The Robotarium

GLOBALLY IMPACTFUL OPPORTUNITIES, CHALLENGES, AND LESSONS LEARNED IN REMOTE-ACCESS, DISTRIBUTED CONTROL OF MULTIROBOT SYSTEMS

SEAN WILSON, PAUL GLOTFELTER, LI WANG, SIDDHARTH MAYYA, GENNARO NOTOMISTA, MARK MOTE, and MAGNUS EGERSTEDT

Digital Object Identifier 10.1109/MCS.2019.2949973
Date of current version: 16 January 2020

Distributed control has emerged as a major focus in the systems and controls area, with multiagent robotics playing a prominent role both as a canonical instantiation of a system, where control decisions must be made by individual nodes across an information-exchange network, and as a rich source of applications [1]–[5]. These applications include environmental monitoring [6], collective materials handling [7], construction [8], disaster response [9], and precision agriculture [10].

The technical developments in multiagent robotics have, in no small part, been driven by control innovations, such as coordinated control strategies for forming shapes or covering areas [11], [12]. Additionally, improvements in computing, sensing, communication, actuation, and power modalities have led to the creation of a number of relatively low-cost, small robots capable of acting in collectives [13]–[15]. However, it is still cost prohibitive for many researchers and educators to construct large-scale testbeds for distributed control that reliably and repeatedly operate tens to hundreds of robots simultaneously.

Beyond the initial cost, maintaining a multirobot testbed is complex and time-consuming; as a consequence, most research on distributed control of multirobot systems is validated in simulations instead of with hardware. Although high-fidelity robotics simulators are available and useful [16]–[18], their complicated software environments can make them difficult to set up and expand upon to faithfully simulate new robotic scenarios [19]. When considering the multiagent robotic domain specifically, the setup complexity and computational power required for high-fidelity simulations increases as the number of simulated agents and interactions increases. These issues can push researchers and educators to use simplified simulations, which typically neglect real-world implementation issues, such as wheel slip, friction, networking, computation time, and actuator constraints.

A solution to the difficulties associated with validating distributed control algorithms in hardware or high-fidelity simulators is the Robotarium, a remotely accessible, publicly available multirobot testbed located at the Georgia Institute of Technology. By allowing users to implement their algorithms remotely on physical robots, they can bypass drawbacks of building their own testbed or testing in a simulated environment (see "Summary"). The Robotarium also serves as a first step toward a standardized multirobot testbed where algorithms can be tested, compared, and validated by individuals worldwide on the same computational hardware, communication structure, and robots. This platform exposes users around the world to the theory-to-practice gap, accelerates the implementation of theory on distributed control of multirobot systems, and, ultimately, democratizes access to a state-of-the-art research facility.

The Robotarium was the first remote-access testbed for multirobot experiments available to the public [20]. Other remote-access robotics platforms have since been developed for more specific applications. A multirobot project similar to the Robotarium, named *Duckietown*, was released to the public recently [21]. Duckietown, originally developed as an inexpensive vision-based navigation testbed to be placed in classrooms and laboratories around the world, has incorporated a remotely accessible component centered around the AI Driving Olympics challenge [22]. Unlike the Robotarium, which is designed to be remotely accessible by a general audience with support for a wide range of high-level applications, Duckietown is aimed at the specific task of vision-based autonomous navigation, with hardware and software design focused on autonomous vehicle challenges. The Synchronized Position Hold Engage Reorient Experimental Satellites (SPHERES) remote-access testbed [23], [24] aboard the *International Space Station* serves as another example of a successful publicly available robotics testbed. Unlike the general multirobot nature of the Robotarium, the SPHERES project is specifically focused on satellite formation flight and docking algorithms in a microgravity environment.

The technical specifications of the original Robotarium have been described previously in [20]. Since then, the Robotarium has been through a significant upgrade (Figure 1) and has been open for general worldwide use since 22 August 2017. The explicit mission of the Robotarium was originally to provide controls and robotics investigators from all over the world with free access to a state-of-the-art, multirobot testbed that can host and support a vast range of research pursuits. However, this release brought with it many unknowns centered around users, use cases, and demand. This article addresses the user-centric evolution of the Robotarium testbed and reports the user impact and outcomes from creating a multirobot laboratory, focusing explicitly on distributed control that is remotely accessible to the world at large.

The following sections will give a general introduction to the design of the testbed, as well as the submission process, discuss the demographics of the user base, analyze the types of experiment submissions to the Robotarium, introduce some of the challenges associated with allowing unknown and possible inexperienced users to access a

Summary

The experimental validation of robot control algorithms is expensive in terms of monetary funding as well as the time and expertise required for development and maintenance. To remove these barriers, the Robotarium, a remote-access, multirobot research facility, was developed to enable investigators, educators, and students from around the world to deploy and validate their algorithms on robotic hardware free of charge. The Robotarium has been available to the public for approximately two years, and in that time, it has gained hundreds of users from every continent except Antarctica and executed more than 1000 experiments. The use cases of the Robotarium have been diverse, ranging from multiagent formation control to traffic modeling to biomimicry algorithms. This article provides a general introduction to the design of the testbed, explains the submission process, discusses the demographics of the user base, analyzes the types of experiments submitted, introduces some of the challenges associated with allowing unknown (and possibly inexperienced) individuals to access a state-of-the-art research testbed, and addresses how user experiences and feedback help guide enhancements.

FIGURE 1 The Robotarium testbed with GRITSBot X robots charging around the perimeter.

state-of-the-art research testbed, and address how user experiences and feedback help guide its enhancements.

THE ROBOTARIUM

The Robotarium, shown in Figure 1, is a remotely accessible swarm-robotics testbed, designed to help users quickly prototype and validate their distributed control strategies through implementation on physical robots without the overhead of setting up their own hardware or high-fidelity simulation. The testbed is also an opportunity for operators to compare the performances of similar algorithms by deploying them into a standardized hardware environment. If a user chooses to distribute his or her code, its performance can be verified by anyone around the world and leveraged in future endeavors or projects. An explicit ambition is to democratize access so that world-class research infrastructure is available beyond well-endowed institutions; the platform is free to be used by anyone from anywhere in the world for academic investigation or educational purposes.

By design, the Robotarium enables individuals to deploy many different types of control algorithms and execute highly varied application scenarios, such as path-planning algorithms, task allocation problems, and behavior compositions. It is constructed to handle some the challenges associated with an always-on, remotely accessible hardware platform, which include a user-friendly simulation interface, reliable and robust hardware, efficient power management with automatic recharging capabilities, and provably safe long-term operation given unknown user control commands. This section will briefly describe the Robotarium testbed, simulation application programming interface (API), and the submission process.

Testbed Design

The Robotarium is located in a converted classroom on the campus of the Georgia Institute of Technology. The testbed where experiments are deployed is a 12×14-ft^2 custom elevated platform. The walls of the testbed are outfitted with Qi inductive chargers that allow the robots to autonomously charge themselves between submissions and during experiments for which they are not required, as shown in Figure 2. This setup keeps the robots available for use over long time scales without human intervention. Wireless inductive chargers were chosen over conductive rail charging for general safety. Conductive charging always has a chance of being shorted, which can be catastrophic on an autonomous testbed that is left unattended for long periods of time. More importantly, the Robotarium is designed to be a highly visible and toured space, and conductive charging has the potential to harm visitors who may accidentally bridge the charging leads.

Eight Vicon motion-capture cameras [25] are mounted above the perimeter of the testbed to track each robot's motion for data acquisition and control purposes. Each robot is tracked through a unique, nonsymmetrical pattern of Vicon Pearl markers, seen on the tops of the robots in Figure 2. The Vicon motion-capture system was chosen over the original single visual camera-tracking system due

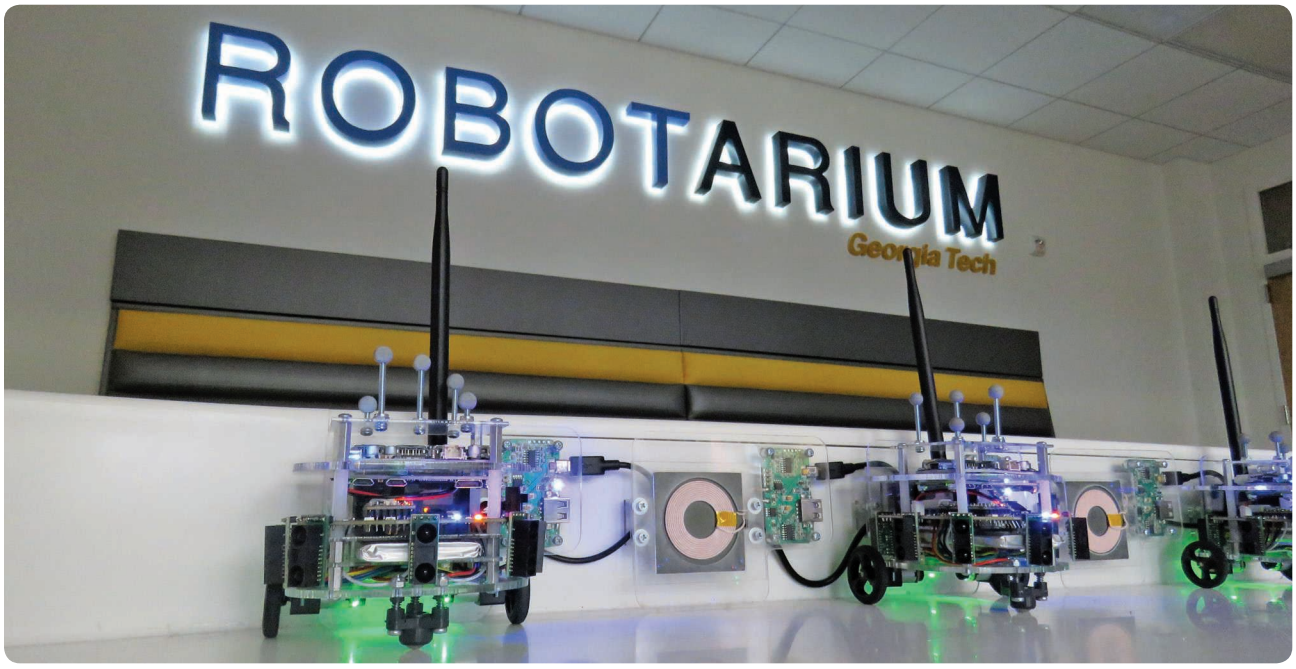

FIGURE 2 A group of GRITSBot X robots charging on the inductive charging stations surrounding the Robotarium.

to its ability to easily observe a larger volume and return submillimeter-precision pose information at a maximum rate of 120 Hz for each robot in the arena. From a hardware safety standpoint, the speed, accuracy, and precision of the Vicon system allows the Robotarium to detect potentially harmful situations during the execution of an unknown experiment, which can then be autonomously corrected and averted.

Additionally, a 2-megapixel ELP camera [26], used for automatic video capture of experiments, is placed over the center of the testbed. This allows individuals from around the world to review their experiments quickly to identify potential flaws or undesirable behaviors and use the video to supplement presentations of their work. To convey additional information during the execution of the experiment, an Optoma EH200ST projector [27] allows users to project time-varying environmental backgrounds onto the testbed during their experiments. These backgrounds can be general landscapes (for example, an urban city with roads, a forest with spreading fire, or the inside of a building), visualizations of environmental functions (such as potential fields or flow fields), or projections of potentially useful information (including battery voltage, robot number, and desired control vectors). Figure 3 shows an example usage of the projector where desired robot goal locations are projected onto the testbed along with associated color rings and robot state information.

During its first year and a half of operation, the Robotarium was populated with custom differential-drive mobile robots, named *GRITSBots* [13]. After this period, the GRITSBot was updated to a larger custom-made differential-drive platform, the GRITSBot X (seen in Figure 2) for more reliable long-term operation. The differential-drive type of robot was chosen due to its well-studied nature, wide usage, general applicability to many multirobot scenarios, and design simplicity. However, the general software framework of the Robotarium is designed in a modular way, allowing the future integration of any robot with Wi-Fi communication and wireless charging capabilities, depending on demand or long-term operational needs.

Beyond the functional design of the space for the remote user, a major effort was put forward to enable the Robotarium to provide a theatrical and curated experience for students, educators, researchers, and the general public. The room has various cosmetic design elements, including a large illuminated sign, viewing couches, and large observation windows that allow passersby the opportunity to watch and be inspired by the testbed in action. Making the space highly visible, instead of having it locked away for use by investigators exclusively, elevates the Robotarium's impact beyond the academic research community. For example, the large couches facing the testbed provide visiting tours the opportunity to experience the Robotarium as a welcoming and entertaining learning experience, more like a planetarium than a cold, clinical robotics laboratory. These choices expand the user base and general awareness of the Robotarium in a grassroots way through word of mouth into communities that could find the testbed useful but are not the anticipated individuals who would be targeted.

The Robotarium Simulation Application Programming Interface

The Robotarium simulation API is available in Matlab and Python [28], [29]. The purpose of the simulator is to enable

users to rapidly prototype their distributed control algorithms and receive feedback about their implementation feasibility before sending them to be executed by the robots on the Robotarium. Originally, Matlab was chosen as the default development language due to its wide use in academia as a high-level prototyping tool for control designers and roboticists as well as by mathematicians, biologists, social scientists, and those in other disciplines focused on distributed decision making. However, to provide a free alternative to Matlab, as requested by individuals outside of academia, a simulation environment is also available in Python.

When users run their simulation locally, the API produces an animated figure window with a cartoon representation of the robots moving within a bounding box that represents the edges of the workspace of the Robotarium testbed. Operators are not allowed to alter the bounding box or the cartoon representation of the robots, but they may customize the environment inside the bounding box as much as they desire. Anything a user can plot in a Matlab or Python figure window will be displayed. This enables operators to visualize static or dynamic environments that give their experiments context and provide additional information about the robots through time-varying labels or markers. For example, Figure 4(a) and (b) shows snapshots from the figure produced by the Robotarium simulation API when running a script that controls 20 robots (represented by the orange squares) to form the Georgia Institute of Technology logo, which is plotted behind them at the end of the experiment for reference. If users do not wish to see the simulation visualization, it can be suppressed for faster runtime.

Simulation results produced by the Robotarium simulator are not based on high-fidelity interaction or motion physics, due to the existence of other well-maintained simulation software and the ease of transferring a Robotarium simulation to hardware for validation. The simulator models the motion of the differential-drive robots through forward integration of a unicycle model

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

where $\mathbf{x} = [x\ y\ \theta]^\top$ is the position and orientation of the robot and $v, \omega \in \mathbb{R}$ represent the commanded linear and angular velocities of the robot, respectively. To include the geometry of the robot, the unicycle dynamics in (1) may be formed as the differential-drive dynamics

$$\dot{\mathbf{x}} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r/2 & r/2 \\ r/l & -r/l \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}. \tag{2}$$

In (2), the angular velocities for the right and left wheel, $\omega_r, \omega_l \in \mathbb{R}$ (transformed through kinematic relations including the robot axle length, l, and wheel radius, r), replace v and ω.

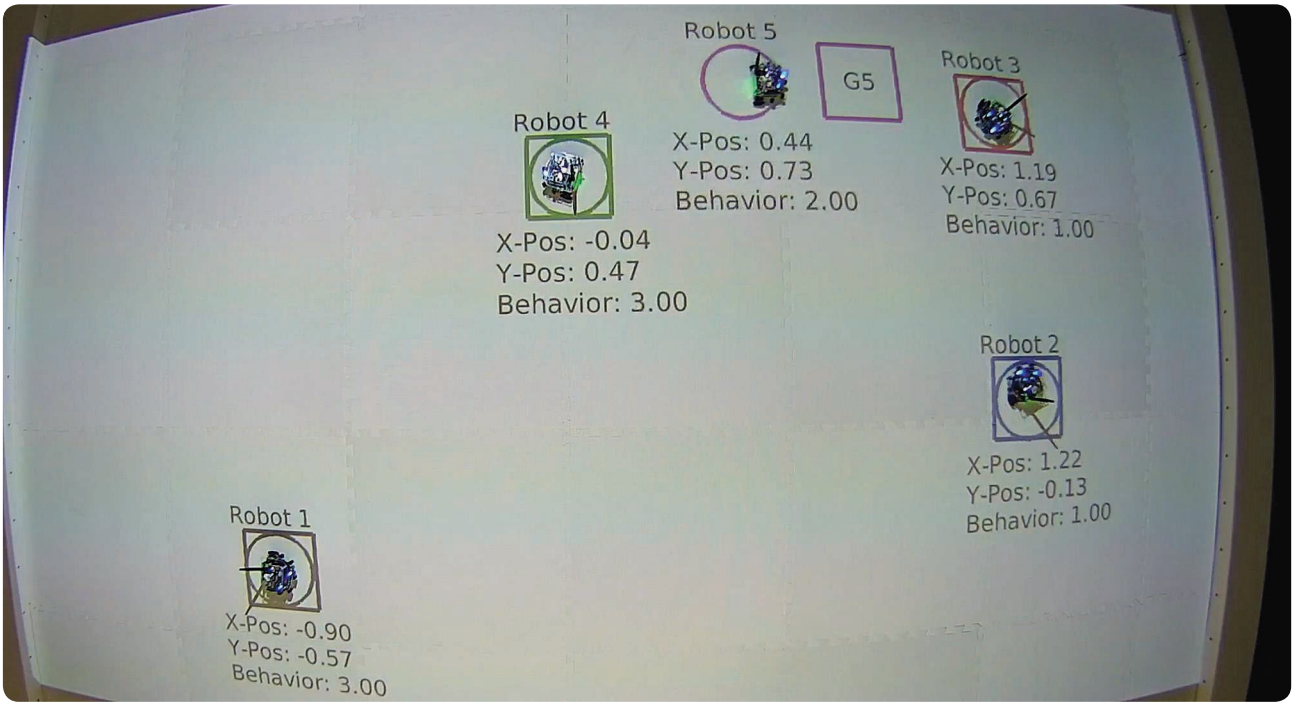

FIGURE 3 An overhead image of an experiment on the Robotarium, which projects the simulated behavior state, number identification, and current position displayed in text around each robot. The colored squares with correlated color circles centered on each robot's current position indicate the desired goal locations for each robot.

Operators may issue commands to control individual robot motion through v and ω in (1). In the simulation, motor dynamics are not considered. However, the transformation in (2) is utilized to check whether user-issued velocity commands will cause actuator saturation or enter the nonlinear low-speed domain of the individual motors on the physical robots on the Robotarium. If infeasible commands are detected, individuals are warned in the API that their simulated results will most likely be different from the experimental ones if the script is submitted in its current state.

To move the simulated robots and, eventually, the physical robots on the Robotarium, users must produce a matrix of unicycle velocity commands for each robot at each time step. How these commands are generated is entirely up to the operator. If they need help, individuals have access to global state information for all of the robots, which can be manipulated as needed. The onboard proximity sensors, encoders, and inertial measurement unit of the GRITSBot X are currently unavailable to users, as sufficient models of their performance must be created for faithful simulation. The original platform (the GRITSBot) did not have external sensors for the operator to access. To sidestep this temporary shortcoming, the tracking information available on the Robotarium has been leveraged by individuals to produce control commands (through individual, local, or global robot state feedback) and manipulated to cause robots to react based on simulated sensor feedback.

Finally, when a simulation concludes, users may save any variables from their workspace that they are interested in as MAT or NumPy files. These data can be from purely artificial sources (such as robot behavioral states) or related to the Robotarium hardware (including global state information from tracking data or individual robot battery voltage). After a simulation concludes, the API can display a summary of potential issues with the simulation that would cause rejection of the submission to the Robotarium, for example, robots traveling outside of the workspace or being too close to one another (a potential robot collision scenario).

Experiment Submission Process

Individuals access the Robotarium through a public web interface at www.robotarium.org. The site contains information about the remote-access testbed, contact information for specific inquiries, frequently asked questions from the user base,

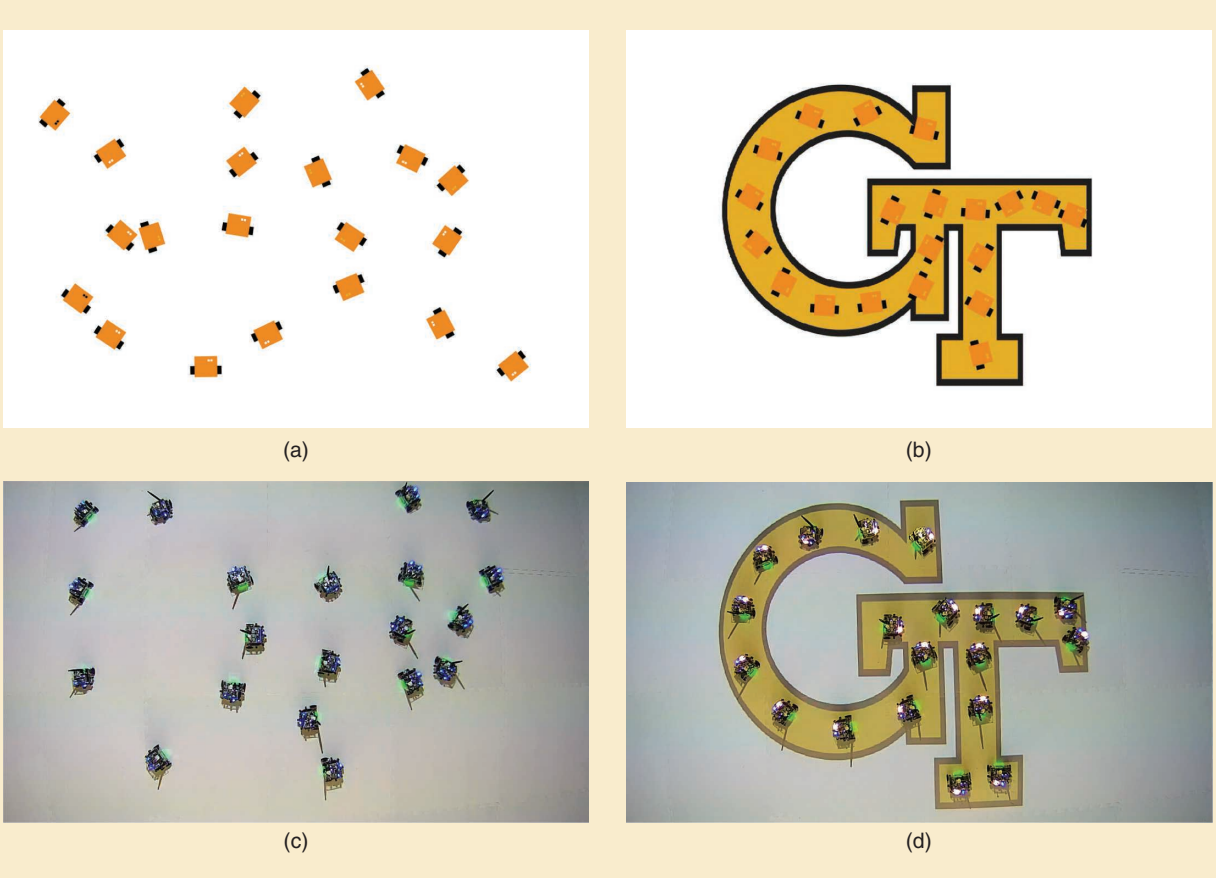

FIGURE 4 Zoomed-in screenshots of the (a), (b) simulation and (b), (c) returned video from the Robotarium for an algorithm that drives 20 GRITSBot X robots to form the Georgia Institute of Technology logo. The (a) initial and (b) final formation of the robots in the Robotarium simulation figure. The (c) initial and (d) final formation of the robots in the deployed Robotarium experiment.

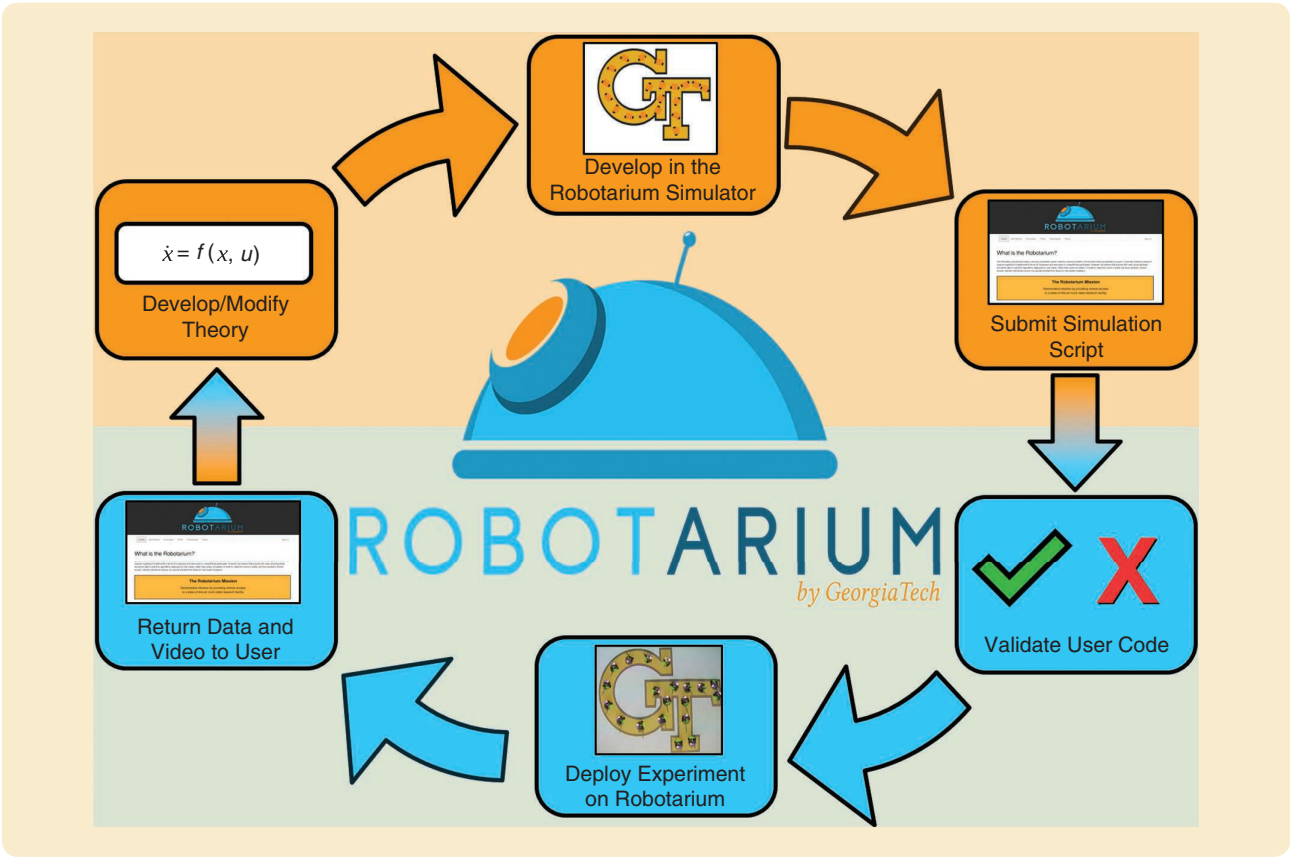

FIGURE 5 The Robotarium submission process. The user development and submission steps are highlighted in orange, while the internal Robotarium processes are shown in blue.

```
%% General Code Formatting
% Sean Wilson 2018

% Set the number of robots desired for the experiment.
N = 20;

% Set up the Robotarium object.
r = Robotarium('NumberOfRobots', N, 'ShowFigure', 'true');

% Set the number of iterations for the experiment. Each time
% step when deployed on the Robotarium hardware is ~0.033s.
iterations = 1000; % ~30 second experiment

for i = 1:iterations
    % Get the current positions of all the robots.
    x = r.get_poses();

    %% Main code here.

    % Set the velocities of the robots.
    r.set_velocities(1:N, dx);

    % Send the velocity commands to the set of robots.
    r.step()
end

r.debug(); %Prints errors that can cause the submission to be rejected.
```

FIGURE 6 An example code snippet from the Robotarium API in Matlab.

and links to the Robotarium simulation API repositories.

To submit their custom simulation scripts to be executed by the robots on the Robotarium testbed, users must first create an account on the website. In addition to being able to submit files through the website, this allows them to track the status of their submitted experiments, receive email updates when their experiments are completed, view the data and video returned from their experiments, and interface with the Robotarium support team. To create an account, potential operators are asked to complete an online form providing their name, email address, city, state/province, country, institute or employer (optional), position or title (optional), collaborators (optional), and reason(s) for using the Robotarium. The information about demographics and usage

reasons is collected to better identify the user base of the Robotarium and its motivations.

After creating an account, users enter the development cycle depicted in Figure 5. They begin with theorizing their multiagent control scheme; then, the produced algorithm must be implemented in the Robotarium simulation API. A basic example code snippet from the Robotarium Matlab API is pictured in Figure 6. When operators are satisfied with the performance of the simulated experiment, their main script (together with any custom supplementary functions) should be uploaded to the Robotarium website. When uploading the code through a simple drag-and-drop or file-searching interface, users must also complete an experiment form containing a brief experiment description, estimated experiment time, and the number of robots required for the experiment (Figure 7).

The submitted user code is stored indefinitely in a database on a server located in the Robotarium facility, but individuals can delete their experiments from the database at any time through the Web interface. This long-term storage allows operators to resubmit the same code at a later date through the Web interface without local access to the files, exchange individual files of previous submissions, or request that a Robotarium administrator help with an error. Submitted code is never accessed or viewed by a Robotarium team member without explicit user permission. Experiments are accessed through the database autonomously only by the server to be run on the platform. Individuals with additional concerns about privacy are able to upload non-human-readable files to be run (such as MEX files). Neither the Robotarium team nor the Georgia Institute of Technology asserts any rights to the code submitted or videos produced by remote users.

After operators submit their code to the website, it is pulled by the server and run in the simulation environment to check for problems, such as compilation mistakes, runtime problems, inclusion of libraries not available on the server, excessive runtime, robot collisions, or robots leaving the workspace. If any issues are found during this verification simulation, the experiment is rejected, and users receive a log file with specific details about the error. If the

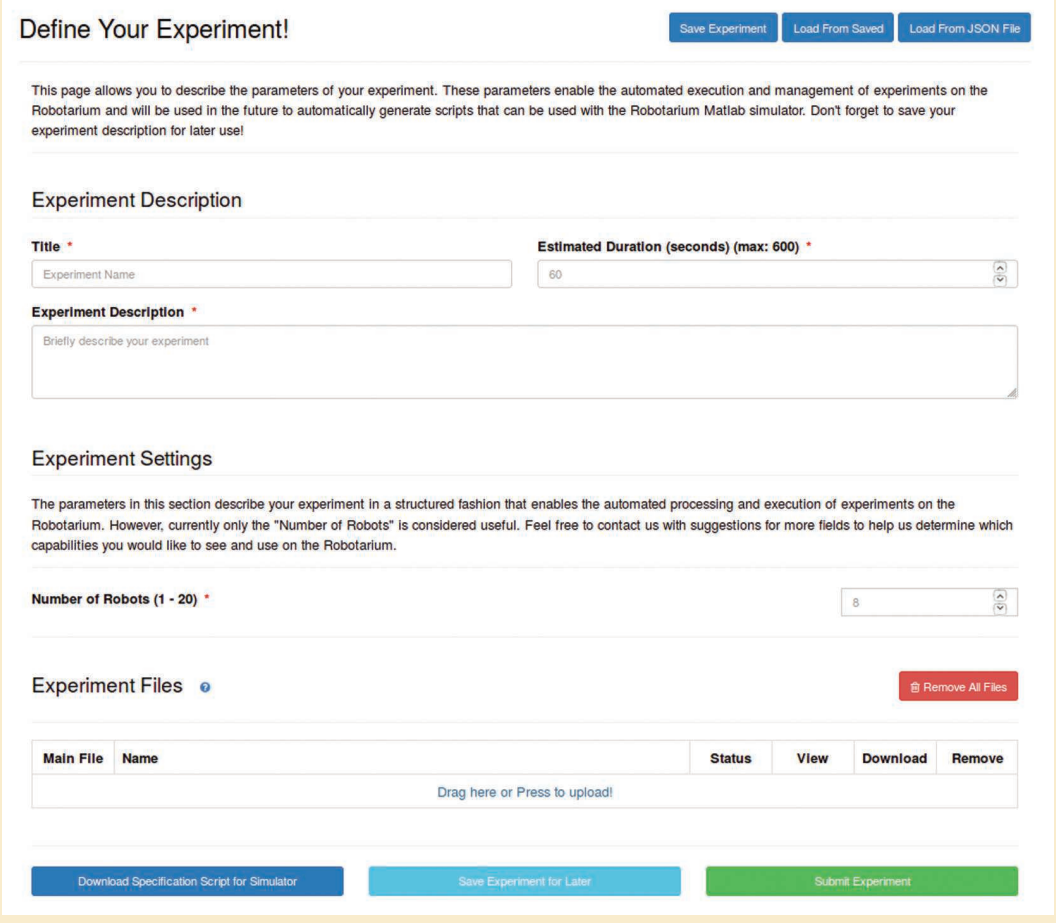

FIGURE 7 The Robotarium experiment submission form.

simulation runs without issue, it is forwarded to a first-come, first-served queue to be run on the Robotarium.

The server autonomously pulls the first verified experiment in the queue and queries the robots on the testbed to select the N most charged robots, where N is the number of robots specified by the individuals through the submission form on the website. If the robot with the Nth lowest voltage is not above a voltage threshold deemed safe to run, the system waits until enough robots are charged. When the system is ready to execute the pulled experiment, the N robots selected depart from their charging stations to a random configuration in the middle of the testbed, and the user-submitted script is run from this point.

All experiments executed are a centralized realization of the submitted algorithm. The central server runs the user simulation and receives data from individual robots as well as the Vicon tracking system. The unicycle control inputs, created by the operator's script and any required information from the testbed, are sent from the server to the individual robots to follow. During an experiment, the background individuals see in their simulation (without the simulated robots and bounding box) is projected onto the testbed. In past submissions, operators have created city environments for the robots to traverse; forests with fires that spread and are extinguished based on the robots' actions; visual markers, such as colored rings around each robot representing its internal behavior state; lines between robots showing simulated communication links; and many other visual feedback scenarios. Projections based on the location of each robot are made based on the position feedback from the Vicon tracking system. Figure 4 shows a snapshot from the returned overhead video of the simulation described previously (which makes 20 robots form the Georgia Institute of Technology logo) being executed on the Robotarium with the logo that was added to the background of the simulator projected onto the testbed.

After an experiment finishes executing or is rejected in the verification step, the system autonomously sends an email to the submitting users informing them about their experiment's updated status. In its current state, the Robotarium runs autonomously from 9 a.m. to 5 p.m. EST Monday through Friday. If individuals submit during operational times, they can expect results back within 10 min if the experiment queue is empty or, based on current demand, as long as overnight if there are many experiments in the queue ahead of theirs. If an experiment runs successfully, users are returned an overhead visual camera feed of the experiment and any data saved by their submitted script. The experiment completes when the operators are satisfied with the returned data. However, if errors occur during implementation, users can learn where their algorithm breaks down during the experiment, work to modify their theory and code to alleviate the implementation issue, and quickly reenter the submission cycle to test their improvements.

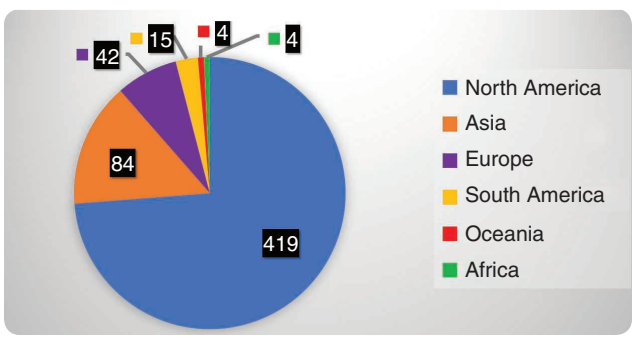

FIGURE 8 The Robotarium user account demographics, grouped by self-reported continental regions. The numbers indicate operators from each region. These data were gathered from user-submitted application forms.

UNDERSTANDING THE ROBOTARIUM USER BASE AND SUBMITTED EXPERIMENTS

This section discusses who the Robotarium users are, why they use the testbed, what types of experiments they submit, and what they hope to achieve. The Robotarium has users from every continent around the world except for Antarctica! The explicit ambition behind the Robotarium project is to democratize access to research-quality robotic hardware. This aspect of the mission involves making this robotic hardware not only available but also accessible and useful to the individuals who can benefit most from this platform. Thus, identifying current operators and understanding their goals ultimately determines whether the Robotarium has achieved its goal of being a widely used platform for distributed control of multirobot systems. The data presented in this section include user and experiment statistics collected from the opening of the Robotarium to the public from August 22, 2017 through July 5, 2019.

Robotarium User Demographics

In total, the Robotarium has had 568 individuals register to use the platform. Figure 8 shows numbers of people who have created Robotarium user accounts grouped by their self-reported continental regions. The platform is mainly used by researchers and students from North American universities, which is likely due to the media coverage in the United States and early adoption by the U.S. university system. However, there is still a significant presence of account holders from outside of North America, which demonstrates the breadth of impact of a remotely accessible testbed.

Of the operators who have created accounts, 219 have submitted custom experiments to be executed on the platform. Figure 9 shows the self-reported continental regions of residence for account holders who have submitted custom code to be run on the Robotarium. Currently, approximately 40% of users who create a Robotarium account have deployed an experiment on the physical testbed. There are various reasons why people create accounts without submitting any experiments to run. For example, some individuals explicitly state that they only want to use the Robotarium simulation API without submitting. In addition, multiple

accounts have been created by individuals working on team projects, with only one individual from the group submitting an algorithm. Users who have created accounts more recently are likely in the developmental stage of their algorithm and will submit work to the platform at a later date. Less-experienced operators may want to use the platform but do not yet have the programming or technical knowledge to develop a functioning algorithm. Finally, there are most likely users who create accounts after hearing or reading about the platform who simply never have the time to create a submission.

The self-reported reason each individual has for using the Robotarium varies. The most common reported theme is related to the well-known resource entry barrier to the multirobot domain. Due to limited finances, space, time, and/or expertise, many are unable to access robotics platforms and tracking equipment capable of testing their algorithms. Many universities, research institutions, and secondary schools simply do not have robotics facilities. For those that do, the platforms are typically not accessible or easily usable by those outside of the laboratory that maintains them or classroom that leverages them. One user from Brazil explained that his reason for using the Robotarium was to push beyond this barrier and experience the complications that arise when transferring theory to application, saying, "[The] Robotarium will be used to learn principles of robotics and control and automation engineering. There is a lack of hardware at the university where I study. Therefore, practical lessons of those subjects are affected as well as my learning experience. [The] Robotarium would be an awesome solution to this specific matter."

A less practical but equally important reason some individuals have chosen to use the Robotarium is to garner more interest in their research and be motivated by seeing their work come to life in hardware. Yunus Sahin, a Ph.D. candidate at the University of Michigan, used the Robotarium to test a temporal logic-based control synthesis for multiagent systems that resulted in publication [30]. In the experiment, 10 robots are deployed for various tasks (such as populating or avoiding some regions, visiting virtual charging stations, and not visiting some regions until a specific event occurs), simulating an emergency response scenario. When asked about the impact the Robotarium had on his work, Sahin responded, "The most important benefit of using the Robotarium, for me, is the motivation and inspiration I get from seeing my algorithm working on actual robots. Furthermore, my research attracts more attention when I use the Robotarium videos in my papers/presentations."

Some users test their algorithm on the Robotarium for convenience: it is a tedious process to set up and deploy algorithms on a robotic swarm, even if a person has access to them. It is extremely time-consuming to program low-level collision avoidance and motion controllers or debug common problems with hardware failure and communication protocols. The Robotarium gives operators a chance to sidestep these common complications. For example, Ravi Haskar (a Ph.D. candidate from Stanford University) used the Robotarium to test a distributed control strategy generated by deep reinforcement learning for fighting forest fires [31]. When asked why he used the Robotarium for his experiments, he said, "The Robotarium allowed me to conduct hardware experiments without worrying about the overhead of setting up robots and other associated hardware first. The simulation environment allowed me to easily determine if the setup of a given experiment would show the details I needed, without also running the entire hardware system."

Finally, some users choose to deploy their algorithms on the Robotarium due to its standardizing nature. Typically, educational modules or robotics experiments in multirobot systems are completed on hardware infrastructure that is not easily replicable. As a result, produced code and results are hard to share, verify, and compare when robots, computational hardware, and tracking systems vary across locations. However, any code developed for an algorithm deployed on the Robotarium can be easily shared, modified, and expanded to be implemented on the same hardware and produce metrics that are comparable without worry about different implementation hardware. Ramviyas Nattanmai Parasuraman, a postdoctoral research associate from Purdue University, used the platform to test his multipoint rendezvous algorithm [32] for this very reason, saying, "[The Robotarium] has greatly enabled realistic and immediate verification of our multirobot consensus and formation algorithms, through simulations and real experiments with the Robotarium platform. Using the platform, we were able to validate our algorithms and provide a reproducible code that can be implemented and tested anywhere."

The Robotarium user statistics offer confirmation this remote-access testbed has a worldwide impact. The individual feedback from Robotarium operators provides encouraging testimonials that, beyond global impact, the Robotarium is fulfilling its democratization goals. It is promising that operators agree the testbed is helping break down the resource barrier to entry for multirobot education and research, providing inspiration to its users, enabling algorithms to be tested in hardware at a more rapid pace by alleviating hardware

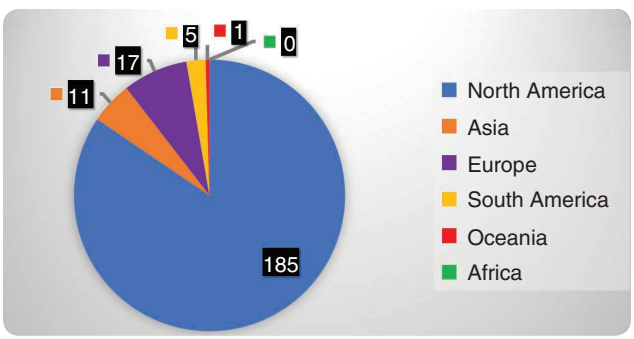

FIGURE 9 The users who have submitted custom experiments to be run on the Robotarium, broken into continental regions. The numbers indicate operators from each region. These data were gathered from user-submitted application forms.

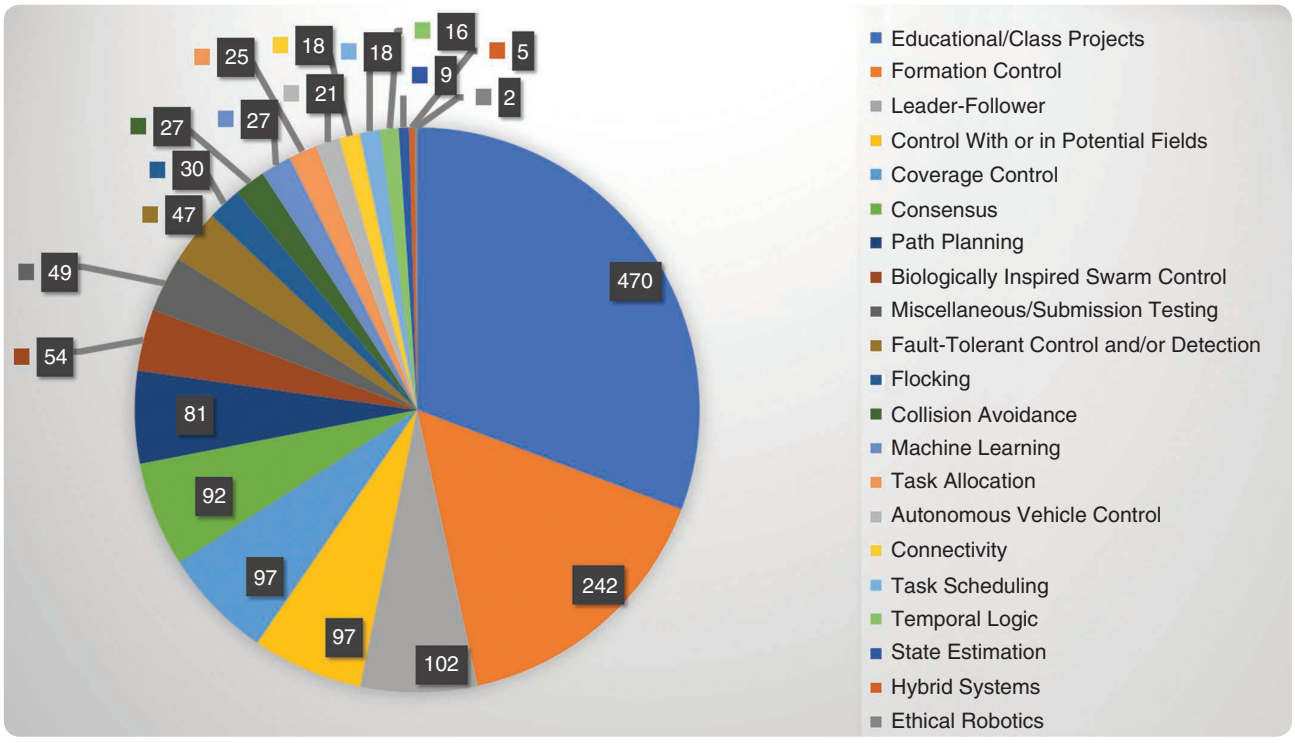

FIGURE 10 The types of experiments executed on the Robotarium, grouped into generalized research areas. The numbers indicate the quantity of each experiment type. The legend is sorted by the number of experiments in each area, from largest to smallest. These data were gathered from user-submitted experiment forms.

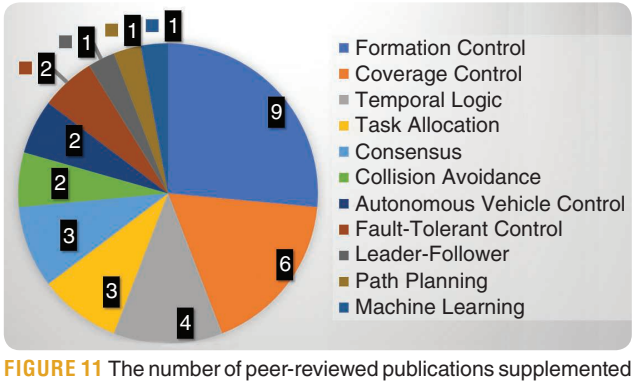

FIGURE 11 The number of peer-reviewed publications supplemented by Robotarium experiments, grouped into generalized research areas. The numbers indicate the quantity of experiments for each subject. The legend is sorted by the number of publications of each type, from largest to smallest. These data were gathered through Google Scholar based on the peer-reviewed article citations of [20] containing experimental validation from the Robotarium.

setup and debugging time, and making the first steps toward a standardized hardware testbed.

Submitted User Experiments

When the Robotarium was created, it was intended to be utilized by controls investigators focused on distributed systems who did not have the resources to test their multiagent algorithms. The supportive tools developed, robots used, and attainable experimental data were chosen to assist an audience well versed in robotics applications. Figure 10 depicts the types of experiments, grouped into generalized research areas based on self-reported descriptions, that have been executed on the Robotarium. Although most of these experiments have tested control and robotics problems, there are a significant number of use cases (and, thus, users) outside of the originally assumed demographic making use of the platform.

The majority of research-oriented submissions to the Robotarium test solutions to collective control problems (that is, formation and coverage control), which makes sense as they are the rich, multiagent domains that this platform was originally targeting. However, there are also interesting and unexpected uses of the Robotarium, such as for high-level autonomous vehicle control to alleviate traffic congestion and studies of individual insect behaviors that reproduce group-level performance seen in biological systems. Outside of the expected controls investigators and roboticists, these experiments show that mathematicians, traffic engineers, biologists, social scientists, and other investigators are also using the Robotarium. These use cases are unintended but valuable consequences of building a remotely accessible testbed.

The research-based submissions to the Robotarium have contributed to 35 published peer-reviewed papers [33]. Figure 11 shows the breakdown of these publication topics. These publications show that, beyond feeling useful to users, the experiments being run on the Robotarium are capable of providing research-quality data that are enabling investigators to publish their algorithms with hardware validation in a variety of fields.

Beyond experimental investigation, the platform is unexpectedly being leveraged substantially by collegiate-level educators and individual students for educational purposes, as indicated by their 470 submissions. A significant number of individuals are developing project-based courses around the Robotarium. For example, the Robotarium was used to supplement a minicourse introducing control of multiagent systems in France, where students implemented consensus and formation control algorithms. Some classes within the Georgia Institute of Technology are implementing Robotarium projects as an easy and safe way for students studying on campus or remotely to transfer their classroom knowledge to a controlled robotics laboratory setting. For example, it is used by the Georgia Institute of Technology's Cyberphysical Design and Analysis course, offered both online and on campus, for which students are tasked with developing Robotarium code to move a single robot between waypoints by developing their own open- and closed-loop controllers. The goal of this project is to have the students experience the purpose of feedback control systems through implementation on a physical system. It is also used by the Georgia Institute of Technology's Network Control Systems course, in a final project where students form teams to create competing capture-the-flag algorithms, shown in Figure 12.

In addition to specific minicourses and collegiate classroom assignments, some of these educational submissions come from individuals furthering their education by transferring theories learned in class to practice on the Robotarium. Some users have simply taken example code from the website to form shapes or drive to points and, then, adjusted the parameters to see the results. This group, with interests outside of research, includes secondary school teachers who are working to integrate the platform into their curriculum as a way to create more engaging projects that spark students' interest in mathematics, science, engineering, and programming.

The variety of experiment types deployed on the Robotarium shows that the project is achieving its democratizing goal of being accessible and usable by a diverse population. These use cases demonstrate that the Robotarium is not only being widely used in a geographic sense but also in an application sense.

ADDRESSING REMOTE-ACCESS CHALLENGES TO IMPROVE USER EXPERIENCE AND HARDWARE SAFETY

The opening of the Robotarium to the world and its high adoption rate revealed many challenges associated with a remote-access multirobot testbed. The reported experiment demographics and operator feedback have driven the development of the Robotarium to make it more robust, useful, and accessible to the user base. This includes the development of tools and guides that help those who are not robotics experts as well as those with experience control swarms of physical robots. This remote-use experience has also required safe automation routines that strike a balance between enabling our user base and guaranteeing the safe long-term operation of the testbed. This section will briefly describe the general challenges associated with robotics remotely accessible by an unknown audience and detail the Robotarium approaches to addressing these challenges and operator feedback.

Building a User Base and Providing Equal Access

Arguably, the most important challenge to overcome with a remotely accessible testbed is to develop a user base. A platform, such as the Robotarium, is only as strong as its operators, who deploy experiments and provide feedback. The famous quote from the movie *Field of Dreams*, "If you build it, they will come," definitely does not hold for the Robotarium! The growth of the user base has been driven by mostly directed strategies, but it has also come from unintended side effects of the outreach efforts.

When remote-access use of the Robotarium was first allowed, advertising was first directed to the academic robotics community through common mailing lists and discussions at conferences. This established a small but experienced and devoted user base. This small initial group of operators mostly comprised roboticists, who understood hardware difficulties and provided valuable ideas for useful tools that would better the experience.

After the initial usage by the academic robotics community, media outlets were contacted about the platform, its capabilities, and its relevance in the hope that they would report on it. This was successful and most notably produced an article in *The Wall Street Journal* [34]. The initial reporting on the Robotarium caused a chain reaction of more outlets covering the platform, which allowed the news to reach an extremely broad audience.

Traditional advertising for and fortunate reporting about the Robotarium contributed greatly to the creation of a user base, but these are not entirely responsible for the account holder growth; part of the contribution comes from interest garnered from outreach efforts. The Robotarium hosts biweekly open houses and scheduled tours that provide engaging educational experiences for those visiting the Georgia Institute of Technology. Although the main purpose of

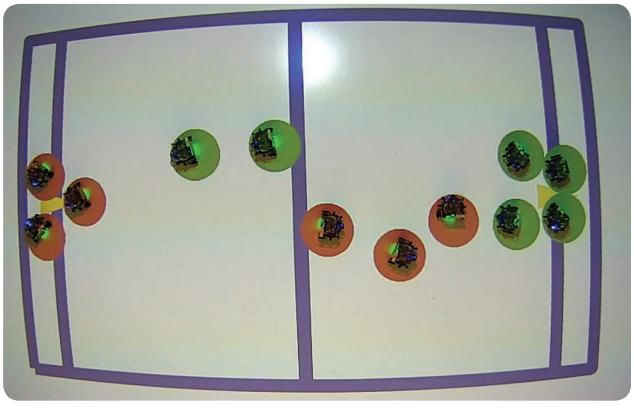

FIGURE 12 A snapshot of a project from the Georgia Institute of Technology's Network Control System class, where robots are competing in a game of capture the flag.

The growth of the user base has been driven by mostly directed strategies, but it has also come from unintended side effects of the outreach efforts.

these tours is community outreach, they have had an unintentional effect of growing the user base as visitors began to use the testbed, and their experience has spread by word of mouth to their acquaintances.

Due to the healthy growth of the Robotarium user base, the frequency at which experiments are being submitted is increasing. This makes scheduling experiment runtime in a fair way an important challenge. For the past two months, typical usage of the Robotarium has been approximately 5–15 experiments per day. The weeks leading up to major robotics conferences or class project submission deadlines can be even more demanding, and experiment submissions can spike to between 30 and 60 per day. Most experiments run between 5 and 10 min each, which can easily consume an entire day when considering robot downtime for charging. To solve the scheduling problem, experiments are put into a first-come, first-served queue based on time of submission. If a user submits a large number of experiments simultaneously, other experiments submitted afterward are slotted between them. This ensures that no single operator can monopolize the system. The first-come, first-served scheduling method was chosen over a standard time slot reservation system since individuals are not present at the Robotarium to observe their experiment firsthand. Also, since the system is autonomous, delays caused by charging between experiments and other unforeseeable circumstances would compound, making it impossible to schedule exact time slots.

Assistive Software Tools for Users

An unexpected challenge with opening the Robotarium to the world was—and still is—creating an accessible API for the diverse user base. Originally, the Robotarium was released with several well-documented example scripts that implemented standard multirobot algorithms, such as rendezvous, formation control, and leader–follower. It was assumed that the user base would be multirobot researchers with enough previous experience to understand the control of differential-drive robots and the mathematics behind these algorithms. As is evident from the experiment characteristics and based on feedback from Robotarium users, this assumption was not correct.

The Robotarium API provides many prebuilt functions that assist with algorithm development and common implementation challenges. The most heavily used support functions provide

- transforms of high-level abstractions to create motion models that better incorporate the physical platform's motion constraints
- controls for individual robot motion through position, pose, and velocity controllers
- checks for whether robot state conditions have been achieved
- initialization routines that position robots randomly in the arena
- collision avoidance functions that avoid collisions with other robots and simulated obstacles in the environment
- functions to create simulated connections between individual robots to highlight network and graph-theoretic properties.

These functions help users prototype higher-level algorithms by removing the need for them to develop low-level controllers or obstacle-avoidance behaviors. For example, an individual may wish to assume simplified dynamics (for example, a single integrator) over the unicycle dynamics needed to control the robots. Additionally, for operators with limited robotics, controls, and programming experience, providing controllers with intuitive inputs (such as a set of waypoints for the robot to drive to or desired velocity direction) lifts the burden of controlling multiple nonlinear, nonholonomic differential-drive robots.

When creating control commands for the robots on the Robotarium to follow, users must send desired linear velocity, v, and angular velocity, ω, commands that correspond to the unicycle model in (1). However, it is sometimes desirable to utilize simplified dynamics (such as a single integrator) when designing algorithms. Accordingly, the Robotarium provides utilities to map single-integrator dynamics to unicycle dynamics. This can be accomplished through standard proportional-integral-derivative controllers that enable the differential-drive robot to track the single-integrator control vector. However, on the Robotarium, this is achieved through a near-identity diffeomorphism (NID) between the single-integrator and unicycle models described in "Mapping Single-Integrator Dynamics to Unicycle Control Commands."

Users have access to supporting functions that apply this NID with a default offset distance l, or they may choose to set the offset to whatever distance they would like. However, selecting a small l can result in high rotational velocity commands that may be unrealizable by the actuators on robots on the Robotarium. Some individuals do not consider this practical limitation for diffeomorphism. For this reason, additional variations of this function have been provided that also limit the produced angular velocity.

Not all users want to operate at the velocity-input level. For example, some of the educational uses for secondary

Mapping Single-Integrator Dynamics to Unicycle Control Commands

When differential-drive robots are being controlled, it can be more intuitive to consider single-integrator velocity commands. The simplicity of assuming that a robot can follow desired linear velocity commands directly is also amenable to the analysis of high-level mobile robot algorithms. However, the common differential-drive robot chassis cannot follow single-integrator commands directly due to the nonholonomic constraint imposed by the no-slip condition on its wheels (that is, similar to a car, differential-drive robots cannot drive sideways). One effective solution to this problem is to use a near-identity diffeomorphism (NID) between the desirable single-integrator model and the more accurate unicycle model [S1]. The main idea of this method is to perturb the unicycle model slightly and show that there exists a diffeomorphism between a lower-dimensional version of the unicycle model dynamics and single-integrator dynamics.

Consider a differential-drive robot in a global reference frame O with full-state $\mathbf{x} = [x\ y\ \theta]^\top$, as depicted in Figure S1. Let $\mathbf{x}_p = [x\ y]^\top$ represent the global position of the robot. Then, consider the following output of the state defined by

$$s(\mathbf{x}) = \mathbf{x}_p + l\begin{bmatrix}\cos(\theta)\\ \sin(\theta)\end{bmatrix}, \tag{S1}$$

where $0 < l \in \mathbb{R}$. Geometrically, $s(\cdot) \in \mathbb{R}^2$ represents a point orthogonal to the wheel axis of the robot along the perpendicular bisector of the axle at a distance l. The body velocity of the robot in the global frame can be modeled through the unicycle model

$$\dot{\mathbf{x}} = \begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}. \tag{S2}$$

Note that $[v\ w]^\top$ represents the unicycle control inputs of linear and angular velocity, respectively. The possible instantaneous velocities of the look-ahead point, $s(\mathbf{x})$, are not restricted by the nonholonomic constraint of the differential-drive platform. For intuition, consider a right-handed local coordinate frame for the robot where the x-direction is in the same direction as the linear velocity vector v in Figure S1. Given a pure rotational velocity command, the look-ahead point will instantaneously translate in the y-direction of the local frame. Similarly, if a pure translational velocity command is followed, the look-ahead point will instantaneously translate in the x-direction of the local frame. Through the NID relating (S1) to (S2), it is possible to determine the single-integrator dynamics of $s(\mathbf{x})$ given unicycle velocity control inputs. If this relation is invertible, it will be possible to calculate unicycle velocity control inputs that cause $s(\mathbf{x})$ to follow given single-integrator dynamics.

Differentiating (S1) with respect to time yields

$$\dot{s}(\mathbf{x}) = \dot{\mathbf{x}}_p + l\dot{\theta}\begin{bmatrix}-\sin(\theta)\\ \cos(\theta)\end{bmatrix}. \tag{S3}$$

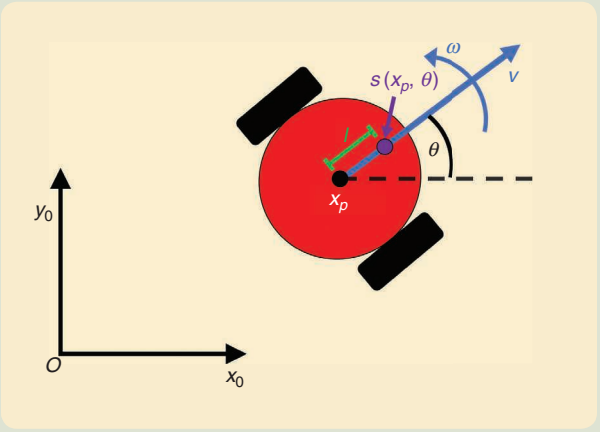

FIGURE S1 A cartoon representation of a differential-drive robot, with the variables used in the diffeomorphism superimposed. The unicycle model state variables consist of x- and y-positions ($x_p = [x\ y]^\top$), represented by the black point in the center of the robot, and the heading of the robot (θ) with respect to a global reference frame O. The unicycle model control inputs, represented by the blue arrows, consist of the linear velocity (v) and angular velocity (ω) of the body. The perturbed reduced state variables ($s(x_p, \theta)$) are represented by the purple point projected a distance l in the direction of the velocity vector.

Substituting the unicycle dynamic model for $\dot{\mathbf{x}}_p$ and $\dot{\theta}$ provides the desired relation

$$\dot{s}(\mathbf{x}) = R_l(\theta)\begin{bmatrix}v\\ \omega\end{bmatrix}, \tag{S4}$$

where

$$R_l(\theta) = \begin{bmatrix}\cos(\theta) & -l\sin(\theta)\\ \sin(\theta) & l\cos(\theta)\end{bmatrix}. \tag{S5}$$

The inverse of $R_l(\theta)$, given by

$$R_l^{-1}(\theta) = \begin{bmatrix}\cos(\theta) & \sin(\theta)\\ -\frac{1}{l}\sin(\theta) & \frac{1}{l}\cos(\theta)\end{bmatrix}, \tag{S6}$$

exists for every θ, as long as $l \neq 0$. As such, any desired dynamics for the point $s(\mathbf{x})$ [that is, $\dot{s}(\mathbf{x})$] may be mapped to a corresponding unicycle control input. That is, the input generated by a single-integrator-based algorithm may be considered as $\dot{s}(\mathbf{x})$ and mapped to an input for the differential-drive robot as

$$\begin{bmatrix}v\\ \omega\end{bmatrix} = R_l^{-1}(\theta)\dot{s}(\mathbf{x}). \tag{S7}$$

Selecting a small l can result in high rotational velocity commands that may be unrealizable when applied to the differential-drive robot.

REFERENCE

[S1] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential epsilon-tracking and epsilon-stabilization of first-order nonholonomic SE(2) vehicles," in *Proc. IEEE Amer. Control Conf.*, vol. 6, Nov. 2002, pp. 4690–4695.

school students involve developing fundamental programming skills and turning to the Robotarium as a motivating use case. Thus, they do not want to take a distracting tangent into velocity control to enable students to use the platform. Research-oriented individuals want to test and implement path-planning algorithms that generate waypoints for the robot to traverse without concern for the low-level controller moving the robot between the waypoints. To facilitate use cases such as these, the Robotarium supplies a number of position controllers based on

Barrier Functions for Collision Avoidance

Safety in dynamical systems, which can be represented as the forward invariance of a subset of the state space, has been studied since 1940, when Nagumo provided necessary and sufficient conditions for set invariance [S2]. These conditions were rediscovered and refined 30 years later in [S3] and [S4]. In particular, for a dynamical system

$$\dot{x} = f(x),$$

with $x \in \mathbb{R}^n$ and given a safe set $C \subset \mathbb{R}^n = \{x \in \mathbb{R}^n \mid h(x) \geq 0\}$ (defined as the zero superlevel set of a smooth function $h : \mathbb{R}^n \to \mathbb{R}$, where $(\partial h/\partial x)(x) \neq 0$ for all $x \in \partial C$), the boundary of the safe set C, the following necessary and sufficient condition for set invariance holds:

$$C \text{ is invariant} \quad \Leftrightarrow \quad \dot{h}(x) \geq 0 \quad \forall x \in \partial C.$$

In their current formulations, control barrier functions (CBFs) were introduced in [37] to ensure the safety of dynamical systems. Given a general nonlinear system in control affine form

$$\dot{x} = f(x) + g(x)u, \tag{S8}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the input, and f and g are two Lipschitz continuous vector fields, the goal is to ensure that the state x never leaves the safe set C. CBFs, defined in Definition S1, can be used to achieve this goal.

DEFINITION S1: CONTROL BARRIER FUNCTION [S5]

Consider the dynamical system in control affine form (S8), and let $C \in \mathcal{D} \subset \mathbb{R}^n$ be the zero superlevel set of a continuously differentiable function $h : \mathcal{D} \to \mathbb{R}^n$; that is, $C = \{x \in \mathcal{D} \mid h(x) \geq 0\}$. If there exists a Lipschitz continuous extended class $\mathcal{K}_\infty$ function α such that, for the system (S8) and for all $x \in \mathcal{D}$,

$$\sup_{u \in \mathcal{U}} \{L_f h(x) + L_g h(x) u\} \geq -\alpha(h(x)), \tag{S9}$$

then h is a CBF. $L_f h(x)$ and $L_g h(x)$ denote the Lie derivatives of h in the directions of the vector fields f and g, respectively.

The following theorem summarizes two important properties of CBFs, which can be used to ensure both forward invariance and asymptotic stability of safe sets.

THEOREM S1 [S5]

Consider a dynamical system in control affine form (S8) and the zero superlevel set $C \subset \mathbb{R}^n$ of a continuously differentiable function h, as defined earlier. Then, any Lipschitz continuous controller u such that (S9) holds for all $x \in \mathcal{D}$ renders the set C forward invariant and asymptotically stable, that is,

$$x(0) \in C \Rightarrow x(t) \in C \quad \text{for all } t \geq 0 \quad \text{(forward invariance)}, \tag{S10}$$

$$x(0) \notin C \Rightarrow x(t) \to \in C \quad \text{as } t \to \infty \quad \text{(asymptotic stability)}, \tag{S11}$$

where $x(0)$ denotes the state x at time $t = 0$, and $x(t) \to \in C$ means the distance between x and the set C decreases; that is, x converges to the set C.

Implementation: A Centralized Approach

Consider a set of robots whose motion is modeled as single integrators with dynamics

$$\dot{s}_i = u_i, \tag{S12}$$

where $i \in \{1, \ldots, N\}$ for a group of N robots. The ensemble state and control input are

$$s = [s_1^\top, \ldots, s_N^\top]^\top, \; u = [u_1^\top, \ldots, u_N^\top]^\top. \tag{S13}$$

A pairwise collision-avoidance constraint may be encoded as the barrier function

$$h_{ij}(s_i, s_j) = \|s_i - s_j\|^2 - d^2, \tag{S14}$$

where $d > 0$ denotes the distance by which s_i and s_j must remain separated. Note that

$$\nabla_{s_i} h_{ij}(s_i, s_j) = 2(s_i - s_j). \tag{S15}$$

Let $\nabla_s h_{ij}^k(s_i, s_j)$ indicate the kth 2D component of the vector $\nabla_s h_{ij}(s_i, s_j)$. Then,

$$\nabla_s h_{ij}^k(s_i, s_j) = \begin{cases} 2(s_i - s_j) & k = i \\ 2(s_j - s_i) & k = j \\ 0 & \text{otherwise.} \end{cases} \tag{S16}$$

Ensuring that

$$\nabla_s h_{ij}(s_i, s_j)^\top u \geq -\gamma h(s_i, s_j)^3 \tag{S17}$$

for every $i \in \{1, \ldots, N-1\}$, $j \in \{i+1, \ldots, N\}$ guarantees that all robots avoid collisions [36], [38].

The following quadratic program (QP) produces such a safe controller u^* for every time step:

$$u^*(s) = \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \|u - u^{\text{nom}}\|^2 \tag{S18}$$

$$\text{subject to } \nabla_s h_{ij}(s_i, s_j)^\top u \geq -\gamma h(s_i, s_j)^3, \tag{S19}$$

proportional control of the look-ahead point used in the NID described in "Mapping Single-Integrator Dynamics to Unicycle Control Commands."

The Robotarium also provides pose controllers, such as the nonlinear controller from [35]. In this case, the system, with state $\mathbf{x} = [x\ y\ \theta]^\top$, may be stabilized to a goal pose $\mathbf{x}_{\text{goal}} = [x_{\text{goal}}\ y_{\text{goal}}\ \theta_{\text{goal}}]^\top$. With $\mathbf{x}_p = [x\ y]^\top$ and $\mathbf{x}_{p,\text{goal}} = [x_{\text{goal}}\ y_{\text{goal}}]^\top$, let

$$e = \|\mathbf{x}_p - \mathbf{x}_{p,\text{goal}}\|, \tag{3}$$

$$\forall i \in \{1, \ldots, N-1\},\ j \in \{i+1, \ldots, N\}. \tag{S20}$$

Here, u^{nom} denotes the nominal control input. Therefore, u^* represents a controller that avoids collisions while remaining as close as possible to the nominal control input.

The program in (S18) results in a controller that not only prevents collisions but is also minimally invasive (that is, it alters the nominal controller as infrequently and as little as possible). This property implies that the robotic hardware is protected while also ensuring that the deployed algorithm runs effectively.

From a control perspective, (S18) is a fully centralized controller. The QP requires position and desired control input information from all pairs of robots and coordinates the controllers accordingly. In a real-world setting, such information may not be available to all agents; however, controlled testbeds can take advantage of the global tracking and communication in a laboratory setting to solve such programs. In fact, this centralization ensures maximum coordination among the robots, which reduces the chance of deadlock (that is, $u^*(s) = \mathbf{0},\ u^{\text{nom}} \neq \mathbf{0}$).

Implementation: A Decentralized Approach

From a computational perspective, the QP in (S18) has been solved at 100 Hz for 40 robots on the Robotarium; moreover, this runtime is possible with substantially underoptimized code. For larger numbers of robots, the runtime may be improved by decentralizing (S18), allowing the program to be solved in parallel. In [20], it is shown that a decentralized version of the QP in (S18) may be solved at 200 Hz (in parallel) for more than 100 robots. Moreover, this parallelization decreases the computational complexity of adding more robots.

The decentralized QP is formulated based on the observation that (S18) depends on satisfying the inequality

$$\nabla_s h_{ij}(s_i, s_j)^\top u \geq -\gamma h(s_i, s_j)^3 \tag{S21}$$

for each $i \in \{1, \ldots, N-1\},\ j \in \{i+1, \ldots, N\}$. If every pairwise constraint is included in the QP, then this QP becomes centralized. In particular,

$$\nabla_s h_{ij}(s_i, s_j)^\top u = \nabla_{s_i} h_{ij}(s_i, s_j) u_i + \nabla_{s_j} h_{ij}(s_i, s_j) u_j. \tag{S22}$$

The centralized QP simultaneously chooses u_i and u_j to satisfy the barrier-function constraint. However, the following alternate formulation separates the selection of u_i and u_j.

Consider the QP

$$u_i^*(s) = \underset{u \in \mathbb{R}^2}{\operatorname{argmin}} \|u - u^{\text{nom}}\|^2, \tag{S23}$$

$$\text{subject to } 2(s_i - s_j)^\top u_i \geq -\frac{1}{2}\gamma h_{ij}(s_i, s_j)^3, \tag{S24}$$

$$\forall j \in \{1, \ldots, N\}/\{i\}. \tag{S25}$$

If every agent solves (S23), then

$$\nabla_s h_{ij}(s_i, s_j)^\top u = \nabla_{s_i} h_{ij}(s_i, s_j) u_i + \nabla_{s_j} h_{ij}(s_i, s_j) u_j, \tag{S26}$$

$$\geq -\frac{1}{2}\gamma h_{ij}(s_i, s_j)^3 - \frac{1}{2}\gamma h_{ij}(s_i, s_j)^3, \tag{S27}$$

$$\geq -\gamma h_{ij}(s_i, s_j)^3. \tag{S28}$$

Accordingly, all robots remain collision free. This formulation still requires that all robots have information about all other robots, yet the actual QP in (S23) may be solved in a decentralized fashion (that is, each agent may solve the QP individually).

Unicycle Dynamic Considerations

The controller from (S18) or (S23) may be mapped to unicycle dynamics given the near-identity diffeomorphism described in "Mapping Single-Integrator Dynamics to Unicycle Control Commands." Specifically, the QP in either (S18) or (S23) ensures that

$$\|s_i - s_j\| \geq d \tag{S29}$$

for all $i \in \{1, \ldots, N\},\ j \neq i$. Setting s_i as the near-identity diffeomorphism look-ahead point, $s_i(\mathbf{x}_i)$, at an offset distance l yields a mapping between the single-integrator dynamics and unicycle control input. Moreover, ensuring that

$$d \geq r + 2l, \tag{S30}$$

where $r > 0$ denotes the size of the robot, implies that

$$\|\mathbf{x}_{p,i} - \mathbf{x}_{p,j}\| \geq r. \tag{S31}$$

Thus, the robots remain a minimum safe distance from each other, ensuring their safety.

REFERENCES

[S2] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proc. Physico-Math. Soc. Jpn. 3rd Ser.*, vol. 24, pp. 551–559, May 1942.

[S3] J.-M. Bony, "Principe du maximum, inégalité de harnack et unicité du probleme de cauchy pour les opérateurs elliptiques dégénérés," *Ann. Inst. Fourier (Grenoble)*, vol. 19, no. 1, pp. 277–304, 1969.

[S4] H. Brezis, "On a characterization of flow-invariant sets," *Commun. Pure Appl. Math.*, vol. 23, no. 2, pp. 261–263, 1970.

[S5] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th European Control Conf. (ECC)*, 2019, pp. 3420–3431.

The impetus behind giving users remote access to physical robots is to allow for ideas that work on paper and in simulations to be hardened against real-world issues.

$$\phi = a\tan 2(y_{\text{goal}} - y, x_{\text{goal}} - x) - \theta_{\text{goal}}, \tag{4}$$

$$\alpha = \phi + (\theta_{\text{goal}} - \theta). \tag{5}$$

The controller in [35] is given by

$$v = \gamma\cos(\alpha)e,\ \ w = k\alpha + \gamma\frac{\cos(\alpha)\sin(\alpha)}{\alpha}(\alpha + h\phi), \tag{6}$$

with $\gamma, k, h > 0$. Although discontinuous at the origin, this controller asymptotically stabilizes the system to $\mathbf{x}_{\text{goal}}$. The Robotarium provides a function that allows users to set the gains, γ, k, h, or use the default tuned gains.

Based on experience with running hundreds of experiments, many submissions fail when using the provided well-tuned position and pose controllers due to infeasible guard conditions to change behaviors. For example, in a waypoint-following example, robots will proceed to the next waypoint only after reaching the previous one. Many individuals make the condition of reaching a desired position, pose an exact equivalence, or with unachievable bounds (for example, within a 1-mm envelope and a 1° orientation error). This is possible in simulation, which can mislead users inexperienced with hardware into thinking this is also realizable experimentally. In implementation, the tracking system has unavoidable noise in its returned data, and robot actuators have limitations on minimum achievable speeds, making it impossible to reach any pose state exactly. For this reason, the Robotarium also provides state checking with default envelopes that are achievable by the system. As with all other provided functions, users can customize these envelopes to make them larger or smaller, as desired.

Balancing Control Intent, User Failure, and Hardware Safety

A fundamental challenge associated with creating a research platform for broad, remote usage is that it must be flexible enough to allow users to submit a variety of experiments in different stages of refinement. As a consequence, a failed experiment may be as useful as a successful one for research purposes and learning progress. Thus, the system must support a truly vast array of possible types of controllers and scenarios. At the same time, the system must be safe, and operators should not be given the freedom to accidentally break the robots on the Robotarium. A careful tradeoff between flexibility and safety must be struck.

The impetus behind giving users remote access to physical robots is to allow for ideas that work on paper and in simulations to be hardened against real-world issues; experiments should be expected to fail due to unexpected or unmodeled circumstances. The most common implementation difficulties that can cause experiment failures are computational times that are too long for real-time control, aggressive individual robot controllers that do not account for motor dynamics, control commands that do not account for the nonlinear nature of the robots, communication delays and drops, and tracking or robot motion error. Although some of these failures result in errors that do not pose a threat to the hardware, others can cause collisions or motor failure.

If operators are allowed too much freedom, these unexpected problems in their experiments may cause the robots to collide and (at best) get stuck or (at worst) damage each other. To mitigate this issue, as discussed in [20], the Robotarium deploys control barrier certificates [36]–[39] as a way of augmenting the uploaded controllers in a provably "minimally invasive" manner. In this application of barrier certificates, the robots execute controllers that are as close to the user-specified controllers as possible in a least-squares sense, while satisfying a differential safety constraint (that is, the barrier certificate). Details of this approach are given in "Barrier Functions for Collision Avoidance."

Although this construction is inherently safe and strikes a balance between flexibility and safety, it does mean that what is tested is not exactly what the individual uploaded, as the barrier certificate safety measures may alter the motion of individual robots significantly. For some operators, this is perfectly acceptable, as they are concerned only with high-level planning algorithms. Users who are more concerned with lower-level control may save their control inputs before and after the barrier certificates are applied to determine when the algorithm was affected. With this information, individuals can alter the algorithm or develop their own obstacle-avoidance behavior and, if it passes the validation simulation, run their own algorithm.

CONCLUSIONS AND FUTURE DEVELOPMENTS

The influence of and interest in the Robotarium has exceeded initial expectations. The platform is being used by controls, robotics, and other unexpected investigators around the world to improve the speed at which they can validate their work and make their distributed control algorithms more resilient to often-overlooked real-world

The Robotarium exposes users around the world to the theory-to-practice gap, accelerates the implementation of theory on distributed control of multirobot systems, and, ultimately, democratizes access to a state-of-the-art research facility.

issues. Beyond research, it is being used to accelerate the learning of future controls engineers and roboticists worldwide by bringing experience with physical systems into classrooms that otherwise would not be able to afford or maintain robotics platforms. Based on user feedback, the access to a testbed, such as this, is impacting individuals' research and education. However, from the experiences of operating a publicly available testbed, there are improvements that can be made to expand its capabilities, allow more investigators outside of the robotics field to easily use it, and simplify its maintenance and operation.

The Robotarium serves as the first-of-its-kind remote-access testbed. Such systems improve substantially when they are widely adopted and used by a diverse community. Currently, the testbed is being used by researchers and educators from a multitude of backgrounds for a wide range of applications. With the continual adoption and refinement of the Robotarium, it can serve as a starting point for remote-access testbeds in other fields to accelerate inquiry and improve education.

ACKNOWLEDGMENTS

We would like to thank Daniel Pickem for his foundational work on the Robotarium and Paul Wilson, III, for developing the original Robotarium website interface. We also extend our gratitude to Yunus Sahin, Ravi Haskar, and Ramviyas Nattanmai Parasuraman for sharing their experiences using the Robotarium and how it has impacted their research. Additionally, we thank them for allowing us to share parts of their investigations that involved the Robotarium and their impressions of the platform.

This work was supported by the U.S. National Science Foundation under grants 1531195 and 1544332 and by the U.S. Office for Naval Research under grant N00014-17-1-2323.

AUTHOR INFORMATION

Sean Wilson (Sean.T.Wilson@gatech.edu) is a research engineer in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He received the M.S. and Ph.D. degrees in mechanical engineering from Arizona State University in 2017 and the B.A. degrees in physics and mathematics from the State University of New York at Geneseo in 2012. He served as a postdoctoral fellow at the Georgia Institute of Technology. His research interests include remote-access robotic hardware and control of multiagent robotic systems.

Paul Glotfelter is a Ph.D. candidate in the School of Robotics at the Georgia Institute of Technology and a member of the Georgia Robotics and Intelligent Systems Laboratory. His research interests include the composition of specifications and controller synthesis for robotic systems. He received the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology and the B.S. degree in computer engineering from York College of Pennsylvania.

Li Wang received the B.S. degree in mechanical engineering from Huazhong University of Science and Technology in 2012, the M.S. degree in electrical and computer engineering from Clemson University in 2014, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2018. He is currently a senior autopilot controls engineer at Tesla, Inc., Palo Alto, California. His research interests include control theory, robotics, and machine learning.

Siddharth Mayya received the B.Tech. degree in electronics and communication engineering from the Manipal Institute of Technology in 2014 and the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology in 2016. He is currently working toward the Ph.D. degree in electrical and computer engineering at the Georgia Institute of Technology. His research interests include swarm robotic systems, bio-inspired algorithms, and control theory.

Gennaro Notomista is a robotics Ph.D. student at the Georgia Institute of Technology. He received the B.S. degree in mechanical engineering from the Universit degli Studi di Napoli "Federico II" in 2012, the M.Eng. degree in automotive engineering from the Technische Hochschule Ingolstadt in 2015, and the M.S. degree in mechanical engineering from the Universit degli Studi di Napoli "Federico II" in 2016.

Mark Mote is a Ph.D. candidate in robotics at the Georgia Institute of Technology. His research interests include trajectory planning, runtime assurance, and formal verification of cyberphysical systems.

Magnus Egerstedt is the Steve W. Chaddick School Chair and professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He received the M.S. degree in engineering physics and the Ph.D. degree in applied mathematics from the Royal

Institute of Technology, Stockholm, Sweden, and the B.A. degree in philosophy from Stockholm University, Sweden. He was a postdoctoral scholar at Harvard University. His research interests include control theory and robotics, with particular focus on the control and coordination of complex networks, such as multirobot systems, mobile sensor networks, and cyberphysical systems. He is a Fellow of the IEEE and has received a number of teaching and research awards, including the Ragazzini Award from the American Automatic Control Council, the Outstanding Doctoral Advisor Award and the HKN Outstanding Teacher Award from Georgia Tech, and the Alumni of the Year Award from the Royal Institute of Technology.

REFERENCES

[1] D. Zhang, S. K. Nguang, and L. Yu, "Distributed control of large-scale networked control systems with communication constraints and topology switching," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 47, no. 7, pp. 1746–1757, July 2017. doi: 10.1109/TSMC.2017.2681702.

[2] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004.

[3] K. Elamvazhuthi and S. Berman, "Optimal control of stochastic coverage strategies for robotic swarms," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2015, pp. 1822–1829. doi: 10.1109/ICRA.2015.7139435.

[4] M. Wang, W. Daamen, S. P. Hoogendoorn, and B. van Arem, "Cooperative car-following control: Distributed algorithm and impact on moving jam features," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1459–1471, May 2016. doi: 10.1109/TITS.2015.2505674.

[5] F. Gao, X. Hu, S. E. Li, K. Li, and Q. Sun, "Distributed adaptive sliding mode control of vehicular platoon with uncertain interaction topology," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6352–6361, Aug. 2018. doi: 10.1109/TIE.2017.2787574.

[6] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robot. Autom. Mag.*, vol. 19, no. 1, pp. 24–39, Mar. 2012. doi: 10.1109/MRA.2011.2181683.

[7] G. Habibi, Z. Kingston, W. Xie, M. Jellins, and J. McLurkin, "Distributed centroid estimation and motion controllers for collective transport by multi-robot systems," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2015, pp. 1282–1288. doi: 10.1109/ICRA.2015.7139356.

[8] H. Ardiny, S. Witwicki, and F. Mondada, "Are autonomous mobile robots able to take over construction? A review," *Int. J. Robotics, Theory Applicat.*, vol. 4, no. 3, pp. 10–21, 2015. [Online]. Available: http://ijr.kntu.ac.ir/article_13385.html

[9] J. León, G. A. Cardona, A. Botello, and J. M. Calderón, "Robot swarms theory applicable to seek and rescue operation," in *Intelligent Systems Design and Applications*, A. M. Madureira, A. Abraham, D. Gamboa, and P. Novais, Eds. Cham, Switzerland: Springer-Verlag, 2017, pp. 1061–1070.

[10] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *Proc. 14th IEEE Int. Conf. Advanced Video and Signal Based Surveillance (AVSS)*, Aug. 2017, pp. 1–6. doi: 10.1109/AVSS.2017.8078478.

[11] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, Mar. 2015.

[12] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton, NJ: Princeton Univ. Press, 2010.

[13] D. Pickem, M. Lee, and M. Egerstedt, "The gritsbot in its natural habitat: A multi-robot testbed," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2015, pp. 4062–4067. doi: 10.1109/ICRA.2015.7139767.

[14] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 3293–3298.

[15] S. Wilson et al., "Pheeno, a versatile swarm robotic research and education platform," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 884–891, July 2016. doi: 10.1109/LRA.2016.2524987.

[16] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Proc. Int. Conf. Simulation, Modeling, and Programming for Autonomous Robots*, 2010, Vol. 6472. Berlin: Springer, pp. 51–62.

[17] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Proc. 2nd Int. Conf. Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR'10*, pp. 51–62. Berlin: Springer-Verlag, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1947545.1947555

[18] O. Michel, "Webots: Professional mobile robot simulation," *Int. J. Advanced Robotics Syst.*, vol. 1, no. 1, pp. 39–42, 2004.

[19] R. Tellez, "A thousand robots for each student: Using cloud robot simulations to teach robotics," in *Robotics in Education*, M. Merdan, W. Lepuschitz, G. Koppensteiner, and R. Balogh, Eds. Cham, Switzerland: Springer-Verlag, 2017, pp. 143–155.

[20] D. Pickem et al., "The robotarium: A remotely accessible swarm robotics research testbed," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, July 2017, pp. 1699–1706. doi: 10.1109/ICRA.2017.7989200.

[21] L. Paull et al., "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017, pp. 1497–1504. doi: 10.1109/ICRA.2017.7989179.

[22] "AI driving Olympics," The Duckietown Foundation, 2018. [Online]. Available: https://www.duckietown.org/research/AI-Driving-olympics/

[23] S. Mohan, A. Saenz-Otero, S. Nolet, D. W. Miller, and S. Sell, "Spheres flight operations testing and execution," *Acta Astronaut.*, vol. 65, nos. 7–8, pp. 1121–1132, 2009.

[24] J. Enright, M. Hilstad, A. Saenz-Otero, and D. Miller, "The SPHERES guest scientist program: Collaborative science on the ISS," in *Proc. IEEE Aerospace Conf.*, vol. 1. Apr. 2004, p. 46. doi: 10.1109/AERO.2004.1367588.

[25] Vicon Motion Systems Ltd., "Vicon." Accessed on: Jan. 2, 2019. [Online]. Available: https://www.vicon.com/products/camera-systems/vantage

[26] Elpcctv.com, "HD 1080p mini ATM IP camera with 3.6mm HD lens," 2018. Accessed on: Jan. 2, 2019. [Online]. Available: http://www.elpcctv.com/hd-1080p-mini-atm-ip-camera-with-36mm-hd-lens-p-62.html

[27] Optima USA, "Optima USA EH200ST," 2018. Accessed on: Jan. 2, 2019. [Online]. Available: https://www.optoma.com/us/product/eh200st/

[28] The Robotarium Organization, "Robotarium Matlab simulator," 2019. [Online]. Available: https://github.com/robotarium/robotarium-matlab-simulator/

[29] The Robotarium Organization, "Robotarium Python simulator," 2019. [Online]. Available: https://github.com/robotarium/robotarium_python_simulator/

[30] Y. E. Sahin, P. Nilsson, and N. Ozay, "Synchronous and asynchronous multi-agent coordination with cLTL+ constraints," in *Proc. IEEE 56th Annu. Conf. Decision and Control (CDC)*. Dec. 2017, pp. 335–342. doi: 10.1109/CDC.2017.8263687.

[31] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 1067–1074.

[32] R. Parasuraman, J. Kim, S. Luo, and B. Min, "Multipoint rendezvous in multirobot systems," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 310–323, 2020. doi: 10.1109/TCYB.2018.2868870.

[33] D. Pickem et al., "The robotarium: A remotely accessible swarm robotics research testbed," Google Scholar. Accessed on: July 5, 2019. [Online]. Available: https://scholar.google.com/scholar?cites=15167587093940557062&as_sdt=80005&sciodt=0,11&hl=en

[34] M. Korn, "This robot lab has no idea what its robots are doing," *Wall St. J.*, Aug. 15, 2017. Accessed on: Jan. 2, 2018. [Online]. Available: https://www.wsj.com/articles/a-room-full-of-remotely-operated-robots-what-could-go-wrong-1502810136

[35] G. Casalino, M. Aicardi, A. Bicchi, and A. Balestrino, "Closed-loop steering for unicycle-like vehicles: A simple Lyapunov like approach," *IFAC Proc. Volumes*, vol. 27, no. 14, pp. 335–342, 1994. doi: 10.1016/S1474-6670(17)47335-6.

[36] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, 2017. doi: 10.1109/TRO.2017.2659727.

[37] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decision and Control*, Dec. 2014, pp. 6271–6278. doi: 10.1109/CDC.2014.7040372.

[38] 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.

[39] 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.