie at the limit of the friction circle, the slip angles are large for all values of steering. We observe typical counter steering behavior when examining the slip angle and yaw rate plots from Figure 2.15 and Figure 2.16, respectively. Under drift conditions, the sign of the yaw rate is opposite to the sign of the steering angle and slip angle, meaning that while the rotation of the vehicle body in the inertia frame rotates in one direction, the angle of the steering wheels point in the opposite direction. We remark that with the two-state model we assume the rear tires are saturated entirely in the lateral direction, which is not accurate, but still gives insight for how a more complex model would behave. The equation for the yaw dynamics shows that the magnitude of the front lateral

force F_y^f must nearly balance that which is coming from the rear lateral force in order to achieve a steady state condition

$$L_f F_y^{f,eq} \cos\delta = L_r F_y^{r,eq}$$

which is reflected in the curves in Figures 2.17 and 2.18, and the plots look nearly identical. This result also indicates that rear tire saturation leads, particularly in the lateral direction, to front tire saturation to balance the yaw dynamics at steady state. All forces induced on the front tires will arise in the lateral direction since we are assuming a rear-wheel drive vehicle.

Phase portrait analysis

Phase portrait analysis also gives insight into the behavior of the system around equilibria. These plots illustrate the evolution of an autonomous (fixed input) two-state system starting from several initial conditions. For nonlinear systems, the resulting curves may either converge to stable equilibrium, diverge from an unstable equilibrium, get pulled into a limit cycle, or exhibit chaotic behavior. In general, analytical forms for the curves in phase portraits are solved by dividing the differential equations, isolating state variables terms on either side of the equality, and then integrating. This technique is known as separation of variables (Fourier method):

$$\frac{\dot{r}}{\dot{\beta}} = \frac{-r + (1/mv_x)(F_y^f \cos\delta + F_y^r)}{1/I_z(L_f F_y^f \cos\delta - L_r F_y^r)} \tag{2.44}$$

It is nontrivial, however, to separate the terms, so we instead construct the phase portrait by gridding the state space, and then feeding those points and the initial conditions into the two-state system equation in (2.43), with fixed input, and then use an ordinary differential equation solver to propagate the dynamics forward. Figure 2.19 illustrates the resulting back curves from evolving the system at each grid point. The blue arrows are direction fields that are computed by evaluating the system gradient at each gridded point. Figure 2.19 shows three system equilibria, one stable and two unstable ones, denoted by a crossed, red diamond and an open, red circle, respectively. The direction fields all indicate that the system naturally evolve to the stable state at ($\beta = \beta = -0.0025, r = -1.6927$).

The two unstable equilibria differ in terms of motion. The equilibrium point at ($\beta = 0.3558, r = -1.9130$) indicates that the vehicle rotates in the direction of the turn, while the equilibrium point at ($\beta = -1.3454, r = 1.9130$) indicates counter-steer with very large lateral movement. The opposite signs of the yaw rate r and the steering angle δ indicate counter-steering, and the large slip angle β indicates large lateral motion, or 'sideways' motion. Both points exhibit tire saturation, but the unstable equilibrium at ($\beta = -1.3454, r = 1.9130$) represents the desired drift behavior because of the large side slip angle and counter-steering.

The classification of equilibria as stable or unstable is further verified by linearizing the system at each equilibrium point $\nabla_z f(z, u)$ and computing the eigenvalues. Since we are

considering a continuous system, eigenvalues that are in the negative left half plane have stable poles and thus represent locally stable equilibrium point, or nodes. Conversely, if at least one eigenvalue is in the positive right half plane, then the equilibrium point is unstable. We verify the locally stable properties of the equilibrium point at $(\beta = 0.04, r = -1.77)$ by computing the eigenvalues as shown below:

$$A_1 = \nabla_z f(z = \begin{bmatrix} 0.04 & -1.77 \end{bmatrix}^\top, u = -0.35) \approx \begin{bmatrix} -2.9264 & -0.9997 \\ 0.0031 & -0.3772 \end{bmatrix} \tag{2.45a}$$

$$\lambda_1 = \text{eig}(A_1) \approx \{-2.92, -0.37\} \tag{2.45b}$$

The same computation for the other two equilibria show that they are unstable due to a positive eigenvalue.

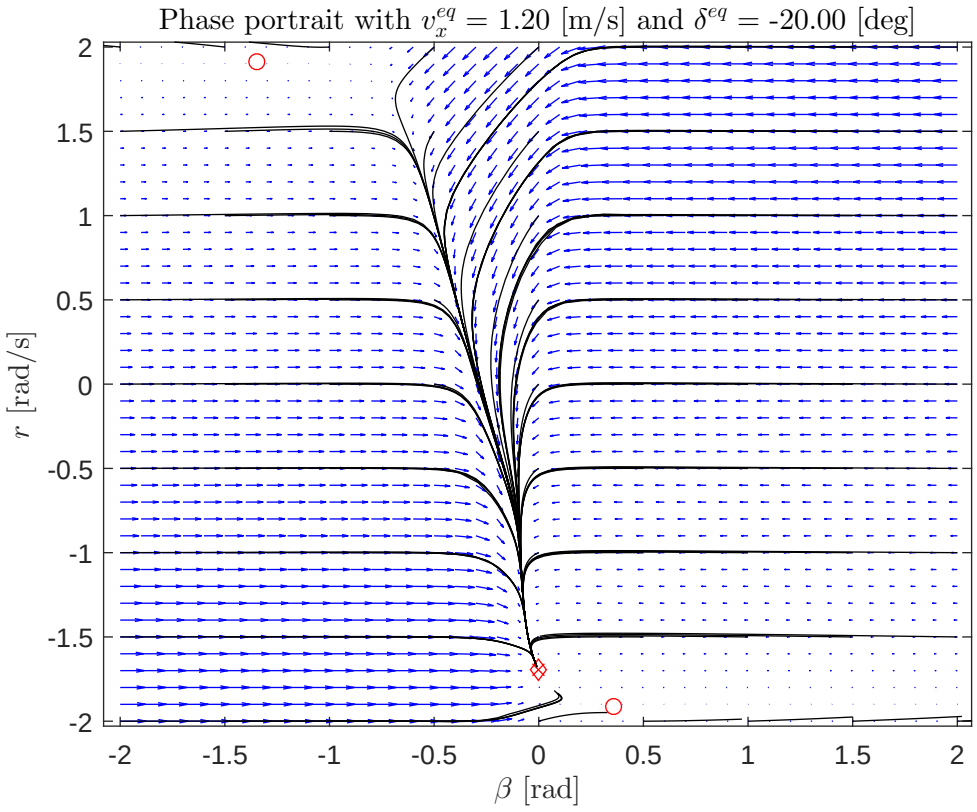


Figure 2.19: The phase portrait illustrates three equilibria associated with the vehicle running at $v_x = 1.20$ [m/s] and δ^{eq} = -20.00 [deg]. The small, open, red circles at the top and bottom of the plot indicate unstable drift equilibrium, and the diamond-shaped red marker around at around ($\beta = 0.04$,$r = -1.77$) indicates a stable equilibrium, or normal cornering condition. The black lines and blue arrows indicate fields or how the system starting at any state within the space shown would evolve. As expected, all systems would move toward the stable equilibrium state.

Figure 2.20 illustrates a phase portrait for the case when the steering is positive, $\delta = +20$. The new phase portrait similarly shows two unstable nodes and one stable node, but with the stable equilibrium point shifted to ($\beta = 0.00$,$r = 1.69$).

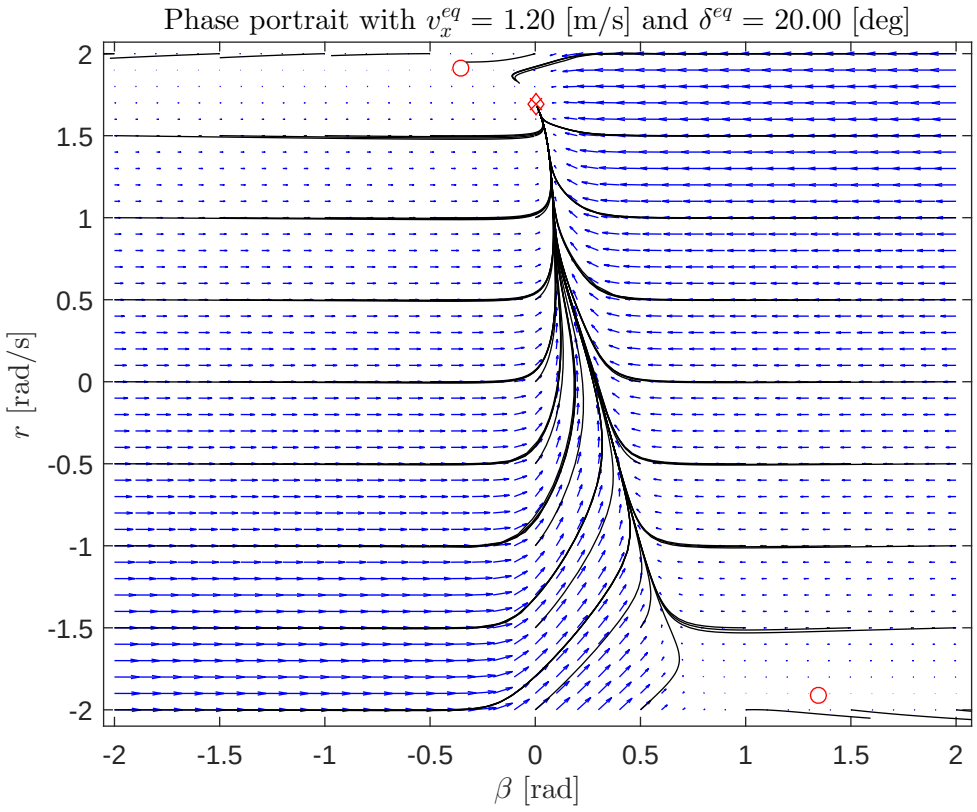


Figure 2.20: The phase portrait illustrates three equilibria associated with the vehicle running at $v_x = 1.20$ [m/s] and $\delta^{eq} = +20.00$ [deg]. The small, open, red circles at the top and bottom of the plot indicate unstable drift equilibrium, and the diamond shaped red marker around at around ($\beta = 0.00$,$r = 1.69$) indicates a stable equilibrium, or normal cornering condition. The black lines and blue arrows indicate fields or how the system starting at any state within the space shown would evolve. As expected, the system moves toward the stable equilibrium state.

Three-state equilibria analysis

The previous section provide insight into the two states of focus during drift dynamics, slip angle β and yaw rate r. This section discusses equilibrium analysis with a three-state, lumped-tire vehicle model, which now accounts for longitudinal dynamics. We have the state vector $z = \begin{bmatrix} v_x & \beta & r \end{bmatrix}^\top$ and the input vector $u = \begin{bmatrix} \delta & F_x^r \end{bmatrix}$. As with the two-state model, the side-slip dynamics $\dot{\beta}$ comes from dividing the lateral dynamics by the longitudinal velocity,

using the approximation $\beta \approx \frac{v_y}{v_x}$. We have the following system model:

$$\dot{v}_x = v_y r + \frac{1}{m}(F_x^r - F_y^f \sin\delta) \tag{2.46a}$$

$$\beta = -r + \frac{1}{m v_x}(F_y^r + F_y^f \cos\delta) \tag{2.46b}$$

$$\dot{r} = \frac{1}{I_z}(L_f F_y^f \cos\delta - L_r F_y^r) \tag{2.46c}$$

With the new system model, we have three equations but five unknowns, the three states v_x, β, r and the two inputs δ, F_x^r. Again, we have an underdetermined system of equations, so we fix the longitudinal velocity v_x and the steering angle δ, and then solve for the remaining variables. The main difference in our analysis between two and three-state models is that we can now observe how the rear longitudinal force F_x^r behaves under steady state conditions.

To conduct the analysis, we first fix the longitudinal velocity v_x, then grid the steering angle δ, and repeatedly solve equation (2.46) for the equilibrium states under the same three conditions as before. In all three conditions, the lateral force from the front and rear tires follow the Pacejka tire model. The difference between the cornering and drifting conditions lies in the constraint of the rear longitudinal force F_x^r. Under typical cornering conditions, we leave the rear longitudinal force as a free variable in the nonlinear solver. Under drifting conditions, however, we constrain the rear force to be equal to the maximum value available provide by the friction circle:

$$F_x^r = \pm\sqrt{(\mu F_z^r)^2 - F_y^r} \tag{2.47}$$

This constraint enforces tire saturation in either the clockwise or counter-clockwise directions. We use the same parameter values as in the two-state model to solve the system of equations. A sample MATLAB program for this three-state equilibrium analysis is given in Appendix A for reference.

Figures 2.21 through 2.25 illustrate the equilibrium values for the states and inputs across gridded steering angles δ for a fixed longitudinal velocity of $v_x = 1.2$ [m/s]. The results for the equilibria slip angle and yaw rate mimic the results for the two-state model, which show small values for cornering and large values for drifting. The new insight comes from the equilibrium values for the rear longitudinal force F_x^r in Figure 2.23. The rear force reaches saturation for large steering angles, regardless of the constraint imposed by equation (2.47), as shown by the converging curves for the cornering and drift conditions. Also, based on the magnitude of the longitudinal rear force, the lateral force takes a large portion of the total frictional force available during drift, which accounts for the large lateral motion or 'side-ways' motion that we observe during drift maneuvers.

Figures 2.26 and 2.27 illustrate graphically the motion of the vehicle under drift conditions, with vectors of the direction of the lateral force on the tires.

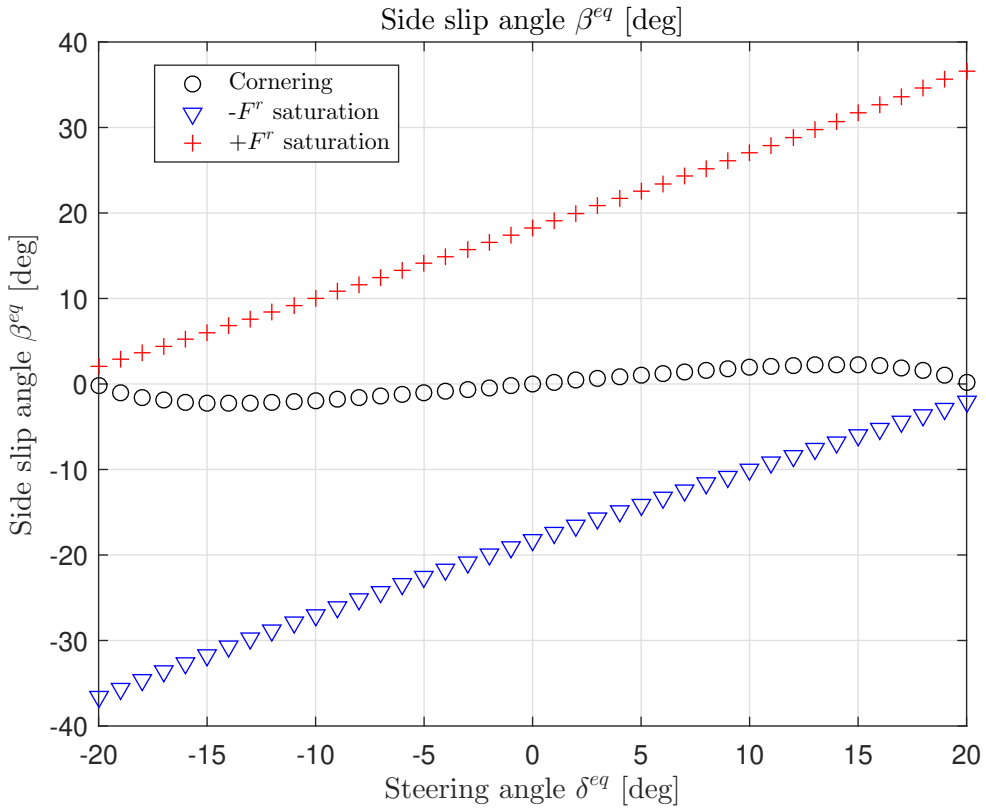


Figure 2.21: The side slip angle β does not grow beyond a couple of degrees under normal cornering conditions (black circled curve). Under rear tire saturation (blue and red curves), the slip angles grow to a large value.

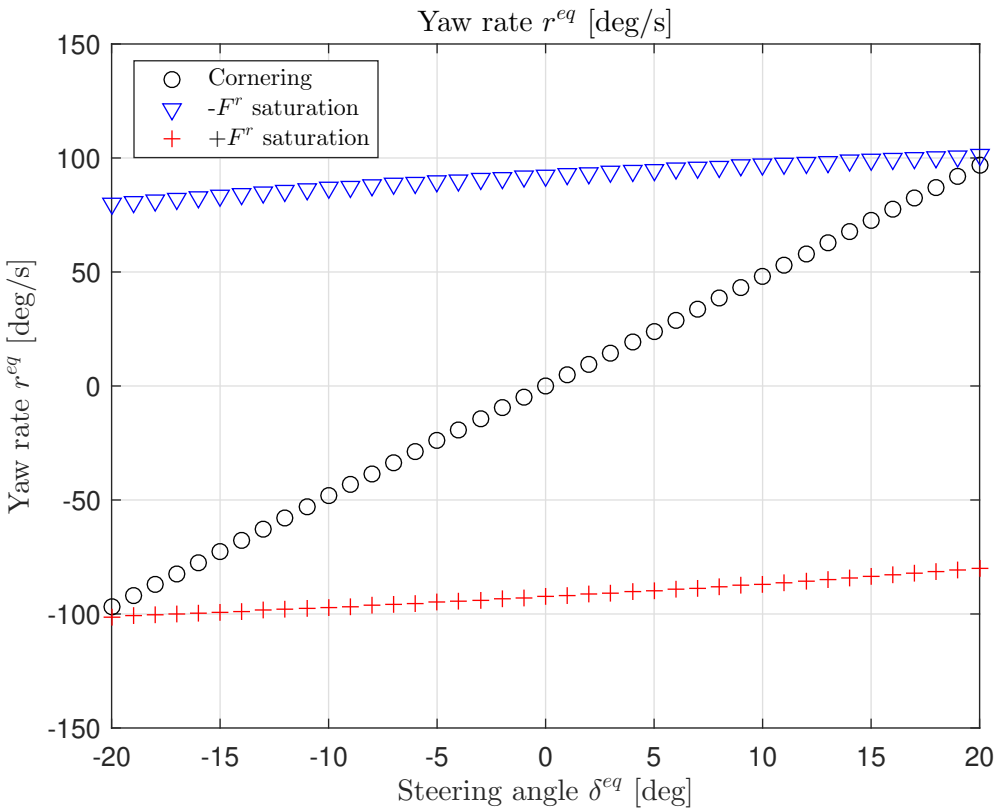


Figure 2.22: The yaw rate r scales linearly with the magnitude of the steering angle under typical cornering conditions (black circled curve). Under rear tire saturation (blue and red curves), the yaw rate maintains a large value.

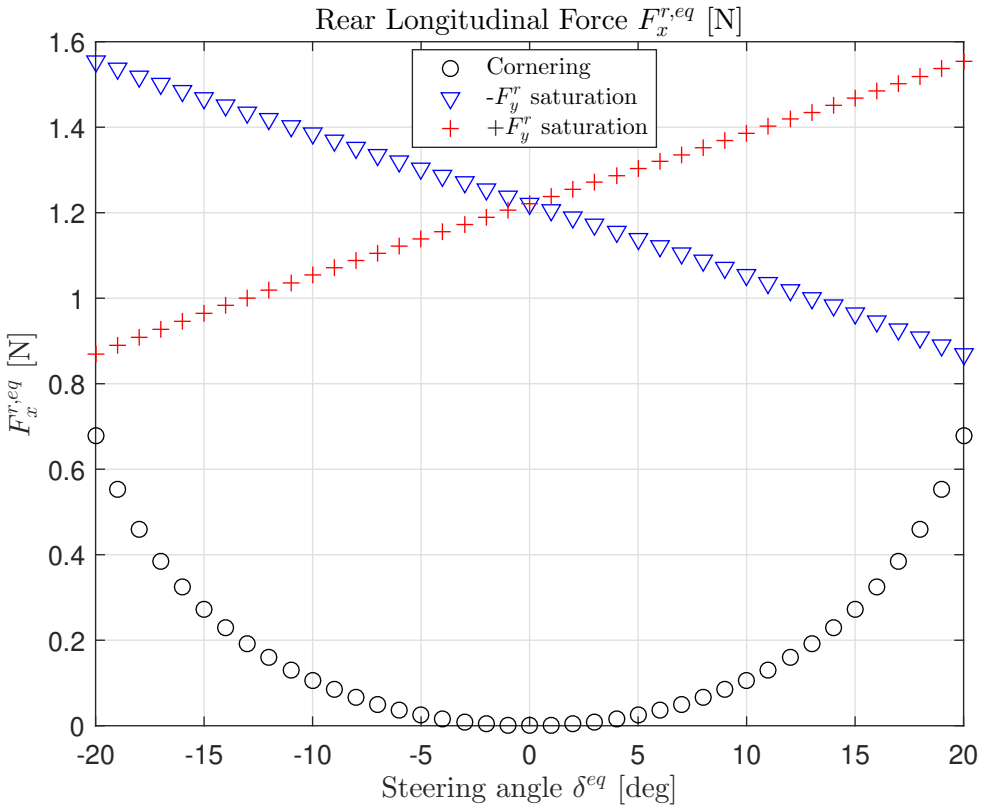


Figure 2.23: The rear longitudinal force F_x^r scales quadratically with the steering angle under cornering conditions. Under drift conditions (blue and red curves), the rear force takes the maximum value possible available from friction, which scales approximately linearly with the steering angle.

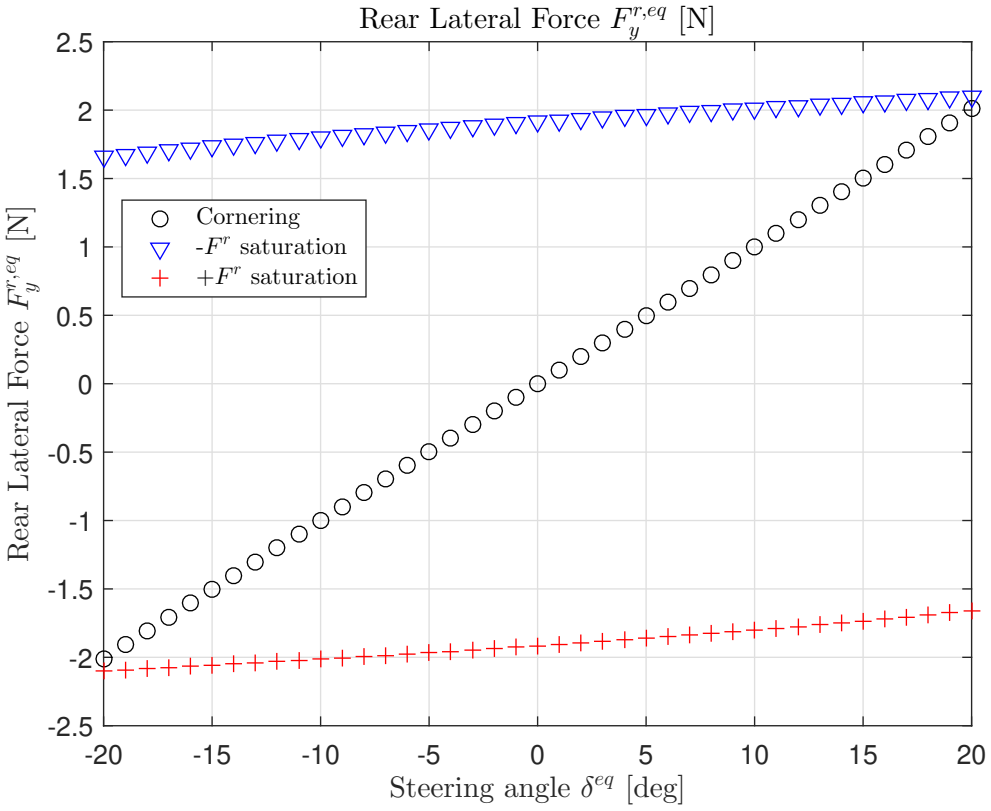


Figure 2.24: The rear lateral force $F_y^{r,eq}$ scales linearly with the magnitude of the steering angle under typical cornering conditions (black circled curve). Under drift conditions (blue and red curves), the rear lateral force is saturated and shares the total available frictional force with the rear longitudinal force $F_x^{r,eq}$.

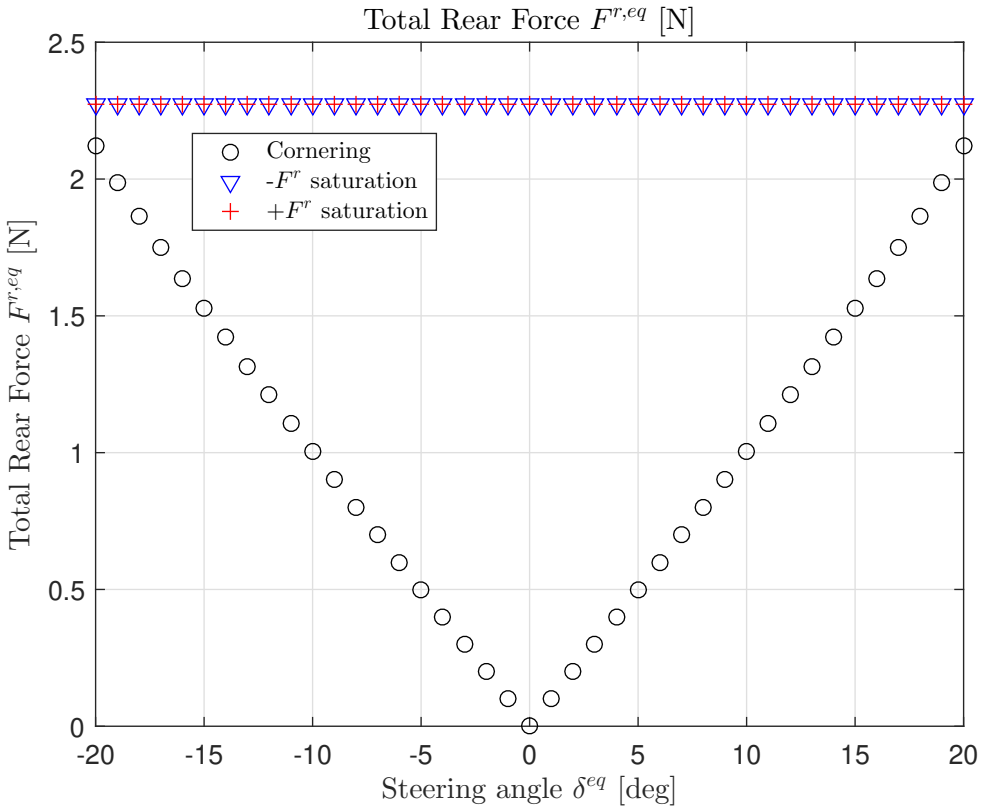


Figure 2.25: The total rear force under cornering conditions scales approximately linearly with the steering angle. This result makes sense as the lateral force operates in the linear region of the tire model. Under drift conditions (blue and red curves), the rear tires are saturated for all steering angles, which results in the straight horizontal line at the top at value (μF_z^r).

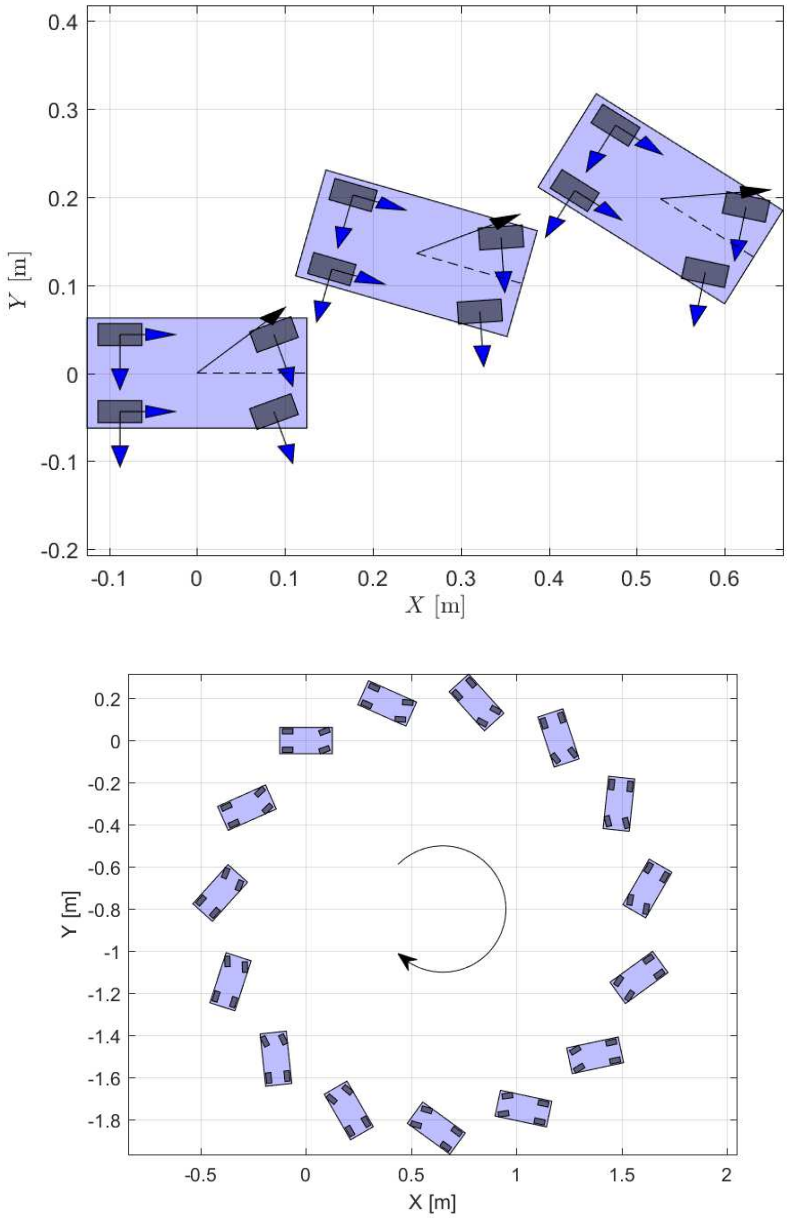


Figure 2.26: Top: The diagram shows the tire forces acting on the vehicle over three time steps. The blue arrows indicate the force vectors acting on each tire. The black arrow is the velocity vector, and the angle between the dashed line and the velocity vector is the slip angle β. Bottom: The bottom diagram shows the complete steady state motion of the vehicle. Equilibrium values: $v_x^{eq} = 1.20$ [m/s], $\beta^{eq} = 36.63$ [deg], $r^{eq} = -79.99$ [deg/s], $\delta^{eq} = 20$ [deg], $F_x^{r,eq} = 1.5535$ [N], $F_y^{2r,eq} = -1.6587$ [N].

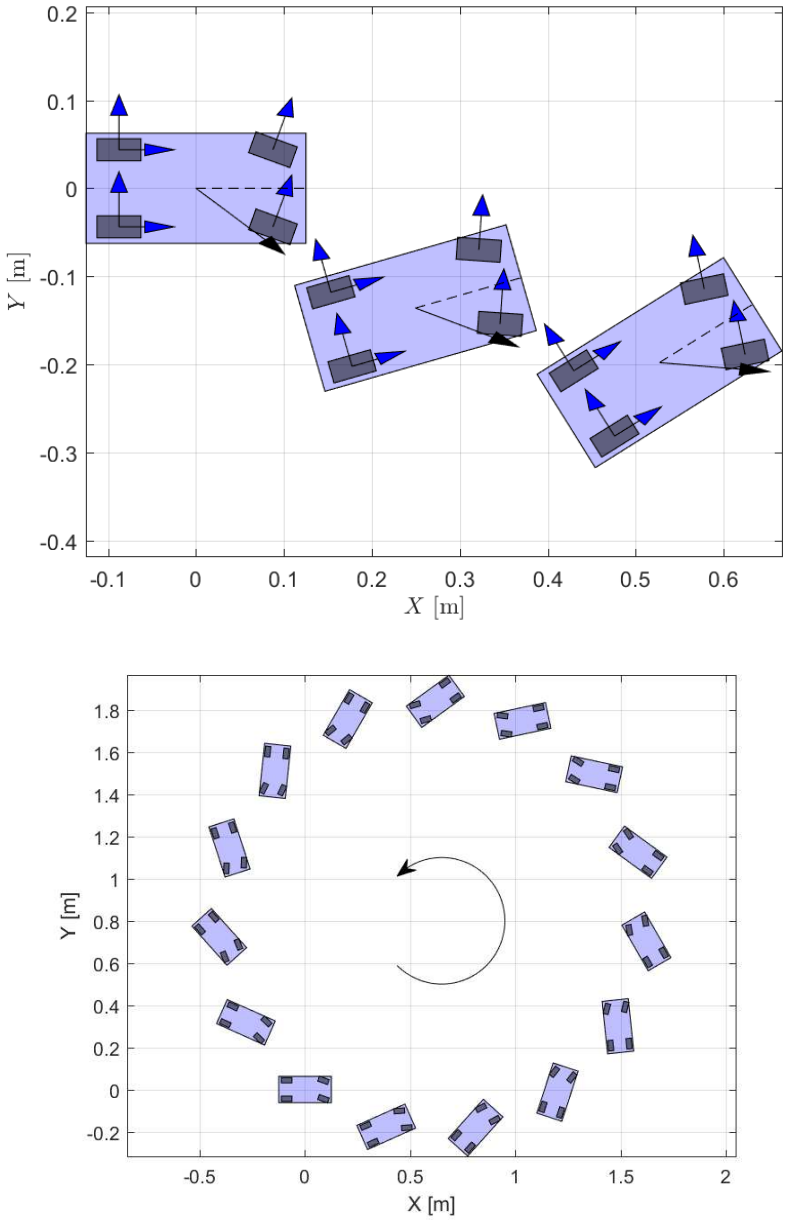


Figure 2.27: Top: The diagram shows the tire forces acting on the vehicle over three time steps. The black arrow is the velocity vector, and the angle between the dashed line and the velocity vector is the slip angle β. Bottom: The bottom diagram shows the complete steady state motion of the vehicle. Equilibrium values: $v_x^{eq} = 1.20$ [m/s], $\beta^{eq} = -36.63$ [deg], $r^{eq} = 79.99$ [deg/s], $\delta^{eq} = -20$ [deg], $F_x^{r,eq} = 1.5535$ [N], $F_y^{r,eq} = 1.6587$ [N]. The equilibrium state represents a mirror opposite of the state in Figure 2.26.

We conclude the section on equilibrium analysis by examining how changes in the nominal longitudinal velocity v_x affect the resulting equilibria plots. The purpose of studying how parameter variation affects the resulting distribution of equilibria would be to check if any bifurcations emerge from our system. Bifurcations occur when small parameter variations cause sudden qualitative changes in the behavior of the system.

Figures 2.28 through 2.30 all illustrate the changes in the equilibrium values when the velocity changes by $\pm$ 0.5 [m/s]. The most notable qualitative trend by increasing velocity is that the magnitude of all equilibrium variables (β, r, F_x^r) decrease as the longitudinal velocity increases for all values of steering angles δ. This trend implies that as the vehicle travels faster, the steering angle does not need to reach as large values as at low speeds in order to saturate the tires. From Figure 2.28, for example, at $v_x = 1.2$ [m/s] the vehicle transitions from cornering to drifting at $\delta \approx \pm 20$ [deg], as indicated by the intersection of the circled and crossed black curves. At $v_x = 1.7$ [m/s], however, the transition occurs at $\delta \approx \pm 10$ [deg]. Conversely, for slower speeds, the transition occurs at a larger steering angle. Qualitatively, at higher speeds, the drift will not look as exaggerated as at lower speeds, since the magnitude of the counter-steer and the slip angle (sideways motion) are relatively small.

2.8 Conclusion

This chapter provided an overview of vehicle models and tire models and examined the merits and limitations of each model, especially in the content of extreme maneuvers like drift. This chapter also examined the phenomenon of drift by conducting a steady state analysis and discussing the resulting equilibrium states and inputs.

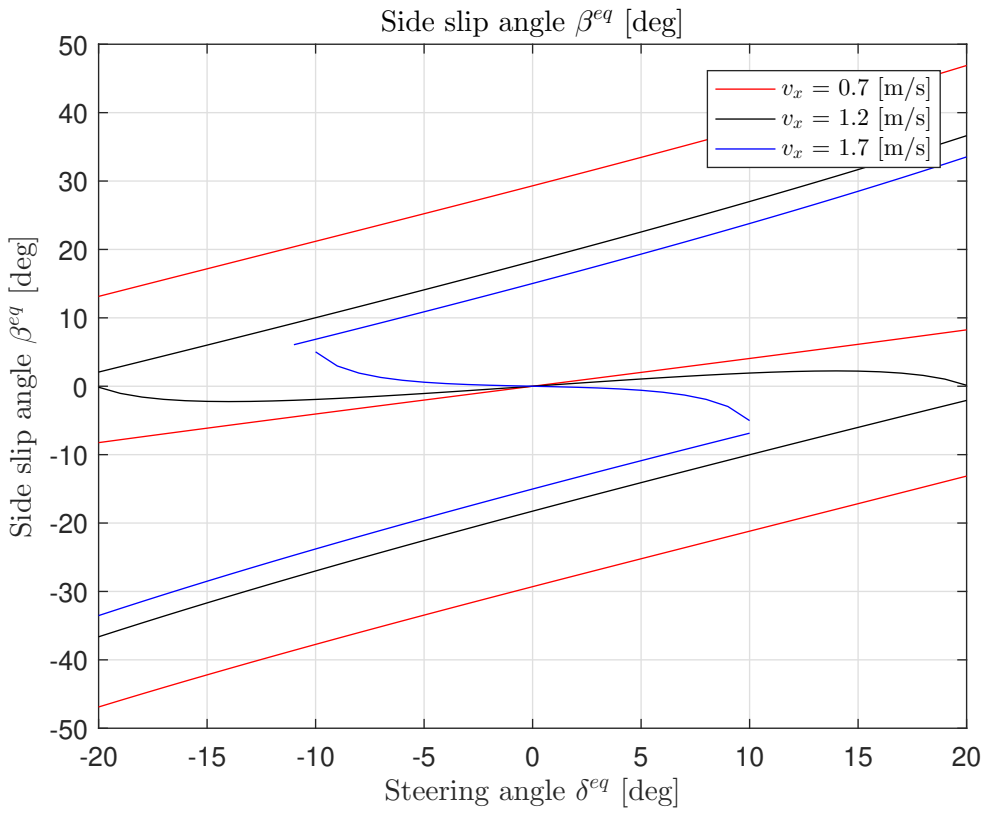


Figure 2.28: During drift conditions, the magnitude of the side slip angle β decreases as the magnitude of the longitudinal velocity increases for all steering angles.

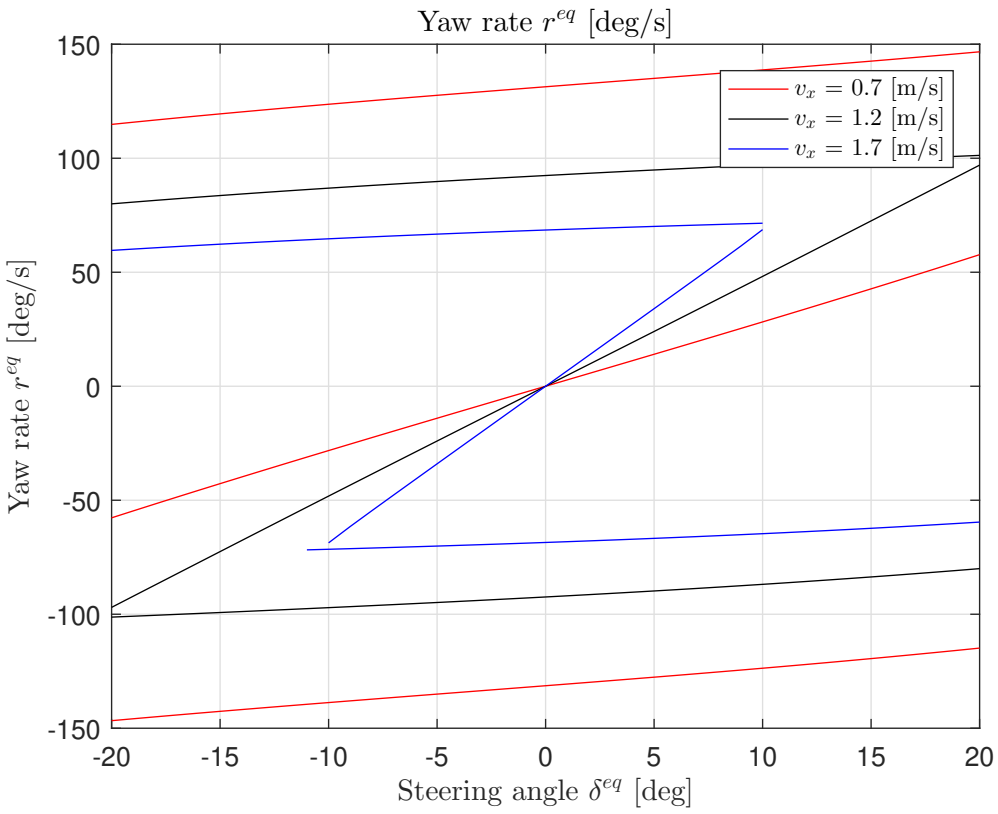


Figure 2.29: During drift conditions, the magnitude of the yaw rate r decreases as the magnitude of the longitudinal velocity increases for all steering angles.

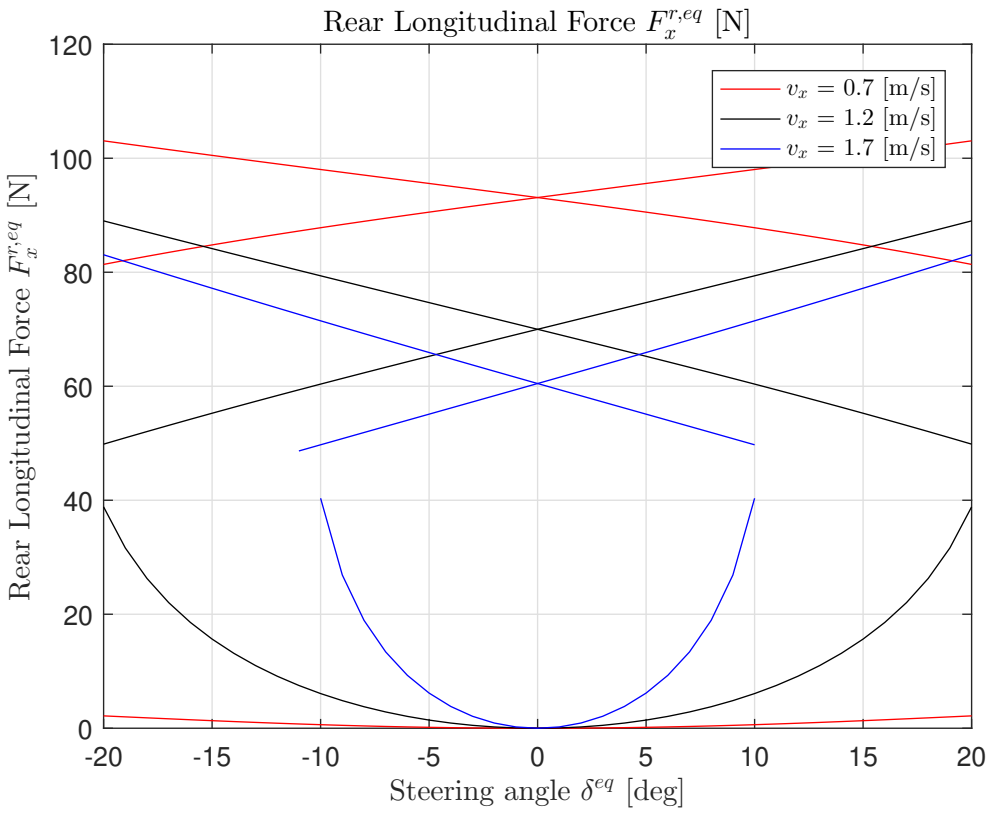


Figure 2.30: During drift conditions, the magnitude of the rear longitudinal force F_x^r decreases as the magnitude of the longitudinal velocity increases for all steering angles.

Chapter 3

Berkeley Autonomous Race Car

This chapter presents a robotic platform developed by the author in the Model Predictive Control lab called the Berkeley Autonomous Race Car (BARC) Project. The BARC platform started as an initiative to bring a low-cost, open-source automotive robot for research and instructional purposes related to autonomous driving. Since the launch of the project in 2015, the platform has gone through many hardware modifications and software revisions to enhance the usability of the platform. The bill of material and source code for BARC have been provided online at http://www.barc-project.com/ in an effort to make our research contributions more accessible to others in the field. To the best of the author's knowledge, the BARC platform has been replicated or been the basis of similar mobile robotic platforms at research institutions across the world, including Seoul National University, Israel Institute of Technology, Clemson University, and ETH, among others.

The BARC platform revolves around a 1/10 scale remote control (RC) that serves as the primary mechanical system. The RC is equipped with a suite of sensors, a microcontroller, cabling, and other peripherals that enable users to program the RC for autonomous driving applications. Figure 3.1 shows the current version of the BARC platform and Figure 3.2 shows the first generation model, which is showcased in many of the project videos uploaded online.

The BARC platform has served as a stable, easy-to-use platform for both research and instruction. A number of research projects, including autonomous drifting, learning-based MPC for racing, vision-based lane keeping, among other algorithms, have all been developed and tested on the BARC. Additionally, the BARC platform has served as the robotic platform for a number of student projects and courses at UC Berkeley, including ME 190J - Model Predictive Control, ME C 231B - Experiential Advanced Control Design II, and ME 131 - Vehicle Dynamics and Control.

At a high level, the BARC platform can be divided into three sections: mechanical design, electrical design, and software design. In this chapter, we give a brief overview of other platforms in the research community, then provide an overview of the design considerations and decisions that went into the development of the BARC platform, and then discuss the recent applications of BARC in the classroom.

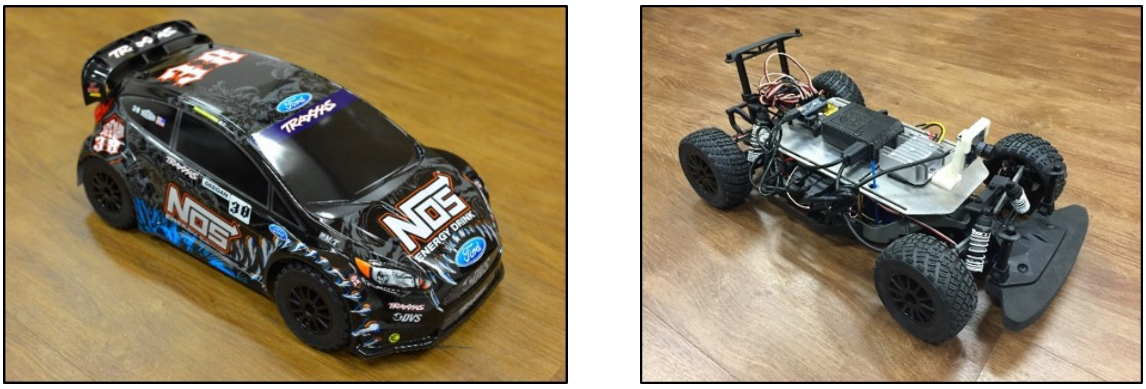


Figure 3.1: Berkeley Autonomous Race Car (BARC) - second generation.

Figure 3.2: Berkeley Autonomous Race Car (BARC) - first generation.

3.1 Platform Review

Over the past five years, several small-scale robotic platforms have emerged in an effort to advance research and development in the space of autonomous driving. Many of these platforms are based on a small-scale race car, typically 1/10 scale, fitted with a mechanical scaffold to support the online electronics.

Karaman and others from MIT Lincoln labs jointly launched an open-source platform called, RACECAR [27], based on a 1/10-scale Traxxas rally car equipped with a sensor stack that includes LiDAR, IMU, a depth camera and a stereo camera. RACECAR uses the Nvidia Jetson TX1 as the main microcontroller to run the control algorithms and perform vision based control. The TX1 features 256 Nvidia CUDA core and a 64-bit CPU, making it a high-performing micro-controller. At the time of writing, Nvidia has launched its second generation micro-controller called the TX2. Overall, the RACECAR platform serves as strong robotic platform for research and education, and has been used for educational outreach projects targeted to high school students.

The University of Pennsylvania has launched a similar 1/10 scale robotic platform and

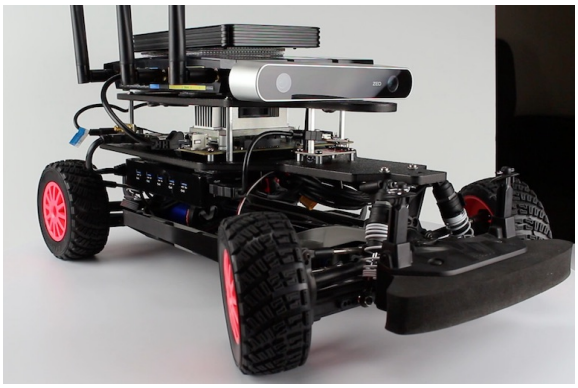


Figure 3.3: The RACECAR is an open-source platform from MIT. Image from [27]

competition series called f1/10th [8] based on a the Traxxas NOS Deegan 38 Rally 1/10 scale car. The hardware is similar to that of the RACECAR from MIT, with an Nvidia Jetson TX1 as the onboard microcontroller and a suite of sensors, including the SparkFun 9 DoF Razor IMU, Hokuyu UST-10LX Scanning Laser Rangefinder (LiDAR), and ZED 2K Stereo Camera, and an additional 3 cell, 11.4 V LiPo battery to support the higher power requirements of the onboard electronics. The home website http://f1tenth.org/index also includes a bill of materials and documentation to replicate the platform.

Figure 3.4: The f1/10 is an open-source platform from Penn Engineering. Image from [8].

Regh and others from the Georgia Institute of Technology have developed a 1/5th scale platform called AutoRally [20]. The RC is based on the HPI Baja 5SC 1/5 Gas RC and has a sensor suite that includes a Flea3 Color USB 3.0 Camera, a Microstrain 3DM-GX4 -25 IMU and a NavTech Eclipse P306 GPS module. The computing platform is based on ASUS Z170I motherboard with an Intel LGA 1151 processor. The platform is very specialized with an exhaustive bill of materials that specifies many of the mechanical components for the RC chassis, including the ESC, servo, suspension system, among other parts. This type of construction differs from the chassis in the BARC, RACECAR, and f1/10 platforms, which are sold as complete ready-to-go units from distributors, in that several mechanical

components are acquired separately from the chassis frame. Additionally, the HPI Baja 5SC 1/5 RC uses gasoline as the energy source for the engine, whereas the other 1/10 scale platforms use either NiMH or LiPo batteries.

Figure 3.5: The AutoRally Robot is a testbed for perception and control from the Georgia Institute of Technology. Image from [20].

Within the last couple of years, a community of hobbyists and engineers have developed a low-cost robotic platform called Donkey. The chassis is based on a 1/16 Scale Truck from Exceed with a Raspberry Pi 3 as the computing hardware. The primary sensor is the wide-angle Raspberry Pi camera. The Donkey car has become popular among electronic hobbyists because of the low-cost and organized competitions and meetup to promote the development of the platform.

Figure 3.6: Donkey is an open source project for self-driving cars. Image from [34].

In terms of commercial products, an Israeli company called Cogniteam has released the Hamster Micro UVG [4]. The chassis is custom designed from Cogniteam and comes equipped with two Raspberry Pi 3 as the computing platform. The sensor suite includes a 360 degree LiDAR, HD Camera, 3D Compass, GPS and wheel encoders.

Overall, all these RC platforms, from research groups, hobbyists, and companies have developed into strong platforms for specific applications and target audiences. The BARC

Figure 3.7: Hamster is a small robust autonomous robot for research and prototype development from Cogniteam. Image from [4].

platform we present in this thesis is best categorized as a relatively low-cost, open-source RC platform for research and instruction. The bill of material for the BARC platform is about 650 USD at the time of writing (excluding LiDAR and GPS) and has an assembly time of two to three hours. The RACECAR and f1/10 platforms share a lot of similarities with the BARC in terms of chassis selection, but differ in choice of computing hardware, power distribution scheme and sensor suite. The f1/10 has a more powerful computational platform with the Nvidia TX1 (≈500 USD) because of the GPU, but it comes at a cost six times that of the Odroid XU-4 (≈70 USD). The BARC platform uses a single battery to power the actuators and on-board electronics, whereas other platforms have two separate batteries, which adds to weight and cost. Additionally, the BARC platform takes an estimated three hours to assemble, which is comparable to the f1/10, but the bulk of assembly time with the BARC platform comes from preparing the encoder units (e.g. soldering, coloring, mounting), which other platforms, including f1/10, do not include.

3.2 Mechanical Components

The main RC mechanical components of the vehicle include the chassis, suspension, transmission system and wheels. The on-board electronics control the mechanical system. We discuss the selection of the chassis and the fabricated components to support and mount the electronics.

Chassis

The BARC platform uses the Ford Fiesta ST Rally 1/10 scale electric RC car from Traxxas as the chassis. The Rally RC comes as a ready-to-go unit, equipped with a Titan 12 turn 550 motor, a Traxxas 2056 High-torque servo, XL-5 Electronic Speed Control (ESC) unit and a 2.4 GHz Top Qualifier (TQ) Radio Transmission system for remote control. The Ford Fiesta

has a suspension system with oil-filled shocks, and has a four-wheel drive configuration, with a transmission shaft transmitting torque from the rear axle to the front axle. The RC car is built larger and more rugged than other 1/10-scale RC cars because it is designed for outdoor rally racing.

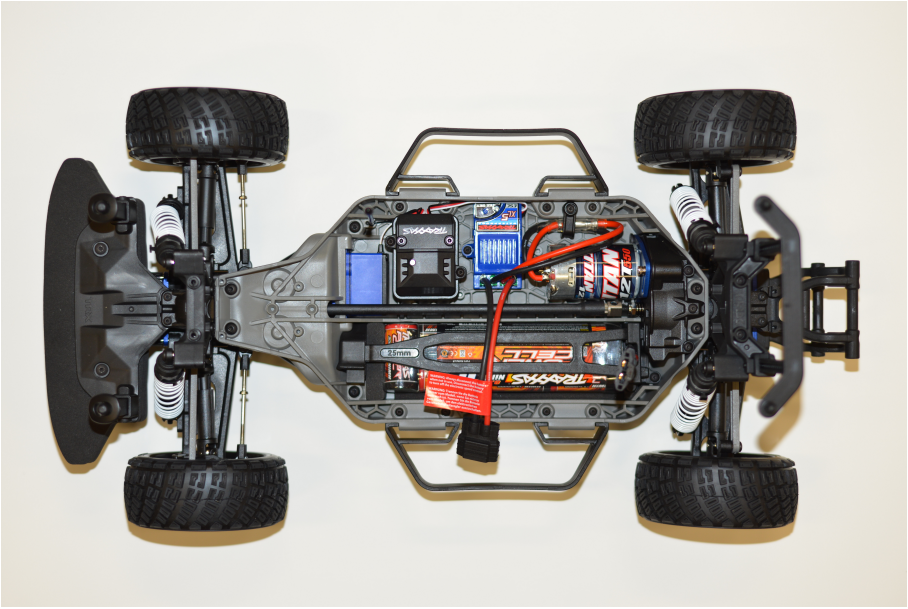


Figure 3.8: Top view of Traxxas Ford Fiesta ST Rally 1/10.

(a) XL-5 Electronic Speed Control

(b) Titan 12 turn 550 motor

Figure 3.9: The Traxxas Ford Fiesta comes equipped with a Bushed DC motor and an Electronic Speed Control unit.

The Titan 12-turn 550 motor operates as a brushed DC electric motor. Internally, the motor has a coil that is wrapped around armatures that extend from a central axle. Magnets are placed along the interior surface of the motor case, and when a current is passed through

the coil, a magnetic force is induced that causes the armature to rotate. To make the armature rotate in the same direction, a commutator at one end of the motor makes the current reverse direction. Brushed DC motors are identified by the two terminals (positive and negative) at the side. Brushless DC motors, on the other hand, usually have three terminals.

The XL-5 Electronic Speed Control (ESC) unit controls the amount of current and voltage that is delivered to the motor. The ESC has four power ports, two ports (positive and negative terminals) are for power from the onboard battery, and the other two ports are to deliver voltage and current to the motor. The ESC receives control signals in the form of pulse width modulation (PWM) signals from the receiver-transmitter box through Futaba cables. The receiver-transmitter box in turn receives command signals wirelessly from the user through the remote-control joystick.

Pulse-width modulation (PWM) signals are modulated voltage signals that can transmit encoded information or power through a voltage square wave. The PWM signal operates only at either high or level and transmits information based on the duty cycle, the fraction of the time the voltage is high over a signal period. Figure 3.10 illustrates the relationship between duty cycle and average power delivered (dashed-red line). The Traxxas XL-5 ESC interprets the PWM signals as voltage levels, and then applies gains to those signals which are then sent to the motor. For our platform, we directly send PWM signals to the ESC to control both the motor and the servo.

In the initial planning of the BARC platform, we considered designing a custom chassis, but due to time constraints and performance requirements, we decided to use a chassis from a professional RC manufacturer. The RC chassis from Traxxas has already been engineered with performance and cost in mind, and usually comes with product warranties and reliable customer service. One particularly important downside to note, however, is the uncertainty of when the product will be discontinued and updated with a newer model. We encountered this problem with the first generation BARC platform based on the Basher RZ-4 1/10 Rally Racer from HobbyKing. The chassis was selected when the project began in fall 2015 due to low-cost, good performance, durability, and high availability, but as of summer 2017, the Basher model was discontinued and we had to switch to another RC model as the primary chassis for the BARC platform.

Manufactured Components

The BARC platform requires mechanical structures to support to the microcontroller and sensors. We designed a 'top deck' plate, shown in Figure 3.11, to overlay on the central part of the chassis. Four holes on the perimeter of the deck are used to mount it to the Traxxas chassis using hexagonal M3 standoffs. The other holes in the interior are used to fasten sensor mounts to the deck. We choose Aluminum 5052-H32 as the plate material with chromate conversion coating using Alodine as a finish to ward off corrosion.

The interior holes are fitted with CLSM3-2 Self-Clinching Nuts from PennEngineering (PEM). The self-clinching nuts adheres itself to the plate through application of a parallel

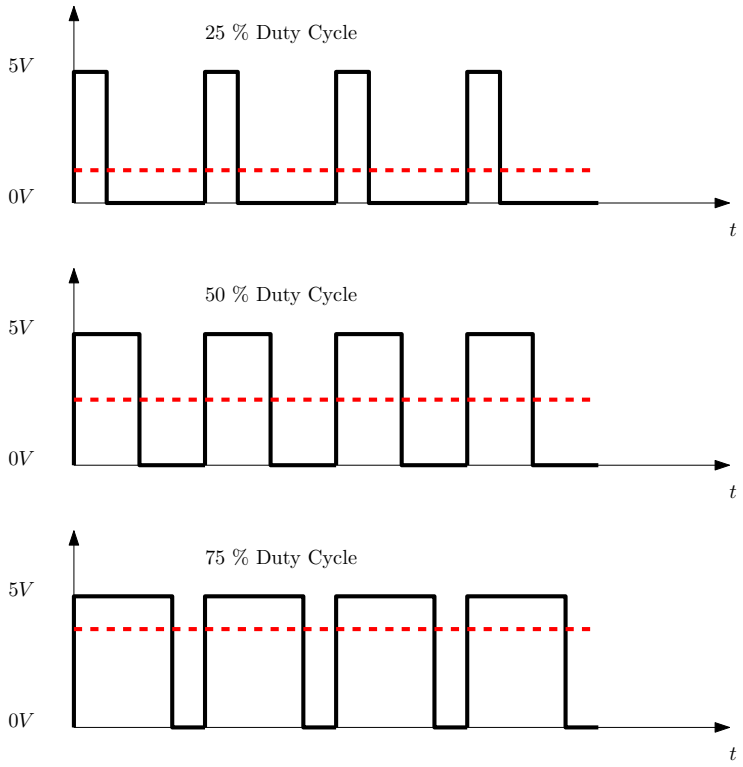


Figure 3.10: Pulse Width Modulation can encode information or control power delivered to electrical devices. From top to bottom, the PWM curves illustrate 25%, 50%, and 75% duty cycles, respectively. The red dashed line shows the average voltage value over the cycle. A higher duty cycle means more power is delivered to connected devices.

squeezing force on either side of the nut. These nuts have an M3 internal thread that allows M3 screws to be fastened to it securely. The nuts are plated with zinc to prevent material damage from using stainless M3 screws. The top plate is manufactured using a water jet cutter. The schematic for the top plate is included in Appendix B.

In addition to the main plate covering chassis, we also designed mounts for the sensors. All the sensor mounts are fabricated through a fused deposition modeling process via a 3D printer. All sensor components are fabricated using Acrylonitrile Butadiene Styrene (ABS). The latest designs for the sensor mounts are available on the BARC project website.

3.3 Electrical Components

Autonomous vehicles need sensors to gather information about the motion of the vehicle and about the surroundings, and a microcontroller to process that information and make control actions. In this section, we outline the sensor selection and the computing platform for the BARC platform. We choose low-end, low-cost devices in order to keep the cost of

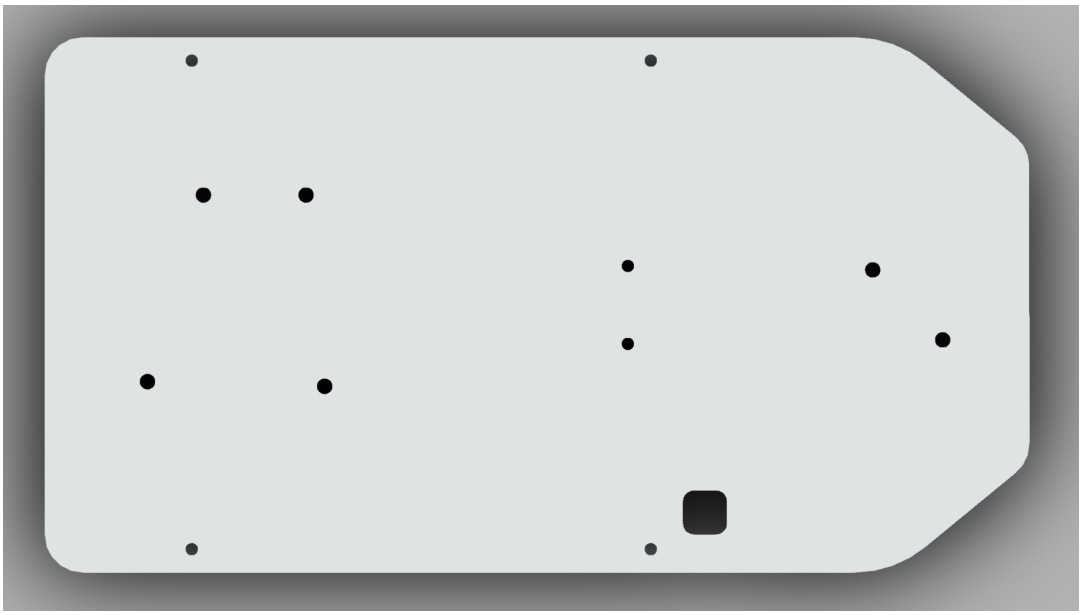


Figure 3.11: Top view of the BARC chassis base plate.

the platform relatively low.

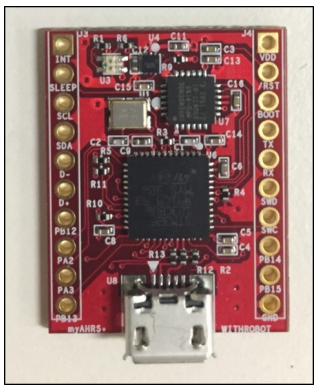
Inertial Measurement Unit

The inertial measurement unit (IMU) measures linear acceleration, angular velocity, and orientation using a combination of accelerometers, gyroscopes, and magnetometers. We use the myAHRS+ IMU from HardKernel, shown in Figure 3.12a, for the BARC platform due to low cost, reasonable performance, and online support and documentation.

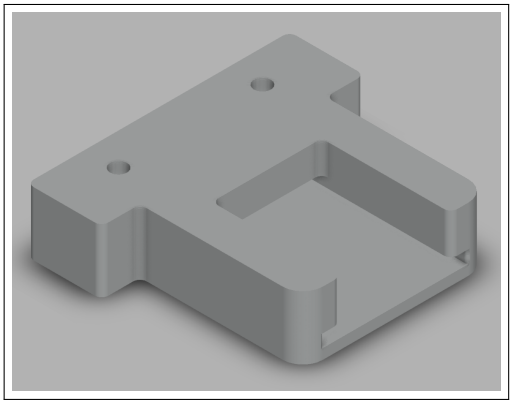
From the specification data [12], the myAHRS+ IMU can measure up to $\pm 16g$ for acceleration, ± 2000 degrees per second (dps) for rotation, and $\pm 1200\mu T$ for orientation. The IMU has a micro-USB port that is used to read measurement data through serial communication. Figure 3.12b shows the mount designed to house the IMU. We place the IMU on the underside of the chassis near the center of gravity of the chassis.

Encoders

Encoders provide rotational information about spinning parts. We designed a basic encoder unit using the QRE113 Miniature Reflective Object Sensor from SparkFun, shown in Figure 3.13a. The sensor has a phototransistor that sends a signal based on the amount of light reflected from a close surface. We design a mount and a disk with colored partitions for the sensor to detect. The encoder is mounted on the hub of each tire and the disk is mount on the interior rim of each tire, as shown in Figures 3.13b and 3.13c respectively. Whenever the disk segment in front of the sensor switches from a light to dark surface, the QRE113



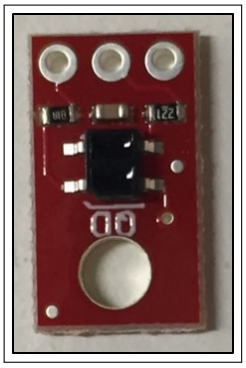
(a) myAHRS+ IMU sensor



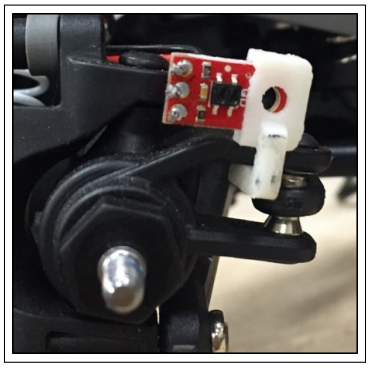
(b) ABS mount for IMU

Figure 3.12: The myAHRS+ (Altitude Heading Reference System) from HardKernel features an accelerometer, gyroscope, and magnetometer.

sends a signal which is detected by the microcontroller. The microcontroller has an interval variable that keeps a tally, and increments it every time a switch occurs.



(a) QRE113 Sensor



(b) Encoder mount

(c) Encoder disk

Figure 3.13: The encoder unit consists of a QRE113 Reflective Object Sensor from SparkFun that detects changes in changes from the encoder disk.

Using this encoder setup, we estimate the velocity of the wheel using two methods. In the first method, assuming entire forward or backward motion, we count the number of times the disk has alternated from a dark to light partition, over a fixed time interval, and then scale the result to the wheel's geometric proportions.

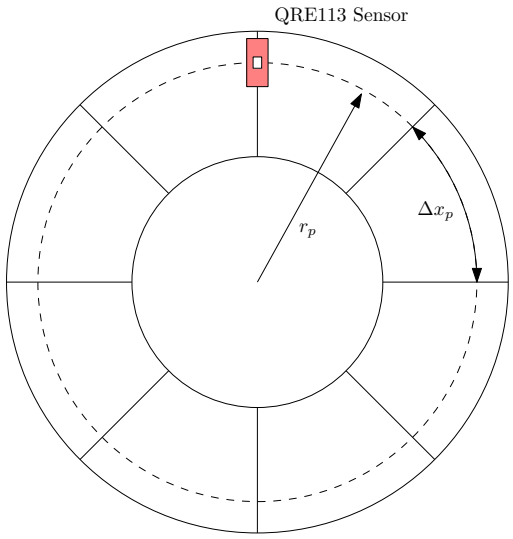


Figure 3.14: Velocity is computed by counting the number of switches between light and dark partition over a fixed time interval, and then using geometric information of the encoder unit.

We estimate the velocity every $\Delta t = 50$ ms, using the following approximation:

$$v \approx \frac{\Delta s}{\Delta t} \tag{3.1a}$$

$$= \frac{(c_k - c_{k-1})\Delta x_p}{\Delta t} \tag{3.1b}$$

$$= \frac{2(c_k - c_{k-1})\pi r_p}{N\Delta t} \tag{3.1c}$$

where c_k is the total number of counts at time step k, Δx_p is the arch distance of a single partition (measured along the circle in front of the sensing unit), r_p is the radius from the tire hub to the sensor, and N is the number of partitions (in our disk design, eight partitions). Figure 3.14 illustrates the geometric quantities in the velocity approximation.

Alternatively, we can estimate the velocity by measuring the time elapsed by rotating just one partition. This method requires precise timing information of the moment when the sensor detects a transition between a light and dark partition. In terms of implementation, we use interrupt signals and timing functions to compute the amount of time elapsed. The velocity is estimated as follows:

$$v \approx \frac{\Delta s}{\Delta t} \tag{3.2a}$$

$$= \frac{2\pi r_p}{N(t_k - t_{k-1})} \tag{3.2b}$$

where t_k is the time for the k-th transition between a light and dark partition.

Between equations (3.1) and (3.2), the latter provides a better method of estimating velocity. Equation (3.1) only provides integral counts for the number of partitions rotated, so the estimate will always return a lower bound for the velocity. In practice, however, equation (3.1) works better at high speeds and equation (3.2) works better at low speeds.

Ultrasound

Sonar sensors provide measurements of distances between the sensor and the nearest object along the line of sight using high frequency sound waves. These sensors measure the time of flight for a sound wave that has been transmitted from and reflected back to the sensor from nearby objects. The sensor then maps the flight time to a distance measurement.

(a) LV-MaxSonar-EZ1 (b) Sonar mount

Figure 3.15: The Ultrasonic Range Finder provides proximity measurements from zero to six meters.

We use the Ultrasonic Range Finder-LV-MaxSonar-EZ1 from MaxBotix to detect nearby objects. This sensor provides limited environmental information, since it only emits a single, straight sound beam. Nonetheless, for the low cost, this sonar sensor is useful for local object detection.

Global Positioning

Global position systems provide information about location on earth in terms of longitude, latitude, and altitude coordinates. GPS sensors listen for signals received from satellites orbiting the earth, and then use timing information from at least three satellites to estimate its position using trilateration. A differential GPS system builds on the conventional GPS system by adding a ground-based, fixed base station with a known position. These stations then broadcast corrections to local GPS sensors to update their position estimates.

We use the Reach RS+ differential GPS system from Emlid for position estimates. The product is still in the early stages of development at the time of writing as the company is a start up, but the product provides ±2 cm accuracy for a price point (≈ 700 USD) significantly lower than other high-end systems (> 10000 USD).

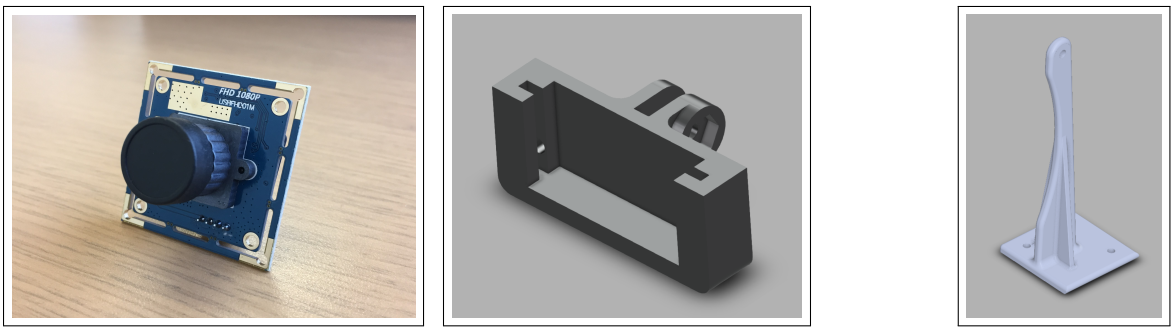
Figure 3.16: The Marvelmind GPS kit is designed to provide ±2 cm accuracy for indoor navigation.

For indoor localization, we use the Indoor Navigation System kit from Marvelmind. The kit consists of four beacons and one rover unit that use sonar measurements to triangulate position. The rover unit is mounted on the mobile robot and measures the timing of sound waves from beacons to triangulate its position. The kit provides centimeter level accuracy and works well when the beacons are fully charged and mounted in a room with solid walls.

Camera

Cameras provide a rich source of environmental information. With the growth of techniques like deep learning for object detection and image segmentation, cameras have become an increasingly important sensor to include on any robotic platform. Cameras for robotic applications commonly come in one of three forms: monocular, stereo, and 3D-RGBD. Monocular cameras come equipped with one image sensor and one set of lenses, while stereo cameras come with two sensors and two sets of lenses. Multiple sensors enable image-processing algorithms to estimate depth for the objects inside the image based on displacements. RGBD cameras are a recent technology and come with a color (RGB) camera and a dedicated depth (D) sensor unit. Devices like the Microsoft Kinetic initially used an infrared projector and infrared sensor to get depth information.

Among the three types of cameras, monocular cameras sell for the lowest price by a wide margin. For our BARC platform, we use a low-cost (≈50 USD) 2 Mega-Pixel 1920x1080P Monocular camera from ELP. The camera has 180 degree wide-angle lens with a USB 2.0 port to transmit image data. The camera can be configured to deliver 280 x 720 pixel images at 60 frames per second (fps). For most of the image-based control applications, however, we use 480 x 640 P at 30 fps. We designed a base fixture and camera-head mount to position the camera at the front of the chassis, looking forward to the front of the vehicle. Figures



(a) ELP 2MP Camera (b) Head mount (c) Base fixture

Figure 3.17: The ELP 2MP Monocular Camera can deliver 280 x 720 P images at 60 fps

3.17b and 3.17c show the camera head mount and base fixture, respectively.

LiDAR

Light detection and ranging (LiDAR) sensors have also become popular in robotics application for generating high-resolution maps. LiDAR sensors measure distance information by emitting pulsed laser light at objects in front of the sensor and then timing the return reflected pulses.

Figure 3.18: The RP LiDAR A2 scans 360 deg with a distance range of 12 meters.

Aside from differential GPS kits, LiDAR sensors cost significantly more than other sensors and computing hardware. We have not extensively used LiDAR for research projects on the BARC platform, but we have found the RP LiDAR by Slamtec to work well for object detection. At the time of writing, the RP LiDAR sells at a price (≈500 USD) a fraction of the cost of high-end LiDAR sensors from Velodyne (4000 USD+). Given recent trends in the market, however, these high-end sensors will likely become less expensive.

Networking components

Wireless routers provide access to Internet and establish private networks. We use a wireless router to remotely access onboard computing hardware and launch experiments on the BARC platform.

(a) ZyXEL Wireless Router (b) Edimax EW-7811Un Wi-Fi USB

Figure 3.19: The ZyXEL router and Edimax Wi-Fi USB devices can both transmit data up to 150 Mbps.

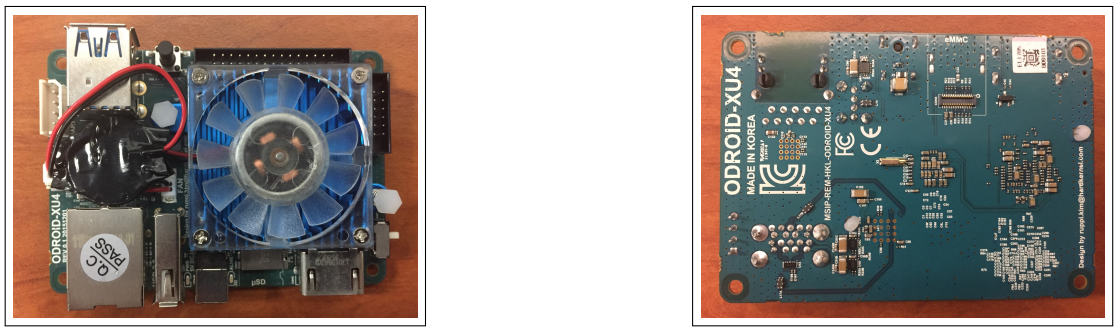
During the initial stages, we used the ZyXEL MWR102 because of the low cost, but later started using the Edimax EW-7811Un Wi-Fi USB instead. The Edimax Wi-Fi USB uses the RTL8188CUS chipset, which is compatible with an open-source utility called hostapd that creates wireless access points. In essence, this software enables the Edimax dongle to behave as a wireless router. This software-enabled capability is useful because the Edimax Wi-Fi dongle takes smaller form factor and costs less than the ZyXEL router.

Computing Device

The BARC platform uses the Odroid XU-4 from HardKernel as the primary computing device. The Odroid XU-4 is a single-board, credit-card sized computer that has nearly all the capabilities of a laptop, albeit with less processing resources and a small form factor. The Odroid XU-4 runs on a Samsung Exynos 5422 Cortex-A15 GHz and Cortex-A7 Octa core CPU based on the ARM architecture. Among the various architectures for embedded devices (e.g. AMD, ARM, ColdFire, TriCore, PowerPC, x86), ARM has become prevalent in small computing devices like Odroid, Freedom Board, Raspberry Pi, Arduino, TX2 as well as in mobile devices.

The Odroid uses an eMMC (electronic Multi-Media Controller) chip as the primary storage for the operating system, applications and user data. eMMC chips are a type of solid state storage technology prevalent in smart phone devices. The Odroid XU-4 is also equipped with a slot to use microSD cards as the storage medium, but eMMC read / write speeds are faster.

The Odroid XU-4 comes equipped with a set of common peripherals, like 1 USB 2.0 port,



(a) Odroid XU-4 front

(b) Odroid XU-4 back

Figure 3.20: The Odroid XU-4 runs on an 8-core ARM Cortex processor.

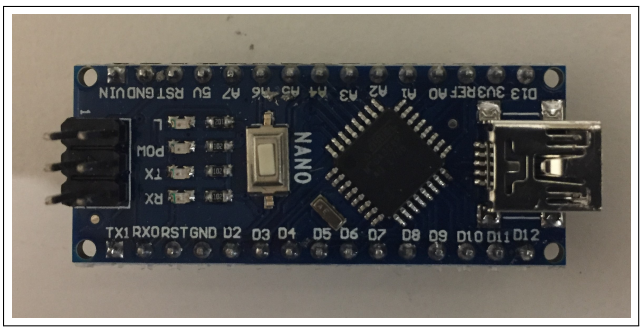
2 USB 3.0 ports (stacked vertically), an Ethernet port, an HDMI connector, a 5V / 4A DC power connector and a fan. There is also arrays of GPIO pins around two edges of the XU-4, but they are not used in our system configuration.

It is important to note that the Odroid XU-4 should be supplied with a steady voltage of 5 V and current of up to 4 A, especially with the newer models, as suggested by HardKernel. We use a voltage regulator to provide a stable 5 V output, with up to 5 A of current. In the past, we discovered that voltage regulators that output 3 A or less, especially with the new Odroid XU-4 models, did not supply sufficient current for the Odroid to boot. As a result, the Odroid would fail to load the operating system, and the system would shut down and attempt to reboot, sometimes corrupting data in the process.

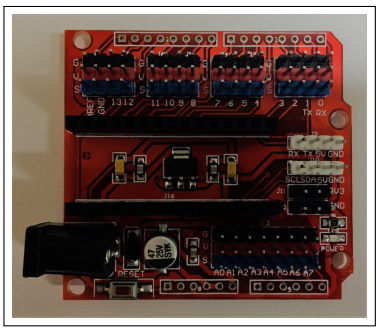
Depending on the model, the XU-4 comes with an electric fan mounted above the CPU to provide temperature regulation when the chip gets warm. Some models use passive heat pipes instead.

The compute power of the XU-4 is not directly comparable with a desktop or PC running on Intel chips, but it still serves as a powerful computing platform for a relatively low cost (≈75 USD). Compared to a Raspberry Pi 3, the Odroid offers better performance in terms of integer and floating point calculation and memory bandwidth according to the Unixbench benchmark scores [13]. Among other candidate computing devices, like the Raspberry Pi, Nvidia TX2, BeagleBond, Huawei HiKey 960, we chose the Odroid XU-4 because it offered strong computing performance for a low price. The Nvidia TX2 offers a lot of promise for research in embedded controls performing intensive imaging processing or running deep neural nets, which exploit the structure of the net and parallelize operations by using the GPU architecture. For the BARC platform, however, we focus on optimal model-based control techniques, like model predictive control, which do not benefit as much from the GPU architecture. Future direction for research in the lab may incorporate more vision-based algorithms, in which case, adopting a platform like the TX2 would be appropriate.

In addition to the Odroid, we also use an Arduino Nano to perform some computations, but mostly perform communication with the actuators and sensors through the general purpose input/out (GPIO) pins. The Arduino Nano, shown in Figure 3.21a, features the



(a) Arduino Nano



(b) Arduino Nano Expansion Shield Module

Figure 3.21: The Arduino Nano features the ATmega328 microcontroller with an AVR architecture

ATmega328 microcontroller with an AVR architecture and has 22 digital IO pins and 8 analog IO pins. The Arduino has very low power consumption (≈19 mA) and takes a small form factor (18 x 45 mm). The Odroid XU-4 also has a slot of GPIO pins, but we chose to use the Arduino because we can use off-the-shelf, low-cost expansion boards to easily connect to sensor with Futaba wires. Furthermore, the Arduino has extensive documentation and examples that made software development much quicker. Figure 3.21b shows the expansion board; the Arduino mounts in the center of the board. The Arduino has a mini-USB port which is used to communicate serially with the Odroid XU-4. At a high level, the Odroid XU-4 runs all the high level state estimation and control algorithms, then sends the command signals to the Arduino. The Arduino, in turn, converts those commands into low level PWM signals to send to the actuators. Sensors signals received by the Arduino are processed and then transmitted to the Odroid.

Software Component

The Odroid XU-4 operates as a full-fledged embedded computer, running Ubuntu Mate 16.04 for ARM as the operating system (OS). As such, users need knowledge of basic Linux commands to navigate file systems, setup configuration files and write software. We have prepared image files available for download at the home website (http://www.barc-project.com/) that already has several useful utilities and open-source software installed. We also have a GitHub repository with the latest software for the BARC platform at https://github.com/MPC-Berkeley/barc. The repository includes a folder with files that set up environment variables and install additional useful utilities.

The BARC platform uses ROS, an open-source library and set of tools for programming robots. The architecture abstracts low level details, like timing, communication protocols, concurrency, and other processes by providing a simple programming paradigm with a well-written, easy-to-use set of application programming interfaces (API). The ROS infrastructure is build on the concepts of nodes, topics and messages. In short, a node is a computer program

(i.e. file) that processes some data. That data is then packaged into a message format, and then published, or broadcast, onto a topic, for other nodes to listen to. In other words, ROS uses a many-to-many communication paradigm to decouple nodes, so that each node (or process) can be written independently of the others. For an in-depth explanation, refer to the tutorials and documentation at http://www.ros.org/.

In the field of robotics, researchers have begun to invoke cloud technologies to process large amounts of information for artificial intelligence and machine learning applications, a trend called "cloud robotics". In light of these efforts to make data more accessible and amenable to heavy processing, we incorporated cloud-service functionality to the BARC platform, through which users can conduct experiments and then push all the data collected to the cloud. The service is built from a local Django server running on the Odroid, and a corresponding remote, master Django server running on an Amazon E2 server. The software is based off an open-source project called Dator [47]. The software base has API functions that we integrate into ROS so that after conducting an experiment, all signals are collected, processed, stored locally and then uploaded online.

3.4 Power System Architecture

At a high level, the power system architecture for the BARC consists of the components: the battery, the wires, and the electrical connectors.

Battery selection

The source of energy for the RC comes from the battery. The battery is an electro-chemical cell that has an electric potential (i.e. voltage) that comes from a difference in potential between the positive and negative electrodes. A battery consists of five major components: electrodes - anode and cathode, separators, terminals, electrolyte, and a case. When an electric load, like the motor, is connected to the battery, then current flows through the circuit and generates a torque through the motor [6]. For RC vehicles, Nickel-Metal-Hybrid (NiMH) and Lithium-ion batteries are common types. The chemistry and physical assembly of each battery differ, but batteries are assessed in terms of storage capacity, discharge rate, cost, weight, physical volume and safety.

Most RC batteries also come with a C-rate specification. The C-rate refers to the rate at which a battery can charge or discharge all of its energy [46]. The normalized unit C-rate is 1C, which indicates the rate at which a battery is fully charged or discharged in 1 hour. A C-rate of 2C would indicate a charging (discharging) rate twice as fast, in 30 minutes. Likewise, a C-rate of 0.5 C would indicate a charge (discharge) rate twice as slow, i.e. a charge (discharge) period of two hours.

For the BARC platform, we use Lithium-Ion (LiPo) batteries because they can store the same amount or more of electrical energy in a smaller form factor than NiMH battery. Figure 3.22 shows the LiPo battery model we use for our platform. This battery supplies

Figure 3.22: Traxxas 7.4V 10000 mA LiPo battery.

power to both the vehicle actuators and the onboard computing hardware and sensors. We chose a battery model with a large energy storage capacity. At full charge, the battery has approximately (7.4 V x 10 A x 1 hr $\approx$ 266 kJ) of energy. By rough calculation, that amount of energy can keep the Odroid XU-4, and all connected peripherals (e.g. Arduino, sensors, wifi-dongle), powered on for roughly (266 kJ / (5V x 4A) $\approx$) 3.5 hours if it operates at the maximum current rating. Of course, the vehicle actuators can consume a lot of energy depending on speed and amount of steering, but in practice, we find the battery to power the BARC platform can last for two hours, usually more.

Electrical Connectors

Electrical connectors for hobby vehicles (e.g. RC cars, RC planes and RC boats) range in form factor, size, and electrical specifications. Typical considerations in port selection are durability, current flow, ease of use (plug/unplug) and installation, contact pressure and cost. For applications in which we need the motor to provide high torque to get to high speeds, the electrical power system, wires and connectors, need to be able to handle high current flow (high amperage) without burning up, while providing a firm contact pressure so that the power is delivered from the battery to motor during high speed maneuvers, under which the vehicle may experience high frequency vibrations. In this section, we briefly discuss the various types of electrical connectors commonly used in hobby vehicles.

High-current connectors

Dean connectors (also known as T connectors) are two port, low-resistance, polarized connects used heavily for RC applications. The name comes from the company that manufactures them, W. S. Deans Corporation. The connectors are spring loaded to ensure a firm connect when two pairs of Dean connectors are connected. The Dean connectors are rated to handle 30 Amps of continuous load and provide firm contact pressure [37].

The XT60 connector is another type of high-current, low resistance connector. The connector comes from a Hong-Kong based company. The connectors are made from nylon

Figure 3.23: Dean

with gold-plated spring pins to establish high surface area contact with the wires. The XT60 connectors, as the name suggests, are rated up to 60 Amps of continuous current draw [38]. The XT30 is another electrical connector from the same family, but of a smaller form factor, and can handle up to 30 Amps continuous current draw.

Figure 3.24: XT60

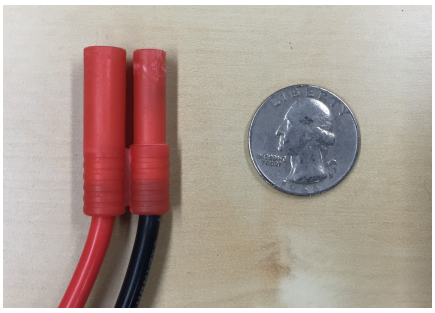
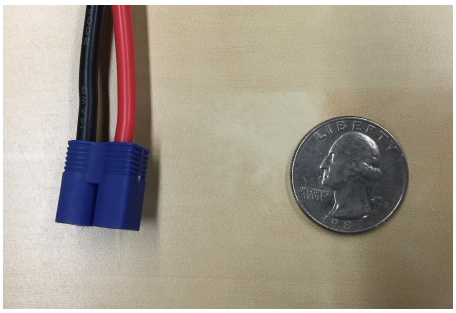


Figure 3.25: Bullet

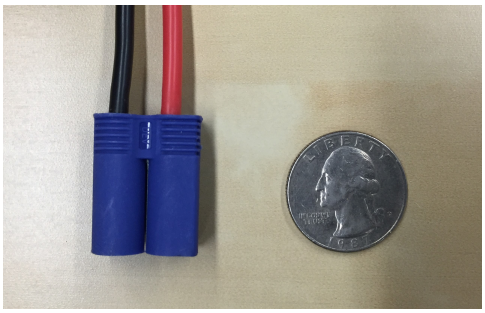
The bullet connector is one of the most prevalent electrical connectors in the domain of RC products. The connectors are coated in 24K gold and provide low-resistance, large contact area and a solder window to allow for easy installation. RC bullet connectors can handle up to 80 Amps of continuous current and 120 Amps of short time-scale peak current [36].

The EC family of connectors are similar in design and mechanism to the bullet connectors, but vary in the size and current capacity. These connectors are more common among aircraft RC vehicles than in vehicle RC vehicles. The EC2, EC3, EC5 connectors can handle up to 20 Amps, 60 Amps and 120 Amps, respectively, of continuous current [17] [18] [19]. The required wire gauge differs among the EC connectors. EC5 can handle significantly higher current flow than EC2, but requires a larger gauge wire.

The Traxxas connector is a proprietary connector made by Traxxas, an RC manufacturing company based in Texas, that is primarily for Traxxas products. To the best of the author's knowledge, no public information about the current flow rating for the connect have been provided in a specification sheet, but it is inferred that connector can handle up to 30 Amps, since the limiting factor as marketed is the current limit of the wire used [41]. The Traxxas



(a) EC3



(b) EC5

Figure 3.26: EC family of connectors.

connector comes with a large black plastic housing and unique mating configuration such that it is not possible to accidentally connect the poles backwards (i.e. reverse polarity protection). The texture of the housing make the connector easy to grip and plug/unplug two mating connectors. The Traxxas connector housing is larger than other options, so it is best to use only when space is not a tight constraint.

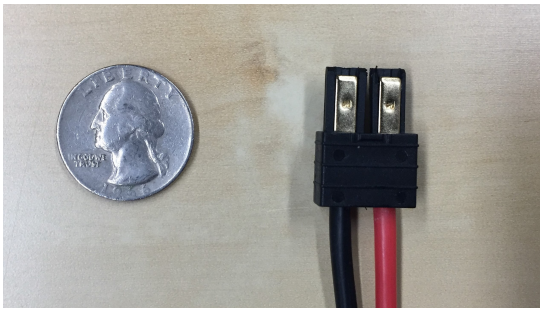


Figure 3.27: Traxxas

Figure 3.28: Anderson power pole

The Anderson power pole connector is a proprietary connector by Anderson Powerpole Product (APP) and has become a common connector for high power DC applications. As with other connector families, the power pole connectors have a variety of housing options and current rating specifications that are listed on the vendor website, but 15/30/45 current ratings are common. The housing color also indicates the current rating. The Anderson connectors differ from other connectors in that the positive and negative terminal are not fixed to each other as with other connectors; a slot mechanism between the two ports allows the positive and negative ports to readily attach and detach. This attaching/detaching mechanism is particularly helpful when size and geometry constraints for the power system are strict (i.e. limited space and wire routing options for power transmission).

Along with each connector, manufacturers also produce adapters to change from one type of connection to another. Figure 3.29 illustrate an adapter converting from a Traxxas to a

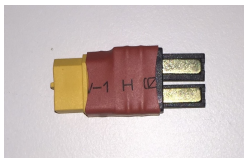


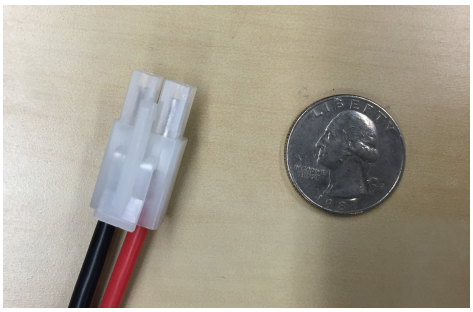
Figure 3.29: Traxxas-XT60 Adapter

Figure 3.30: XT60 splice

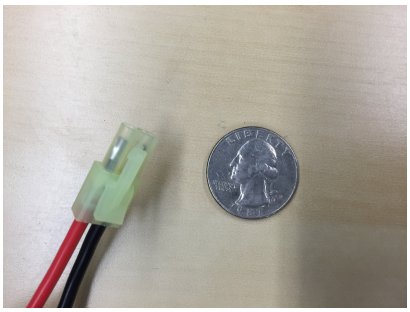
XT60 connector. Splicers, as the one shown in Figure 3.30, split power flow from one source to multiple destinations. For BARC, we use both adapter and splicers to distribute power to the motor and servo, as well as the onboard computing hardware and sensors. We discuss this more in detail shortly.

Low-current connectors

Low-current connectors distribute power and channel information between low-power computing devices and sensors. These sensors are found across the spectrum in small-consumer electronic devices.



(a) Tamiya



(b) Mini-Tamiya

Figure 3.31: Tamiya connectors.

The Tamiya connectors belong to a class of low-current electronic connectors that are prevalent in RC model vehicles. The sockets for the Tamiya connectors make them firm and stable when connected. It is a particularly common connection for battery packs. The connector are designed for various current ratings depending on wire gauge and wire length, but ratings from 5 A up to 15 A are available [39].

The JST is a common low-current connector from the Japanese company JST (Japan Solderless Terminal) used in many electronic hobby products (e.g. batteries, RC vehicles, and on some PCBs). Mechanically, the JST connector housing has mating male and female connectors that slide until a latch holds the parts firmly together. The JST housing is plastic, which also limits the current rating, since high current would cause the electrical connector to become very hot, potentially melting the surround plastic. The JST connectors

Figure 3.32: JST

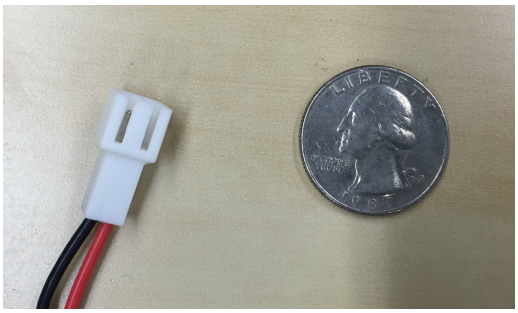


Figure 3.33: XHP-2

are typically rated for 5 Amps or less. In spite of the limited current rating, JST connectors are still prevalent in the RC community.

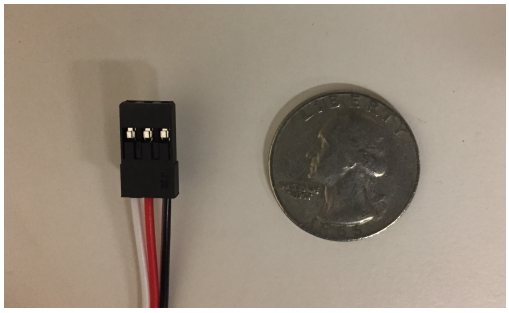


Figure 3.34: The Futaba cable is prevalent among low-powered actuators and sensors.

For our platform, we use Futaba connectors. Futaba connectors have become a de facto standard for servos. Many RC vendor use Futaba connectors for transmitting command signals between the ESC and the actuators. Because of their ubiquity in electronics, many suppliers have designed 'extension boards' for micro-controller like the Arduino that include Futaba cable interfaces. These boards enable users to quickly connect with sensors and low-powered actuators without the need to solder. For BARC, we use an Arduino Nano extension board to connect with encoders, RX-TX box, and command signal lines for the servo and motor using the Futaba cable interface.

Power distribution

All electric hardware and mechanical actuators on the chassis receive power from the LiPo battery. Unlike other platforms that use a separate power bank to deliver power to the electric hardware, we choose instead to use one battery and distribute power to components through splicers, adapters, and voltage regulators. Figure 3.35 illustrate the power flow for all system components.

The voltage regulator provides a stable output voltage independent of the input load and current, as long as the input voltage is above the specified threshold. We use a Drok

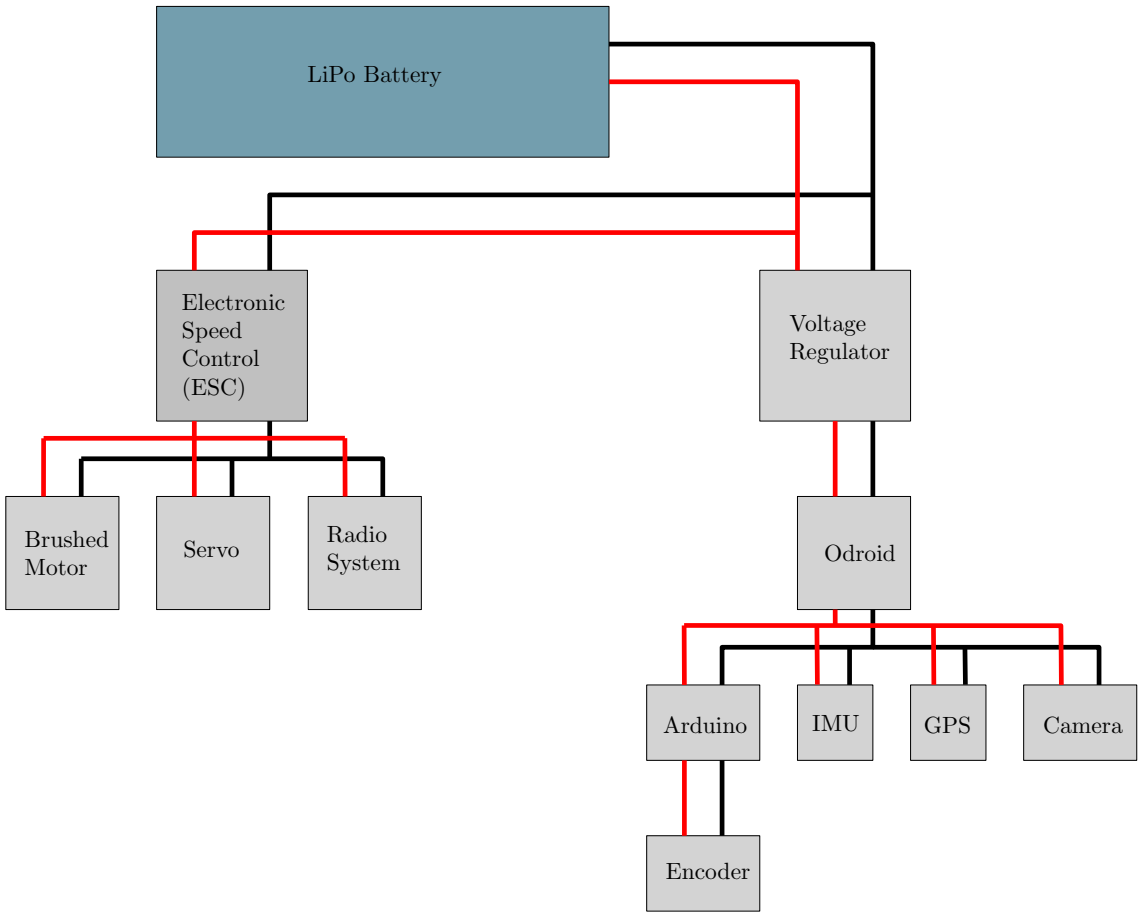
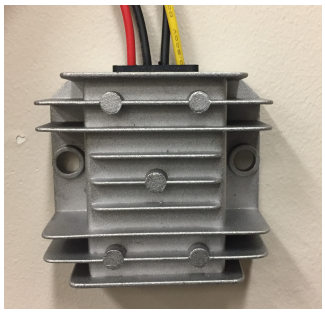


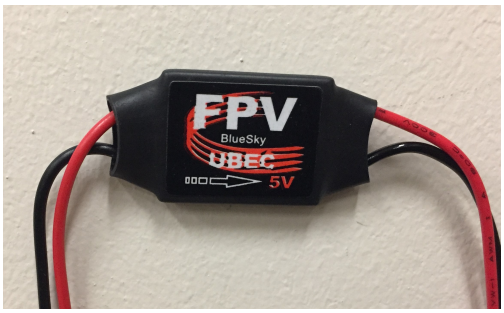
Figure 3.35: The Power distribution for BARC platform uses a single battery as the energy source.

3005ADJ DC/DC voltage regulator to step down the battery voltage from 7.4 V to 5 V, and output up to 5 A of current to power the Odroid. Previously, we had used the FPV voltage regulator, which also output 5 V, but only up to 3 A, which we have discovered, sometimes, was not enough to power the Odroid XU-4 during boot up.

The BARC platform uses a single large battery as the only source of energy, which reduces mass, volume, and cost by eliminating the need for separate energy supplies. We use off-the-shelf manufactured adapters and splicers to achieve this configuration. The LiPo battery and the ESC both come with the Traxxas style connector. Since Traxxas connectors are only used with Traxxas products, we use adapters to convert to XT60 connectors. From the battery, we can then use an XT60 splicer to distribute power to both the ESC and the voltage regulator. The voltage regulator steps down the voltage to power the Odroid, which in turn, powers all the peripherals. Another motivation for this power configuration stems from ease of assembly. The Drok terminals require soldering to have the right connector, but aside from that, all other cabling is easily connected by pushing the connectors together.



(a) Drok voltage regulator



(b) FPV voltage regulator

Figure 3.36: Voltage regulators output a stable direct current voltage independent of the input voltage and current loads.

3.5 Teaching Applications

Since the project inception, the BARC platform has developed into a stable, robotic platform suitable for research and instruction. At UC Berkeley, we have integrated the platform into the curriculum of a ME 131 - Vehicle Dynamics and Control, by complementing academic exercises in control design with hands-on implementation labs using BARC.

In spring 2016, we introduced the first generation platform into the curriculum by offering students that opportunity to use BARC as a part of their final project. Many student teams wrote and submitted final projects on tire-slip control and vision-based lane keeping. On the BARC website, we uploaded videos of students who wanted to showcase their final project.

In spring 2018, we developed a series of lab exercises to make BARC a central component of the curriculum. The goal was to bridge to gap between theory and experiment, as well as teach the students a set of skills useful for robotics. According to feedback from students, while the BARC platform represented a difficult challenge due to lack of experience with robotics, many students found the course rewarding and beneficial to their engineering education.

The sets of lab exercises for ME 131 spanned the following topics:

- Assembly for BARC platform
- Introduction to Linux and ROS
- Using BARC with ROS for actuator control
- System Identification for steering and longitudinal dynamics
- Cruise control for longitudinal dynamics using PID
- Path following using image-based control

- Drift parking

Moving forward, the curriculum may change to include more content on control applications.

3.6 Conclusion

This chapter gave an introduction to the Berkeley Autonomous Race Car platform and an overview of the design decisions for the mechanical, electrical and software components. We developed BARC as a low-cost, open-source platform for research and instruction, and have provided documentation and other resources on our BARC webpage and the Git repository to allow others to easily replicate the platform at their home institutions.

Chapter 4

Planning and Control of Drift Maneuvers

In this chapter, we discuss planning and control algorithms for drift maneuvers. The first part of this chapter focuses on control design for autonomous steady state drifting. The results from the equilibrium analysis in Chapter Two shape the design of the control policy. The second part of this chapter transitions to the planning and control of two transient drift maneuvers: drift parking and drift cornering. The focus is on the ideas of sample-based path planning and mixed open-loop, closed-loop control as a complete framework to perform autonomous drift maneuvers. The ideas presented to control each one of these maneuvers are verified from experimental results using the BARC platform.

4.1 Autonomous Steady State Drifting

This section presents a controller that achieves autonomous steady state drift using *only* onboard sensors. The controller is based on an infinite-horizon LQR formulation using a single-track rigid body bicycle model with a Pacejka tire model. The state estimation is based on an extended Kalman filter that uses only IMU, encoders and camera. Related research projects have achieved autonomous drift in passenger or RC cars using motion capture systems or a differential GPS, which provide very accurate measurements of position, but require infrastructure that constrains the location of the experiment. One contribution of this thesis is to demonstrate how to design a conventional LQR controller using only onboard sensors to achieve steady state drift, which frees the experiment location. The proposed control design is validated on the BARC platform.

Background

Drift, or high side slip cornering, occurs when a skilled driver intentionally maneuvers a vehicle to cause loss of traction in the wheels, but still maintains control of the vehicle.

As discussed in Chapter 2, drift causes a large side-slip angle β to emerge, which roughly speaking, indicates to what extent the vehicle is traveling 'side-ways'.

For sustained steady state drift, researchers have validated various control techniques. Hindiyeh and coauthors developed a controller with a successive loop structure using a bicycle model and a tire brush model [14]. Velenis and coauthors developed a sliding model controller to maintain drift using a seven-state vehicle model with rear differentials [45]. Cutler used reinforcement learning with a motion caption system to achieve sustained drift [5]. All experimental validation for drift control algorithms have used a motion capture systems and/or a differential GPS system [25] [15], which provide very accurate measurements of position, but require infrastructure that constrains the location of the experiment.

Vehicle Model

The vehicle dynamics are captured using a single-track rigid body model with lumped rear wheel and front wheels, discussed in Chapter 2. The state vector is $z = \begin{bmatrix}\beta & v_x & \dot{\psi}\end{bmatrix}^\top$, where β is the slip angle, v_x is the longitudinal velocity, and $r = \dot{\psi}$ is the yaw rate. The input vector is $u = \begin{bmatrix}\delta & F_x^r\end{bmatrix}^\top$, where δ is the steering angle and F_x^r is the rear force, assuming a rear wheel drive system.

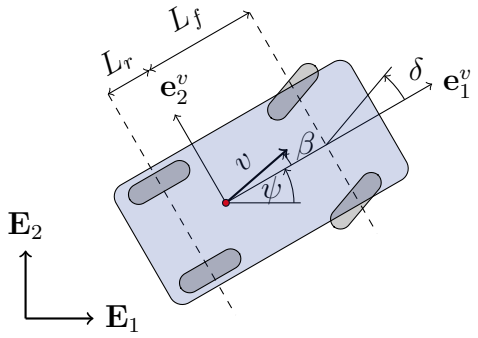


Figure 4.1: Vehicle model schematic

The equations of motion for the vehicle dynamics are obtained from the balance of linear and angular momentum. The resulting set of differential equations in state space form are given below:

$$\dot{\beta} = \frac{1}{mv_x}(F_y^f + F_y^r) - r \tag{4.1a}$$

$$\dot{r} = \frac{1}{I_z}(L_f F_y^f - L_r F_y^f) \tag{4.1b}$$

$$\dot{v}_x = \frac{1}{m}(F_x^r - F_y^f \sin\delta) + v_x r\beta \tag{4.1c}$$

where F_y^r is the lateral force at the lumped rear tire. The parameters m and I_z are the mass and moment of inertia about the z-axis, respectively, and L_f, L_r are lengths from the CoG to the axles, as shown in Figure (4.1). Recall that the dynamic equations in (4.1) formed the basis of the stability analysis in Chapter 2. We use the equilibrium state as the reference state for the control design. For the tires, we use a Pacejka model to estimate the lateral force at high tire slip angles:

$$F_y(\alpha) = \begin{cases} \mu F_z \sin(C \tan^{-1}(B\alpha)) & : |\alpha| \leq \alpha_{cr} \\ F_y^{\max} \text{sign}(\alpha) & : |\alpha| > \alpha_{cr} \end{cases} \tag{4.2}$$

where the front and rear tire side slip angles are:

$$\alpha_F = \tan^{-1}\left(\beta + \frac{rL_f}{v_x}\right) - \delta \tag{4.3}$$

$$\alpha_R = \tan^{-1}\left(\beta - \frac{rL_r}{v_x}\right) \tag{4.4}$$

and $F_y^{\max}$ is the maximum lateral force as modeled by the 'friction circle', defined as follows:

$$F_y^{\max} = \sqrt{(\mu F_z)^2 - F_x^2} \tag{4.5}$$

where μ is the coefficient of friction, F_z is the load force, B and C are fitting parameters, and α_{cr} is the critical slip angle, beyond which the tires cannot generate additional force:

$$\alpha_{cr} = \tan\left(\sin^{-1}\left(\frac{\sqrt{F_z^2 - F_x^2}}{\mu F_z}\right)/C\right)/B. \tag{4.6}$$

We remark that this vehicle model serves as the basis for the control design and the extended Kalman filter for state estimation. One difference between the model in the controller and estimator, however, lies in the equation for longitudinal dynamics. The estimator incorporates more forces acting on the vehicle (like a nonlinear drag term) to provide a more accurate estimate of v_x, while the controller neglects those effects, simplifying the control design process.

Equilibrium Analysis

The details of the procedure for equilibrium analysis are summarized at the end of Chapter 2. We briefly remark here that the equilibrium state and input, z^{eq} and u^{eq}, respectively, come from the solutions to the system of nonlinear equations in $\dot{z} = f(z, u) = 0$.

The equilibrium states of the three-state model are shown in Figures (4.2) to (4.4). The plots illustrate two kinds of equilibrium: normal cornering (circle) and drifting (asterisks).

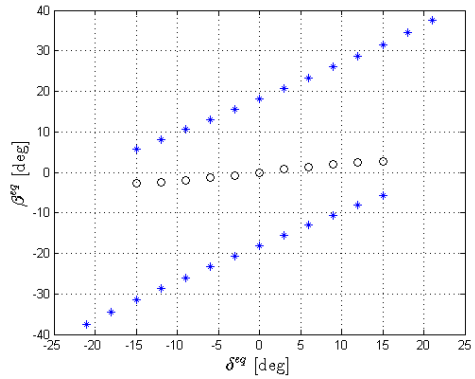


Figure 4.2: Equilibrium sideslip angle vs steering angle

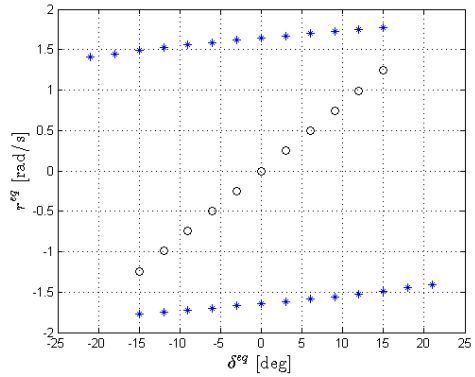


Figure 4.3: Equilibrium yaw rate vs steering angle.

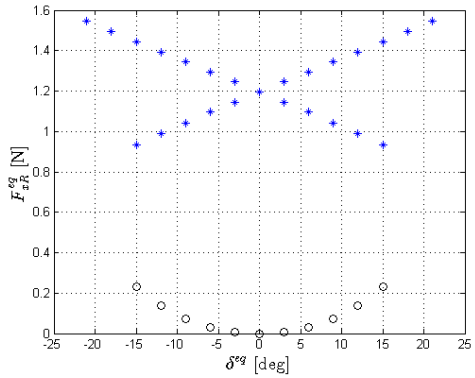


Figure 4.4: Equilibrium rear wheel force vs steering angle.

For normal cornering, the sideslip angle is small, and the yaw rate and steering angle hold the same sign. For drifting, the side slip angle and yaw rate are large, and the rear tire forces are operating at the friction limits. Figures 4.2 through 4.4 illustrate the solutions to the

equilibrium analysis.

The control design is based on a single equilibrium point that is selected as the reference state for the controller. Any equilibrium point under drift conditions (asterisks) can be selected as the reference state.

Model Identification and State Estimation

We now discuss the procedure to identify the values of system parameters, in particular those from the Pacejka tire model, for the BARC platform and then briefly discuss the formulation of the state observer.

Longitudinal dynamics

The longitudinal dynamics primarily serve to provide information on the longitudinal velocity of the vehicle v_x. The equilibrium analysis discussed in the previous section assumes a constant longitudinal velocity for the steady state drift maneuver, so it is important to build a model of the longitudinal dynamics that can be incorporated into a state estimation scheme. We use a longitudinal dynamics model that accounts for the effects of surface friction, drag force, and the motor input force. Based on equation (4.1), we define the following expression for the longitudinal rear force F_x^r:

$$F_x^r = F_{\text{motor}} + F_f + F_{\text{drag}} \tag{4.7}$$

which consists of the input motor force F_{motor}, the friction force F_f, and the drag force $F_{\text{drag}} = C_D v_x^2$, where C_D is the aerodynamic drag coefficient. In other words, the total longitudinal rear input force F_x^r that acts on the vehicle is not just the motor input force coming from the actuator, but the net effect of forces from the motor force and friction forces. We add these terms in our observer to enhance model fidelity, and hence compute a more accurate estimate of the state. In the section on controller design, however, we intentionally ignore the air drag force (i.e. assume $F_x^r = F_{\text{motor}}$) so that we can still apply an LQR controller by exploiting the linear structure of the system.

In order to solve for the parameters F_f and C_D, we perform a nonlinear least squares optimization using the longitudinal model in (4.8), formulated as:

$$\begin{aligned} \min_{b,F_f,C_D} \quad & \left|\left| \dot{v}_x - \frac{1}{m}(F_{\text{motor}} + F_f + C_D v_x^2) \right|\right|_2^2 \\ \text{s.t.} \quad & F_{\text{motor}} = b u_{\text{motor}} \\ & b, F_f, C_D \geq 0 \end{aligned} \tag{4.8}$$

where F_f, C_D, and b are optimization variables constrained to be positive. The objective function aims to find the best friction coefficient values and input gain that make the longitudinal dynamics expression $F_x = ma_x$ hold. To generate experimental data, we run short step-input tests with a fixed steering angle, $\delta = 0$ and fixed motor input u_{motor} from the

BARC. Multiple tests are conducted with various step-input final values to get a representative sample of longitudinal velocity data from the RC. The servo is calibrated so that $\delta = 0$ results in straight motion of the RC. From the system equations, straight motion only engages longitudinal dynamics, so the optimization program can compute the best parameters to make the expression $F_x - ma_x = F_x - m\dot{v}_x = F_x^r - m\dot{v}_x = 0$ hold.

The data for the optimization routine comes primarily from the encoder. The IMU did provide longitudinal acceleration $a_x = \dot{v_x}$ measurements, but since the filtered signal was still noisy, in the end, the longitudinal dynamic equation in the objective function of program (4.8) was discretized so that velocity measurements from the encoders could directly be used. The step-input motor signals were also known. The discretized optimization program is shown below:

$$\begin{aligned} \min_{b,F_f,C_D} \quad & \left|\left| \sum_{k=0}^{T} v_x[k+1] - v_x[k] - \Delta t \left(\frac{1}{m}(F_{\text{motor}}[k] + F_f + C_D v_x[k]^2) \right) \right|\right|_2^2 \\ \text{s.t.} \quad & F_{\text{motor}}[k] = b u_{\text{motor}}[k] \qquad \forall k \in \{0, 1, ..., N\} \\ & b, F_f, C_D \geq 0 \end{aligned} \tag{4.9}$$

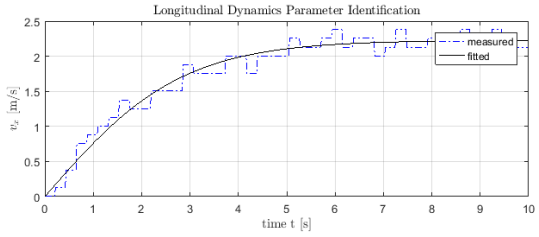


Figure 4.5: The optimized frictional and input gain parameters accurately longitudinal dynamics of the RC.

The optimization routine in 4.8 is conducted for a number of step input tests, and the resulting values of the optimization parameters are averaged. All the signals are expressed as time series data with time index k. Figure 4.5 illustrates the prediction of longitudinal model against experimental data.

Tire model Identification

The Pacejka model is used to describe the lateral forces that are produced from the tires. The model parameters are identified using a nonlinear least squares routine that attempt to equate both sides of the side-slip angle and yaw rate dynamic equations. The routine below

capture the full optimization program:

$$\min_{\{\beta^{(k)}\}_{k=0}^{N},B,C,\mu} \sum_{i=0}^{N} \left(\frac{1}{mv_x^{(k)}}(F_y^{f,(k)} + F_y^{r,(k)}) + r^{(k)} \right)^2 + \left(\frac{1}{I_z}(L_f^{(k)} F_y^{f,(k)} - L_r^{(k)} F_y^{r,(k)}) \right)^2 \quad (4.10a)$$

$$\text{s.t.} \qquad \forall k \in \{0, 1, ...N\}$$

$$F_z^{r,(k)} = \frac{mgL_f^{(k)} - v_x^{(k)}\beta^{(k)} r^{(k)} hm}{L} \qquad F_z^{f,(k)} = mg - F_z^{r,(k)} \quad (4.10b)$$

$$L_f^{(k)} = \frac{F_z^{r(k)}}{F_z^{f,(k)} + F_R^{z,(k)}} L \qquad L_r^{(k)} = \frac{F_z^{f,(k)}}{F_z^{f,(k)} + F_z^{r,(k)}} L \quad (4.10c)$$

$$a_f^{(k)} = \tan^{-1}(\beta^{(k)} + \frac{r^{(k)} L_f^{(k)}}{v_x^{(k)}}) - \delta^{(k)} \qquad a_r^{(k)} = \tan^{-1}(\beta^{(k)} - \frac{r^{(k)} L_r}{v_x^{(k)}}) \quad (4.10d)$$

$$F_y^{f,(k)} = -\mu F_z^{f,(k)} \sin(C \tan^{-1}(B\alpha_f^{(k)})) \quad (4.10e)$$

$$F_y^{r,max,(k)} = \sqrt{(F_z^{r,(k)})^2 - (F_x^{r,(k)})^2} \quad (4.10f)$$

$$F_y^{r,\text{paj},(k)} = -\mu F_z^{r,(k)} \sin(C \tan^{-1}(B\alpha_r^{(k)})) \quad (4.10g)$$

$$F_y^{r,(k)} = \min(F_y^{r,\max,(k)}, F_y^{r,\text{paj}}) \quad (4.10h)$$

where k superscript refers to the k-th experiment from N experiments. The optimization program (4.10) finds the optimal values of tire model parameters $B, C, \mu \in \mathbb{R}$ and side-slip angle values $\{\beta^{(k)}\}_{k=0}^{N}$. The optimization routine contains many nonlinear expressions in the constraints that come from the system dynamics and tire models in (4.1) and in (4.4). The objective function (4.10a) aims to equate both sides of the dynamic equations in the system model (4.1) by minimizing the sum of residuals. Expressions (4.10b) - (4.10c) account for the effects of weight transfer between the front and rear tires. These effects of weight transfer are included to get a more accurate estimate of the tire model parameters. The additional complexity to the system dynamics is acceptable since the optimization program is run offline. Expressions in (4.10d) calculate the tire slip angle, and expressions (4.10e) - (4.10h) calculate the lateral force in the front and rear tires.

As in the optimization program (4.9), the input to program (4.10) are vehicle parameters and time series data from sensors and actuators (servo for steering). To generate data, the RC runs with a fixed motor and steering angle command until the RC reaches steady state, at which point the steady state longitudinal velocity v_x, steering angle δ, and yaw rate r are recorded. After several tests from a range of steering angles input δ and motor inputs F_{xR}, optimization program (4.10) is solved. We remark that the optimization program (4.10) contain more expressions than other tire model identification procedures. This stems from the limitation that an accurate estimate of the lateral velocity v_y is difficult to obtain using only the encoder and IMU measurements, both of which are noisy. Since the lateral velocity is necessary for computing the tire slip angles, we estimate it indirectly by making β an optimization variable that best fits that data.

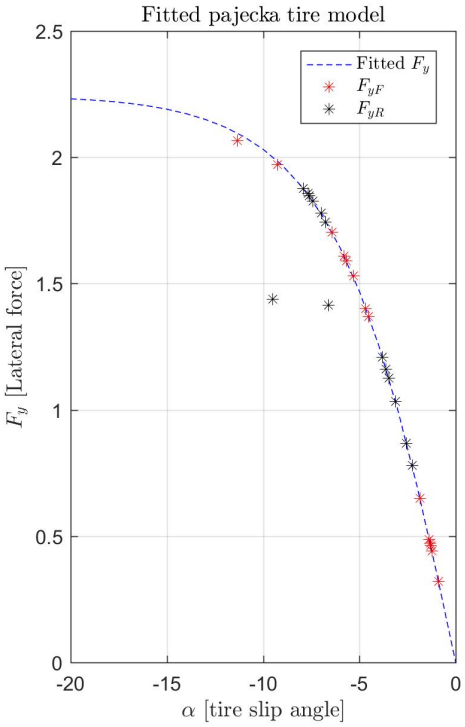


Figure 4.6: The optimized tire model parameters from program (4.10) fit the experimental data. The red and black asterisks represent tire forces estimated directly from data using the dynamic equations, and the dashed blue line comes from the tire model with optimized parameters and slip-angle estimates from data

Optical flow

During drift, the measurement readings for the encoder are no longer reliable, due to slip and we must use other sensor information to estimate velocity. For this project, we use an on-board camera to estimate velocity using a technique called optical flow. Optical flow calculates the motion between two image frames by computing spatial and temporal gradients of the brightness, and then solving the following equation:

$$I_x v_x + I_y v_y + I_t = 0 \tag{4.11}$$

where I_x, I_y are the partial derivatives of the image intensity with respect to position, I_t is the partial derivative of the image intensity with respect to time, v_x is the optical flow along the $x-$direction, and v_y is the optical flow along the $y-$direction. The image intensity is a scalar number between 0 (black) and 255 (white) for images that are converted from color to gray-scale. Equation (4.11) applies to each pixel and each time step, which results in a longitudinal and lateral velocity value computed for each pixel.

We remark that the lateral and longitudinal estimate directly coming for optical flow first needed to be calibrated against a good longitudinal velocity estimate from the encoders. Secondly, the optical flow algorithm computes intensity gradients, meaning the images need to change from frame to frame, so the surround environment should be rich in features. We found that a downward-facing camera installed at the rear of the car gave poor optical flow results. After inspecting the sequence of input images while the RC drove along a straight path along an indoor hallway, we discovered the images changed significantly between any two consecutive frames, to the point where each frame had little to no pixel data preserved from the previous frame (in other words, the image scene was entirely new for each frame), so the gradient computations were inaccurate. Furthermore, the tiled-surface do not provide a feature-rich scene with which the optical flow algorithm could compute gradients. In the end, we mounted the camera directly above the CoG and pointed the camera upward, facing the ceiling. We observed much more stable performance when running the experiment indoors at low-speeds (less than 3 m/s). The estimated longitudinal and lateral velocity of the RC CoG was taken as the average of the velocity estimates over the center group of image pixels.

More details of the optical flow algorithm can be found in [10]. For implementation, we ran the optical flow algorithm using the Open Computer Vision (OpenCV) library.

State Estimation

For state estimation, we applied an Extended Kalman Filter (EKF) that uses the vehicle model in (4.1), with the longitudinal dynamics in (4.7) and the tire model in (4.5). We model process noise w_k and measurement noise v_k at time step k as Gaussian with zero mean and covariance Q_k and R_k , respectively, as shown below:

$$w_k \sim \mathcal{N}(0, W_k) \tag{4.12a}$$
$$v_k \sim \mathcal{N}(0, V_k) \tag{4.12b}$$

and then apply the following prediction and update equations to get a state estimate:

$$\mu_{t+1|0:t} = A\mu_{t|0:t} + B\mu_t \tag{4.13a}$$
$$\Sigma_{t+1|0:t} = A\Sigma_{t|0:t}A^\top + Q \tag{4.13b}$$
$$K_{t+1} = \Sigma_{t+1|0:t}C^\top \left(C\Sigma_{t+1|0:t}C^\top + R\right)^{-1} \tag{4.13c}$$
$$\mu_{t+1|0:t+1} = \mu_{t+1|0:t} + K_{t+1}\left(z_{t+1} - (C\mu_{t+1|0:t} + d)\right) \tag{4.13d}$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1}C)\Sigma_{t+1|0:t} \tag{4.13e}$$

where $\mu_{t+1|0:t}$ is the a priori state estimate, $\mu_{t+1|0:t+1}$ is the a posteriori state estimate, $\Sigma_{t+1|0:t}$ is the a priori estimate covariance, and $\Sigma_{t+1|0:t+1}$ is the a posteriori estimate covariance.

Control Design

The objective of the controller is to stabilize the vehicle around an equilibrium drifting state which we designate as the reference state $\bar{z} = z^{\text{eq}}$. We design an LQR controller offline according to the steps in Table 4.1 and then run it online according to Table 4.2.

Table 4.1: Offline procedure

1 Define the reference state $\bar{z} = z^{\text{eq}}$
2 Linearize vehicle model about reference state
3 Compute LQR control policy $u = \bar{u} + \Delta u$ $\Delta u = -K\Delta z$
4 Compute the input limits $u_{\min}, u_{\max}$ using the friction circle
5 Find the region of attraction where LQR is stabilizing and does not violate input limits (discussed in the next section)

The LQR method provides a state-feedback policy for our MIMO system to balance the conflicting objectives of having a high yaw rate in one direction with a steering angle in the other (i.e. counter-steer). In order to use LQR, the system dynamics (4.1) are linearized with respect to the reference state $\bar{z} = z^{\text{eq}} = \begin{bmatrix}\beta^{\text{eq}} & v_x^{\text{eq}} & r^{\text{eq}}\end{bmatrix}^\top$ and reference input $\bar{u} = u^{\text{eq}} = \begin{bmatrix}F_y^{f,\text{eq}} & F_x^{r,\text{eq}}\end{bmatrix}^\top$. The system dynamics are then transformed to reflect the error dynamics.

$$\Delta\dot{z} = A\Delta z + B\Delta u \tag{4.14}$$

where $\Delta z = z - \bar{z}$ and $\Delta u = u - \bar{u}$

In the linearized system dynamics, the front lateral force and rear longitudinal force act as inputs to the system since they enter the system dynamics in a predominantly linear way, aside from the $\sin\delta$ term in (4.1c), which is treated as a parameter. More importantly, using

Table 4.2: Online procedure

1 Use PI controller to bring vehicle to target velocity in reference state
2 Apply open-loop maneuver to drive vehicle into the LQR region of attraction
3 Initiate LQR controller and map inputs to low level actuator commands

the tire model, the front lateral force can be mapped to a steering angle δ, which is then sent as an actuator command.

The analytical form of the entries for the system matrix A and the input matrix B are given as follows

$$A = \begin{bmatrix} \frac{\partial \dot{\beta}}{\partial \beta} & \frac{\partial \dot{\beta}}{\partial r} & \frac{\partial \dot{\beta}}{\partial v_x} \\ \frac{\partial \dot{r}}{\partial \beta} & \frac{\partial \dot{r}}{\partial r} & \frac{\partial \dot{r}}{\partial v_x} \\ \frac{\partial \dot{v}_x}{\partial \beta} & \frac{\partial \dot{v}_x}{\partial r} & \frac{\partial \dot{v}_x}{\partial v_x} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{4.15a}$$

$$B = \begin{bmatrix} \frac{\partial \dot{\beta}}{\partial F_{yF}} & \frac{\partial \dot{\beta}}{\partial F_{xR}} \\ \frac{\partial \dot{r}}{\partial F_{yF}} & \frac{\partial \dot{r}}{\partial F_{xR}} \\ \frac{\partial \dot{v}_x}{\partial F_{yF}} & \frac{\partial \dot{v}_x}{\partial F_{xR}} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \tag{4.15b}$$

with the following analytical expressions for the non-zero elements of system matrix A:

$$a_{12} = -1 \tag{4.16a}$$

$$a_{13} = -\frac{F_y^f + F_y^r}{m v_x^2} \tag{4.16b}$$

$$\begin{aligned} a_{31} = \frac{\partial \dot{v}_x}{\partial \beta} &= -\frac{1}{m} F_y^f \frac{\partial \sin\delta}{\partial \beta} + v_x r \\ &= -\frac{F_y^f \cos\delta}{m} \frac{\partial \delta}{\partial \alpha_f} \frac{\partial \alpha_f}{\partial \beta} + v_x r \\ &= \left(-\frac{F_y^f \cos\delta}{m} \right) (-1) \left(\frac{1}{1 + (\beta + \frac{r L_f}{v_x})^2} \right) + v_x r \\ &= \frac{F_y^f \cos\delta}{m} \frac{v_x^2}{v_x^2 + (\beta v_x + r L_f)^2} + r v_x \end{aligned} \tag{4.16c}$$

$$a_{31} = \frac{F_y^f \cos\delta}{m} \frac{L_f v_x}{v_x^2 + (\beta v_x + r L_f)^2} + v_x \beta \tag{4.16d}$$

and the following analytical expressions for the non-zero elements of control B:

$$b_{11} = \frac{1}{mv_x} \tag{4.17a}$$

$$\begin{aligned} b_{12} &= -\frac{1}{mv_x}\frac{\partial F_y^r}{\partial F_x^r} \\ &= -\frac{1}{mv_x}\frac{\partial\sqrt{(\mu F_z^r)^2 - (F_x^r)^2}}{\partial F_x^r} \\ &= -\frac{1}{mv_x}\frac{F_{xR}}{\sqrt{(\mu F_z^r)^2 - (F_x^r)^2}} \end{aligned} \tag{4.17b}$$

$$b_{21} = \frac{L_f}{I_z} \tag{4.17c}$$

$$b_{22} = \frac{L_r}{I_z}\frac{F_x^r}{\sqrt{(\mu F_z^r)^2 - (F_x^r)^2}} \tag{4.17d}$$

$$b_{31} = -\frac{\sin\delta}{m} \tag{4.17e}$$

$$b_{32} = \frac{1}{m} \tag{4.17f}$$

Note that for terms containing rear tire force components in the control matrix B, we apply the constraint $(\mu F_z)^2 = F_x^2 + F_y^2$ since the rear tire is saturated during drift.

The LQR controller is based on the system dynamics in (4.14) with the following quadratic cost function:

$$J = \int_{t=0}^{\infty} (\Delta z^T Q \Delta z + \Delta u^T R \Delta u) dt \tag{4.18}$$

where $Q = Q^T \geq 0$, $R = R^T \geq 0$ are positive definite matrices and u is the control input that minimizes the cost function given by:

$$\Delta u = -R^{-1} B^T P \Delta z = -K \Delta z \tag{4.19}$$

where $P = P^T > 0$, which can be obtained by solving the Algebraic Riccati Equation:

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0 \tag{4.20}$$

The controller used here employs a feedforward-feedback structure, with F_y^f and F_x^r acting as model inputs. These model inputs are then mapped to motor (for F_x^r) commands and servo (for δ) commands that control the vehicle using the tire model in (4.10) and the longitudinal dynamics model in (4.9).

Since the LQR controller tries to maintain a state near the drift reference state, the front

wheels are often nearly or completely saturated. This means that the synthetic inputs F_y^f, F_x^r have a limited bandwidth to operate before saturating.

The LQR controller needs to stabilize the system about the desired operating point under the constraint of tire saturation. As the rear tires operate under saturation conditions, we still want the feedback control policy to satisfy the following constraint:

$$u_{\min} \leq \bar{u} + \Delta u \leq u_{\max} \tag{4.21}$$

where $u_{\min}$ and $u_{\max}$ are set based on the fiction limit. In light of this constraint, we want to compute a region of attraction under which the LQR feedback policy with stabilize the RC about drift, such that if the vehicle state enters that region and the controller activates, then the RC state will stay within the region and converge towards the target state.

Region of Attraction

In order to compute the region of attraction under input limits, we use Lyapunov's second method for stability. The method states that given a function $V(x) : \mathbb{R}^n \to \mathbb{R}$, if the following conditions hold true:

- $V(x) = 0$ if and only if $x = 0$
- $V(x) > 0$ if and only if $x \neq 0$
- $\dot{V}(x) = \frac{d}{dt}V(x) < 0$ for all values $x \neq 0$

then the system is stable in the sense of Lyapunov. Using the P matrix from the solution of the Ricatti equation, we can observe that the system is stable:

$$V(\Delta z) = \Delta z^\top P \Delta z \tag{4.22a}$$

$$\begin{aligned}
\dot{V}(\Delta z) &= \frac{d}{dt}\left(z^\top P \Delta z\right) \\
&= \Delta \dot{z}^\top P \Delta z + \Delta z^\top P \Delta \dot{z} \\
&= (A\Delta z + B\Delta u)^\top P \Delta z + \Delta z^\top P (A\Delta z + B\Delta u) \\
&= (A\Delta z - BR^{-1}B^T P \Delta z)^\top P \Delta z + \Delta z^\top P (A\Delta z - BR^{-1}B^T P \Delta z) \\
&= \Delta z^\top (AP^\top + PA - 2PBR^{-1}B^\top P)\Delta z && (4.22b) \\
&= -\Delta z^\top (Q + PBR^{-1}B^\top P)\Delta z && (4.22c) \\
&= -\Delta z^\top (Q + PBR^{-1}RR^{-1}B^\top P)\Delta z && (4.22d) \\
&= -\Delta z^\top (Q + K^\top R K)\Delta z && (4.22e)
\end{aligned}$$

This formulation, however, assumes the system inputs are unbounded, but in reality, the vehicle actuators saturate due to physical limits from the friction circle and steering system.

The state space region where the LQR is stabilizing and does not violate the input limits is formulated by the following optimization program:

$$\max_{\Delta z,\gamma} \quad \gamma \tag{4.23a}$$

$$\text{s.t.} \quad H\Delta z \leq w \qquad \forall \Delta z \in \{\Delta z \in \mathbb{R}^n : \Delta z P \Delta z \leq \gamma\} \tag{4.23b}$$

where $H = \begin{bmatrix} -K \\ K \end{bmatrix}$ and $w = \begin{bmatrix} u_{\max} - \bar{u} \\ \bar{u} - u_{\min} \end{bmatrix}$. The objective function in (4.23a) aims to maximize the region of attraction and the constraint in (4.23b) enforces limits on the system inputs. To simplify notation, we define the set $S(\gamma) = \{\Delta z \in \mathbb{R}^n : \Delta z P \Delta z \leq \gamma\}$

We can rewrite optimization program (4.23) into the following equivalent program:

$$\max_{\Delta z,\gamma} \quad \gamma \tag{4.24a}$$

$$\text{s.t.} \max_{\Delta z \in S(\gamma)} h_i \Delta z \leq w_i \qquad \forall i \in \{1, 2, ..., 2m\} \tag{4.24b}$$

where we re-express H and w as $H = \begin{bmatrix} h_1^\top & \dots & h_{2m}^\top \end{bmatrix}^\top$ and $w = \begin{bmatrix} w_1^\top & \dots & w_{2m}^\top \end{bmatrix}^\top$, with h_i and w_i representing the i-th row of H and w, respectively. The variable m is the dimension of the input vector u. Now we first focus on solving the optimization problem within constraint (4.24b). We can recast it to the following formulation:

$$\max_{\Delta z} \quad h_i \Delta z \tag{4.25a}$$

$$\text{s.t.} \quad \Delta z^\top P \Delta z \leq \gamma \tag{4.25b}$$

The formulations are nearly identical. We transformed constraint (4.23b) into (4.24b) by casting an optimization problem inside the left-hand side of constraint (4.24b).

For optimization program (4.25), we treat the value γ as a parameter and solve the program using the KKT conditions. First, we write the Lagrangian as:

$$L(\Delta z, \mu) = h_i \Delta z + \mu(\Delta z^\top P \Delta z - \gamma) \tag{4.26}$$

Now, we write the stationarity condition (4.27a) and complementary slackness condition

(4.27b) from the KKT conditions as below:

$$\nabla_x L(\Delta z, \mu) = h_i^\top + 2\mu P \Delta z = 0 \tag{4.27a}$$

$$\Delta z = -\frac{1}{2\mu} P^{-1} h_i^\top$$

$$\mu(\Delta z^\top P \Delta z - \gamma) = 0 \tag{4.27b}$$

$$\gamma = \Delta z^\top P \Delta z$$

Using the results from the KKT conditions, we can solve for the value of the Lagrangian multiplier μ:

$$\begin{aligned}
\gamma &= \Delta z^\top P \Delta z \\
&= \left(-\frac{1}{2\mu} P^{-1} h_i^\top\right)^\top P \left(-\frac{1}{2\mu} P^{-1} h_i^\top\right) \\
\mu &= \pm \frac{\sqrt{h_i P^{-1} h_i^\top}}{2\sqrt{\gamma}}
\end{aligned}$$

We now evaluate the value of the objective function (4.25a) by substituting the results for Δz and μ from above:

$$\begin{aligned}
h_i \Delta z &= h_i(-\frac{1}{2\mu} P^{-1} h_i^\top) \\
h_i \Delta z &= \pm \sqrt{\gamma} \sqrt{h_i P h_i^\top}
\end{aligned}$$

The optimization program (4.25) has two solutions, but we only take the positive one since we are solving a maximization problem. Next, we can substitute this result inside the constraint of (4.24) to arrive at the following program:

$$\max_{\Delta z, \gamma} \quad \gamma \tag{4.28a}$$

$$\text{s.t.} \quad \sqrt{\gamma} \sqrt{h_i P h_i^\top} \leq w_i \qquad \forall i \in \{1, 2, ..., 2m\} \tag{4.28b}$$

From (4.28), we can directly use the inequality constraint to solve for γ since the rest of

the terms in the constraint are given data quantities. We re-write the program as follows:

$$\max_{\Delta z, \gamma} \quad \gamma \tag{4.29a}$$

$$\text{s.t.} \quad \gamma \leq \frac{w_i^2}{h_i P h_i^\top} \qquad \forall i \in \{1, 2, ..., 2m\} \tag{4.29b}$$

Using constraint (4.29b), we can now write an explicit solution to the original program in (4.23):

$$\gamma = \min_i \frac{w_i^2}{h_i P h_i^\top} \tag{4.30}$$

The bound γ provides a conservative bound for the region of attraction. During online control, the first conducts an open loop maneuver based on how expert drivers initiate drift. The open-loop maneuver consists of driving the vehicle straight at the reference velocity, then steering aggressively to the left ($\delta > 15\,\text{deg}$) and accelerating briefly, and then counter-steering to the right. The open-loop maneuver is meant to bring the vehicle into the region of action. To determine if the vehicle state has entered the region of attraction, we simply check if the condition $\Delta z P \Delta z \leq \gamma$ is true. Once the system has reached a state under which the check condition is true, then the vehicle is inside the region of attraction and the closed loop LQR policy is activated. We remark that the open-loop maneuver is not guaranteed to bring the vehicle into the region of attraction, but based on experimental results, the system in practice enters the RoA as long as the yaw rate after the open-loop maneuver is at or above the reference yaw rate. The methods in this chapter on path planning can be used to systematically find an open-loop maneuver to bring the vehicle into the RoA.

Experimental Platform

To test the proposed LQR feedback policy, we used the first generation BARC platform. At the time of testing, we have not yet begun development of the second-generation BARC platform.

The parameters of RC-car and tires are summarized in Table 4.3.

Table 4.3: RC-car parameters

Parameter	Value	Parameter	Value
m [kg]	1.95	h [m]	0.05
I_z [kg $\cdot$ m^2]	0.24	B	7.4
L_f [m]	0.125	C	1.25
L_r [m]	0.125	μ	0.234

Simulation Results

The controller was initially validated using CarSim, a high fidelity vehicle simulation software. We used a hatchback vehicle as the plant since the software suite did not provide any models at the scale of an RC-car. The parameters of the vehicle and tires while stationary are summarized in Table 4.4. For simulation, we set the reference state at $\beta = -0.32$ rad,$r = 0.36$ rad/s, and $v_x = 25$ m/s, with an initial state of of $\beta^{eq} = 0$ rad,$r^{eq} = 0$ rad/s, and $v_x^{eq} = 25$m/s. We also designed a short open-loop maneuver to bring the vehicle in a region of the state space that would active the closed loop LQR controller. Once in the LQR region of attraction, the controller stabilizes the vehicle around that reference state.

Table 4.4: CarSim vehicle parameters

Parameter	Value	Parameter	Value
m [kg]	1830	h [m]	0.59
I_z [kg $\cdot$ m^2]	3287	B	12.6
L_f [m]	1.4	C	1.41
L_r [m]	1.65	μ	1.00

The vehicle states and control inputs during simulation are shown in Figure (4.7) and Figure (4.8). The blue lines in Figure (4.7) show the simulation measurements and the red lines show the control reference. For $t < 2$s, the open-loop maneuver is active, and at $t = 2$s, the closed loop controller is activated. As Figure (4.7) shows, the state of the vehicle converges to a steady-state condition, and reaches the reference state of $\bar{z} = \begin{bmatrix} -0.48, 0.39, 26 \end{bmatrix}^\top$ around $t = 10$s. From the simulation results, we observe an interplay between the yaw rate and side slip angle dynamics, namely that in order for the magnitude of the side slip angle to increase, the magnitude of the yaw rate needs to decrease. Decreasing the rear lateral force decreases the yaw rate, but also decreases the longitudinal velocity. The LQR controller rapidly manipulates the steering angle and longitudinal force to satisfy these competing objectives. We observe this from time $t = 3.5$s onward.

Experimental Results

As in simulation, we applied an open loop control input and a closed loop controller for the RC vehicle. The vehicle states and control inputs are shown in Figure (4.9) and Figure (4.10), respectively, with an initial vehicle state around $z = \begin{bmatrix} 0, 0, 2.7 \end{bmatrix}^\top$, and the desired reference state at $\bar{z} = \begin{bmatrix} -0.36, 1.62, 1.2 \end{bmatrix}^\top$. Again, the blue lines denote the measured values and the red lines denote the control reference.

For the open loop maneuver, the RC-car accelerates to the target longitudinal speed and then turns to the left. Shortly after turning, the LQR controller is activated, and the RC counter-steers (positive to negative steering angle), but the yaw rate still maintains a

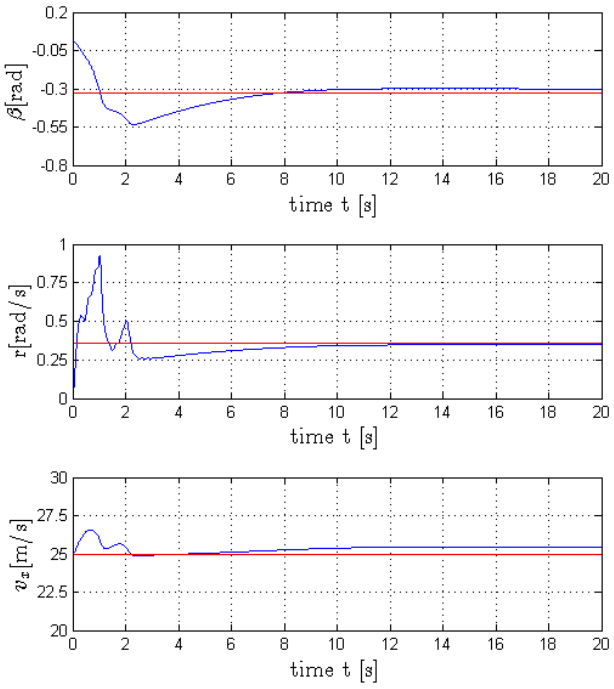


Figure 4.7: The CarSim vehicle state converges to the reference state.

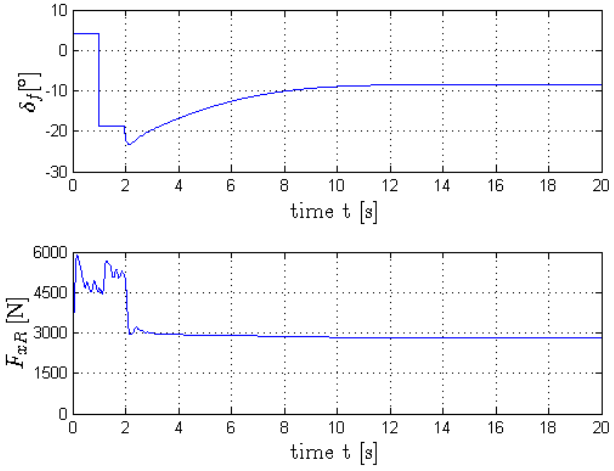


Figure 4.8: The CarSim vehicle input converges to the reference input.

high positive value, meaning that while the steering wheel is pointing in one direction, the vehicle is actually rotating in the other direction. At the same time, the rear wheel torque notably increases. The peak value in side slip plot at 9s is due to the sudden jump in the

lateral velocity measurement. A video of the experiment can be found at http://www.barc-project.com/projects/.

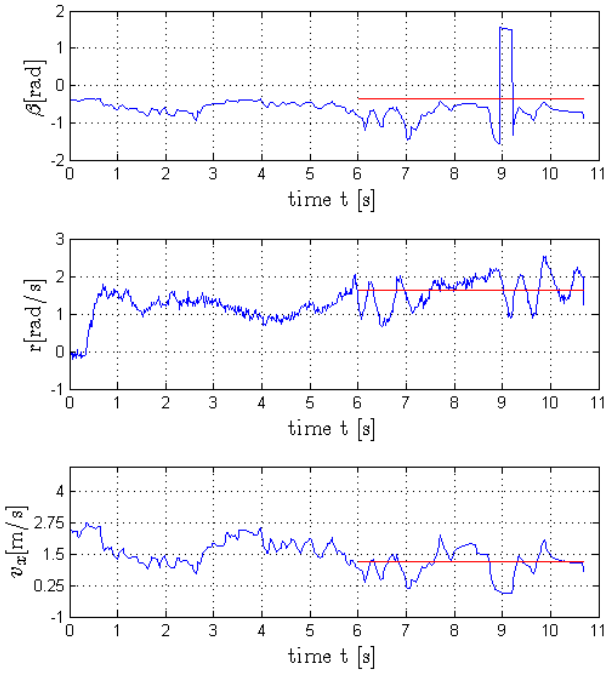


Figure 4.9: The RC state oscillates about the reference state.

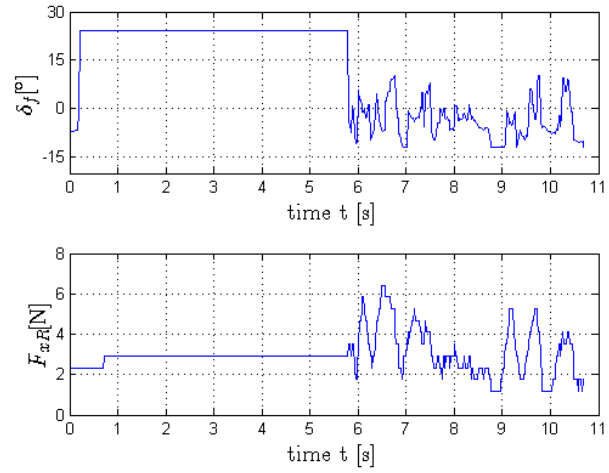


Figure 4.10: The RC state oscillates about the reference input.

The experimental results for drift parking overall maintained drift for about one-lap, but then fell out of drift. Aside from model mismatch, one difficult challenge to sustained steady drift was state estimation using only onboard sensors with only onboard computational resources from the Odroid. The optical flow algorithm provided noisy estimates of the velocity, and would often return completely invalid measurements or simply not return any measurement. The IMU and encoder gave noisy measurements. We attempted other estimation approaches like simultaneous localization and mapping (SLAM) to provide pose estimates that could be fed into the Extended Kalman Filter, but the Odroid did not have enough computational resources in the GPU to process the image data from the SLAM algorithms quickly. On the Odroid, the SLAM algorithm would run at 3 Hz.

4.2 Path Planning and Mixed Open-loop Closed-loop Control

Path planning and control represent two critical modules for autonomous vehicles. The path planning unit finds a sequence of states and inputs to bring the vehicle system from an initial state to final state by breaking down the movement task into discrete motions. The control unit then tracks the reference trajectory returned from the path planner on the actual system.

Path planning has a long history of research and study, and has been a popular topic in recent years with the rise of autonomous vehicles. At a high level, three popular strategies are commonly employed to generate paths: sample based, grid based, and optimization based methods.

Sample-based methods like Rapidly Exploring Random Tree (RRT) work on the principle of starting from an initial state, sampling an state or input, propagating the system forward, and then repeating until the goal state is reached. The various implementations of those steps give rise to different algorithms, but the core idea is to draw samples and apply them to the system until a path from the start to the end state is found. Sample-based algorithms work well on high dimensional spaces.

Grid-based methods put a discretized grid over the state space and then search for a sequence of actions through the gridded nodes to connect the start and end nodes. Nodes are data structures that incorporate state information as well as linkage information for the algorithm (i.e. which previous node and what action led to this node). Algorithms like depth-first search, breath-first search, A start (A*) explore the space one node at a time starting from an initial node. The search algorithm uses a stack or heap data structure to keep track of explored nodes and a basic criteria (breath-first or depth-first) to select which state to expand next. Algorithms like A* exploit domain knowledge of the space and use a heuristic function to decide which node to expand. Grid-based algorithms work well on low-order spaces, but suffer from the curse of dimensionality as the space dimension grows.

Optimization-based methods work by casting the path-planning problem into a finite

time optimal control problem. The objective function usually penalizes input energy or input changes over consecutive time steps, and the constraints enforce system dynamics and enforce desired start and end states. As with search-based method, optimization-based methods work well on low-order systems. Optimization solvers can exploit mathematical properties like linearity or convexity to reduce solve times and guarantee optimal solutions, but in practice, most physical system exhibit nonlinear properties. Nonlinear solvers return locally optimal solutions with quality that often depend on the initial conditions.

For path planning of high slip maneuvers like drift, grid-based and optimization-based techniques stumble because of the high-order system dynamics needed to adequately describe the motion of the vehicle. We propose a new simple, sample-based technique based on the observation that drift maneuvers can be described effectively as combinations of step input commands. Instead of sampling inputs commands, we sample parameters that characterize the command signals for drift maneuvers. For example, by observing an expert driver conducting a drift parking maneuver, we notice that he executes three precisely timed, yet basic actions: apply handbrake, turn steering wheel, and apply brake pedal. As shown in Figure 4.11, the command signals are parameterized by signal parameters.

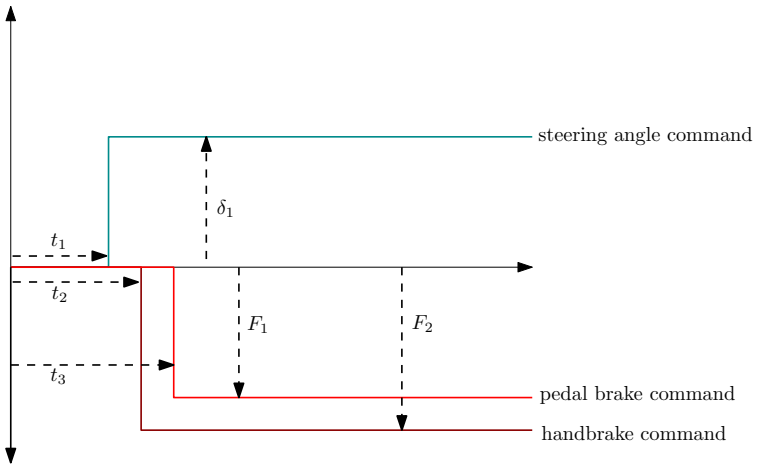


Figure 4.11: The control commands for aggressive maneuvers like drift roughly take the form of step functions that can easily by parameterized.

With those command signals, we can simulate the system forward for the duration of the command signal, and then test if the system landed in the desired set of terminal states. We explore this idea in more detail later when we discuss path planning for drift parking and drift cornering, but the main idea is simple - represent the control action as a step input command, and sample the parameters (magnitude and duration) that characterize it.

We now discuss control for tracking the reference commands and states from the path planner. The premise of the control design in this chapter is that given a fixed environment, vehicle systems operating in open-loop over short time scales behave deterministically, even if the dynamics are complex. This proposition means that when we initiate a drift maneuver

at approximately the same initial state and operate briefly in open-loop (i.e. using a fixed sequence of input commands), then the system will have a predictable, repeatable response.

We now move to the design of a mixed open-loop closed-loop control scheme for switching between the two modes of operation. We formulate an optimization program that takes in a reference trajectory $\bar{z}_{t\,t=0}^{N}$ from the path planner, and experimental data from an open-loop maneuver $(\{z_k^{\text{data}}\}_{t=L}^{N}, \{u_k^{\text{data}}\}_{t=L}^{N})$, where L is the time-step the maneuver begins, and write the program as follows:

$$\min_{\mathbf{z},\mathbf{u},\mathbf{z}_{CL},\mathbf{u}_{CL},\mathbf{z}_{OL},\mathbf{u}_{OL},m} \quad \sum_{k=0}^{N}(z_k - \bar{z}_k) \tag{4.31a}$$

$$\text{s.t.} \tag{4.31b}$$

$$z_{k+1} = f(z_k, u_k) \qquad \forall k \in \{0, 1, ...L\} \tag{4.31c}$$

$$z_{k+1} = z_{k+1}^{CL} m + z_{k+1}^{OL}(1-m) \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31d}$$

$$u_{k+1} = u_{k+1}^{CL} m + u_{k+1}^{OL}(1-m) \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31e}$$

$$z_{k+1}^{CL} = f(z_k, u_k) \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31f}$$

$$z_k^{OL} = z_k^{\text{data}} \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31g}$$

$$u_k^{OL} = u_k^{\text{data}} \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31h}$$

$$||z_k - z_k^{OL}|| \leq mM \qquad \forall k \in \{L, L+1, ...N-1\} \tag{4.31i}$$

$$m \in \{0, 1\} \tag{4.31j}$$

$$z_0 = z(0) \tag{4.31k}$$

where $z, z^{OL}, z^{CL} \in \mathbb{R}^n$ and $u, u^{OL}, u^{CL} \in \mathbb{R}^m$ are the optimal state and input, and n and m are the dimension of the state and input, and m is a binary variable. The bold face notation $\mathbf{z}$ indicates a set of optimization variables $\bar{z}_{k\,k=0}^{N}$.

The optimization program in (4.31) represents an optimal way of deciding if the system should operate in closed-loop or in open-loop. The objective function in (4.31a) follows a tracking formulation, where deviations from the desired reference trajectory are penalized. For the initial part of the maneuver, constraint (4.31c) forces the system to evolve according to a dynamics model f that sufficiently well describes the motion of the vehicle. At the moment the open-loop maneuver could start, we decompose the optimal state and input variable into z^{CL}, u^{CL} and z^{OL}, u^{OL}, in constraints (4.31d) - (4.31e), which represent closed loop and open-loop operation. The binary variable m takes forces from the main optimal state variable z to evolve according to either the open-loop dynamics or the closed-loop dynamics. The model for the open-loop dynamics comes from the experimental data $(\{z_k^{\text{data}}\}_{t=L}^{N}, \{u_k^{\text{data}}\}_{t=L}^{N})$, which is given in constraints (4.31g)-(4.31h). From our proposition, we argue this open-loop model is valid because the vehicle system behaves deterministically over short time scales. Constraint (4.31i) ensures that the system would operate in open-loop

for the duration of the maneuver. Constraint (4.31i) also ensures that the transition from closed-loop to open-loop are connected by the same state at time-step $k = L$. After solving the program, if the program return $m = 1$, the system operates in closed-loop, otherwise the system operates in open-loop.

We remark that optimization program (4.31) represents a simple, theoretical formulation for deciding between when to operate in closed-loop and open-loop. In the following sections on drift parking and drift cornering, we discuss control designs that are based on the ideas in this formulation.

4.3 Autonomous Drift Parking

The previous section focused on steady drift, which is just tracking a reference state. In the remaining sections, we focus on drift maneuvers that require planning. In this section, we focus on the design of a control scheme to park a vehicle with a sliding maneuver.

Drift parking represents an extreme maneuver that is beyond the skilled set of the average driver, requiring adept use and timing of the handbrake, pedal brake and steering wheel. The maneuver causes the vehicle to rotate rapidly and slide, nearly sideways (i.e. high side slip angle) into a desired parking spot.

The control scheme consists of an initial closed-loop segment, followed by an open-loop segment, which in combination closely track a reference drift-parking trajectory. The method is validated using the BARC platform.

Background

During motor sport events, many of the worlds best drivers deliberately operate their vehicle with saturated tires in order to perform a quick turn or a slide. Controlling drift maneuvers requires precise timing and coordination of multiple inputs. We observe from demonstration that for short-duration maneuvers like drift parking, however, the input commands are simple step-input combinations of input commands. While the dynamics of a transient drift maneuver are difficult to model accurately (and thus control using model-based techniques), the expert driver executes a basic sequence of control actions (turn wheel, apply handbrake, apply pedal brake). This observation motivates the design of a mixed closed-loop and open-loop architecture. When the vehicle dynamics are well modeled, the controller operates under a closed-loop algorithm, when not, the controller applies an open-loop sequence of commands. Such approaches are found through the literature [1], [16], [23].

Kolter and coauthors [23] worked on a similar project for drift parking. They designed a mixed open-loop closed-loop strategy using a probabilistic method called multi-model LQR based on a demonstration of the desired maneuver. In contrast, we designed a *deterministic* algorithm for mixed open and closed loop control that generalizes to other types of extreme maneuvers.

The following sections discuss the vehicle model, path planning, control design and state

estimation.

Vehicle model

We adopt two separate vehicle models to capture the system dynamics, a kinematic model and a kinetic model. The kinematic model forms the basis of the nonlinear MPC tracking control for the first part of the drift-parking maneuver, and the kinetic model forms the basis of the nonlinear observer for the entire experiment.

The state and input vectors for the kinematic model are $z = \begin{bmatrix} x & y & \psi & v \end{bmatrix}^\top$ and $u = \begin{bmatrix} \delta & a \end{bmatrix}$, respectively. The equations of motion are given as follows:

$$\dot{x} = v \cos \psi \tag{4.32a}$$
$$\dot{y} = v \sin \psi \tag{4.32b}$$
$$\dot{\psi} = \frac{v}{L} \tan \delta \tag{4.32c}$$
$$\dot{v} = a \tag{4.32d}$$

The state and input vectors for the dynamic bicycle model are $z = \begin{bmatrix} \beta & r & v_x \end{bmatrix}^\top$ and $u = \begin{bmatrix} \delta & F_x^r \end{bmatrix}^\top$, respectively. The equations of motion are given as follows:

$$\dot{\beta} = \frac{1}{m v_x}(F_y^f + F_y^r) - r \tag{4.33a}$$
$$\dot{r} = \frac{1}{I_z}(L_f F_y^f - L_r F_y^r) \tag{4.33b}$$
$$\dot{v}_x = \frac{1}{m}(F_x^r - F_y^f \sin \delta) + v_x r \beta \tag{4.33c}$$

As with the steady-state drift project, we also apply the Pacejka model to estimate the lateral force at high tire slip angles. For more details on the dynamic model, refer to section (4.1).

Control Design

Nearly all advanced driving maneuvers have some high side slip element, which manifests as sliding or drifting. Such maneuvers are beyond the skill set for the average driver, and therefore represent an interesting problem for control engineers. These high side slips maneuvers are difficult to control for two reasons: poor model fidelity (in highly nonlinear regimes) and actuator saturation. Additionally, the driver has three inputs at his disposal which he needs to coordinate: the steering angle, δ, the handbrake, $F_{\text{h-brank}}$, and the pedal brake, $F_{\text{p-brake}}$.

In the control architecture described later, we employ a mixed open-loop and closed-loop control approach. The controller operates in closed-loop during the initial tracking segment, under well-modeled conditions with a low side-slip angle, and then operates in open-loop for the actual drift maneuver. To illustrate this, we show an example parallel parking trajectory in Figure (4.12). The part of the trajectory encircled in green consists mainly of straight driving and slight turning, which is well-modeled. The part encircled in red involves is the high side-slip sliding motion, which is difficult to model accurately.

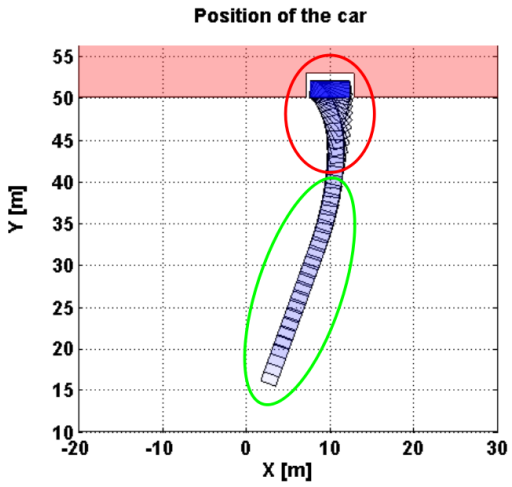


Figure 4.12: The vehicle has a low side-slip angle in the highlighted green segment and a high-slip angle in the red segment.

Path Planning

The objective of the path planner is to generate an input sequence that results in the vehicle sliding into the parking spot without any collisions (i.e. the vehicle remains within boundaries of the parking spot at all times). On a typical full-scale vehicle, the input signals are steering angle, throttle, hand brake and pedal brake. The handbrake acts only on the rear wheels, while the pedal brake acts on both the front and rear wheels [3]. For the 1/10-scale RC car used for experimentation, we use steering angle and longitudinal force as inputs, $u = \begin{bmatrix}\delta & F_x^r\end{bmatrix}^\top$. The RC has a single motor that functions as both an accelerator and a brake.

For the path planner, each input signal takes the form of a step function (or sum of multiple step functions) that is parameterized by two parameters: step time and magnitude. The rule-based path planner samples these parameters from a uniform distribution $\texttt{unif}(a, b)$, where a and b represent the lower and upper bounds of the sample space. The limits of the sample domain a, b are based on observing expert driver conducting extreme maneuvers.

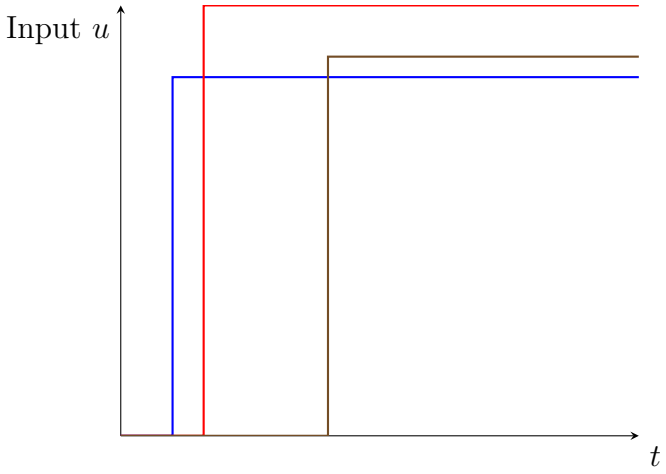
During a drift parking maneuver, the driver executes a sequence of precisely timed actions. He first swerves the vehicle with the steering wheel, next pulls the handbrake (causing the

vehicle's wheels to lock), and then applies the pedal brake. Based on this sequence of maneuvers (i.e. first steer, next use handbrake, then pedal brake), the authors select an appropriate range (a, b) for each parameter. This allows the path planner to sample from a smaller range and more quickly find a successful input sequence.
Each input signal is of the form below:

$$\delta(t) = \begin{cases} 0 & : t \leq t_1 \\ \delta_1 & : t_1 < t \end{cases} \tag{4.34}$$

$$F_{\text{h-brake}}(t) = \begin{cases} 0 & : t \leq t_2 \\ F_{\text{hb}} & : t_2 < t \end{cases} \tag{4.35}$$

$$F_{\text{p-brake}}(t) = \begin{cases} 0 & : t \leq t_3 \\ F_{\text{pb}} & : t_3 < t \end{cases} \tag{4.36}$$



where $t_1 \leq t_2 \leq t_3$.

After specifying the sampling space for each input signal, the following algorithm is used to search for a successful trajectory:

Algorithm 1 Path planner for drift parking

1: Define obstacles
2: Define initial state z_0
3: Define upper iteration limit M
4: **while** $i \leq M$ **do**
5: $\quad \{u_t\}_{t=0}^{T} \leftarrow \texttt{unif}(a, b)$
6: $\quad \{z_t\}_{t=0}^{T} \leftarrow f(z_0, U)$
7: $\quad$ **if** collisionFree($\{z_t\}_{t=0}^{T}$) **then**
8: $\quad\quad \{\bar{u}_t\}_{t=0}^{T} \leftarrow \{u_t\}_{t=0}^{T}$
9: $\quad\quad \{\bar{z}_t\}_{t=0}^{T} \leftarrow \{z_t\}_{t=0}^{T}$
10: $\quad\quad$ **return** $(\{\bar{z}_t\}_{t=0}^{T}, \{\bar{u}_t\}_{t=0}^{T})$
11: $\quad$ **end if**
12: $\quad i \leftarrow i + 1$
13: **end while**

In Algorithm 1, the input sequence $\{u_t\}_{t=0}^{T}$ is generated from parameters that are sampled from the uniform distribution $\texttt{unif}(a, b)$. The initial state z_0 and input sequence $\{u_t\}_{t=0}^{T}$ are then fed into the vehicle model to generate the vehicle's trajectory. Lastly, the function *collisionFree* checks if the vehicle remains within the parking spot boundaries. If successful, then the algorithm returns the reference state trajectory $\{\bar{z}_i\}_{t=0}^{T}$ and reference input trajectory $\{\bar{u}_i\}_{t=0}^{T}$, otherwise it continues to sample and generate other candidate step functions. The vehicle model f from line 6 comes from CarSim, a high fidelity vehicle dynamics software. For the experiments, the vehicle's trajectory was obtained from a demonstration. This corresponds to the Algorithm (1) terminating in one iteration since the demonstration trajectory is collision free. Note, the reference trajectory is *discrete*, since these states and inputs are time stamped values that we record during the simulation or experiment.

Algorithm 1 takes on a simple structure and produces a successful trajectory because we sample from a narrow input space, which is informed from observations (i.e. the bounds of the sample space are based on expert demonstration). Demonstration data also informs the initial state from which we initiate the sampled drift parking maneuver.

Path Following and Switched Control

The controller tracks a drift parking trajectory by switching between a nonlinear MPC program and a linear feedforward-feedback control policy. The controller determines automatically when to switch based on the feasibility of the MPC optimization routine. At each time step, the controller first localizes itself to the nearest point on the reference trajectory, and then attempts to solve an MPC optimization routine to track the reference trajectory in the specified time horizon.

The controller first locates the nearest point on the reference trajectory $\{z_i\}_{t=0}^{T}$ from the current state estimate $\hat{z}_t$ according to the optimization program in equation (4.37):

$$D_t = \min_i \|z_i - \hat{z}_t\|_2 \tag{4.37}$$

We use the index i to construct a target tracking trajectory $\{\bar{z}_i\}_{i=0}^N, \{\bar{u}_i\}_{i=0}^N$ for the MPC at each time step, where N is the horizon length. The target tracking trajectory is a subset of the reference trajectory.

The MPC routine attempts to track the target trajectory by solving the following optimization problem at each time step t:

$$\min_{\{z_k\}_{k=0}^N, \{u_k\}_{k=0}^{N-1}} \quad \sum_{k=0}^{N-1} (\|z_k - \bar{z}_k\|_Q^2 + \|u_k - \bar{u}_k\|_R^2) + \sum_{k=1}^{N-1} \|\Delta u_k\|_{R_\Delta}^2 + \|z_N - \bar{z}_N\|_P^2 \tag{4.38a}$$

$s.t$

$$z_{k+1} = f(z_k, u_k) \qquad \forall k \in \{0..., N-1\} \tag{4.38b}$$

$$\Delta z_k = T_s^{-1}(z_k - z_{k-1}), \quad \Delta z_k \in \Delta\mathcal{Z} \qquad \forall k \in \{1..., N\} \tag{4.38c}$$

$$\Delta u_k = T_s^{-1}(u_k - u_{k-1}), \quad \Delta u_k \in \Delta\mathcal{U} \qquad \forall k \in \{1..., N-1\} \tag{4.38d}$$

$$\|(e_1 + e_2)^\top (z_k - \bar{z}_k)\| \leq D_t \qquad \forall k \in \{0..., N\} \tag{4.38e}$$

$$\|e_3^\top (z_k - \bar{z}_k)\|_2^2 \leq \epsilon \qquad \forall k \in \{0..., N\} \tag{4.38f}$$

$$z_k \in \mathcal{Z} \qquad \forall k \in \{0..., N\} \tag{4.38g}$$

$$u_k \in \mathcal{U} \qquad \forall k \in \{0..., N-1\} \tag{4.38h}$$

$$z_0 = \hat{z}_t \tag{4.38i}$$

In MPC program (4.38), $\{z_k\}_{k=0}^N$ and $\{u_k\}_{k=0}^{N-1}$ represent the optimized state and input variables, and $\{\bar{z}_k\}_{k=0}^N$ and $\{\bar{u}_k\}_{k=0}^N$ denote the target state and input vectors, with k indexing the time step. For the objective function, $Q, P \in \mathbb{S}_{++}^{n\times n}$ represents the state penalty and terminal state penalty matrix, respectively. $R, R_\Delta \in \mathbb{S}_{++}^{m\times m}$ represents the input penalty and input difference penalty matrix respectively. $\mathbb{S}_{++}^{n\times n}$ denotes the set of positive definite symmetric matrices of size n by n, where n is the dimension of the state vector and m is the dimension of the input vector. The norm operator is defined as $\|z\|_Q^2 = z^\top Q z$.

The objective function (4.38a) follows a standard tracking MPC formulation, in which deviations from the reference state and reference input trajectories are penalized. The term $\Delta u_k^\top R_\Delta \Delta u_k$ penalizes large changes in the input over consecutive time steps, which produces a smooth input path. The matrices Q, R, R_Δ, P are all user-defined penalty matrices that trade off weights between the multiple objectives expressed in the objective function.

Constraint (4.38b) imposes system dynamics, which are the kinematic equation given from (4.32a) - (4.32d)), and the initial condition constraint is given in (4.38i). Constraints (4.38c) and (4.38d) limit state and input changes over consecutive time steps, where T_s denotes sampling time in the system. Constraint (4.38g) and (4.38h) limit the state and input magnitude.

Constraint (4.38e) places a hard limit on the maximum position deviation from the trajectory, where e_i is the unit vector along the i-th axis and D_t is the last estimated deviation from the trajectory. Constraint (4.38f) places hard limits on the deviation of the heading angle.

The MPC controller tracks the trajectory closely when the dynamics are well modeled, notably during the initial phase of the maneuver, which brings the vehicle to the target speed and orientation from which to initiate drift. Once the controller attempts to track the actual aggressive segment of the drift maneuver with large steering and brakes, the vehicle begins to slide and the model becomes unreliable. The MPC program will become infeasible, and will fail to find a solution that can track the trajectory. This behavior is expected since we only supply a kinematic model and enforce strict constraints to follow the state trajectory of a high side-slip maneuver. When the controller returns an infeasible solution status, the control system switches to a feedforward-feedback control policy with a linear gain K. The feedforward term, however, dominates the behavior of the control policy. The overall switched control architecture is given by Algorithm 2.

Algorithm 2 Switched control algorithm

Inputs
    $\{z_i\}_{t=0}^{T} \quad \{u_i\}_{t=0}^{T} \quad \hat{z}_t$
while while $t < t_f$ **do**
    $y \leftarrow$ measurement
    $\hat{z}_t \leftarrow \text{ekf}(\hat{z}_{t-1}, y_t, u_{t-1})$
    $D_t \leftarrow \min_k \|z_k - \hat{z}_t\|_2$
    $k \leftarrow \text{minarg}_k \|z_k - \hat{z}_t\|_2$
    $\{\bar{z}_i\}_{i=0}^{N} \leftarrow \{z_i\}_{i=k}^{k+N}$
    $\{\bar{u}_i\}_{i=0}^{N-1} \leftarrow \{u_i\}_{i=k}^{k+N-1}$
    $[\text{status}, U^{\text{MPC}}] \leftarrow \text{solveMPC}(\{\bar{z}_i\}_{i=0}^{N}, \{\bar{u}_i\}_{i=0}^{N-1}, \hat{z}_t, D_t)$,
    if status = feasible **then**
        $u_t \leftarrow u_0^{\text{MPC}}$
    else
        $u_t \leftarrow \bar{u}_0 + K_P(\hat{z}_t - \bar{z}_0)$
    end if
end while

Control algorithm (2) is structured to track the drift park maneuver. At each time step, the algorithm gets a new measurement y and estimates its state using an extended Kalman filter (EKF), next it localizes itself with respect to the reference trajectory $\{z_i\}_{t=0}^{T}$, then extracts $N+1$ reference states and inputs to form the target trajectory $\{\bar{z}_i\}_{i=0}^{N}, \{\bar{u}_i\}_{i=0}^{N-1}$ for the MPC. Lastly, the controller attempts to solve the MPC, and when it fails, it switches to a linear feedforward-feedback control policy using only the input reference information.

During the drift slide, the control of the rear wheels operate in open-loop. The controller sends a locking command to the wheels, which causes sliding due to the angular momentum

already generated from the initial aggressive turn. The steering wheel operates primarily in open-loop during the drift, but receives feedback from the yaw angle ψ measurement coming from the IMU. We use a proportional controller with a constant gain matrix $K_p \in \mathbb{R}^{p \times n}$, where p is the number of control inputs and n is the dimension of the state vector. The matrix K_p is almost entirely zero, except for one diagonal entry $k_{\Delta\psi}$, which multiplies the yaw angle error.

The value of $k_{\Delta\psi}$ improves tracking, but the feedforward term $\bar{u}_t$ dominates the control signal. For a simple proportional controller, setting $k_{\Delta\psi}$ too large would result in oscillatory motion about the target trajectory.

Experimental Results

The BARC platform was used to test the algorithm. As with the steady-state drifting experiment, an EKF with a dynamic bicycle model was designed to localize the vehicle. Accuracy was limited since only on-board sensors (IMU and encoders) were available at the time of the experiment. Localization is not an issue when the vehicle initiates the drift maneuver when swerving the steering wheel since the rest of the input commands all operate in open loop.

All experiments were conducted in an empty indoor room with the reference trajectory obtained from a demonstration. The tracking performance with and without the proposed architecture is shown in the Figure 4.13 and Figure 4.14. Using just a pre-recorded open loop sequence of commands, the vehicle not only fails to track the reference trajectory, but follows different paths with each trial, even under static environmental conditions. Also, a strong bias in the steering system can be observed from figure 4.14. The proposed algorithm had an average tracking error of around 0.24 m RMS, while the pure open loop one has an average error of around 1.5 m RMS. Since measurements for position could only be estimated using onboard sensor and an EKF algorithm, the plotted trajectories deviate from the ground truth, but the final position was estimated by manually measuring the position of the vehicle with measuring tape. The target parking spot was narrow and fixed between static cardboard boxes

We tested both accuracy and repeatability of our algorithm by trying to track the same trajectory multiple times. Experimental results show that the proposed algorithm is applicable in practice with using only low-cost on-board sensors. Video of the maneuver can be found in http://www.barc-project.com/projects/ and the frame-by-frame breakdown is shown in Figure 4.15.

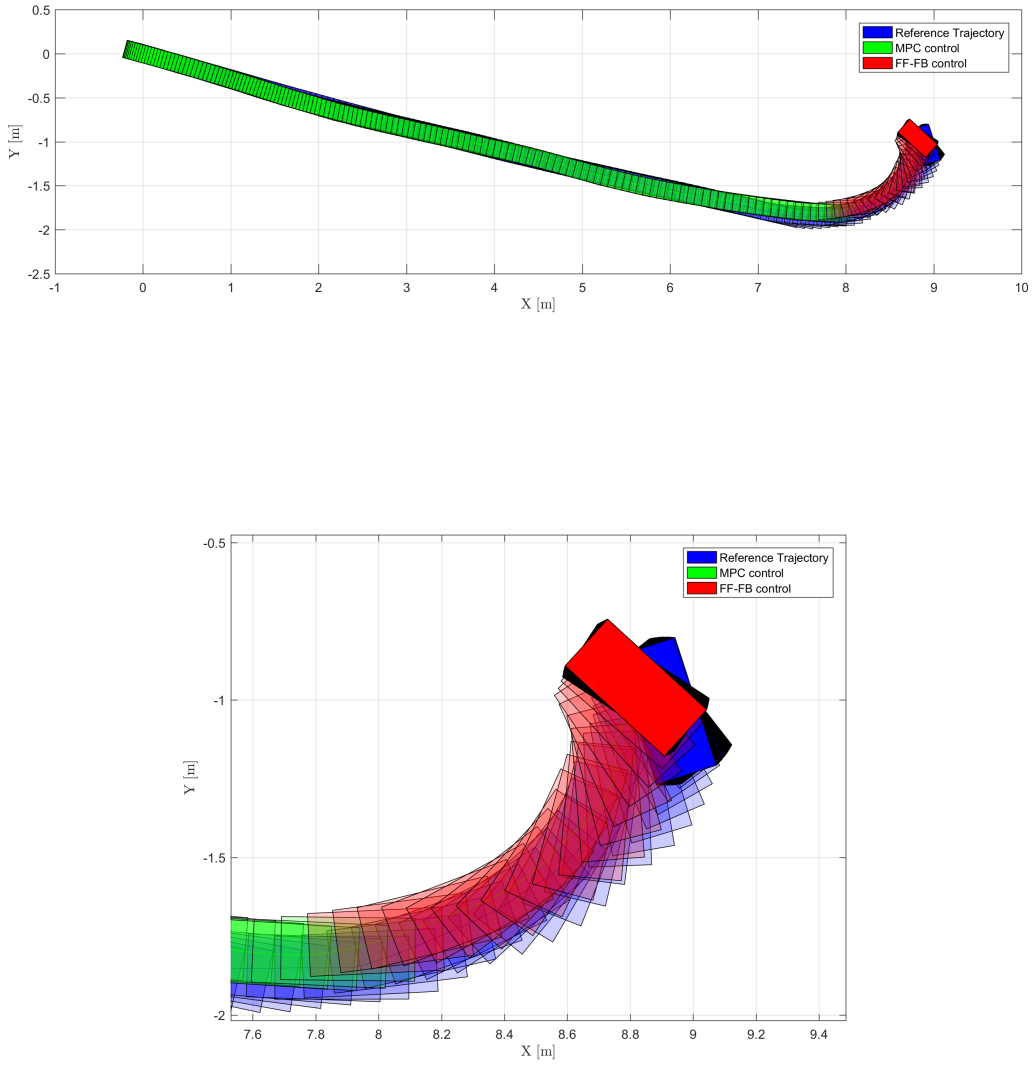


Figure 4.13: The switched controller tracks the reference trajectory closely. The blue trajectory denotes the reference trajectory. All other trajectories which track the reference trajectory are denoted in red and green, which indicate when the control operates under MPC and when it operates under the feedforward (FF) - feedback (FB) control policy.

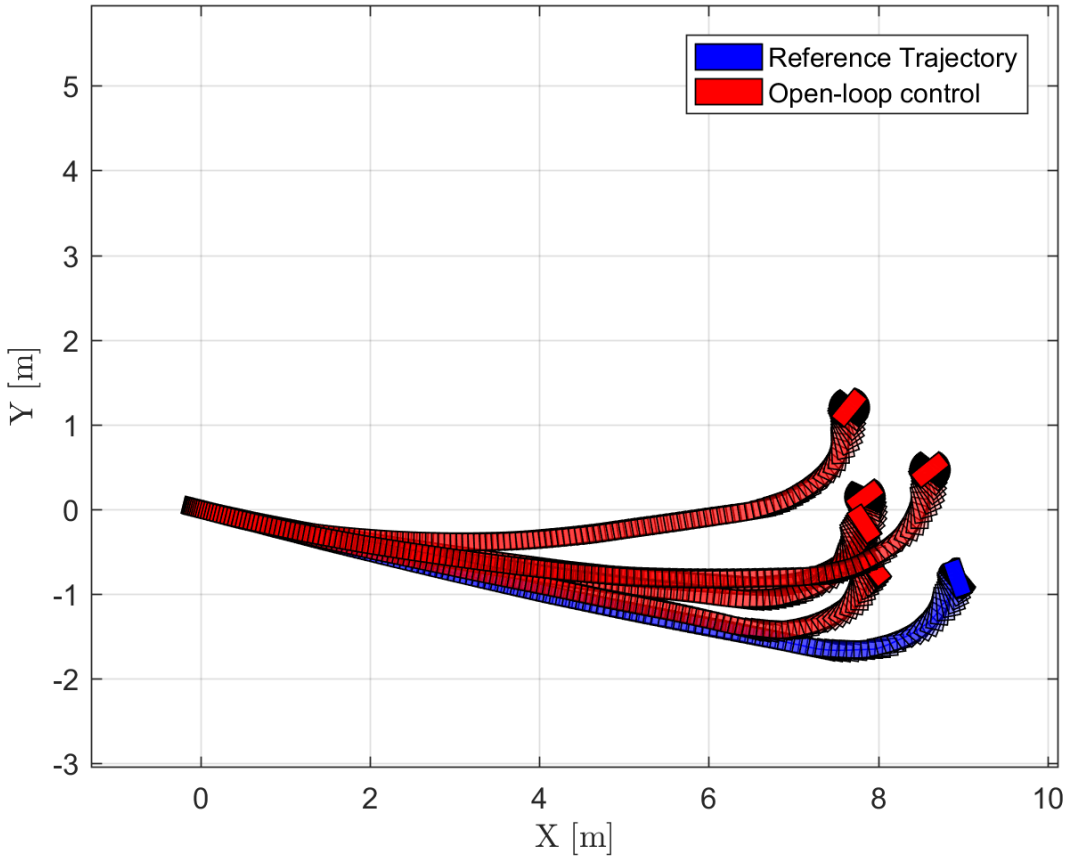


Figure 4.14: Applying a pure open-loop controller over a long time horizon fails to track the reference trajectory. The blue trajectory denotes the reference trajectory and the red trajectories are all experimental results from applying the same control inputs as the reference trajectory in open-loop.

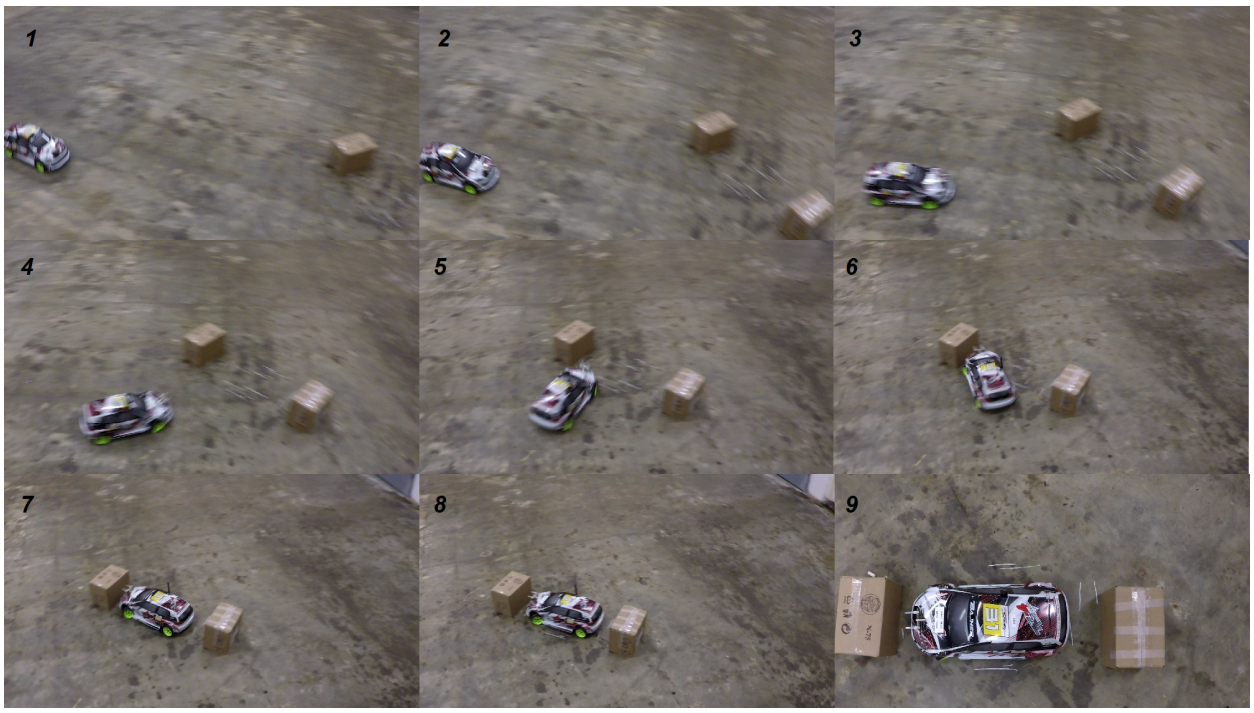


Figure 4.15: The frame-by-frame image shows the vehicles approaching, turning, and then sliding into the parking spot between the boards during an experimental run.

4.4 Autonomous Drift Cornering

In the previous section, we explored planning and control for autonomous parking. We now explore another extreme autonomous maneuver, drift cornering. Drift cornering occurs when a driver turns a corner very quickly and loses traction with the ground surface, causing the vehicle to drift. As with other drifting maneuvers, expert drivers in rally races engage in drift cornering often to minimize lap time.

In this section, we extend the planning and control strategy for autonomous drift cornering. The planning algorithm for drift parking applies for drift cornering. The control strategy also utilizes a switched open-loop (OL) closed-loop(CL) structure to track the cornering trajectory, but instead evaluates the cost of OL and CL policies to determine when to switch.

Background

Drift cornering has emerged as a research topic in the past decade for vehicle applications. Velenis and others investigated optimal drift maneuvers in simulation to minimize the time need to round a turn and exit with a maximum corner velocity [43] [44]. Kolter and others applied a mixed open-loop and closed-loop maneuver using a probabilistic method called multi-model LQR to repeatedly slide into a parking spot with a full scale vehicle [23]. Tavernini and others utilized nonlinear optimal control theory to investigate the optimality of the handbrake cornering technique for a front wheel drive vehicle [40].

The work in the following sections builds on the planning and control strategy in drift parking and applies it now to drift cornering, with modifications in the controller.

Vehicle Model

We capture the vehicle dynamics using a six-state bicycle model with linear front and rear wheel tire forces. The state and input vector are $z = \begin{bmatrix} v_x & v_y & r & x & y & \psi \end{bmatrix}^\top$ and $u = \begin{bmatrix} \delta & F_x^r \end{bmatrix}$, respectively.

The single track bicycle model is very similar as the one used in the steady state drift and drift parking projects. The primary difference lies in the equation of the lateral dynamics. For drift cornering, position is important, so the state vector includes longitudinal and lateral velocity, which can be integrated and rotated to give position in the global coordinate frame. For the steady state drift project, the controller only needs to track the target slip angle, yaw rate and longitudinal velocity, the actual position of the vehicle is not important. The

following dynamic bicycle model is used in the control design of corner drift maneuver:

$$\dot{v}_x = \frac{1}{m} F_x^r \tag{4.39a}$$
$$\dot{v}_y = \frac{1}{m}(F_y^f + F_y^r) - v_x r \tag{4.39b}$$
$$\dot{r} = \frac{1}{I_z}(L_f F_y^f - L_r F_y^r) \tag{4.39c}$$
$$\dot{X} = v_x \cos\psi - v_y \sin\psi \tag{4.39d}$$
$$\dot{Y} = v_x \sin\psi + v_y \cos\psi \tag{4.39e}$$
$$\dot{\psi} = r \tag{4.39f}$$

where F_y^f and F_y^r are the lateral forces on the front wheel and real wheel. The parameters m and I_z are the mass and moment of inertia about $\mathbf{e}_3$ axis. L_f and L_r represent the distance from the front and rear axles to the CoG.

For the tires, we use a linear model to estimate the lateral force rather than a nonlinear Pacejka model as in the previous two projects. Parameters in linear tire models are easier to estimate and work well when the tire is not saturated. Additionally, for the drifting maneuver, the controller will heavily operate in open-loop (model-free). We use the following equations to estimate the lateral forces on each wheel:

$$F_y^f = -C_f \alpha_f \tag{4.40a}$$
$$F_y^r = -C_r \alpha_r \tag{4.40b}$$

where C_i is the cornering stiffness of front or rear wheel and α_i is the side slip angle of the wheel, for $i = \{f, r\}$, expressed as:

$$\alpha_f = \frac{v_y + r L_f}{v_x} - \delta \tag{4.41}$$
$$\alpha_r = \frac{v_y - r L_r}{v_x}. \tag{4.42}$$

In state space form, the dynamic equations are completely expressed as shown below:

$$\dot{z} = f(z, u) \tag{4.43}$$

Path Planning

As with steady state drift and drift parking, drift cornering is characterized by large side slip angles, counter-steering and tire force saturation. For path planning of transient drift

maneuvers, we adopt the same strategy as for parking drift maneuvers - we observe expert drivers and characterize their control inputs as simple combinations of step-input commands, then parameterize the input profiles and sample.

During a drift cornering process, the driver executes a sequence of simple, but carefully timed actions. The driver initially drives straight before the track corner, then he turns the corner and applies a large rear-drive torque. At this point, the vehicle begins to slide, and the driver then both counter-steers and decreases the rear-drive torque. Without this counter steer, the vehicle would spin out. As the driver exits the corner, the vehicle recovers to a stable condition, with a small side slip and non-saturated tires. The general sequence of drift cornering inputs is illustrated in Figure 4.16.

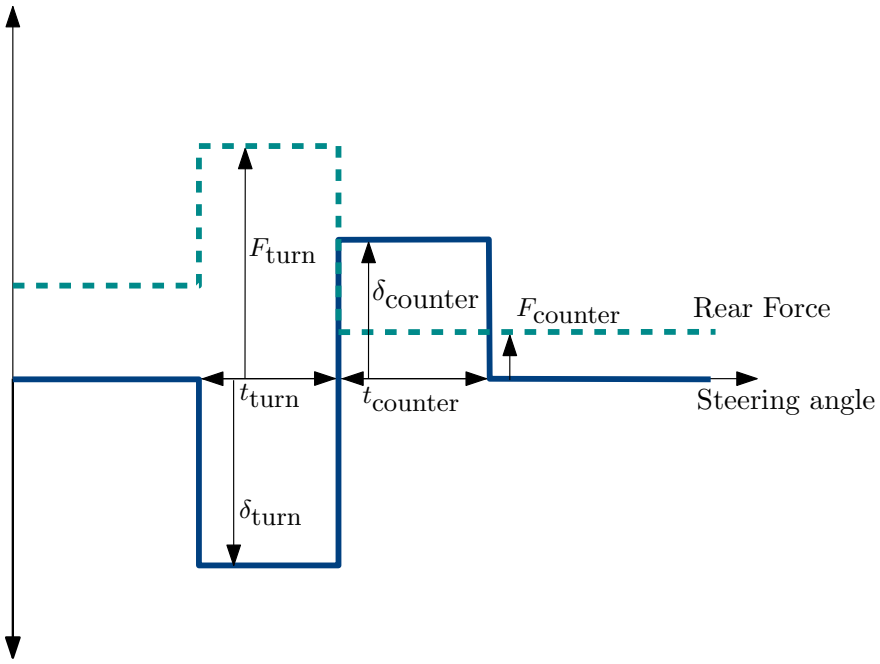


Figure 4.16: Parametrized control sequence for drift cornering

The generated input sequences for the input steering angle δ and the input rear force F_{xR} are parametrized by the following parameters:

t_{turn} – duration of initial turn and throttle
δ_{turn} – steering angle of the turn in phase
F_{turn} – magnitude of rear force during initial turn
t_{counter} – duration of the counter steering
δ_{counter} – steering angle of the counter steer
F_{counter} – magnitude rear force during counter steering

Table 4.5: Rule-based path planning algorithm

1 Define the initial vehicle state and track boundary
2 Sample each parameter from a uniform distribution
3 Conduct experiments using inputs based on the sampled parameter set
4 Check if the resulting trajectory is drifting inside the corner. If unsuccessful, return to step 2.

$$\delta(t) = \begin{cases} 0 & : t \leq t_1 \\ \delta_{\text{turn}} & : t_1 < t \leq t_1 + t_{\text{turn}} \\ \delta_{\text{counter}} & : t_1 + t_{\text{turn}} < t < t_1 + t_{\text{turn}} + t_{\text{counter}} \end{cases} \tag{4.44}$$

$$F_{xR}(t) = \begin{cases} F_{\text{nominal}} & : t \leq t_1 \\ F_{\text{turn}} & : t_1 < t \leq t_1 + t_{\text{turn}} \\ F_{\text{counter}} & : t_1 + t_{\text{turn}} < t < t_1 + t_{\text{turn}} + t_{\text{counter}} \end{cases} \tag{4.45}$$

Table (4.5) outlines the steps of the rule-based path planning algorithm. The planner uniformly samples each parameter to generate an input sequence for drift cornering. The sampling domain for each parameter is based on general observations from expert drivers, noting features like steering angles and duration. These observations guide the boundaries of the sampling domain to become narrow, which reduces the time to sample a sequence of parameters that produce an input sequence that results in drift cornering.

During a simulation or experiment in which the vehicle drifts the corner successfully, we set the recorded states and inputs as the reference trajectory. This reference trajectory $\{\bar{z}_i\}_{t=0}^{T}$, $\{\bar{u}_i\}_{t=0}^{T}$ then guides the design of the mixed open-loop closed-loop control strategy.

We remark that one additional user-defined parameter in the path planning process is the initial longitudinal velocity of the vehicle v_x. The rule-based algorithm assumes the vehicle is starting from a start with a positive longitudinal velocity and zero lateral velocity and zero yaw rate. The force quantity F_{nominal} acts on the vehicle before the drift maneuver to maintain a user-define target velocity. The first step in Algorithm (4.5) requires the user to define the initial longitudinal velocity (only non-zero component of initial state), and a boundary box (in terms of X and Y coordinates) from which to initiate the sampled drift maneuver.

Control law

The mixed open-loop closed-loop controller stabilizes the vehicle around a reference drift corner trajectory. Model-based controllers using conventional bicycle models with Pacejka

Table 4.6: Offline segment

1 Define the reference trajectory $\{\bar{z}_i\}_{t=0}^{T}$, $\{\bar{u}_i\}_{t=0}^{T}$ from output of path planner
2 Linearize vehicle model along the designed reference trajectory
3 Define the feedforward-feedback control policy $\Delta u_i = -K_i \Delta z_i$ $u_t = \bar{u}_i + \Delta u_i$
4 Compute errors between predicted and reference states $e_i = \hat{\bar{z}}_i - \bar{z}_i$

Table 4.7: Online segment

1 Find the nearest point in the designed trajectory
2 Propagate vehicle dynamics forward in time for both closed-loop and open-loop policies
3 Compare cost functions of predicted closed-loop and predicted open-loop trajectories
4 Select the control policy command with less cost

tire models do not track a drift corner trajectories well because complex transient dynamics during the maneuver. During a short period of time, however, a fixed open-loop input sequence launched from the same initial state (under static environmental conditions) produces a repeatable response ([23]), even for systems undergoing complex dynamic processes like drift cornering. We exploit the deterministic behavior of complex dynamics under a short duration and design a mixed open-loop closed-loop controller that is designed to switch between closed-loop and open-loop operation.

As with the drift parking controller, this mixed control strategy computes an optimal control policy offline using LQR (rather than online using MPC), but has a different mechanism for when deciding to operate in open-loop. The previous formulation based the switch on feasibility conditions from the MPC solver output, but in this formulation, we based the switch on values from the cost function.

At a high level, the mixed OL-CL control design is broken down into two parts, an offline segment and online segment. The offline segment in Table (4.6) computes optimal feedback policies, and the online segment in Table (4.6) determines whether to operate in closed-loop (using the pre-computed feedback policy) or in open-loop. Details are discussed next.

Offline segment

The offline segment focuses designing the optimal feedback (i.e. CL) control policies that the mixed controller uses when running online. We use LQR to design a sequence of control policies by first linearizing the vehicle model in (4.39) about the state and input to get the matrices $A = df/dz$ and $B = df/du$. The analytical expressions of the non-zero entries of the system matrix A_{ij} are given below, where i and j refer to the row and column, respectively.

$$a_{21} = \frac{C_f + C_r}{mv_x^2} v_y - \frac{L_r C_r - L_f C_f}{mv_x^2} r - r \tag{4.46a}$$

$$a_{22} = -\frac{C_f + C_r}{mv_x} \tag{4.46b}$$

$$a_{23} = \frac{L_r C_r - L_f C_f}{mv_x} - v_x \tag{4.46c}$$

$$a_{31} = -\frac{L_r C_r - L_f C_f}{I_z v_x^2} v_y + \frac{L_f^2 C_f + L_r^2 C_r}{I_z v_x^2} r \tag{4.46d}$$

$$a_{32} = \frac{L_r C_r - L_f C_f}{I_z v_x} \tag{4.46e}$$

$$a_{33} = -\frac{L_f^2 C_f + L_r^2 C_r}{I_z v_x} \tag{4.46f}$$

$$a_{41} = \cos\psi \tag{4.46g}$$

$$a_{42} = -\sin\psi \tag{4.46h}$$

$$a_{46} = -v_x \sin\psi - v_y \cos\psi \tag{4.46i}$$

$$a_{51} = \sin\psi \tag{4.46j}$$

$$a_{52} = \cos\psi \tag{4.46k}$$

$$a_{56} = v_x \cos\psi - v_y \sin\psi \tag{4.46l}$$

$$a_{63} = 1 \tag{4.46m}$$

and the non-zero entries of the control matrix B are given as following

$$b_{12} = \frac{1}{m} \tag{4.47a}$$

$$b_{21} = \frac{C_f}{m} \tag{4.47b}$$

$$b_{31} = \frac{L_f C_f}{I_z} \tag{4.47c}$$

Next, we numerically evaluate the matrices A and B at each reference state and input

pair $\{\bar{z}_i\}_{t=0}^{T}$, $\{\bar{u}_i\}_{t=0}^{T}$, where i indexes a single state/input pair:

$$A_i = \left.\frac{df}{dz}\right|_{\substack{z=\bar{z}_i \\ u=\bar{u}_i}} \qquad B_i = \left.\frac{df}{du}\right|_{\substack{z=\bar{z}_i \\ u=\bar{u}_i}} \tag{4.48}$$

For each reference pair, the error dynamics of the system are given by:

$$\Delta \dot{z}_i = A_i \Delta z_i + B_i \Delta u_i \tag{4.49}$$

where $\Delta z_i = z - \bar{z}_i$ and $\Delta u_i = u - \bar{u}_i$.

The optimal feedback policy is based on LQR with the following quadratic cost function:

$$J = \int_{t=0}^{\infty} (\Delta z^\top Q \Delta z + \Delta u^\top R \Delta u) dt \tag{4.50}$$

with state penalty matrix $Q \in \mathbb{S}_{++}^{n\times n}$ and input penalty matrix $R \in \mathbb{S}_{++}^{m\times m}$. The control input $\Delta\mathbf{u}$ that minimizes the cost function is given by the following

$$\Delta u = -R^{-1} B^T P \Delta z = -K \Delta z \tag{4.51}$$

where $P \in \mathbb{S}_{++}^{n\times n}$, which can be obtained by solving Riccati equation. For *each* reference state and input pair $(\bar{z}_i, \bar{u}_i)$ we recorded during the path planning phase, we now have a feedback matrix K_i.

$$u_t = \bar{u}_i + \Delta u_i \tag{4.52}$$

For the last part of the off-line design, we compute the state error between the states predicted by the vehicle model f from (4.39) and the states measurements from the true system. The true system in this case refers to either a high fidelity simulator like CarSim or a physical experiment, depending on which system was used to produce the reference trajectory.

To compute the state error, we first propagate the dynamics for the entire duration of the trajectory using the vehicle model in (4.39), as shown in equation (4.53), where T_s is the time step between consecutive state/input pairs:

$$\hat{z}_{i+1} = \hat{z}_i + T_s f(\hat{z}_i, \bar{u}_i) \tag{4.53a}$$

$$\hat{z}_0 = \bar{z}_0 \tag{4.53b}$$

Then we calculate error $\mathbf{e}$ between the predicted state $\hat{\mathbf{z}}$ and the reference state $\mathbf{z}^{\text{ref}}$ as shown below:

$$e_i = \hat{z}_i - \bar{z}_i \tag{4.54}$$

Online segment

The online controller performs two actions: (a) search for the nearest reference trajectory point, and (b) apply either an open-loop or closed-loop input command. For the first part, at each time step, the controller finds the 'nearest' reference position $\bar{z}_i$ by performing the optimization routine below:

$$\underset{\bar{z}_i \in \{\bar{z}_i\}_{i=0}^{T}}{\arg\min} \|z_t - \bar{z}_i\|_2 \tag{4.55}$$

where z_t is the current position estimate.

Next the online controller applies either a closed-loop or open-loop input. To decide among the two, the controller propagates the vehicle model (4.39) forward in time for n steps starting from the current state estimate z_t. The controller propagates the dynamics twice, once using an open-loop input sequence and once using a closed-loop one, adding the state error terms at each time step, as shown below:

$$\hat{\bar{z}}_{i+1} = \hat{\bar{z}}_i + T_s f(\hat{\bar{z}}_i, u_i) + e_{i+k} \tag{4.56a}$$

$$u_i = \begin{cases} \bar{u}_{i+k} + K_{i+k}(\hat{\bar{z}}_i - \bar{z}_{i+k}) & : \text{CL} \\ \bar{u}_{i+k} & : \text{OL} \end{cases} \tag{4.56b}$$

$$\hat{\bar{z}}_0 = z_t \tag{4.56c}$$

The state errors terms $\mathbf{e}$ computed from equation (4.53) are added in equation (4.56) to capture model mismatch between the vehicle model and the plant, especially during drifting. The controller selects the input sequence that results in a smaller tracking error, expressed in terms of the cost function below:

$$J = \sum_{k=0}^{n} (\hat{\bar{z}}_k - \bar{z}_{i+k})^\top Q (\hat{\bar{z}}_k - \bar{z}_{i+k}) \tag{4.57}$$

In general, the controller selects a closed-loop command in a well-modeled region and selects an open-loop maneuver in poorly-modeled regions (i.e. drifting).

Experimental Results

The mixed control strategy was tested on the BARC platform with the following physical parameters.

The experiments were conducted in an indoor space with the RC vehicle set to the same initial position for each test. We began by running a series of open-loop tests using the rule-based algorithm, recording the inputs during each test. After the tests, we selected a drift-cornering trajectory that didn't collide with the track boundaries, and set it as the reference trajectory. Next, we ran another two sets of experiments, one using the proposed algorithm, the other using the recorded inputs from the reference trajectory.

Table 4.8: RC-car parameters

Parameter	Value	Parameter	Value
m [kg]	1.95	L_r [m]	0.125
I_z [kg $\cdot$ m^2]	0.24	C_f [N/rad]	1.76
L_f [m]	0.125	C_r [N/rad]	1.76

The blue paths in Figure 4.17 and Figure 4.18 show the reference trajectories. The green paths in Figure 4.17 are the experimental results when the control commands are exactly the same with inputs of reference trajectory. The green paths in Figure 4.18 are the results when the proposed mixed open-loop and closed-loop controller is applied.
These results show that by using the pre-recorded open-loop control commands, the vehicle fails to track the reference trajectory. By using the proposed mixed open-loop and closed-loop strategy, the vehicle can repeatedly track the reference path. The vehicle recovers from drifting reference errors during the straight segment of the track.
We tested the repeatability of the proposed control algorithm by tracking the same reference trajectory multiple times. The experimental result shows that the proposed control algorithm is applicable in practice by using only low-cost sensors. A video of the experiment can be found at barc-project.com.

4.5 Conclusion

This chapter explored planning and control algorithms for steady state drifting, drift parking, and drift cornering. This chapter start by discussing the control design for steady state drift. The strategy centers on using a drift equilibrium reference state to design an LQR control policy. The closed-loop policy initiates after a basic open-loop maneuver brings the vehicle into the region of attraction. The second part of this chapter focused on the planning and control of drift parking and drift cornering. The section discussed the using the ideas of sample-based path planning and mixed open-loop closed-loop control as a complete framework to perform autonomous drift maneuvers. These ideas presented to control each one of these maneuvers was verified from experimental results using the BARC platform.

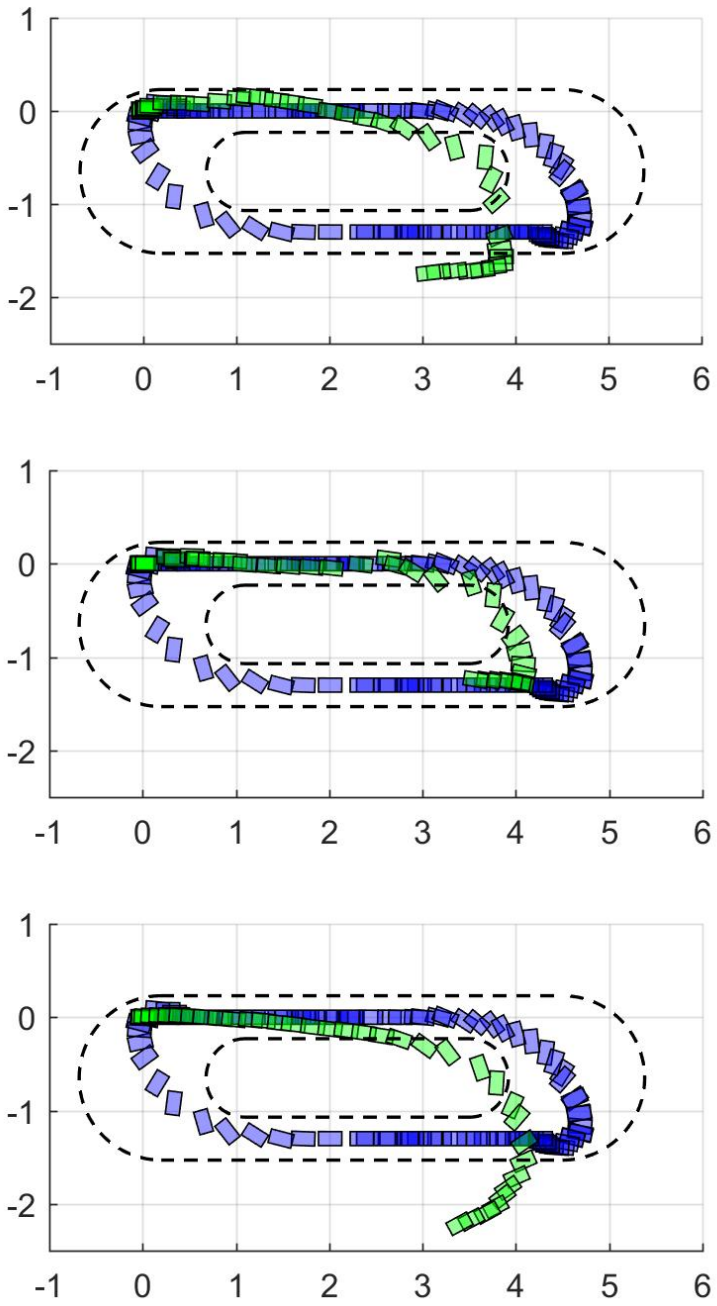


Figure 4.17: Pure open loop control for the entire duration of the experiment results in poor tracking behavior.

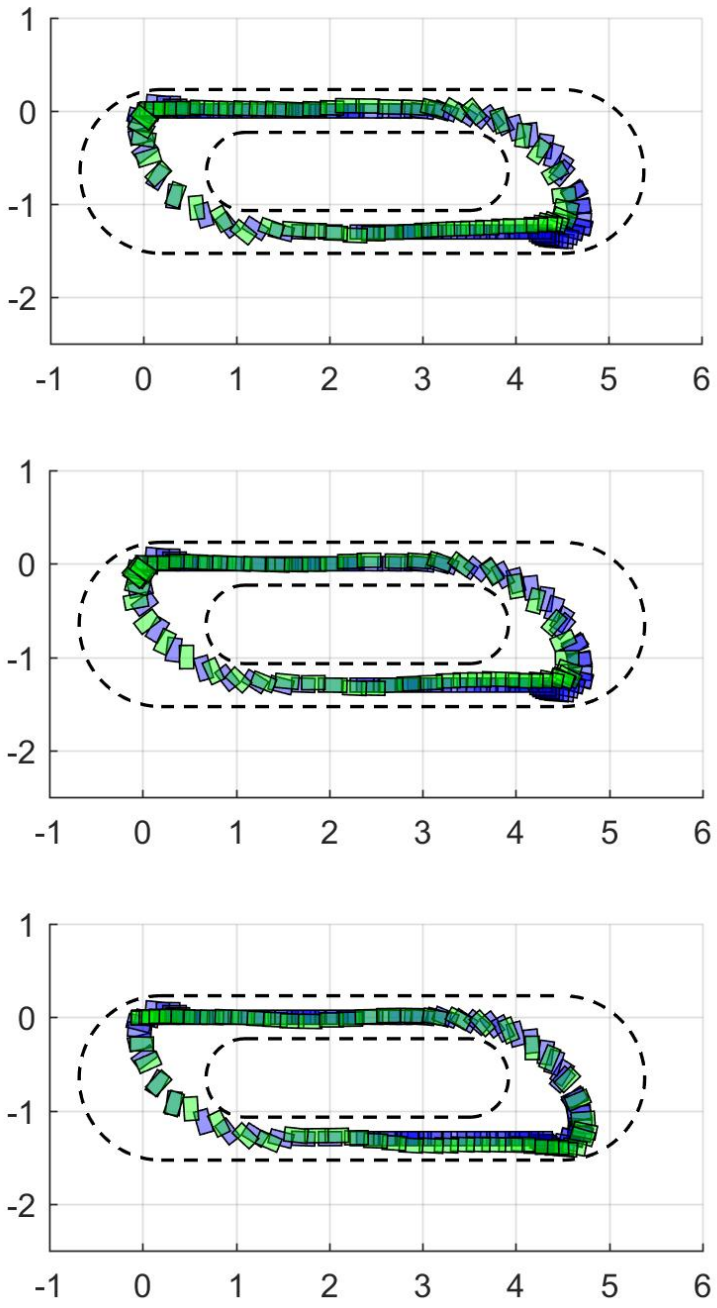


Figure 4.18: The mixed open-loop closed-loop control strategy consistently tracks the reference trajectory.

Chapter 5

Conclusion

The dissertation investigated planning and control of drift maneuvers and outlined the development of a robotic platform for research and instruction in autonomous driving. For the topics of drift parking and drift cornering, we discussed a path planning strategy and then a mixed open-loop closed-loop control scheme for conducting transient, complex drift maneuvers. The control strategy uses mixed open-loop and closed-loop control, based on the observation that vehicle systems behave deterministically over a short duration when operating in open-loop (i.e. fixed sequence of input commands) from approximately the same initial condition, even if the dynamics are complex and difficult to model.

Looking forward, some of the ideas presented in this dissertation remain open for further investigation. The results from the mixed open-loop closed-loop framework work that illustrate the method can work in practice, but it would be interesting to develop theoretical guarantees about the stability of the controller using tools like reachability analysis. A similar extension could be to provide theoretical guarantees about switching between a collection of open-loop controllers and closed-loop controllers that control various maneuvers. For the work on the region of attraction, we only computed a boundary that accounted for input saturation, but it would be interesting to see how nonlinear mapping for system inputs affect the stability of the LQR controller.

Another direction for future work lies in the continued development of small-scale vehicle platforms like the BARC. The BARC aims to be accessible to those interested in autonomous driving and uses much of the same type of hardware and software in full-scale autonomous vehicles. More effort can be placed into organizing the software structure so that the transition to a full-scale platform is easier.

Appendix A

Equilibrium Analysis

The following MATLAB program performs state analysis on a with using a three state dynamic bicycle model

```
% drift-equilibrium analysis
% @author: jon gonzales
clc;
close all;
clear all;

%% define system parameters
% vehicle model
m       = 1.98;         % mass
Iz      = 0.24;         % moment of inertia about z-axis
vx      = 1.2;          % longitudinal velocity
Lf      = 0.125;        % distance from CoG to front axel
Lr      = 0.125;        % distance from CoG to rear axel
g       = 9.81;         % gravity
mu      = 0.234;        % coefficient of friction

% tire model parameters (Pacejka model)
B           = 7.4;
C           = 1.2;
D           = -m*g*mu/2;

%% grid steering angle
df_max          = 20;
df_eq         = (-df_max:1:df_max)*pi/180;
[~,nAngles]     = size(df_eq);

%% define storage variables
```

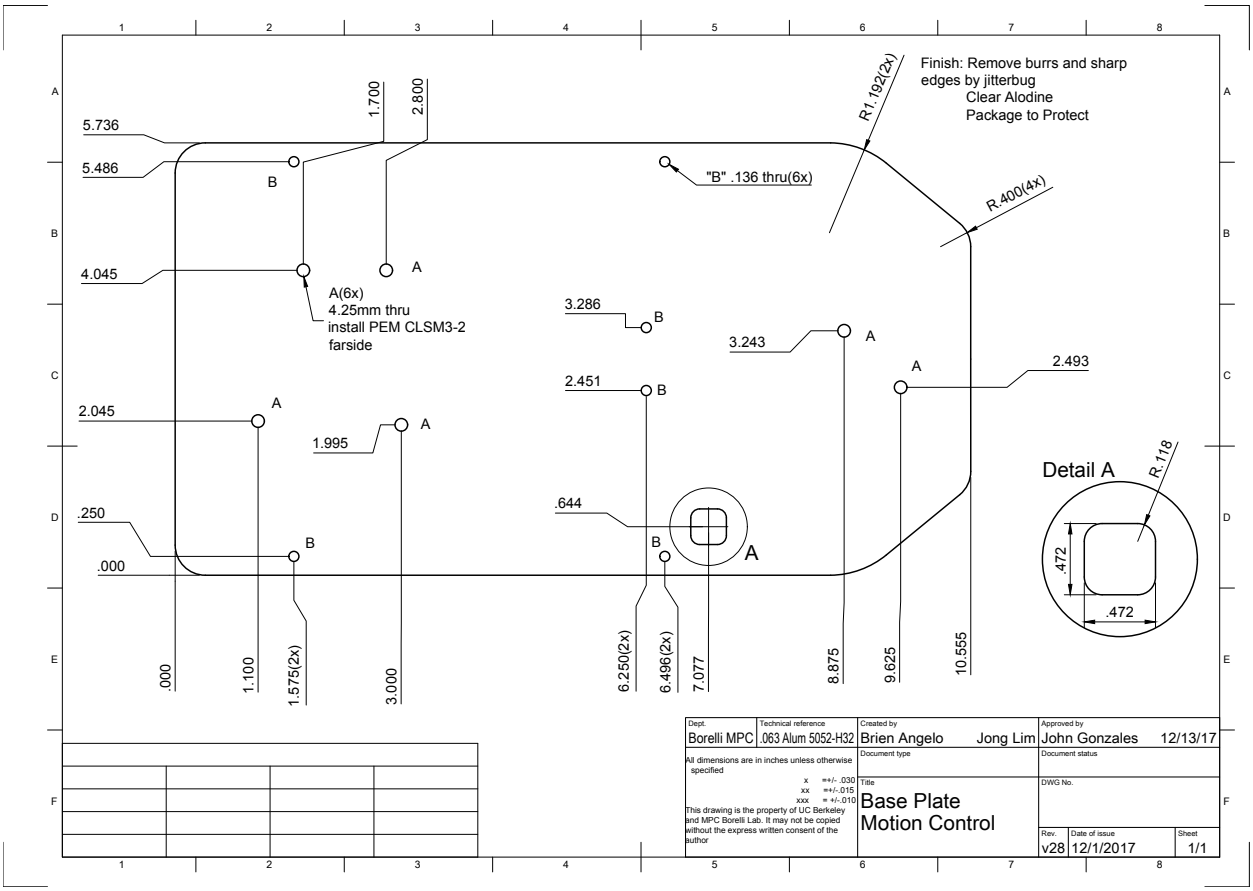
```
28 bta_eq       = zeros(3,nAngles);
29 r_eq         = zeros(3,nAngles);
30 Ffy_eq       = zeros(3,nAngles);
31 Frx_eq       = zeros(3,nAngles);
32 Fry_eq       = zeros(3,nAngles);
33
34 %% set operating modes
35 mode     = {'cornering','drift-left','drift-right'};
36 nModes      = numel(mode);
37
38 %% get eq values
39 for i = 1:nModes
40     for j = 1:nAngles
41         % define symbolic variables
42         syms bta r Frx
43         df = df_eq(j);
44
45         % compute slip angles
46         af = atan(bta + Lf*r/vx)- df;
47         ar = atan(bta - Lr*r/vx);
48
49         % compute front lateral force
50         Ffy = D*sin(C*atan(B*af));
51
52         % compute rear latearl force
53         if strcmp(mode{i},'cornering')
54            Fry = D*sin(C*atan(B*ar));
55         end
56         if strcmp(mode{i},'drift-left')
57             Fry = sqrt(D^2 - (Frx)^2);
58         end
59         if strcmp(mode{i},'drift-right')
60             Fry = -sqrt(D^2 - (Frx)^2);
61         end
62
63         % define system equations
64         dbta     = (Ffy*cos(df) + Fry)/(m*vx) - r;
65         dr       = (Lf*Ffy*cos(df) - Lr*Fry)/Iz;
66         dvx      = 1/m*(Frx - Ffy*sin(df)) + r *vx*bta;
67
68         % solve for equilibria
69         sol              = vpasolve([dbta dr dvx],[bta r Frx]);
```

```
70          bta_eq(i,j)      = sol.bta;
71          r_eq(i,j)        = sol.r;
72          Frx_eq(i,j)      = sol.Frx;
73
74          % compute front lateral force
75          af               = atan(sol.bta + Lf*sol.r/vx)- df;
76          Ffy_eq(i,j) = D*sin(C*atan(B*af));
77
78          % compute rear latearl force
79          if strcmp(mode{i},'cornering')
80              ar           = atan(sol.bta - Lr*sol.r/vx);
81              Fry_eq(i,j) = D*sin(C*atan(B*ar));
82          end
83          if strcmp(mode{i},'drift-left')
84              Fry_eq(i,j) = sqrt(D^2 - (sol.Frx)^2);
85          end
86          if strcmp(mode{i},'drift-right')
87              Fry_eq(i,j) = -sqrt(D^2 - (sol.Frx)^2);
88          end
89      end
90  end
```

Appendix B

Schematics

The following schematic gives the technical drawing for the 'top deck' plate that mounts on top of the Traxxas chassis



Bibliography

[1] Christopher G Atkeson and Stefan Schaal. "Learning tasks from a single demonstration". In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. Vol. 2. IEEE. 1997, pp. 1706–1712.

[2] Ashwin Carvalho. "Predictive Control under Uncertainty for Safe Autonomous Driving: Integrating Data-Driven Forecasts with Control Design". PhD thesis. University of California, Berkeley, 2016.

[3] Imon Chakraborty, Panagiotis Tsiotras, and Jianbo Lu. "Vehicle posture control through aggressive maneuvering for mitigation of T-bone collisions". In: *IEEE Conference on Decision and Control*. 2011.

[4] COGNITEAM. *Hamster Micro AUGV*. 2015. URL: `http://www.cogniteam.com/assets/docs/hamster/hamster4_brochure.pdf`.

[5] Mark Johnson Cutler. "Reinforcement learning for robots through efficient simulator sampling". PhD thesis. Massachusetts Institute of Technology, 2015.

[6] Sandeep Dhameja. "1.3 Introduction to Electric Vehicle Batteries". In: *Electric Vehicle Battery Systems*. Elsevier, 2002. ISBN: 978-0-7506-9916-7. URL: `https://app.knovel.com/hotlink/khtml/id:kt00BJ04I1/electric-vehicle-battery/introduction-electric`.

[7] Johannes Edelmann and Manfred Plöchl. "Handling characteristics and stability of the steady-state powerslide motion of an automobile". In: *Regular and Chaotic Dynamics* 14.6 (2009), p. 682.

[8] Penn Engineering. *F1/10 platform*. [Online; accessed June 1, 2018]. 2017. URL: `http://f1tenth.org/index`.

[9] Paolo Falcone et al. "Predictive active steering control for autonomous vehicle systems". In: *IEEE Transactions on control systems technology* 15.3 (2007), pp. 566–580.

[10] David J. Fleet and Yair Weiss. *Handbook of Mathematical Models in Computer Vision*. Springer, 2006, pp. 628–647.

[11] Massimo Guiggiani. *The science of vehicle dynamics: handling, braking, and ride of road and race cars*. Springer Science & Business Media, 2014.

[12] HardKernel. *myAHRS+*. [Online; accessed June 1, 2018]. 2015. URL: https://www.hardkernel.com/main/products/prdt_info.php?g_code=G141464363369.

[13] HardKernel. *Odroid XU-4 CPU/RAM PERFORMANCE*. [Online; accessed June 1, 2018]. 2015. URL: https://www.hardkernel.com/main/products/prdt_info.php9.

[14] Rami Y Hindiyeh and J Christian Gerdes. "A Controller Framework for Autonomous Drifting: Design, Stability, and Experimental Validation". In: *Journal of Dynamic Systems, Measurement, and Control* 136.5 (2014), p. 051015.

[15] Rami Yusef Hindiyeh. "Dynamics and Control of Drifting in Automobiles". PhD thesis. Stanford University, 2013.

[16] Jessica K Hodgins and Marc H Raibert. "Biped Gymnastics". In: *The International Journal of Robotics Research* 9.2 (1990), pp. 115–128.

[17] HorizonHobby. *EC2 Device Connector*. URL: https://www.horizonhobby.com/ec2-device-connector-%5C%282%5C%29-eflaec201 (visited on 04/26/2018).

[18] HorizonHobby. *EC3 Device Connector*. URL: https://www.horizonhobby.com/ec3-battery-connector-%5C%282%5C%29-eflaec302 (visited on 04/26/2018).

[19] HorizonHobby. *EC5 Device Connector*. URL: https://www.horizonhobby.com/ec5-battery-connector-%5C%282%5C%29-eflaec502 (visited on 04/26/2018).

[20] School of Interactive Computing at the Georgia Institute of Technology. *Autorally platform*. [Online; accessed June 1, 2018]. 2018. URL: https://autorally.github.io/.

[21] Matt Jardin. *Improving Mass Moment of Inertia Measurements*. 2010. URL: https://www.mathworks.com/company/newsletters/articles/improving-mass-moment-of-inertia-measurements.html (visited on 01/30/2016).

[22] Jean-Jacques. *Dirk Stratton - Drift-Vette*. [Online; accessed June 1, 2018]. 2016. URL: https://www.flickr.com/photos/zenzak35/27497438246/in/photostream/.

[23] J Zico Kolter et al. "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 839–845.

[24] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE. 2015, pp. 1094–1099.

[25] Alexander Liniger, Alexander Domahidi, and Manfred Morari. "Optimization-based autonomous racing of 1: 43 scale RC cars". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.

[26] *Magni*. [Online; accessed June 1, 2018]. 2018. URL: https://robots.ros.org/magni/.

[27] MIT. *RACECAR - A Powerful Platform for Robotics Research and Teaching*. 2015. URL: https://mit-racecar.github.io/ (visited on 06/01/2018).

[28] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[29] Giovanni Palmieri et al. "A robust lateral vehicle dynamics control". In: *10th International Symposium on Advanced Vehicle Control* (2010).

[30] Karl Popp and Werner Schiehlen. *Ground vehicle dynamics*. Springer Science & Business Media, 2010.

[31] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[32] Guinness World Records. *Tightest parallel parking record beaten at new Mini launch – Guinness World Records*. [Online; accessed June 1, 2018]. 2012. URL: https://www.youtube.com/watch?v=q3BGkOKVMUU.

[33] *ROSbot*. [Online; accessed June 1, 2018]. 2018. URL: https://robots.ros.org/rosbot/.

[34] Will Roscoe. *Donkey Car*. [Online; accessed June 1, 2018]. 2017. URL: http://www.donkeycar.com/.

[35] DC Shoes. *Ken Block's Gymkhana THREE, Part 2; Ultimate Playground*. [Online; accessed June 1, 2018]. 2010. URL: https://www.youtube.com/watch?v=4TshFWSsrn8.

[36] SparkFun. *4mm Supra X Gold Bullet Connectors*. URL: https://hobbyking.com/en_us/4mm-supra-x-gold-bullet-connectors-10-pairs.html (visited on 04/26/2018).

[37] SparkFun. *Deans Connector - M/F Pair*. URL: https://www.sparkfun.com/products/11864 (visited on 04/26/2018).

[38] SparkFun. *XT60 Connectors - Male/Female Pair*. URL: https://www.sparkfun.com/products/10474 (visited on 04/26/2018).

[39] Tamiya. *Powerpole Connectors - PP15 to PP45 : up to 55 Amps*. URL: https://www.greatplanes.com/miscproducts/connectors.php (visited on 05/01/2018).

[40] Davide Tavernini et al. "The optimality of the handbrake cornering technique". In: *Journal of Dynamic Systems, Measurement, and Control* (2014).

[41] Traxxas. *Traxxas High-Current Connector*. URL: https://traxxas.com/products/parts/accessories/highcurrentconnector?t=overview (visited on 04/26/2018).

[42] *TurtleBot*. [Online; accessed June 1, 2018]. 2018. URL: https://www.turtlebot.com/about/.

[43] E Velenis and P Tsiotras. "Minimum time vs maximum exit velocity path optimization during cornering". In: *2005 IEEE international symposium on industrial electronics*. 2005, pp. 355–360.

[44] Efstathios Velenis, Panagiotis Tsiotras, and Jianbo Lu. "Optimality properties and driver input parameterization for trail-braking cornering". In: *European Journal of Control* 14.4 (2008), pp. 308–320.

[45] Efstathios Velenis et al. "Steady-state drifting stabilization of RWD vehicles". In: *Control Engineering Practice* 19.11 (2011), pp. 1363–1376.

[46] John Warner. "3.3 Battery Terms". In: *Handbook of Lithium-Ion Battery Pack Design - Chemistry, Components, Types and Terminology*. Elsevier, 2015. ISBN: 978-0-12-801456-1. URL: https://app.knovel.com/hotlink/khtml/id:kt00UCJJU6/handbook-lithium-ion/battery-terms.

[47] Bruce Wootton. *Dator - Cloud Data For Robots*. [Online; accessed June 1, 2018]. 2015. URL: https://github.com/MPC-Berkeley/barc/tree/master/Dator.