The Robotarium: An open, remote-access, multi-robot laboratory

Daniel Pickem[1], Eric Squires[2] and Magnus Egerstedt[3]

Abstract— **Multi-robotic testbeds are an integral part of multi-agent research, yet they are expensive to develop and operate, which obstructs access. This in turn makes them unaffordable for most but a select few researchers at well-endowed universities, slows the rate of progress of multi-agent research, and limits the number of educational multi-robot tools available to students. In this paper we propose a shared, remotely accessible multi-robot testbed - the *Robotarium* - that aims at remedying these issues and enabling researchers to remotely access a state-of-the-art multi-robot test facility. The Robotarium hosts custom-designed miniature robots that are able to recharge their batteries autonomously, enables users to remotely upload code to the robots, and allows users to run their multi-robot experiments remotely.**

I. Introduction

Multi-robot research has seen an almost explosive growth during the last decade, with a number of coordinated control algorithms being developed for tasks ranging from environmental monitoring (e.g. [7], [9], [36]) to collective material handling (e.g. [28]). This growth has been driven by a combination of algorithmic advances and increased hardware miniaturization and cost reduction. However, despite the reduction in cost, it is still a prohibitively costly proposition to go from theory and simulation, via a few robots, all the way to a truly deployed large-scale robot system, and there are only a handful of laboratories around the world that can field massive numbers of robots in the air, under water, or on the ground, e.g., [17], [23], [11], [33], [34].

To advance multi-robot research further, actual deployment is crucial since it is increasingly difficult to faithfully simulate all the issues associated with making multiple robots perform coordinated tasks due to the increased task complexity. To bridge this theory-practice gap, a typical indoor multi-robot research lab is equipped with a number of mobile robots, a camera system for indoor localization, as well as additional supporting infrastructure (communications, computation, software, safety, machining, etc.). The bare-bones cost for such a lab can be as high as \$250K[1]. As a consequence, research in multi-agent robotics has largely become a resource competition as opposed to an ideas competition and the high barrier to entry excludes many researchers from participating. This paper describes the Robotarium – an open, remote-access multi-robot system explicitly designed to address these issues by providing access that is flexible enough to allow for a number of different scientific questions to be asked and different coordination algorithms to be tested. And, although a number of elegant, remote-access robot systems have been developed in the past ([17], [11], [31], [4]), what makes the Robotarium different is its explicit focus on supporting multi-robot research, as opposed to, say, education or single-robot systems.

At its core, the Robotarium is a multi-robot laboratory, where mobile robots can coordinate their behaviors in a collaborative manner. And, during experiments, the Robotarium provides users with both a live video stream of the experiments as well as the scientific data produced by the experiment. In this paper, we discuss how this multi-robot laboratory is structured and, in particular, how the explicit focus on being a remote-access research platform informs the design. The outline of the paper is as follows. Section II presents an overview of currently available testbeds in the multi-robot and sensor networks domain. Section III discusses design considerations that guided the development of the Robotarium prototype. Section IV explains the hardware components of the current instantiation of the Robotarium in detail while Section V presents the accompanying simulation interface. Finally, Section VI shows an exemplary remote experiment and Section VII concludes this paper.

II. Background

Today's robotics research relies increasingly on experimental verification on robotic hardware. Yet, few shared or remotely accessible testbeds are available to the general research community. The Robotarium's main goal is not to provide another isolated testbed. Instead, it aims at relieving researchers of the requirement to set up their own facilities by providing a shared testbed on which time slots for usage can be scheduled.

In the world of shared multi-agent testbeds, only a handful are publicly available.[2] One example of a shared robotic platform is the PR2 Remote Lab at Brown University ([25], [26], [31]), which makes a PR2 robot available for public use. Instead of a single robot, the Robotarium aims at providing access to a whole swarm of robots.

*This research was supported by Grant No. 1531195 from the U.S. National Science Foundation.

[1]Daniel Pickem is a Ph.D. student in robotics at the Georgia Institue of Technology, Atlanta, GA 30332, USA daniel.pickem@gatech.edu

[2]Eric Squires is a Ph.D. student in Electrical and Computer Engineering at the Georgia Institue of Technology, Atlanta, GA 30332, USA esquires3@gatech.edu

[3]Magnus Egerstedt is with the School of Electrical and Computer Engineering, Georgia Institue of Technology, Atlanta, GA 30332, USA magnus@gatech.edu

[1]This estimate is based on a typical setup using Khepera wheeled ground robots, an OptiTrack motion capture system, and the required communications and computing infrastructure.

[2]A comprehensive overview of robotic and sensor network testbeds is given in [15]

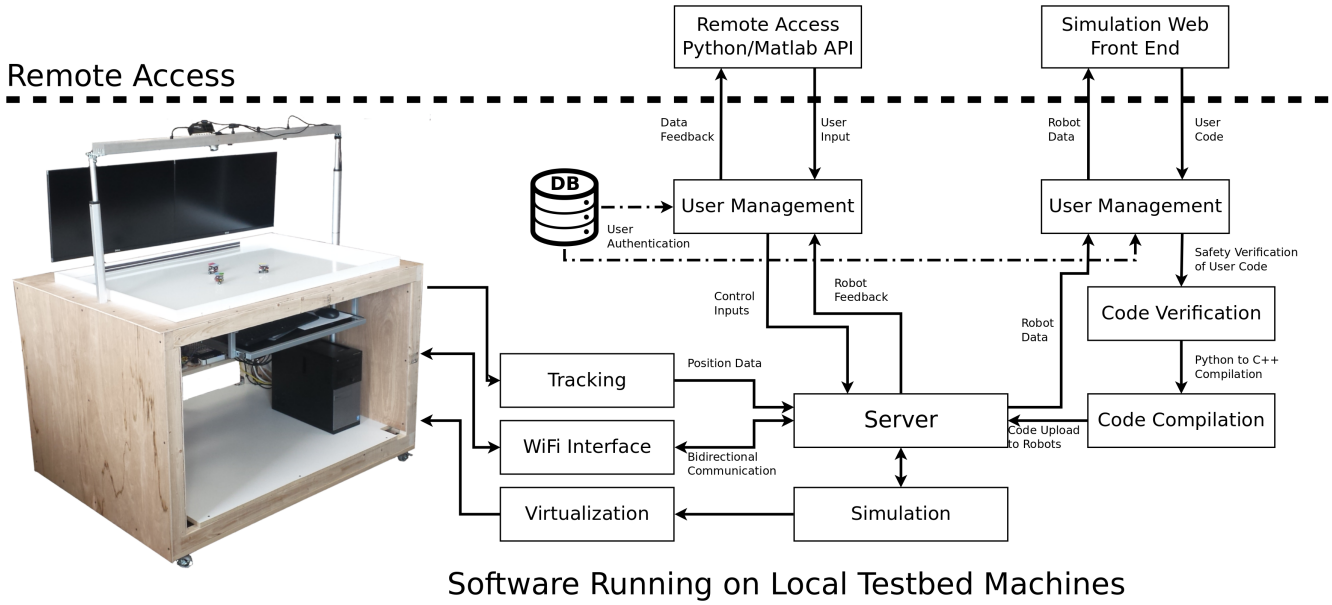


Fig. 1: System architecture overview. The current prototype includes components that are executed locally on Robotarium infrastructure as well as user-facing components that run on remote user machines (APIs or simulation front end). Three components interact directly with the robot hardware - tracking, wireless communication, and virtualization. The remaining components handle user management, code verification and upload, as well as coordination of user data and testbed-generated data.

Another remotely accessible multi-robot testbed designed for research purposes was proposed in [23], [21]. Though similar to the Robotarium in capabilities and infrastructure, the used robots are significantly larger and more expensive than the Robotarium robots. A similar testbed - although designed for indoor and outdoor use - was developed at Oklahoma State University. The cooperative multivehicle testbed (COMET, [8]) also relies on a small number of comparably expensive robots.

Remotely accessible testbeds have also proven useful in educational contexts. For instance, *Robotic Programming Network (RPN)* has been made available recently (see [4]). Unlike the Robotarium, RPN relies mostly on simulation (only a single Nao robot is available), but provides the required infrastructure to control robots in an open, remote-access fashion. Additionally, a remote robotics lab designed for investigating localization and path planning is discussed in [19]. The system allows simulation code to be developed in a web-based user interface remotely and then run on the hardware system. One disadvantage of this testbed is that only a single robot is available for public use. Finally, Robotnacka is a remotely accessible testbed aimed at young students learning about programming and robotics [29]. It is available 24/7 through an online interface, has automatic charging through docking stations to create a maintenance-free environment, and allows virtual robots to be used. Though several physical robots are provided, extending this system to a whole swarm of robots would require significant amounts of space due to the robots large footprint. The comparably small size of the Robotarium robots (see Section IV) allows tens of robots to be used on a similarly sized testbed.

Robotics testbeds have also emerged from the wireless sensor network domain. The Mobile Emulab, for example, closely resembles the Robotarium in that it provides a shared, remotely accessible multi-agent testbed with mobile nodes [17]. Mobile Emulab's main focus, however, lies on sensor networks and evaluating mobility-related network protocols. Thus, it uses mobility only as a means to establish static sensor network topologies and not as an inherent component of dynamic multi-agent experiments. Unlike the Robotarium, Mobile Emulab requires extensive resources - both in terms of space requirements and financial resources.

Similar to Mobile Emulab, MiNT [10] and MiNT-m [11] aim at reducing the space requirements to carry out multi-hop wireless research. Their computing nodes are only capable of limited motion because of their tethered nature. Each mobile node is controlled in a centralized fashion and mobility is used again to establish static sensor topologies. ORBIT, another system whose main purpose is the evaluation of wireless network protocols on sensor networks, emulates motion of the nodes by dynamic binding of host computers to wireless transceivers [32] and is also remotely accessible. However, nodes are not truly mobile like the robots of the Robotarium are.

A more recently developed testbed - CONET (Cooperating Objects Network of Excellence, [12]) - combines wireless sensor networks with mobile robotic platforms. The testbed covers a total area of 500m^2, houses six large mobile robots and numerous stationary sensor nodes. Though remotely

accessible and impressive in its size and potential to host a large number of robots, it requires significant resources and is unlikely to be easily replicable like the Robotarium is.

Two other shared testbeds in this context are the DeterLab [24] and the PlanetLab [6], both of which are providing capabilities to networking and security researchers. Although successful in their domain, these testbeds cannot be used for realistic cyber-physical systems research because of their immobile computing nodes.

A number of isolated testbeds have been developed at various institutions for a number of locomotion modalities. Hovercraft vehicles have been used in testbeds at Caltech [16] and the University of Illinois [35]. Testbeds at the University of Pennsylvania rely on micro UAVs [22] as well as a combination of ground and aerial vehicles [5], [14]. Similarly, the multi-vehicle testbed at MIT [18] and the testbed at Brigham Young University [20] use a combination of ground and aerial vehicles. Although their focus is an entirely different one than the Robotarium's, these testbeds are in principle capable of remote operation.

Among the variety of inexpensive robotic testbeds that have been presented in the literature one stands out above all for the sheer scale - the Kilobot testbed. Although it is a closed, non-shared testbed, it is used for large scale experiments in 2D self-assembly and collective transport with up to 1024 robots ([33], [34]). However, the Kilobot's locomotion modality (vibration motors) makes it less suitable for a general purpose multi-robot testbed.

As part of the Robotarium's mission of providing access to multi-robot testbeds, the underlying robotic architecture needs to fit as many applications as possible, which is why the core of the testbed are wheeled miniature ground robots. Specifically we use the GRITSBot, a miniature robot we designed (see [30]) with similar capabilities to most commonly used wheeled robotic platforms used in academia. This robot architecture was chosen because a number of tasks can be accomplished with generic wheeled ground robots, for example vehicle routing [2], coverage control [3], or collective exploration [27]. In this sense, the Robotarium's aims at providing not just an affordable but also flexible testbed.

III. 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. Similar to open-source software that provides access to high quality software, the Robotarium aims at providing universal access to state-of-the-art robotic infrastructure and enabling fast prototyping of multi-agent and swarm robotic algorithms. The Robotarium was designed primarily as a research and educational tool that will function as a gateway to higher STEM education on one hand and as an accessible and capable research instrument on the other hand. As such it is conceivable that a pervasive robotic testbed such as the Robotarium will have to exhibit a subset or all of the following high-level characteristics to fulfill its intended use effectively.

- Simple and inexpensive replicability of the system - both the testbed itself and the robots it contains.
- Intuitive interaction with and data collection from the testbed
- Tight and seamless integration of the simulation workflow for algorithm prototyping and code execution on the robots.
- Minimization of both cost and maintenance effort while keeping the robots and testbed extensible.
- Built in safety and security measures to protect the system from damage and misuse.

These desired high-level characteristics can be mapped onto more specific constraints that inform the hardware design as well as the software architecture - both low-level controls on the robots as well as the coordinating server applications. An overview of our current instantiation of such a remote-access multi-robot testbed is shown in Fig. 1. Whereas this implementation already serves as a fully functional small-scale prototype, a full-fledged Robotarium implementation should include the following features.

- Large numbers of low-cost robots (on the order of hundreds, see Section III-A)
- Convenience features to simplify maintenance of large collectives of robots (see Section IV-B)
- Immersive user-experience through a fully remotely accessible testbed with live video and data streaming (see Section III-B)
- Public interface to allow users to schedule time on the testbed (see Section III-C)

These design requirements can be categorized along three dimensions, specifically robot design, user-experience, and network design.

A. Robots

The Robotarium is meant to provide a well integrated, immersive user experience with the smallest possible footprint, and features that allow a large swarm to be maintained effortlessly. Such tight integration is only possible with custom hardware. At the core of the Robotarium are therefore our custom-designed robots - the GRITSBots (see [30]). These miniature robots also ensure that the user community is not limited to simulating robots locally, but is also able to deploy its own low-cost, high-performance robots in conjunction with the Robotarium robots - robots that already integrate the remote-access aspect as a key characteristic of their design. With ease of deployment in mind, we have designed the GRITSBots that are low-cost, user-friendly, simple to maintain, and tie in seamlessly with the Robotarium. The design specifics of the robots are described in detail in Section IV.

B. User Experience

Being an integrated research instrument, the user experience needs to be a vital part of the design of a testbed

such as the Robotarium. On the instrumentation side, the Robotarium is equipped with cameras that provide a video stream of the experiments, tracking cameras for localization, and projectors for adding virtual robots to the Robotarium floor that behave as if they were actual physical robots. These virtual robots enable interaction with other virtual and physical robots alike, where such interactions include both collision and obstacles avoidance.

C. Network Design

The shared nature of the Robotarium requires precautions to be taken against unauthorized access or abusive use of the system. Access to such a testbed will therefore have to be managed through a user verification and authentication system (for example LDAP in combination with SSH). Users will only be able to access the robots they have been approved to use during the their assigned time slot. These access control mechanisms ensure security for the Robotarium by managing outside threats. Closely linked to security is the safety aspect of the system, i.e. ensuring that the system does not damage itself. Therefore a vital component of the Robotarium's software architecture will have to be code verification to guarantee the avoidance of damage to the hardware through faulty, corrupted, or malicious code.

In addition to safety and security, network design has to take delay-tolerance into account. It is conceivable that a remotely accessible testbed has to accommodate user-testbed interaction on different timescales and with different delay tolerances. As shown in Fig. 1, two options for remote access are enabled by the Robotarium. Delay-insensitive applications and algorithms can remotely close the feedback loop through the provided APIs. This method will prove useful for quick prototyping and testing of algorithms that do not require high update rates, large amounts of data to be transfered, or a large number of robots to be involved. This use-case would apply to largely autonomous robots that require occasional user input to, for example, switch operating modes.

Delay-sensitive applications that require closing the feedback loop locally can make use of the second track of remote operation. User code is initially simulated in the provided web front end. After initial user testing and verification, the code is then uploaded to the Robotarium, undergoes formal code verification, compilation, and upload to the robots, which then execute it in a purely local fashion. Sample applications include large-scale swarm experiments requiring large numbers of robots or delay-critical applications that require robots to react quickly to sensor information.

IV. Prototype Design

In this section, we detail the current instantiation of the Robotarium. It should be noted, already at this point, that the Robotarium, by design, has to evolve over time in response to user needs in order to provide an effective research instrument as opposed to a static showcase. Fig. 1 shows the currently implemented components of the Robotarium. The testbed hardware, the tracking camera feedback loop, wireless communications, the coordinating server application, APIs, and simulation infrastructure are fully functional, while initial versions of the code verification and compilation modules are being developed. In this section, we will focus on the robot design as well as the testbed setup.

A. Robot Design

The Robotarium leverages the GRITSBot, a miniature robot that we introduced in [30]. It is an inexpensive differential drive miniature robot featuring a modular design that allows hardware capabilities to be adapted easily to different tasks. A key feature that makes the GRITSBot the basis for the Robotarium is that it allows for a straightforward transition from typical multi-robot systems because it closely resembles popular platforms in capabilities and architecture (such as the Khepera robots).

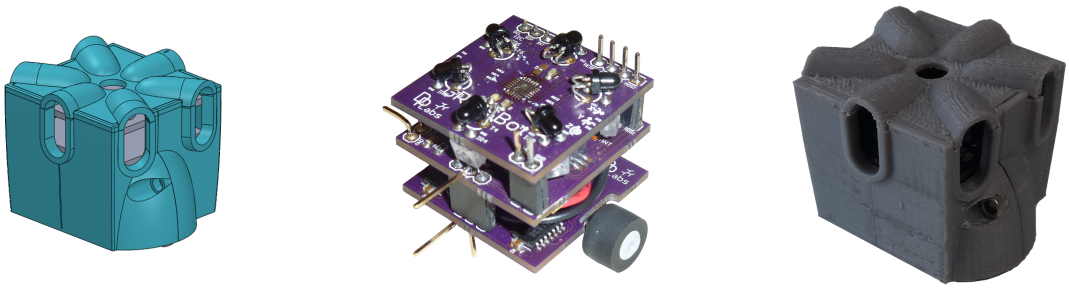
The robot's main features include (i) high resolution and accuracy locomotion through miniature stepper motors, (ii) range and bearing measurements through infrared distance sensing, (iii) global positioning system an overhead camera system, and (iv) communication with a global host through a wireless transceiver. This section briefly discusses these features and summarizes design changes compared to the first revision of the robot in [30].

1) Actuation: The actuation system of the GRITSBot is based on two miniature stepper motors, whose main advantage is the ease of acquiring accurate odometric information. By replacing common wheel encoders with counting steps, one obviates the need for motor speed estimation.

2) Processing: The main processor on the GRITSBot is an ESP8266 microcontroller running at up to 160 MHz, fast enough to handle wireless communication, pose estimation, low-level control of the robot (including the nonlinear velocity and position controller), as well as high-level behaviors. A second microcontroller on the motor board - an Atmega 88 - is responsible for motor control, i.e. ensuring the precise timing required to run the stepper motors at speeds up to 8 rotations per second.

3) Communication: The main ESP8266 microcontroller doubles as a WiFi transceiver supporting the IEEE 802.11 B/G/N standards. Unlike the wireless transceivers used on the GRITSBot in [30], WiFi offers much higher bandwidth but comes at the cost of higher power consumption (on average 150 mA). To offset the reduced battery life, we have doubled battery capacity compared to [30]. The benefits of WiFi however far outweigh its increased power consumption. WiFi offers a reliable communication channel based on standard UDP sockets and a single WiFi access points is able to service hundreds of clients.

4) Sensing: The current sensor board of the GRITSBot houses six infrared distance sensors with a range of up to 10 cm and can be equipped with a digital compass, gyroscope, and an accelerometer. An additional battery voltage and current sensor is located on the main board. The modular architecture of the robot allows to easily extend or change capabilities of the robot by replacing the sensor board with



(a) 3D model (b) Electronics of the robot (c) Prototype of robot shell

Fig. 2: The current revision of the GRITSBot.

a custom board or simply stacking a second sensor board on top.

B. Testbed

The design of the GRITSBot allows a single user to easily operate and maintain a large collective of robots through built-in features such as (i) automatic sensor calibration, (ii) automatic battery charging, (iii) wireless (re)programming, and (iv) automated registration with the overhead tracking system of the robots after powering them up. A future feature that will significantly enhance multi-agent experiments is local communication, which the robots sensor board supports with its dual-use infrared distance sensors.

1) Camera System: The overhead tracking system is based on standard webcams (currently Microsoft LifeCam Studio HD cameras). The video stream is fed into an OpenCV-based blob tracking algorithm recovering blob positions that are associated with individual robots. While x/y position information is recovered through the blob tracking, the orientation of the robots has to be estimated. Currently, this orientation estimation is done through Kalman filtering on the robots with observations being the position updates supplied by the camera system.

2) Recharging: Arguably the most crucial component of a self-sustaining and maintenance-free testbed is an automatic recharging mechanism for the robots. The GRITSBot has been designed for autonomous recharging through two extending magnetic prongs that can connect to magnetic charging strips built into the arena walls. This setup together with global position control through the camera feedback loop allows the GRITSBot to autonomously recharge its battery.

3) Sensor Calibration: Since the IR distance sensors of the robot only measure voltages that correspond to distances, we have to establish a mapping between measured voltage and distance. Variations in sensor quality require each robot to be calibrated separately and possibly repeatedly throughout its lifetime. Therefore, we have developed a calibration tool that provides an automated mechanism for the calibration of the robots IR sensors. A detailed description of the automated calibration feature can be found in [30].

4) Wireless Reprogramming: The main ESP8266 microcontroller supports over-the-air programming (OTA), which enables wireless reprogramming of individual robots, groups of robots, or even reprogramming of the whole swarm in a broadcast fashion. It is even possible for one robot to reprogram another, which offers an array of research challenges in the area of wireless security, as well as evolutionary and collaborative robotics.

V. Simulation

In addition to the hardware framework described in Section IV, we have also developed a simulator for the Robotarium that supports more rapid prototyping than hardware alone can support. Simulator code can additionally be used for real hardware experiments, reducing the turnaround time for testing a prospective algorithm. More specifically, the simulator incorporates the following:

- A communication framework that models network congestion and bit rate errors. In addition, the simulation provides a flexible format for specifying packets.
- Sensing including 6 IR sensors, battery current, and battery voltage levels.
- Customizable global computer behavior for investigating the interaction of slow, global information with fast, local information.
- Customizable robot behavior including a low level controller, an obstacle avoidance controller, a neighborhood manager, and a state estimator. Users are not limited to a homogeneous implementation.
- A web-based graphical user interface that uses WebGL. A screenshot of the simulator running a rendezvous controller with 100 robots is shown in Fig. 3.
- Logging for every robot at every timestep of a simulation returned in a simple data structure for analysis.

The implementation emphasizes usability by using a high-level language (Python) and incorporating a modular framework for specifying robot and global computer behavior. Further, a library of behaviors for the global computer and robot components are provided for users to choose from rather than having to implement all custom components.

While usability is paramount in the simulation, reasonable levels of fidelity have been achieved without sacrificing runtime by writing a large portion of the simulation in

C/C++.[3] In particular, the physics model incorporates simple collisions and commanded wheel velocities, the communications model incorporates network congestion and bit rate errors, and actuators are subject to failure as power levels decrease. Thus, users can investigate for example the effects of translating a unicycle model to a differential drive robot, the effects of scale on communication capabilities, or the effect of robots dropping out of a formation due to power constraints.

Code written for the simulator should be all that is necessary to conduct an experiment with the actual hardware. In the current implementation, we achieve this through a Python API which calls the simulation robot controller for commanded wheel velocities. In the future, we intend to translate simulation Python code into equivalent C/C++ code that can be run on the robots for a truly distributed implementation. This can be achieved through the Python Abstract Syntax Tree module, which will allow us to step through logical chunks of user Python code and translate it into equivalent C/C++ code. We will also give users the option of verifying that their code compiles prior to submitting their code for the hardware experiment.

Thus, users will be able to write their algorithms in a high level language, test them in an easy-to-use simulation to verify their algorithms, and submit them for actual hardware experiments for a tight integrated environment.

VI. Remote Experiments

As shown in Fig. 1, two options for remote interaction with the Robotarium are available. On the one hand, a user can control the robots through a Python or Matlab API. On the other hand, code can be prototyped remotely in simulation and then uploaded to the robots after a verification and code rewriting step. These two options operate on different time scales since the feedback loop is either closed remotely over a network (in case of API usage) or locally on the robots (in case of code upload). In this section we will explain both methods in detail and present preliminary findings based on a remote experiment using the API approach.

A. Remote Interaction with the Robotarium

Referring to Fig. 1, the left branch of the remote access procedures is currently already fully implemented. Remote users can log in to our system through a secure shell (SSH) and execute their custom code through either a Python or Matlab API. These APIs allow the user to send commands (target positions, velocities, etc.) to the robot as well as retrieve data from them (position updates, battery status, etc.). Due to varying degrees of latency observed in closing the feedback loop in experiments, it proved advisable to run the code locally on the Robotarium machines. Hence, Matlab code was uploaded to and executed on the testbed's computing infrastructure. In this local setup, it was possible to close the velocity control feedback loop over the local WiFi network at sufficiently high update rates and minimal delays.

The alternative option of code submission and automatic formal verification opens up promising avenues for further research in the area of remote-access testbeds. Currently the process of code submission, verification, and upload to the robots is done manually for reasons of security and safety of the system. Full automation and integration of code verification into the software framework requires the guaranteed avoidance of damage to the robots even in the face of potentially malicious code (for example through the use of barrier functions to avoid undesirable states of the system [1]). In its final implementation, a user provides prototype Python code in the simulation web front end and submits it to the Robotarium at which point the code will be automatically verified (as mentioned in Section V). Since at the current time, this is still a manual process as opposed to a fully automated toolchain, usage of the API for remote access is the more feasible option for rapid prototyping.

B. Circular Path Following Experiment

As a first remote-access experiment, a relatively small-scale case was tested. The remote experiment shown in this section is a circular path following controller based on [13] with code remotely provided by Dr. Jorge Cortés from UCSD. The provided Matlab implementation was uploaded to the Robotarium machines and run locally. The algorithm controlled a team of three robots (see Fig. 4 for snapshots of the experiment). Minor modifications to the code had to be made to update positions through the Robotarium back end instead of through Matlab simulation. The control inputs (namely linear and rotational velocities) were computed according to the following control law.

$$\begin{aligned} v_i &= \frac{k_1 \theta_{i,des}}{2\pi} \\ \omega_i &= \frac{v}{r} + k_2 r \left[(x_i - c_x)\cos(\theta_i) + (y_i - c_y)\sin(\theta_i) \right] \end{aligned}$$

where $c = \{c_x, c_y\}$ are the coordinates of the center of the circle and r its radius. The constants k_1 and k_2 are appropriately chosen gain values (in this experiment $k_i = 1.0$). The desired heading of robot i is computed according to

$$\theta_{i,des} = \min_{j \in \mathcal{N}_i} \operatorname{atan2}\left(y_j - c_y, x_j - c_x\right) - \theta_{i,rel}$$

where $\mathcal{N}_i$ is the current neighborhood of robot i and $\theta_{i,rel} = \operatorname{atan2}\left(y_i - c_y, x_i - c_x\right)$ is its relative angle.

VII. Conclusion

In this paper, we have presented a remotely accessible, shared testbed that makes state-of-the-art multi-robotic systems available to a large audience of researchers and students. The Robotarium presents an affordable and flexible testbed that can be cheaply replicated and is easily extensible. These features also guided the design of the robot at the core of the Robotarium - the GRITSBot. This miniature low-cost robot was designed around the idea of remote access and

[3] Python written in C/C++ cannot be stepped through with a debugger so pure python equivalent code has been provided for users to easily step through otherwise compiled code.

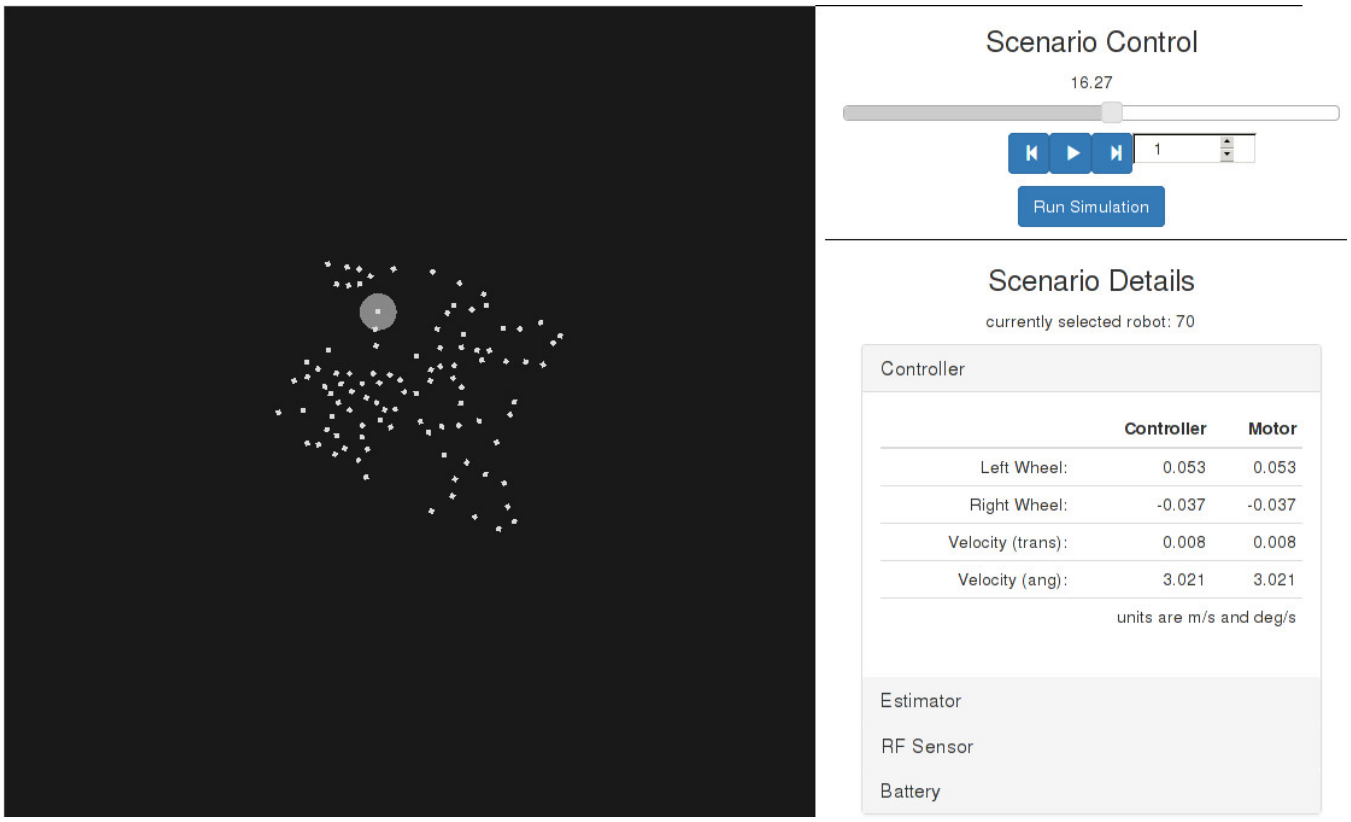


Fig. 3: A screenshot of the simulation web interface executing a rendezvous controller.

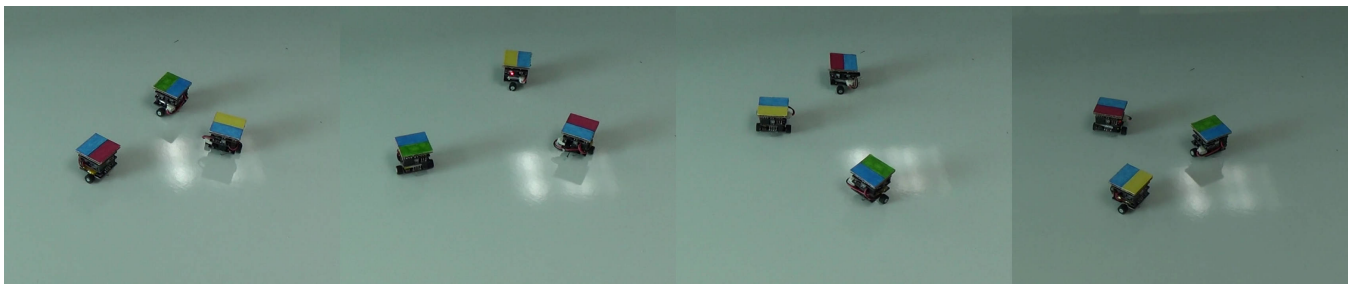


Fig. 4: Snapshots of the circular path following controller executed on a team of three robots.

with a focus on ease of use, ease of maintainability, and extensibility through a modular architecture.

Much like the GRITSBot itself, the Robotarium aims to provide intuitive interaction and high usability through remote access. Matlab and Python APIs allow users to remotely control the robots while a web-based simulator enables rapid prototyping of multi-agent algorithms including code transfer and execution on the robots. The Robotarium's main goal is to provide both a model and a physical instantiation of a remotely accessible multi-robot facility and as a result relieve researchers from having to set up and maintain their own costly testbeds. Through a remotely accessible swarm of low-cost miniature robots together with an almost maintenance-free testbed, cost will no longer be a limiting factor in multi-agent research.

ACKNOWLEDGEMENT

We would like to thank Dr. Jorge Cortés and his team from the University of California San Diego for the submission of the circular path following code shown in Section VI.

REFERENCES

[1] A.D. Ames, J.W. 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*, pages 6271–6278, Dec 2014.

[2] A. Arsie, K. Savla, and E. Frazzoli. Efficient routing algorithms for multiple vehicles with no explicit communications. *Automatic Control, IEEE Transactions on*, 54(10):2302–2317, 2009.

[3] G. Arslan, J.R. Marden, and J.S. Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[4] G.A. Casan, E. Cervera, A.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*, pages 6101–6106, 2015.

[5] L. Chaimowicz, B. Grocholsky, J.F. Keller, V. Kumar, and C.J. Taylor. Experiments in multirobot air-ground coordination. In *Robotics and Automation (ICRA), 2004 IEEE International Conference on*, volume 4, pages 4053–4058, April 2004.

[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: An overlay testbed for broad-coverage services. *SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[7] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243–255, 2004.

[8] D. Cruz, J. McClintock, B. Perteet, O.A.A. Orqueda, Y. Cao, and R. Fierro. Decentralized cooperative control - a multivehicle platform for research in networked embedded systems. *Control Systems, IEEE*, 27(3):58–78, June 2007.

[9] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. Sukhatme. Robomote: enabling mobility in sensor networks. In *Information Processing in Sensor Networks (IPSN), 2005 International Symposium on*, pages 404–409, 2005.

[10] P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: a miniaturized network testbed for mobile wireless research. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2731–2742, March 2005.

[11] Pradipta De, Ashish Raniwala, Rupa Krishnan, Krishna Tatavarthi, Jatan Modi, Nadeem Ahmed Syed, Srikant Sharma, and Tzi-cker Chiueh. Mint-m: an autonomous mobile wireless experimentation platform. In *Mobile Systems, Applications and Services, 2006 International Conference on*, pages 124–137, 2006.

[12] 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.

[13] M.I. El-Hawwary and M. Maggiore. Global path following for the unicycle and other results. In *American Control Conference, 2008*, pages 3500–3505, June 2008.

[14] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas. Cooperative air and ground surveillance. *Robotics Automation Magazine, IEEE*, 13(3):16–25, Sept 2006.

[15] A. Jimnez-Gonzlez, J. R. M. de Dios, and A. Ollero. Testbeds for ubiquitous robotics: A survey. *Robotics and Autonomous Systems*, 61(12):1487–1501, 2013.

[16] Z. Jin, S. Waydo, E.B. Wildanger, M. Lammers, H. Scholze, P. Foley, D. Held, and R.M. Murray. Mvwt-ii: the second generation caltech multi-vehicle wireless testbed. In *American Control Conference, 2004. Proceedings of the 2004*, volume 6, pages 5321–5326 vol.6, June 2004.

[17] D. Johnson, T. Stack, R. Fish, D.M. 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*, pages 1–12, 2006.

[18] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How. Coordination and control experiments on a multi-vehicle testbed. In *American Control Conference, 2004. Proceedings of*, volume 6, pages 5315–5320, June 2004.

[19] S. Kodagoda, A. Alempijevic, Shoudong Huang, M. de la Villefromoy, M. Diponio, and L. Cogar. Moving away from simulations: Innovative assessment of mechatronic subjects using remote laboratories. In *Information Technology Based Higher Education and Training (ITHET), 2013 International Conference on*, pages 1–5, Oct 2013.

[20] T.W. McLain and R.W. Beard. Unmanned air vehicle testbed for cooperative control experiments. In *American Control Conference, 2004. Proceedings of the 2004*, volume 6, pages 5327–5331, June 2004.

[21] N. Michael, J. Fink, S. G. Loizou, and V. Kumar. Architecture, abstractions, and algorithms for controlling large teams of robots: Experimental testbed and results. In *Robotics Research*, pages 409–419. Springer, 2011.

[22] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *Robotics Automation Magazine, IEEE*, 17(3):56–65, Sept 2010.

[23] Nathan Michael, J. Fink, and V. Kumar. Experimental testbed for large multirobot teams. *Robotics Automation Magazine, IEEE*, 15(1):53–61, 2008.

[24] J. Mirkovic and T. Benzel. Teaching cybersecurity with deterlab. *Security Privacy, IEEE*, 10(1):73–76, Jan 2012.

[25] S. Osentoski, G. Jay, C. Crick, B. Pitzer, C. DuHadway, and O.C. Jenkins. Robots as web services: Reproducible experimentation and application development using rosjs. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 6078–6083, May 2011.

[26] S. Osentoski, B. Pitzer, C. Crick, G. Jay, S. Dong, D. Grollman, H. B. Suay, and O. C. Jenkins. Remote robotic laboratories for learning from demonstration. *International Journal of Social Robotics*, 4:1–13, June 2012.

[27] L. E. Parker and A. Howard. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *International Journal of Robotics Research*, 25:431–447, 2006.

[28] K. Petersen, R. Nagpal, and J. Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. In *Robotics: Science and Systems (RSS), 2011 Conference on*, 2011.

[29] P. Petrovic and R. Balogh. Deployment of remotely-accessible robotics laboratory. *International Journal of Online Engineering (iJOE)*, 8(S2):31–35, 2012.

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

[31] B. Pitzer, S. Osentoski, G. Jay, C. Crick, and O.C. Jenkins. Pr2 remote lab: An environment for remote development and experimentation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3200–3205, 2012.

[32] 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 Conference, 2005 IEEE*, volume 3, pages 1664–1669, March 2005.

[33] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3293–3298, 2012.

[34] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal. Collective transport of complex objects by simple robots: theory and experiments. In *Autonomous Agents and Multiagent Systems (AAMAS), 2013 International Conference on*, pages 47–54, 2013.

[35] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, and G. Dullerud. Multivehicle systems control over networks. *IEEE Control Systems Magazine*, 26(3):56–69, 2006.

[36] M. Zhu and S. Martínez. Distributed coverage games for energy-aware mobile sensor networks. *SIAM Journal on Control and Optimization*, 51(1):1–27, 2013.