2017 IEEE Conference on Control Technology and Applications (CCTA)
August 27-30, 2017. Kohala Coast, Hawai'i, USA

# Realizing Simultaneous Lane Keeping and Adaptive Speed Regulation on Accessible Mobile Robot Testbeds

Xiangru Xu, Thomas Waters, Daniel Pickem, Paul Glotfelter, Magnus Egerstedt, Paulo Tabuada, Jessy W. Grizzle and Aaron D. Ames

*Abstract*— **This paper presents experimental results on novel robot testbeds that allow the evaluation of the simultaneous implementation of adaptive speed regulation and lane keeping in a safe, education-centric, and inexpensive manner. The underlying algorithms are based on a control Lyapunov function for performance, a control barrier function for safety, and a real-time quadratic program for mediating the conflicting demands of performance and safety. The Robotarium used for this work allows students, as well as researchers less experienced with hardware, to experiment with advanced control concepts in a safe and standardized environment.**

## I. Introduction

Designing controllers that enforce different and sometimes conflicting objectives is a recurring challenge in many real systems, such as robotic and automotive systems. This is especially crucial for systems in which stringent safety-critical specifications must be guaranteed at all times, while also providing the performance expected by a user [1]. Advanced Driver Assistance Systems (ADAS) are a prime example, where passenger and commercial vehicles are being outfitted with multiple safety or comfort modules [2]. Lane keeping, for example, controls a vehicle's steering to keep it within its lane, while adaptive cruise control regulates a vehicle's speed to a user-set constant when there is no preceding vehicle in the lane, and maintains a safe following distance when a preceding vehicle is detected [3], [4]. Because ADAS control modules can be activated concurrently in today's vehicles, designing provably correct control software for the simultaneous operation of two or more control modules is crucial and has attracted considerable attention (see [5], [6], [7] and references therein).

Set invariance is a widely used means to both specify and prove safety properties [8], which is often established through the use of barrier functions (also known as barrier certificates). The barrier function has proved popular because it provides a certificate of set invariance without the difficult task of computing a system's reachable set [9], [10]. Inspired by the automotive safety-control problems, a control barrier function (CBF) is proposed in [11], which extends the normal barrier function condition to only requiring a single sub-level set to be controlled invariant, and extends barrier functions from ODEs to control systems. When CBFs are combined with control Lyapunov functions (CLFs) representing a control objective through a quadratic programming (QP) framework, families of control policies that guarantee safety can be designed; moreover, the control objective is mediated whenever safety and performance are in conflict. This CBF-CLF-QP approach has been used in various applications such as automotive safety-critical systems, bipedal robots, and multi-agent systems [12], [13], [14], [15], [16], [17].

This paper uses the Khepera robot testbed [18] and the Robotarium testbed [19] to explore the real-time hardware implementation of adaptive speed regulation and lane keeping simultaneously using the CBF-CLF-QP approach. Exploring a hardware implementation of CBF-CLF-QPs allows us to check for potential challenges that arise due to modeling errors, sensor sampling rates, or accuracy limitations of real systems, paving the way for future testing of the algorithms in full-sized vehicles. Furthermore, the implementation serves as an educational example to show how the Robotarium allows students to work with a reasonably sophisticated safety-critical control problem in a (personally) safe and relatively inexpensive setting. For comparison purposes, the Khepera testbed, which uses a costly OptiTrack camera system and Khepera robots, is also used to implement the CBF-CLF-QP algorithms.

The main contributions of this paper include:

- the hardware implementation of a provably correct controller that achieves adaptive speed regulation and lane keeping simultaneously;
- the comparison of experimental results on the Robotarium with results on a more costly Khepera system;
- illustrating a setting where students can gain hands-on familiarity with a safety-critical system in an inexpensive and safe manner.

The remainder of the paper is structured as follows. Section II introduces the robot model and testbeds that are used in the experiments. Section III presents the CBF-CLF-QP control method and Section IV introduces the implementation details in the experiments. The experimental results and their comparison with the simulation results are given in Section V, and finally, some conclusions in Section VI.

This work is supported by NSF grants CNS-1239037, CNS-1239055 and CNS-1544332.

X. Xu and J. W. Grizzle are with the Depart. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, {xuxiangr,grizzle}@umich.edu. P. Tabuada is with the Depart. of Electrical Engineering, UCLA, CA, tabuada@ucla.edu. W. Thomas is with the School of Mechanical Engineering, D. Pickem, P. Glotfelter and M. Egerstedt are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, {thomas.waters,fdaniel.pickem, paul.glotfelter,magnus}@gatech.edu. A. D. Ames is with the Depart. of Mechanical and Civil Engineering, California Institute of Technology, Pasadena CA, ames@cds.caltech.edu.

## II. Robot Model and Experimental Platforms

In this section, we first present the robot model that is utilized in the experiments. Next, we introduce two experimental platforms, namely, the Khepera robot testbed and the Robotarium, which are used to demonstrate the adaptive speed regulation and lane keeping safety algorithms.

### A. The Unicycle Robot Model

The standard unicycle model is given in (1) as

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix} v\cos(\psi)\\ v\sin(\psi)\\ \omega\end{bmatrix}. \tag{1}$$

Figure 1 shows the coordinates $(x, y), \psi, v, \omega$ representing the 2D position, the orientation, and the longitudinal and angular velocities of the robot, respectively.

When the longitudinal force and the angular torque are taken as inputs to the model, we have $\dot{v} = \frac{u_l}{m}$ and $\dot{\omega} = \frac{u_a}{I_z}$ where $u_l$ and $u_a$ are the force and torque control inputs, respectively, $I_z$ is the moment of inertia about the z-axis, and $m$ is the mass of the robot. The relative degree of the $x$ and $y$ states for $u_l$ and $u_a$ are not equal, which is inconvenient for the input-output feedback linearization that will be explained later. To overcome this, we follow [20], [21] and references therein and choose a point of interest located a distance $a > 0$ forward of the wheel axis, as shown in Figure 1. Noting that the change of coordinates modifies the derivative of the longitudinal velocity term, with the addition of a centripetal acceleration term represented by $a\omega^2$, we arrive at the unicycle model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{v}\\ \dot{\psi}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} v\cos(\psi) - a\omega\sin(\psi)\\ v\sin(\psi) + a\omega\cos(\psi)\\ \frac{u_l}{m} - a\omega^2\\ \omega\\ \frac{u_a}{I_z}\end{bmatrix}. \tag{2}$$

In what follows, we denote $\mathbf{x} = [x, y, v, \psi, \omega]^\top$.

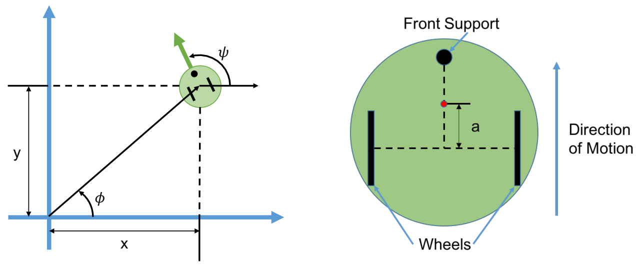

Fig. 1: (left) States of the unicycle robot. (right) Modified point of interest.

***Remark 1:*** Compared with the model in [20], [21], which has the longitudinal and angular reference velocities as the input, the model (2) has the longitudinal force and the angular torque as input, which are used in our controller design shown later.

The angular position of the robot with respect to the origin is denoted by $\phi$ (see Figure 1) and is useful for the path tracking algorithm used in the experiments. Clearly, $\phi = \operatorname{atan}(y/x)$ and its time derivative is

$$\dot{\phi} = \frac{\sqrt{(a\omega\cos(\phi-\psi) - v\sin(\phi-\psi))^2}}{\sqrt{x^2+y^2}}. \tag{3}$$

In this paper, the robots move along a desired path defined in polar coordinates by

$$R_{path} = R + b\sin(n\phi). \tag{4}$$

Here, $R$ is the mean radius of the path, $b$ is the amplitude of the sinusoidal variation of the path, and $n$ is the number of periods in the path.

### B. Experimental Testbeds

This subsection introduces the two testbeds that are used for the experiments: the Khepera robots and the Robotarium.

*1) Khepera Robot Testbed:* The Khepera robot testbed was provided by the GRITS lab at the Georgia Institute of Technology [18]. A Khepera robot is shown in Figure 2.

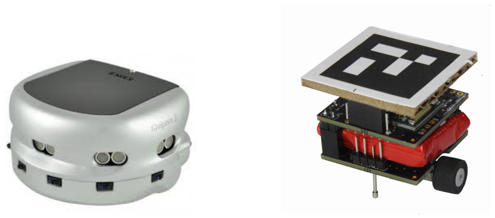

Fig. 2: (left) Khepera III robot, (right) GRITSBot from the Robotarium.

**Sensing**. A model-based solution to the speed regulation and lane keeping control problems requires knowledge of each robot's position, orientation, and velocity. In the Khepera robot testbed, the position and orientation data are collected using 10 OptiTrack S250e motion capture cameras.

**Actuation**. The Khepera III robot uses two DC motors, where each motor actuates a single wheel in the differential drive system. The two motors are powered by a shared 7.4V, 1350mah LiPo battery. The input signal to each motor corresponds to shaft speed and is transmitted to the motor via pulse-width modulation (PWM). For later use, it is important to note that the PWM signal, because it commands motor shaft speed, does not correspond to either the force or torque control input used in the model. The force and torque inputs from the adaptive speed regulation and lane keeping controllers will be integrated through the model to produce equivalent motor speeds, which will then be converted to a PWM-command signal for use in the control loop and the embedded electronics.

**Embedded Computing**. Each Khepera III robot is equipped with a 600MHz ARM processor and 128Mb RAM, embedded Linux, and a WiFi module for communicating via a wireless router. Control inputs are computed on a centralized computer and sent to the robot via WiFi.

*2) The Robotarium:* The Robotarium was conceived because multi-robot testbeds constitute an integral and essential part of the multi-robot research cycle, yet they can be prohibitively expensive, complex, and time-consuming to develop, operate, and maintain. As a swarm-robotic testbed that can be accessed remotely through a web interface[1], the Robotarium gives users the flexibility to test a variety of multi-robot algorithms (see [19],[14]). In particular the Robotarium tackles the challenge of robust, long-term, and safe operation of large groups of robots with minimal operator intervention and maintenance. In its current implementation, the Robotarium contains 20 miniature ground robots, the GRITSBots shown in Figure 2 (see [22]). These inexpensive, differential-drive robots simplify the operation and maintenance of the Robotarium through features such as: (i) automated registration with a server and overhead tracking system, (ii) automatic battery charging, and (iii) wireless (re)programming.

Unlike the Khepera testbed, the Robotarium also offers a MATLAB-based simulator that closely approximates the behavior of the GRITSBots through a parameterized unicycle model and a model of measurement latency. Therefore, controls code developed using the Robotariums simulator can be deployed onto the Robotarium with no or minor modifications. This simulator gives users the ability to rapidly iterate through simulation and testing phases, allowing for a straightforward implentation process.

**Sensing**. Similar to the Khepera testbed, the Robotarium relies on centralized overhead tracking. Instead of an OptiTrack system, however, the Robotarium employs a web camera-based setup that uses a single Microsoft Lifecam HD camera running at an update rate of 30 Hz and a resolution of 1280x720 pixels. ArUco tags[2] attached to each GRITSBot allows the system to determine the robot's position and orientation.

**Actuation**. A GRITSBot is equipped with two miniature stepper motors, each actuating a single wheel. The advantage of stepper motors is that their velocity can be determined without encoders by simply counting the number of steps a motor has moved. The additional complexity of controlling stepper motors is handled via a custom motor board that houses an Atmega168 microcontroller and executes a velocity controller onboard. Each GRITSBot is powered by a single 3.7V, 400 mAh LiPo battery resulting in a runtime of up to 40 minutes on a single charge.

**Embedded Computing**. A GRITSBot is equipped with an ESP8266, a WiFi-enabled microcontroller equipped with 160 KB of RAM running at 160 MHz. Given these specifications, a GRITSBot is not capable of hosting an operating system, yet it is powerful enough to handle wireless communication, pose estimation, low-level control, as well as high-level behaviors. Similar to the Khepera-based setup, control inputs are computed on a centralized computer and sent to the robot via WiFi.

## III. Autonomous Automotive Control Methods

The approach to achieve lane keeping and adaptive speed regulation simultaneously is briefly introduced in this section. By encoding the safety specifications as CBF conditions and the performance as CLF conditions with relaxation parameters, the control policy is generated by solving an online QP that combines CBFs and CLFs.

### A. Control Barrier Functions

Some basic results in [12] are reviewed first. Given a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, define a closed set $\mathcal{C}$, which is assumed to be nonempty and without isolated points, by

$$\mathcal{C} = \{x \in \mathbb{R}^n : h(x) \geq 0\}. \tag{5}$$

Consider an affine control system of the form

$$\dot{x} = f(x) + g(x)u, \tag{6}$$

with $f$ and $g$ locally Lipschitz continuous, $x \in \mathbb{R}^n$, and $u \in U \subset \mathbb{R}^m$.

***Definition 1:*** [12] Given a set $\mathcal{C} \subset \mathbb{R}^n$ defined by (5), the continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ is called a (zeroing) control barrier function defined on set $\mathcal{D}$ with $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$, if there exists a constant $\gamma > 0$ such that

$$\sup_{u\in U} \left[L_f h(x) + L_g h(x)u + \gamma h(x)\right] \geq 0,\ \forall x \in \mathcal{D}. \tag{7}$$

Given a CBF $h$, for all $x \in \mathcal{D}$, define the set

$$K_{\text{zcbf}}(x) = \{u \in U : L_f h(x) + L_g h(x)u + \gamma h(x) \geq 0\}. \tag{8}$$

The following result guarantees the forward invariance of $\mathcal{C}$ when inputs are selected from $K_{\text{zcbf}}(x)$.

***Theorem 1:*** [12] *Let $\mathcal{C} \subset \mathbb{R}^n$ be a set defined by* (5) *for a continuously differentiable function h. If h is a CBF on $\mathcal{D}$, then any locally Lipschitz continuous controller $u : \mathcal{C} \to U$ satisfying $\forall x \in \mathcal{D}$, $u(x) \in K_{\text{zcbf}}(x)$, will render the set $\mathcal{C}$ forward invariant.*

The unicycle model (2) can be expressed as an affine control system shown in (6), with $f(\mathbf{x}), \mathbf{g}(\mathbf{x})$ given in an obvious way.

**Adaptive Speed Regulation.** Similar to the adaptive cruise control on vehicles, in adaptive speed regulation of mobile robots, the following robot must always maintain a safe time-headway with the lead robot, and achieve a user-set longitudinal speed whenever possible.

While achieving the user-set speed is a soft constraint that will be discussed in the next subsection, maintaining a safe time-headway is a hard constraint, which can be expressed as $D \geq \tau v_f$ where $D$ is the distance between the lead and following robots, $v_f$ is the speed of the following robot, and $\tau$ is the minimum allowable time headway, in seconds, between the two robots. Therefore, the following CBF is chosen for this speed regulation safety specification,

$$h_{asr} = D - \tau v_f. \tag{9}$$

**Lane Keeping.** The objective of lane keeping is to keep the robots within its lane boundary. Therefore, the lane keeping specification for the robot can be expressed as $|y_{lat}| \leq d_{\max}$ where $y_{lat}$ represents the lateral displacement of the robot w.r.t. the desired path in road fixed coordinates,

[1]See www.robotarium.org

[2]ArUco is an OpenCV-based library for Augmented Reality applications.

and $d_{\max}$ is the width of the path. Different CBFs can be used, such as the one introduced in [23]

$$h_{lk} = d_{max} - sgn(v_{lat})y_{lat} - \frac{1}{2}\frac{v_{lat}^2}{a_{max}}, \tag{10}$$

where $sgn(\cdot)$ is the sign function, $a_{max}$ is the maximum allowable lateral acceleration and $v_{lat}$ is the lateral velocity of the robot in road-fixed coordinates, or the following CBF

$$h_{lk} = 1 - \frac{y_{lat}^2}{d_{max}^2} - \frac{1}{2}v_{lat}^2. \tag{11}$$

Both (10) and (11) ensure that $h_{lk} \geq 0$ implies $|y_{lat}| \leq d_{\max}$.

## B. Control Lyapunov Functions

While the safety specifications need to be respected at all times, there are three performance objectives that should be achieved as much as possible: 1) $v \to v_d$, where $v_d$ is the desired longitudinal velocity of the following robot; 2) $\omega \to 0$, which serves to create a smoother path along the course; 3) $(x, y) \to (R_{path}\cos(\phi), R_{path}\sin(\phi))$ where the right hand side is the tracking point in the desired path. To implement the performance objectives, the goal is to drive the following three outputs to zero:

$$\begin{aligned}\eta_1 &= v - v_d,\\ \eta_2 &= \omega,\\ \eta_3 &= \begin{bmatrix} x - R_{path}\cos(\phi) \\ y - R_{path}\sin(\phi)\end{bmatrix}.\end{aligned}$$

***Remark 2:*** It is interesting to point out that driving $\eta_2$ and $\eta_3$ to zero are contradictory objectives, since $\eta_2$ being zero requires the robot to move in a straight line while $\eta_3$ being zero requires the robot to track the desired path with a curved trajectory. We will show how these conflicting objectives are considered as "soft constraints" and are balanced in a QP framework by some relaxation variables in Subsection III-C, as well as simulation and experiment results in Section V.

For $i = 1, 2, 3$, to achieve exponential convergence of $\eta_i$ to zero (irrespective of other outputs), we utilize a special class of control Lyapunov functions $V(x)$ termed *exponentially stabilizing control Lyapunov function (ES-CLF)* [24]. For the outputs $\eta_1, \eta_2$, the control Lyapunov functions are taken as $V_1(x) = (v - v_d)^2, V_2(x) = \omega^2$. For the output $\eta_3$, because

$$\dot{\eta}_3 = \begin{bmatrix} \dot{x} - \dot{R}_{path}\cos(\phi) + \dot{\phi}R_{path}\sin(\phi) \\ \dot{y} - \dot{R}_{path}\sin(\phi) - \dot{\phi}R_{path}\cos(\phi)\end{bmatrix},$$

where $\dot{R}_{path} = nb\dot{\phi}\cos(n\phi)$ and $\dot{\phi}$ is given in (3), the output $\eta_3$ has relative degree 2. Implementing routine input-output linearization and using the technique in [24] yields the CLF $V_3(x) = [\eta_3^\top, \dot{\eta}_3^\top]P[\eta_3^\top, \dot{\eta}_3^\top]^\top$, where

$$P = \begin{bmatrix} \sqrt{3} & 0 & 1 & 0 \\ 0 & \sqrt{3} & 0 & 1 \\ 1 & 0 & \sqrt{3} & 0 \\ 0 & 1 & 0 & \sqrt{3}\end{bmatrix}.$$

For each $V_i,\ i = 1, 2, 3$, the set of control inputs that exponentially stabilizes $\eta_i$ is given as

$$K_i(\mathbf{x}) = \{u | L_f V_i(\mathbf{x}) + L_g V_i(\mathbf{x})u + c_i V_i(\mathbf{x}) \leq 0\} \tag{12}$$

where $c_i (i = 1, 2, 3)$ is a positive constant, which is a tunable parameter specifying the convergence rate.

***Remark 3:*** It is impossible to input/output linearize the robot system (2) for the output $[\eta_1, \eta_2, \eta_3^\top]^\top$, because there are only two inputs. Since the CLF condition will be treated as "soft constraint", different CLFs can be designed separately to achieve their respective objective as we do above.

## C. CBF-CLF-based Quadratic Programs

The CLFs and CBFs developed in preceding subsections are unified in a QP to generate a min-norm controller:

$$\begin{aligned}\mathbf{u}^*(\mathbf{x}) = &\underset{\mathbf{u}=[u_l, u_a, \delta_1, \delta_2, \delta_3]^\top}{\operatorname{argmin}} \quad \mathbf{u}^T H \mathbf{u} \\ s.t. \quad & A_{\text{clf}}^i(\mathbf{x})\mathbf{u} \leq b_{\text{clf}}^i(\mathbf{x}),\ i = 1, 2, 3, \\ & A_{\text{asr}}(\mathbf{x})\mathbf{u} \leq b_{\text{asr}}(\mathbf{x}), \\ & A_{\text{lk}}(\mathbf{x})\mathbf{u} \leq b_{\text{lk}}(\mathbf{x}),\end{aligned} \tag{13}$$

where $A_{\text{clf}}^1(\mathbf{x}) = [L_g V_1(\mathbf{x}), -1, 0, 0]$, $A_{\text{clf}}^2(\mathbf{x}) = [L_g V_2(\mathbf{x}), 0, -1, 0]$, $A_{\text{clf}}^3(\mathbf{x}) = [L_g V_3(\mathbf{x}), 0, 0, -1]$, $b_{\text{clf}}^i(\mathbf{x}) = -L_f V_i(\mathbf{x}) - c_i V_i(\mathbf{x}),\ i = 1, 2, 3$, $A_{\text{asr}}(\mathbf{x}) = [L_g h_{asr}(\mathbf{x}), 0, 0, 0]$, $b_{\text{asr}}(\mathbf{x}) = -L_g h_{asr}(\mathbf{x}) - \gamma_1 h_{asr}(\mathbf{x})$, $A_{\text{lk}}(\mathbf{x}) = [L_g h_{lk}(\mathbf{x}), 0, 0, 0]$, $b_{\text{lk}}(\mathbf{x}) = -L_g h_{lk}(\mathbf{x}) - \gamma_2 h_{lk}(\mathbf{x})$, $H := \text{diag}\{p_1, ..., p_5\} \in \mathbb{R}^{5\times 5}$ are the weight matrix with penalty weight $p_i > 0$, $\gamma_1, \gamma_2$ are given positive constants, and $\delta_i \geq 0 (i = 1, 2, 3)$ are relaxation parameters. These relaxation variables enable us to have controllers with different, potentially conflicting, objectives, whose priority can be changed by tuning $p_i,\ i = 3, 4, 5$, with larger value implying more priority on that objective.

The optimization problem (13) can be solved by QP solvers such as the *quadprog* function in MATLAB. The inputs $u_l, u_a$ generated are applied to (2) ensuring the robot always satisfies the safety specifications and achieves the performance objectives when $\delta_i$ are sufficiently small.

## IV. Controller Implementation

This section explains the implementation of the adaptive speed regulation and lane keeping control algorithms on the Khepera and Robotarium testbeds. The implementation differs from that of a standard vehicle because the actuators are not "force-torque-based", but rather, "speed-based". The implementations methods for both the Khepera robots and the Robotarium follow the same general steps shown in Figure 3, with the exception of a few noted differences.

To start, pose data on both the Khepera testbed and the Robotarium are acquired through an overhead tracking system and include the 2D position and orientation of each robot. While the Khepera testbed relies on the proprietary OptiTrack motion capture system to provide pose data using reflective infrared markers (at 50 Hz), the Robotarium uses a single web camera and an OpenCV-based tag tracker in conjunction with ArUco tags (at 30 Hz). The Robotarium's tracker uses open-source software packages and is also freely available at `https://github.com/robotarium`.

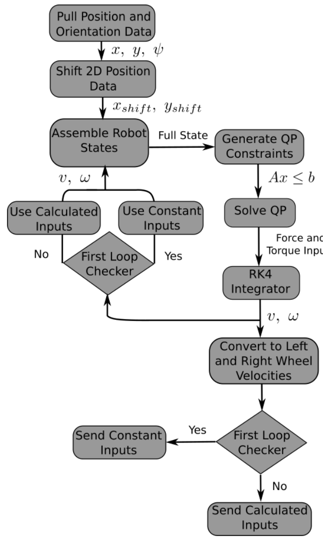

Fig. 3: Flowchart describing the control generation loop in the experimental implementation.

The acquired 2D position data represent the center position of the robot and these values must be shifted in order to coincide with the modified unicycle model described in Section II. This shift is done according to $x_{shift} = x + a\cos(\psi), y_{shift} = y + a\sin(\psi)$. Following the shift of coordinates, the states for each robot are assembled in the order shown in (2). The 2D position states, $x$ and $y$, are taken from the shift calculation in the previous step, while $\psi$ is drawn directly from the data acquisition hardware and $\phi$ is calculated from the position data. Longitudinal velocity $v$ and angular rate $\omega$ are taken from the velocity and angular velocity control inputs sent to the robots in the previous loop. In order to avoid a singularity in the CLF based controller on the first loop, the longitudinal velocities of both robots are set to their desired values, $v_{d_l}$ and $v_{d_f}$, and the angular velocities are set to zero. This initialization causes the robots to have a nonzero positive velocity before the QP-based controller takes full control.

The assembled states $\mathbf{x}$ are used to calculate the matrices $A^i_{\text{clf}}, b^i_{\text{clf}}$, $A_{\text{asr}}, b_{\text{asr}}$, $A_{\text{lk}}, b_{\text{lk}}$ of the QP (13) in Section III. We use MATLAB's *quadprog* function to solve the QP (13) for the force and torque inputs $u_l, u_a$ in real time. Because both the Khepera and Robotarium robots receive longitudinal and angular velocities as inputs, the force and torque control inputs are integrated using the model's kinematics and a fourth-order Runge-Kutta integration method.

Linear and angular velocity inputs computed by solving the CBF-CLF-QP (13) are converted to wheel velocities and sent to the robots via WiFi. As noted, in the initial loop, constant velocity commands are sent to avoid a controller singularity. Both the Khepera-based testbed and the Robotarium rely on Matlab-based APIs to send wheel velocity commands via UDP sockets to the robots. While velocity updates are sent to the Khepera robots at 50 Hz, the Robotarium's robots receive updates at 30 Hz , which is the update rate limitation imposed by the web camera.

## V. Experimental Results

In this section, we will demonstrate the effectiveness of the CBF-CLF-QP controller through experiments on Khepera robots and the Robotarium.

In order to show that the QP framework can deal with different objectives while always ensuing the safety specifications, we let the path tracking controller for the following robot be turned off for a period of time during the experiments. Specifically, the constraint with $V_3$ is removed from the QP (13) when the "off" mode is conducted, and added to the QP (13) again when the "on" mode is conducted, with all the other constraints kept the same. By doing this, we simulate the robot attemping to leave the lane.

The parameters for all experiments are shown in Table I, where $R, b, n$ are the parameters of the desired path defined by (4), $d_{max}$ is the width of the lane, $\tau$ is the time-headway in (9) and $v_{d_f}$ and $v_{d_l}$ are the desired velocities for the following and lead robots, respectively. For each experiment, the initial conditions are the same: the following robot starts at the position $(x, y) = (R, 0)$, with the lead robot positioned ahead by 25% to 50% of a path revolution, and the robots are oriented tangent to the path at their starting positions with a small longitudinal and zero angular velocity.

TABLE I: Experiment Parameters

| Para. | $R$ | $b$ | $n$ | $d_{\max}$ | $\tau$ | $v_{d_f}$ | $v_{d_l}$ |
|---|---|---|---|---|---|---|---|
| Fig.6 | 0.9 | 0.23 | 3 | 0.15 | 1.8 | 0.2 | 0.1 |
| Fig.9 | 0.25 | 0.06 | 3 | 0.04 | 3 | 0.075 | 0.05 |
| Unit | m | m | - | m | s | $m/s$ | $m/s$ |

### A. Experiments on Khepera Robots

This subsection summarizes the execution of the on/off path tracking experiments on the Khepera robot testbed, where CBFs $h_{asr}$ in (9) and $h_{lk}$ in (11) are used.

Figure 4 shows the value of CBFs $h_{asr}$ and $h_{lk}$ of the following robot during the experiment, with the simulation results depicted as well, which are run under the same conditions. Here, the path tracking controller turns off at $t = 20s$ and resumes at $t = 45s$. As can be seen from Figures 4, both CBFs are positive for all time, which means that the safety specifications are always satisfied.

Figure 5 shows the value of CLFs $V_1, V_2, V_3$ for the same experiment and simulation, where penalty weights $p_3 = 10^5$, $p_4 = 1$, and $p_5 = 10^3$ are used such that the controller put more emphasis on $V_1$ (achieving the desired speed) and $V_3$ (tracking the path) while less on $V_2$ (reducing the angular velocity). As can be seen from Figure 5, before 20 seconds, the values of $V_1, V_2, V_3$ are quite smooth; when the tracking controller turns off at $t = 20$, the value of $V_2, V_3$ fluctuates quite a bit since the penalty weight on $V_2$ is small and removing the constraint of $V_3$ from the QP poses no restriction on $V_3$ during this period; when the tracking controller turns on again, the value of $V_1, V_2, V_3$ become smooth again. The mismatch between the experimental and

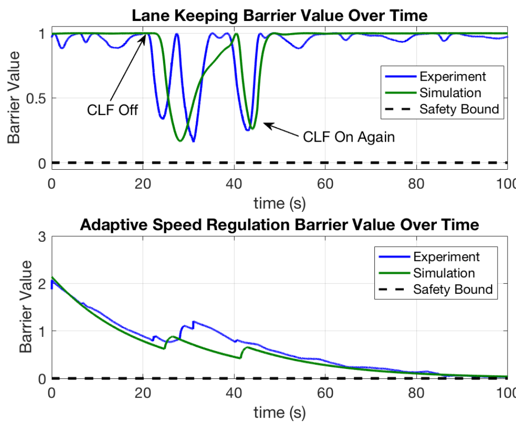

Fig. 4: Value of CBFs in the experiment and simulation on Khepera robots. (top) Value of $h_{lk}$ where positiveness implies that the robot is within the boundary. (bottom) Value of $h_{asr}$ where non-negativeness implies the specification $D \geq \tau v_f$ is satisfied.

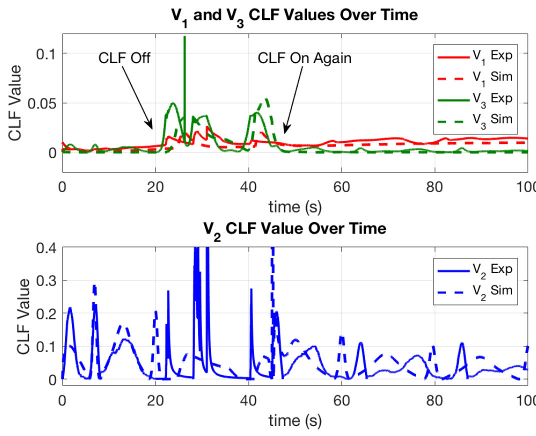

Fig. 5: Value of CLFs in the experiment and simulation on Khepera robots. (top) Value of $V_1, V_3$. (bottom) Value of $V_2$.

simulation data in Figure 4 and Figure 5 can be attributed to calibration and modeling errors.

Figure 6 shows the following Khepera robot's trajectories based on the experimental data and one snapshot of the experiment. It can be seen that even when the tracking objective is removed at that point, the robot is repelled back to the lane when it attempts to leave due to the constraint of the lane keeping CBF.

## B. Experiments on Robotarium

This subsection summarizes the execution of the on/off path tracking experiments on the Robotarium testbed, where CBFs $h_{asr}$ in (9) and $h_{lk}$ in (10) are used.

Figure 7 shows the value of CBFs $h_{asr}$ and $h_{lk}$ of the following robots during the Robotarium experiment, with the corresponding simulation results depicted as well. The path tracking controller turns off at $t = 10s$ and resumes at $t = 42s$. As we can see from Figure 7, both CBFs are

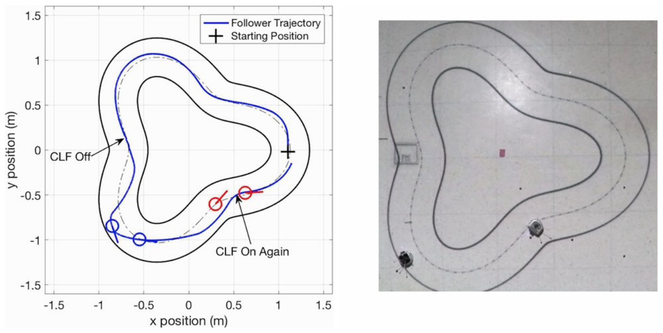

Fig. 6: (left) Trajectories of the following Khepera robot (blue line) during the on/off path tracking experiment. (right) Snapshot of the Khepera experiment during the off mode.

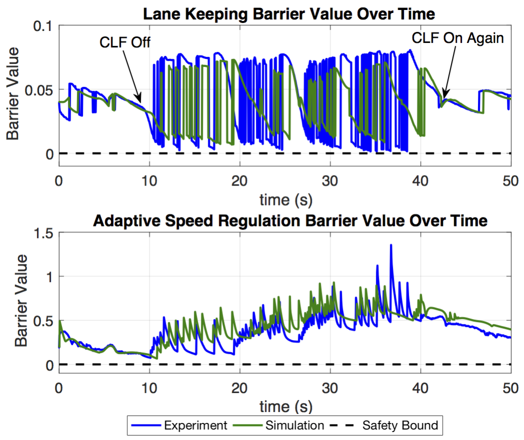

Fig. 7: Value of CBFs in the experiment and simulation on Robotarium with on/off path tracking CLF. (top) Value of $h_{lk}$ where positiveness implies satisfaction. (bottom) Value of $h_{asr}$ where non-negativeness implies satisfaction.

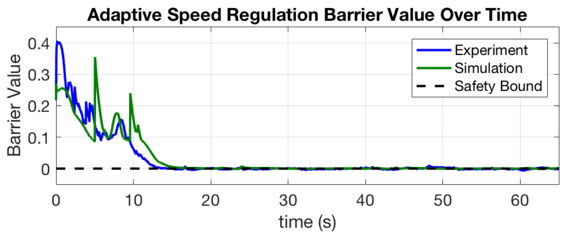

Fig. 8: Value of $h_{asr}$ in the Robotarium experiment and simulation when the path tracking controller is turned on for all time. Non-negativeness of $h_{asr}$ means satisfaction of the adaptive speed regulation specification.

positive for all time, which means that the lane keeping and adaptive speed regulation specifications are always satisfied. Compared with the results on the Khepera robots in Figure 4, the value of $h_{asr}$ and $h_{lk}$ here are both noisier. This difference is likely due to the size differences between the two testbeds and the fact that the Robotarium runs at a lower update rate (30Hz) than the Khepera testbed (50Hz).

As a comparison, Figure 8 shows the value of CBF $h_{asr}$ when the path tracking controller is turned on for the entire experiment, with the corresponding simulation results depicted as well. As one can see, with given model and calibration errors, $h_{asr}$ remains predominantly positive for all time, which means that the adaptive speed regulation specification is always satisfied. Particularly, when $h_{asr}$ is close to $0$, the minimum time headway $\tau$ is achieved.

Figure 9 shows the following robots' trajectories based on the experimental data on Robotarium as well as one snapshot of the experiment. It can be seen that the following robot approaches the lane boundary and is repelled from the boundary because of the lane keeping CBF.

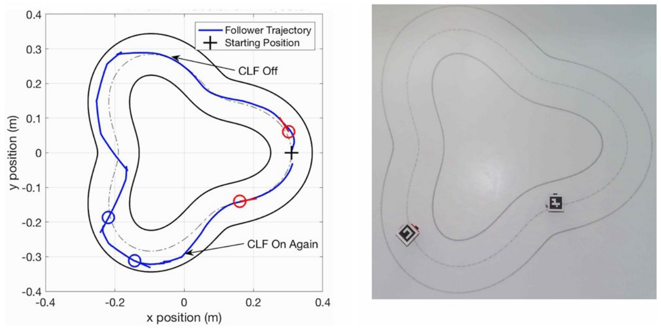

Fig. 9: (left) Trajectories of the following robot (blue line) on Robotarium during the on/off tracking experiment. (right) Snapshot of the Robotarium experiment during the off mode.

In addition to the results presented above, video for the "on/off" experiments can be found online in [25]. An additional "decaying" experiment, which is not presented here for the space limitation, can be also found in [25].

## VI. Conclusions

In this paper, the real-time implementation of lane keeping and adaptive speed regulation was experimentally evaluated on two robot testbeds, based on a CBF-CLF-QP approach. Our results showed the effectiveness of the CBF-CLF-QP framework for multi-objective controller design with safety constraints, and its potential for implementation on ADAS control software. These results were achieved on accessible mobile testbeds—a key advantage of this approach is that it provides students hands-on experience with rather sophisticated control software where safety, in the sense of formal methods, is a primary factor. Additional advantages include the low cost of the experiments, and in the case of the Robotarium, the fact that multiple groups of faculty and students can compare results on a common platform. The hope is that this will allow for the rapid prototyping and deployment of safety-critical controllers among a wide audience of researchers.

## References

[1] J. C. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24rd International Conference on Software Engineering*. IEEE, 2002, pp. 547–550.

[2] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: approaches, lessons and challenges," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4649–4672, 2010.

[3] K. L. Talvala, K. Kritayakirana, and J. C. Gerdes, "Pushing the limits: From lanekeeping to autonomous racing," *Annual Reviews in Control*, vol. 35, no. 1, pp. 137–148, 2011.

[4] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, 2003.

[5] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1294–1307, 2016.

[6] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *arXiv:1609.06807*, 2016.

[7] S. Dai and X. Koutsoukos, "Safety analysis of automotive control systems using multi-modal port-hamiltonian systems," in *Hybrid Systems: Computation and Control*, 2016, pp. 105–114.

[8] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[9] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Trans. on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[10] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, 2004, pp. 477–492.

[11] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[12] X. Xu, P. Tabuada, A. D. Ames, and J. W. Grizzle, "Robustness of control barrier functions for safety critical control," in *IFAC Conf. on Analysis and Design of Hybrid Systems*, 2015, pp. 54–61.

[13] A. Mehra, W.-L. Ma, F. Berg, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *American Control Conference*. IEEE, 2015, pp. 1411–1418.

[14] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *IEEE Conference on Decision and Control*, 2016, pp. 2659–2664.

[15] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *American Control Conference*. IEEE, 2015, pp. 4542–4548.

[16] Q. Nguyen and K. Sreenath, "Optimal robust control for constrained nonlinear hybrid systems with application to bipedal locomotion," in *American Control Conference*. IEEE, 2016, pp. 4807–4813.

[17] X. Xu, "Control sharing barrier functions with application to constrained control," in *IEEE Conference on Decision and Control*, 2016, pp. 4880–4885.

[18] S. G. Lee and M. Egerstedt, "Controlled coverage using time-varying density functions," *IFAC Proceedings Volumes*, vol. 46, no. 27, pp. 220–226, 2013.

[19] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," *arXiv:1604.00640*, 2016.

[20] C. D. L. Cruz and R. Carelli, "Dynamic modeling and centralized formation control of mobile robots," in *IEEE Annual Conference on Industrial Electronics*, 2006, pp. 3880–3885.

[21] F. N. Martins, M. Sarcinelli-Filho, T. F. Bastos, and R. Carelli, "Dynamic modeling and adaptive dynamic compensation for unicycle-like mobile robots," in *IEEE International Conference on Advanced Robotics*, 2009, pp. 1–6.

[22] D. Pickem, M. Lee, and M. Egerstedt, "The GRITSBot in its natural habitat - a multi-robot testbed," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4062–4067.

[23] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, 2017 (to appear).

[24] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[25] "Implementing simultaneous lane keeping and adaptive speed regulation in the robotarium," https://youtu.be/VgoEcOFwJ2M.