ANALYSIS OF VARIOUS ADAPTIVE CRUISE CONTROLLERS VIA EXPERIMENTAL IMPLEMENTATION

A Dissertation

by

AAKAR MEHRA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

Chair of Committee,	Aaron Ames
Committee Members,	Pilwon Hur
	Shankar P. Bhattachryya
Head of Department,	Andreas A. Polycarpou

August 2015

Major Subject: Mechanical Engineering

ABSTRACT

Adaptive cruise control (ACC) testing requires minimum of two cars and a platform where the two cars can be tested for a continuous time. Here a custom-built platform and software are presented for testing various ACC algorithms on scaled model cars. There are multiple techniques being studied for driver convenience and safety automation systems for production vehicles: electronic stability control, adaptive cruise control, lane keeping, and obstacle avoidance. Presented here are some novel control framework that gives formal guarantees of correctness that go beyond traditional PID-based controllers for ACC that do not, inherently, have proofs that satisfy. In the first approach, safety constraints – maintaining a valid following distance from a lead car are represented by control barrier functions (CBFs), and control objectives – achieve a desired speed – are encoded through control Lyapunov functions (CLFs). While the same safety constraints are formulated using Linear Temporal Logic (LTL) for synthesizing the control software module using abstraction based controllers in the second approach. In the longer run, each interacting software module is endowed with specifications, under certain environment assumptions, the module is guaranteed to meet its specifications.

For the CBF-CLF approach, the different objectives can be unified through a quadratic program (QP), with constraints dictated by CBFs and CLFs that balance safety and control objectives in an optimal fashion. Similarly for the abstraction controllers, PESSOA and Polyhedral Control Invariant Set approaches are correct-by-construction. The end result was the experimental demonstration of these methodologies on scale-model cars, for which the CBF-CLF and abstraction based controllers were implemented in real-time.

DEDICATION

I dedicate my thesis and research to my family and my friends who have supported me since the beginning not only in studies but almost everything else. Also to all the people who drive a vehicle and consider automated driver assistance as a potential future technology in cars.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Aaron Ames for his continuous guidance, motivating push and support through my research work. His enthusiasm towards experimental verification of control theory has been encouraging for me to learn and implement this thesis. Along with my advisor, I'd like to appreciate support provided by other professors and students on the Cyber Physical Project who coordinated along with this research, specially Dr. Jessy Grizzle (University of Michigan) and Dr. Paulo Tabuada (University of California- Los Angeles). I would also like to thank my colleagues from AMBER Lab: Forrest Berg, Wen-Loong Ma, Eric Cousineau, Jake Raher and Huihua Zhao, for sharing valuable technical ideas with me and motivating me during the times when it was hard to come up with a good solution.

I would like to thank my parents and their continuous support for my studies and allowing me to come to the US for my higher studies. My friends here and back in India all have been extremely supportive for my decisions in life. Last but very important, I would like to extend my sincere gratitude towards my advisory committee to guide me through my masters program. It was their courses that inspired my interest in control theory and robotics to flourish during my research time.

NOMENCLATURE

ACC	Adaptive Cruise Control
CPS	Cyber Physical System
NHTSA	National Highway and Traffic System Administration
CCC	Conventional Cruise Control
ICC	Intelligent Cruise Control
CLF	Control Lyapunov Functions
CBF	Control Barrier Functions
RCBF	Reciprocal Control Barrier Functions
ZCBF	Zeroing Control Barrier Functions
PCIS	Polyhedral Controlled Invariant Set
PATH	Program of Advanced Technology for the Highway
QP	Quadratic Program
ABS	Anit-lock Braking System
MPC	Model Predictive Control
ROBDD	Reduced Ordered Binary Decision Diagram
BLDC	Brush Less Direct Current
LIPO	Lithium Ion-Polymer
PWM	Pulse Width Modulations
ROS	Robotic Operating System
ECU	Electronic Control Unit
PID	Proportional Integrator Differential

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION *

Automobiles are good examples of complex Cyber-Physical Systems (CPS) due to the tight coupling between the physical world (car/road dynamics) and the several layers of hardware and software used for control purposes. Recent efforts by automakers to increase the level of automation in cars bring a new sense of urgency to the difficult problem of formally verifying these systems. Adaptive cruise control (ACC) is being developed and deployed on passenger vehicles [17] due to its promise to enhance driver convenience, safety, traffic flow, and fuel economy [26], [27], [40]. ACC is a multifaceted control problem because it involves asymptotic performance objectives (driving at a desired speed) is, subject to safety constraints (maintaining a safe distance from the lead car), and has constraints based on the physical characteristics of the car and road surface (maximum acceleration and deceleration). This control problem is made more challenging by the fact that the various objectives can often be in conflict, such as when the desired speed is faster than the speed of the leading car. Provably satisfying the safety-oriented constraints is of paramount importance.

According to a 2008 survey conducted by the National Highway Traffic Safety Administration [13], there were 10.2 million car crashes, out of which 9.48 million were due to human error, i.e., 93% of all the car crashes in the U.S. are caused by mistakes made by the driver. Although these numbers recently have decreased (possibly due to stricter laws), technology has yet to find satisfactory solutions to preventing accidents. These numbers have motivated a significant amount of research

*Portions of this thesis have been reprinted with permission from Adaptive Cruise Control: Experimental Validation of Advanced Controllers on Scale-Model Cars by A. Mehra, W.L. Ma, P. Tabuada, J. Grizzle and A. D. Ames.

in utilizing onboard sensing, computation and control to assist human drivers; some examples include: cruise control, Anti-lock Braking Systems (ABS), traction control, obstacle avoidance, improved traffic flow and fuel economy [26], [27].

Conventional cruise control [39] (CCC) has been successfully implemented in almost all cars in the United States, yet it does not actively take into account collision avoidance. Adaptive cruise control (ACC), which aims to unify CCC with safety-related constraints [31], is being actively studied from a variety of perspectives [25],[30], [34]. Mitsubishi was the first company to start the concept of ACC in 1995, designing the Preview Distance Control, a laser-based approach, to match the velocity of the vehicle to its immediate predecessor.

A variety of solutions have been proposed since then (see the survey paper [41]). The most relevant to the approach taken here is based on Model Predictive Control, which is natural in the ACC setting due to the existence of multiple control objectives [25], [30]. As a means to experimentally validate advanced automotive controllers like ACC, previous research by the Program of Advanced Technology for the Highway (PATH) focused on creating platooning between vehicles on the highways. Intelligent cruise control (ICC) is a variant of ACC which prioritizes autonomous driving by designing controllers with braking systems that require minimal manual interference explored by a lot of literature, such as [21], [17]. Controlled braking systems that allow a vehicle to perform emergency stops when necessary, then return to the set point velocity was investigated by [15].

Even though the use of barrier functions unified with Lyapunov functions provides a novel approach to ACC using quadratic programs [29], they have been used in a other fields of study, [32], [20], [12] and [24]. A recent approach to handle the verification problem is to synthesize control software using correct-by-design methods. These are techniques that synthesize both, the control software as well as a

proof of its correctness, so that a-posteriori verification is not required. All these techniques were implemented on the custom-built platform for this thesis.

1.1 Control Techniques

Two major approaches were analyzed experimentally to solve the ACC problem, Lyapunov-like controllers using barrier certificates and correct-by-design control using formal methods. In the following sections, each approach is introduced along with multiple techniques used in each method. It is important to realize that both the approaches have not been implemented on an actual physical system in any literature and was the first time that these algorithms were practically verified.

1.1.1 Control Lyapunov Function and Control Barrier Functions

The simulated and experimental validation of an optimized controller mathematically accounts for both the safety and the comfort of the driver. The safety-critical nature of the problem necessitates controllers that are formally correct, i.e., give guarantees of safety. As a means to address the issue of multiple constraints, [6], [31], controllers were presented that give proofs of safety while simultaneously achieving speed related control objectives. In particular, safety constraints were formulated as CBFs and speed regulation related control objectives were encoded as CLFs; these representations allowed for the formulation of a quadratic program (QP) that dynamically balanced these potentially conflicting specifications.

Two different types of CBFs were analyzed experimentally. Reciprocal control barrier function (RCBF) uses an inequality that satisfied the safety constraints. The inequality was converted into a *barrier function*, $B = \frac{1}{h(x)}$, which allowed the function to grow to infinity as it approached the boundary. Based on a similar concept, a Zeroing control barrier function is also derived from the same inequality. This function is of the form $B = h(x)$ and was restricted to stay positive for all time.

1.1.2 Formal Methods

ACC systems have been designed using various methods such as MPC [30] and sliding mode control. A comprehensive survey on ACC systems can be found in a paper on research on ICC systems [41]. Although these techniques provide controllers enforcing the specification, there is no guarantee that the actual implementation of MPC based controllers is correct. Unlike MPC and Sliding mode control approaches, a controller was synthesize as well as its software implementation in a correct-by-design manner so that no verification was required.

In [31], a controller was synthesized using correct-by-design control software for ACC while assuming a lead vehicle operating at a constant speed. Two different methods for the synthesis of control software were used. The first method reformulates the problem as the computation of reachable sets directly on the continuous state space, while the second method used finite-state abstractions. The resulting correct-by-design controllers were deployed on scale model cars.

1.2 Implementation Method

Mathematical simulation is not sufficient to provide proofs of real world behavior, i.e. they must be experimentally verified to validate the theory. Therefore, a novel experimental platform is introduced utilizing scale model cars (see Fig. 2.1) to test the various ACC controllers. This platform was custom built for the purpose of testing formal results before translating them to full scale realization. The controllers were implemented in real-time on an autonomous following car while the lead car was manually controlled so as to simulate realistic driving conditions. The end result was the experimental validation of online optimization based controllers for ACC (videos can be found at links [1],[1]).

2. SYSTEM DYNAMICS AND MODELLING

A physical system that gives an approximate model, if not completely accurate, is required to develop control theories. Commercially available vehicle has multiple degrees of freedom, therefore, modelling the complex Cyber-Physical system is extremely difficult. Even if an approximate model can be derived from the equations of motion, the environmental factors (roads, wind, tire pressure etc) were inherent hidden variables. To develop the proposed control theories, the system was simplified to get an acceptable model for the required ACC problem.

The vehicle was modeled as a (lumped) point mass moving in a straight line with the steering fixed in one position. A free body diagram is shown in Fig. 2.1 which results in the following equations of motion.

$$m\frac{dv}{dt} = F_w - F_r \tag{2.1}$$

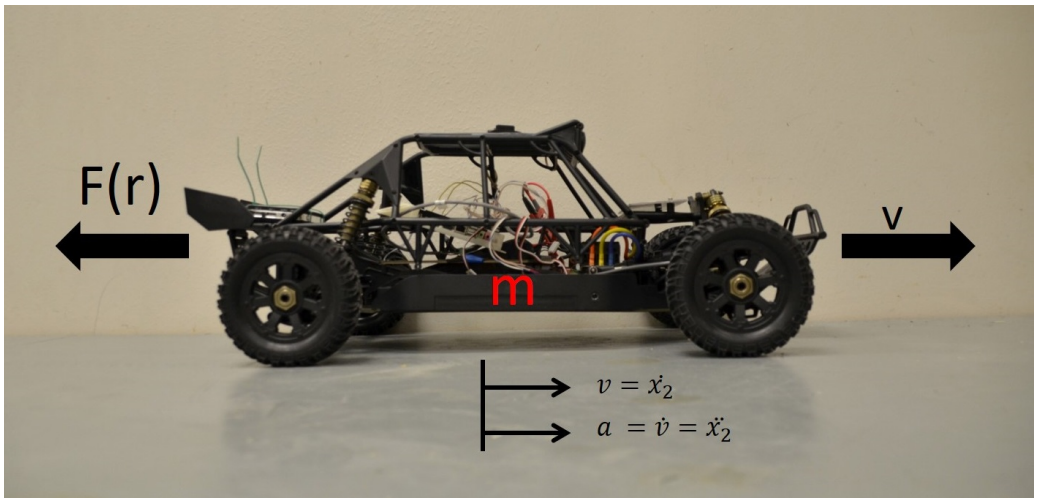


Figure 2.1: Dynamics on a free body diagram of a vehicle.

where m and v are the mass and velocity of the car, F_w is the force generated by the contact of the wheels with the road, and

$$F_r = f_0 - f_1 v - f_2 v^2. \tag{2.2}$$

is the total resistive force acting on the vehicle, in which f_0, f_1 and f_2 are various coefficients of friction forces that can be calculated empirically. For this specific research all the constant were determined by analysis on actual cars and fitting the data points using higher order polynomials equations. All parameters used are listed in 2.1. These parameters were scaled according to the ratio of the scale model car during implementation.

Furthermore, the distance D between the car and the lead car is specified by the equation:

$$\frac{d}{dt}D = v_0 - v \tag{2.3}$$

where v_0 and v are the velocities of the lead and controlled car, respectively. In this case the constant lead car velocity was assumed. In the future sections, this equation is used for demonstration of the feasibility of controllers for the given model. Note that the velocity of the lead car was considered both constant, v_0, and time varying, $v_l(t)$. Experimentally, $v_l(t)$ will governed by the user manually controlled the lead car, and sensed through the boom encoder. The equation then merely changed to,

$$\frac{d}{dt}D = v_l(t) - v \tag{2.4}$$

2.1 State Space Representation

Development of the control theory system required the simplistic state space form. By defining $x = (x_1, x_2)$ with x_1 the position of the vehicle, x_2 the velocity and $z = D$ to be the distance between the two cars, the governing equations was converted to a nonlinear ODE:

$$\dot{x} = \underbrace{\begin{bmatrix} x_2 \\ -F_r/m \end{bmatrix}}_{f(x,z)} + \underbrace{\begin{bmatrix} 0 \\ 1/m \end{bmatrix}}_{g(x,z)} u \tag{2.5}$$

$$\dot{z} = \underbrace{v_l - x_2}_{q(x,z,t)}, \tag{2.6}$$

where $u = F_w$, the control input. The state space equations now represent an affine nonlinear system, with $f(x, z)$ as the dynamics based only on states and $g(x, z)$ as the constant which was multiplied by the control input, u. The output of this affine system was considered to be $h(x) = x_2$.

Parameter	value	Unit
g	9.81	kg/s^2
m	9.07	kg
f_0	0.1	N
f_1	5	$N \cdot s/m$
f_2	0.25	$N \cdot s^2/m$
v_0	3	kg/s
ϵ	10	–
γ	10^{-4}	–
c_a	0.8	–
c_d	1.2	–
p_{sc}	10^5	–
p_{cc}	10^{10}	–

Table 2.1: Parameters Used in Simulation and Experiments

2.1.1 ACC Control Constraints

The goal was to validate the requirements of ACC, including: collision avoidance, adaptive velocity control, and driving comfort constraints.Three classes of constraints, hard soft and comfort, will form the basis for the development of an advanced online-optimization based controller for the ACC problem.

Hard Constraints: The constraint with the highest priority prevented the following vehicle from colliding with the lead car. This constraint should never be violated under any circumstance. Considering the simple rule stated in [43]: the minimum distance between two cars must be "half the speedometer", which is represented mathematically as:

$$D \geq \frac{v}{2} \tag{HC1}$$

Soft Constraints: As the standard objective of cruise control, the controller should be able to track a specified desired speed v_d when adequate headway is assured. In other words:

$$\text{Drive} \qquad v - v_d \to 0 \tag{SC1}$$

Comfort Constraint: While satisfying hard and soft constraints, it is of necessity to reduce the peak forces generated by the car in emergency situations. That is, prevent sudden accelerations so that the driver can experience a comfortable ride if at all possible. This can be achieved by constraining the acceleration and the deceleration of the vehicle as an inequality constraint:

$$-c_d g \leq \frac{F_w}{m} \leq c_a g. \tag{CC1}$$

where c_d and c_a were the factors of g for deceleration and acceleration, respectively.

2.2 Hybrid Model Representation

The ACC problem was also represented as a hybrid system with two different modes and reset maps switching between the two modes [31]. To include a lead vehicle in the system description, we use a hybrid system model was used with two discrete modes M_1 and M_2, called `no lead car` and `lead car mode`, respectively. The `lead car mode` M_2 has an additional continuous state h which measures the headway to the lead car. The continuous dynamics of M_1 are those of (2.1) while the continuous dynamics in mode M_2 contain an additional equation describing the dynamics of the headway given by (2.1) and (2.4). The admissible velocity for the controlled car is bounded by the set $\mathcal{V} = [v^{min}, v^{max}]$ with $v^{min} \geq 0$.

The two modes have different state spaces, mode M_1 has a state space $\mathcal{V}$ while mode M_2 has the state space $\mathcal{V} \times \mathcal{H}$, where $\mathcal{H} = [0, h^{max}]$ is the bounded by the maximum range of the radar. Along with the model for the following car, the lead car was modeled in the following way,

$$\dot{v_l} = a_l \tag{2.7}$$

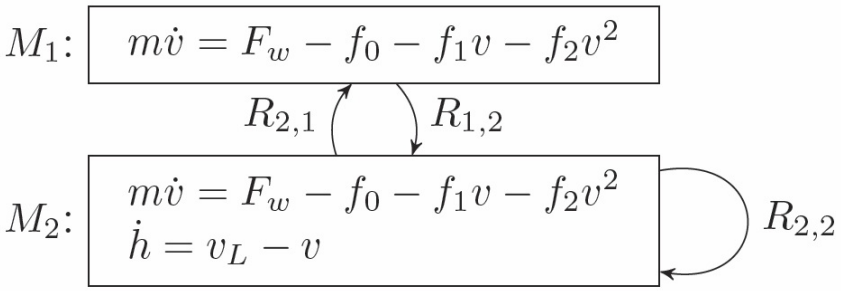


Figure 2.2: Hybrid system model.

Where a_l was the acceleration of the lead car, bounded by the set $a_l \in [a_l^{min}, v_l^{max}]$ and the velocity by $\mathcal{V}_l = [0, v_l^{max}]$.

In practice, the system was in M_2 if there was a car within the radar range, and in M_1 otherwise. Switching between the two states was governed by lane changes of lead cars, which were modeled using reset maps $R_{1,2} : \mathcal{V} \to 2^{\mathcal{V}\times\mathcal{H}}, R^{2,1} : \mathcal{V} \times \mathcal{H} \to \mathcal{V}$ and $R^{2,2} : \mathcal{V} \times \mathcal{H} \to 2^{\mathcal{V}\times\mathcal{H}}$

$$\begin{aligned} R_{1,2}(v,h) &= \{(v,\bar{h}) : \bar{h} \in \mathcal{H}\}, \\ R_{2,1}(v,h) &= \{v\}, \\ R_{2,2}(v,h) &= \{(v,\bar{h}) : \bar{h} \in \mathcal{H}\}. \end{aligned} \tag{2.8}$$

Here $R_{1,2}$ modeled a transition from the `no lead car` mode M_1 to the `lead car` mode M_2, where the headway was initialized to some $\bar{h} \in \mathcal{H}$. Similarly, $R_{2,2}$ models situations where the radar reading suddenly changes as a result of lane changes undertaken by cars in front. The hybrid model can be visualized in the Fig. 2.2.

For reasons of comfort, the force F_w generated by the ACC software to the range was limited

$$F_w^{min} \leq F_w \leq F_w^{max} \tag{2.9}$$

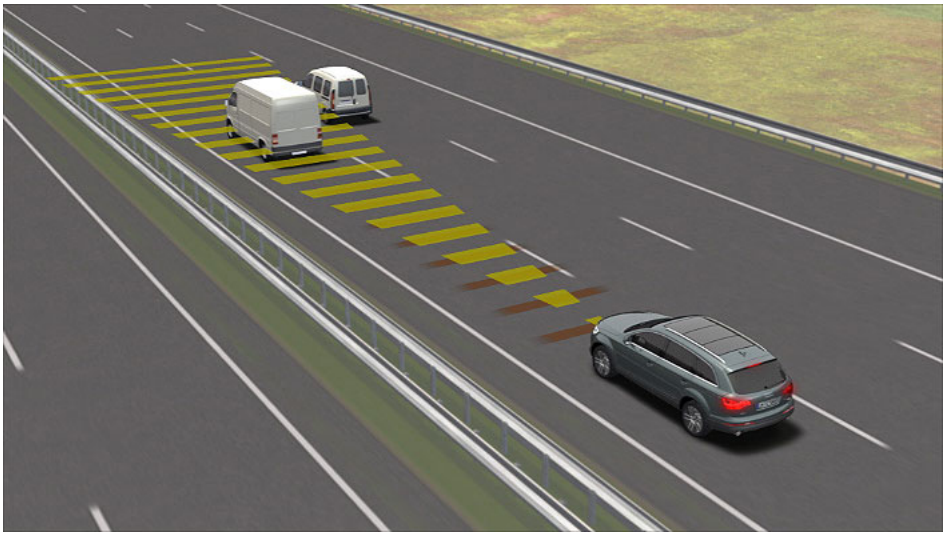


Figure 2.3: A figure depicting the distance measurement by the sensors on the controlled car. (*Image Credits: Audi Motors*)

2.2.1 ACC Formal Specification

In this section the adaptive cruise control requirements are formalized using Linear Temporal Logic (LTL). Introducing the *time headway*, defined as $\tau = h/v$, the requirements given by the International Organization of Standardization, [22, Chapter 6], are summarized as follows.

1. The input constraint (2.9) needs to be satisfied at all times;
2. A lower bound on the time headway τ_{min} needs to be satisfied at all times;
3. A desired lower bound on time headway τ_{des} to the lead vehicle and a desired upper bound v_{des} for the velocity should be eventually reached and maintained henceforth.

The above requirements were then represented in the form of sets. First, the set S_U was introduced which defines the input constraint and S which defined the minimum

headway constraint as follows

$$S_U := [F_w^{\min},\ F_w^{\max}], \tag{2.10}$$

$$S := \{(v, h, v_L) \in \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L : h/v \geq \tau_{min}\}\,. \tag{2.11}$$

The goal set G was then presented to express the last requirement of the standard. The set G represented all states that satisfy the desired time headway and the desired velocity upper bound. The system should reach this set eventually and stay in it forever

$$G := \{(v, h, v_L) \in \mathcal{V} \times \mathcal{H} \times \mathcal{V}_L : h/v \geq \tau_{des}, v \leq v_{des}\}. \tag{2.12}$$

The ACC specification was expressed using the atomic propositions S_U, S and G, the propositional logic conjunction "$\wedge$", and the temporal operators always "$\Box$" and eventually "$\Diamond$". The LTL formulas were interpreted over infinite sequences (ξ, ν) where the signal $\xi : \mathbb{N} \to \mathbb{R}^3$ is a sample-and-hold trajectory of (2.1), (2.2) and (2.7) given the input signal $\nu : \mathbb{N} \to \mathbb{R}$ generated by the ACC. These sequences were the behavior of the closed-loop system.

A behavior (ξ, ν) was said to satisfy $\Box p$ or $\Diamond p$ if the atomic proposition p is true at all times or eventually at some time, respectively. A closed-loop system satisfies an LTL formula φ if every behavior, i.e., (ξ, ν), satisfies φ. (Detailed explanation of the syntax and semantics of LTL can be found in [42].)

The ACC specification can be described by the LTL formula ψ:

$$\Box S_U \wedge \Box S \wedge \Diamond \Box G. \tag{2.13}$$

The first and second terms guarantee input and time headway τ^{min} constraints are satisfied all the time, while the third term ensures that the system will eventually satisfy, and maintain, the lower bound on time headway τ^{des} and the upper bound on velocity v^{des}. While (2.13) does not describe all the requirements in [22], it already illustrates some of the difficulties arising on the synthesis of controllers enforcing multiple objectives and constraints.

2.3 Objectives Derived from the Specifications

Given correct-by-design controllers enforcing the LTL specification (2.13) on the model (2.1), (2.2) and (2.7) and the state space based model, (2.5), the objectives were:

1. To implement these controllers on the experimental platform described in the following sections;

2. To compare the theoretical guarantees of these controllers against simulation and experimental results;

3. To discuss the ease of implementing these controllers on the experimental platform.

3. PROPOSED CONTROL THEORIES

As presented in the previous chapters, this thesis takes four different control methods in consideration for experimental testing. Before we get into the details of implementation on the hardware, it is important to see the construction of each controller individually. All the approaches have been presented, discussed and analyzed using basic simulation results to show the feasibility of solutions.

The first two techniques use a combination of Lyapunov functions and barrier functions to develop two advanced control methods for the ACC problem. Both of them use the same Lyapunov function but they vary in the style of the barrier function used, which allows the change in construction of the controllers. The last two techniques use the formal methods approach using linear temporal logic. Each of them uses the same goal set, while using two different ways to solve the problem. Each of these control theories are discussed in detail in the following sections. At this point it is also important to mention that this is an experimental based thesis and not theory based, so while discussing the controllers, detailed mathematical proofs will not be included here. Merely the results of the theorems will be used to show the construction of the controllers.

3.1 Multiple Objective Optimizing using Quadratic Programs

This section will focus on the construction of a Lypanov-like controller using a reciprocal barrier function. To design a controller that provably enforces the Hard Constraint (HC1), it is natural to utilize control barrier functions (CBFs) to ensure that this constraint is satisfied for all time, [4], [5]. In particular, by converting units

to m and s, the hard constraint (HC1) can be restated as:

$$h(x, z) = z - 1.8x_2 \geq 0, \tag{3.1}$$

which yields the admissible set $\mathcal{C}$ given by:

$$\mathcal{C} = \{(x, z) \in \mathbb{R}^3 : h(x, z) \geq 0\}, \tag{3.2}$$

$$\partial\mathcal{C} = \{(x, z) \in \mathbb{R}^3 : h(x, z) = 0\}, \tag{3.3}$$

$$\text{Int}(\mathcal{C}) = \{(x, z) \in \mathbb{R}^3 : h(x, z) > 0\}. \tag{3.4}$$

As previously stated, the Lyapunov function candidate, for tracking the desired velocity is the same. The chosen control Lyapunov function candidate and the setup of the quadratic program to balance between the multiple objectives in a unified fashion is discussed. Before getting into experimental implementation and analysis, verification of the feasibility of solution for both constant and time varying lead car velocity is done by simulating the system in MATLAB.

3.1.1 Reciprocal Control Barrier Function

Barrier functions – first utilized in optimization [11] – are now common throughout the control and verification literature due to their natural relationship with Lyapunov-like functions [38], [44],[45]. This thesis uses a novel form of barrier function, B, which was introduced in [6], associated with a set, $\mathcal{C}$, i.e., $B(x) \to \infty$ as $x \to \partial\mathcal{C}$, and proves that if B satisfies Lyapunov-like conditions, then forward invariance of $\mathcal{C}$ is guaranteed. Existing formulations of barrier functions assume invariant level sets of B [23], i.e., $\dot{B} \leq 0$, whereas here B is allowed to grow when it is far way

from the boundary of $\mathcal{C}$, i.e., it is only require that

$$\dot{B} \leq \frac{\gamma}{B} \tag{3.5}$$

where γ can be positive. This inequality relaxes the set of solutions possible for B contained within $\mathcal{C}$ with the relation that $\dot{B}$ will grow when the solutions are far away from the boundary of $\mathcal{C}$. And as the solution approaches the boundary, the rate of growth slows.

Based on the motivation provided above, now there is a need to formulate a barrier function that gives the same guarantees. Firstly, the basic definition of a barrier function as used in [6],

Definition 1: For the dynamical system (2.5), a function $B : \mathcal{C} \subset \mathbb{R}^n \to \mathbb{R}$ is a **reciprocal barrier function (BF)** for he set $\mathcal{C}$ defined by (3.2)-(3.4) for a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ if there exist a locally Lipschitz class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ such that, for all $x \in Int(\mathcal{C})$,

$$\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))} \tag{3.6}$$

$$\dot{B}(x) \leq \frac{1}{\alpha_3(h(x))} \tag{3.7}$$

The condition (3.6) essentially shows that the barrier function will be bounded by two functions of the form $\frac{1}{\alpha(h(x))}$ such that as the solutions reach the boundary, it grows to infinity.

Using the above definition and the notion of the construction of a CLF, as seen in [7], a definition for reciprocal control barrier functions is constructed and presented as,

Definition 2: Let $\mathcal{C} \subset \mathbb{R}^n$ be defined as above for a continuously differentiable

$h : \mathbb{R}^n \to \mathbb{R}$, then a function $B : \mathcal{C} \to \mathbb{R}$ is a **reciprocal control barrier function (RCBF)** if there exist class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \gamma > 0$ such that,

$$\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))} \tag{3.8}$$

$$\inf_{u \in U}[L_f B(x) + L_g B(x) u - \frac{\gamma}{B(x)}] \leq 0 \tag{3.9}$$

for all $x \in Int(\mathcal{C})$. Taking the problem at hand, lets choose the CBF candidate as:

$$B\big(h(x,z)\big) = B(x,z) = \frac{1}{z - 1.8x_2}, \tag{3.10}$$

with associated derivative:

$$\dot{B}(x,z,t,u) = \underbrace{-\frac{1.8F_r + m(v_l(t) - x_2)}{m(z - 1.8x_2)^2}}_{L_f B} + \underbrace{\frac{1.8}{m(z - 1.8x_2)^2}}_{L_g B} u. \tag{3.11}$$

Based on the definition 2 given above along with the fact that $(x, z) \in Int(\mathcal{C})$, the provided B is a valid RCBF if it satisfies (3.5), which leads to

$$\inf_{u \in U} \Big[L_f B(x,z) + L_g B(x,z) u - \frac{\gamma}{B(x,z)}\Big] \leq 0. \tag{HC1-RCBF}$$

3.1.2 Zeroing Control Barrier Function

As the previous subsection defines the reciprocal barrier function, on similar grounds and motivation, now a zeroing barrier function is defined. Considering the same set $\mathcal{C}$ defined by (3.2)-(3.4) to provide the following definition, as seen in [46].

Definition 3: For the dynamical system (2.5) and the set $\mathcal{C}$, a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$; if there exist a locally Lipschitz class $\mathcal{K}$ functions

α_1 and a set $\mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$such that,

$$L_f h(x) \geq -\alpha(h(x)), \forall x \in \mathcal{D} \tag{3.12}$$

then the function h is called a **zeroing barrier function (ZBF)** From this definition, analogous to the control Lyapunov functions, the zeroing control barrier functions can be defined.

Definition 4: Let $\mathcal{C} \subset \mathbb{R}^n$ be defined as above for a continuously differentiable $h : \mathbb{R}^n \to \mathbb{R}$, the function h is called a **zeroing control barrier function (ZCBF)** if there exist class $\mathcal{K}$ functions α such that,

$$\inf_{u \in U} [L_f h(x, z) + L_g h(x, z) u + \alpha h(x, z)] \geq 0, \forall x \in \mathcal{D} \tag{HC1-ZCBF}$$

3.1.3 Control Lyapunov Function

This section used the mathematical methodology used in [6] to build the soft constraints (see (SC1)) based on control Lyapunov functions [4], [8],[9], [16]. To track a desired velocity the control law should drive

$$y(x, z) = x_2 - v_d \to 0. \tag{SC1}$$

For this relative 1 degree output, we choose the Lyapunov function candidate:

$$V(y) = y^2 \tag{3.13}$$

which yields

$$\dot{V}(y) = \underbrace{-\frac{2y}{m} F_r}_{L_f V} + \underbrace{\frac{2y}{m}}_{L_g V} u. \tag{3.14}$$

According to definition 3 in [7], since $V(y)$ satisfies $c_1\|y\|^2 \leq V(y) \leq c_2\|y\|^2$, $V(y)$ is a valid exponentially stabilizing control Lyapunov function (ES-CLF) if

$$\inf_{u\in U}[L_f V(y) + L_g V(y)u + \epsilon V(y)] \leq 0 \tag{3.15}$$

is satisfied. In other words, with a proper choice of control input u, the output $y(x, z)$ will be exponentially driven to zero, thereby enforcing velocity tracking. However, this function needs to be converted into constraints on the (x, z). By defining

$$\begin{aligned}\psi_0(x,z) &= -\frac{2(x_2 - v_d)}{m}F_r + \epsilon(x_2 - v_d)^2 \\ \psi_1(x,z) &= \frac{2(x_2 - v_d)}{m}\end{aligned} \tag{3.16}$$

we can then construct the CLF constraint:

$$\psi_0 + \psi_1 u \leq \delta_{sc}, \tag{SC1-CLF}$$

where δ_{sc} is a relaxation factor. Note that it is this relaxation factor that makes the constraint a soft constraint.

Along with the soft constraint, the bounds on forces are also balanced in the quadratic program. Note that since the comfort constraint is also a conditional constraint and it directly acts on the control input, by modifying (CC1) by adding the relaxation factor δ_{cc}:

$$\begin{aligned}u &\leq c_a mg + \delta_{cc} \\ -u &\leq c_d mg + \delta_{cc}\end{aligned} \tag{CC}$$

3.1.4 Quadratic Programs

To unify all the constraints defined according to the ACC problem, an online quadratic program (QP) based controller will provably satisfy the requirements, [8], [29]. To construct a cost function To construct a cost function for the QP, notions from feedback linearization [35] to develop a cost that will favor convergence to the control objective (achieving a desired speed) are utilized. In particular, a control input that satisfies (3.14) is given by:

$$u = \frac{1}{L_g y}(-L_f y + \mu) = F_r + m\mu \tag{3.17}$$

where μ is the control input for the linearized output dynamics. To minimize the control effort μ, the cost function of QP is chosen as:

$$\mu^T \mu = \frac{1}{m^2}(u^T u - 2u^T F_r + F_r^2). \tag{3.18}$$

By combing the above constraints the ACC CLF-CBF based QP control law is given by:

$$u^*(x,z) = \underset{\mathbf{u}=\begin{bmatrix} u \\ \delta_{sc} \\ \delta_{cc} \end{bmatrix} \in \mathbb{R}^3}{\text{argmin}} \quad \frac{1}{2}\mathbf{u}^T H_{\text{acc}} \mathbf{u} + F_{\text{acc}}^T \mathbf{u} \tag{ACC QP}$$

$$s.t. \quad A_{\text{clf}}\mathbf{u} \leq B_{\text{clf}} \tag{CLF}$$

$$A_{\text{cbf}}\mathbf{u} \leq B_{\text{cbf}} \tag{BCF}$$

$$A_{\text{cc}}\mathbf{u} \leq B_{\text{cc}} \tag{CC}$$

In which,

$$H_{\text{acc}} = 2\begin{bmatrix} \frac{1}{m^2} & 0 & 0 \\ 0 & p_{sc} & 0 \\ 0 & 0 & p_{cc} \end{bmatrix}, \quad F_{\text{acc}} = -2\begin{bmatrix} \frac{F_r}{m^2} \\ 0 \\ 0 \end{bmatrix} \tag{3.19}$$

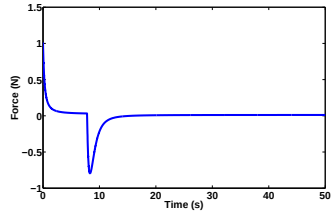
and $A_{\text{clf}}, B_{\text{clf}}$ and $A_{\text{cbf}}, B_{\text{cbf}}$ are the inequality constraints obtained from (HC1-RCBF), (SC1-CLF):

$$\begin{aligned} A_{\text{clf}} &= \begin{bmatrix} \psi_1(x,z) & -1 & 0 \end{bmatrix}, \\ B_{\text{clf}} &= -\psi_0, \\ A_{\text{cbf}} &= \begin{bmatrix} L_g B(x,z) & 0 & 0 \end{bmatrix}, \\ B_{\text{cbf}} &= -L_f B(x,z) + \frac{\gamma}{B(x,z)}. \end{aligned}$$

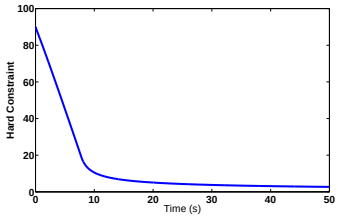
$$A_{\text{cc}} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix}, \qquad B_{\text{cc}} = \begin{bmatrix} c_a mg \\ c_d mg \end{bmatrix}.$$

where p_{cc} is the user-defined penalty for the relaxation. Importantly, because as higher priority has to be given to comfortable driving experience over velocity regulation, it is necessary to set $p_{sc} \ll p_{cc}$, where p_{sc} is the penalty for violating the soft constraint. The matrices A_{cbf} and B_{cbf} are constructed using the two different barriers discussed in the previous subsections. This also shows the ease in interchanging the matrices from the two controllers $A_{rcbf}, A_{zcbf}, B_{rcbf}$ and B_{zcbf}, while implementation.

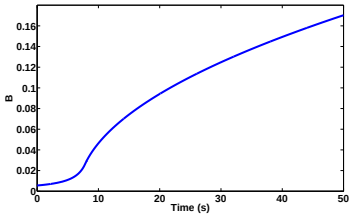
Note that while the output of the control law is a direct input to the dynamic system for the simulation, for the experimental setup the actual input to the system



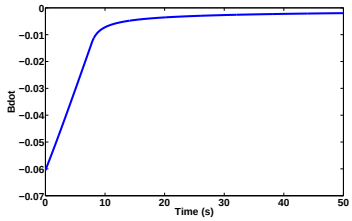
(a) Force Input (N)



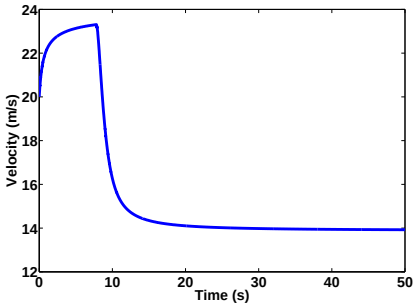
(b) Hard Constraint



(c) B



(d) $\dot{B}$



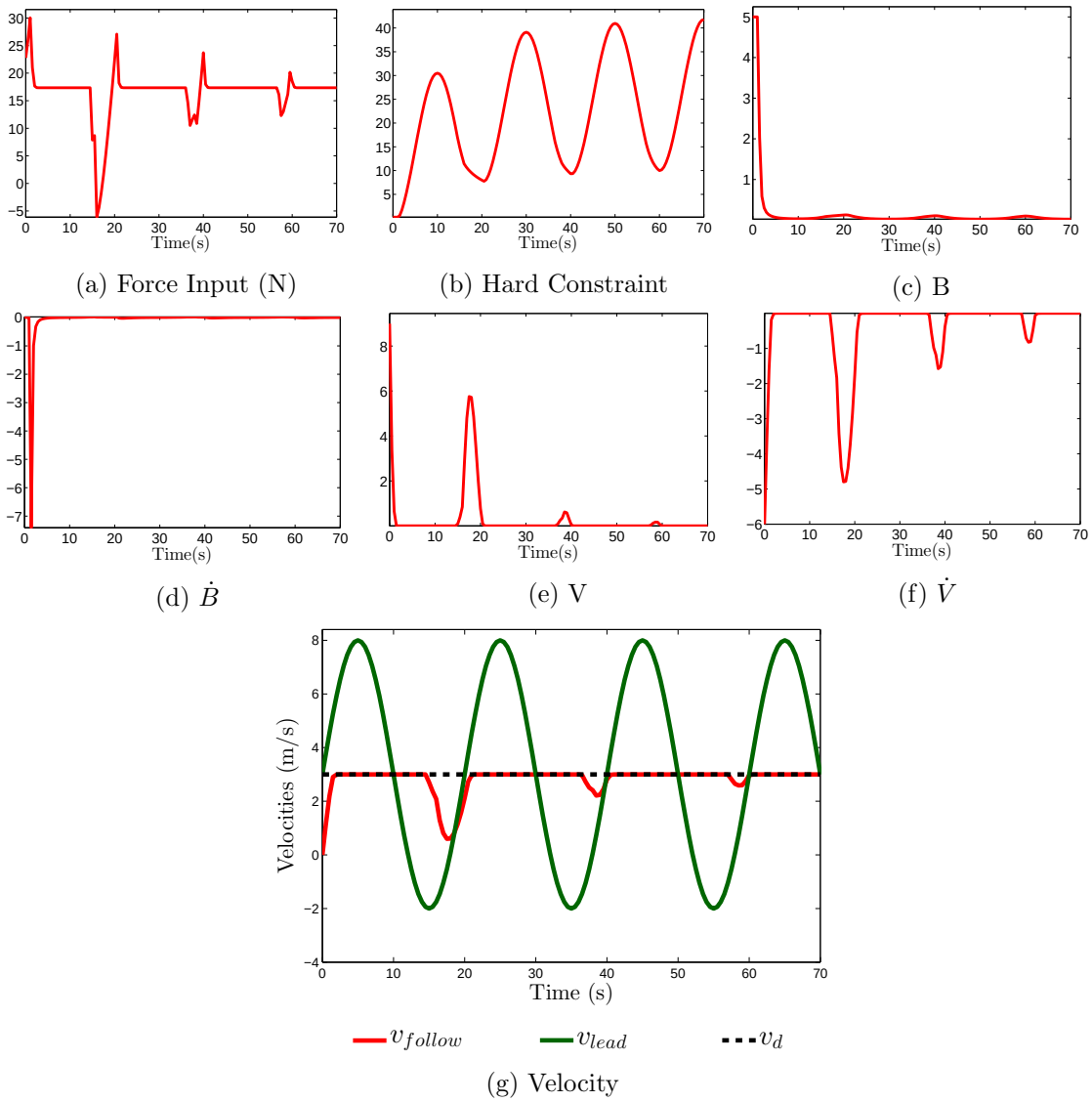
(e) Velocity

Figure 3.1: Simulation results for the case when the lead car has a constant velocity using RCBF.

is the PWM pulse to the motor. To best mirror the control framework on physical experiments, we integrate the output of the QP (ACC QP) one step forward by using the dynamics of the system to find the internal velocity via:

$$v_{\mathrm{u}} = v_{\mathrm{previous}} + (u - F_r)t_{\mathrm{loop}} \tag{3.20}$$

where t_{loop} is the control period (loop rate)in experiment. The end result is a control input for the nonlinear dynamics that will guarantee the safety hard constraint and



(a) Force Input (N) (b) Hard Constraint (c) B

(d) $\dot{B}$ (e) V (f) $\dot{V}$

(g) Velocity

Figure 3.2: Simulation results for the case when the lead car has a sinusoidal velocity profile using RCBF.

adaptively use minimum effort to adjust the velocity of the vehicle for both good comfort and tracking performances. The results of the simulation verification are shown in FIG. For the time varying simulation, the following sine wave is considered: $3 + 5sin(0.1\pi t)$. The constant velocity case is simulated for RCBF and the time varying for the ZCBF to test both types of barrier function candidates.

3.1.5 Simulation Results

To demonstrate the validity of this construction, the CLF-CBF QP controller (ACC QP) was implemented in an ideal environment: simulation. First results are for a constant velocity profile for the lead car and second for which the lead car is moving at a sinusoid velocity as given by $v_l(t) = 3 + 5\sin(0.1\pi t)$.

As shown in Fig. 3.1 the variables clearly show that when the following car achieves the desired speed ($23m/s$) and it gets close to the lead car, as seen in the hard constraint. This activates the barrier and we can see that the speed reduces to that of the lead car which is fixed at $14m/s$. More importantly, it is seen that the barrier function always stays positive and the derivative of the barrier is always negative.

Fig. 3.2 shows the control objectives have been achieved for a time varying velocity, i.e., safety is always maintained while the desired speed is achieved whenever possible. In particular, when the system starts from $(x, z) \in \mathcal{C}$, since it is close to $\partial\mathcal{C}$, the hard constraint (Fig. 3.2b) activates the CBF to take effect on the following car and thereby modulate its speed. Therefore, with a high value of $B(x, z)$ and $\dot{B}(x, z, u, t)$, the following car moves much slower than the lead car to stay away from the safety imposed barrier. When the hard constraint grows, i.e, the relative distance is within a safe range, the CLF constraint will dominate the QP controller and yield a desired cruise velocity regulation. Therefore, as the lead car was moving forward and backward, which occasionally caused the CBF to slow down the following car, the growing relative distance will eventually disable the hard constraint and leave the CLF controlling the velocities with bounded accelerations. The simulation results thereby verify the validity of the proposed controller in simulation.

3.2 Formal Control Methods

This section provides a very brief discussion on the technique behind the controller synthesis for two different formal methods. As this is an experimental thesis, the details of the mathematical proofs are not provided here. The first technique that is discussed here is based on finite-state abstraction using a tool called PESSOA. The controller synthesis for this was done by students and professors at the University of California Los Angeles. The second controller uses a linearized plant and solves the ACC problem on a continuous state space using polyhedral controlled invariant sets (PCIS). This theory was developed by the students and professors of University of Michigan. The two approaches used to synthesize correct-by-design controllers for the ACC problem are explained in the following subsection along with some graphical analysis.

3.2.1 Solution by PCIS Computation

An outline of how the synthesis problem defined in the previous section can be solved by PCIS computations. The problem is an example of a *reach-stay-avoid* problem, in the sense that the specification dictates that a *goal set* should be reached and kept invariant, while avoiding an *unsafe* set. To solve a single reach-stay-avoid problem, the approach followed is initiated in [31]:

1. Linearize and integrate the dynamics to obtain a discrete-time affine system.
2. Employ reachability computations for discrete-time affine systems to reason about polyhedral set invariance and reachability.
3. Implement a control strategy based on the polyhedral sets obtained in Step 2.

In this thesis the main focus is only on implementation issues.

3.2.1.1 Linearization and Integration

To obtain a discrete-time affine system, the dynamics (2.5) of the following car are linearized around a nominal velocity $\bar{v}$. For the final closed-loop system to be correct, correctness is required in the linearized system to imply correctness in the original system (2.1) and (2.4). To achieve this, the inputs of the linearized system are constrained by an amount corresponding to the maximal linearization error in the domain. This allows the original system to mimic the performance of the linearized system while still keeping the total control within the allowed bounds. For typical vehicle parameters, the conservativeness introduced by the linear approximation is small [31].

The second step is to integrate the linearized continuous-time dynamics for a time step ΔT. Selecting an appropriate time step constitutes a trade-off between performance and computational complexity. On one hand, the closed-loop system will be provably correct only at discrete instants separated by ΔT. By bounding how much the continuous-time system can deviate from its discrete-time counterpart as in [18], correctness guarantees can be obtained also in continuous time at the cost of conservativeness (which increases with ΔT). On the other hand, a smaller ΔT will increase the number of iterations needed for the controlled-invariant set algorithms to converge (or be ϵ-close to its point of convergence).

3.2.1.2 Set Computation and Implementation

Given an affine discrete-time system and a reach-stay-avoid problem defined by a goal set G and an unsafe set U (in this case U is the complement of the safe set S), first a controlled-invariant set $C_0 \subset G$ contained inside the goal set is computed. Essentially, employing polyhedron algorithms from [10, 14] modified with novel techniques for addressing disturbances, with bounds both on the disturbance state itself

and its rate of variation, to accommodate both $\mathcal{V}_l$ and $\dashv_l$. Such assumptions make the maximal controlled-invariant set non-convex in this application, so it cannot be represented by a single convex polyhedron. Therefore the controlled-invariant set is represented as the union of overlapping polyhedra.

Given a controlled-invariant set, subsets of the state space $C_1, C_2, \ldots$ are computed with the property that C_i can be reached from C_{i+1} in time ΔT while avoiding the unsafe set U. These sets, together with the control policy, "when in C_i, go to $C_{\max(0,i-1)}$" then constitute a solution to a reach-stay-avoid problem.

To implement this control strategy, the model predictive control (MPC) is used, which allows us to pose set membership constraints as linear inequalities in a quadratic program (QP) that is solved online. Basically, the choice of weights in the QP selects a single correct control action among the infinite number of correct control actions represented by the sets. Before running the controller, a collection of linear inequalities representing the sets C_i must be loaded into memory. In this case, a time discretization step $\Delta T = 0.1s$ was used which resulted in a controller consisting of 11 sets, each represented by 14 convex polyhedra.

3.2.2 Solution via PESSOA

In this section, the synthesis of a controller that enforces the specification given by (2.13) on system (2.1) and (2.4) is discussed using the MATLAB toolbox PESSOA [23], based on the correct-by-design controller synthesis techniques described in [37].

Controller synthesis is performed on an abstraction Σ of system (2.1) and (2.4), which is computed by discretizing the state space, input space and time. The discretization produces a finite-state transition system $\Sigma = (Q, U, \delta)$, where Q is the set of states, U is the set of inputs, and $\delta : Q \times U \to 2^Q$ is the transition function. The abstraction is constructed so that there exists an approximate alternating simulation

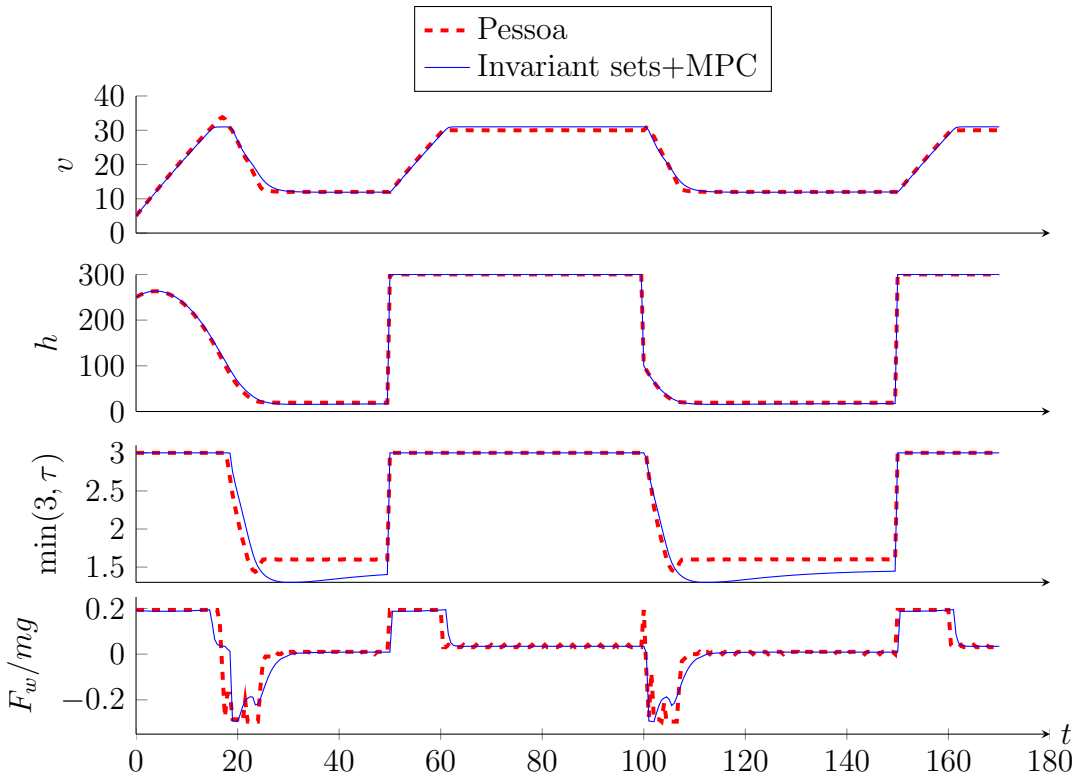


Figure 3.3: Simulation results for the case when the lead car has a constant velocity.

relation [33] from the abstraction to the continuous system. The existence of this simulation relation ensures that controllers synthesized for the abstraction can be refined to controllers enforcing the specification on the continuous system [37].

It is worth mentioning that PESSOA uses reduced ordered binary decision diagrams (ROBDD) [3], a memory-efficient data structure, to store the abstraction as well as the synthesized controller. The implementation of such controllers on a digital platform is straightforward since it amounts to querying the ROBDD at each sampling time.

3.2.3 Controller Synthesis via PCIS vs PESSOA

PESSOA and PCIS are two complementary approaches to synthesize correct-by-design control software. PCIS exploits the linearity of the dynamics and of the atomic propositions (semi-linear sets) to synthesize a control policy directly on the continuous state space. It offers the advantage of not requiring the construct of a finite abstraction, which is the main computational bottleneck in PESSOA. However, since it exploits linearity, it requires the linearization of the nonlinear dynamics. PESSOA can work directly with nonlinear models but it requires the construction of a finite abstraction, a computationally demanding operation. Once such an abstraction is available; however, controllers enforcing arbitrary LTL specifications can be handled and termination of all the relevant algorithms is always guaranteed. When using PCIS, only a smaller class of requirements can be handled and termination is forced whenever the sets being computed do not change much from one iteration to the next. Fig. 3.3 shows the constant lead car velocity results for both the controllers in comparison. The plots prove that the force on the wheels is always bounded and as the distance between the cars reduces, the speed of the controlled reduces as well. An important thing to note is that the time headway (τ) is always positive and does not violate the constraints put on it. These results have been shown only to visually realize the guarantee that the controller is working; they were simulated by students at University of Michigan and University of California Los Angeles.

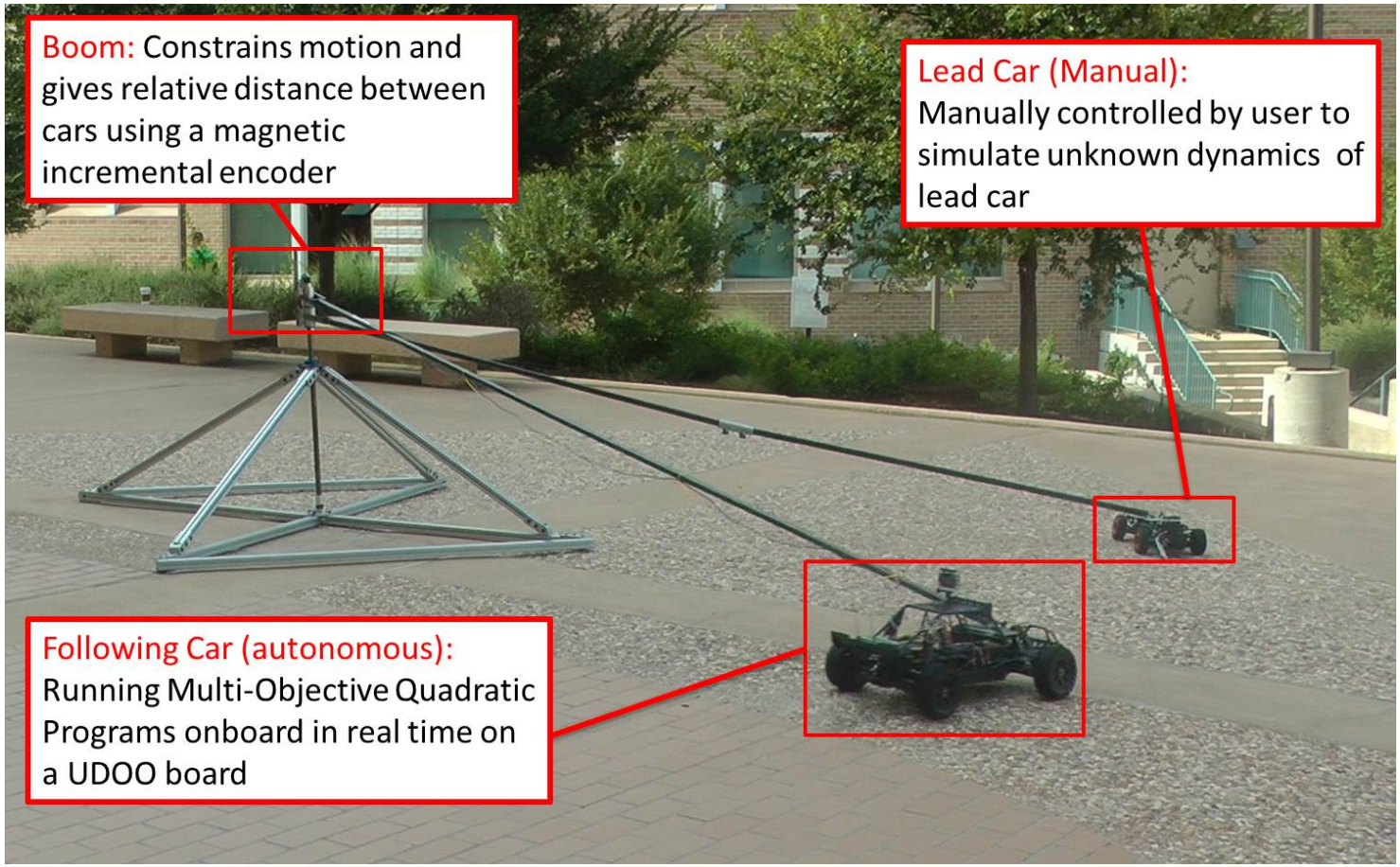


Figure 3.4: Experimental setup with the electric car on the boom.

4. EXPERIMENTAL REALIZATION

As the main result of this thesis, the control theories proposed in the previous chapter have to implemented on a physical experimental platform to verify their feasibility on the hardware. For the implementation, a novel platform was custom-designed specifically to solve the requirements of the ACC problem in Fig. 3.4. In this chapter, the details of the hardware setup along with the electronic setup is discussed, giving details on the software-hardware interfacing.

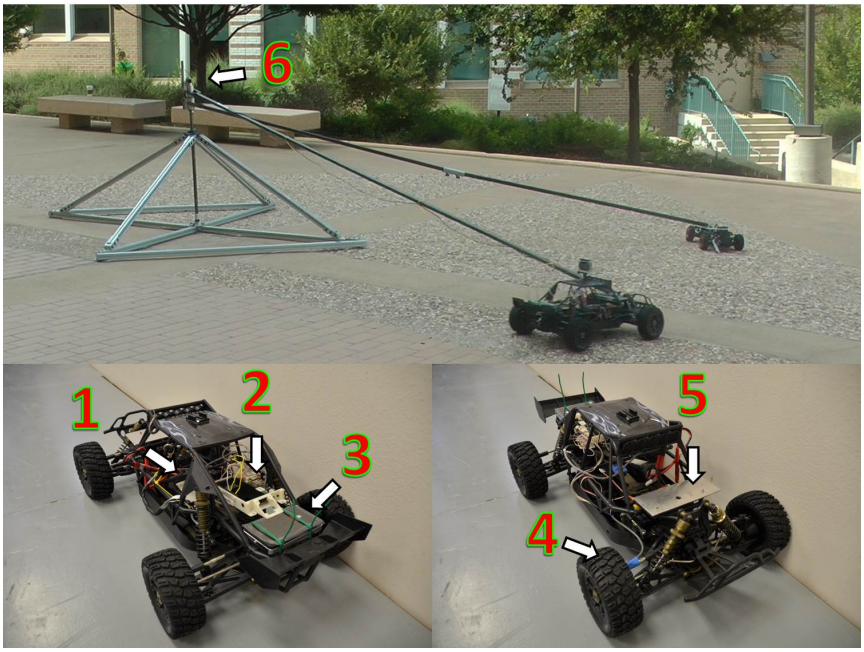


Figure 4.1: Experimental Setup. The boom restricts motion to a circle. As shown in figure: **(1)** Electric motor, **(2)** On-board UDOO **(3)** Battery for the UDOO board, **(4)** Hall sensor and magnets, **(5)** Boom attachment plate, **(6)** Magnetic encoder on the central shaft to measure the relative distance.

4.1 Experimental Hardware Setup

This section begins by discussing the experimental platform that will be used to evaluate formal constructions. This setup is shown in Fig. 3.4 and detailed in Fig. 4.1.

In order to maintain an appropriate balance between realism and complexity, two electric, remote controlled cars powered by brushless DC (BLDC) motors were chosen as the test vehicles for the experiments (see Fig. 4.1). The following car is a all wheel drive, $1/5th$ scaled model and the lead car is a rear wheel drive, $1/8th$ scaled model. The chassis was machined out of aluminum and came equipped with hydraulic shocks. The damping from the shocks was not taken into account in order to keep the simplicity of the overall dynamics. This is an important point to realize as this will create bias in the results due to the lack of proper modeling of the system. The vehicle is powered by a 22.2 V, 5000 mAh Lithium Ion-Polymer(LIPO) battery allowing the vehicle to achieve speeds of more than 10 m/s. The control algorithms running online on the autonomous car, are coded at an embedded level on an electrical development board.

To eliminate lateral motion, both cars are rigidly attached to a central shaft via a boom; see Fig. 3.4. A similar two dimensional setup has already been implemented in several robotic experiments, e.g., in the context of bipedal locomotion [28]. Note that the two cars are attached to their respective booms with a universal ball joint mounted near the front axle in order to ensure self-correction of lateral disturbances. Additionally, the location of the ball joint serves as a steering mechanism, further supporting the assumption of 2D motion of the cars.

4.2 Electronic Setup

In order to validate the proposed CLF-CBF QP, PESSOA and PCIS controllers on the test bed discussed above, hardware-software interface, along with a high level controller running the software that mathematically calculates the control algorithm in real time, is required. In other words, three major requirements for the experimental realization are: sensing, actuation and the feedback system

Sensing. At each control cycle, the current values of the speed of the two cars and the distance between them are read from various sensors. The velocity of the following car is obtained using a Hall effect sensor, which has been mounted on the wheel hub of the front wheel with two small magnets placed 180 degrees apart on the inside of the wheel; see Fig. 4.1. While Lidar and radar are two common devices used for estimating the headway in production cars [43], the special mechanical setup in this case allows us to measure the relative distance between the two cars using a magnetic incremental encoder mounted on the central shaft.

Actuation. The electronics of the cars are designed such that speed is set via a Pulse Width Modulation (PWM) signal to the onboard Electronic Control Unit (ECU), which converts it into a three phase voltage command to the motor. Because direct actuation of wheel torque is not possible, the PWM signal is generated using the velocity command described later in (3.20). The electric car does not have a separate actuator for braking, hence velocity is regulated using only positive wheel force and the combination of rolling resistance and aerodynamic drag forces.

Embedded Computing. The software implementation of the proposed controllers has been divided into two stages: a high-level controller — running on the Robotic Operating System (ROS) in a Linux environment, and a low-level controller running on an Arduino DUE board combined with the ECU on the car.

Algorithm 1 UDOO Module, High Level Controller

Input: Current velocity of controlled car;
Input: Relative distance between the two cars;
Input: Current velocity of the lead car;
Enable ROS Master;
Run ROSSERIAL to communicate with low level;
Connect to remote laptop through SSH;
Enable Electronic Speed Control (ESC) for the car;
repeat
    Wait till all communication is established
until (ESC == Enable)
Set up parameters for the model;
while (ROSSERIAL == Running) **do**
    Define loop rate for high level controller;
    Read ROS messages, Current Velocity and Relative Distance;
    Calculate actual time for the loop using loop rate;
    if Error in Calculation **then**
        Report Errors and Stop QP / ROBDD calculation;
    else
        if Data recieved from any sensor **then**
            Initialize the commanded velocity for the QP / ROBDD;
            Convert relative distance value into relative velocity (m/s);
            Calculate lead car velocity by finite differencing;
            Set up parameters for QP / ROBDD;
            Calculate torque via CBF / PCIS / PESSOA.
            if Solution on boundary of safe set **then**
                Take $v_{\text{comm}} = 0$ to simulate braking on the car;
            else
                Calculate v_{comm} via one-step forward integration;
            end if
            Send velocity data to low level controller
        end if
        Log data onto board via remote laptop over SSH;
    end if
end while
Disable ROS Master;

High-Level Controller: The UDOO board runs Ubuntu 12.10 LTS and ROS Groovy at a sampling rate of 200 Hz. The controllers have been coded in C++ for efficient execution as a ROS node that is also used to record data at all time. The resistive force as mentioned in (2.2) uses average coefficients derived by testing on production cars, so when implementing on scale model cars, the equation is scaled by the same factor as the scale of the car. Considering the RCBF and ZCBF type

Algorithm 2 Arduino Module- Low level

```
Compile Arduino code using IDE;
Communicate with ROSSERIAL node on ROS Master;
Enable Electronic Speed Control (ESC) for the car;
repeat
    Set parameters for low level controller;
until All communication is established
while ( ROSSERIAL == Running ) do
    Initialize all GPIO pins;
    Define pins for Motor, Hall Sensor and Magnetic Encoder;
    if ESC == Enabled then
        Send initialization sequence for ESC;
    end if
    Calibrate the relative distance;
    Wait for messages from high level controller;
    if PWM Signal == Active then
        Send respective pulse value to motor;
        Read data from hall sensor for wheel velocity;
        Read data from magnetic encoder on central shaft;
        Convert hall data into velocity (m/s);
        Convert encoder data into relative distance (m);
        Publish calculated data on ROS Master;
        Subscribe for current v and v_comm data on the Master;
        Calculate error between v and v_comm;
        if error > 0 then
            Proportional gain as Kp_a;
        else
            Proportional gain as Kp_d;
        end if
        Calculate new PWM signal using P-controller;
        Send the PWM signal to the motor;
        Log data onto board via remote laptop over SSH;
    end if
end while
Disable Electronic Speed Control;
Kill the Arduino code;
```

control, when the two cars are started from rest at a minimal distance apart, it is interpreted by the controller as a violation of the barrier. As a result, in the simulation it is seen that the barrier is breached. Therefore, a limit on the v_{qp} is set to make it zero whenever the B goes negative. Pseudo-code of the high-level implementation for the controller based on Barriers (Green font), PCIS (Red font) and via PESSOA (Blue font) can be seen in Algorithm 1.

Low-Level Controller: To establish communication with the hardware, a low-level controller is setup that reads data from the sensors and actuates the motors. The high- and low-level controllers share data across a ROS server, acting as a hardware-software interface. With less computation than the high-level controller, the low-level controller can run at a rate of 57600 Hz. The pseudo-code of this controller is given in Algorithm 2.

5. CONCLUSION: VALIDATION OF EXPERIMENTAL AND SIMULATION RESULTS

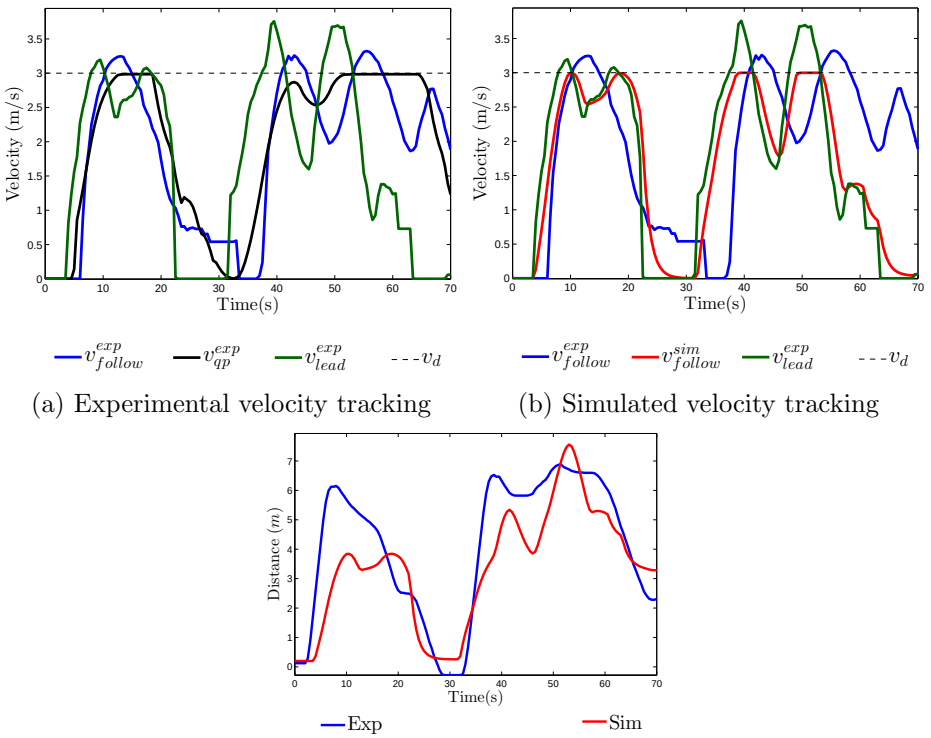
In this section, the simulated performance and the experimental performance of the RCBF, ZCBF, PESSOA and PCIS controllers are analyzed side by side. Importantly, the successful experimental implementation of all the proposed controllers was establish as exhibited in [1], [2].

The barrier function controllers are realized first and the results are discussed in the section below. The PESSOA and PCIS controllers are implemented on the physical hardware next and their results follow the barrier function section. Some of the plots that provide correctness of the controllers are given in Appendix A. One important point to note is that these results have been published by the author in two different conferences as mentioned on the first page.

5.1 Barrier Function Methods

As mentioned in the previous chapters, two types of control barrier functions (CBFs) are constructed, which are unified with a control Lyapunov function (CLF) and optimized using quadratic programs. First, the RCBF type function is realized on the hardware and the results are discussed using the relevant mathematical variables of the controllers.

As can be observed in Fig. 5.1 and Fig. 5.2a, the velocity of the following car is consistent between simulation and experiment. Fig. 5.1a and Fig 5.2a both show all the experimental velocities recorded during the tests; here, v_{qp}^{exp} is the desired velocity calculated from (ACC QP) using the one step forward integration method (3.20), v_{follow}^{exp} is the actual velocity of the car, v_{lead}^{exp} is the velocity of the lead car and v_d is the velocity of the following car. As can be observed by comparing Fig.5.1c



(a) Experimental velocity tracking

(b) Simulated velocity tracking

(c) Relative Distance

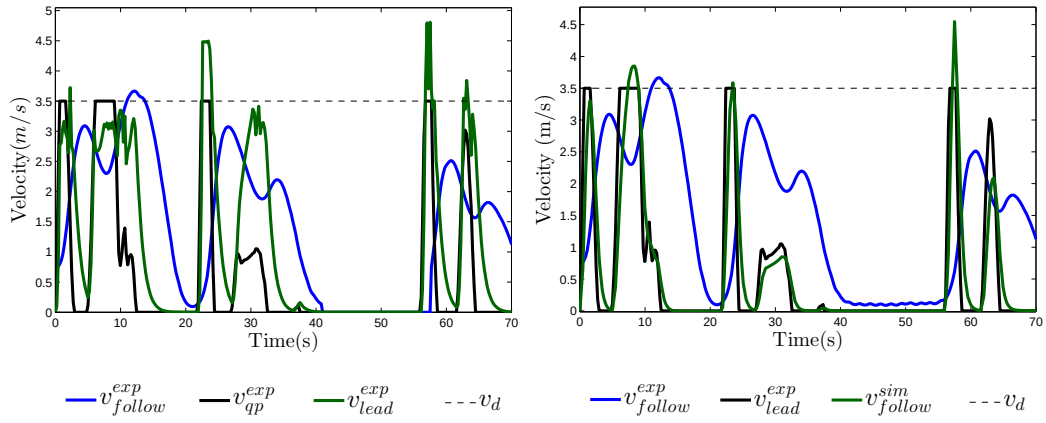
Figure 5.1: Tracking of a desired velocity (on the following car) subject to variable speed on the lead car, both in simulation and experiment. The velocity in both cases is modulated based upon the relative distance between the two cars car.

with Fig. 5.1a, the experiment was successful in that the velocity of the following car is directly modulated by the relative distance, hence the RCBF. Similarly, Fig.5.2c and Fig. 5.2a can be compared to show velocities are in accordance with the relative distance. Similar results can be seen when considering the simulation results obtained by using the experimental lead car data as shown in Fig. 5.1b and Fig. 5.2b; here, the simulation results, v^{sim}_{follow}, are directly compared with the experimentally observed values v^{exp}_{follow}. As expected, the simulation results achieve better velocity tracking, yet these results still accurately predict the behavior seen in simulation.

In Fig. A.3, all of the relevant mathematical quantities from both the simulation

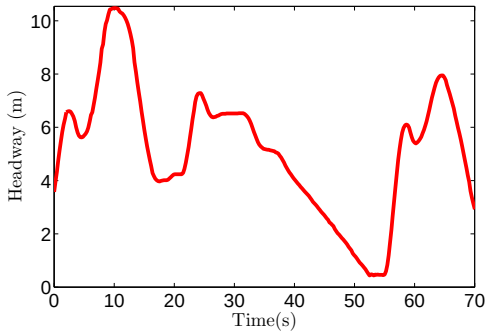
and experimental results are shown for RCBF (with experimental results in the left column and simulation results in the right). Overall, good agreement is shown between the two cases, subject to some notable differences between the two cases. As expected, at the level of force input, there are notable differences between simulation and experiment, probably due to unmodeled phenomena—yet the force input magnitudes are similar in both cases. Differences in the hard constraint behavior, and thus the barrier function, can also be seen. This is likely due to factors related to delays in sensing that propagate through the system; yet, ev en with these practical issues, the hard constraint is always positive (modulo calibration, which introduced a slight bias) indicating the the control law properly enforces the safety constraints. Finally, good agreement is seen between the behavior of the control Lyapunov function, V, and its derivative, $\dot{V}$, indicating the ability to accurately capture the speed regulation-related aspects of the problem.

Similarly in Fig. A.4, all of the relevant mathematical quantities from both the simulation and experimental results are shown for ZCBF (with experimental results in the left column and simulation results in the right). The force input has minute differences between simulation and experiment, but in general good agreement is seen between the two. The differences are again probably due to unmodeled phenomena. The hard constraint and barrier functions look similar as they are exactly the same in theory, but in experiment they are calculated by different sensors, thereby showing certain differences. The important point to note here is that the barrier stays positive at all times and shows that the vehicle maintains a safe distance from the lead vehicle. Finally, good agreement is seen between the behavior of the control Lyapunov function, V, and its derivative, $\dot{V}$, indicating the ability to accurately capture the speed regulation related aspects of the problem.



(a) Experimental velocity tracking

(b) Simulated velocity tracking



(c) Relative Distance

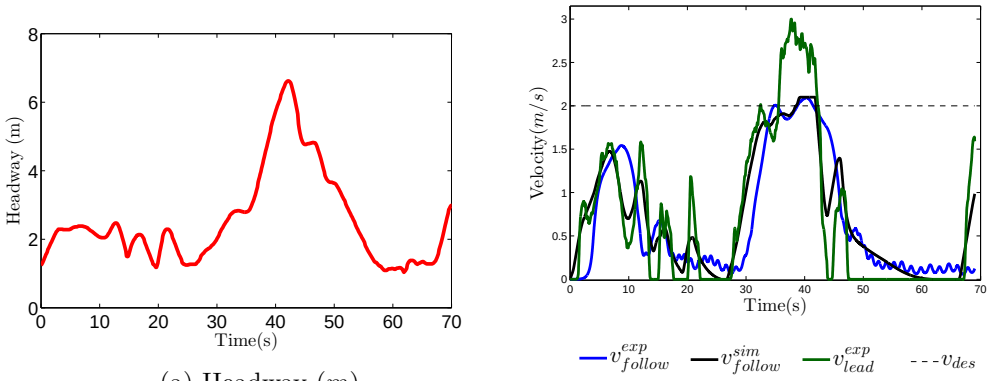
Figure 5.2: Tracking of a desired velocity (on the following car) subject to variable speed on the lead car, both in simulation and experiment. The velocity in both cases is modulated based upon the relative distance between the two cars car.

5.2 Results for Formal Methods

The experimental setup was utilized to implement the two correct-by-design controllers.

Using this substitution for commanded torque, the controllers presented in 3.2.1 and 3.2.2 were successfully implemented as demonstrated in the video [1].

The controllers on the processor board (UDOO Quad), required only 20 MB of memory space. Each controller uses a set of libraries that are included in the main executable file. The PCIS based controller calls upon the Eigen and the Quadratic



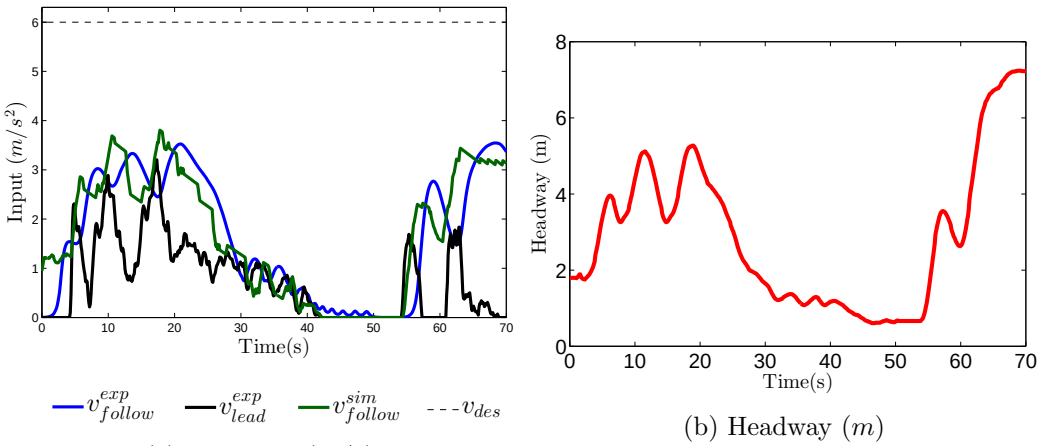
(a) Headway (m)

(b) Velocities (m/s)

Figure 5.3: Experimental and simulation results for the PCIS controller.

Program libraries [19] wirtten by Ben Stevens, for all the mathematical calculations required to produce the control input. The controller synthesized by PESSOA is stored in a ROBDD, which is queried in the main control loop by a standard BDD package, CUDD ([36]). Once the controller is compiled on ROS, it can be activated using ROS commands, to execute at a frequency of 200 Hz displaying the current data on a terminal window. Note that correctness of the implementation depends on the correctness of all the used third party components such as ROS, Arduino. Although it is possible to implement the synthesized controllers without third party modules and thus provide stronger guarantees on the implemented software, we decided not to do so in order to reduce the time and effort dedicated to implementation.

To validate both controllers, the experimental results are presented side by side and discussed in comparison to the respective simulation results. For the sake of comparison, the simulations use the lead car velocity and headway data recorded during the experiments. Moreover, since the lead car was remotely operated by a human, it was not possible to perform experiments for both controllers where the lead car velocity and headway were the same. In order to facilitate the interpretation



(a) Velocities (m/s)

(b) Headway (m)

Figure 5.4: Experimental and simulation results for the controller via PESSOA.

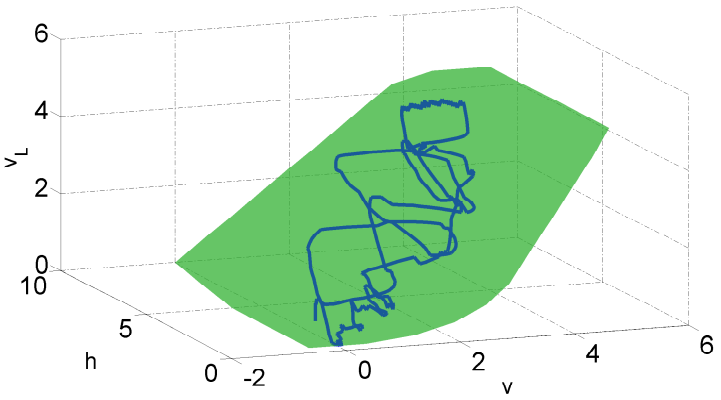
of the results, we plot the desired time headway and velocity as dashed lines. The specification is met whenever experimental or simulation results are below the dashed line for v_{des} and above the dashed line representing τ_{des}.

In Fig. 5.3 and A.1, experimental results for the PCIS controller are compared against simulated results. Fig. A.1a shows that the commanded wheel force from the experiments, as well as from the simulation, always stays within the upper and lower bounds set by the specification. For this experiment we used $F_w^{\min} = -0.07mg$ and $F_w^{\max} = 0.05mg$, where m and g are the mass of the car and the gravity due to acceleration, respectively. In Figures A.1b and 5.3b, it can be seen that specifications are enforced during the experiments as well as during the simulations, i.e., the time headway was always above the desired value of 1 while the velocity was always below the desired value of 2. Also a reasonable agreement can be seen between experiment and simulation. Hence all the ACC specifications are met; i.e., the input constraint is satisfied at all times, a lower bound on time headway and an upper bound on the velocity are achieved and maintained.

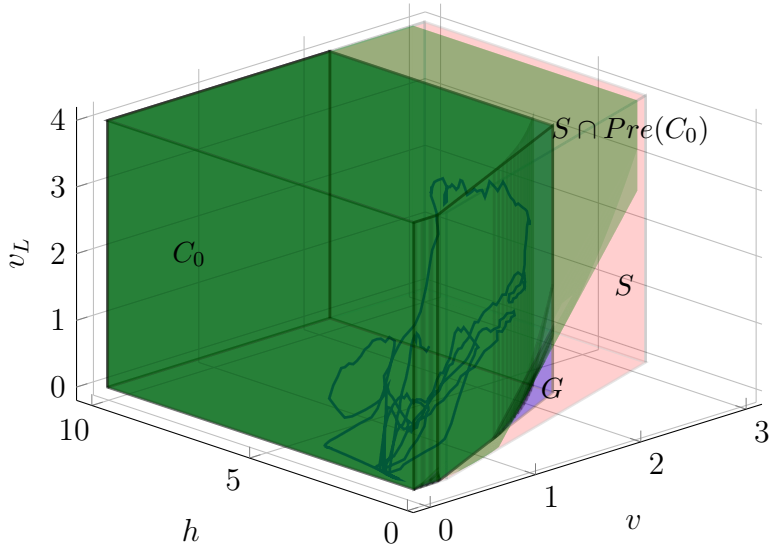
The performance of the controller presented in Section 3.2.2 is also evaluated by comparing the experimental results against the simulated results; see Fig. 5.4 and Fig. A.2. As shown in Fig. A.2a the input force satisfies the constraints in the simulations. For this experiment the specification bounds were $F_w^{\min} = -0.3mg$ and $F_w^{\max} = 0.2mg$. In Figures A.2b and 5.4a, we can see that the time headway and velocity specifications are met during simulations and experiments, i.e., the time headway was always above 1 and the velocity was always below the desired value $v_{des} = 6$. Moreover, a good agreement between simulation and experimental results can be appreciated in Figures A.2b and 5.4a.

Finally, to demonstrate the successful implementation of the two controllers, we show the controller domain for both methods. Fig. 5.5b shows a 3D trajectory (v_{follow}^{exp}, h, and v_{lead}^{exp}) of the controlled car plotted along with the PICS controller domain. The specification is guaranteed to be satisfied as long as the trajectory remains in the controller domain. Similarly, for the PESSOA-based controller, the controller domain and the experimental 3D trajectory are plotted in Fig. 5.5a. In each case, the trajectories remain in the domain, thereby experimentally validating the synthesized correct-by-design controllers.

The overall hardware-software experiments proved successful for both controllers. Although the implementation of correct-by-design controllers required more effort than a traditional PID controller, the authors felt that such effort is more than worth it given the guaranteed correctness and performance, especially when dealing with complex specifications. We also concluded that the computational requirements to execute both controllers in real-time are within reach of most modern platforms.



(a) Solution via PESSOA



(b) Solution via PCIS

Figure 5.5: Plots showing the Controller Domains for the two controllers with the experimental trajectory plotted with it.

5.3 Comparative Analysis

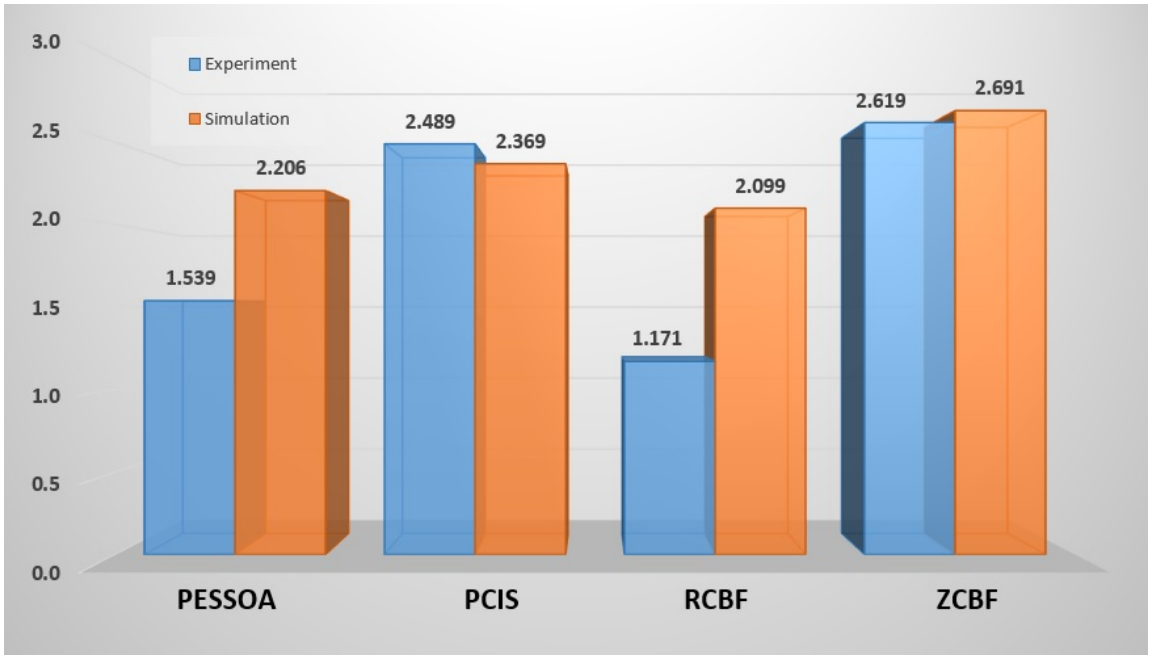
The four controllers implemented, as seen in the previous sections and chapters, have different theories behind the construction of each. It is obvious that the objectives behind all of them are the same. So, to compare the four together, a metric is required such that the controllers can be compared for safety along with tracking

various velocities.

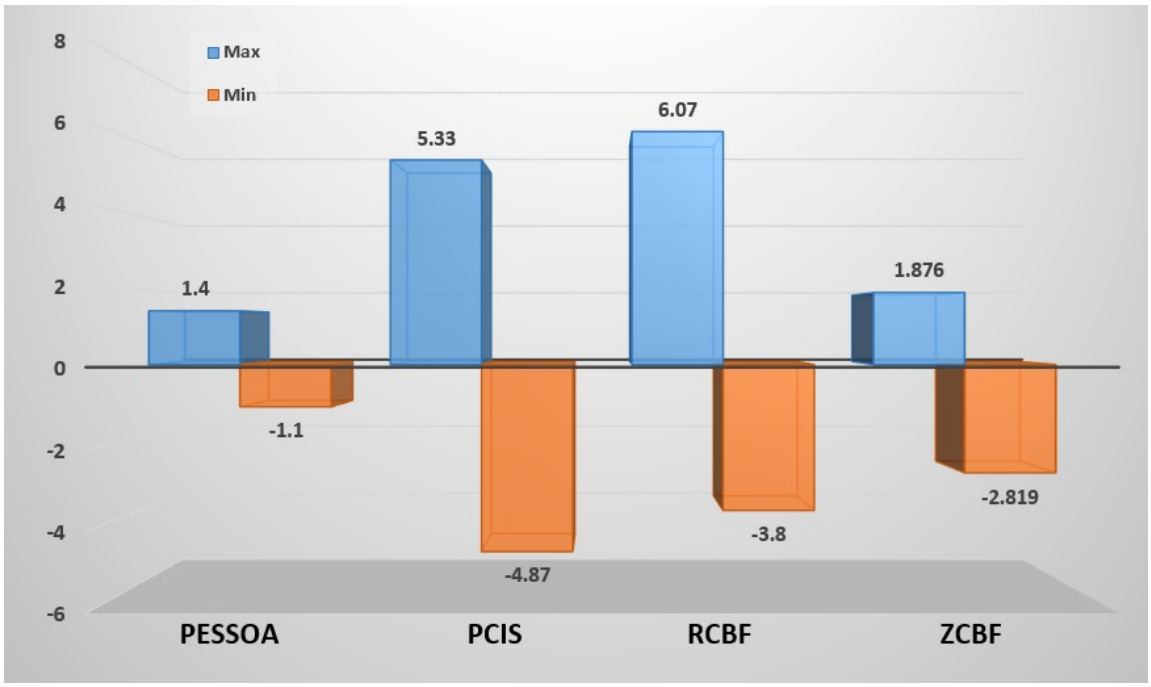
In this section, a formal way is developed to achieve this comparison to see which controller works better in which specific situation. Four different comparative metrics chosen are : Time Headway Margins, Force Gradients, Tracking Error v/s v_l and Tracking Error v/s v_d. The idea behind choosing these certain metrics is to analyze the safety (time headway), comfort (force gradients) and tracking velocities accordingly. Each of these metrics are plotted as bar graphs for all four methods with experimental and simulation results.

In Fig. 5.6a, the time headway margin is plotted. It is important to note that the minimum time headway (τ_{min}) for all the controllers is set at $1s$. To construct these plots, the minimum normalized value of te experimental and simulated time headways data was taken. In general it is seen that the experimental values are lower than the simulation headways. This is due to the fact that the controllers work better in an ideal environment. All the experimental values for each of them are above $1s$, proving that all the methods show safety and do not ever violate the time headway barrier. It is clearly seen that the ZCBF approach shows the best results and with PCIS and PESSOA following closely.

Fig 5.6b shows the force gradients for the various techniques. This plot proves the comfort features of the controllers. The numerical gradient is calculated for the force data collected during the tests. For each controller, the minimum and maximum values is taken to compare which method proves to be the most comfortable. Lower the value, smoother the ride for the passengers. This validates that the forces on the PESSOA controller has the least gradient, while the RCBF approach shows higher forces gradients. This can be easily validated against the theory, as the abstractions for the PESSOA are done on force inputs hence keeping them in bounds. The RCBF approach on the other hand has no formal force constraints acting during the tests.



(a) Time Headway Margin

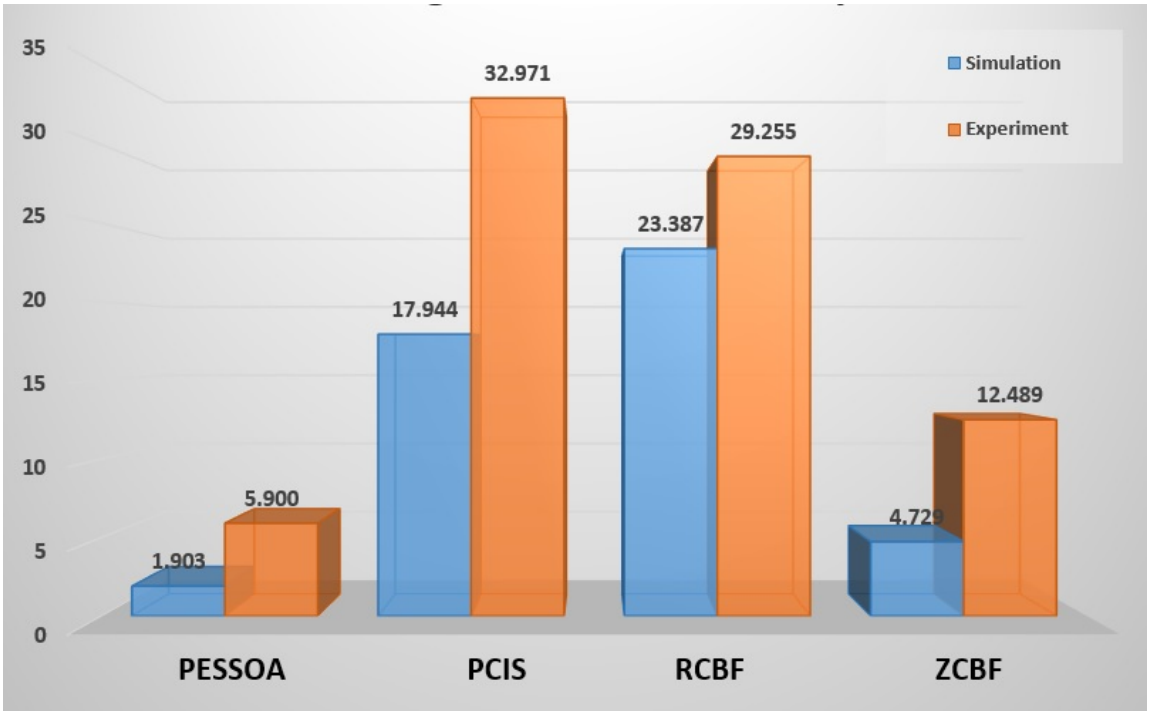


(b) Force Gradient

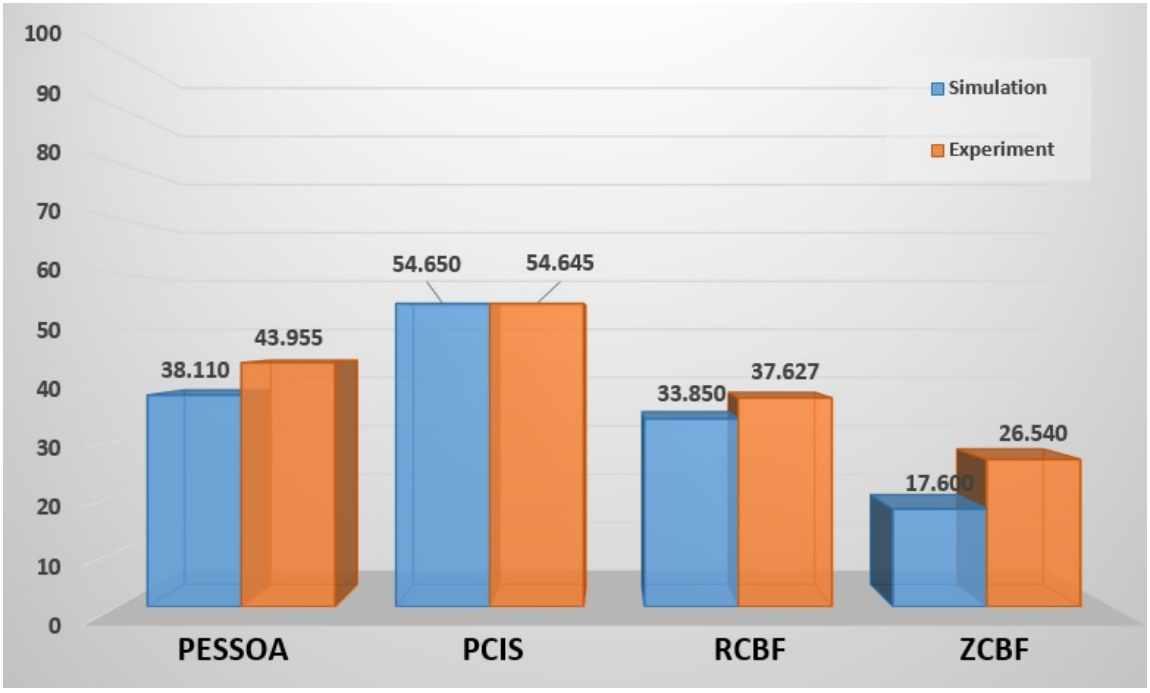
Figure 5.6: Plots showing the safety comparisons and the force gradient to compare comfort of riders.

The Fig. 5.7a and Fig. 5.7b show the tracking results for the following car velocity against lead car and the desired velocities, respectively. These values are calculated by taking the normalized errors between the two velocities. It is easily notable that lower the tracking errors better the tracking for of the velocities. In the case of the tracking of the lead car, it is seen that the PESSOA controller again shows really good agreement. The simulation results of course are lower than the experiment as there are always hardware delays and sensors noise. PCIS and RCBF both show poor tracking, which is not an issue with the working of the controllers but is due to the hardware problems.

The next sub-figure shows the tracking error against the desired velocity, this is calculated using the normalizing the errors between the two velocities. Similar to the previous graphs, lower the value better is the performance. It is to be kept in mind that these results are on custom built hardware so the data is not as consistent in experiment as we expect in theory. The results of the ZCBF controller is seen to be the best and PCIS approach still has issues with tracking the desired velocity.



(a) Tracking Error v/s v_{lead}



(b) Tracking Error v/s v_{des}

Figure 5.7: Plots showing the tracking errors while there is a lead car and when there is not car in front.

REFERENCES

[1] Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars. `http://youtu.be/9Du7F76s4jQ`.

[2] Experimental Validation of Correct-by-Design Adaptive Crusie Control Software on Scale-Model Cars. `http://youtu.be/i1w6BWMnTDE`.

[3] S. B. Akers. Binary decision diagrams. *Computers, IEEE Transactions on*, 100(6):509–516, 1978.

[4] A. D. Ames, K. Galloway, and J. W. Grizzle. Control Lyapunov Functions and Hybrid Zero Dynamics. *To appear, Proc. 51st IEEE Conf. Decision and Control*, 2012.

[5] A. D. Ames, K. Galloway, J. W. Grizzle, and K. Sreenath. Rapidly Exponentially Stabilizing Control Lyapunov Functions and Hybrid Zero Dynamics. *IEEE Trans. Automatic Control*, 59(4):115–1130, 2014.

[6] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *Conference on Decision and Control*, Dec 2014.

[7] A. D. Ames, K., Galloway, and J.W. Grizzle. Control lyapunov functions and hybrid zero dynamics. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference*, pages 6837–6842, 2012.

[8] A. D. Ames and M. Powell. Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs. In *D.C. Tarraf (ed.), Control of Cyber-Physical Systems, Lecture Notes in Control and Information Sciences 449*, 2013.

[9] A. Bacciotti and L. Rosier. *Liapunov functions and stability in control theory*. Springer, 2005.

[10] D. P. Bertsekas. Infinite Time Reachability of State-Space Regions by Using Feedback Control. *Automatic Control, IEEE Transactions on*, 17(5):604–613, 1972.

[11] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] L. Dai, T. Gan, B. Xia, and N. Zhan. Barrier certificates revisited, 2013. (preprint).

[13] S. Darbha and K. R. Rajagopal. Intelligent cruise control systems and traffic flow stability. *Transportation Research Journal, Vol. C*, Dec 1998.

[14] Elena De Santis, Maria Domenica Di Benedetto, and Luca Berardi. Computation of Maximal Safe Sets for Switching Systems. *Automatic Control, IEEE Transactions on*, 49(2):184–195, 2004.

[15] T. C. Karthik S. F. A. Arvind Raj R., S. B. Sandhiya Kumar. Cruise control operation from zero to preset speed simulation and implementation. In *International Journal of Information and Education Technology, Vol. 1, No. 1 ISSN: 2010-3689*, April 2011.

[16] K. Galloway, K. Sreenath, A. D. Ames, and J. W. Grizzle. Torque saturation in bipedal robotic walking through control Lyapunov function-based quadratic programs. *IEEE Access*, 3:323–332, 2015.

[17] K. Galloway, K. Sreenath, A. D. Ames, and J.W. Grizzle. Torque saturation in bipedal robotic walking through control lyapunov function based quadratic programs. arXiv preprint arXiv:1302.7314, 2013.

[18] A. Girard. Reachability of Uncertain Linear Systems Using Zonotopes. In *Proceedings of the International Conference on Hybrid Systems Computation and Control*, pages 291–305, 2005.

[19] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[20] S.-C. Hsu, X. Xu, and A. D. Ames. Control barrier functions based quadratic programs with application to bipedal robotic walking. In *American Control Conference*, 2015.

[21] P. A Ioannou and C.-C. Chien. Autonomous intelligent cruise control. *IEEE Trans. Veh. Tehnol.*, 42(4):657–672, 1993.

[22] ISO 15622:2010 (E). Intelligent transport systems – adaptive cruise control systems – performance requirements and test procedures. Technical report, International Organization for Standardization, 2010.

[23] M. Mazo Jr, A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *Computer Aided Verification*, pages 566–569. Springer, 2010.

[24] H. Kong, F. He, X. Song, W. Hung, and M. Gu. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In *CAV2013, volume 8044 of LNCS*, pages 242–257. Springer, 2013.

[25] S. Li, K. Li, R. Rajamani, and J. Wang. Model predictive multi-objective vehicular adaptive cruise control. *IEEE Trans. Control Syst. Technol.*, 19(3):556–566, 2011.

[26] C.-Y. Liang and H. Peng. Optimal adaptive cruise control with guaranteed string stability. *Vehicle System Dynamics*, 32((4-5)):313330, 1999.

[27] C.-Y. Liang and H. Peng. O string stability analysis of adaptive cruise controlled vehicles. *JSME International Journal Series C*, 43((3)):671677, 2000.

[28] W-L. Ma, H-H. Zhao, S. Kolathaya, and A. D. Ames. Human-inspired walking via unified pd and impedance control. In *IEEE Conference on Robotics and Automation*, May 2014.

[29] B. Morris, M. Powell, and A. D. Ames. Sufficient conditions for the lipschitz continuity of QP-based multi-objective control of humanoid robots. In *IEEE Conference on Decision and Control*, 2013.

[30] G.J.L. Naus, J. Ploeg, M.J.G. V. de Molengraft, W.P.M.H. Heemels, and M. Steinbuch. Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach. *Control Eng. Pract.*, 18(8):882–892, 2010.

[31] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Preliminary results on correct-by-construction control software synthesis for adaptive cruise control. In *Proc. of the IEEE CDC*, 2014.

[32] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris. Multi-objective control for multi-agent systems using Lyapunov-like barrier functions. In *IEEE Conference on Decision and Control*, Florence, Italy, 2013.

[33] G. Pola and P. Tabuada. Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization*, 48(2):719–733, 2009.

[34] M. Z. Romdlony and B. Jayawardhana. Uniting control Lyapunov and control barrier functions. In *IEEE Conference on Decision and Control*, pages 2293–2298, 2014.

[35] S. Sastry. *Nonlinear Systems: Analysis, Stability and Control*. New York : Springer, 1999.

[36] F Somenzi. Cudd: Cu decision diagram package, release 2.5. 0, 2012. *http://vlsi. colorado. edu*.

[37] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[38] K. P. Tee, S. S. Ge, and E. H. Tay. Barrier lyapunov functions for the control of output-constrained nonlinear systems. *Automatica*, 45(4):918 – 927, 2009.

[39] R. R. Teetor. Speed control device for resisting operation of the accelerator. In *U.S. Patent Office*, 1950.

[40] B. V. Arem, C.J.G. van Driel, and R. Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Trans. Intell. Transp. Syst.*, 7(4):429–436, 2006.

[41] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Trans. Intell. Transp. Syst.*, 4(3):143–153, 2003.

[42] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency*, pages 238–266. Springer, 1996.

[43] K. Vogel. A comparison of headway and time to collision as safety indicators. In *Accident Analysis & Prevention Volume 35*, pages 427–433.

[44] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. In *Proceedings of the 7th IFAC Symposium on Nonlinear Control System*, 2007.

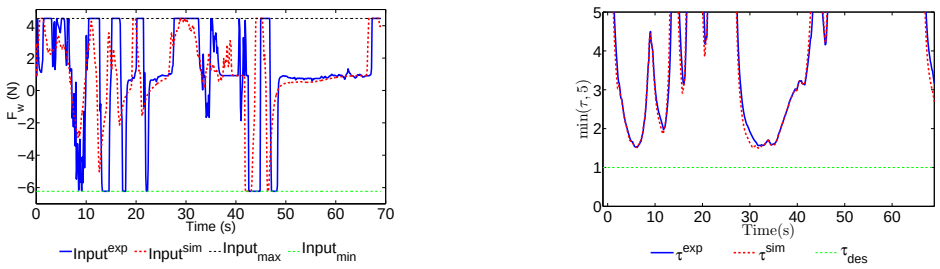
[45] R. Wisniewski and C. Sloth. Converse barrier certificate theorem. In *52nd IEEE Conference on Decision and Control*, 2013.

[46] X. Xu, P. Tabuada, A. D. Ames, and J. W. Grizzle. Robustness of control barrier functions for safety critical control. In *IFAC Conference on Analysis and Design of Hybrid Systems*, 2015. (submitted).

APPENDIX A

EXPERIMENTAL RESULTS : RELEVANT VARIABLES

This appendix provides all the relevant mathematical variables from the experiments tested during this thesis.



(a) Force (N) (b) Time Headway (s)

Figure A.1: Experimental and simulation results for the PCIS controller.

Controller	Actual	Simulated
PESSOA	1.538	2.206
PCIS	2.489	2.369
RCBF	1.171	2.099
ZCBF	2.619	2.691

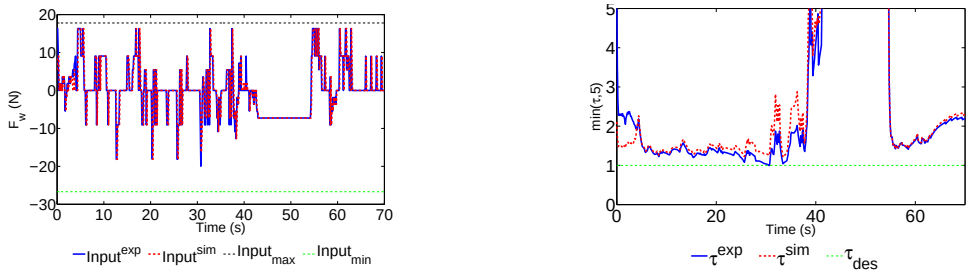
Table A.1: Time Headway

Controller	Max	Min
PESSOA	1.4	-1.13
PCIS	5.338	-4.878
RCBF	6.07	-3.8
ZCBF	1.876	-2.819

Table A.2: Force Gradient

Controller	$\|\|v_d - x_2\|\|$	$\|\|v_d - v_{sim}\|\|$
PESSOA	43.995	38.110
PCIS	54.645	54.65
RCBF	37.625	33.850
ZCBF	26.540	17.6

Table A.3: Tracking Errors on desired velocities.

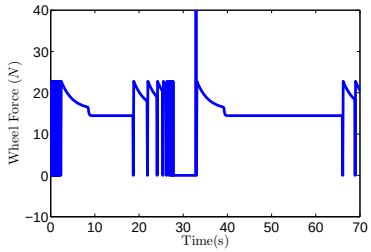


(a) Force (N) (b) Time Headway (s)

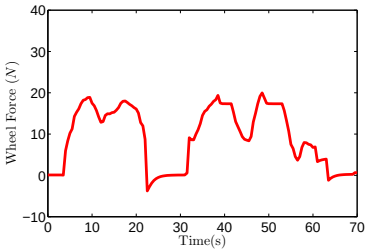
Figure A.2: Experimental and simulation results for the controller via PESSOA.

Controller	$\|\|v_l - x_2\|\|$	$\|\|v_l - v_{sim}\|\|$
PESSOA	5.90	1.903
PCIS	32.971	17.944
RCBF	29.255	23.387
ZCBF	12.489	4.729

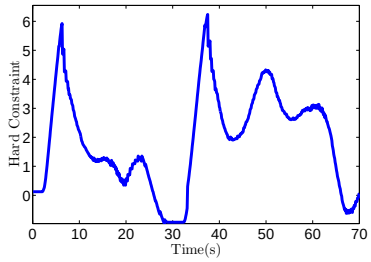
Table A.4: Tracking Errors on Lead car velocities.



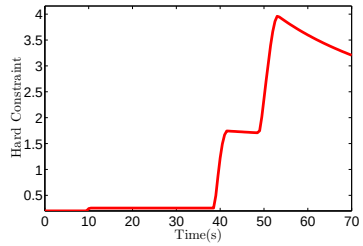
(a) Force Input (N)



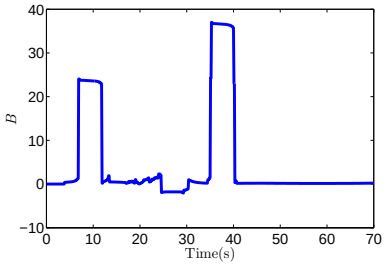
(b) Force Input (N)



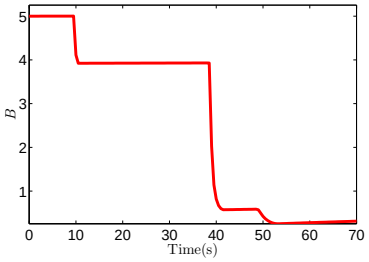
(c) Hard Constraint



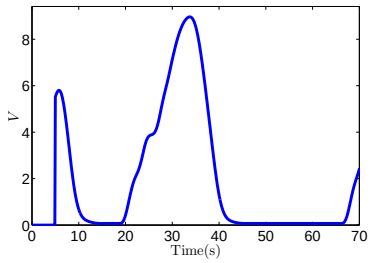
(d) Hard Constraint



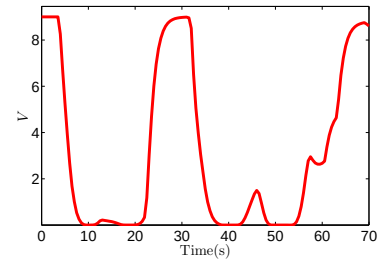
(e) B



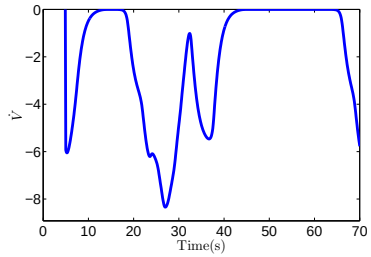
(f) B



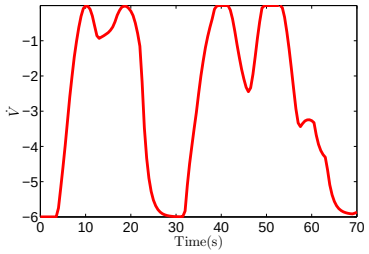
(g) V



(h) V

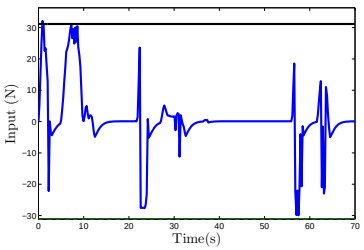


(i) $\dot{V}$

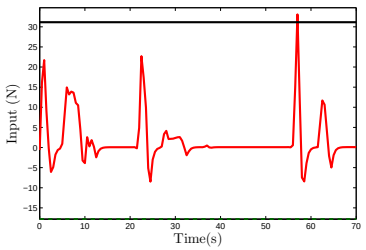


(j) $\dot{V}$

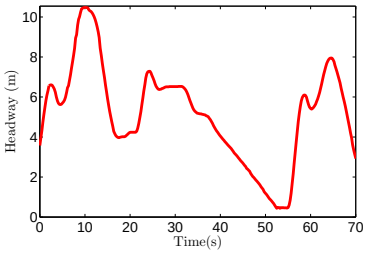
Figure A.3: Experimental results (left column) and simulation results (right column) for all of the relevant variables for RCBF controller.



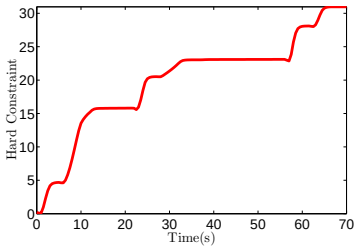
(a) Force Input (N)



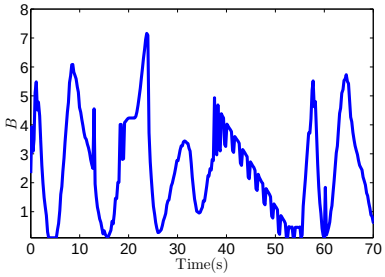
(b) Force Input (N)



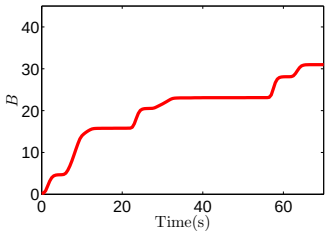
(c) Headway



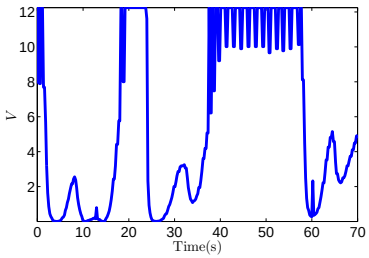
(d) Headway



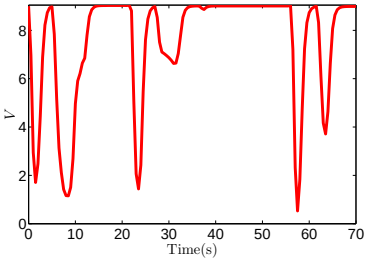
(e) B



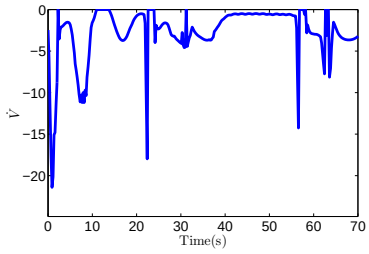
(f) B



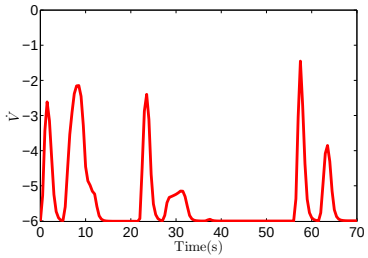
(g) V



(h) V



(i) $\dot{V}$



(j) $\dot{V}$

Figure A.4: Experimental results (left column) and simulation results (right column) for all of the relevant variables.