Adaptive Cruise Control: Experimental Validation of Advanced Controllers on Scale-Model Cars

Aakar Mehra[1], Wen-Loong Ma[1], Forrest Berg[1], Paulo Tabuada[2], Jessy W. Grizzle[3] and Aaron D. Ames[1]

***Abstract*— Recent advances in automotive technology, such as, sensing and onboard computation, have resulted in the development of adaptive cruise control (ACC) algorithms that improve both comfort and safety. With a view towards developing advanced controllers for ACC, this paper presents an experimental platform for validation and demonstration of an online optimization based controller. Going beyond traditional PID based controllers for ACC that lack proof of safety, we construct a control framework that gives formal guarantees of correctness. In particular, safety constraints—maintaining a valid following distance from a lead car—are represented by control barrier functions (CBFs), and control objectives—achieving a desired speed—are encoded through control Lyapunov functions (CLFs). These different objectives can be unified through a quadtraic program (QP), with constraints dictated by CBFs and CLFs, that balances safety and the control objectives in an optimal fashion. This methodology is demonstrated on scale-model cars, for which the CBF-CLF based controller is implemented online, with the end result being the experimental validation of an advanced adaptive cruise controller.**

I. Introduction

According to a 2008 survey conducted by the National Highway Traffic Safety Administration, 93% (i.e., 9.48 Million out of 10.2 Million) of all the car crashes in the U.S. are caused by mistakes made by the driver. Although, according to the survey conducted in 2013, these numbers have slightly decreased, satisfactory technological solutions to prevent accidents have yet to be developed. As a result, researchers have more impetus to solve this problem using onboard sensing, computation and control to assist human drivers. Cruise control, Anti-lock Braking Systems (ABS), traction control, obstacle avoidance and improved traffic flow are a few examples [14], [15]. Conventional cruise control [23] (CCC) has been successfully implemented in almost all production cars in the United States, yet has not actively taken into account collision avoidance. Adaptive cruise control (ACC), which aims to unify CCC with safety related constraints [17], is being actively studied from various perspectives [19], [11]. Mitsubishi was the first company to start the concept of ACC in 1995, designing the *Preview Distance Control*, a method that matches the velocity of the vehicle to its immediate leader based on a laser distance measurement system.

This work is supported by National Science Foundation through the CPS Awards 1239055, 1239037 and 1239085.

[1]Department of Mechanical Engineering, Texas A&M University, College Station, Texas 77843, email: {aakar.m89, wenlongma, fberg and aames}@tamu.edu

[2]Department of Electrical Engineering, University of California at Los Angles, email: tabuada@ucla.edu

[3]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI , email: grizzle@umich.edu

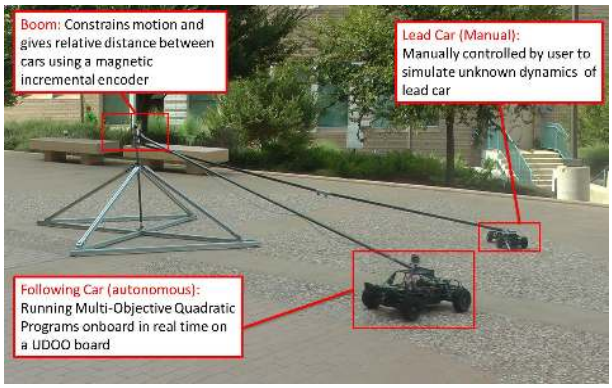


Fig. 1: Experimental setup with the electric car on the boom

To experimentally evaluate advanced automotive controllers like ACC, previous research by the Program of Advanced Technology for the Highway (PATH) has focused on creating platoons between vehicles on the highways, e.g., multiple controlled cars follow a lead car even for lane changes [22]. These results were achieved using radar sensor detection by continuously calculating the headway distance to maintain the desired velocity. Intelligent cruise control (ICC) is a variant of ACC explored by [12], [24], [5] which prioritizes autonomous driving by designing controllers with braking systems that require minimal manual interaction. Reference [7] investigates a braking system that allows the vehicle to perform emergency stops when necessary, then return to the set point velocity. More progressive control methods, for example, satisfying Lyapunov Stability criterion, have been shown to create smooth traffic flow in [6].

The main contribution of this paper is the simulation and experimental validation of a controller that mathematically accounts for both safety and comfort of the driver. The safety critical nature of the problem necessitates controllers that are formally correct, i.e., give guarantees of safety. To address this, in [2], [17], controllers were presented that give proofs of safety while simultaneously achieving speed related control objectives. Of special interest here are methods that utilize online optimization to maximize the achievement of performance goals (via CLFs), subject to safety (CBFs) constraints, and even bounds on actuation. In particular, safety constraints are formulated as CBFs and speed regulation related control objectives are encoded as CLFs; these representations allow for the formulation

of a quadratic program (QP) problem that dynamically adjusts these potentially contradictory specifications. These constructions are revisited in the context of a lead vehicle with variable speed and used to derive a QP based controller that formally ensures safety. In addition to simply simulating the resulting controller to validate its correctness, to more sufficiently validate the feasibility of the framework in real world application, we introduce an experimental platform utilizing scale-model cars (see Fig. 1) to test the QP based ACC controllers. In addition to simulating the resulting controller to validate its correctness, this paper validates the feasibility of the QP-CBF-CLF approach in a real-time embedded environment and its ability to handle model uncertainty. In particular, the QP based controller is implemented on an autonomous following car while the lead car is manually controlled . The end result is the experimental validation of online optimization based controllers for ACC that simulate realistic driving conditions.

The structure of this paper is as follows: Sect. II introduces the experimental scale-model car test bed, the corresponding dynamical system model, and the constraint specifications involved in the ACC problem. In Sect. III we introduce the control framework that is implemented experimentally. In particular, we begin by translating the safety specifications of the ACC problem to control barrier functions and speed related objectives to control Lyapunov functions. Sect. IV discusses the embedded level implementation of the control algorithm on the experimental platform. Finally, Sect. V and Sect. VI conclude the validation of the controller via simulation and experimental results.

II. System Dynamics and Experimental Setup

This section presents a novel experimental platform (see Fig. 1) for testing advanced control algorithms for Adaptive Cruise Control (ACC). This platform consists of two scale-model cars, constrained to a 2D sagittal plane (via a boom) to allow for the detailed study of speed regulation and collision avoidance. After discussing the experimental platform, we introduce the nonlinear model of the autonomous (following) car that will be used to test the controller. Finally, we propose the necessary constraints: safety, speed and wheel force that will be used to construct the ACC problem in Sect. III.

A. Experimental Setup

We begin by discussing the experimental platform that will be used to evaluate formal constructions. This setup is shown in Fig. 1 and detailed in Fig. 2.

In order to maintain an appropriate balance between realism and complexity, we chose two electric, remote controlled cars powered by brushless DC (BLDC) motors as the test vehicles for the experiments (see Fig. 2). 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. 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.

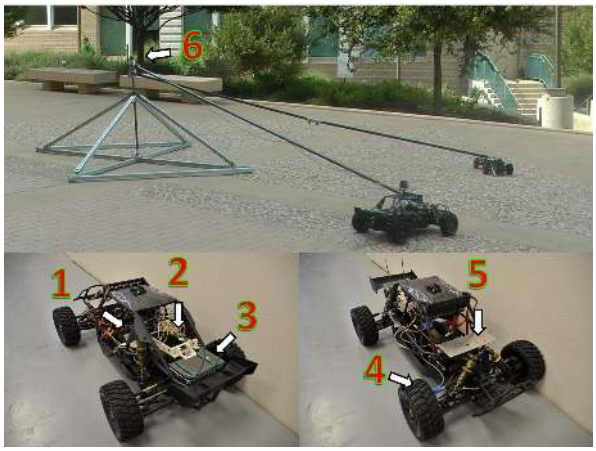


Fig. 2: Experimental Setup. The boom restricts motion to a circle. As shown in figure: **(1)** Electric motor, **(2)** On-board UDOO **(3)** Battery for the UDOO board, **(4)** Hall sensor and magnets, **(5)** Boom attachment plate, **(6)** Magnetic encoder on the central shaft to measure the relative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

To eliminate lateral motion, both cars are rigidly attached to a central shaft via a boom; see Fig. 1. A similar two dimensional setup has already been implemented in several robotic experiments, e.g., in the context of bipedal locomotion [16]. 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.

B. Nonlinear Dynamics

This ACC - equiped is modeled as a point mass system subject to various forces as illustrated in the free body diagram shown in Fig. 3. Thus, resulting in the equations of motion of the form:

$$m\frac{dv}{dt} = F_w - F_r, \tag{1}$$

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

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

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. All parameters used in this paper are listed in TABLE I.

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

$$\frac{d}{dt}D = v_l(t) - v, \tag{3}$$

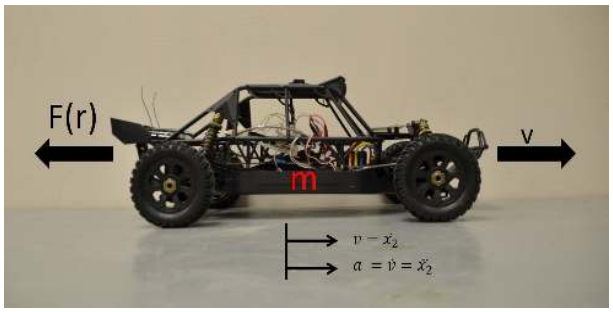


Fig. 3: Dynamics on a free body diagram of a vehicle

where $v_l(t)$ and v are the velocities of the lead and controlled car, respectively. Note that the velocity of the lead car, $v_l(t)$, is assumed to be a time varying function (this is in contrast to previous work by the authors [2] where it was assumed to be constant). Without some assumptions on the lead car, there can be no correctness. $v_l(t)$ will be governed by the user manually controlling the lead car and sensed through the boom encoder.

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 can be converted to a nonlinear ODE:

$$\dot{x} = \underbrace{\begin{bmatrix} x_2 \\ -\frac{F_r}{m} \end{bmatrix}}_{f(x,z)} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{g(x,z)} u, \tag{4}$$

$$\dot{z} = \underbrace{v_l - x_2}_{q(x,z,t)}, \tag{5}$$

where $u = F_w$ is the control input. We now introduce the constraints on the dynamics of this system as dictated by the ACC problem.

C. Control Objectives

With the goal of validating the requirements of ACC, including: collision avoidance, adaptive velocity control, and driver comfort, this section will present three classes of constraints. These constraints will form the basis for the development of an advanced online-optimization based controller for the ACC problem.

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 I: Parameters Used in Simulation and Experiments

Hard Constraint: The constraint with the highest priority is to prevent the following vehicle from colliding with the lead car—this constraint should never be violated under any circumstance. For the purposes of this paper, we consider the simple rule stated in [25]: the minimum distance between two cars, must be "half the speedometer", which is represented mathematically as:

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

where D is in *meters* and v is in *kilometers per hour*

Soft Constraint: 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 optimal to reduce the peak forces generated by the car in emergency situations. For example, the comfort constraints would prevent sudden jerks so that the driver can experience a comfortable ride. This can be achieved by constraining the acceleration and the deceleration of the vehicle through an inequality constraint:

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

where c_d and c_a are the factors of g for deceleration and acceleration, respectively. Similar bounds are assumed for the lead vehicle's acceleration.

III. Control Framework

The goal of this section is to develop a nonlinear online optimization based controller that formally guarantees the precise specifications of the ACC problem. In particular, to ensure satisfaction of the hard constraint, we utilize the framework of control barrier functions [18], [26], [13] and, specifically, the formulation presented in [2]. Soft constraints are viewed as control objectives, and represented by control Lyapunov functions [8], [3]. Finally, the hard constraints, soft constraints and the comfort constraints are unified into a single control framework through the use of a quadratic program (QP) [4], [9].

A. Hard Constraints as Control Barrier Functions

To construct 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. Motivated by previous work [2] for the case where the lead car is moving at a constant velocity, this paper will develop a control barrier function for a varying lead car velocity: $v_l(t)$. In particular, by converting units to m and s, the hard constraint (HC1) can be restated as:

$$h(x, z) = z - 1.8x_2 \geq 0, \tag{6}$$

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

$$\mathcal{C} = \{(x, z) \in \mathbb{R}^3 : h(x, z) \geq 0\}, \tag{7}$$

$$\partial\mathcal{C} = \{(x, z) \in \mathbb{R}^3 : h(x, z) = 0\}, \tag{8}$$

$$\text{Int}(\mathcal{C}) = \{(x, z) \in \mathbb{R}^3 : h(x, z) > 0\}. \tag{9}$$

Then the CBF candidate B can be chosen as:

$$B\big(h(x,z)\big) = B(x,z) = \frac{1}{z - 1.8x_2}, \tag{10}$$

with associated derivative:

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

Based on Definition 2 from [2] and the fact that for $(x,z) \in \mathrm{Int}(\mathcal{C})$, it follows that $1.8x_2 - z < 0$, $B(x,z)$ is a valid CBF if it satisfies

$$\dot{B}(x,z,t,u) \leq \frac{\gamma}{B(x,z)}, \tag{11}$$

where γ is a positive constant. This leads to

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

Therefore, by Theorem 1 of [2] any admissible control input $u \in U$ satisfying (HC1-CBF) will guarantee that $B(x,z)$ is a valid control barrier function, i.e., any $(x_0, z_0) \in \mathcal{C}$ will stay in $\mathcal{C}$ for all time for any control law satisfying (HC1-CBF).

B. Control Lyapunov Functions for Soft Constraints

In this section, we revisit the mathematical methodology used in [2] to build the soft constraint based on Control Lyapunov Functions (CLFs) [8]. 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 as:

$$V(y) = y^2, \tag{12}$$

which yields

$$\dot{V}(y) = \underbrace{-\frac{2y}{m}F_r}_{L_f V} + \underbrace{\frac{2y}{m}}_{L_g V} u. \tag{13}$$

According to Definition 3 in [3], 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

$$\inf_{u \in U} [L_f V(y) + L_g V(y) u + \varepsilon V(y)] \leq 0, \tag{14}$$

is also satisfied. In other words, with a proper choice of control input u, the output $y(x,z)$ will be exponentially driven to zero, which enforces velocity tracking. However, this function needs to be converted into constraints that are functions of (x,z). To achieve this, by defining

$$\begin{aligned} \psi_0(x,z) &= -\frac{2(x_2 - v_d)}{m}F_r + \varepsilon(x_2 - v_d)^2, \\ \psi_1(x,z) &= \frac{2(x_2 - v_d)}{m}, \end{aligned} \tag{15}$$

we can then construct the CLF constraint:

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

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

C. QP based Controller

Following [2], we will develop an online quadratic program (QP) based controller that will provably satisfy the hard constraints, while achieving the soft and comfort constraints whenever possible. To construct a cost function for the QP, we utilized notions from feedback linearization [21] to develop a cost that will favor convergence to the control objective (achieving a desired speed). In particular, a specific example of a control input that satisfies (13) is given by:

$$u = \frac{1}{L_g y}(-L_f y + \mu) = F_r + m\mu, \tag{16}$$

where μ is the control input for the linearized output dynamics (see [21]). To minimize the control effort μ, the cost function of QP is chosen as:

$$\mu^T \mu = \frac{1}{m^2}(u^T u - 2u^T F_r + F_r^2). \tag{17}$$

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

$$\begin{aligned} u^*(x,z) = \underset{\mathbf{u} = \begin{bmatrix} u \\ \delta_{sc} \\ \delta_{cc} \end{bmatrix} \in \mathbb{R}^3}{\mathrm{argmin}} \quad & \frac{1}{2}\mathbf{u}^T H_{\mathrm{acc}} \mathbf{u} + F_{\mathrm{acc}}^T \mathbf{u} && \text{(ACC QP)} \\ \text{s.t.} \quad & A_{\mathrm{clf}}\mathbf{u} \leq B_{\mathrm{clf}}, && \text{(CLF)} \\ & A_{\mathrm{cbf}}\mathbf{u} \leq B_{\mathrm{cbf}}, && \text{(BCF)} \\ & A_{\mathrm{cc}}\mathbf{u} \leq B_{\mathrm{cc}}. && \text{(CC)} \end{aligned}$$

In which,

$$H_{\mathrm{acc}} = 2\begin{bmatrix} \frac{1}{m^2} & 0 & 0 \\ 0 & p_{sc} & 0 \\ 0 & 0 & p_{cc} \end{bmatrix}, \quad F_{\mathrm{acc}} = -2\begin{bmatrix} \frac{F_r}{m^2} \\ 0 \\ 0 \end{bmatrix}, \tag{18}$$

and $A_{\mathrm{clf}}, B_{\mathrm{clf}}$ and $A_{\mathrm{bcf}}, B_{\mathrm{bcf}}$ are the inequality constraints obtained from (HC1-CBF), (SC1-CLF):

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

Note that since the comfort constraint is also a conditional constraint and it directly acts on the control input, $A_{\mathrm{cc}}, B_{\mathrm{cc}}$ can be obtained by modifying (CC1) by adding the relaxation factor δ_{cc}:

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

This results in:

$$A_{\mathrm{cc}} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix}, \quad B_{\mathrm{cc}} = \begin{bmatrix} c_a mg \\ c_d mg \end{bmatrix},$$

where p_{cc} is the user-defined penalty for the relaxation. Because we want to give higher priority to comfortable driving experience over velocity regulation, it is necessary

to set $p_{sc} \ll p_{cc}$, where p_{sc} and p_{cc} are the penalties on the soft constraints and comfort constraint, respectively.

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 is the PWM command sent 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:

$$v_{\text{qp}} = v_{\text{previous}} + \frac{(u - F_r)t_{\text{loop}}}{m}, \tag{19}$$

where t_{loop} is the loop rate on the hardware. The end result is a control input for the nonlinear dynamics that will guarantee the safety via hard constraint and adaptively use minimum effort to adjust the velocity of the vehicle for both good comfort and tracking performances.

IV. Experimental Realization

In order to validate the proposed CBF-CLF QP controller on the test bed discussed in Sect. II, hardware-software interface along with a high level controller that mathematically calculates the solution to the ACC QP online, are required. In other words, three major requirements for the experimental realization are: sensing, actuation and the embedded level computing.

Sensing. To address the ACC problem specifications, as discussed in Sect. III, the speed of the cars and the distance between them are to be fed into the controller via various sensors. Velocity measurements of the following car in experiment are achieved through the use of a Hall effect sensor, mounted on the wheel hub of the front wheel (see Fig. 2) with two small magnets placed 180 degrees apart on the inside of the same wheel. Common measurements of the headway distance in production vehicles is through radar or lidar [25]. However, taking advantage of the special boom setup, the relative distance between two cars can be measured by the magnetic incremental encoder mounted on the central shaft.

Actuation. The system considered has a single control output: velocity. The electric car used as the following car (Fig. 2) has a three phase BLDC motor that is governed by pulse width modulation (PWM) signals sent to the Electronic Speed Control (ESC) unit which converts it into a three phase voltage. The electric car does not have a separate actuator to apply a braking force to the wheel, therefore the velocity of the car is regulated through positive wheel force and resistive forces only.

Embedded Computing. Realization of the proposed controller has been divided into two stages: a high-level controller which is running ROS (Robotic Operating System) on Ubuntu, and a low-level controller realized by an Arduino DUE board and the ESC on the car. Both high- and low-level controllers are running on the UDOO board with an embedded Arduino board, which is powered by a quad core processor and, therefore, has the ability to achieve the online control requirements. In particular, the sampling rate for the ROS master is set to be $200\ Hz$.

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 (t_{loop}) using loop rate;
    if Error in Calculation **then**
        Report Errors and Stop QP calculation;
    else
        if Data recieved from any sensor **then**
            Initialize the internal velocity for the QP;
            Convert relative distance value into relative velocity (m/s);
            Calculate lead car velocity by finite differencing;
            Set up parameters for QP;
            Calculate torque (F_w) via *CLF-BCF* QP.
            if Barrier function < 0 **then**
                Take $v_{qp} = 0$ to simulate braking on the car;
            else
                Calculate v_{qp} 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 on the processor. It is mounted inside the car and solves the optimization problem (ACC QP), online. The controller has been coded as a ROS Node in C++ to improve the efficiency of execution as well as record data being generated during the tests. The resistive force as mentioned in (2) uses average coefficients derived by testing on production cars, so when implementing on scaled model cars, we scale the equation by the same factor as the scale of the car. In real world scenarios, as seen in CCC, the aerodynamic drag might cause high resistive forces on the car. Which can be compensated by modifying the force equation coefficients (2) according to the respective car model. Algorithm 1 presents the pseudo-code of the embedded implementation at the high level.

Low-Level Controller: Acting at the low-level, this controller has less computation so it runs at a faster frequency of $57600\ BaudRate$. The algorithm runs on the Arduino DUE board as a ROS Node communicating with the ROS Master. This allows us to connect motors and sensors creating an interface between the hardware and software. It is responsible for all the communication to and from the actuators and sensors. To summarize the functions of this controller, we present the pseudo-code running on the Arduino in Algorithm 2.

Algorithm 2 Arduino Module- Low level

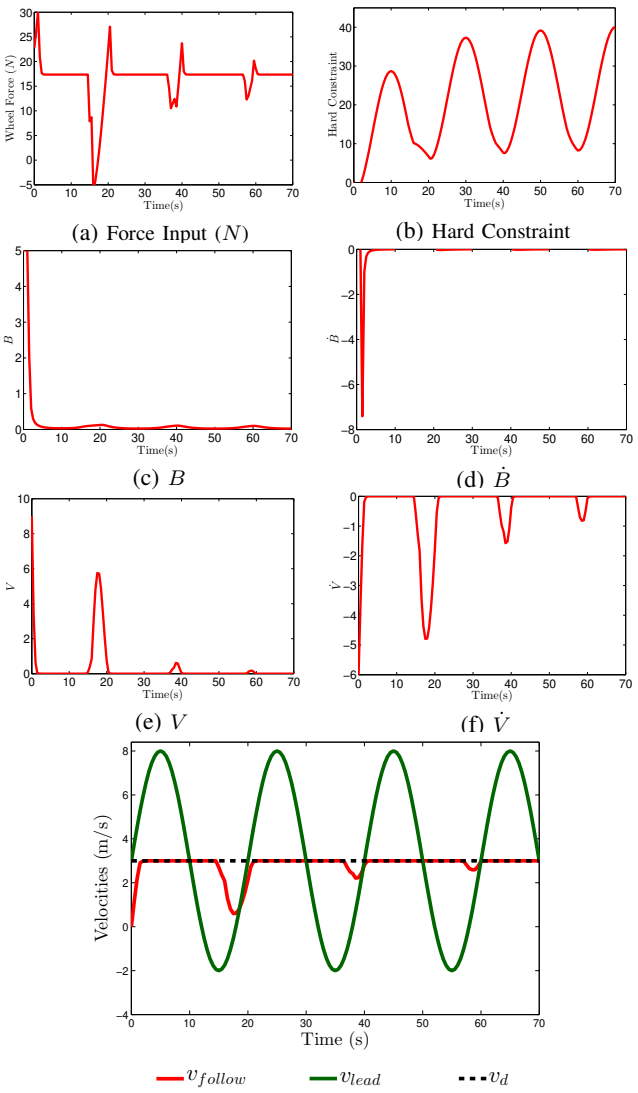
```
Compile Arduino code using IDE;
Communicate with ROSSERIAL node on ROS Master;
Enable Electronic Speed Control (ESC) for the car;
repeat
    Set parameters for low-level controller;
until All communication is established
while ( ROSSERIAL == Running ) do
    Initialize all GPIO pins;
    Define pins 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 in (m/s);
        Convert encoder data into relative distance in (m);
        Publish calculated data on ROS Master;
        Subscribe for current v and v_qp data on the Master;
        Calculate error between v and v_qp;
        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;
```

V. Results

In this section, the simulated performance and the experimental performance of the CBF-CLF QP controller are analyzed side by side. Importantly, we establish the successful experimental implementation of the controller (ACC QP) as exhibited in [1].

A. Simulation Results

As discussed in Sect. III, previous results [2] considered the case when the lead care velocity was constant. On the contrary, this paper constructed a control scheme that allows for a lead care with time-varying velocity. To validate this presented framework, the CBF-CLF QP controller was first tested in an idealized environment: simulation. The lead car velocity was chosen to be sinusoidal waveform, given by $v_l(t) = 3 + 5\sin(0.1\pi t)$. As shown in Fig. 4, the control objectives were accomplished (i.e., safety is always maintained while the desired speed is achieved whenever possible). In particular, when the system starts at initial conditions $(x_0, z_0) \in \mathcal{C}$, close to the boundary $\partial\mathcal{C}$, the hard constraint (Fig. 4b) forces the CBF constraint to modulate the speed of the following car. Therefore, with a high value of B and $\dot{B}$, the following car moves much slower than the lead car to maintain the safety imposed barrier. When the hard constraint increases (i.e. the relative distance is within a safe range) the soft constraint will influence the QP controller and yield a desired cruise velocity regulation. The acceleration



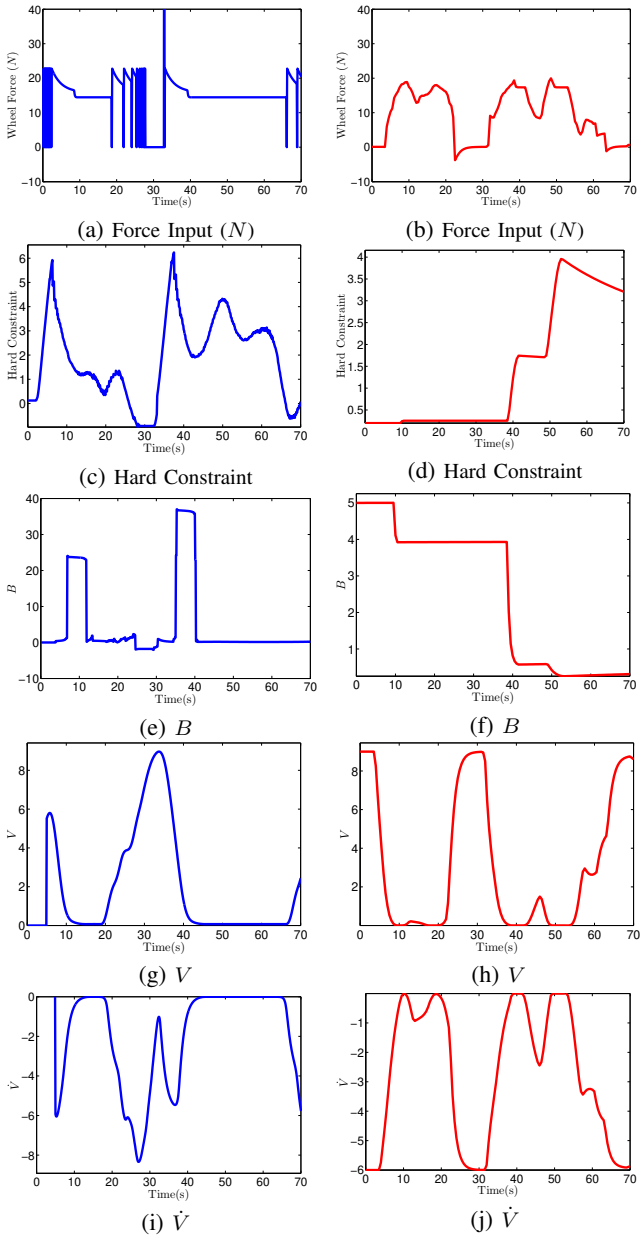
(a) Force Input (N) (b) Hard Constraint (c) B (d) $\dot{B}$ (e) V (f) $\dot{V}$ (g) Velocity

Fig. 4: Simulation results with sinusoidal lead car velocity profile.

bounds restrain the speed modulation of the following car to a smooth profile. Hence the simulation results verify the validity of the proposed controller.

B. Experimental Results

Now, we describe the results of experimental implementation of the CBF-CLF QP controller (ACC QP), and present the corresponding experimental results, and compare them to simulation results that utilize the experimentally recorded lead car velocity profile. In particular, the velocity data for the manually controlled lead car is collected from the boom encoder and utilized in simulation (through the fitting of a high order polynomial to the data). This serves as the connection point between the simulation and experimental results as shown in Fig. 6. The consequent simulation results can be seen alongside the experimental data in Fig. 5 and Fig. 6 to provide concrete comparisons.



(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) $\dot{V}$

Fig. 5: Experimental results (left column) and simulation results (right column) for all of the relevant variables.

In Fig. 5, all of the relevant mathematical quantities are shown to allow for direct comparison between experimental plots on the left and simulation plots on the right. Overall, good agreement is shown between the two cases, subject to some notable differences. In the case of force input we can see some discrepancies, which presumably accounts for the lack of system model specifications—yet the force input magnitudes are similar in both cases. Some calibration errors and delays in sensing that propagate through the system, introduce a slight bias in the behavior of the hard constraint and the barrier function, B. Even with these practical issues, the hard constraint is predominantly positive, indicating the proper enforcement of the safety constraints. Finally, good accordance is seen between the behavior of the control Lyapunov function, V, and its derivative, $\dot{V}$, indicating the ability to regulate speed when the control barrier function is inactive.

As seen in Fig. 6, the velocity of the following car is consistent between the simulation and experiment. Fig. 6a shows all the experimental velocities recorded during the tests. v_{qp}^{exp} is the velocity calculated from (ACC QP) using the one step forward integration method (19), 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 set desired velocity of the following car. Fig. 6a illustrates the effectiveness of the CLF based control constraint, which allows tracking of a set point velocity. Similar outcomes can be seen when considering the simulation results obtained by using the experimental lead car data as shown in Fig. 6b. The simulated velocity, v_{follow}^{sim}, is compared with the experimentally observed values, v_{follow}^{exp}. As expected, the simulation results achieve better velocity tracking, yet these results still accurately represents the behavior seen in experiments. Finally, comparing Fig. 6c with Fig. 6a, we see that the velocity of the following car is directly modulated by the relative distance and the lead car velocity. This allows us to conclude that the proposed control method, encoded by (ACC QP), has been successfully realized experimentally on scale-model cars.

VI. Conclusion

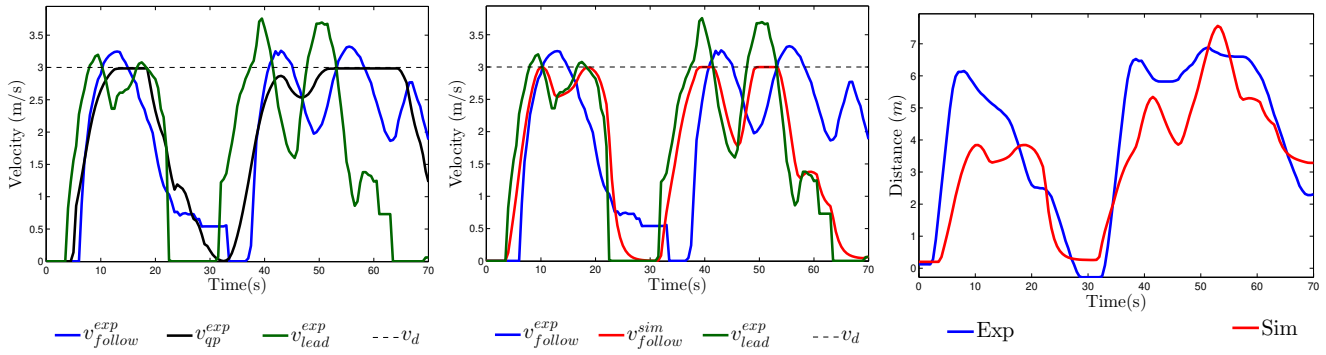
The main contribution of this paper is an experimental platform that is used to evaluate advanced controllers for ACC. The presented controller (ACC QP) was able to handle multiple objectives—safety, speed regulation and comfort—in a unified fashion. This online optimization based controller was realized experimentally, where it was shown to satisfy the safety specifications imposed by the control barrier function while achieving adaptive speed regulation as encoded by the control Lyapunov function. To avoid errors arising from hardware or communication time delays, the experimental test speed was appropriately scaled to the vehicle. The experimental results were then confirmed by comparing the data against the results of a simulation using the same lead car velocity. The comparison illustrated that the behavior of the experimental controller for the ACC problem corresponds with theory. The results of the control implementation, with failure modes indicating the safety critical nature of ACC, are available online as a video [1].

VII. Acknowledgment

The authors would like to thank Dr. Koushil Sreenath from Carnegie Mellon University for the discussions on barrier functions, Michael Zeagler for the construction of the mechanical setup and other members of AMBER Lab, specifically, Eric Cousineau, Jacob Reher, Ryan Sinnet, Shishir Kolathaya, Huihua Zhao, Johnathan Horn, Eric Ambrose, and Victor Parades for all the technical support.

References

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



(a) Experimental velocity tracking results (b) Simulated velocity tracking results (c) Relative Distance

Fig. 6: 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.

Fig. 7: Tiles displaying the changing distance of the lead car as seen from the perspective of the following car with the time stamp from the experiment. The tiles, and time stamps therein, can be compared to the relative distance plot in Fig. 6c.

[2] 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.

[3] A. D. Ames, J.W. Grizzle K. Galloway, and K. Sreenath. Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics. In *IEEE Transactions on Control, vol. 59, no. 4.*, pages 876–891, 2014.

[4] 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.

[5] C.V. Driel B.V. Arem and R. Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *Intelligent Transportation Systems, IEEE Transactions on*, 7(4):429–436, 2006.

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

[7] 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.

[8] R. A. Freeman and P. V. Kokotović. *Robust Nonlinear Control Design*. Birkhäuser, 1996.

[9] 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.

[10] St. Germann and R. Isermann. Nonlinear distance and cruise control for passenger cars. In *American Controls Conference*, June 1995.

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

[12] P. A. Ioannou and C. C. Chien. Autonomous intelligent crusie control. In *IEEE Transactions On Mechatronics, Vol. 42, No. 1*, Nov 1993.

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

[14] C-Y. Liang and H. Peng. Optimal adaptive cruise control with guaranteed string stability. *Vehicle System Dynamics*, 32(4-5):313–330, 1999.

[15] C-Y. Liang and H. Peng. String stability analysis of adaptive cruise controlled vehicles. *JSME International Journal Series C*, 43(3):671–677, 2000.

[16] 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.

[17] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. D. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Preliminary results on correct-by-construction control software synthesis for adaptive cruise control. In *Conference on Decision and Control*, Dec 2014.

[18] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

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

[20] S-I. Sakai, H. Sado, and Y. Hori. Motion control in an electric vehicle with four independently driven in-wheel motors. In *IEEE/ASME Transactions On Mechatronics, VOL. 4, NO. 1*, March 1999.

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

[22] S. E. Shladover, C. A. Desoer, J. K. Hedrick, M. Tomizuka, J. Walrand, W-B Zhang, D. H. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown. Automatic vehicle control developments in the PATH program. In *IEEE Transactions On Vehicular Technology, Vol. 40, No. 1*, Feb 1991.

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

[24] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *Intelligent Transportation Systems, IEEE Transactions on*, 4(3):143–153, 2003.

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

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