GTernal: A Robot Design for the Autonomous Operation of a Multi-Robot Research Testbed

Soobum Kim[1], Paden Davis[1], Nathan Yam[1], Samuel Coogan[1], and Sean Wilson[2]

[1] Georgia Institute of Technology, Atlanta GA 30332, USA,
{skim743, pdavis77, nyam3, sam.coogan}@gatech.edu,
[2] Georgia Tech Research Institute (GTRI), Atlanta, GA, 30318 USA,
Sean.Wilson@gtri.gatech.edu

Abstract. This paper presents GTernal, an open-source robot designed for multi-robot experimentation with a focus on reducing the overhead labor required to setup and conclude multi-robot experiments. This paper discusses the robot's hardware design choices and the supporting algorithms that together enable the automation of the time intensive initialization, clean up, and charging routines associated with multi-robot experimentation. The efficacy of the presented design and algorithms are validated by the continuous execution of 453 experiments over a three day period on the Robotarium testbed.

Keywords: multi-robot system, automation, collision avoidance

1 Introduction

Creating a multi-robot research testbed is in general a labor intensive task as it requires the development and integration of the hardware and software necessary to experimentally validate novel algorithms. Beyond this, the cost to instantiate these multi-robot testbeds, in terms of time and money, is also often prohibitive, leading many theoretical findings in the area of multi-robot research to be validated in simulations. To date, a number of different robot platforms with low cost designs and small footprints have been introduced in an effort to make multi-robot systems more accessible to researchers [1,2,3,4,5,6].

In practice, operating a multi-robot testbed can be as labor intensive as creating it. For example, to run a multi-robot experiment, a researcher typically needs to turn each robot on, charge and replace the batteries of each robot, update each robot with the code to execute the experiment, and place each robot into a desired initial configuration. Similarly, after completing an experiment, a researcher needs to go through a cleanup procedure including retrieving each robot, powering them down, and charge their batteries for a future use. These relatively mundane procedures can occupy a large amount of a researcher's time. This necessitates a robot design that facilitates automating the repetitive and labor intensive procedures associated with multi-robot experiments.

This paper presents GTernal (pictured in Fig. 1), a robot platform designed for multi-robot research, focusing on the design choices and supporting algorithms that enable the platform to autonomously perform the mundane and time-consuming initialization and cleanup routines surrounding multi-robot experimentation. An early prototype of the GTernal was briefly introduced in [7,8] as GRITSBot X. Although [7,8]

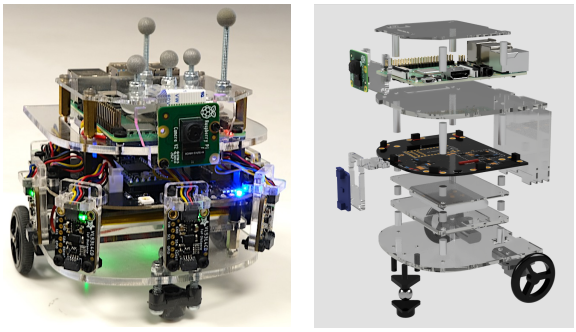


Fig. 1. The GTernal robot (left) and the exploded rendering of it (right).

discuss the design philosophies for the robot, they do not provide specific details on the robot's design, the algorithms that support its ability to automate experiment execution, nor validate its performance over long time scales, as the primary focus of these papers is on the Robotarium, a remotely accessible multi-robot research testbed. In the past 5 years since the initial deployment of GRITSBot X in the Robotarium, the robot design has gone through a series of design iterations culminating in the GTernal, whose design enables the autonomous execution of many multi-robot experiments over long periods of time with minimal human intervention.

The effectiveness of the GTernal's design and associated automation algorithms is validated through the continuous execution of experiments by a group of 10 GTernals on the Robotarium over a three day period. During this time, 453 experiments were autonomously conducted with only 6 human interventions required during the period.

This paper is organized as follows. In Section 2, GTernal is compared with other existing robot platforms. The robot design, including its processors, sensors, locomotion, and power system, is discussed in Section 3. In Section 4, the initialization, charging, and barrier deadlock resolution algorithms that are used with the GTernal platform to reduce the required human labor to instantiate and conclude multi-robot experiments are explained. The onboard collision avoidance feature of GTernal is introduced in Section 5, and the effectiveness of the overall robot design for multi-robot testbed automation is discussed in Section 6. Finally, Section 7 concludes the paper.

2 Existing Platforms

Over the past decade, various robotic platforms have been developed to validate multi-robot research in a laboratory setting. This section motivates the development and design of the GTernal robot by comparing its design and capabilities with these existing robotic platforms, listed in Table 1. To facilitate the comparison process, we categorize the robot platforms into two groups: 1. Miniature platforms (robots with a diameter of 5 cm or less) and 2. Non-miniature platforms.

Miniature robot platforms typically focus on enabling the deployment of large numbers of robots simultaneously in an indoor setting. To facilitate this, the robots are designed to be cost effective, have small footprints, and be relatively simple to assem-

Table 1. Comparison of Multi-Robot Platforms

Robot	Processors	Sensors	Automatic Charging	Cost ($)
GTernal	ARM CORTEX-A72, ARM CORTEX-M7	wheel encoders, power meter, accelerometer, gyroscope, magnetometer, time-of-flight, camera	Y	370
Kilobot [1]	ATmega328	infrared, visible light	N	14
GRITSBot [4]	ATmega328, ATmega168	infrared, accelerometer, gyroscope	Y	50
Zooid [3]	ARM CORTEX-M0	capacitive touch	N	50
mROBerTO 2.0 [5]	ARM CORTEX-M0	infrared, time-of-flight	N	120
e-Puck [9]	ARM CORTEX-M4	infrared, time-of-flight, accelerometer, gyroscope, magnetometer, microphone, camera	N	970
Khepera IV [10]	ARM CORTEX-A8	infrared, ultrasonic, accelerometer, gyroscope, microphone, camera	N	3100
3pi+ [11]	ARM CORTEX-M0+ or ATmega32U4	wheel encoders, infrared, accelerometer, gyroscope, magnetometer, bump	N	150-172
r-one [2]	ARM CORTEX-M3	wheel encoders, accelerometer, gyroscope, light	N	220
Pheeno [6]	ARM CORTEX-A7, ATmega328P	wheel encoders, infrared, accelerometer, magnetometer, camera	N	270
TurtleBot [12]	ARM CORTEX-A72, ARM CORTEX-M7	wheel encoders(4), infrared(4), LiDAR, accelerometer, gyroscope, magnetometer, camera	N	660-1900
SMARTmBOT [13]	ARM CORTEX-A72	infrared, time-of-flight, camera	N	210
JetBot [14]	ARM CORTEX-A57	camera	N	250

ble [1,4,3,5]. Although miniature robots enable researchers to deploy large amounts robots simultaneously in a small indoor space, their small-sized actuators are not robust enough for continuous long term autonomous operations. For instance, the vibration based locomotion of Kilobots makes it difficult for them to move precisely for an extended period of time [1]. Also, miniature robots in general require a flat and clean surface to operate reliably. The small, less powerful actuators of the miniature robots have their motion biased when the testbed's operating surface is not level or has imperfections from natural settling. Beyond this, small motors, like those of the GRITSBots [4], are prone to picking up dusts and small particles on the surface of the testbed, which leads to faster actuator failures. The requirement for testbed surfaces to be clean and flat adds to the effort required for researchers to run multi-robot experiments. More importantly, the small actuators that drive these robots make automating a potential charging process or the process of getting robots on and off the testbed less reliable.

In contrast to miniature robot platforms, non-miniature robot platforms typically have reliable, robust actuation, a wide variety of sensing, powerful onboard computation, and the ability to expand their sensor suite. While not as many can be deployed in the same space as their miniature counterparts, these non-miniature robot platforms have the potential to automate charging, initialization, and cleanup procedures for multi-robot testbeds. The e-puck [9] and e-puck2 [15] are mobile robot platform designed for university level education. These robots are equipped with a wide variety of sensors with different types of extensions available to address specific needs. Khepera IV [10] is a commercially available robot designed for indoor lab applications. Similar to e-puck, Khepera IV is equipped with a variety of sensors and offers several extensions for a range of applications, from research to education. The 3pi+ [11] is a commercially available robot from Pololu Robotics that fills a similar role as the e-puck and Khepera

IV at a cheaper price range. The r-one [2] is an open source robot platform designed for multi-robot research and education. The overall design of the robot is similar to that of 3pi+ robot, but r-one has additional capabilities such as a radio for global control, an infrared beacon for localization, and direct inter-robot communication.

The aforementioned non-miniature robot platforms are relatively older platforms, initially introduced about a decade ago, making reproduction and augmentation of their original open source designs difficult since some of their electrical and mechanical components are no longer available. Although e-puck and Khepera have more recent versions of their original design commercially available, the commercial price of an e-puck2 ($\sim$ \$950[3]) and a Khepera ($\sim$ \$3100 [10]) can make it prohibitively expensive to use the platforms for multi-robot research. The e-puck2 does have extensions available for autonomous charging, but this raises their price further to between $\sim$ \$1470 and $\sim$ \$1525 per robot[3]. The 3pi+ uses AAA batteries, which makes maintaining the battery level of a fleet of robots tedious.

More recently introduced non-miniature robot platforms feature powerful single board computers and open source designs that are current enough to potentially reproduce. Pheeno [6] is a robot designed for research and education. The robot has a modular design where the gripper module or other custom modules can be attached to the core module for additional functionality. TurtleBot [12] is a commercially available design that can support sensor expansion through its single board computer. SMARTmBOT [13] is built for robotics research and education using 3 layers of PCBs and 3D printed parts, providing its adopters the ability to customize it. JetBot [14] is a robot that leverages the NVIDIA Jetson Nano [16], 3D printed parts and commercially available components for its construction.

Although the modern non-miniature platforms offer powerful onboard computation capabilities in addition to more reliable actuators, rich sensor suites, and potential expandability, their current designs are not ideal for automating the mundane processes associated with experimentation on a multi-robot testbed. The TurtleBot is larger than the GTernal with the Burger model having a $\sim 60\%$ larger footprint and the Waffle model having a $\sim 90\%$ larger footprint, which makes it difficult to use many of them for an indoor experiments. In addition, its starting commercial price of $\sim$ \$650 can be cost prohibitive to create a multi-robot system using the platform. Pheeno, SMARTmBOT, and JetBot are more compact and less expensive than TurtleBot filling similar roles. However, all of these platforms, including the Turtlebot, do not natively support automatic charging and are not designed to limit power draw from sensors, actuators, and computation devices when charging which are key features to automate the labor intensive battery maintenance of a multi-robot testbed.

3 Robot Design

The motivation for GTernal design is to create a robot platform suitable for autonomously executing indoor multi-robot experiments over an extended period of time without the need for human intervention for maintenance, setup, or cleanup. This can be achieved

[3] http://www.gctronic.com/shop.php : Accessed April 2, 2024

with a design that provides; 1) reliable actuation and sensor suites to safely navigate the testbed repeatedly while maintaining a small footprint, 2) onboard power sensing and regulation capabilities with the ability for the robot to charge itself to autonomously maintain its battery levels, 3) Over-the-air (OTA) interfacing to reduce programming time and enable global safety routines from a central computer (i.e. stop all robots if a collision occurs) and 4) be repairable and upgradable so the design does not become obsolete from wear and tear or computation advances. Figure 2 shows a high level block diagram of the GTernal design that support these features along with its connections to a potential central computer and tracking system present in most multi-robot testbeds. The rest of the section details the design choices of the GTernal robot.

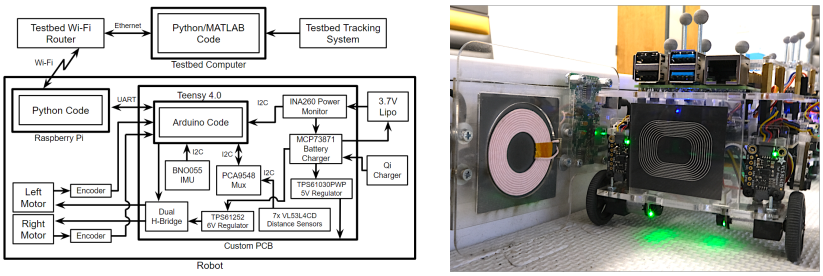


Fig. 2. Block diagram of a GTernal connected to general multi-robot testbed infrastructure (left) and the wireless charging receiver of GTernal and a charging station in the Robotarium (right).

3.1 Compact and Reliable Robot Design

The GTernal, pictured in Figure 1, is a differential drive platform with a small footprint 11 cm $\times$ 9.5 cm $\times$ 9.5 cm (width, length, height). The robot is constructed with a custom laser-cut acrylic chassis, a custom PCB, and commercially available off the shelf components. The design files and guide associated with constructing and running the robot can be found within a publicly available GitHub repository [17].

The most fundamental requirement for a mobile robot to successfully operate for a long time is reliable actuation. Typically, smaller actuators are less powerful and more fragile, but the size of the actuator has a significant impact on the robot footprint. The two motors chosen to drive GTernal are 6V DC micro metal geared motors with 100:1 gear ratio[4]. These motors enable the robot to maintain a small footprint with dimensions of 14.5 mm $\times$ 12 mm $\times$ 40.5 mm while remaining powerful and agile. The motors are coupled with wheels with a diameter of 32 mm, which yield a maximum linear speed of the robot of approximately 26 cm/s. Additionally, these motors come with an attached quadrature encoder with 0.25° resolution. The built-in encoder allows precise feedback control of the motor and enables the robot to detect motor failure.

GTernal is equipped with a fundamental sensor suite consisting of four types of onboard sensors: a power monitor, an inertial measurement unit (IMU), a Raspberry Pi Camera Module 2, and 7 time-of-flight (ToF) range sensors. The onboard power monitoring chip is INA260 capable of measuring the voltage, current, and power with

[4] Part number SKU:FIT0483

precision of 1.25 mV, 1.25 mA, and 10 mW, respectively. The IMU used in GTernal is BNO055, which can produce fused estimates of the robot's orientation as well as raw sensing data. The ToF distance sensors onboard GTernal are VL53L0X, each of which have a field of view (FOV) of 25 degrees and can measure up to 1.2 m with $4-7\%$ accuracy in default mode. These ToF sensors also have the ability to be turned off when not in use to conserve power.

In addition to reliable actuators and essential sensor suite, GTernal is equipped with a Raspberry Pi [18] single board computer to execute high-level algorithms, and a Teensy 4.0 [19] micro-controller to perform low-level hardware control. The Raspberry Pi was chosen over other single board computers because of its large, active user base to help with development, ability to modify power consumption, and identical form factor between versions. The GTernal can be equipped with a Raspberry Pi Zero, 3, or 4, depending on the user's requirements for onboard computational power versus power consumption. The microprocessor (Raspberry Pi) and micro-controller (Teensy) are connected and communicate over a UART serial line, which allows a central control station to interface with a fleet of GTernal robots by flashing them with new firmware, sending them global commands, and receiving sensor data from the robots.

3.2 Wireless Charging and Power Management

Operating mobile robots autonomously for an extended period of time requires an automatic charging system to restore or maintain robots' battery levels between experiments. Therefore, GTernal is equipped with a battery charge management chip, MCP73871, capable of delivering the maximum of 1 A charging current coupled with a Qi wireless charging receiver supporting the maximum charging current of 1A from any standard Qi wireless charging transmitter. The wireless charging module of GTernal is supported in between two 1/16" acrylic plates in the rear of the robot, as seen in Figure 2. The rear of the robot chassis is designed to be flat to minimize the distance between the wireless emitter and receiver. This choice facilitate the charging process managed by an automated procedure that is discussed in Section 4. Similar to motor selection, choosing the battery for the GTernal involves balancing trade-offs: it must have sufficient capacity for extended operation yet remain compact and efficient for wireless charging. Accordingly, GTernal is powered by a 3.7 V 2500 mAh lithium-ion-polymer (LiPo) battery. With this power management setup, the GTernal equipped with a Raspberry Pi 4 and full sensor suite draws approximately 400 mA in idle and 600 mA during typical use. This results in the robot being able to nominally operate for about 3 hours without recharging and about 2 hours continuously while maintaining the ability to recharge while on.

3.3 Repairability, Customizability, and Upgadeability

Although GTernal is designed to be durable, some of its components may need to be replaced after continuous operation. For instance, the motors on the robot fail after running for a long period of time due to wear and tear. Also, the acrylic plates or time-of-flight sensors may be damaged due to collisions caused by hardware/software failures. In order to facilitate the potential robot repair process, GTernal features easily

repairable design. All parts of GTernal are installed on the acrylic plate chassis of the robot through screws and standoffs, and the electrical connections between different components are made through Japan Solderless Terminal (JST) connectors. This allows users to easily replace any damaged components of the robot.

Beyond repairability, GTernal's design also makes the platform flexible in terms of customizability and upgradability. Although GTernal is mainly designed for Raspberry Pi 3s and 4s, it provides users with a degree of freedom on the processor and sensor selections. For instance, users who do not require the heavy utilization of onboard data processing or computations can build a version of GTernals using lower cost processors (Raspberry Pi Zero W or Zero 2 W). Moreover, the users can decide not to install ToF sensors which reduces the price of each platform by $\sim \$100$. If needed, the Raspberry Pis of the robots can be replaced with more powerful ones, or the ToF sensors can be installed on the robots at a later date.

4 Autonomous Initialization and Charging Algorithms

In this section, the algorithms that automate the mundane charging, initialization, and clean up routines associated with multi-robot experimentation with GTernals are presented. The first key algorithm for safe automation of these tasks is collision avoidance as collisions between robots require human intervention to remedy, and they can even damage the robots. In order to prevent inter-robot collisions during automated initialization and charging routines, GTernals utilize control barrier functions (CBFs) [20,7] to ensure that the distance between any pair of robots does not become lower than a specified threshold. However, these CBFs can often lead the robots to a state known as deadlock [21] where the robots do not further progress towards its main objective to avoid collisions with other robots. Since collision avoidance between GTernals needs to be enforced during initialization and charging processes, a deadlock resolution strategy is needed to allow their long term operation.

Algorithm 1: check_deadlock

Input: $x_i, x_i^{\text{old}}, g_i, t, t_i^{\text{old}}$
Output: $deadlock\_flag$
Initialize $deadlock\_flag = false$
if $\|x_i - g_i\| > \epsilon$ **then**
  if $\|x_i - x_i^{old}\| > \gamma$ **then**
    $x_i^{\text{old}} = x_i$;
    $t_i^{\text{old}} = t$;
  end
  if $t - t_i^{old} > \delta$ **then**
    $deadlock\_flag = true$;
  end
end
Return: $deadlock\_flag$

Algorithm 2: initialize_robots

Input: $x, x^{\text{init}}, \mathcal{N}$
Initialize $g = x^{\text{init}}, x^{\text{old}} = \mathbf{0}, t = 0, t^{\text{old}} = \mathbf{0}, t^{\text{deadlock}} = \mathbf{0}$;
Start counting t;
while $t < t_{\text{forward}}$ **do**
  for $i \in \mathcal{N}$ **do**
    $u_i = (v_{\text{max}}, 0)$;
  end
end
while $\|x_i - x_i^{init}\| > \epsilon$, *for any* $i \in \mathcal{N}$ **do**
  for $i \in \mathcal{N}$ **do**
    if *check_deadlock*$(x_i, x_i^{old}, g_i, t, t_i^{old})$ **then**
      if $t - t_i^{deadlock} > \delta$ **then**
        $g_i =$ uniformDistribution(testbedDimension);
        $t_i^{\text{deadlock}} = t$;
      end
    else
      $g_i = x_i^{\text{init}}$;
    end
    $u_i =$ poseControl(g_i, x_i);
  end
  $u =$ barrier(u, x);
end

To this end, Algorithm 1 identifies deadlocks by checking if a robot is stuck at a position and not moving towards its goal point. The algorithm is used to temporarily change the goal point of a robot to a random point drawn from a uniform distribution within the testbed during initialization and charging processes. This small addition to the standard CBF algorithm results in safe robot motion that always reaches the desired goal pose given the goal poses are feasible. In the algorithm, $x_i, x_i^{\text{old}}, g_i \in \mathbb{R}^3$ are the current, previous, and goal poses of robot i, respectively, where each pose is a vector containing an X-coordinate, Y-coordinate, and heading; $t, t_i^{\text{old}}, t_i^{\text{deadlock}}, \epsilon, \gamma, \delta > 0$ are the current time, previous time, previous deadlock time, tolerance for position control to a goal pose, deadlock tolerance distance, and timeout for deadlock detection.

Before executing a multi-robot experiment, robots must be initialized to specified or random positions within the testbed. Algorithm 2 is used to bring robots to a starting configuration from their charging locations. Assuming all robots are at their charging stations, and given the set of indices of robots $\mathcal{N}$ needed for the experiment, the initialization algorithm drives the robots onto the testbed by making them move forward at their maximum linear velocity v_{max} for t_{forward} seconds. Then, the algorithm drives the robots to the initial poses required for the experiment, $x^{init} = [{x_1^{\text{init}}}^\top, \cdots, {x_{|\mathcal{N}|}^{\text{init}}}^\top]^\top \in \mathbb{R}^{3|N|}$, using a proportional pose controller that is augmented for safety using CBFs with the deadlock detection algorithm to resolve deadlocks.

To alleviate the labor involved in removing and charging robots after an experiment, Algorithm 3 autonomously navigates the robots to the wireless charging stations and precisely positions them on the emitters to initiate the Qi 'handshake', thereby begin-

Algorithm 3: automatic_charging

Input: $x, c, c^{\text{approach}}$
Initialize $g = c^{\text{approach}}, c' = c, x^{\text{old}} = \mathbf{0}, t = \mathbf{0}, t^{\text{old}} = \mathbf{0}$;
Start counting t;
while *not all robots being charged* **do**
  for $i \in \mathcal{N}$ *not being charged* **do**
    State 1 (drive to charging approach pose);
      if *check_deadlock*$(x_i, x_i^{old}, g_i, t_i, t_i^{old})$ **then**
        if $t_i - t_i^{deadlock} > \delta$ **then**
          $g_i = \text{uniformDistribution(testbedDimension)};$
          $t_i^{\text{deadlock}} = t_i;$
        end
      else
        $g_i = c_i^{\text{approach}};$
      end
      $u_i = \text{poseControl}(g_i, x_i);$
      $u_i = \text{barrier}(u, x);$
    State 2 (drive towards charger);
      while $\|c_i' - x_i\| > \epsilon$ **do**
        $u_i = \text{poseControl}(c_i', x_i);$
      end
      if *not being charged* **then**
        go to State 3;
      end
    State 3 (drive back to approach pose);
      $c_i' = \text{normalDistribution}(c_i, \sigma);$
      while $\|c_i^{\text{approach}} - x_i\| > \epsilon$ **do**
        $u_i = \text{poseControl}(c_i^{\text{approach}}, x_i)$
      end
    go to State 2;
  end
end

ning the charging process. The algorithm is a state machine comprised of three states. Similar to the initialization phase, the first state safely drives each robot that is not currently being charged to a designated point near the charging stations. In the second state, each robot is driven to the recorded location of its charging station without CBFs to make contact with the charging coil. Considering a typical multi-robot testbed design where multiple charging coils are positioned closely to facilitate side-by-side charging, the docking motion proceeds without safety guarantees. This is to prevent CBFs from influencing the motions of robots, ensuring they accurately reach their designated emitter coil locations.

Wireless charging requires a pair of transmitter and receiver to align with minimal error. In practice, a robot can often fail to connect at this location due to tracking errors and actuator errors. Therefore, in order to make the autonomous wireless charging sys-

tem work reliably, Algorithm 3 repeats the docking process if its charging attempt fails. However, instead of using the fixed charger location, the algorithm chooses slightly different locations for the charger given by a Gaussian distribution around the predefined position of the charging station.

5 Onboard Collision Avoidance

Collision avoidance is an important factor to consider when operating a multi-robot testbed as collisions can damage robots. In the Robotarium, with the predecessors of GTernal, GRITSBot [4] and GRITSBot X [7], inter-robot collision avoidance is implemented using control barrier functions [7]. However, the implementation relies on the proper functioning of a central tracking system and communication networks between a central computer and the robots on the testbed. The predecessor robots of the GTernal do not offer a standalone onboard collision avoidance feature. This is due to the lack of precision and consistency in their onboard infrared (IR) distance sensor measurements and limited computational capabilities. Although the collision avoidance achieved through a central tracking system effectively prevents collisions between robots, an onboard collision avoidance feature on a robot can add a layer of protection if the central tracking system fails, which can be beneficial in various multi-robot studies. For instance, one can test a decentralized algorithm using real robots without explicitly considering collision avoidance even in the absence of a tracking system. Also, arbitrary obstacles can be introduced into the environment to test the robustness or resilience of an algorithm under the presence of unknown obstacles.

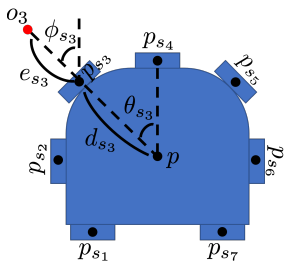


Fig. 3. Top view of a GTernal in the robot coordinate frame where the origin is located at the center of the robot, p. The position of the i^{th} ToF sensor is p_{s_i} which is distance d_{s_i} away from the origin. The angle of the i^{th} sensor with respect to p is θ_{s_i}, and ϕ_{s_i} is its mount angle.

GTernal is equipped with an onboard collision avoidance feature fully utilizing its 7 onboard Time-of-Flight (ToF) sensors. Consider the robot coordinate frame described in Fig. 3 where the origin is at the center of the robot, i.e., $p = [0, 0]^\top$, and the yaw angle origin is with respect to the 4^{th} sensor, i.e., $\theta_{s_4} = 0$. The position of each sensor i can be expressed as $p_{s_i} = d_{s_i}[cos(\theta_{s_i}), sin(\theta_{s_i})]^\top \in \mathbb{R}^2$ where $d_{s_i} > 0$ is the distance between the center of the robot and the i^{th} sensor, and θ_{s_i} is the angle of the sensor with respect to p. The sensor angle θ_{s_i} follows the convention of the unicycle dynamics where the counter-clockwise direction is positive.

With the sensor positions defined in the robot coordinate frame, a control barrier function [20] that prevents collisions between each sensor and its detected obstacle can be defined as

$$h_{s_i}(p, o_i) = \|p_{s_i} - o_i\|^2 - d_{\text{safe}}^2 \geq 0 \tag{1}$$

where $o_i \in \mathbb{R}^2$ is the position of the obstacle detected by the i^{th} sensor, and d_{safe} is a desired safety distance to obstacles. Since the i^{th} sensor has the mount angle of ϕ_{s_i}, the position of the obstacle based on the sensor measurement is given as $o_i = p_{s_i} + e_i[cos(\phi_{s_i}), sin(\phi_{s_i})]^\top$ where $e_i > 0$ is the distance measurement provided by the sensor. In order for (1) to hold in forward time, which implies that the robot does not collide with the detected obstacle, the time derivative of the barrier function needs to satisfy that

$$\dot{h}_{s_i}(p, o_i) = 2(p_{s_i} - o_i)^\top(\dot{p}_{s_i} - \dot{o}_i) \geq -\alpha(h_{s_i}(p, o_i))$$

for all p_{s_i} where α is a class $\mathcal{K}$ function. Since GTernal is a rigid body, it holds that $\dot{p}_{s_i} = \dot{p}$. Accordingly, assuming single integrator dynamics for the robot, i.e., $\dot{p} = u$, similar to what has been done in [22], the controller that prevents collisions between the robot and its detected obstacles is given by the following quadratic program (QP) controller

$$\begin{aligned} u^* &= \arg\min_u \|u - \hat{u}\|^2 \\ \text{s.t. } &- 2e_i[cos(\phi_{s_i}), sin(\phi_{s_i})]u \geq -\alpha(h_{s_i}(p, o_i)),\ \forall i \in \{1, \cdots, 7\} \end{aligned} \tag{2}$$

where $\hat{u}$ is a nominal control input for the robot. Note that the QP minimally modifies the nominal control input of the robot in the sense of l_2 norm to avoid collisions with its detected obstacles. Here, the velocity of each obstacle, $\dot{o}_i$, is assumed to be 0, which seems to imply that the collision avoidance is ensured only with respect to static obstacles. However, collision avoidance between multiple GTernals can also be established with a proper constant multiplied to the class $\mathcal{K}$ function as every GTernal is driven by the controller in (2), which effectively creates the decentralized implementation of the multi-robot collision avoidance barrier certificates discussed in [21].

In fact, GTernal is a differential drive robot which is modeled using unicycle dynamics. Therefore, the single integrator control, u^*, given by the QP controller in (2) needs to be converted to unicycle dynamics. Using near-identity diffeomorphism [23], the single integrator control u^* can be converted to linear and angular velocities of a unicycle as

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} u^*$$

where $l > 0$ is a small distance from the center of the robot to the virtual point projected in front of the center point, and θ is the yaw angle of the robot which is set to 0 as the control input is computed in the robot coordinate frame.

6 Experiments

In order to validate the efficacy of GTernal design and associated algorithms for multi-robot testbed automation, two experiments are discussed. First, we present experiments

that demonstrate the effectiveness of the onboard collision avoidance algorithm, showcasing GTernal's ability to operate safely and independently without a central control authority. Then, an experiment demonstrating the ability of the GTernal platform to autonomously execute large numbers of different experiments over long periods of time is discussed.

6.1 Onboard Collision Avoidance

In this set of experiments, the onboard collision avoidance algorithm of GTernal is demonstrated to be successful in avoiding collisions with unknown obstructions in the testbed as well as inter-robot collisions. In the first experiment, 4 robots were tasked with navigating from one end of the $3.2m \times 2m$ testbed to the other without colliding with unknown obstacles in their way, as shown in Fig. 4. The 4 robots continuously received control inputs from the testbed computer driving them straight from their initial positions to their goal positions across the testbed. The nominal control for each robot i was calculated using a proportional controller, $\hat{u}_i = p_{i,\text{goal}} - p_i$, where $p_{i,\text{goal}}$ is the stationary goal position of the robot, and p_i is the robot's current position obtained through Vicon tracking system. In the arena, 3 obstacles that cannot be detected by the Vicon system were placed so that the robots collide with them if they directly execute their received control inputs from the central computer. However, the robots' onboard collision avoidance algorithm allows them to reach their goal positions without colliding with the obstacles as described in Fig. 4.

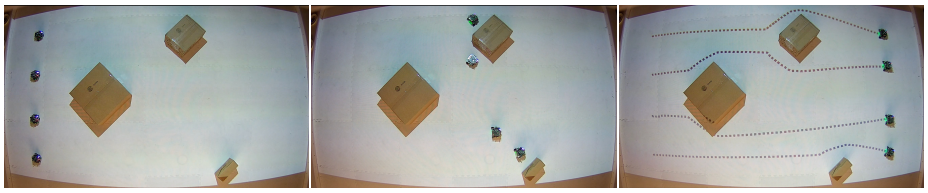


Fig. 4. Four GTernal robots successfully navigating to their goal positions without colliding with unknown obstacles in the arena utilizing their onboard collision avoidance algorithm.

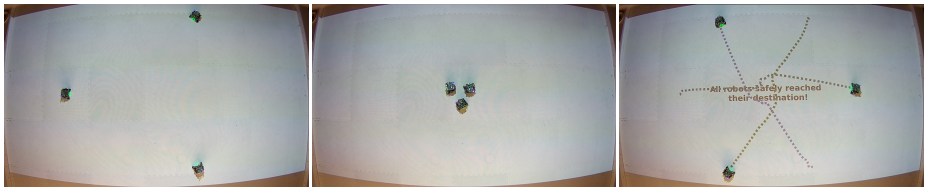


Fig. 5. Three GTernal robots successfully navigating to their goal positions without colliding with each other utilizing their onboard collision avoidance algorithm.

In the second experiment, inter-robot collision avoidance is demonstrated using 3 GTernals as shown in Fig. 5. Initially, the robots are positioned on a circle equally spaced from each other. During the experiment, each robot continuously received its nominal control input from the testbed computer that tries to drive it to the opposite side of the circle in a straight line. As can be seen in Fig. 5, the robots successfully reach their destinations without colliding with each other.

In these experiments, deadlock situations where all robots stop and not progress towards their objectives to avoid collisions did not occur. However, it is worth noting that the onboard collision avoidance algorithm does not guarantee deadlock-free operations as the purpose of this algorithm is to prevent each robot from collisions by minimally modifying the original control when the central avoidance algorithms fail or unknown objects are present on the testbed.

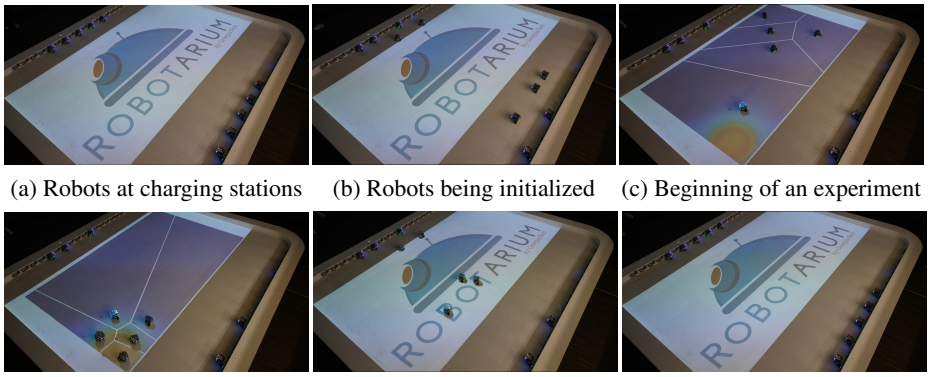
6.2 Testbed Automation

To validate the ability of the presented robot design and associated algorithms to enable the autonomous execution of many multi-robot experiments over long periods of time with minimal human intervention, 10 GTernals with full sensor kits and Raspberry Pi 4s (the highest power draw version) were deployed in the Robotarium to continuously operate for three days. GTernals were connected to the central testbed computer that runs Python or MATLAB scripts. The robots receive linear and angular velocity commands produced by the scripts running on the testbed computer, and the robots send their battery voltage measurements back to the computer along with any other user experiment data. The localization of the robots was done using Vicon tracking system.

During their three day deployment, the robots executed unknown experiments submitted by random Robotarium users as well as two default experiments involving 5 robots if the experiment queue of remote user submissions was empty. For each experiment, the robots autonomously drove themselves onto the testbed using Algorithm 2 and used Algorithm 3 to drive robots off the testbed and recharge the batteries of the robots. This constant experimentation execution tested the effectiveness of the proposed GTernal robot running the automated initialization and charging routines. The only times experiments would not be executed continuously is when the number of robots with a suitable voltage (3.7 V while charging) is less than the required number of robots for the experiment. In this low voltage case, the testbed would allow the robots to charge for 5 minutes before checking voltages again to run the experiment.

The first default experiment used in between user submitted experiments is coverage control [24] where the robots aim to cover a Gaussian distribution with a mean randomly generated within the testbed. The second default experiment used in between user submitted experiments is swapping positions on a circle safely using control barrier functions (CBFs) [7] for collision avoidance. These experiments were chosen due to the non-predetermined nature of the trajectories of the robots in these experiment which randomizes the power consumption and final positions of the robots.

The battery voltage level of 10 GTernals were monitored through the robots' INA260 power monitoring chip during the 3 days of experiment. Figure 7 shows the evolution of the battery voltage level of all robots. During the experiment, work was being done on the building housing the testbed causing periodic Vicon errors. These Vicon tracking errors caused 6 charging failures that required human intervention to alleviate over the three day period, shown as annotations in Figure 7. It is worth noting that the charging failure, which occurred around 30 hours of operation, was not addressed for three hours, leading to the complete discharge of a robot. This allowed other robots to recharge as the autonomous experiment execution was halted during this time due to the charging error. Outside of this event the robots remained operational while executing 453 total



(a) Robots at charging stations (b) Robots being initialized (c) Beginning of an experiment

(d) End of an experiment (e) Robots returning to chargers (f) Robots charging

Fig. 6. Autonomous initialization, charging, and execution of an experiment where 5 GTernals executing the coverage control algorithm with a Gaussian density function.

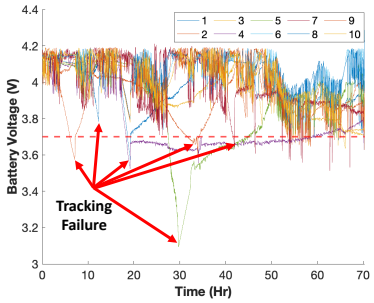


Fig. 7. Battery voltages of 10 GTernals during their continuous experiment execution over 3 days.

experiments consisting of 211 Robotarium user experiments and 242 default experiments.

7 CONCLUSIONS

Operating a multi-robot testbed is extremely labor intensive. This paper presents the GTernal robot and necessary algorithms to automate the tasks of initializing robots onto a tesbed, removing them from a testbed, and maintaining their charge. The robot design and algorithms are validated through continuous experiment execution over three days resulting in 453 experiments run with only 6 failures requiring human intervention.

ACKNOWLEDGMENT

The authors would like to thank Sixing Chen, Chih-Chun Huang, and Jacob Skinner for their contributions to the development and testing of GTernal.

References

1. M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3293–3298.
2. J. McLurkin, A. J. Lynch, S. Rixner, T. W. Barr, A. Chou, K. Foster, and S. Bilstein, *A Low-Cost Multi-robot System for Research, Teaching, and Outreach*. Springer Berlin Heidelberg, 2013, pp. 597–609.
3. M. Le Goc, L. H. Kim, A. Parsaei, J.-D. Fekete, P. Dragicevic, and S. Follmer, "Zooids: Building blocks for swarm user interfaces," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 97–109.
4. D. Pickem, M. Lee, and M. Egerstedt, "The gritsbot in its natural habitat - a multi-robot testbed," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4062–4067.
5. K. Eshaghi, Y. Li, Z. Kashino, G. Nejat, and B. Benhabib, "mroberto 2.0 – an autonomous millirobot with enhanced locomotion for swarm robotics," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 962–969, 2020.
6. S. Wilson, R. Gameros, M. Sheely, M. Lin, K. Dover, R. Gevorkyan, M. Haberland, A. Bertozzi, and S. Berman, "Pheeno, a versatile swarm robotic research and education platform," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 884–891, 2016.
7. S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.
8. S. Wilson, P. Glotfelter, S. Mayya, G. Notomista, Y. Emam, X. Cai, and M. Egerstedt, "The robotarium: Automation of a remotely accessible, multi-robot testbed," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2922–2929, 2021.
9. F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. M. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:16705005
10. K-Team. Khepera iv. [Online]. Available: https://www.k-team.com/khepera-iv
11. Pololu. 3pi. [Online]. Available: https://www.pololu.com/category/76/3pi-robots-and-accessories
12. O. S. R. Foundation. Turtlebot. [Online]. Available: https://www.turtlebot.com
13. W. Jo, J. Kim, R. Wang, J. Pan, R. K. Senthilkumaran, and B.-C. Min, "Smartmbot: A ros2-based low-cost and open-source mobile robot platform," 2022.
14. Jetbot. [Online]. Available: https://jetbot.org/master/index.html
15. e-puck 2 documentation. [Online]. Available: http://www.gctronic.com/doc/index.php/e-puck2
16. NVidia. Jetson nano. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit
17. Robotarium Organization. GTernal GitHub Repository. [Online]. Available: https://github.com/robotarium/GTernal
18. Raspberry pi. [Online]. Available: https://www.raspberrypi.com
19. PJRC. Teensy 4.0. [Online]. Available: https://www.pjrc.com/store/teensy40.html
20. A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

21. L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.
22. A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
23. R. Olfati-Saber, "Near-identity diffeomorphisms and exponential /spl epsi/-tracking and /spl epsi/-stabilization of first-order nonholonomic se(2) vehicles," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 6, 2002, pp. 4690–4695 vol.6.
24. J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.