The Robotarium: A remotely accessible swarm robotics research testbed

Daniel Pickem, Paul Glotfelter, Li Wang, Mark Mote, Aaron Ames, Eric Feron, and Magnus Egerstedt

***Abstract*— This paper describes the *Robotarium* – a remotely accessible, multi-robot research facility. The impetus behind the Robotarium is that multi-robot testbeds constitute an integral and essential part of the multi-robot research cycle, yet they are expensive, complex, and time-consuming to develop, operate, and maintain. These resource constraints, in turn, limit access for large groups of researchers and students, which is what the Robotarium is remedying by providing users with remote access to a state-of-the-art multi-robot test facility. This paper details the design and operation of the Robotarium and discusses the considerations one must take when making complex hardware remotely accessible. In particular, safety must be built into the system already at the design phase without overly constraining what coordinated control programs users can upload and execute, which calls for minimally invasive safety routines with provable performance guarantees.**

I. Introduction

Coordinated control of multi-robot systems has received significant attention during the last decades, with a number of distributed control and decision algorithms being developed to solve a wide variety of tasks, ranging from environmental monitoring to collective material handling. These developments have been driven by a confluence of algorithmic advances, increased hardware miniaturization, and cost reduction, and a number of compelling multi-robot testbeds have been developed (e.g., [1], [2]). However, despite these advances, it is still a complex and time-consuming proposition to go from theory and simulation, via a small team of robots, all the way to a robustly deployed, large-scale multi-robot system. Yet, actual deployment is crucial to advance multi-robot research since it is increasingly difficult to faithfully simulate all the issues associated with making multiple robots perform coordinated tasks. Even more difficult is the discovery of new issues based on analytical models of multi-robot systems alone.

The Robotarium, as shown in Figure 1, is an open, remote-access multi-robot testbed, explicitly designed to address this theory-simulation-practice gap by providing access to a testbed that is flexible enough to allow for a number of different scientific questions to be asked, and different coordination algorithms to be tested. What sets the Robotarium apart from other testbeds is its explicit focus on supporting *safe remote-access multi-robot research*, as opposed to testbeds that can be accessed remotely but do not explicitly consider formal safety measures or have an educational, as opposed to a research focus. Throughout this work, we will interpret safety as the avoidance of damaging collisions and quantify it through safety scores that determine whether user code is allowed to execute without further safety measures or not. The Robotarium makes hardware remotely accessible to both trusted and untrustworthy or malicious users while avoiding damage in a provable way. In this paper, we discuss how the Robotarium is structured and, in particular, how the explicit focus on being a *flexible* and *safe* remote-access research platform informs the design.[1]

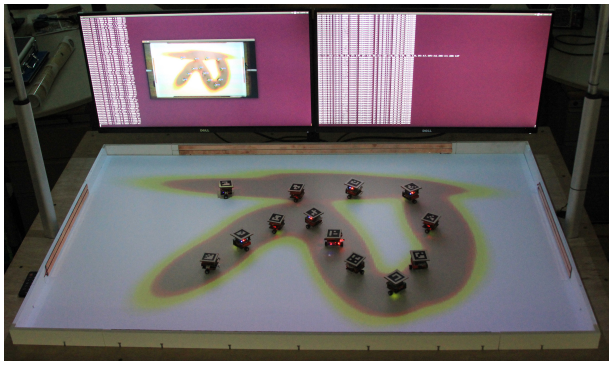


Fig. 1. Example of a coverage control algorithm executed on the Robotarium using 13 GRITSBot robots. The desired density function is projected onto the testbed arena in the shape of the letter R.

II. Related Work

In this section, we briefly survey the field of remote access testbeds that have been successful in their respective domains and broadly categorize them along the following dimensions: multi-robot testbeds, sensor network testbeds, and remotely accessible educational tools. A comprehensive overview can be found in [4]–[6] and the references therein.

A. Multi-Robot Testbeds

Numerous remotely accessible multi-robot testbeds with a focus on robot mobility have been proposed over the years - for example the Mobile Emulab [1], or the HoTDeC testbed [7]. A comprehensive list of multi-robot testbeds can be found in [5]. Generally speaking, testbeds in this domain contain robots that occupy a significantly larger footprint and are more expensive than the Robotarium robots, which is an inherent obstruction to using large numbers of robots.[2] The

*This research was sponsored by grants No. 1531195 and 1544332 from the U.S. National Science Foundation.

The authors are with the Georgia Institute of Technology, Atlanta, GA 30332, USA, {daniel.pickem,paul.glotfelter,liwang,mmote3,ames,feron, magnus}@gatech.edu.

[1]The report from a recent NSF Workshop on Remotely Accessible Testbeds [3] identified this inherent safety/flexibility tension as one of the key questions when pursuing a "science of remote access".

[2]A GRITSBot can be built for approximately $60 or bought pre-assembled for approximately $100 - see the bill of materials at www.robotarium.org

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

main difference between these testbeds and the Robotarium, however, is that the Robotarium explicitly addresses the robot safety aspect of remote accessibility such that provable damage avoidance of the Robotarium's physical assets is guaranteed even with untrustworthy or malicious users. The testbeds mentioned above in principle allow remote access for cases where a user can be trusted to not damage the hardware but do not explicitly address the safety issues involved once users have been approved for use. Unlike these testbeds, the Robotarium is inherently safe to operate because built-in (online and offline) safety measures prevent users from causing accidental or purposeful damage to the robots.

B. Sensor Networks Testbeds and Cybersecurity

Some of the earliest remote-access testbeds reside in the sensor networks and cybersecurity domains. Limiting access to largely immobile computing and sensing nodes mitigated the risk of making them publicly accessible. This category includes testbeds such as ORBIT [8], FIT IoT-Lab [9], DeterLab [10] and CONET [11], with the ORBIT testbed [8] and the DeterLab [10] standing out for their over a decade long remote operation. Of these testbeds, the FIT IoT-Lab [9] and CONET [11] contain mobile sensing nodes which both require significant resources and spaces to operate. The FIT IoT-Lab's over 200 mobile nodes are spread across six sites in France. Compared to the Robotarium's focus on mobility and coordinated control, FIT Iot-Lab's main focus, however, lies on communications and networking research in an IoT context. For a comprehensive overview of networking testbeds refer to [6] and the references therein.

C. Educational Testbeds

A number of testbeds have originated in the educational domain. For example, the Robotic Programming Network (RPN) [2] makes a single humanoid robot remotely accessible while Robotnacka [12] provides access to three mobile robots. A comprehensive overview of other educational testbeds can be found in [4]. While providing the required infrastructure for remote access, compared to the Robotarium, most educational testbeds contain small numbers of robots and are not explicitly designed to be research platforms for multi-robot or swarm robotics experiments.

III. The Robotarium

The Robotarium is a swarm-robotic research testbed that is accessible through a public web interface and gives users the flexibility to test a variety of multi-robot algorithms (see examples of remote experiments in Section V-B). 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. The continuous operation of the Robotarium highlights the need for automated maintenance, which relies on robust position tracking, automated battery recharging, and provably collision-free execution of motion paths. In this section, we will outline how these requirements guided the design of the first Robotarium instantiation, and elaborate on both the hardware and software architectures of the Robotarium.

A. Design Considerations

As a shared, remotely accessible, multi-robot facility, the Robotarium's main purpose is to lower the barrier of entrance into multi-agent robotics. Therefore, for the Robotarium to fulfill its intended use effectively, it has to implement a number of high-level design requirements aimed at accessibility, ease of maintenance, intuitive interaction, and safe and secure code execution.

- Enable inexpensive replication of the Robotarium through low-cost, open-source robots (currently up to 20 robots are available).
- Enable intuitive interaction with and simple data collection from the Robotarium through a public web interface that facilitates code submission and data/video retrieval.
- Enable a seamless switch between development in simulation and execution on the physical robots facilitated by a data-driven characterization of the simulation-hardware gap of the robots.
- Minimize the cost and complexity of maintaining a large collective of robots through convenience features such as automatic charging and tracking.
- Integrate safety and security measures to protect the Robotarium from damage and misuse through guaranteed collision avoidance.

B. Prototype Design

This section elaborates on the hardware and software components of the Robotarium as well as their interaction with each other, the robots, and the users (see Fig. 2). The Robotarium hardware includes the robots themselves, the position tracking system, the wireless communication hardware, as well as a charging system built into surface of the arena. The software back end consists of the simulation and virtualization infrastructure (also used for simulation-based code verification), interaction components (APIs), and the coordinating server application.

1) Hardware: The Robotarium is meant to provide a well integrated, immersive user experience with a small footprint (the current testbed measures $130\times90\times180$ cm), and features that allow a large swarm to be maintained effortlessly. At the core of the Robotarium are our custom-designed robots - the GRITSBots (introduced in [13]). These inexpensive, miniature differential drive robots simplify operation and maintenance of the Robotarium through features such as (i) automated registration with the server and overhead tracking system, (ii) automatic battery charging, and (iii) wireless (re)programming. While the initial implementation of these features has been described in detail in [13], major design revisions warrant a review and update.[3]

- *Robots:* The GRITSBot's modular design consists of a main board handling high-level intelligence, connectivity, power conditioning, and charging as well as a motor board responsible for the robot's motion. Compared

[3]A detailed description of the robot's hardware design including its design and code files are open-source and can be downloaded at https://github.com/robotarium/GRITSBot_hardware_design.

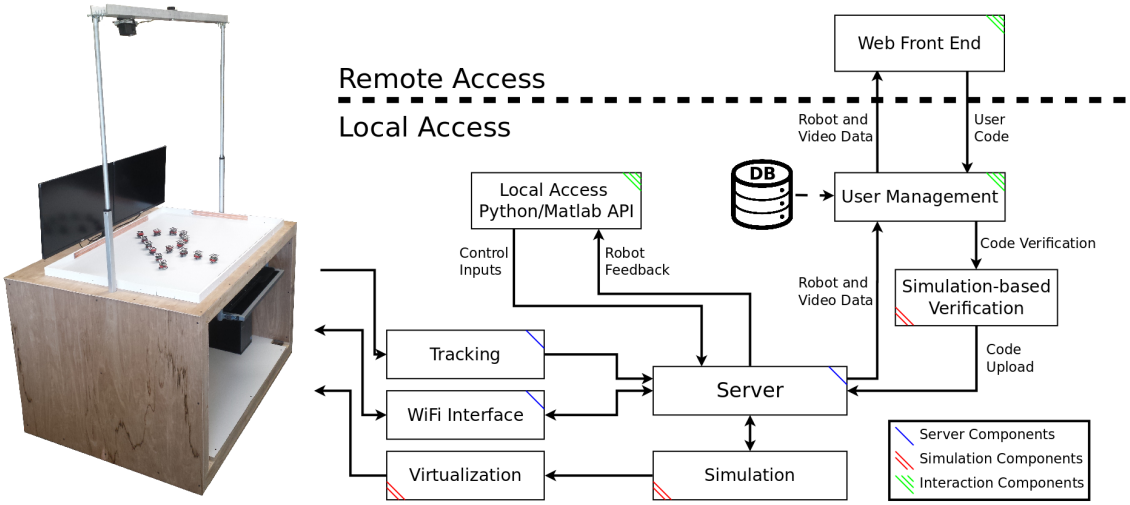


Fig. 2. System architecture overview. The Robotarium includes components that are executed locally on Robotarium infrastructure as well as user-facing components that run on remote user machines (web front end). Three components interact directly with the robot hardware – tracking, wireless communication, and virtualization. The remaining components handle user management, code verification and upload to the server, as well as coordination of user data and testbed-generated data.

to the initial specifications in [13], the GRITSBot has undergone multiple design iterations. Specifically, the robot's main board has been upgraded with the WiFi-enabled ESP8266 chip popular in the Internet of Things community. Operating at 160 MHz, this chip is capable of handling wireless communication, pose estimation, low-level control of the robot, as well as high-level behaviors. The ESP8266 chip's WiFi transceiver supports the IEEE 802.11 B/G/N standards with a bandwidth of up to 54 MBit/s. Note that the ESP8266 chip also enables wireless over-the-air code upload to the robots, which allows the Robotarium back end to upload new firmware to the robots within seconds. Equipped with a 400 mAh LiPo battery, the robot is capable of operating up to 40 minutes on a single charge while recharging takes approximately 45 minutes. The GRITSBot's motor board features an Atmega 168 microcontroller, which controls the stepper motors. Compared to the design shown in [13], the motor board has been equipped with additional introspective sensors (specifically motor current and temperature sensors) to enable predictive diagnostics of the robot's hardware state. Note that the robots are currently not equipped with sensor boards since distance sensing can be emulated through the back end server which tracks the positions of all robots.

- *Tracking:* The global position of all robots is retrieved through an overhead tracking system and is required to close the position control feedback loop. The Robotarium uses a single webcam in conjunction with ArUco tags for tracking.[4] Note that most decentralized algorithms do not rely on global position updates but rather sensor data. However, system maintenance such as recharging robots automatically or setting up an experiment relies on global position data.
- *Charging:* A crucial component of a self-sustaining testbed is an automatic recharging mechanism for its robots. The GRITSBot has been equipped with a wireless charging system for autonomous charging through a receiver coil attached to the robot (see Fig. 3b) and transmitters built into the Robotarium arena surface (both devices rely on the Qi wireless charging standard). Automatic recharging of robots is an essential aspect that will enable the long-term use of robots and the automated management of the Robotarium hardware with minimal operator intervention (see Section V-A) and at the same time make the continuous operation of the Robotarium economically feasible.

2) Software: The software components managing the operation of the Robotarium can be broadly grouped into three categories: simulation-based components, components enabling the interaction with the testbed, and coordinating server applications (see Fig. 2).

- *Simulation:* The simulation capabilities of the Robotarium are leveraged in three distinct ways: prototyping of user code, verification of user-provided code, and adding virtual robots. The simulators enable users to prototype and test their algorithms on their own machines before submitting them for execution on the Robotarium.[5] Once submitted, but before being executed on the testbed, the same simulation infrastructure verifies collision-free execution of user code (see Section IV-A). Additionally, the simulator also allows adding virtual robots to the testbed arena that are capable of interacting with the physical robots through the server back end.

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

[5]The simulators are currently implemented in Matlab and Python and available in the 'Downloads' section at http://robotarium.org/

- *Interaction:* These components govern how users can interact with the Robotarium. Two principal methods of interaction are enabled by the components shown in Fig. 2: local access through provided APIs as well as remote interaction through web-based code upload.[6] Local access requires users to connect to the Robotarium's WiFi network and is primarily used for development purposes. Remote access, on the other hand, requires users to implement and test algorithms in simulation before submitting them to the Robotarium via its web interface at www.robotarium.org. Submitted code undergoes simulation-based verification that checks for error- and collision-free execution before code is executed on physical robots (see Section IV-A).[7]
- *Coordination:* The server application is the central coordinating instance responsible for executing user code, routing commands and data to and from robots, transmitting global position data to the robots, and managing simulated virtual robots. In addition the server logs all generated data and makes them available to users. Note that the robots execute a velocity controller onboard. However, the user's control code is executed on the server and essentially remote controls the robots by providing velocity control inputs to the robots. This centralized execution increases overall robustness of the Robotarium, simplifies data logging as well as establishing formal safety guarantees, and facilitates automatic maintenance.[8] Note that centralization can lead to communication and computation bottlenecks. Given the total bandwidth of 802.11g WiFi of 54 MBit/s and typical bandwidth requirements of 3 KBytes/s/robot the theoretical upper limit is 18,000 robots. Clearly, WiFi collisions, buffering issues, interference, and other issues will practically lower that number. However, operating hundreds of robots simultaneously is well within the limits of the Robotarium's WiFi-based communication architecture. Computational scalability is addressed in Section IV-B.

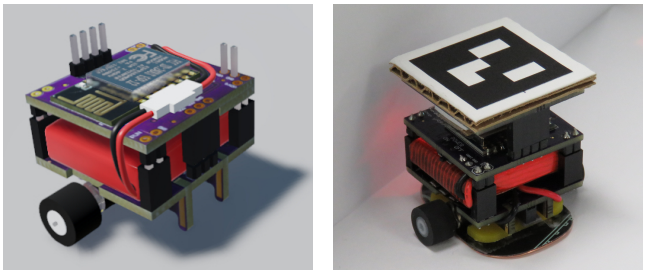
IV. Safety

Allowing remote users to control the Robotarium's physical equipment poses inherent risks to the integrity and safety of the hardware. To avoid damaging the hardware, a combination of offline simulation-based verification and online collision avoidance using barrier certificates is employed. By default, the execution of all user-supplied control code will be safe-guarded using barrier functions (see Section IV-B). However, users can bypass this online safety mechanism if they achieve a sufficiently high safety score during the offline simulation-based verification step (see Section IV-A).

[6]Note that no real-time teleoperation of robots is enabled for security reasons. Submitted user-code is executed locally on the Robotarium server and as such latency is negligible.

[7]Remote access to the Robotarium requires manual screening of applicants. Interested users can apply for access privileges via the website www.robotarium.org.

[8]Note, however, that other decentralized communication architectures such as peer-to-peer communication are also enabled by the WiFi-based communication architecture.



(a) 3D rendering of the GRITSBot (b) A GRITSBot wirelessly charging on a base station of the Robotarium.

Fig. 3. 3D model and current hardware implementation of the GRITSBot.

A. Simulation-based Verification

In this section we introduce an offline method to characterize the *safety* of a particular experiment and its suitability for deployment on the Robotarium. A *safety score* measures the frequency and severity of collisions in simulation to predict the collision behavior of an experiment once executed on hardware. This method relies on stochastic Monte Carlo simulations to compute the expected values of damage and safety over multiple simulation runs (currently 50). Uncertainties in the robots' dynamics model, their initial positions, as well as the observation model of the overhead camera are accounted for through added Gaussian noise. Note that strict collision-free execution is not enforced but minor collisions below certain damage thresholds are allowed. A sufficiently high safety score of an experiment allows execution on the Robotarium without additional online safety mechanisms (see Section IV-B).

To formalize safety scores, let the index set of N robots be $\mathcal{M} = \{1, 2, ..., N\}$. Then the function $D \in \mathbb{R}^+$ computes the cumulative *damage* done to N robots over a time horizon T and is defined as follows.

$$D = \sum_{i \in \mathcal{M}} D_i \triangleq \sum_{i \in \mathcal{M}} \int_0^T I_i(t)\delta_i(t)dt,$$

Here, $\delta_i(t)$ captures the rate at which robot i's kinetic energy is lost through a collision at time t and $I_i(t)$ is an indicator function that is 0 if robot i is not colliding with any other object at time t and 1 otherwise. In discrete time D_i can be approximated using the work-energy principle as follows.

$$D_i \approx \sum_{k=0}^{T/\Delta t-1} I_i(k\Delta t)\frac{m}{2}\left[v^2(k\Delta t) - v^2((k+1)\Delta t)\right], \quad (1)$$

where $m = 60g$ is the mass of the robot, $\vec{v}$ is its velocity vector such that $v^2 = \vec{v} \cdot \vec{v}$, and $\Delta t = 1/30s$ is the time step of the system governed by the tracking camera update rate. Note that Eqn. (1) assumes that a reduction in robot i's velocity is proportional to a loss in kinetic energy and

therefore approximates damage.[9] The cumulative *safety score* S is then defined as,

$$S \triangleq 1 - \frac{D}{D_{max}},$$

where D_{max} is the maximum allowable damage threshold for the entire experiment. Note that S is a unit-less value $S \in (-\infty, 1]$ which captures the energy lost from collisions over the time span of the experiment. In a similar fashion, we can limit the allowable damage done to any one robot i in the swarm by defining an individual safety score $s_i \triangleq 1 - \frac{D_i}{d_{i,\max}} \in (-\infty, 1]$ with respect to a separate threshold $d_{i,\max} \in \mathbb{R}^+$. Experiments are allowed to proceed for unmodified deployment in the Robotarium when both the cumulative and individual safety scores are positive.

B. Safety Barrier Certificates

The Robotarium uses *Safety Barrier Certificates* to guarantee provably collision-free behavior of all robots, which ensures the following three principles.

- All robots are provably safe in the sense that collisions are avoided.
- Users' commands are only modified when collisions are imminent.
- Collision avoidance is executed in real-time (in excess of 30 Hz update rate).

Safety barrier certificates are enforced through the use of control barrier functions, which are Lyapunov-like functions that can provably guarantee forward set invariance, i.e. if the system starts in the safe set, it stays in the safe set for all time. A specific class of maximally permissive control barrier functions was introduced in [14], whose construction provides the basis for the minimally invasive safety guarantees afforded by the Robotarium.

Consider a team of N mobile robots with the index set $\mathcal{M} = \{1, 2, ..., N\}$. Each robot i uses single integrator dynamics of the form $\dot{x}_i = u_i$, where $x_i \in \mathbb{R}^2$ is the planar position of robot i, and $u_i \in \mathbb{R}^2$ is its input velocity.[10] Additionally, robot i's velocity u_i is bounded by $\|u_i\| \leq \alpha, \forall i \in \mathcal{M}$. Let $x = [x_1^T, x_2^T, ..., x_N^T]^T$ and $u = [u_1^T, u_2^T, ..., u_N^T]^T$ denote the aggregate state and velocity input of the entire team of robots. To avoid inter-robot collisions, any two robots i and j need to maintain a minimum safety distance D_s between each other. This requirement is encoded into a pairwise safe set $\mathcal{C}_{ij},\ \forall\ i \neq j$, which is a super level set of a smooth function $h_{ij}(x)$,

$$\mathcal{C}_{ij} = \{x_i \in \mathbb{R}^2 \mid h_{ij}(x) = \|x_i - x_j\|^2 - D_s^2 \geq 0\}. \quad (2)$$

[9]The maximum energy loss possible from an inelastic collision involving a single GRITSBot weighing at most 60g directed towards a wall at its maximum speed of 0.1 m/s is 9.9 μJ.

[10]Single integrator dynamics can be easily mapped to the GRITSBot's unicycle dynamics using a nonlinear inversion method. It is also important to note that safety barrier certificates can be extended to more complex dynamical systems as well.

The function $h_{ij}(x)$ is called a control barrier function, if the admissible control space

$$K_{ij}(x) = \left\{ u \in \mathbb{R}^{2N} \;\middle|\; \frac{\partial h_{ij}(x)}{\partial x} u \geq -\gamma h_{ij}(x) \right\}, \quad (3)$$

is non-empty for all $x_i \in \mathcal{C}_{ij}$. It was shown in [15] that if the control input u stays in $K_{ij}(x)$ for all time, then the safe set $\mathcal{C}_{ij}$ is forward invariant. In addition, the forward invariance property of $\mathcal{C}_{ij}$ is robust with respect to different perturbations on the system.

Combining (2) and (3) as well as the single integrator dynamics, the velocity input u needs to satisfy

$$-2(x_i - x_j)u_i + 2(x_i - x_j)u_j \leq \gamma h_{ij}(x),\ \forall\ i \neq j.$$

This inequality can be treated as a linear constraint on u when the state x is given, i.e., $A_{ij}u \leq b_{ij},\ \forall\ i \neq j$, where

$$\begin{aligned} A_{ij} &= [0, \ldots, \underbrace{-2(x_i - x_j)^T}_{\text{robot } i}, \ldots, \underbrace{2(x_i - x_j)^T}_{\text{robot } j}, \ldots, 0] \\ b_{ij} &= \gamma h_{ij}(x). \end{aligned}$$

Similar constraints must be established for the workspace boundary. The corresponding safety set of robot i with regards to the boundary is denoted by $\bar{\mathcal{C}}_i$, and the corresponding constraints by $\bar{A}_i u_i \leq \bar{b}_i,\ \forall\ i \in \mathcal{M}$.

Combining these constraints – *all* pairwise collisions and collisions with the workspace boundaries – results in the safety set for the entire team as

$$\mathcal{C} = \prod_{i \in \mathcal{M}} \Big\{ \bigcap_{\substack{j \in \mathcal{M} \\ j \neq i}} \mathcal{C}_{ij} \bigcap \bar{\mathcal{C}}_i \Big\}.$$

The forward invariance of the safe set $\mathcal{C}$ is guaranteed by the safety barrier certificates, which are defined as

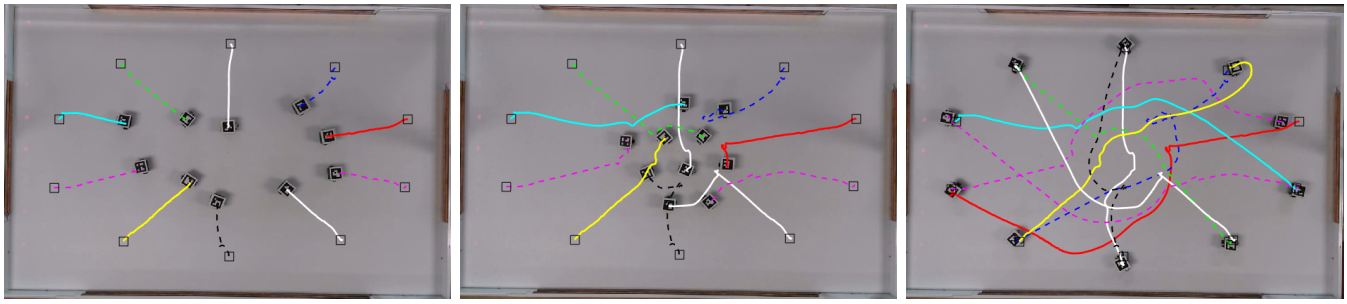
$$K(x) = \left\{ u \in \mathbb{R}^{2N} \;\middle|\; A_{ij}u \leq b_{ij}, \bar{A}_i u_i \leq \bar{b}_i,\ \forall\ i \neq j \right\}. \quad (4)$$

These safety barrier certificates define a convex polytope $K(x)$ in which safe control commands must stay. By constraining users' control commands to within $K(x)$, the Robotarium is guaranteed to operate in a provably collision-free manner.

The *minimally invasive* nature of barrier certificate-enabled collision avoidance stems from the fact that the deviation between the user-specified control signal and the actual, safe, executed signal is minimized, subject to the safety constraints through a Quadratic Program (QP)-based controller

$$\begin{aligned} u^* = \underset{u \in \mathbb{R}^{2n}}{\text{argmin}} \quad & J(u) = \sum_{i=1}^{N} \|u_i - \hat{u}_i\|^2 \\ \text{s.t.} \quad & A_{ij}u \leq b_{ij}, && \forall\ i \neq j, \\ & \bar{A}_i u_i \leq \bar{b}_i, && \forall\ i \in \mathcal{M}, \\ & \|u_i\|_\infty \leq \alpha, && \forall\ i \in \mathcal{M}, \end{aligned}$$

where $\hat{u}$ is the user's control command, u^* is the actual control command, and α is the bound for the control input. Note that in the absence of impending collisions (i.e. when the safety barrier certificates in (4) are satisfied), the user's



(a) Time 2.67s (b) Time 9.50s (c) Time 19.83s

Fig. 4. Ten GRITSBots swap positions with active safety barrier certificates. The robots' trajectories are shown together with square markers representing their initial positions.

code is executed faithfully. When violations occur, a closest possible (in a least-squares sense) safe control command is computed and executed instead. An experiment showing ten GRITSBots swapping positions with active safety barrier certificates is shown in Fig. 4, while the corresponding video is referenced in Table II.

Scalability of Safety Barrier Certificates: Safety barrier certificates are computed in a centralized fashion on the Robotarium's back end server and therefore scalability is a concern. As the size of the swarm increases, the number of decision variables (u) in the QP-based controller increases linearly, while the number of pairwise safety constraints grows quadratically. However, a more computationally efficient implementation similar to [16] is possible, where agent i only considers its neighbors for collision avoidance and the certificates computation can be distributed to individual agents. More specifically, the robot's finite physical dimensions limit the maximum robot density. For example, for a minimum safety distance of $D_s = 8cm$ and a neighborhood radius of $20cm$, any given neighborhood can contain at most 26 other robots, which limits the size of each individual robots QP problem to 2 decision variables and at most 26 linear constraints. The computation time of safety barrier certificates for centralized as well as decentralized computation is shown in Table I. Note that the decentralized barrier certificates were computed on a single central computer.[11] Thus, the total computation time T_d is divided by N to characterize the decentralized and fully parallel implementation. As Table I shows, while centralized computation suffers from scaling up the number of robots, the computation time of decentralized safety barrier certificates remains below 10ms even for 100 robots. In fact, decentralized safety barrier certificates can handle 100 GRITSBots with an update frequency of 185Hz and therefore scale to large numbers of robots without compromising update rates.

V. Usage

This section highlights the main usage features of the Robotarium: continuous operation and auto-recharging of robots, safe remote access for external users, and the characterization and closing of the simulation-hardware gap that can prevent the successful execution of controls code on physical robots. The following sections detail each of these features.

A. Long-term Operation

The robust long-term operation of the Robotarium is predicated on a reliable and autonomous charging mechanism for its robots. In this section we therefore provide experimental evidence of the reliability of the GRITSBot's wireless charging mechanism (see Section III-B.1) and the Robotarium's capabilities for continuous operation.

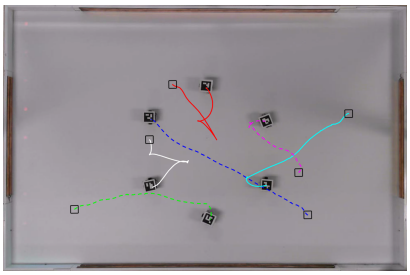
This section summarizes the results of three experiments, each of which made use of three robots and continuously executed an example algorithm over the course of 140, 148, and 240 minutes, respectively. During these times, each robot executed 37, 39, and 60 autonomous recharge cycles, i.e. successfully charged its battery through the wireless chargers.[12] Out of a total of 111, 117, and 180 autonomous charge cycles, manual intervention was only required twice in the first experiment and once in the second, because a robots onboard control software froze. The success rate of the autonomous charging system was therefore 98.1%, 99.1%, and 100%, respectively. A time-lapse video showing the continuous operation of three robots is referenced in Table II.

B. External Users

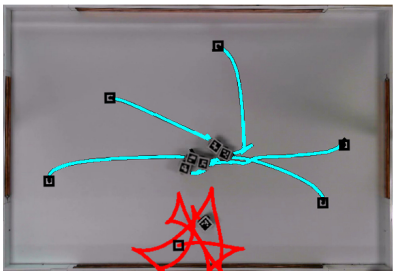
This section presents a selection of external user experiments that have been developed using a Robotarium-provided simulator and executed on the Robotarium using the software infrastructure shown in Section III-B.2 and Fig. 2. These examples were chosen as representative samples since they highlight the breadth of algorithms that can be executed on the Robotarium but also validate its remote access aspect. Note that the corresponding videos are referenced in Table II.

[11] Barrier certificates were computed on an Intel I7 4790 3.6 GHz with 16 GB of memory.

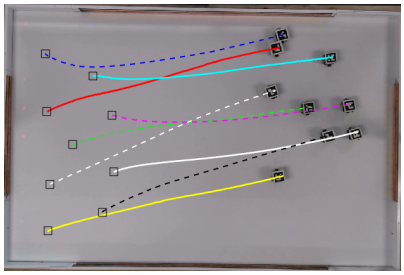
[12] Note that we shortened each charge cycle in the interest of time and therefore on average a charge cycle only lasted approximately 6 minutes - 3 minutes of experiment and 3 minutes of charging. As mentioned in Section III-B.1 the GRITSBot's runtime on a single charge is up to 40 minutes.



(a) Distributed formation control of six GRITSBots assembling a regular hexagon starting at random initial positions.



(b) Fault-tolerant consensus with five collaborating robots (shown in blue) and one malicious robot (shown in red). The collaborative agents achieve rendezvous despite the malicious robot's efforts to prevent it.



(c) Passivity-based attitude synchronization implemented on a team of nine GRITSBots.

Fig. 5. Experimental data from external user experiments rendered onto images of the Robotarium testbed setup. The square markers and curves are initial positions and trajectories of the GRITSBots, respectively.

Swarm Size N	**Centralized Certificates** T_c **(ms)**	**Decentralized Certificates** T_d/N **(ms)**
10	5.6	3.2
40	11.6	3.5
100	78.0	5.4

TABLE I

COMPUTATION TIME OF BARRIER CERTIFICATES PER ITERATION.

1) Distributed Formation Control of Cyclic Formations from the University of Texas, Dallas: This experiment instantiated a distributed formation control algorithm for regular polygonal formations, originally presented in [17], [18]. The controller uses relative position measurements in local coordinate frames and prohibits any inter-agent communication. Note that [18] assumes agents to be points in the plane and does not consider collision avoidance. The successful execution on the Robotarium therefore depended on the use of Robotarium-provided barrier certificates shown in Section IV-B and a mapping from single-integrator dynamics to the unicycle dynamics of the robots. Figure 5a shows the results of this experiment with six robots.

2) Fault-tolerant Rendezvous from the University of Illinois Urbana-Champaign: A second experiment instantiated a fault-tolerant version of the rendezvous algorithm, originally presented in [19]. In this work, agents achieve consensus by moving towards points within a safe set, while maintaining connectivity through extendable sensing capabilities [19]. Because this algorithm models agents as points in the plane and contains no native collision avoidance, the successful execution utilize the single-integrator-to-unicycle mapping and barrier certificates provided by the Robotarium. Figure 5b shows the results of this experiment with six robots.

3) Passivity-based Attitude Synchronization from the Tokyo Institute of Technology: A passivity-based attitude synchronization algorithm, originally presented in [20], was implemented on the Robotarium. Utilizing the passivity property of general rigid-body motion in $SE(3)$, this algorithm was designed to achieve attitude synchronization for a group of rigid bodies with only local information exchanges, i.e., $v_i = v_j, \lim_{t\to\infty} \theta_i(t) - \theta_j(t) = 0, \forall i \neq j$. The successfully execution of this algorithm relied on the following capabilities provided by the Robotarium: 1) specification of a local information exchange graph, i.e., a cycle graph (C_N); 2) a mapping from single-integrator to unicycle dynamics. Figure 5c shows the results of this experiment with eight robots.

C. The Simulation-Hardware Gap

The Robotarium's infrastructure (see Section III-B) is set up such that users prototype their code using the provided simulators and submit the exact same code for execution on the Robotarium's hardware. As such, it is important that the simulators provide a reasonably accurate approximation of the robots behavior. To characterize this simulation-hardware gap, we use linear regression on recorded data to provide a measure of system identification. This characterization relies on the GRITSBots dynamical model, i.e. the unicycle model given by

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{5}$$

where θ is the orientation of the robot, v and ω are its linear and angular velocity, respectively. We assume that each linear regression coefficient appears in the observation model as $\hat{\dot{x}}_i = \alpha_i \dot{x}_i$, where $\hat{\dot{x}}_i$ is an observation of $\dot{x}_i$ that is obtained through the Euler method applied to observed position and orientation values. Let $\hat{\dot{X}}_i \in \mathbb{R}^d,\ i \in \{1, 2, 3\}$ be the collection of observations $\hat{\dot{x}}_i$ and $\dot{X}_i \in \mathbb{R}^d$ be the collection of evaluations of the unicycle model $\dot{x}_i$, where d is the number of data points (here, we use $d = 30000$ data points). Then each coefficient is determined by the least squares linear regression equation

$$\alpha_i = (\dot{X}_i^T \dot{X}_i)^{-1} \dot{X}_i^T \hat{\dot{X}}_i,$$

which yields the following modified unicycle model given by

$$\dot{\tilde{x}} = \text{diag}(\alpha_1, \alpha_2, \alpha_3)\dot{x},$$

Preprint submitted to 2017 IEEE International Conference on Robotics and Automation. Received September 15, 2016.

Algorithm	Link to Video
Safety barriers	https://youtu.be/PWJk-oMcgn4
Long-term Operation	https://youtu.be/PBdrD7ZS-qM
Track-following	https://youtu.be/VIirTkWppkE
Dist. Formation Control	https://youtu.be/DXcF0h8Vld0
Fault-tolerant Consensus	https://youtu.be/AlUyUVoVMu0
Attitude Synchronization	https://youtu.be/cItg_vGv3jo

TABLE II

LIST OF VIDEO REFERENCES.

where $\alpha_1 = 0.8645, \alpha_2 = 0.8119, \alpha_3 = 0.4640$. The used data displayed a linear relationship between $\hat{\dot{x}}$ and $\dot{x}$, ensuring the accuracy of the linear regression. The Robotarium simulators use these values for α_i together with Eqn. (5) to simulate the robot's dynamics. To evaluate the accuracy of the linear regression results, we furthermore implemented a waypoint-following algorithm (see [21]). A video of this experiment is referenced in Table II while the following provides a numeric estimate of the error between simulation and hardware execution. In particular, let $x_{sim}(t)$ be the trajectory in simulation and $x(t)$ the trajectory of the GRITSBot. Then we calculate the average error between the simulated and experimental trajectories as

$$E(x_{sim}(t), x(t)) = \frac{1}{T}\int_0^T \min_{\tau\in[0,T]} \left(\|x_{sim}(\tau) - x(t)\|\right) dt,$$

which for this experiment yielded an error value of

$$E(x_{sim}(t), x(t)) = 0.0052\ m,$$

or an average difference of 5 mm between simulation and hardware execution.

VI. CONCLUSION

In this paper, we have detailed the development of a remotely accessible, multi-robot research facility – the *Robotarium*. Beyond introducing the hardware and software components required to enable remote access of robot swarms, the Robotarium addressed the two key concerns of flexibility and safety. Unlike other remotely accessible testbeds, the Robotarium makes use of formal methods to ensure the safety of its physical assets and the avoidance of damage to the robots. These methods guarantee collision avoidance in a minimally invasive manner without overly constraining the type of control algorithms that can be executed on the Robotarium. To demonstrate the flexibility and versatility of this testbed as well as its use by the community, we have shown a number of external user examples that were deployed on the Robotarium with little implementation overhead and provable collision avoidance.

ACKNOWLEDGEMENTS

We would like to thank Kaveh Fathian, Nicholas Gans, and Mark Spong from the University of Texas in Dallas, Hyongju Park and Seth Hutchinson from the University of Illinois in Urbana-Champaign and Junya Yamauchi and Masayuki Fujita from the Tokyo Institute of Technology for their participation in the early release of the Robotarium.

REFERENCES

[1] D. Johnson, T. Stack, R. Fish, D. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile emulab: A robotic wireless and sensor network testbed," in *Computer Communications (INFOCOM), 2006 IEEE International Conference on*, 2006, pp. 1–12.

[2] G. Casan, E. Cervera, A. Moughlbay, J. Alemany, and P. Martinet, "Ros-based online robot programming for remote education and training," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 6101–6106.

[3] M. Egerstedt and M. Govindarasu. (2015, November) Accessible remote testbeds: Opportunities, challenges, and lessons learned. Last retrieval at March 31, 2016. [Online]. Available: http://gritslab.gatech.edu/home/?p=8515

[4] S. Groeber, M. Vetter, B. Eckert, and H.-J. Jodl, "Experimenting from a distance remotely controlled laboratory (rcl)," *European Journal of Physics*, vol. 28, no. 3, pp. 127–141, 2007.

[5] A. Jimnez-Gonzlez, J. R. M. de Dios, and A. Ollero, "Testbeds for ubiquitous robotics: A survey." *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1487–1501, 2013.

[6] A.-S. Tonneau, N. Mitton, and J. Vandaele, "How to choose an experimentation platform for wireless sensor networks? a survey on static and mobile wireless sensor network experimentation facilities," *Ad Hoc Networks*, vol. 30, pp. 115 – 127, 2015.

[7] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, and G. Dullerud, "Multivehicle systems control over networks," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 56–69, 2006.

[8] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking, 2005 IEEE Conference on*, vol. 3, March 2005, pp. 1664–1669.

[9] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne, "Fit iot-lab: A large scale open experimental iot testbed," in *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, Dec 2015, pp. 459–464.

[10] J. Mirkovic and T. Benzel, "Teaching cybersecurity with deterlab," *Security Privacy, IEEE*, vol. 10, no. 1, pp. 73–76, January 2012.

[11] J. R. M. de Dios, A. Jimnez-Gonzlez, A. de San Bernabe, and A. Ollero, *A Remote Integrated Testbed for Cooperating Objects*. Springer Science & Business Media, 2013.

[12] P. Petrovic and R. Balogh, "Deployment of remotely-accessible robotics laboratory." *International Journal of Online Engineering (iJOE)*, vol. 8, no. S2, pp. 31–35, 2012.

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

[14] A. Ames, J. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Dec 2014, pp. 6271–6278.

[15] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[16] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[17] K. Fathian, D. I. Rachinskii, M. W. Spong, and N. R. Gans, "Globally asymptotically stable distributed control for distance and bearing based multi-agent formations," in *2016 American Control Conference (ACC)*, July 2016, pp. 4642–4648.

[18] K. Fathian, D. I. Rachinskii, T. H. Summers, and N. R. Gans, "Distributed control of cyclic formations with local relative position measurements," in *Decision and Control (CDC), 2016 IEEE Conference on (to appear)*, 2016.

[19] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, May 2016, pp. 358–365.

[20] Y. Igarashi, T. Hatanaka, M. Fujita, and M. W. Spong, "Passivity-based attitude synchronization in se(3)," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1119–1134, Sept 2009.

[21] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle like vehicles via lyapunov techniques," *Robotics Automation Magazine, IEEE*, vol. 2, no. 1, pp. 27–35, 1995.