Louisiana State University
LSU Digital Commons

LSU Master's Theses | Graduate School

June 2019

A Low-Cost Experimental Testbed for Multi-Agent System Coordination Control

Victor Fernandez-Kim
Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses

 Part of the Controls and Control Theory Commons, Electrical and Electronics Commons, Electro-Mechanical Systems Commons, and the Robotics Commons

Recommended Citation

Fernandez-Kim, Victor, "A Low-Cost Experimental Testbed for Multi-Agent System Coordination Control" (2019). *LSU Master's Theses*. 4949.
https://digitalcommons.lsu.edu/gradschool_theses/4949

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.

A LOW-COST EXPERIMENTAL TESTBED FOR MULTI-AGENT SYSTEM COORDINATION CONTROL

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

in

The Department of Mechanical Engineering

by
Victor Fernández-Kim
B.S. Mechanical Engineering, Louisiana State University, 2017
August 2019

Acknowledgments

I would first like to thank Dr. Marcio de Queiroz for his guidance and support throughout my undergraduate and graduate studies. His excellence as a professor and researcher has provided the foundation for me to become a more effective engineer, researcher, and communicator. I would like to thank Dr. Hunter B. Gilbert for his selfless support before and after I entered the graduate program at LSU. I would also like to thank Dr. Supratik Mukhopadhyay for agreeing to serve on my committee.

I thank the members of the Louisiana Space Consortium Group, Dr. T. Gregory Guzik, Colleen Fava, and Doug Granger, for sponsoring this research and their continued support throughout my studies at LSU. This group has provided countless opportunities and experiences for professional development that has been vital to my academic career.

I would also like to thank my research group members, Tairan Liu and Milad Khaledyan, for assistance in clarifying theoretical background and experimental results. Finally I give my deepest gratitude to my family and friends, whose constant support has allowed for the completion of this work.

Table of Contents

List of Tables

List of Figures

Abstract

A multi-agent system can be defined as a coordinated network of mobile, physical agents that execute complex tasks beyond their individual capabilities. Observations of biological multi-agent systems in nature reveal that these "super-organisms" accomplish large scale tasks by leveraging the inherent advantages of a coordinated group. With this in mind, such systems have the potential to positively impact a wide variety of engineering applications (e.g. surveillance, self-driving cars, and mobile sensor networks). The current state of research in the area of multi-agent systems is quickly evolving from the theoretical development of coordination control algorithms and their computer simulations to experimental validations on proof-of-concept testbeds using small-scale mobile robotic platforms. An in-house testbed would allow for rapid prototyping and validation of control algorithms, and potentially lead to new research directions spawned by experimentally-observed issues. To this end, a custom experimental testbed, TIGER Square, has been designed, developed, built, and tested at Louisiana State University.

In this work, the completed design and test results for a centralized testbed is presented. That is, the individual robots follow an overarching control entity and are reliant on a global structure, such as a central processing computer. As part of the validation process, a series of formation control experiments were executed to assess the performance of the testbed. In order to eliminate single-point failures, a multi-agent system must be fully decentralized or distributed. This means that the responsibilities of processing, localization, and communication are distributed to each agent. Therefore, this work concludes with the introduction of a prototype localization module that will be integrated into the existing centralized testbed. This initial step allows for the future decentralization of TIGER Square and opens the path to achieve a fully capable multi-agent system testbed.

Chapter 1
Introduction

1.1. Motivation

A multi-agent system can be defined as a coordinated network of mobile, physical agents that execute complex tasks beyond their individual capabilities. In nature, a wide range of biological systems fall under this definition (e.g. insect swarms, school of fish, flock of birds). This collective behavior results in a biological "super-organism" that accomplishes large-scale tasks such as building structures or gathering resources. These observations of biological systems motivate the efforts to advance the behavior of unmanned robotic platforms to operate in coordinated groups autonomously or under human supervision. In this way, such systems have the potential to impact a variety of military, civilian, scientific, and commercial applications that involve some form of situational awareness. Examples include patrolling, monitoring, surveying, scouting, and element tracking over large geographical areas with unmanned robotic vehicles or mobile sensor networks. More specifically, a group of autonomous (ground, underwater, water surface, or air) vehicles could be deployed in disaster areas without putting first-responders in harm's way (e.g., Hurricane Katrina in 2005, BP oil spill in the Gulf of Mexico, and the Fukushima nuclear disaster). Another potential application is a team of vehicles cooperatively transporting an object too large and/or heavy for a single one to transport. For space missions, a multi-agent system could be deployed as explorers that precede human missions, as crew helpers, or as custodians of assets left behind [1].

In addition to the broad range of applications, engineered multi-agent systems provide several advantages to a system operation: more efficient and complex task execution, robustness when one or more agents fail, scalability, versatility, and adaptability. Conversely, multi-agent systems introduce a host of unique challenges, including coordination and cooperation schemes, distribution of information and subtasks, negotiation between team and individual goals, communication protocols, sensing, and collision avoidance. These

challenges are exacerbated by the fact that a task must often be completed with limited computational, communication, and sensing resources.

As such, in multi-agent system design, a decision must be made between a centralized and a decentralized/distributed coordination scheme. In a centralized scheme, each agent has access to measurement and/or control information from a master entity, such as a central processing unit or a global positioning system (GPS). Therefore, centralized schemes have a single point of failure. They also do not scale well with the number of agents because the processing overhead and number of communication links become prohibitive. In a decentralized scheme, cognitive, sensory, computational, and control information are acquired locally by each agent via onboard hardware and software. Consequently, the cost and complexity of each agent begins to rise. From a practical standpoint, both the centralized and decentralized coordination schemes must be used in unison to provide smooth and robust operation. For example, a multi-agent system could be initially deployed using GPS-based scheme. However, the GPS loses accuracy when the line of sight between the GPS receiver and satellite is obstructed (e.g., clouds, dense vegetation, underwater). In such situations, it would be desirable that the system have the capability of switching to a decentralized mode of operation without degrading the mission performance.

In the literature, multi-agent system coordination covers a broad range of problems: aggregation, consensus, agreement, rendezvous, synchronization, social foraging, flocking, coverage, scheduling, and formation. In this manuscript, the experimental work is focused on the class of formation control problems, specifically acquisition and maneuvering. In formation acquisition, agents are tasked to form and maintain a pre-defined geometric configuration. In formation maneuvering, the agents must simultaneously acquire a formation and move along a specified trajectory. Formation control has its place in applications that require networked mobile agents to maintain optimal positions or specific configurations (e.g. collective static/dynamic target tracking, self-driving vehicles). In these control schemes, it is often desired that the agents converge to spatial configurations irrespective of

its exact, global positions in space. In this way, only the relative positions between agents is necessary, and therefore decentralized schemes can be utilized [1].

The current state of research in the area of multi-agent systems is quickly evolving from the theoretical development of coordination control algorithms and their computer simulations to experimental validations on proof-of-concept testbeds. With this in mind, it is desired to develop an accessible experimental testbed at Louisiana State University for multi-agent system coordination control with centralized and decentralized capability. Such a testbed would allow for experimental work to be conducted in-house, enable the rapid prototyping of control algorithms, and potentially lead to new research directions spawned by experimental observations.

1.2. Related Work

In recent years, a number of research testbeds for multi-agent systems have been developed due to the growing need to evaluate new algorithms and methods; see [2–5] and the references therein. Several considerations come into play when developing a suitable testbed: cost, development time, personnel expertise, available space, ease of use, maintenance, scalability, and applications being targeted. Hardware-wise (mechanical and electrical), the most pressing issues are i) the robots' size and method of locomotion, ii) the sensors used for their localization, and iii) the communication system (robot-robot and/or robot-command station). These three issues are determining factors in the overall design of the testbed because they can significantly affect the total cost. Indeed, a single robotic vehicle with some form of sensing and communication capabilities can cost from as little as $50 to over $2,000 [6]. Commercially available multi-robot platforms boast fully capable, easy-to-use, and assembled robots and are therefore attractive options. However, their cost typically prohibits lesser-funded labs from their use. Alternatively, open-source robot platforms are commonly cheaper options, but may not be suitable for specialized research applications. Therefore, to accommodate resource limitations and desired application requirements, a custom design approach must often be taken.

1.2.1. Robotarium

Among the existing testbeds described in the literature, Georgia Institute of Technology's Robotarium [4] seems to strike a good balance between affordability and functionality. At the time of beginning this project, the Robotarium was still in its prototyping phase, and has since been developed into a large-scale facility at Georgia Tech. As such, the discussion below refers to the prototype version of the Robotarium presented in [4].

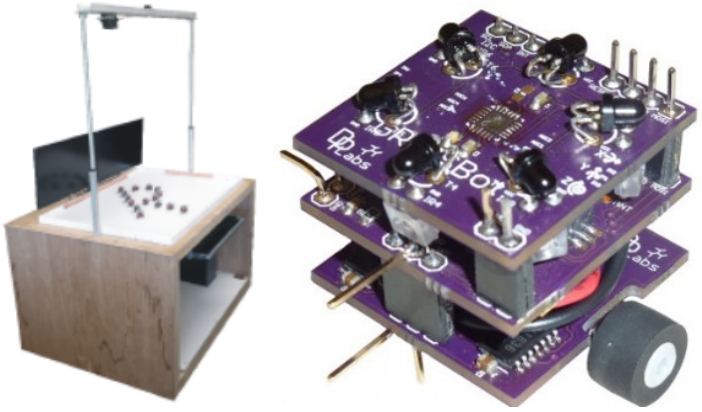

Figure 1.1. The Robotarium (left) and GRITSBot (right) [4, 6]

The Robotarium is a custom-designed, table top testbed (130 x 90 x 180 cm) that provides a plane on which multiple custom-designed robots, called GRITSBots, can navigate (Figure 1.1). An overhead camera provides an absolute position tracking system for use in centralized coordination schemes. The GRITSBot [6] (3 x 3 cm footprint) is a low-cost, wheeled robot equipped with a suite of onboard sensors, wireless communication, battery, and processing boards. The GRITSBot is fully open source and can be constructed for under $50 [4].

The Robotarium originated from the desire to make multi-agent system testbeds remotely accessible to the research community at large. Specifically, remote users submit their control algorithm coded in MATLAB or Python through a website. Then, a lab assistant at Georgia Tech runs the experiment and uploads the experimental data and a video to the user's account on the website. This feature, although being valuable and noble, has an important limitation. Control system experimentation by nature requires the

tuning of the control gains (often by trial-and-error) before the "optimal" performance is achieved. Moreover, the controller often needs to be tested for different initial conditions and/or reference trajectories. That is, multiple experimental runs are typically necessary before one can reach conclusive results. Since the remote-access testbed does not allow the user to independently run the experiment, the experimentation process is much more time consuming and may be incomplete. Ultimately, the Robotarium demonstrates the capability of a low-cost multi-agent experimental testbed. Additionally, as it is open-source, the Robotarium provides a foundation and inspiration for a centralized testbed design that can be modified and upgraded as needed.

1.2.2. Onboard Localization Module

In an indoor centralized coordination scheme, a global positioning system can be implemented using an overhead camera. While overhead cameras provide a solution for absolute and relative localization, such a method cannot provide a truly decentralized result. In order to decentralize the localization capability of the testbed, each robot must be equipped with hardware to measure its relative range and bearing from its neighbors. In the literature, numerous methods and technologies have been used to accomplish distributed localization, each having its advantages and disadvantages. Therefore, the localization method(s) selected must consider the research application, operating conditions, and required performance characteristics.

For robots on the centimeter-scale, cost, size, and scalability of the sensors are of critical importance. Camera and computer vision systems are relatively costly and require complex processing, which is not practical for small-scale robots. Ultrasonic sensors (US) using time-of-flight calculations have proven to be accurate over long ranges, are relatively low-cost, and are less susceptible to occlusion [7, 8]. However, US suffer from slow refresh rates due to slow sound dissipation, especially in enclosed environments [8]. Additionally, these systems are bulky relative to centimetre-scale robots and are dependent on environmental conditions such as humidity and temperature [7–9]. Audio and radio methods suffer from

similar issues, as well as noise interference, and are therefore better-suited for long-range applications on the meter-scale. Alternatively, infrared (IR) methods are capable of higher refresh rates, have physically lighter and smaller hardware requirements, and can be implemented in a number of ways to obtain omnidirectional localization capability [8, 10–12].

1.3. Study Objectives

This work is composed of three principle objectives and is organized as follows:

- The first objective is to design, develop, and build an experimental testbed for centralized multi-agent coordination control. Chapter 2 details the design of the testbed, which includes the arena, robots, and control station. Chapter 3 describes the methods and results of a comprehensive performance evaluation to characterize the system, such as processing and robot hardware. Appendices A and B include all relevant design documents (e.g. schematics, drawings, and user manual).

- The second objective is to validate the testbed by executing a specific set of multi-agent control algorithms under varying experimental parameters in the centralized operating mode. Chapter 4 outlines the formation control algorithms used during these experiments. Chapter 5 presents the experimental results and observations.

- Finally, the third objective is to design, develop, build, and test the required on-board hardware for localized sensing, and thereby decentralized operation. Chapter 6 describes the sensor design and characterization.

Chapter 2
TIGER Square

The scope of this chapter will be describe the design of TIGER Square in its centralized operating mode. The major system components and their relationships are illustrated in the top left image of Figure 2.1. In this coordination scheme, the Control Station (TSCS) handles all higher-level processes, wireless communication, localization, and control velocities.

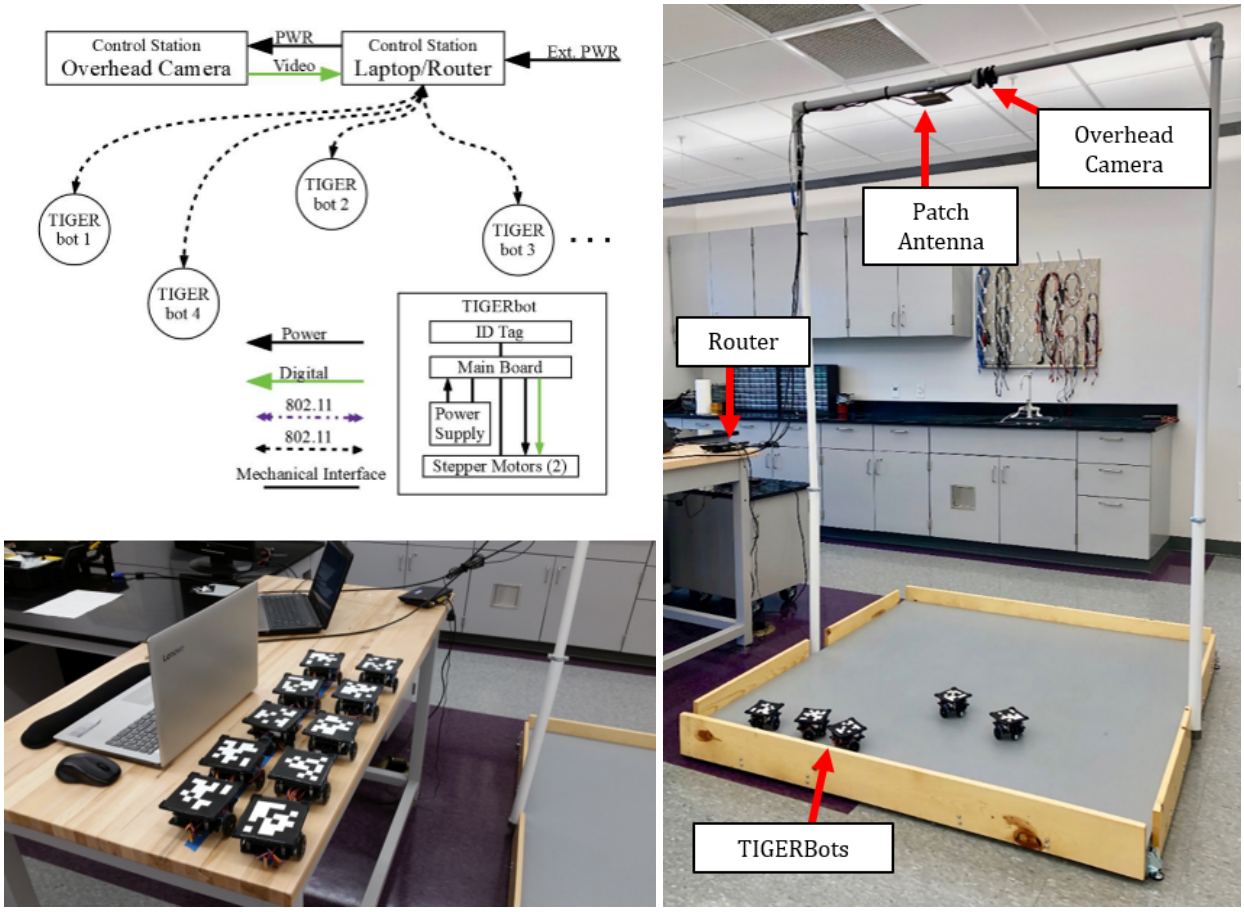

Figure 2.1. Multi-Agent Experimental Testbed, TIGER Square. (Top Left) System Block Diagram. (Bottom Left) Control Station Set Up. (Right) Operating Environment.

2.1. Control Station

The TSCS contains the computational/communication hardware and software needed for control algorithm development and initialization/interruption of an experimental run. Physically, the control station is a laptop computer (Lenovo IdeaPad: Intel Core i7-7500U CPU @ 2.70 GHz, 16 GB RAM, 64 bit, Windows 10), an Ethernet-connected WiFi router (ASUS RT-N12), and a hard-wired overhead camera (plug-and-play HD USB webcam).

2.1.1. Control Software Structure

The control software is a set of MATLAB code designed to wrap around any input algorithm script to execute experiments. Currently, a top-level script is used to run experiments by initializing the system and then running a user-defined control algorithm script, see Figure 2.2. Initialization includes creating a TIGER Square object defined by an external class file. This class file contains configuration properties and high-level macros that can be called by the user (e.g. retrieve robot poses, send velocities, plot saved data). The control algorithm script is meant to be written by the researcher or potential user of the system. These users will modify a basic skeleton script that provides the structure and syntax of an algorithm. Simply put, the user provides single-integrator velocity-level control inputs for each robot and TIGER Square sends off those inputs to the robots, outputs updated poses, and records the data.

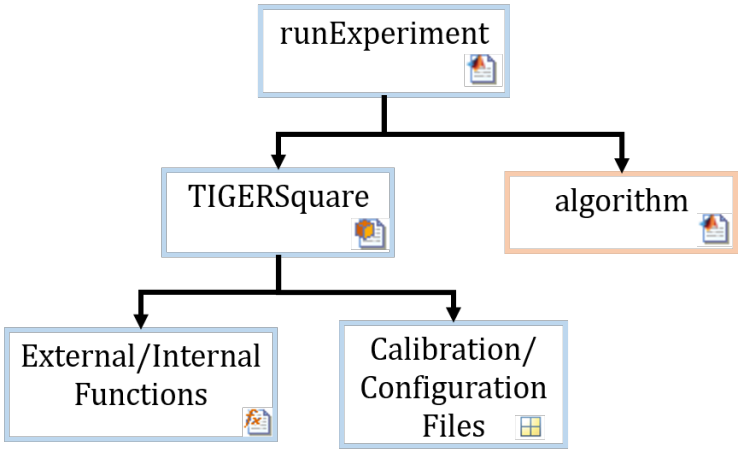

Figure 2.2. TSCS Software Structure.

2.1.2. Centralized Localization

In the centralized operating mode, the TSCS uses the USB webcam to retrieve a video frame of the scene to measure the robot poses within a global framework in real-time. Henceforth, this subsystem will be referred to as the robot *detector*. Each robot has a fiducial marker placed on its top layer that is used to identify and localize the robots. Once a video frame is captured, each robot is localized by extracting a set of features from the frame and matching them to pre-extracted features of the marker reference images (Figure 2.3). By transforming the coordinates of the reference image into the frame image, the coordinates of the marker in the frame are obtained. The coordinates of the robot in the frame are then converted from pixels to world units (mm) with a preset conversion ratio, and saved to a matrix variable. This matching process is repeated for each robot, during each iteration cycle. (Figure 2.4).

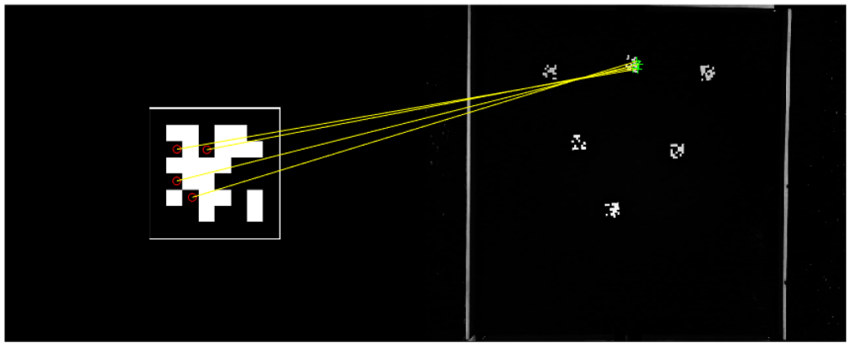

Figure 2.3. Feature Matched Points in a Scene. The scene image is captured with specific camera settings (brightness, sharpness, auto-focus) to improve the image processing.

The robot fiducials are retrieved from AprilTags, an open-source visual fiducial system [13]. However, the detection method does not incorporate the AprilTags code, and instead uses feature point detection functions found in the MATLAB Computer Vision Toolbox. This decision was made to simplify the design of the TSCS software and keep the processing within a single application.

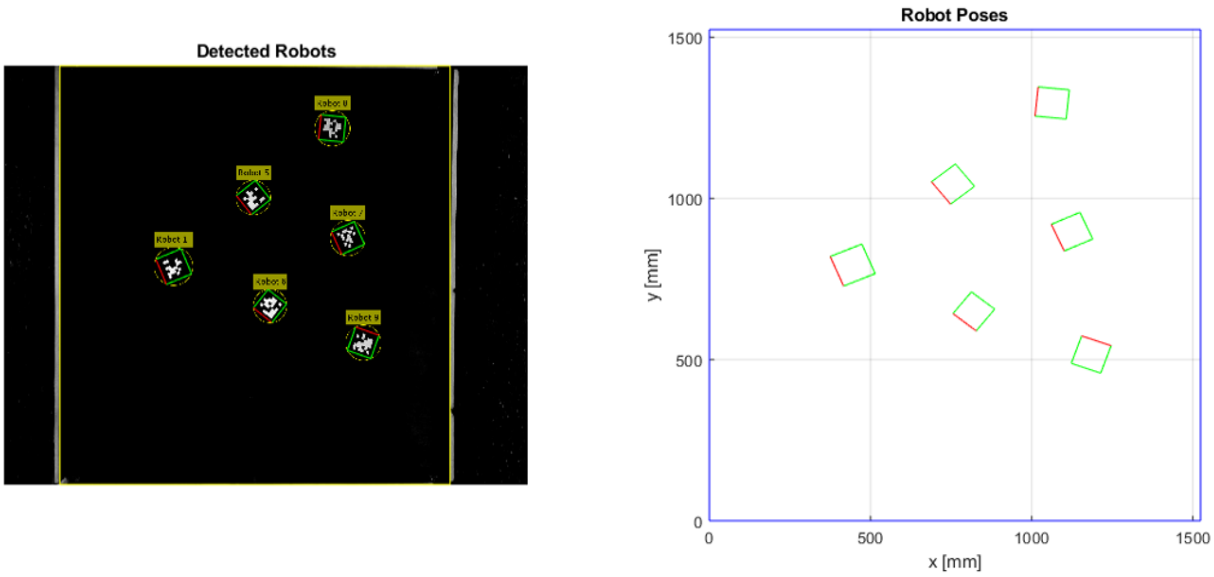

Figure 2.4. Localized Robots in a Scene.

2.1.3. Video Recording

For qualitative performance assessments, video recording is accomplished through a secondary overhead camera and computer. The video is recorded at a resolution of 1024 x 720 pixels and at 30 frames per second. The saved videos can then processed be through a MATLAB script for presentation of results.

2.2. TIGERBots

Each TIGERBot (Figure 2.5) is comprised of a 3D printed chassis, a pair of stepper motors with attached 3D printed wheels, a Li-Po battery pack, and the Main Board PCB. The TIGERBot has a 9 x 9 cm footprint and stands at about 7 cm tall. Inter-board connections are accomplished through stackable header pins and standoffs to provide both electrical and mechanical interfaces. The following subsections discuss the design and features of the TIGERBot. Figure 2.6 illustrates the inter-robot component relationships. Table 2.1 summarizes the general specifications of the TIGERBot.

2.2.1. Processing

As the core of the robot, the Main Board features two microcontrollers, the NodeMCU (a development board built around the WiFi-enabled ESP 8266 chip) and an ATmega

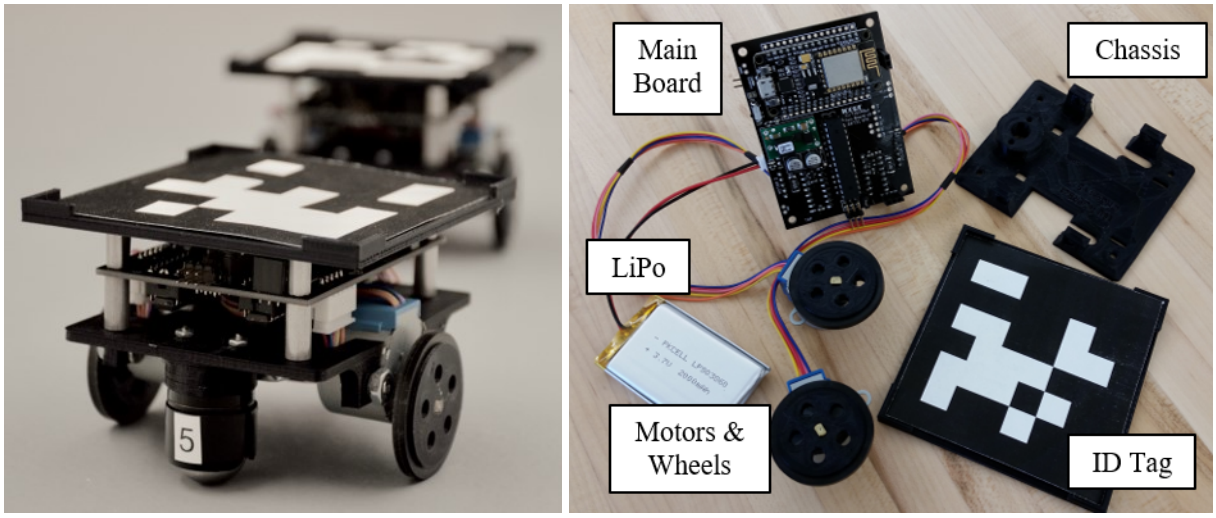

Figure 2.5. TIGERBot Assembly.

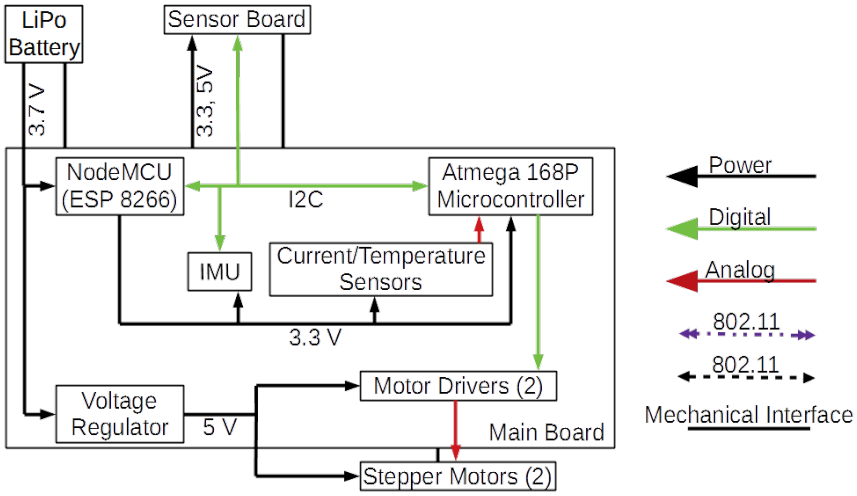

Figure 2.6. TIGERBot System Block Diagram.

Table 2.1. TIGERBot Specifications

Parameter	Value	Units
Max linear speed	3.5	cm/s
Max angular speed	45	deg/s
Geometric footprint	9 x 9 x 7	cm
Average mass	243	g
Operating time (typ)	2	hrs
Cost per unit	78.73	USD

168P chip, outlined in Figure 2.7. The NodeMCU is set as the master microcontroller, and coordinates internal data flow and wireless communications. The ATmega chip controls the two stepper motor and interfaces with the analog introspective sensors. Through an I^2C bus, the NodeMCU sends and receives data from other devices on the board, such as the ATmega chip, the IMU, or potential board attachments.

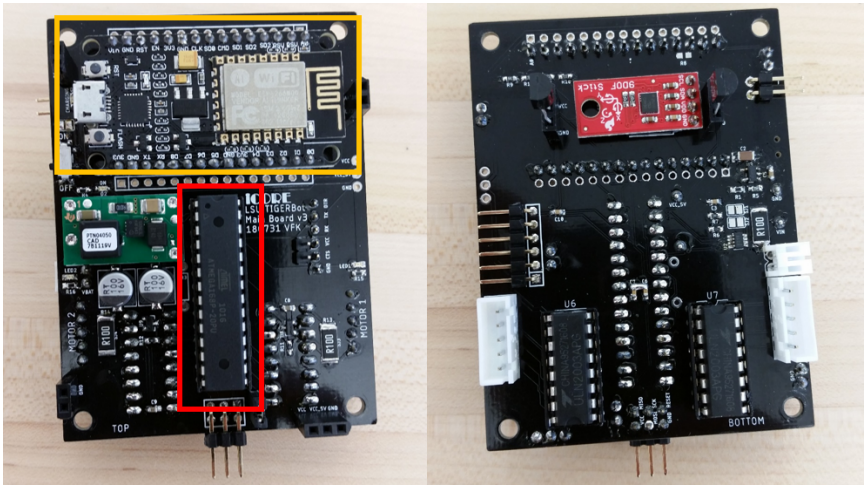

Figure 2.7. Top and bottom view of TIGERBot Main Board. The NodeMCU and ATmega chip are outlined in orange and red, respectively.

2.2.2. Communication

The TSCS includes a private 2.4 GHz router that is hard-wired (ethernet) to the control laptop. In order to minimize campus wireless interference, a directional patch antenna is used with the router and is positioned above the robots on the PVC pipe assembly. The TIGERBot's NodeMCU supports IEEE 802.11 B/G/N standards at 2.4 GHz. In the centralized configuration, the Control Station broadcasts UDP command messages through the wireless router. These commands are received, parsed, and carried out by each robot. The robots can then broadcast messages to be received by the Control Station.

2.2.3. Locomotion

The TIGERBot uses two 28BYJ48 5 V stepper motors for differential drive locomotion. Stepper motors were selected to eliminate the need for wheel encoders and allow for odometry to be completed by simply counting step cycles. The wheels of the robots are

3D printed disks with a center groove for a rubber, square profile, O-ring tire. A 1.9 cm steel-ball castor wheel is positioned at the front of the robot to balance the robot.

Table 2.2. TIGERBot Motor and Wheel Specifications

Parameter	Value	Units
Gearing	64:1	
Frequency	100	Hz
Step angle	5.63	deg/step
Steps per revolution	4076	steps
Cost per motor	2.30	USD
Wheel base	89.25	mm
Wheel diameter	37.00	mm

2.2.4. Introspective Sensors

Each TIGERBot is equipped with a suite of introspective sensors. A battery monitor (INA219) circuit is used to monitor the charge of the battery and total power consumption data. A pair of current sensors (ZXT1009) are integrated into the motor circuit to monitor the power consumption of the motors. Lastly, a 9-DOF Inertial Measurement Unit (IMU) is included for additional localization feedback, if necessary.

2.2.5. Power Regulation

A single-cell (nominal 3.7 V) 2000 mAh LiPo battery is used the power the TIGERBot. The operating voltage from the battery ranges from 3.3 to 4.2 V. As shown in Figure 2.6, the battery supply line feeds into the NodeMCU and a voltage regulator. An embedded step-down regulator on the NodeMCU regulates the battery voltage into a 3.3 V power line. Similarly, a boost regulator featured on the Main Board regulates the battery supply into a 5 V power line. Additionally, the Main Board includes an embedded LiPo charging circuit (MCP7383) that operates on a 5 V input.

2.2.6. Structural Components

Figure 2.8 shows an exploded CAD rendering of the TIGERBot. The components highlighted pink are 3D printed and grey are off-the-shelf. The 3D printed component drawings and itemized bill of materials are shown in Appendix A. In Figure 2.8, the board highlighted blue represents the placement of the Main Board.

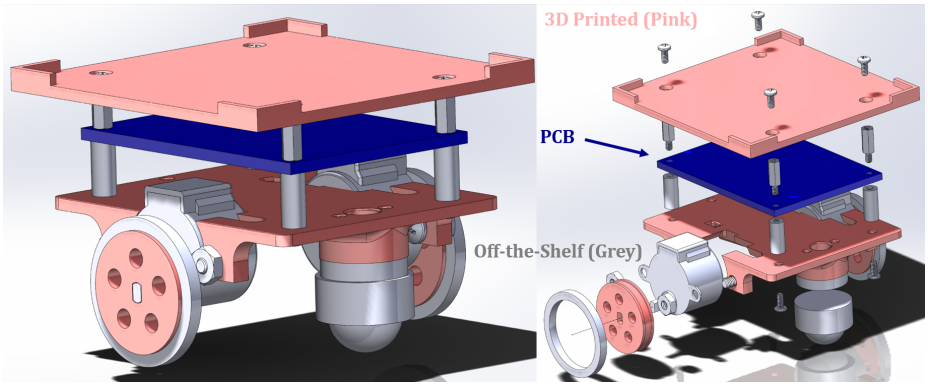

Figure 2.8. TIGERBot Rendering.

2.3. Arena

The arena is a 1.5 x 1.5 m enclosed, flat platform (Figure 2.1) in which the TIGERBots operate. The arena is constructed from a steel frame that supports the top platform: a single, epoxy-coated, Baltic birch plywood sheet. The platform material and epoxy coating provides a non-slip, even surface, at a relatively low cost and weight. The frame also features four adjustable feet placed at the outside corners to allow for an even-level setting. The adjustable feet are set with a 5 mm hex key. The overhead cameras used for the robot detector and video recording are mounted to a telescoping PVC pipe structure that can be vertically adjusted for the desired field of view.

2.4. Cost

In parts and materials alone, the final cost of each TIGERBot for centralized operation is $78.73. The testbed, in its entirety, required about $3300 to build. The proposed and final project budget breakdown are shown in Figure 2.9. An itemized bill of materials can be found in Appendix A.

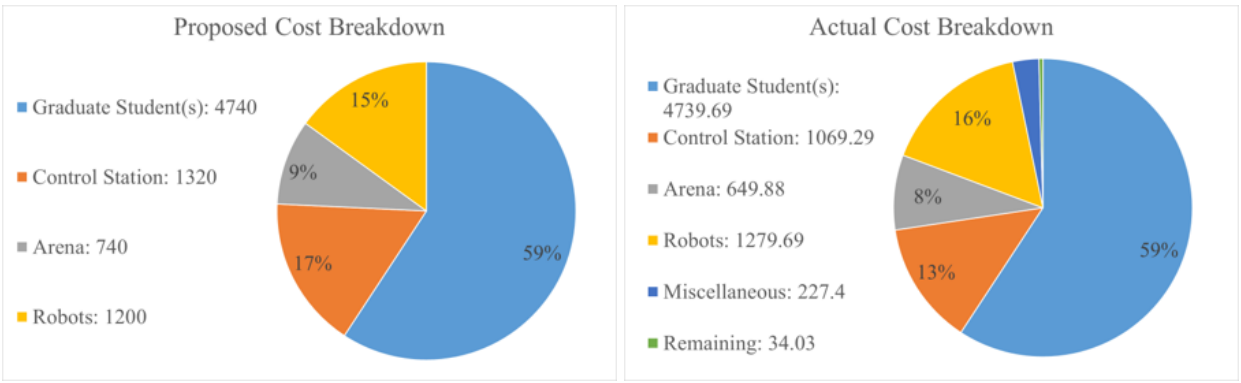

Figure 2.9. TIGER Square Budget Summary.

Chapter 3
Performance Evaluation

An evaluation of TIGER Square and its subsystems will inform potential users of the expected performance and behavior of each component. Each of the following sections summarize the approaches taken to test a subsystem component, the results, and key outcomes.

3.1. Centralized Localization

3.1.1. Localization Speed

The detector is set to capture images with a resolution of 1280 x 1024 pixels. Through trial and error, it was determined that this resolution provided the optimal combination of localization accuracy and speed for the robot detector. During operation, the TSCS will cycle through a process that includes localization of the robots. In order to maintain predictable trajectory behavior, each cycle must be set to a predefined, constant step time. From Figure 3.1 it is clear that the detector localization speed is dependent on the number of robots being detected. A recommended step time curve is obtained by applying a biased linear fit to the data presented in the left panel of Figure 3.1. Therefore, the inverse of the step time curve proves a recommended refresh rate, shown in the right panel of Figure 3.1. For example, with four robots in use, each cycle iteration will require up to 0.18 seconds or the system cycles at about 5.5 Hz.

As it is now, the observed localization frame rate of the detector is far from reaching standard rates being achieved by other real-time object detection methods [14, 15]. While the detection method described in Section 2.1.2 is relatively simple, it is a "brute force" approach that demands higher resolution images for feature matching. Fortunately, due to the speed limitations of the robots (detailed in Section 3.2), the slow detector frame rate sufficiently captures the evolving robot poses in time. Nonetheless, this aspect of the system is a major improvement point and Section 7.2 makes an effort to lay out some options for increased performance.

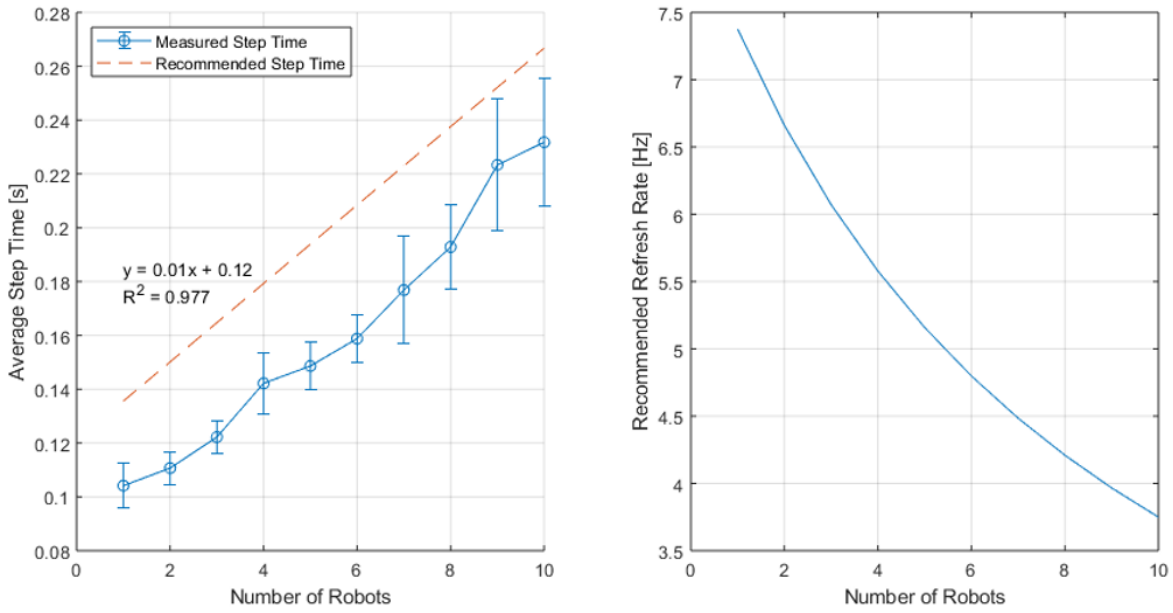

Figure 3.1. Detector Speed Test. Error bars indicate one standard deviation.

3.1.2. Localization Accuracy

An evaluation of the robot detector measurement accuracy was accomplished by randomly placing a robot in the arena and comparing a manual pose measurement to an average (100 samples) pose measurement obtained by the robot detector. In order to manually measure the pose of the robot, the robot was centered on a 3D printed pad or "jig." The distance from the outer edges of the jig to the bottom-left corner of the arena were measured to obtain the x-y position of the robot. The orientation of the robot was manually set using the five degree tick markings along the inner perimeter of the jig. This process was repeated for ten data points, shown in Figure 3.3.

In this evaluation, the measured distance error e_i between the ruler and camera for the ith position of the robot is given by

$$e_i = \left\| p_i^R - p_i^C \right\|, \quad i = 1...10. \tag{3.1}$$

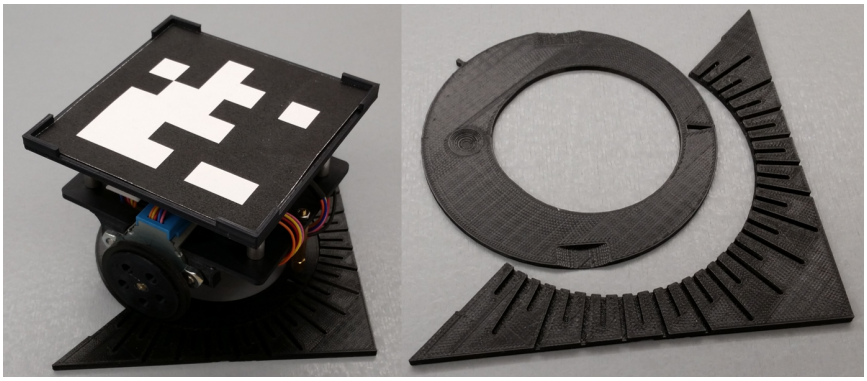

Figure 3.2. 3D Printed Placement Jig. Using this jig, the robot's pose can manually be measured.

where the position is denoted as $p_i = [x_i, y_i]$. Similarly, the heading error β_i is defined as

$$\beta_i = |\theta_i^R - \theta_i^C|, \quad i = 1...10. \tag{3.2}$$

where θ_i denotes the desired heading direction of position i.

Table 3.1. TSCS Localization Errors

	Distance, e_i [cm]	Heading, β_i [deg]
1	0.21	0.03
2	0.57	1.65
3	0.86	0.62
4	1.56	1.73
5	0.89	1.98
6	2.54	2.06
7	1.03	0.72
8	1.13	1.05
9	1.29	0.87
10	1.28	1.60

A summary of the distance and heading errors relative to the physical measurements are presented in Table 3.1. When considering the measurement uncertainties, the two data sets did not consistently match up. The average TSCS position measurements (x and y) each had a standard deviation between 0.2 to 0.3 cm. The average TSCS orientation measurements had standard deviations between 0.5 to 2.4 degrees. The physical position and

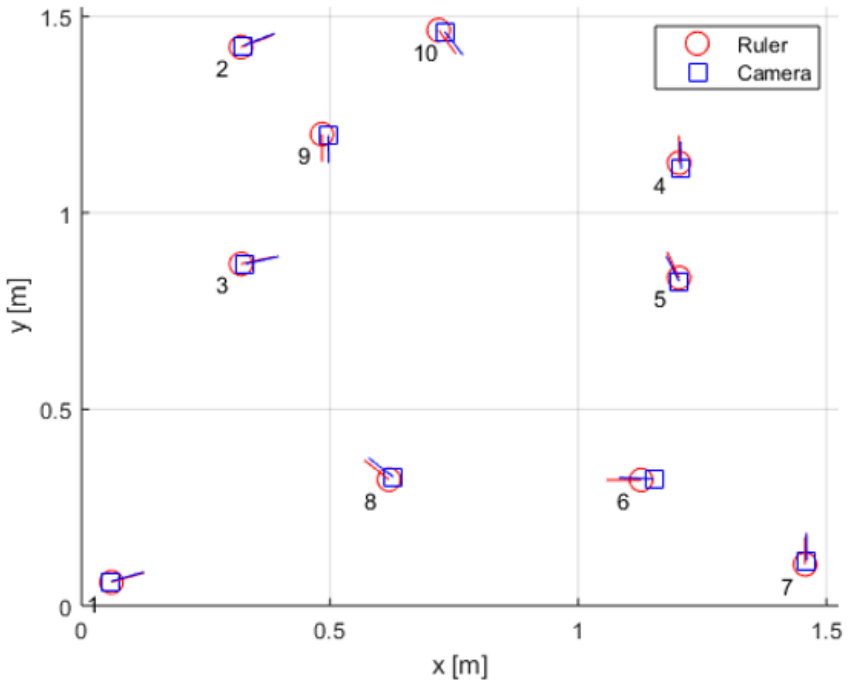

Figure 3.3. Detector Accuracy Test. Protruding lines indicate the robot heading direction.

orientation measurement uncertainties were 0.7 cm and 2.5 degrees, respectively. With this in mind, the angular orientations matched, but some larger distance errors (e.g. position 6) did not agree. Therefore, these results should be considered when assessing multi-agent experiment performance, specifically those focused on distance and heading errors.

3.1.3. Error Handling

In the event that the detector fails to accurately measure the pose of a robot, an estimated pose will be calculated based on a previously measured pose and input velocity, in real-time. These detection errors will typically occur when the feature-matching results in a distorted transformation, false positive, or switched pose. As a result of the detection error, the position of the robot may suddenly change from its previously recorded position. Sudden position changes greater than 5 cm are set to trigger the pose correction algorithm.

3.2. Motors and Odometry

Due to the nature of stepper motors, the TIGERBot does not have a direct method for onboard odometry and instead relies on the timing of digital pulses to control the

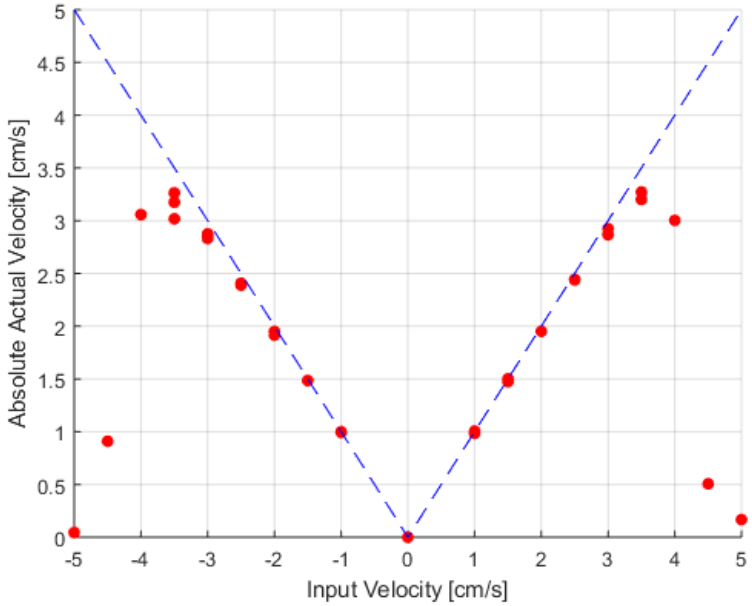

Figure 3.4. Linear Speed Test.

motor speed and estimate odometry. As mentioned in Section 2.2.3, each TIGERBot is equipped with two independently driven 28-BYJ48 motors. In order to measure the open-loop performance of the robot motors, a series of tests were performed. In each of these tests, the TSCS sent a constant input velocity to a single robot and used the robot detector to record its pose with time.

3.2.1. Linear Speed Test

First, the forward and reverse linear velocity of a single robot was tested. The robot was controlled to move in a straight line up to a specified distance for a duration of time proportional to the constant velocity being tested (e.g. 1 meter at a speed of 0.01 m/s for 100 seconds). The average velocity for each trial was calculated from the measured distance traveled and the input duration. This test was repeated for a range of velocities from 1 to 5 cm/s.

From Figure 3.4, it can be noted that until about 2 cm/s, the actual speed accurately follows the desired input speed. At an input speed of about 3.5 cm/s, the motors stall and eventually drop off.

3.2.2. Circular Trajectory Test

Next, a single robot was controlled to move along a circular trajectory at a constant velocity for a specified number of laps and trajectory direction. Figure 3.5 shows the recorded trajectory for a single robot during an test trial. The following subsections discuss the results of these trials amongst multiple robots.

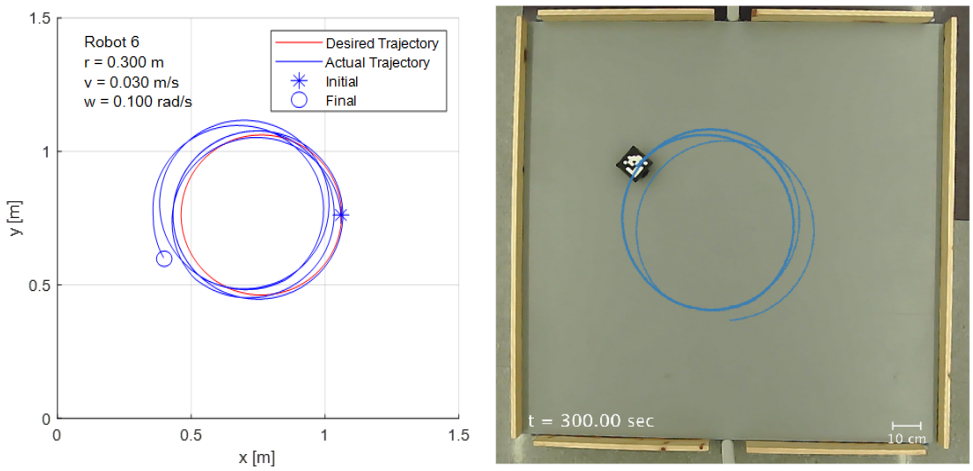

Figure 3.5. Open-Loop Circle Trajectory.

- **Single Robot Trials**

This test had a single robot run through the velocity commands for five laps, in the counter-clockwise direction, for a total of three trials. In these tests, the input linear and angular velocities were a constant 3 cm/s and 0.1 rad/s, respectively.

From observation, the open-loop performance of a single robot would vary between trials. Upon inspecting the trajectory (Figure 3.5) and error plots, it is clear that at these velocity settings, the robot (1) could not maintain the desired velocity and (2) occasionally "missed a step", which caused it to veer off course. The fact that the desired speed could not be maintained meant that the errors to grew linearly. The oscillatory trend in the error was a result of the robot veering off course. For example, at about 160 seconds in Figure 3.6, the heading error for Trial 3 suddenly drops, indicating the robot abruptly turned. Consequently, at this instance, the distance error began oscillating.

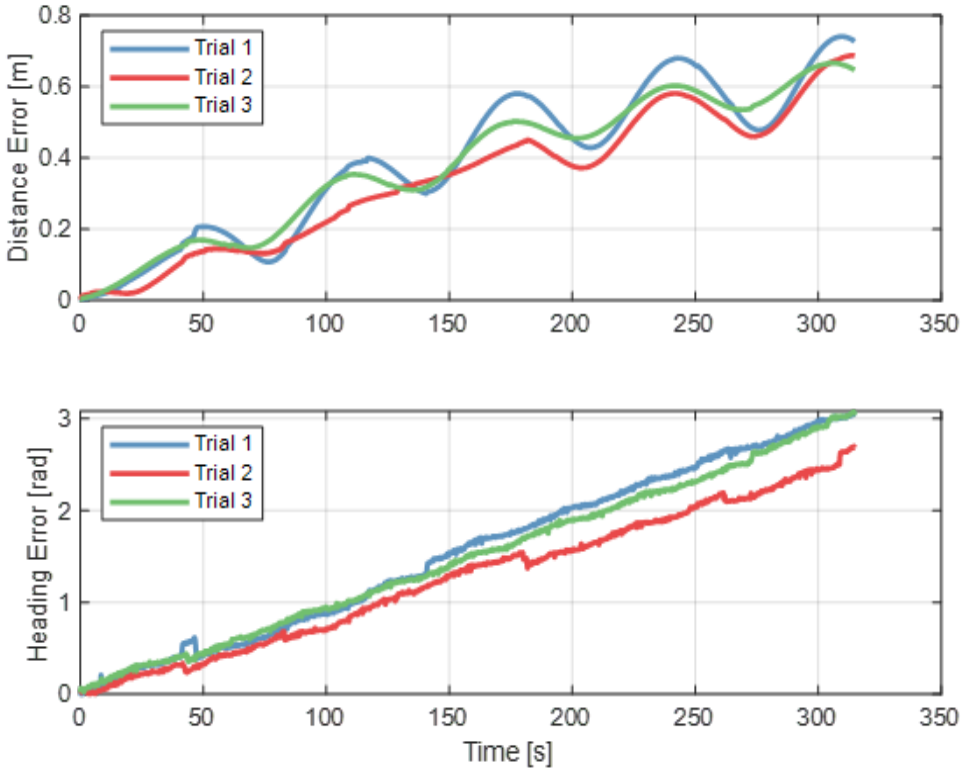

Figure 3.6. Open-Loop Circle Trajectory Errors for Single Robot.

- **Multiple Robot Trials**

This section adds to the former by considering that each robot may perform differently. To demonstrate how well each of the robots followed the open-loop commands, this test had each of the ten robots run through the velocity commands for 2 laps, in the clockwise direction, for a single trial. Overall, the results presented in Figure 3.7 reinforced the observed variance from the single robot trials.

3.2.3. Swapped Motor Test

In the interest of improving the maximum speed of the robots, a pair of stepper motors with identical topology but smaller gear ratio (16:1) was implemented on a single robot and tested using the procedures above. Due to the reduced gearing, the 16:1 motor, henceforth referred to as Motor 2, theoretically has the capability to run four times faster than Motor 1 (64:1 motor). Figure 3.8 shows the linear speed performance of the Motor 2. Overall,

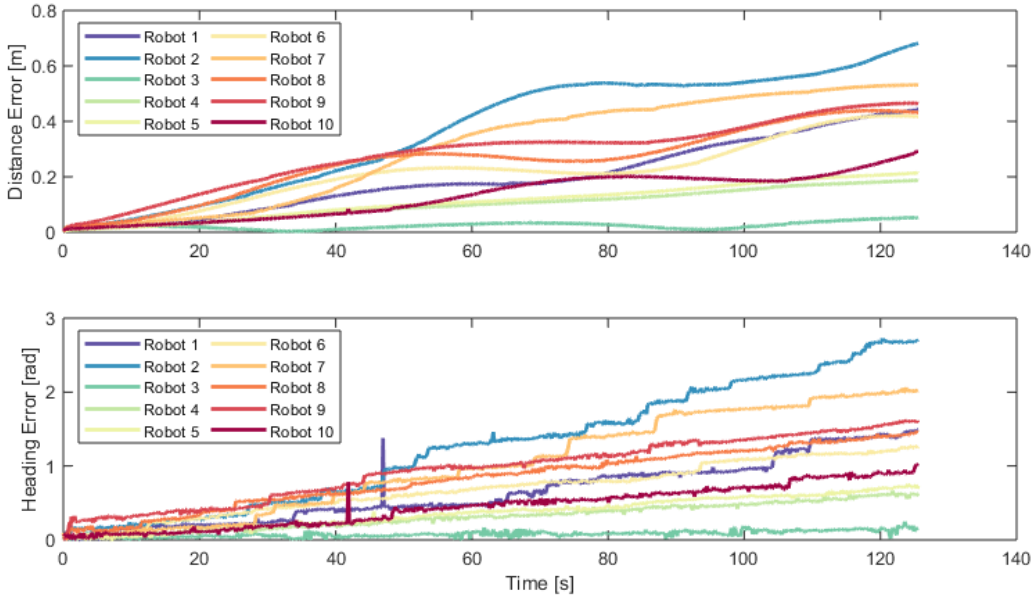

Figure 3.7. Open-Loop Circle Trajectory Errors for Multiple Robots. The heading error spikes on robot 1 and 10 are a result of the pose measurement errors in the robot detector.

the behavior between the two motors are similar: linear up to a certain velocity, at which point the actual linear velocity begins to plateau. In this test, the motors ramp up to an actual linear velocity of about 9 cm/s. Additionally, when running the circular trajectory test with Motor 2, the speed could be increased without noticeable sacrifice to trajectory errors (Figure 3.9).

From these tests, Motor 2 demonstrates increased speed with little to no sacrifice to odometry performance. The unit cost for Motor 2 is only about 5 USD (twice the cost of Motor 1), and therefore appears to be a viable, low-cost option for improving the robot speed limitations.

3.2.4. Discussion

These tests have revealed that at higher input speeds, the stepper motors will struggle to maintain speed and therefore accuracy. It is unclear when exactly missed steps seem to occur, as it may be a result of over-driving the motors or an unexpected delay in the TSCS command sequence. Additionally, the uncertainty in physical dimensions of

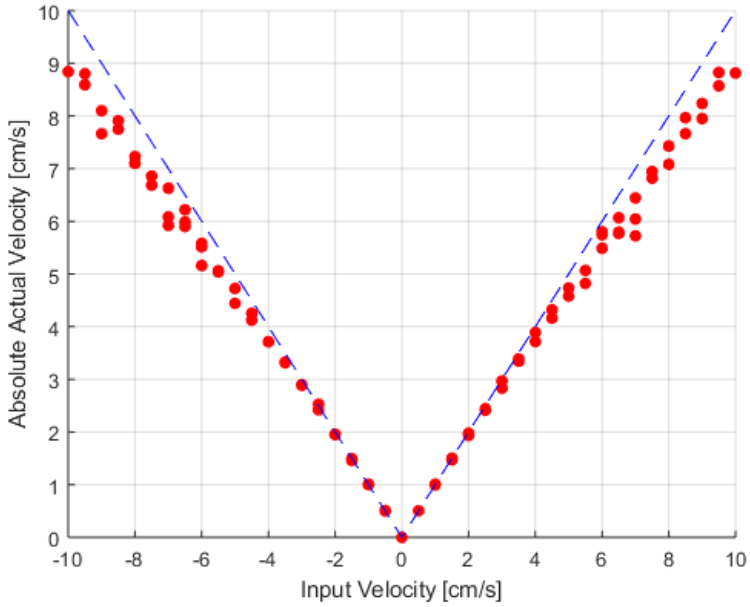

Figure 3.8. Linear Speed Test for Motor 2.

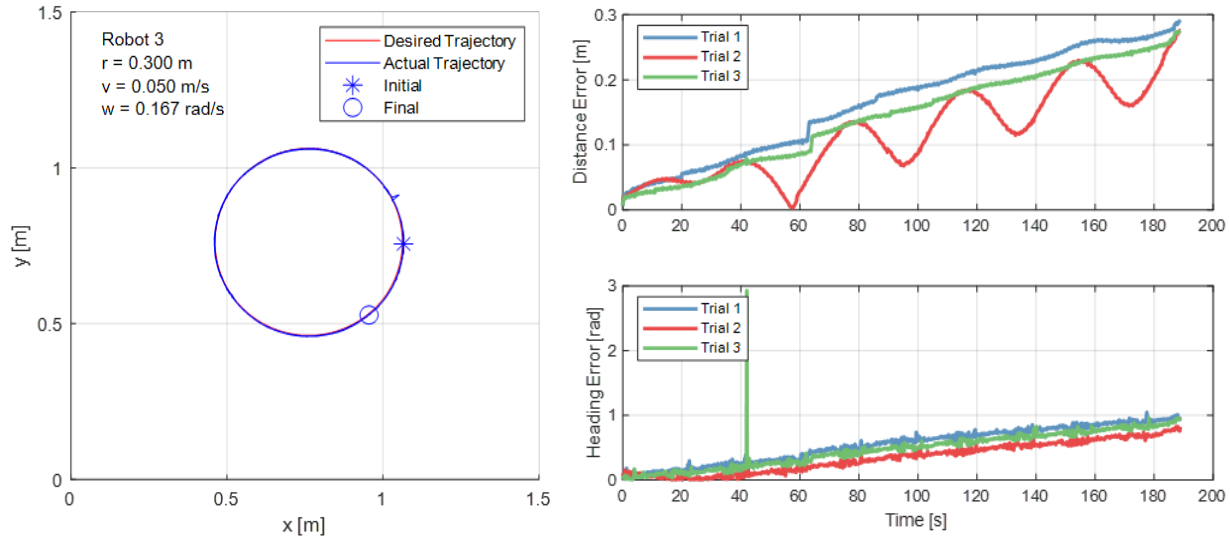

Figure 3.9. Open-Loop Circle Performance for Motor 2. The robot's trajectory for Trial 3 is shown on the left. The heading error spike on Trial 3 is a result of a pose measurement error in the robot detector, which is also seen as a small "blip" in the trajectory plot.

the robot components, gear wearing of the motor, decimal rounding/truncation in computer/onboard processes, and value conversions within the software may be affecting the open-loop performance. That being said, with either Motor 1 or Motor 2, the performance remains predictable when operating within the linear speed regions. Fortunately, due to the positional feedback of the system, these issues are not of critical importance but are considered in Section 7.2 (Future Work).

3.3. Power Consumption

The NodeMCU (wireless communication microcontroller) and stepper motors are the major power-hungry components in the TIGERBot. During normal, continuous wireless operation, the NodeMCU consumes between 200 to 300 mA at 3.3 V. Meanwhile each stepper motor consumes about 300 and 250 mA at 5 V when idling or moving, respectively. The additional electronic components of the TIGERBot are estimated to consume up to 100 mA at 3.3 V. Table 3.2 summarizes the required power based on these figures. Provided the battery has a rated capacity of 2000 mAh at a nominal 3.7 V, the theoretical estimated operating time is about 1.78 hours. The onboard charging circuitry limits the charging current to 500 mA. At the rated capacity of 2000 mAh and an assumed 20% efficiency loss, the robot batteries require up to 4.8 hours to charge from a fully drained state. However, it is not recommended to allow the battery voltage to drop below 3.6 V. The operating and charging time has not explicitly been tested, however the theoretical estimations agree with observations made during regular use of the robots.

Table 3.2. TIGERBot Power Budget

	Qty	Component Current [A]	Operating Voltage [V]	Power Subtotal [W]
NodeMCU	1	0.25	3.3	0.83
Stepper Motors	2	0.60	5.0	3.00
Misc. Components	1	0.10	3.3	0.33
Power Total [W]				4.16

Referring back to Figure 2.6, it is important to remember that the power supply line from the battery is split between the NodeMCU and a power regulator module (PTN04050C). The NodeMCU has an onboard regulator that steps down the battery voltage to 3.3 V, with a maximum current output of 800 mA. Meanwhile, the power regulator featured on the Main Board steps up the battery voltage to 5 V with a maximum current output of 2.4 A. In the centralized mode, only the stepper motors operate on the 5 V supply line and all other components operate on the 3.3 V line.

Chapter 4
Algorithms

As mentioned in Section 1.1, the focus of the experimental work is in formation control problems. The following sections outline the theory behind the formation control algorithms executed in Chapter 5. Additionally, the discussion will provide the necessary understanding to qualitatively evaluate the experiment results and make observations on the testbed performance.

4.1. Graph Theory

When describing multi-agent systems (e.g. shape, communication, sensing, and control topology), it is convenient to use graph theory. More specifically, rigid graph theory, which directly controls inter-agent distances. That is, rigid graph theory naturally ensures desired formation acquisition, and implicitly ensures collision avoidance. This section describes some basic concepts of rigid graph theory in two dimensions and provides the foundation for the control algorithms presented in the subsequent section.

4.1.1. Graph

An undirected graph G is a pair (V, E) where $V = \{1, 2, ..., n\}$ is the set of nodes and $E \subset V \times V$ is the set of undirected edges that connect two different nodes. For any graph, the number of edges, l, in E belongs to the set $l \in \{1, \ldots, N(N-1)/2\}$. The set of neighbors of node i is denoted by

$$\mathcal{N}_i(E) = \{j \in V \mid (i, j) \in E\}. \tag{4.1}$$

A directed graph G is a pair (V, E^d) where the edge set E^d is directed in the sense that if $(i, j) \in E^d$ then i is the source vertex of the edge and j is the sink vertex. For $i \in V$, the out-degree of i (denoted by out(i)) is the number of edges in E^d whose source is vertex i and sinks are in $V - \{i\}$.

In the context of this manuscript, the undirected and directed graph represent the

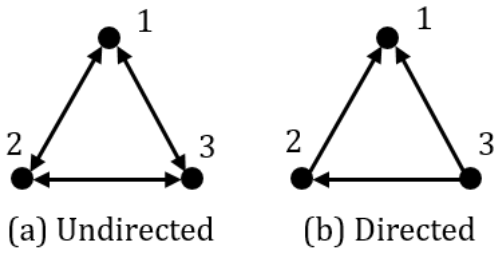

Figure 4.1. Example of an Undirected and Directed Framework with Three Nodes.

tracking connections between agents. For example, in Figure 4.1(b), agent 3 tracks agents 2 and 1.

4.1.2. Framework

A framework is the realization of a graph at specific points in Euclidean space (e.g. 4.1), and can therefore model a physical structure. If $p = [p_1, ..., p_n] \in \mathbb{R}^{2n}$ where $p_i \in \mathbb{R}^2$ is the coordinate of the ith vertex, then a framework F is defined as the pair (G, p). In order to find the length of the edges in a framework, the edge function $\gamma : \mathbb{R}^{2n} \to \mathbb{R}^a$ is defined as [1]

$$\phi(p) = [..., ||p_i - p_j||^2, ...], \ (i, j) \in E \tag{4.2}$$

such that its mth component, $||p_i - p_j||$, relates to the mth edge of E connecting the ith and jth vertices [1].

4.1.3. Rigid Graphs

Two frameworks, (G, p) and $(G, \hat{p})$, are equivalent if $\gamma(p) = \gamma(\hat{p})$, and are congruent if $||p_i - p_j|| = ||\hat{p}_i - \hat{p}_j||, \quad \forall i, j \in V$ [1]. An isometry of $\mathbb{R}^2$ is a bijective map $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$||w - z|| = ||\mathcal{T}(w) - \mathcal{T}(z)||, \ \forall w, z \in \mathbb{R}^2 \tag{4.3}$$

This map includes rotation and translation of the vector $w - z$. Two frameworks are isomorphic if they are correlated via an isometry. Here, the collection of all frameworks that are isomorphic to F are denoted by Iso(F). The rigidity matrix $R : \mathbb{R}^{2n} \to \mathbb{R}^{a \times 2n}$ is

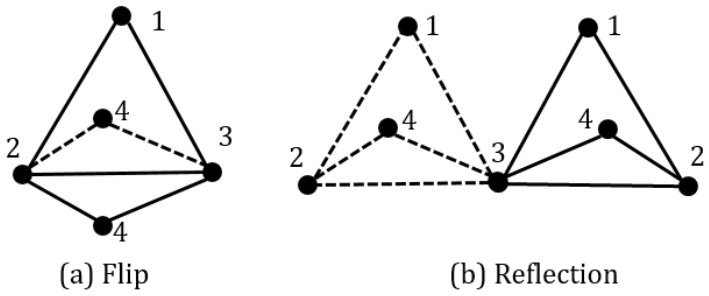

Figure 4.2. Example Framework Ambiguities.

given by

$$R(p) = \frac{1}{2}\frac{\partial \phi(p)}{\partial p} \tag{4.4}$$

where $\text{rank}[R(p)] \leq 2n-3$. In two dimensions, a generic framework (G, p) is infinitesimally rigid if and only if $\text{rank}[R(p)] = 2n-3$. A rigid framework is said to be minimally rigid if and only if $l = 2n-3$ [1].

4.1.4. Framework Ambiguities

If the infinitesimally rigid frameworks (G, p) and $(G, \hat{p})$ are equivalent but not congruent, then they are referred to as ambiguous [1] since framework is not uniquely defined by the edge function. Here the notation $\text{Amb}(F)$ denotes the collection of all frameworks that are ambiguous to the infinitesimally rigid framework F. Common types of ambiguities are shown in Figure 4.2.

4.1.5. Signed Area

The signed area of a triangular framework, $S : \mathbb{R}^6 \to \mathbb{R}$, is defined as [16]

$$S(p) = \frac{1}{2}\det\begin{bmatrix} 1 & 1 & 1 \\ p_1 & p_2 & p_3 \end{bmatrix} = \frac{1}{2}\left(p_3 - p_1\right)^{\intercal} J \left(p_3 - p_2\right) \tag{4.5}$$

where

$$J = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{4.6}$$

Therefore, the signed area is positive if the vertices are ordered counterclockwise and negative if ordered clockwise. Additionally, (4.5) is zero if any two vertices are collocated or if the three vertices are collinear.

4.2. Formation Control Problems

This section introduces two formation control problems for multi-agent systems. Consider a system of n mobile agents where $p_i \in \mathbb{R}^2$ represents the position of agent i in a fixed coordinate frame and $u_i \in \mathbb{R}^2$ is the corresponding control input. In this work, the control input will be based on the single-integrator model which only includes position and velocity variables. This kinematic model is for holonomic (omnidirectional) agents and therefore cannot be directly be applied to nonholonomic agents. This topic is further discussed in Section 4.3.

Let the desired agent formation be modeled by the framework $F^* = (G^*, p^*)$ where $G^* = (V^*, E^*)$, $\dim(V^*) = l$, $\dim(E^*) = a$, $p^* = [p_1^*, \ldots, p_n^*]$, and $p_i^* = [x_i^*, y_i^*]$. The fixed target distance separating the ith and jth agents is given by

$$d_{ij} = \left\| p_i^* - p_j^* \right\| > 0, \quad i, j \in V^*. \tag{4.7}$$

It is assumed that F^* is constructed to be infinitesimally and minimally rigid. The actual formation of the agents is described by the framework $F(t) = (G^*, p(t))$ where $p = [p_1, \ldots, p_n]$ and $p_i = [x_i, y_i]$. The following additional assumptions are made about the desired and actual formations:

- The set where the agents achieve the desired formation is non-empty.
- The formation and sensing graph is the same and fixed.
- The initial conditions of the agents do not satisfy the desired constraints.
- Global positions are not available to the controller.

4.2.1. Formation Acquisition

The formation acquisition problem seeks to control the agents to acquire and maintain a predefined geometric shape in space. Mathematically, this implies that u_i is designed

such that

$$\|p_i(t) - p_j(t)\| \to d_{ij} \text{ as } t \to \infty, \quad i, j \in V^* \tag{4.8}$$

The following definitions will be presented to simplify the notation of the control law. The relative position of agents i and j is defined as

$$\tilde{p}_{ij} = p_i - p_j, \tag{4.9}$$

while their distance error is captured by the variable [1]

$$z_{ij} = \|\tilde{p}_{ij}\|^2 - d_{ij}^2. \tag{4.10}$$

where $z = [..., p_{ij}, ...] \in \mathbb{R}^2$. Given that $\|\tilde{p}_{ij}\| \geq 0$, note that $z_{ij} = 0$ if and only if $\|\tilde{p}_{ij}\| = d_{ij}$.

The control law for an undirected graph is then [1]

$$u = -k_v R^\intercal(\tilde{p})\, z \tag{4.11}$$

which can be rewritten element-wise as

$$u_i = -k_v \sum_{j \in \mathcal{N}_i(E^*)} \tilde{p}_{ij} z_{ij}, \quad i = 1, ..., n \tag{4.12}$$

where $k_v > 0$ is a user-defined control gain that ensures exponential stability.

Notice that only the inter-agent distances are directly controlled, so the actual formation can converge to any isometry of F^*. Ultimately, this means that the resulting formation can be a framework ambiguity. One simple method to avoid the possibility that $F(t) \to \text{Amb}(F^*)$ is to initialize the agents sufficiently close the desired isometry F^* so as to not converge to the ambiguous isometry. However, in practice this solution may not always be possible. Therefore another solution is to add an area constraint to the control law to guarantee the existence of a unique framework in the Euclidean plane, and thus

avoid having the agents converge to an undesired ambiguous framework. To do this, first it is assumed that F^* is constructed to satisfy the following conditions:

- $\text{out}(1) = 0$, $\text{out}(2) = 1$, and $\text{out}(i) = 2, \forall i \geq 3$.
- If there is an edge between agents i and j, the direction must be $i \leftarrow j$ if $i < j$.
- The graph G^* is directed.

The area error is then defined as [16]

$$\tilde{S}_{ijk} = S_{ijk} - S^*_{ijk}, \ (k,i),(k,j) \in E^* \tag{4.13}$$

where $S_{ijk} = S(p)$ with $p = [p_i, p_j, p_k]$ and $S^*_{ijk} = S(p^*)$ with $p^* = [p^*_i, p^*_j, p^*_k]$. The stacked vector of all area errors is given by $\tilde{S} = [\tilde{S}_{123}, ..., \tilde{S}_{ijk}, ...], \ \forall (k,i),(k,j) \in E^*$.

Finally, the distance and area based control law is defined as [17]

$$u_1 = 0 \tag{4.14a}$$

$$u_2 = -\alpha_2 z_{21} \tilde{p}_{21} \tag{4.14b}$$

$$u_k = -\alpha_k \left(z_{ki} \tilde{p}_{ki} + z_{kj} \tilde{p}_{kj} \right) - \beta_k \tilde{S}_{ijk} J^\intercal \left(\tilde{p}_{ki} - \tilde{p}_{kj} \right) \tag{4.14c}$$

for $2 < k \leq N$, $i < j < k$, and $(k,i),(k,j) \in E^*$. The control law is only a function of $\tilde{p}_{ki}$, $\tilde{p}_{kj}$, d_{ki}, d_{kj} and d_{ji} for $i, j \in \mathcal{N}_k(E^*)$. Thus, the control law is decentralized since it only requires the kth agent to measure its relative position to neighboring agents in the directed graph. Additionally, due to the assumed framework conditions, the graph is directed.

4.2.2. Formation Maneuvering

Formation maneuvering builds on acquisition by requiring the agents to simultaneously acquire a formation and maneuver cohesively along a predefined trajectory. Thus, for this problem, an added objective is

$$\dot{p}_i(t) - v_t(t) \to 0 \text{ as } t \to \infty, \quad i = 1, ..., n \tag{4.15}$$

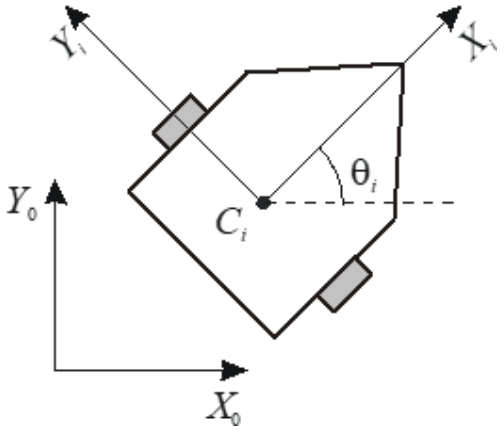

Figure 4.3. Nonholonomic Robotic Vehicle Model with Unicycle Dynamics [18].

where $v_t \in \mathbb{R}^2$ is any continuously differentiable function of time representing the desired translational velocity. It is assumed v_t and $\dot{v}_t$ are bounded for all time.

Therefore, the control law is given by [1]

$$u = u_a + v_t \tag{4.16}$$

where u_a is the formation acquisition control term (such as 4.12 or 4.14). The term $v_t = [v_{d1}, ..., v_{dn}] \in \mathbb{R}^{2n}$ is the desired rigid body velocity defined by

$$v_{di} = v_0 + \omega_0 \times \tilde{p}_{in}, \quad i = 1, ..., n, \tag{4.17}$$

where $v_0(t) \in \mathbb{R}^2$ and $\omega_0(t) \in \mathbb{R}^2$ denote the desired formation translational and angular velocity, respectively. Equation 4.16 can also be rewritten element-wise as [1]

$$u_i = -k_v \sum_{j \in \mathcal{N}_i(E^*)} \tilde{p}_{ij} z_{ij} + v_0 + \omega_0 \times \tilde{p}_{in}, \quad i = 1, ..., n \tag{4.18}$$

4.3. Robotic Vehicle Model

Consider a robotic vehicle, Figure 4.3, where X_0, Y_0 is a fixed reference frame and X_i, Y_i is a moving reference frame attached to the vehicle. The axis X_i is aligned with its heading direction, θ_i, measured relative to X_0. Point C_i corresponds to the vehicle's geometric

center, which is assumed to coincide with its center of rotation. Then it is assumed that the agent motion is governed by the following nonholonomic, unicycle kinematic model

$$\dot{p}_i = S(\theta_i)\eta_i, \quad i = 1, ..., n. \tag{4.19}$$

In (4.19), $p_i = [x_i, y_i, \theta_i]$ denotes the position and orientation of $\{X_i, Y_i\}$ relative to $\{X_0, Y_0\}$ (i.e. pose of the robot). The control input $\eta_i = [v_i, \omega_i]$ includes the ith agent's translational speed v_i in the direction of θ_i, the ith agent's angular speed ω_i about the vertical axis passing through C_i, and

$$S(\theta_i) = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \tag{4.20}$$

The main issue in dealing with a nonholonomic model is that the dimension of the admissible velocity space is smaller than that of the configuration space (i.e. the robot cannot move in all directions). For example, the model shown in Figure 4.3 cannot translate along the Y_i axis. It is therefore implied that a single-integrator model cannot be directly applied to the nonholonomic model. However, because (4.19) is a single-integrator-like equation, a single-integrator model can be implemented on the physical robot platform through the following decomposition

$$\dot{p}_i = \begin{bmatrix} v_i \cos\theta_i \\ v_i \sin\theta_i \end{bmatrix} \coloneqq u_i \tag{4.21}$$

$$\dot{\theta}_i = \omega_i. \tag{4.22}$$

where $u_i = [u_{ix}, u_{iy}]$ is the desired control velocity input. Additionally, let the orientation error be defined as

$$\tilde{\theta}_i = \theta_i - \theta_{di} \tag{4.23}$$

where θ_{di} represents the desired value for θ_i and is calculated as

$$\theta_{di} = \begin{cases} 0, & \text{if } u_{ix} = u_{iy} = 0 \\ \text{atan2}(u_{iy}, u_{ix}), & \text{otherwise} \end{cases} \tag{4.24}$$

Then, the kinematic control law is

$$v_i = u_{ix}\cos\theta_i + u_{iy}\sin\theta_i \tag{4.25a}$$

$$\omega_i = -\beta_i\tilde{\theta}_i + \dot{\theta}_{di}\ ,\ \ i \in V_d \tag{4.25b}$$

$$\dot{\theta}_{di} = \begin{cases} 0, & \text{if } u_{ix} = u_{iy} = 0 \\ \frac{-u_{iy}}{u_{ix}^2+u_{iy}^2}\dot{u}_{ix} + \frac{u_{ix}}{u_{ix}^2+u_{iy}^2}\dot{u}_{iy}, & \text{otherwise} \end{cases} \tag{4.25c}$$

where β_i and k are positive constant control gains, ensures $(z, \tilde{\theta}_i) = 0$ for all $i \in V^*$ is exponentially stable and that the control objectives hold.

The experiments discussed in the following chapter utilize the single-integrator controllers described in Section 4.2. For these experiments, the single-integrator to nonholonomic transformation was accomplished with two different methods. The first method explicitly transforms the control law within the algorithm script using (4.25) to directly compute velocity inputs for unicycle model. The second method computes the single-integrator control input and then transforms the control input through a function that outputs velocity inputs for the unicycle model. This transformation function is adapted from Robotarium's open-source MATLAB simulator code which follows Equation (11) in [19]. While both methods have been used with success on separate algorithms, it is not clear which method results in better controller performance.

Chapter 5
Centralized Mode Experiments

This chapter describes the application of the control laws presented in Chapter 4 on the TIGER Square testbed. The formation control problems executed in these experiments are mature, well-understood, and have even been tested on the previously mentioned Robotarium platform [1, 17, 18, 20]. Therefore, the purpose of these experiments is to make behavior-based observations and comparisons to expected results. In the following sections, a portion of experimental results are presented through post-processed image frames from the video recording camera. These image frames have been annotated to visualize the graph framework, agent labels, and trajectories. Inter-robot connections with arrowheads represent a directed tracking relationship between the two agents in the direction of the arrowhead. On the other hand, a simple connection (with no arrowhead) indicates, bidirectional tracking, or an undirected relationship.

5.1. Formation Acquisition

The following experiments provide a side-by-side comparison of the undirected and directed graph approach to formation acquisition under different initial conditions. Table 5.1 summarizes the formation acquisition experiments and parameter implementations. The objective in these experiments is to form a triangle.

Table 5.1. Summary of Formation Acquisition Experiments

	Graph	Constraints	Initial Conditions	Number of Agents
1	Undirected	Distance	Hexagon	6
2	Undirected	Distance	Collinear	6
3	Undirected	Distance	Reflected	3
4	Directed	Distance	Hexagon	6
5	Directed	Distance + Area	Collinear	6
6	Directed	Distance + Area	Reflected	3

Figure 5.2 (experiments 1 and 4) shows that in the hexagonal initial conditions, both graph schemes produce similar results. However, the undirected coordination scheme reaches the formation configuration faster than the directed scheme. This behavior is

also observed when comparing the distance error plots of each experiment (Figure 5.1). This behavior can likely be due to the fact that for any undirected connection, both agents are working towards regulating the distance [20]. Experiment 1 uses the undirected control law presented in Equation 4.12, while experiment 4 uses the directed control law presented in Equation 4.14 without the area constraint (denoted by the second term in Equation 4.14c).

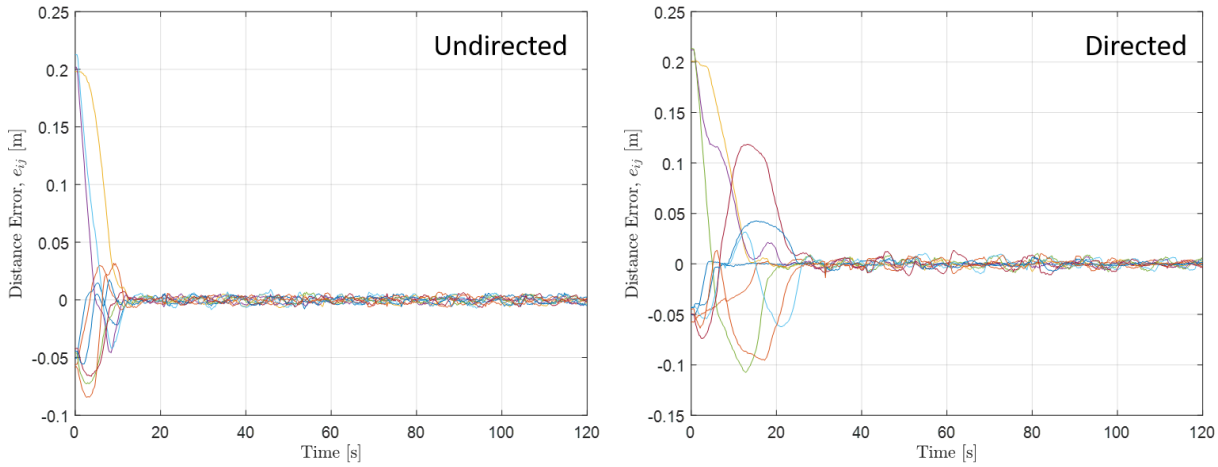

Figure 5.1. Formation Acquisition Distance Errors with Undirected and Directed Graphs. While both sets converge to zero, the undirected scheme reaches the formation shape about 10 seconds faster.

Figure 5.3 (experiments 2 and 5) shows the case where the agents are initialized in collinear positions. Recall that for the undirected, distance-only scheme, if the agents are initially in a collinear formation, they should not be able to leave their initial positions. However, due to measurement error and detection resolution, the physical system converges to an ambiguous framework. Specifically, agent 5 tries to move to a reflected position, which drives agent 2 (and subsequently all other agents) to continuously move left until inevitably colliding with the arena boundaries. Conversely, the directed scheme includes an added area constraint (described in Equation 4.14) that results in successfully converging to the desired formation.

Figure 5.4 (experiments 3 and 6) shows the behavior of the controllers when initialized

in a reflected framework (i.e. the desired framework has agent 2 positioned above agents 1 and 3). Again, the undirected scheme fails to converge to the desired framework, demonstrating the limitations of a distance only controller. Due to space limitations, only three robots were used for these experiments.

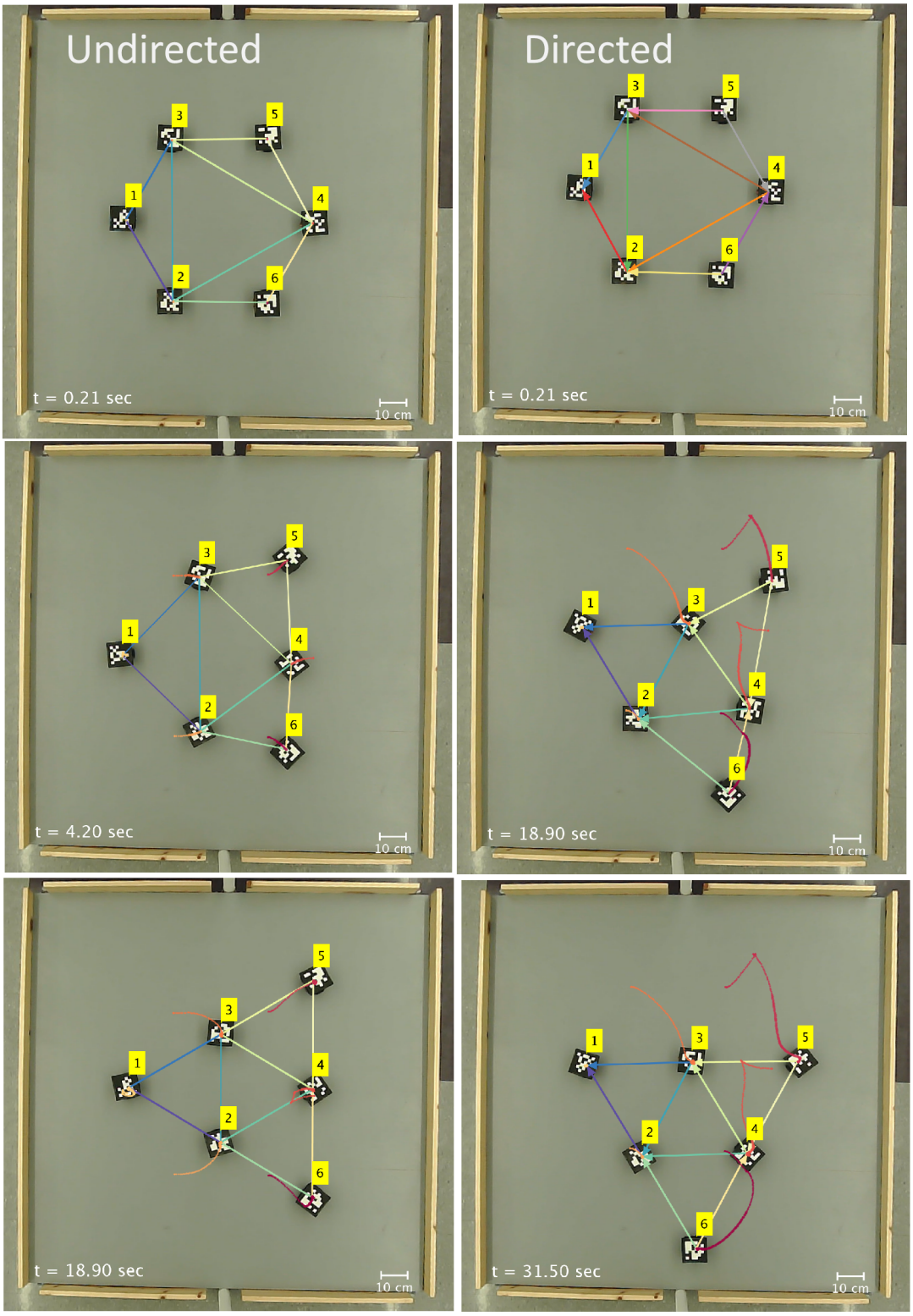

Figure 5.2. Formation Acquisition with Undirected and Directed Graphs. Notice that the directed scheme acquired the formation about 10 seconds after the undirected scheme.

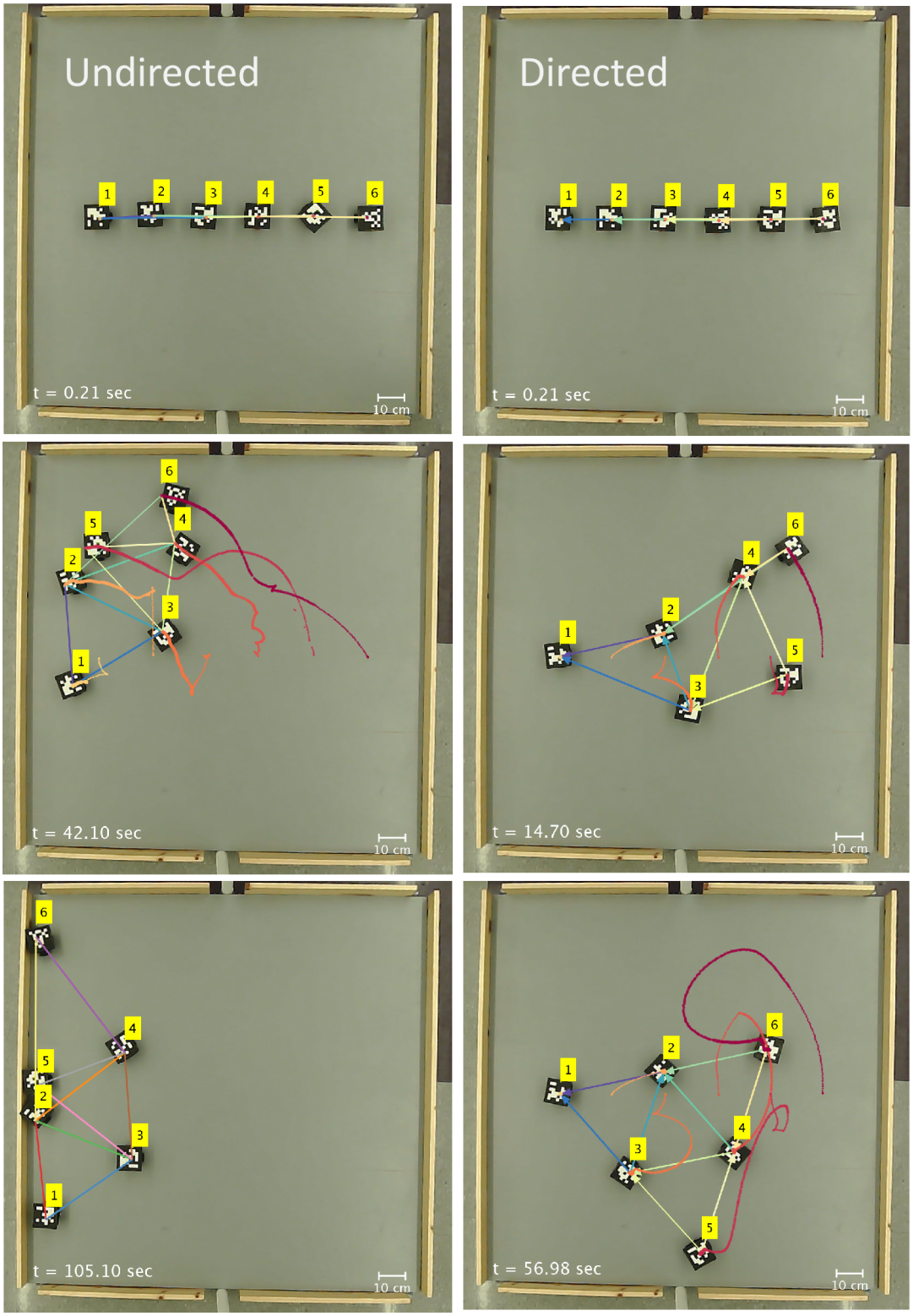

Figure 5.3. Formation Acquisition with Undirected and Directed Graphs from Collinear Initial Positions. While the directed scheme eventually acquires the desired formation, the undirected errors diverge and ultimately reach the boundaries of the arena. Also note that agent 5 is in a "flipped" position.

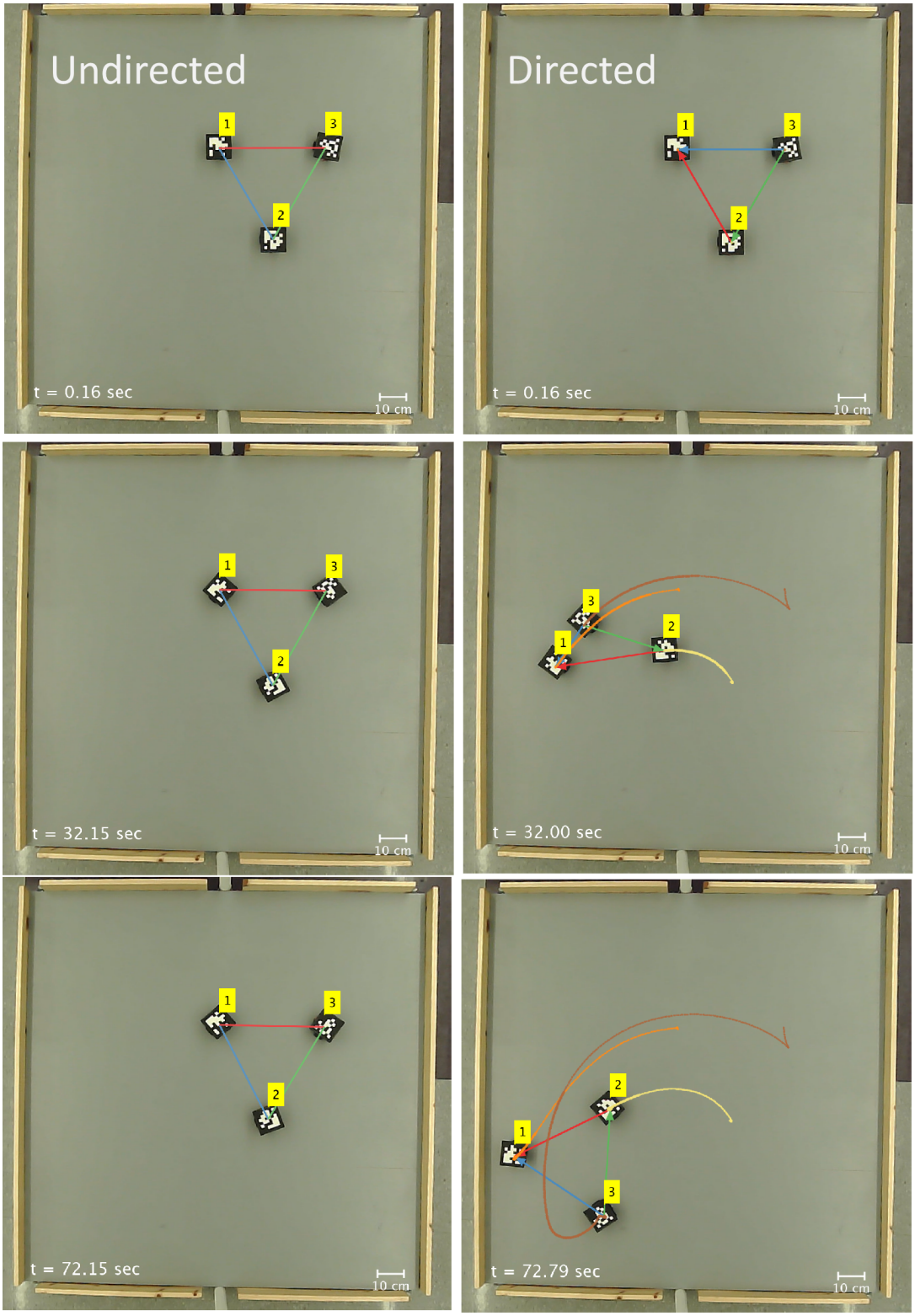

Figure 5.4. Formation Acquisition with Undirected and Directed Graphs from Reflected Initial Positions. The undirected scheme fails to recognize the formation ambiguity and remains in place, while the directed scheme reorients the formation to the desired non-reflected pose.

5.2. Formation Maneuvering

The maneuvering experiments listed in Table 5.2 are intended to compare experimental performance to simulations. When comparing the resulting trajectory and error plots, experiments 1 and 2 both showed a close similarity. In practice, the control gains for the experiments were initially set in simulations and later tuned through trial-and-error on the physical system. Once the "optimal" experiment results were obtained, the simulation was repeated using the experiment control gains. Finally, experiments 3 and 4 demonstrate undirected maneuvering with three and seven robots, respectively. Through the evaluations performed in Chapter 3 and additional stand-alone tests, it is known that the system has the capability to localize and communicate with the 10 collective robots. However, when the control of eight robots was attempted , the movements of the robots were inconsistent and "froze-up." From observation, it is possible that a limitation in communication bandwidth for continuous control of eight robots is the reason behind this behavior.

Table 5.2. Summary of Formation Maneuvering Experiments

	Graph	Constraints	Number of Agents
1	Undirected	Distance	5
2	Directed	Distance + Area	5
3	Undirected	Distance	3
4	Undirected	Distance	7

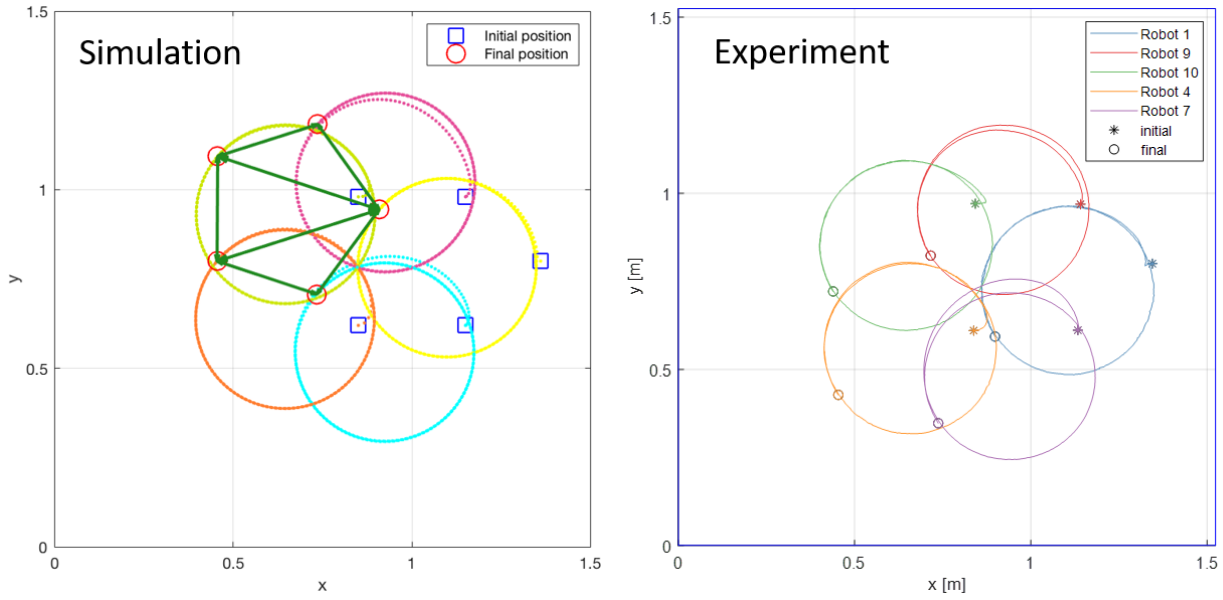

Figure 5.5. Undirected Formation Maneuvering Trajectory, Five Agents.

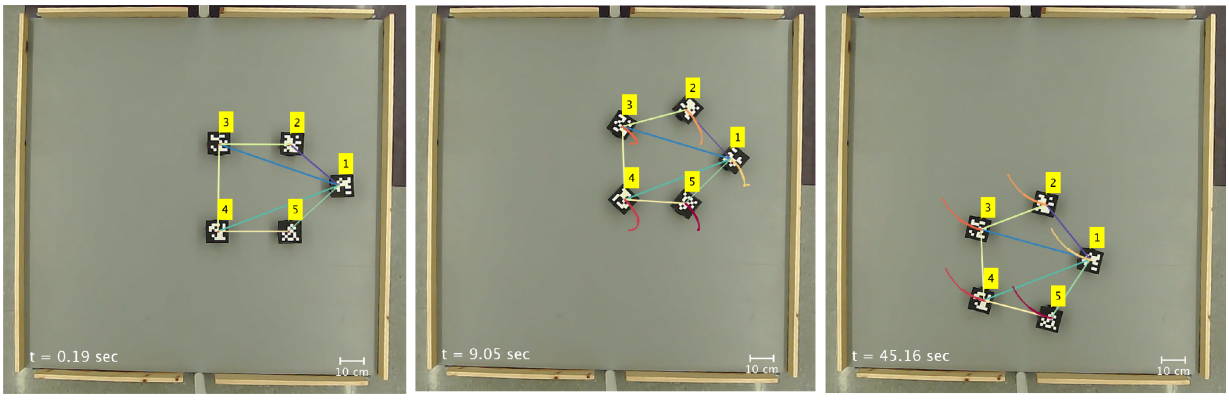

Figure 5.6. Undirected Formation Maneuvering Action Frames. From left to right, the robots begin the experiment from predefined initial conditions and transition into the final acquired formation.

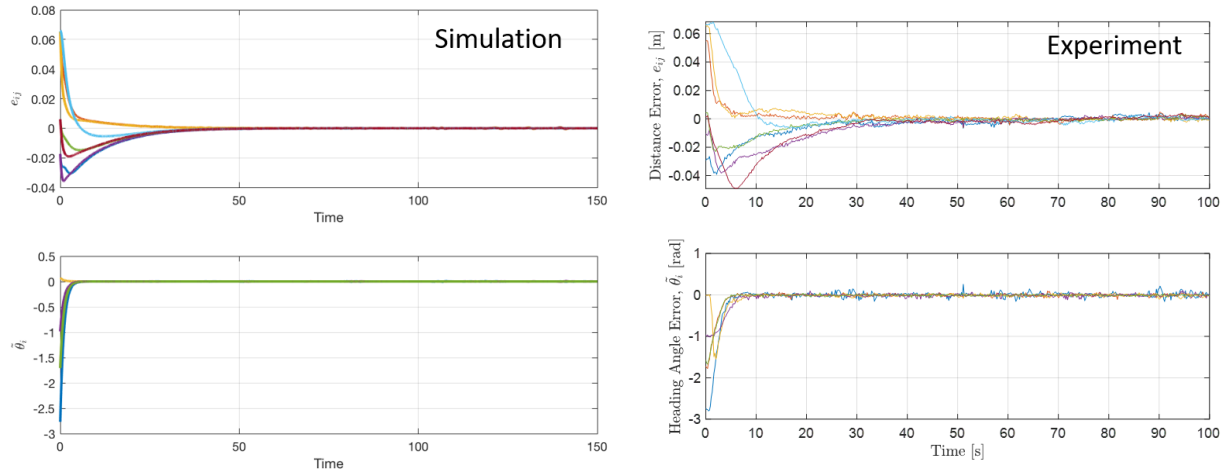

Figure 5.7. Undirected Formation Maneuvering Distance and Heading Error.

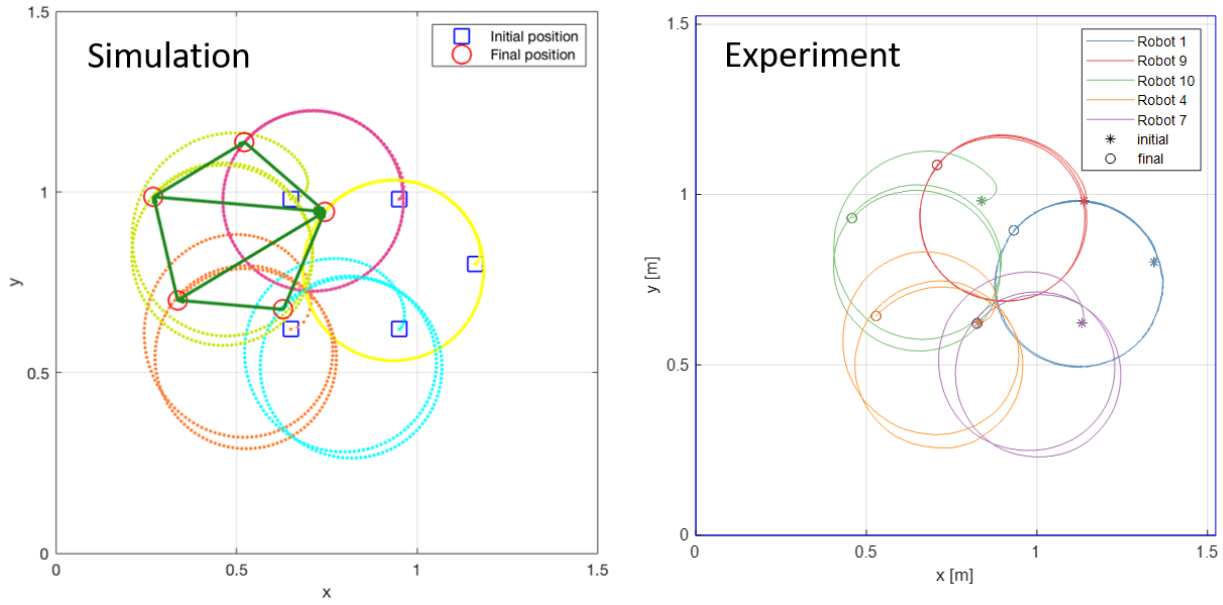

Figure 5.8. Undirected Formation Maneuvering Trajectory, Five Agents.

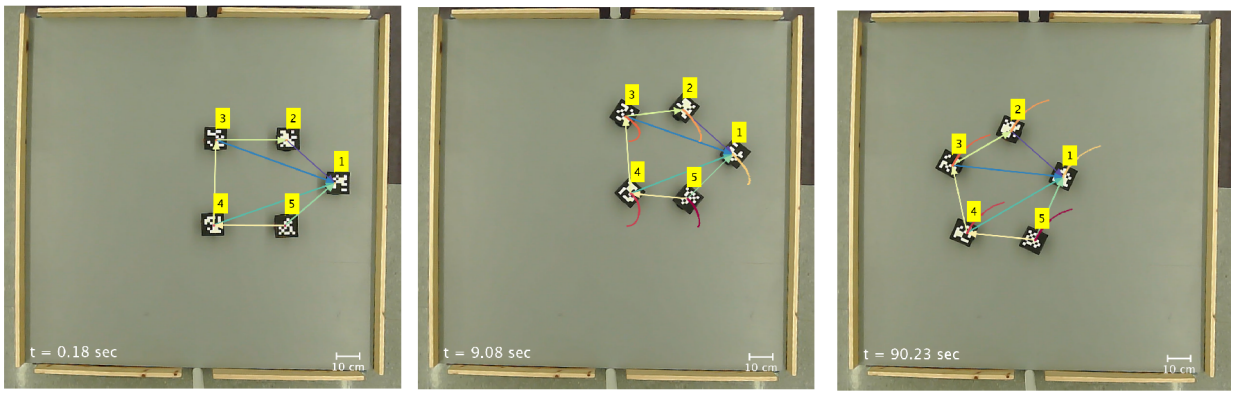

Figure 5.9. Directed Formation Maneuvering Action Frames. From left to right, the robots begin the experiment from predefined initial conditions and transition into the final acquired formation.

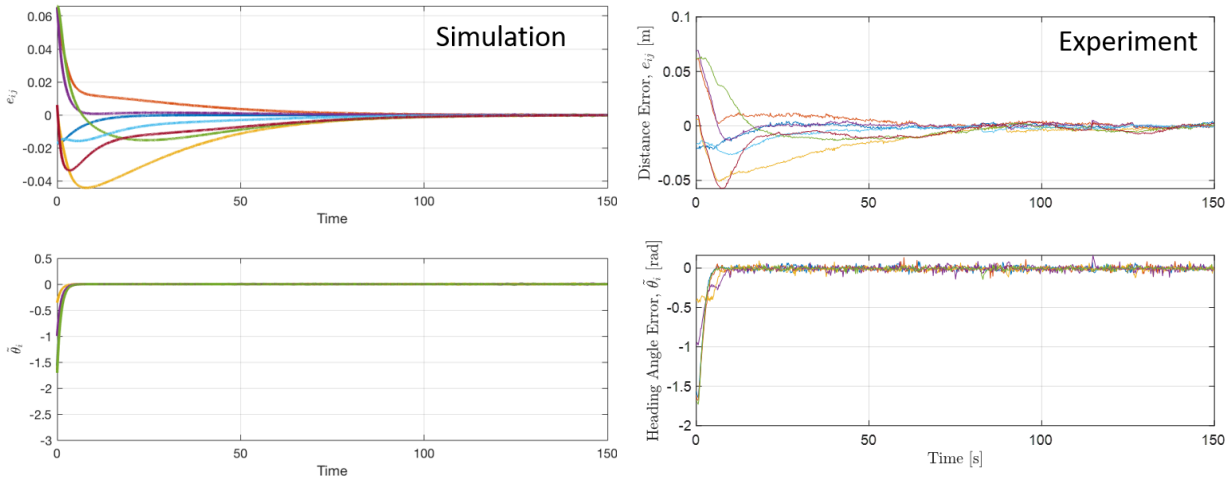

Figure 5.10. Directed Formation Maneuvering Distance and Heading Error.

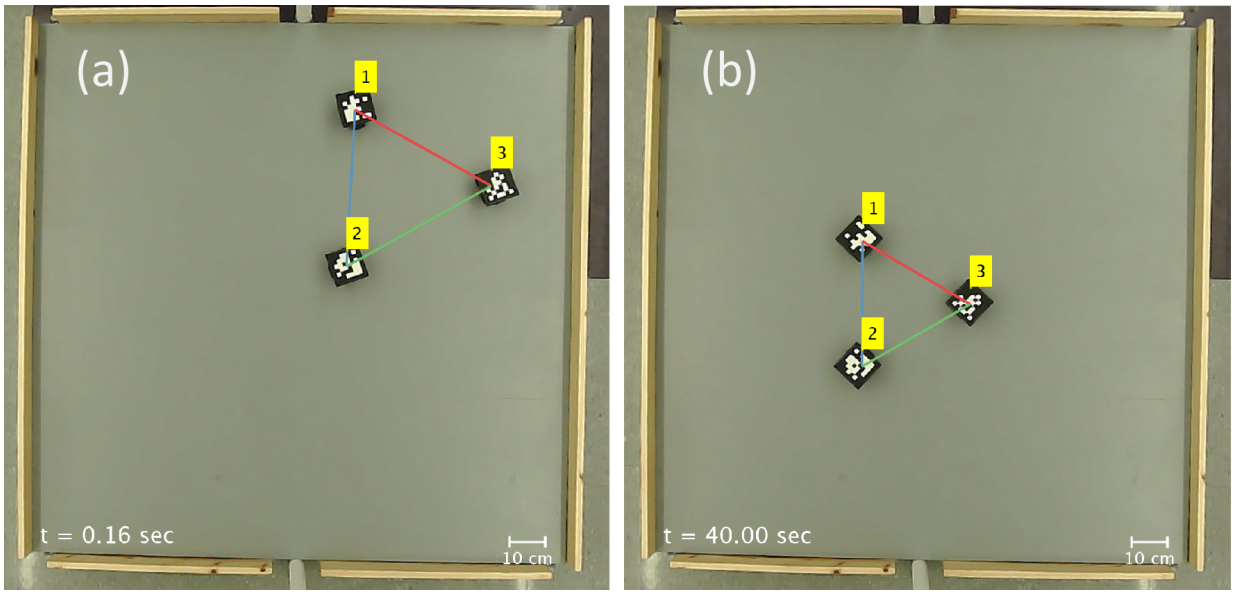

Figure 5.11. Undirected Formation Maneuvering Trajectory, Three Agents.

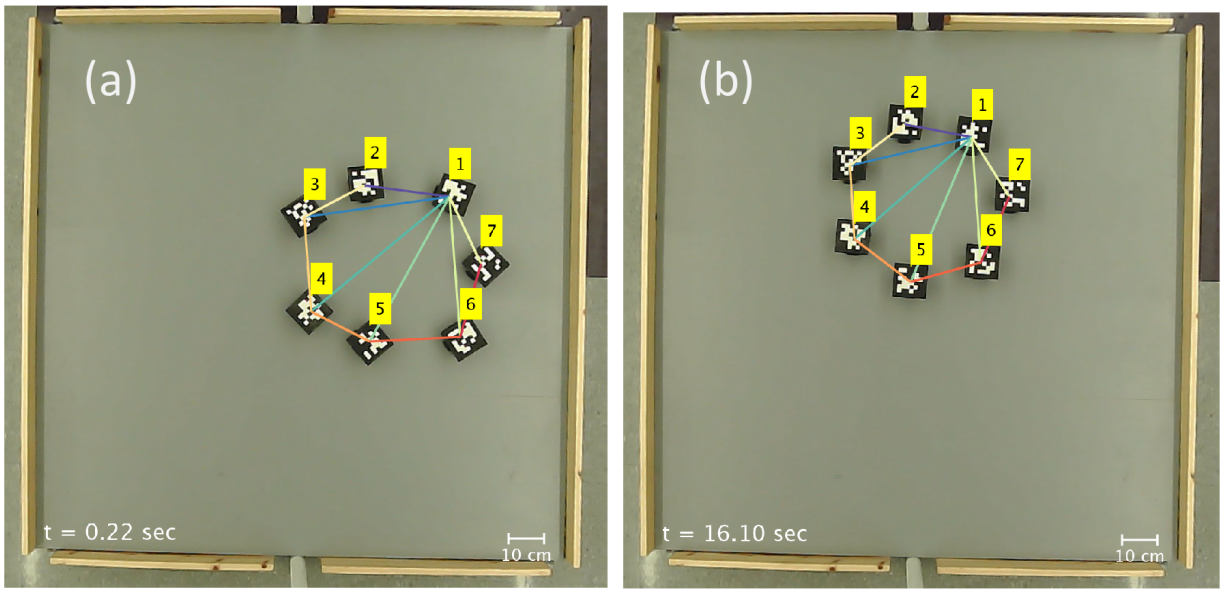

Figure 5.12. Undirected Formation Maneuvering Trajectory, Seven Agents.

Chapter 6
Decentralized Implementation

Up to this point, the testbed has been described in the context of centralized operations. This chapter presents the initial efforts towards decentralizing the testbed and how decentralization will be accomplished moving forward. Section 1.2.2 reviewed modern range and bearing sensing technologies and concluded that infrared (IR) most appropriately satisfies the requirements of real-world collective robots at the centimeter scale. For the purposes of this testbed, the onboard localization sensor design must satisfy the following requirements.

1. Diameter within the current robot foot print (9 x 9 cm).
2. Dynamic range of at least 1 meter.
3. Accuracy better than 15% for both range and bearing.
4. Omnidirectional range and bearing sensing.
5. Refresh rate of at least 30 Hz with 10 robots.
6. Under $100 per unit.

With this in mind, the following sections detail the design of the onboard sensor, sensor characterization test results, and its integration into TIGER Square.

6.1. Sensor Description

Consider an IR receiver is being used to measure the irradiance of a downrange, collinear IR LED. Due to the inverse square law, the typical signal strength acquired from the receiver is relatively high when the LED is placed at a close range, but rapidly drops to near-zero as the LED is moved further away. In other words, as is, the receiver is only dynamically sensitive within a small region. If the signal is amplified to obtain a signal at longer ranges, the output becomes saturated at shorter ranges. Fortunately, the approach presented in [10] introduces a method to increase the dynamic range of an IR receiver without saturating portions of the signal. In this approach, the receiver signal is fed through a cascaded amplifier, where each amplifier stage has a relatively low gain that is tuned to cover a

specific region of the desired range. By doing so, the resulting signal is more linear and the resolution is expanded (see Section 6.2).

For an omnidirectional sensor structure, a set of transceivers (LEDs) and receivers (photodiodes) must be placed along the perimeter of the robot. Thus, the addition of adjacent receivers allows the sensor to triangulate the emitted irradiance and therefore calculate a relative bearing. This triangulation method is the algorithm presented in [21]. Triangulation is accomplished by scanning the photodiodes for the largest signal, r^M, identifying the adjacent signals, r^L and r^R, and applying them to the following equations [21].

$$r = \sqrt{a^2 + b^2} \tag{6.1}$$

$$\theta = \arctan \frac{b}{a} + Q + \phi \tag{6.2}$$

where Q is 2π if θ is negative, else zero, ϕ is the quadrant angle of the photodiode corresponding to r^M, and a and b are defined as

$$a = \frac{r^L + r^R + 2r^M}{2\cos\left(\frac{\pi}{4}\right) + 2} \tag{6.3}$$

$$b = \frac{r^L - r^R}{2\sin\left(\frac{\pi}{4}\right)} \tag{6.4}$$

For full 360° coverage, the placement, beam width, and viewing angle must be carefully selected. In order to minimize resource costs, a conservative amount of LEDs and photodiodes should be used.

The previously mentioned concepts are combined to arrive at the design illustrated in Figure 6.1 and the prototype shown in Figure 6.3. This design utilizes eight photodiodes (±60° half-angle sensitivity) and 16 LEDs (±25° half-angle intensity) for IR sensing. Onboard sensor operations are handled by a Teensy microcontroller that coordinates with the Main Board microcontroller (NodeMCU) through the I^2C bus. Notice that the Teensy drives an onboard multiplexer and LED driver. For IR emitting, a single PWM pin on the

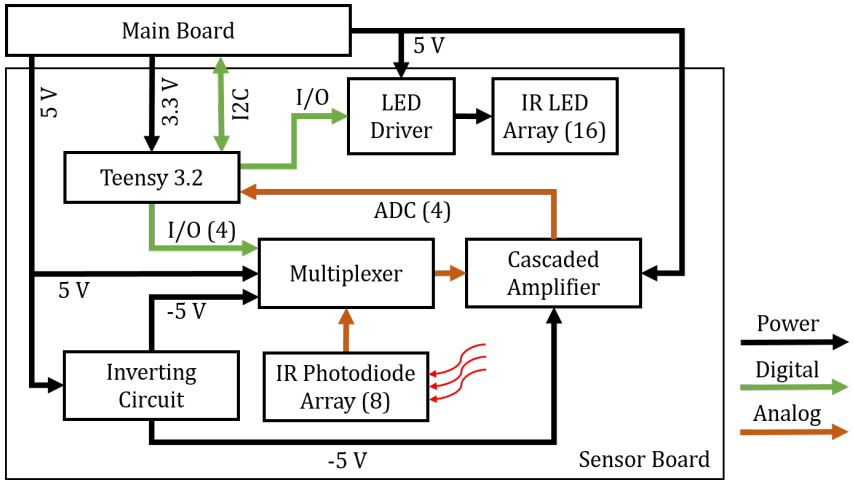

Figure 6.1. Sensor Block Diagram.

Teensy activates a transistor to energize the LED array. For IR receiving, the multiplexer is used to cycle through the available photodiodes, and thereby removes the need to have an amplifier circuit for each photodiode, as shown in Figure 6.2.

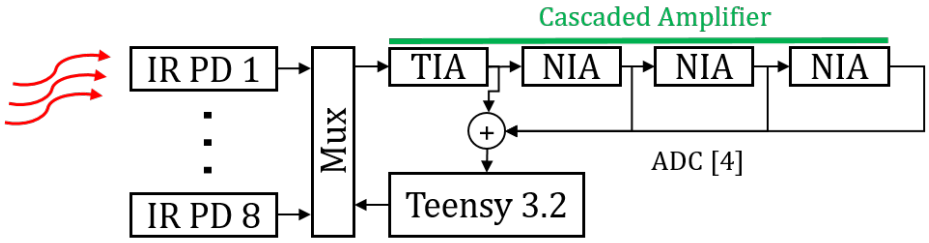

Figure 6.2. Sensor Receiver Block Diagram. Using this architecture, the outputs of eight photodiodes are cycled through a cascaded amplifier to obtain an omnidirectional signal response.

The Sensor Board prototype is shown in Figures 6.3 and 6.4. This prototype has a radius of 4.8 cm and requires $70.35 for each unit. The board schematics and layout are provided in Appendix A. Notice that the cascaded amplifier circuit is on a removable breakout PCB. The cascaded amplifier starts with a transimpedance amplifier (TIA) that feeds into three sequential non-inverting amplifiers (NIA). The TIA has a fixed gain and acts as a pre-amplifier to convert the raw photodiode current to a measurable voltage. From there, the TIA output is fed to the input of the NIA cascade. Each NIA has a 1 kΩ

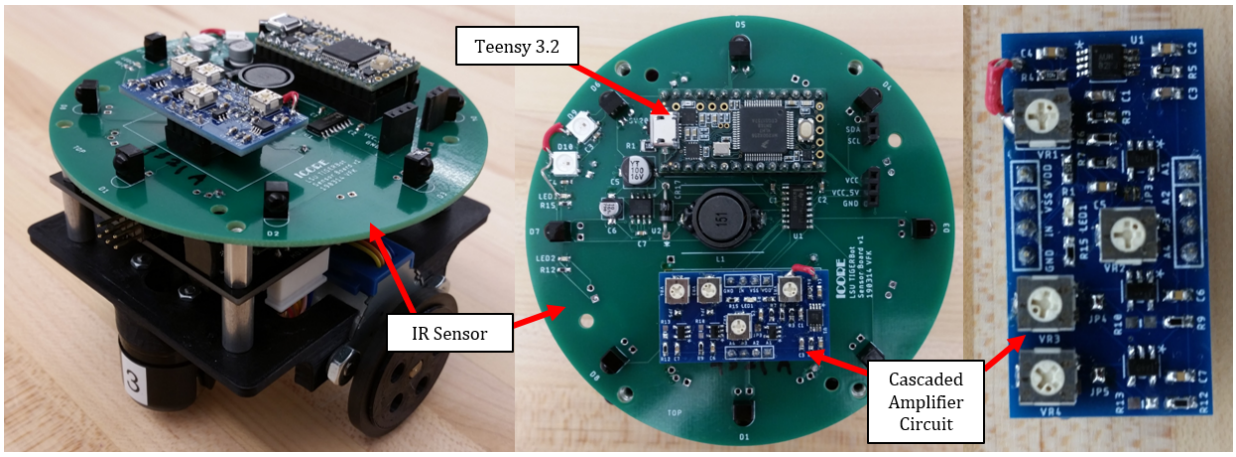

Figure 6.3. Image of Sensor Board. The cascaded amplifier circuit (right) is a removable PCB to allow for easy modifications to the output signal.

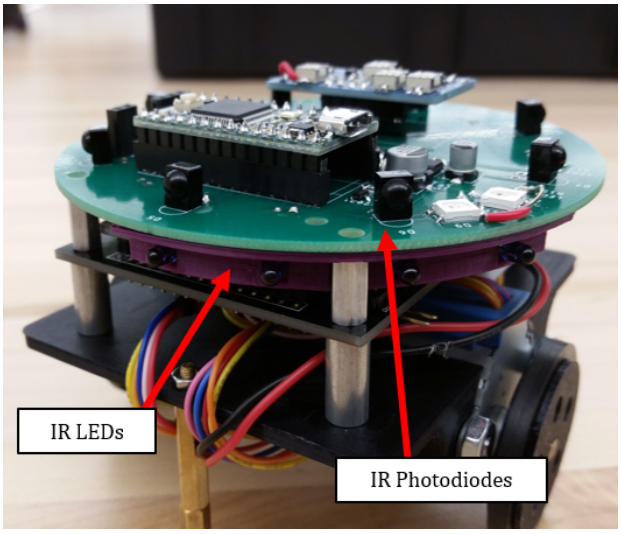

Figure 6.4. Image of LED and Photodiode Array. Here, only four LEDs are installed to demonstrate how an array will be installed. A 3D printed LED frame will be used to keep up to four LEDs properly aligned.

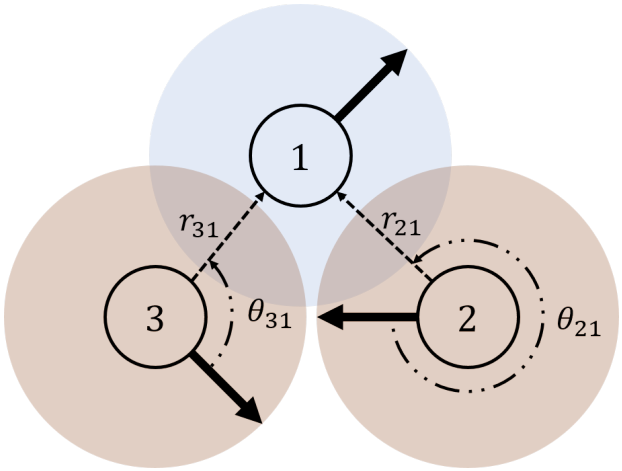

Figure 6.5. Illustration of Onboard Localization Technique. In this example, agent 1 is being localized relative to the nearby agents. The solid arrows represent the heading direction of each respective agent

gain resistor and a tunable feedback resistor. More specifically, the NIA feedback resistor is a nominal 10 kΩ potentiometer in series with a fixed resistor. The fixed resistor can be selected to increase the total gain, or left out by closing a solder jumper. Each NIA output is read out as a measurable voltage. Therefore, four analog voltages are output from the cascade amplifier circuit. In this prototype, each of the 16 IR LEDs is driven with a forward current of 0.1 A, resulting in a maximum power draw of 1.6 A at 5 V (assuming 100% duty cycle). Therefore, two fixed 4.7 Ω, 5 W resistors were used to minimize the number of LED resistors needed to achieve this required power draw for the simultaneously energized LEDs. In hindsight, this decision was not practical for a modular design and carries the risk of burning out LEDs. Thus, the next iteration will distribute the resistance across each LED and will implement a method for a tunable power setting. It is worth mentioning that it is not intended to operate the LEDs at 100% duty cycle. The desired operation of the array is described in the following narrative.

In practice, a turn-based algorithm will be executed to collect the range and bearing of agents relative to each other. For example, in Figure 6.5, three agents are shown. If agent 1 is to be localized, its LED array is briefly activated (illustrated as the blue circle) so that the two nearby agents can sense (orange circles) the emitted irradiance and estimate

a relative range r_{ij} and bearing θ_{ij}. Then, depending on the control or sensing graph, this process is repeated for each of the agents until the required relative values are collected. In order to achieve a localization rate of 30 Hz with a collective of 10 robots and eight photodiodes per robot, the sensor is designed to sample with a frequency of at least 2.4 kHz.

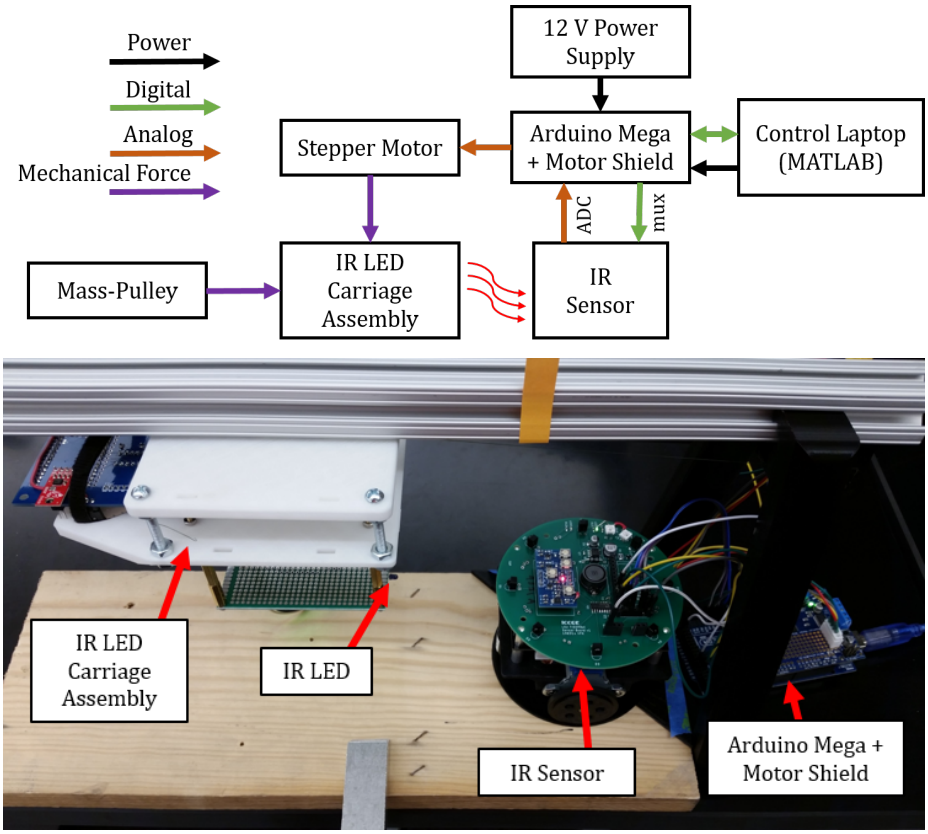

Figure 6.6. Calibration and Testing Fixture. In this setup, the sensor is measuring the irradiance of a single IR led placed on a moving carriage. A stepper motor (controlled with a MATLAB-interfaced Arduino) drives the position of the carriage by reeling a thin wire attached to one side of the carriage base. The connected Arduino also reads in the sensor data and pipes it to a MATLAB workspace for processing.

6.2. Sensor Characterization

6.2.1. Calibration

Before calibrating the sensor, the gains of the cascaded amplifier were selected such that each stage had a small overlap between the range regions. Physically, this meant turning the potentiometer knobs for each amplifier stage. The sensor was calibrated for

a desired range using the fixture shown in Figure 6.6. With this semi-automatic fixture, the signal strength of the amplifier was captured in 0.2 cm intervals up to 90 cm. At each interval, 100 samples for the four stage outputs were collected (i.e. 400 analog signals), averaged, and normalized to produce a single measurement point.

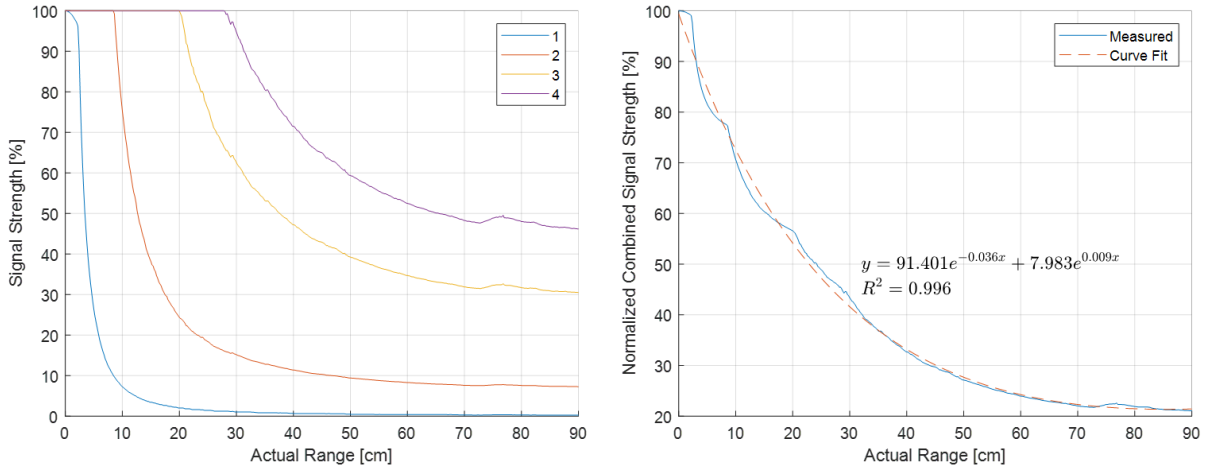

Figure 6.7. Output Response of Sensor Cascaded Amplifier. (Left) The four plotted lines represent the output signal strength of each amplifier stage. When these outputs are summed and normalized (right), a signal response curve fit describing for the full dynamic range is obtained.

Figure 6.7 (left) shows the signal strength curves for each amplifier stage across the full 90 cm range. The four curves were then summed, normalized, and fitted (see right panel of Figure 6.7), to obtain a function describing the global signal strength of a single photodiode on the sensor. Finally, by inverting the axes of Figure 6.7 (right), a calibrated curve fit relating signal strength to measured range is calculated, Figure 6.8. Therefore, provided a normalized signal strength of the sensor, a corresponding range can be estimated.

6.2.2. Range Test

The range accuracy of the sensor was tested using the same test rig presented in Figure 6.6. In this experiment, the carriage was positioned at ten physically measured positions along a 1 meter range, while the sensor position was fixed. For each position, 100 samples were collected and processed to obtain the global signal strength of the sensor. Using the calibration results presented in the previous section, the signal strength results were

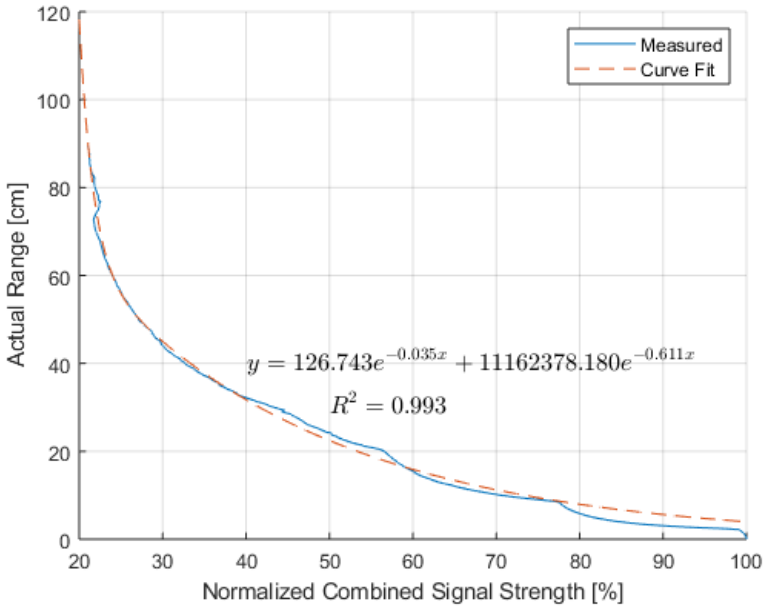

Figure 6.8. Calibration Curve of Sensor Cascaded Amplifier.

converted to a sensor-measured range. The comparison between the actual and measured range are shown in Figure 6.9. The maximum mean deviation was 7.93 cm at 100 cm, and the maximum distance error was 9.82 cm at 100 cm (9.82%).

Note that the previous test used the sensor to measure range using a single diode lined up with the emitting LED. Therefore, when extended to an omnidirectional structure, the signal strength of each of the eight photodiodes can be visualized in Figure 6.10 for different emitting distances.

In the experiment for Figure 6.10, the sensor was placed at a fixed position with an orientation towards 0 ± 2.5 degrees relative to the emitting IR LED. The emitter was then moved to four positions: 7.5, 15, 30, and 60 cm. At each of these positions, the outputs of each diode were sampled 100 times and processed for a single signal strength measurement. This test is simply used to visualize the radial signal strength of the sensor with different ranges.

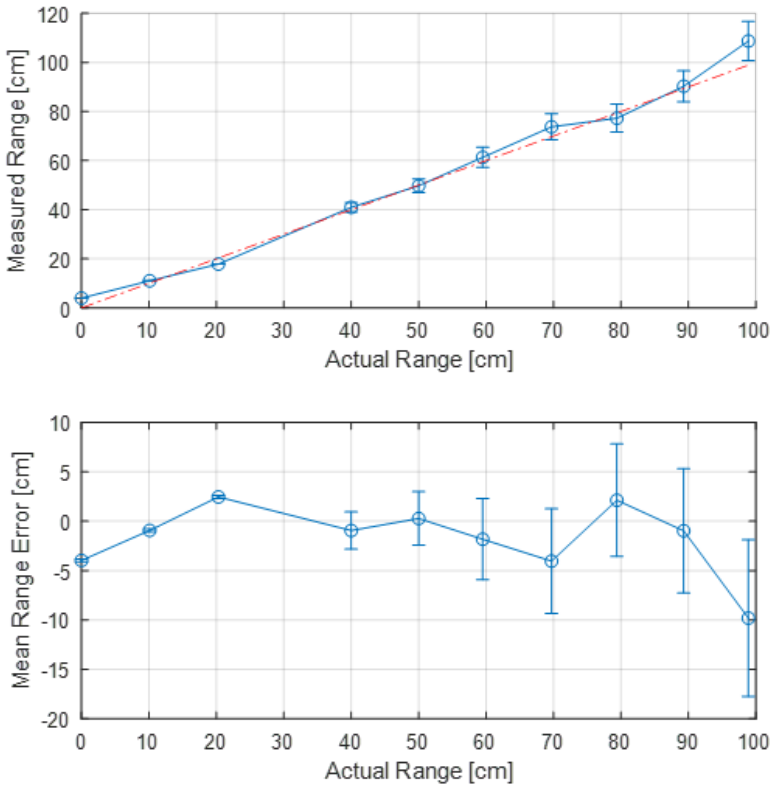

Figure 6.9. Measured Range and Error vs Actual Range. The error bars represent one standard deviation.

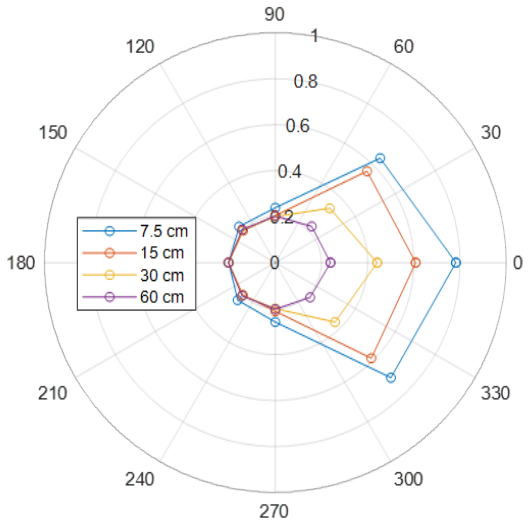

Figure 6.10. Radial Signal Intensity at Varying Ranges.

6.2.3. Bearing Test

To test the bearing measurement accuracy, the LED carriage in Figure 6.6 was held at a fixed position (30 cm away) and the sensor was rotated counter-clockwise from 0 to 180 degrees in 10 ± 2.5 degree increments. At each orientation, 50 samples were collected from the sensor and processed into a single bearing measurement. Figure 6.11 shows the measured versus actual bearing comparison. Note that the measurement error appears to oscillate around zero. This varying behavior is an expected result of the mismatch between the estimated model and actual angular response of the photodiodes [21]. From these results, the maximum observed bearing error is 9.02 at a bearing of 200 degrees.

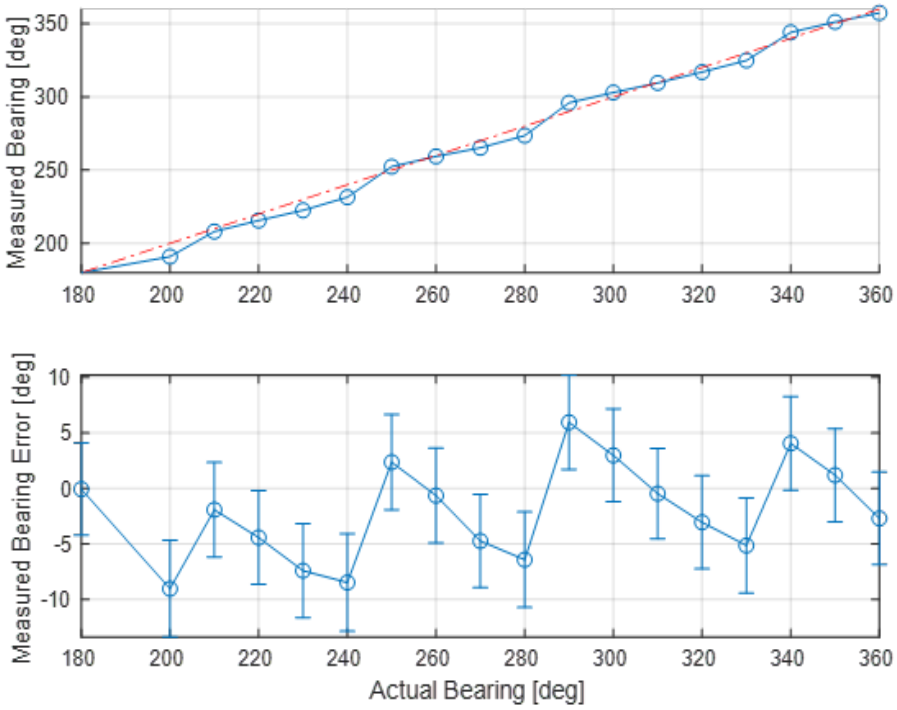

Figure 6.11. Bearing Accuracy Test Results. The measured bearing data points had a consistent standard deviation (not shown) of about 3 degrees. The error bars for the lower plot represent one standard deviation.

6.3. Decentralized Integration to TIGER Square

Understanding that the current hardware version of the Sensor Board is still in its prototyping stage, the following narrative describes the remaining decentralization process. The results in Section 6.2 reveal relatively large errors in range and bearing accuracy. Keep in mind that the calibration was made based on the readings of a single photodiode. Therefore, the resulting calibrated curve cannot be expected to produce identical outputs across all eight photodiodes. Additionally, LED-sensor alignment, LED emission variance, carriage position uncertainty, bearing model estimate mismatch, data acquisition accuracy and resolution, and light reflection all contribute to possible sources of error in the measurement. Moving forward, these issues must be addressed to reach a point where the sensor can produce reliable performance comparable to the centralized localization. It would be ideal to run a full calibration of each photodiode at different angles, but such a characterization may be too dense to be used on robot firmware. For now, the next approach to produce better results will be to improve the testing rig and obtain a single, precisely-obtained calibration curve. In a similar fashion, a solution to accurately regulate the power emission of the broadcasting LEDs must be introduced. This could potentially be done in hardware using a potentiometer or by using hand-selected components that meet the desired specification to give a consistent response.

Figure 6.12 shows a rendering of how the sensor is planned to be hardware-integrated to the existing TIGERBot. In addition to the standoffs, electrical connections will be made between the PCBs through stackable header pins. These connections include power (3.3 and 5 V), ground, and I^2C lines.

In regards to the robot firmware, the Main Board's NodeMCU will be updated to include a set of methods to command the Sensor Board's Teensy. Similarly, the Teensy will be equipped with the firmware necessary to interface with the existing I^2C bus. On the Sensor Board, the Teensy will be programmed to have private methods and macros to drive the functions of the board, such as multiplexing the photodiodes and flashing

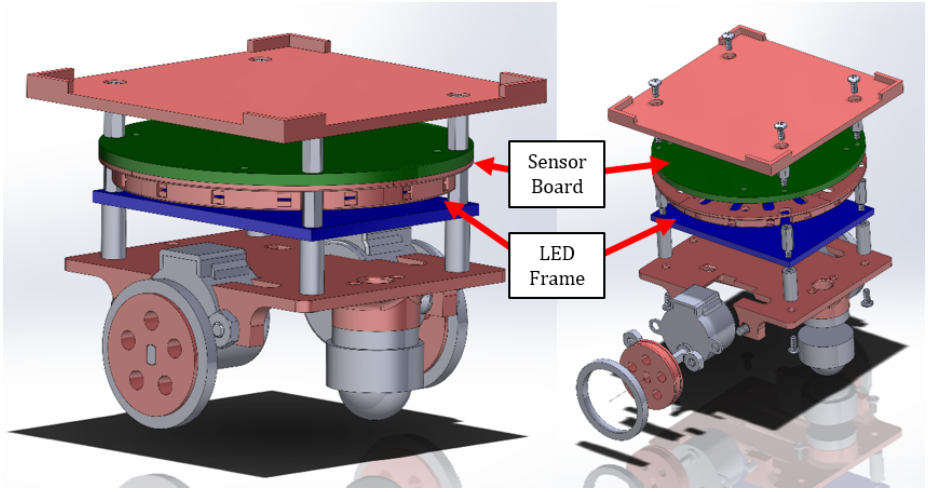

Figure 6.12. TIGERBot Rendering with Sensor Board Add On.

the LEDs. In order to properly implement the turn-based algorithm discussed in Figure 6.5, the robots must synchronize IR LED flashing and photodiode reading. Therefore, a WiFi mesh communication network will be applied when the testbed is in the decentralized operating mode.

Lastly, the TSCS code will be modified to recognize switches in the operating mode. These switches will be manually or automatically executed depending on the experiment objectives.

Chapter 7
Conclusions and Future Work

7.1. Conclusions

In the multi-agent system controls field, the experimental validation of theoretical coordination algorithms is quickly becoming a necessity. To this end, a custom experimental testbed, TIGER Square, has been designed, developed, built, and tested at Louisiana State University. In Chapter 2, the design of the testbed was described. In Chapter 3, the methods and results of system-wide performance evaluations are presented. The analysis of these results can be used to inform expectations regarding the performance and limitations of the testbed during multi-agent coordination experiments. In order to validate the experimental capabilities of the testbed, a set of experiments were executed for two formation control problems: acquisition and maneuvering. In Chapter 4, the background details for the formation control problems were introduced and discussed. Additionally, Chapter 4 laid out the concepts needed to qualitatively interpret the behaviors observed in the subsequent chapter. In Chapter 5, the experimental results of the aforementioned formation control problems were presented. For the formation acquisition experiments (summarized in Table 5.1), the undirected and directed coordination scheme with distance control and distance plus area control, respectively, were compared by varying the initial conditions of each pair of experiments. The outcomes of these acquisition tests correctly demonstrated the limitations of the distance-only scheme, while highlighting the advantages of the distance plus area scheme. For the formation maneuvering experiments (summarized in Table 5.2), the undirected and directed coordination schemes were once again compared when the agents were tasked to translate along a circular trajectory. From this comparison, it is inferred that the directed scheme converges more slowly to the desired formation than the undirected scheme. This conclusion confirmed a similar behavior observed in [20]. Secondly, these maneuvering experiments showed that despite the motor limitations (speed and accuracy) revealed in Section 3.2, the trajectories of the robots closely agreed with those

from the simulated results. In other words, the positional feedback sufficiently accounted for robot pose errors that may arise from the motor limitations. Finally, two maneuvering experiments with three and seven robots are presented to demonstrate the capability of the testbed to effectively control up to seven robots simultaneously.

In Chapter 6, the design and testing results of a prototype range and bearing sensor was reviewed. The sensor, shown in Figure 6.3, utilizes 8 photodiodes and 16 IR LEDs for omnidirectional IR sensing and position broadcasting, respectively. In practice, a turn-based algorithm will be executed so that each robot can be localized relative to the other nearby robots. After setting the gains of the amplifier circuit, a calibration of the sensor provided an exponential function to relate the output signal strength to a range measurement. Range and bearing accuracy experiments were executed to characterize the performance of the sensor. The range test demonstrated measurement errors within ± 5 cm up to an actual range of 90 cm. In other words, the sensor accuracy decreased as the range was increased. On the other hand, the bearing test, shown in Figure 6.11, exhibited an oscillatory error ranging between -10 and 6 degrees. This behavior was attributed to a mismatch in the bearing estimate model and the actual photodiode angular response. Lastly, the remaining tasks to complete the decentralization of TIGER Square were discussed.

7.2. Future Work

As a first generation testbed, the present design has many aspects that can be improved. As previously mentioned, Chapter 3 provided insight into what aspects of the system must be improved for future iterations. In addition to these improvement points, a number of features and refinement ideas are discussed in the following sections. Unsurprisingly, the general theme for these improvements are: upgrading the Control Station, reducing the size of the robots while increasing their capabilities, and expanding the capacity of the testbed.

7.2.1. Optimizing the Control Station

This section summarizes the possible improvement points for the current, centralized testbed design. As such, the topics discussed are ordered from a perceived greatest to least

importance. Therefore, it is obvious that the localization method described in Section 2.1.2 must first be addressed. The testbed localization is a means to an end, so the method chosen and its implications were not prioritized during the development stage. With this in mind, it would be prudent to consider alternatives to the current method. Certainly, effective implementation of computer vision techniques is non-trivial, thus it is recommended to adopt open-source or commercially available localization methods. Referring back to the end of Section 2.1.2, the robots are equipped with AprilTag fiducials, however, the robot detector does not use the AprilTags software. The original AprilTags is implemented in C and later adapted to C++. Then using the MEX file capabilities of MATLAB, it would be possible to call upon the AprilTags source code from the MATLAB environment. Another approach is to eliminate the use of MATLAB for localization and use a separate application that pipes in pose data to the MATLAB workspace. Obviously, it is desired to find a solution that speeds up the localization rate while simultaneously maintaining or improving the current pose detection accuracy. Consequently, the evaluation of alternative methods will require meticulous testing and may not be a cheap endeavor.

Collision avoidance on the testbed is currently accomplished through the use of MATLAB functions adapted from Robotarium's open-source MATLAB simulator code. This collision avoidance algorithm is described in [4]. Through regular use of the testbed, the implemented avoidance algorithm exhibits an inconsistent ability to properly negotiate robot positions in time. Due to the robot movement speed limitations, the occasional "collisions" do not damage the robots, but prove to be a hindrance to the operation. To improve the collision avoidance feature of the control station, it is important to first fully characterize the issue. More specifically, execute tests focused on collision avoidance to understand the occurrence of collisions and what methods can be used to avoid them. Based on these results, a new algorithm must be either developed or adapted into the testbed. Again, this topic is non-trivial and has potential as a long-term project.

Finally, the following points are features that could be implemented to improve the

quality-of-life aspects of the testbed.

- Introduce a file server or repository to unify the access of TIGER Square files.
- Create a MATLAB TIGER Square simulator to enhance the development process and smooth the transition between user-defined algorithms to the TSCS code.
- Adapt the TIGER Square MATLAB code into a Graphical User Interface to facilitate using the testbed.
- Obtain a designated video recording station that automatically synchronizes the start of a video to the experiment

7.2.2. Refining the TIGERBot

The speed of the TIGERBot has been fully tested and discussed in Section 3.2. The alternative, Motor 2, tested in Section 3.2 is a cheap solution to increase the speed of the robots without having to change the hardware or driving method. The transition to these motors would require less than $150 and simply updating the robot firmware to reflect the change in the motor's steps per revolution. Due to their lower efficiency, stepper motors are not often used for wheeled operations, especially those that operate on battery power. Therefore, another consideration is to use DC motors with encoders rather than stepper motors. However, switching to DC motors would require some hardware changes (motor driver, connector, chassis structure), and firmware changes (encoder feedback processing, motor driving methods).

Presently, the onboard IMU is not being used in any capacity. The component has been tested in a rough bench setting on the robots, but not as a feedback sensor, as was originally intended. The necessary code to read the sensor has already been implemented into the robot firmware. However, due to time limitations, the sensor read functions have been omitted and remain unused. It is likely that the sensor would not produce positional feedback that is more accurate than the robot detector, but its value is worth investigating.

In a similar regard, a number of introspective sensors included on the existing version of the TIGERBot are not actively used. Namely, the motor current and temperature sensors

were originally intended to provide status warnings. However, though regular use, it has become apparent that the motors do not reach a state that would induce such warnings. When considering other avenues for improvement on the Main Board, the AVR ATmega chip can be replaced with a more modern microcontroller that has wider capabilities in a smaller package. Finally, the size of the board can be reduced by removing all previously mentioned extraneous components, minimizing the number of breakout components, maximizing the number of surface-mount components, and, of course, restructuring the layout.

7.2.3. Expanding the Capacity of TIGER Square

In this section, the term "capacity" is not limited to the physical size of the testbed. Instead, it refers to a set of topics that would broaden the testbed's capabilities in the sense of experiment variety. Indeed, the first physical change would be to enlarge the arena. However, this change may require up to $500 to fully implement. Additionally, a larger arena would demand changes to the localization method. Specifically, a camera with a wider viewing angle or multiple cameras processed simultaneously. However, both of these options require more processing power, and may result in even slower localization rates. Therefore, the enlargement of the arena should be considered after the localization method issue has been addressed.

Of course, the full decentralization of TIGER Square is a major way to expand the testing capabilities of the testbed. In this work, the initial steps towards decentralization are presented. As expected, these preliminary results demonstrate that there is still a wide margin for improvement in even just the hardware implementation. In this regard, Section 6.3 discusses what tasks remain for the full decentralization of the testbed.

References

[1] M. de Queiroz, X. Cai, and M. Feemster, *Formation Control of Multi-Agent Systems: A Graph Rigidity Approach*. Hoboken, NJ: Wiley, 2019.

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

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

[4] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[5] Y. Tan and Z. yang Zheng, "Research Advance in Swarm Robotics," *Defence Technology*, vol. 9, no. 1, pp. 18–39, 2013.

[6] D. Pickem, M. Lee, and M. Egerstedt, "The GRITSBot in its natural habitat - A multi-robot testbed," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 4062–4067, 2015.

[7] F. Rivard, J. Bisson, F. Michaud, and L. Dominic, "Ultrasonic Relative Positioning for Multi-Robot Systems," in *IEEE International Conference on Robotics and Automation*, pp. 323–328, 2008.

[8] L. Mao, J. Chen, Z. Li, and D. Zhang, "Relative localization method of multiple micro robots based on simple sensors," *International Journal of Advanced Robotic Systems*, vol. 10, pp. 1–9, 2013.

[9] F. Ijaz, H. K. Yang, A. W. Ahmad, and C. Lee, "Indoor Positioning : A Review of Indoor Ultrasonic Positioning systems," in *15th International Conference on Advanced Communications Technology (ICACT)*, pp. 1146–1150, IEEE, 2013.

[10] J. F. Roberts, T. S. Stirling, J. C. Zufferey, and D. Floreano, "2.5D infrared range and bearing system for collective robotics," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 3659–3664, 2009.

[11] J. F. Roberts, T. Stirling, J. C. Zufferey, and D. Floreano, "3-D relative positioning sensor for indoor flying robots," *Autonomous Robots*, vol. 33, no. 1-2, pp. 5–20, 2012.

[12] N. Farrow, J. Klingner, D. Reishus, and N. Correll, "Miniature six-channel range and bearing system: Algorithm, analysis and experimental validation," in *IEEE International Conference on Robotics and Automation*, pp. 6180–6185, IEEE, 2014.

[13] E. Olson, "AprilTag : A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation*, pp. 3400–3407, 2011.

[14] J. Wang and E. Olson, "AprilTag 2 : Efficient and robust fiducial detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4193–4198, 2016.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 06 2016.

[16] B. D. Anderson, Z. Sun, T. Sugie, S. ichi Azuma, and K. Sakurama, "Formation shape control with distance and area constraints," *IFAC Journal of Systems and Control*, vol. 1, pp. 2 – 12, 2017.

[17] T. Liu and M. Khaledyan, "Directed Formation Control of n Planar Agents with Distance and Area Constraints," in *American Control Conference*, 07 2019.

[18] M. Khaledyan and M. de Queiroz, "Translational maneuvering control of nonholonomic kinematic formations: Theory and experiments," pp. 2910–2915, 06 2018.

[19] X. Cai and M. de Queiroz, "Adaptive Rigidity-Based Formation Control for Multi-robotic Vehicles With Dynamics," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3-4, pp. 79–84, 2015.

[20] P. Zhang, M. de Queiroz, M. Khaledyan, and T. Liu, "Control of directed formations using interconnected systems stability," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, 10 2018.

[21] J. Pugh, X. Raemy, C. Favre, R. Falconi, and A. Martinoli, "A fast onboard relative positioning module for multirobot systems," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 151–162, 2009.

Appendix A
Design Documents

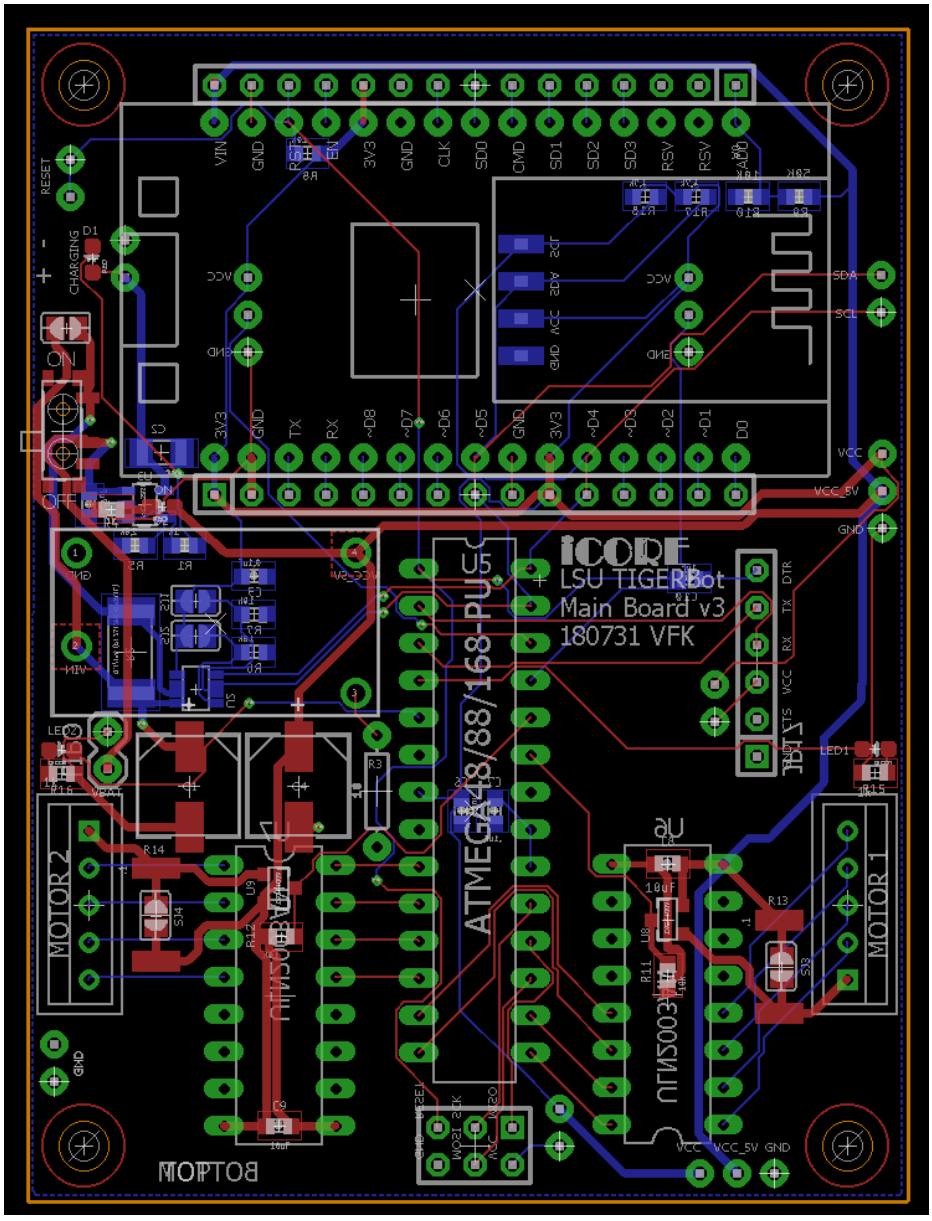

Figure A.1. TIGERBot Main Board PCB Layout.

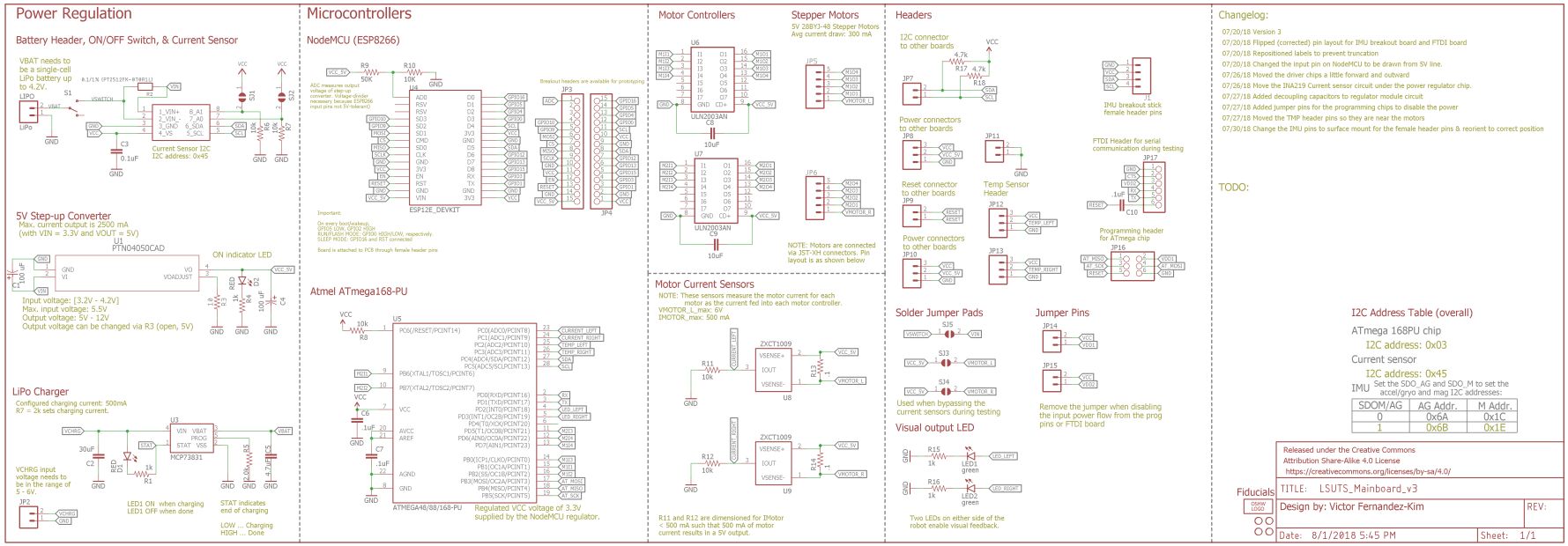

Figure A.2. TIGERBot Main Board Schematic.

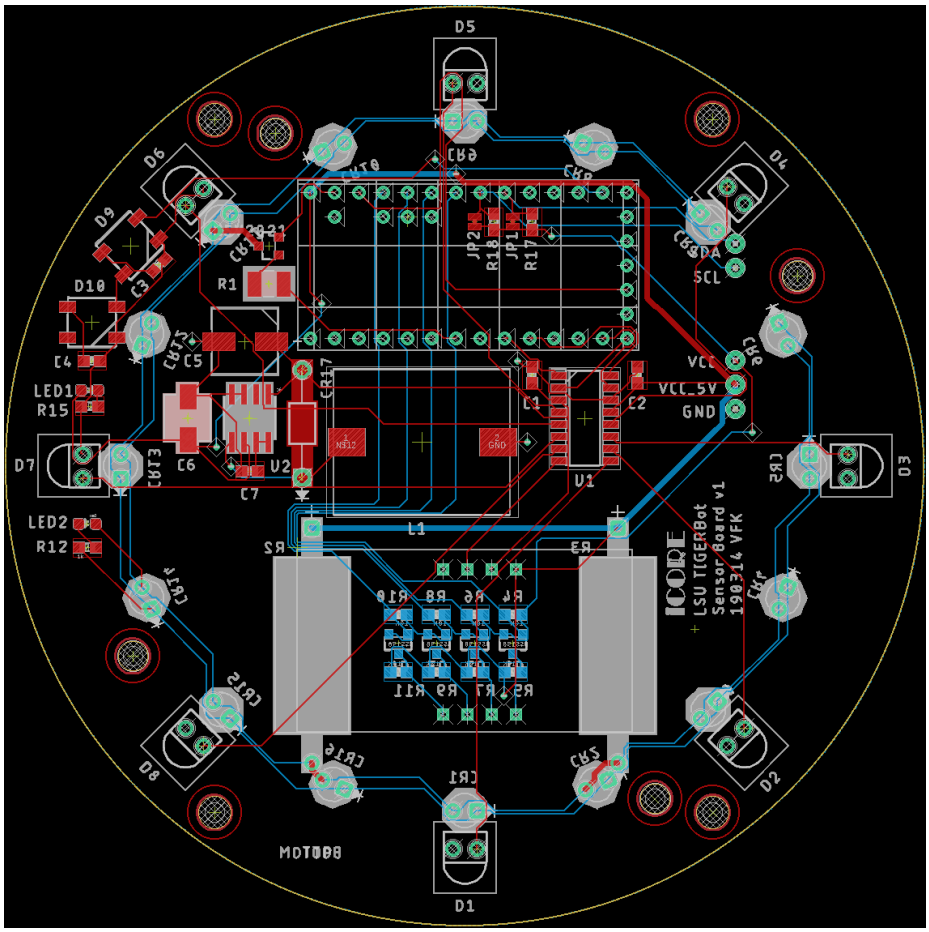

Figure A.3. TIGERBot Sensor Board PCB Layout.

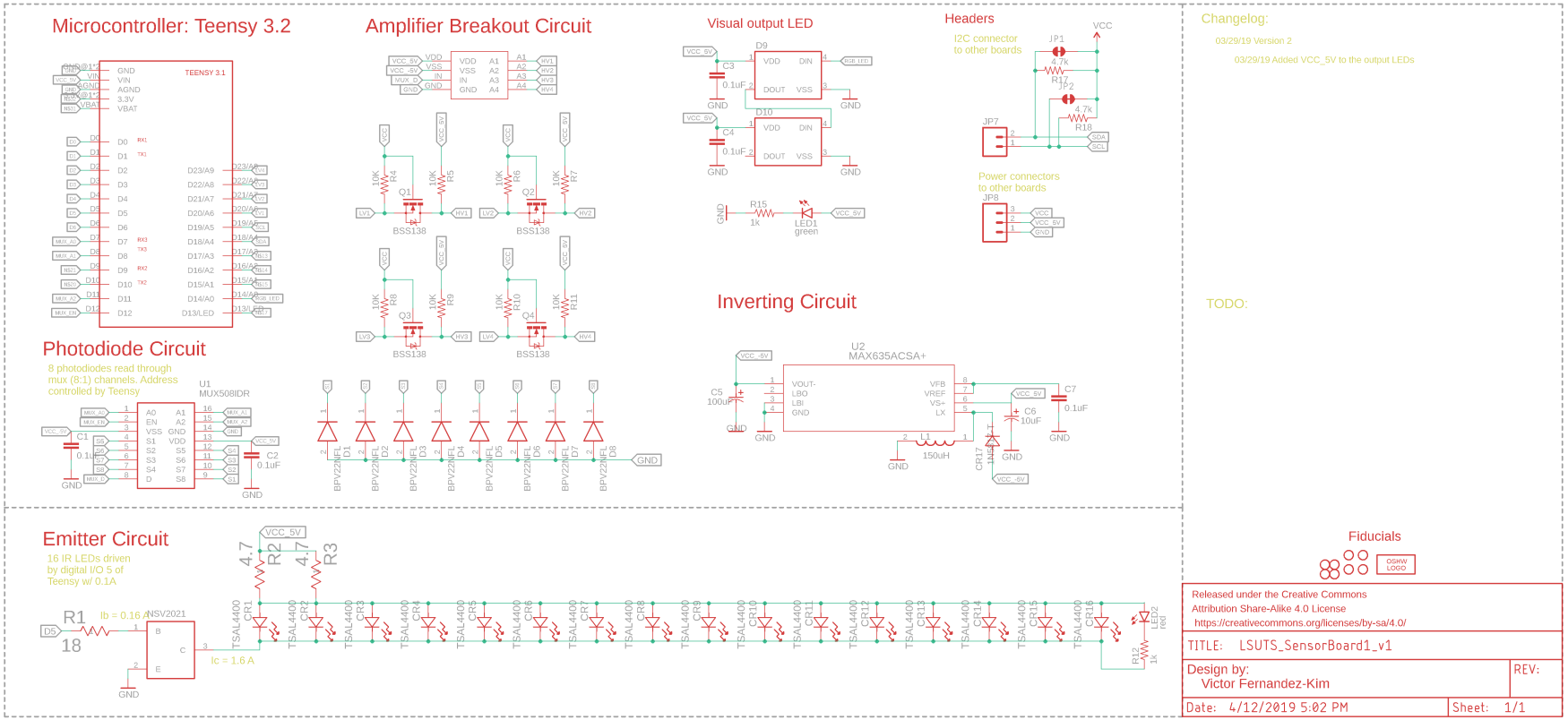

Figure A.4. TIGERBot Sensor Board Schematic.

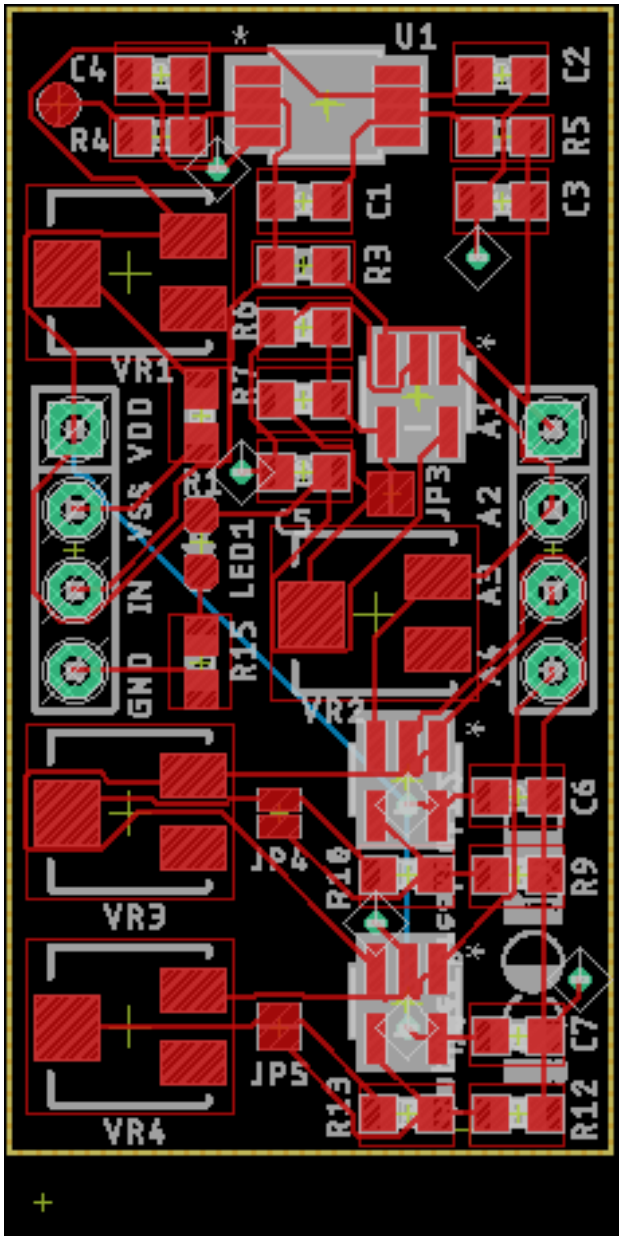

Figure A.5. TIGERBot Cascaded Amplifier Breakout Layout.

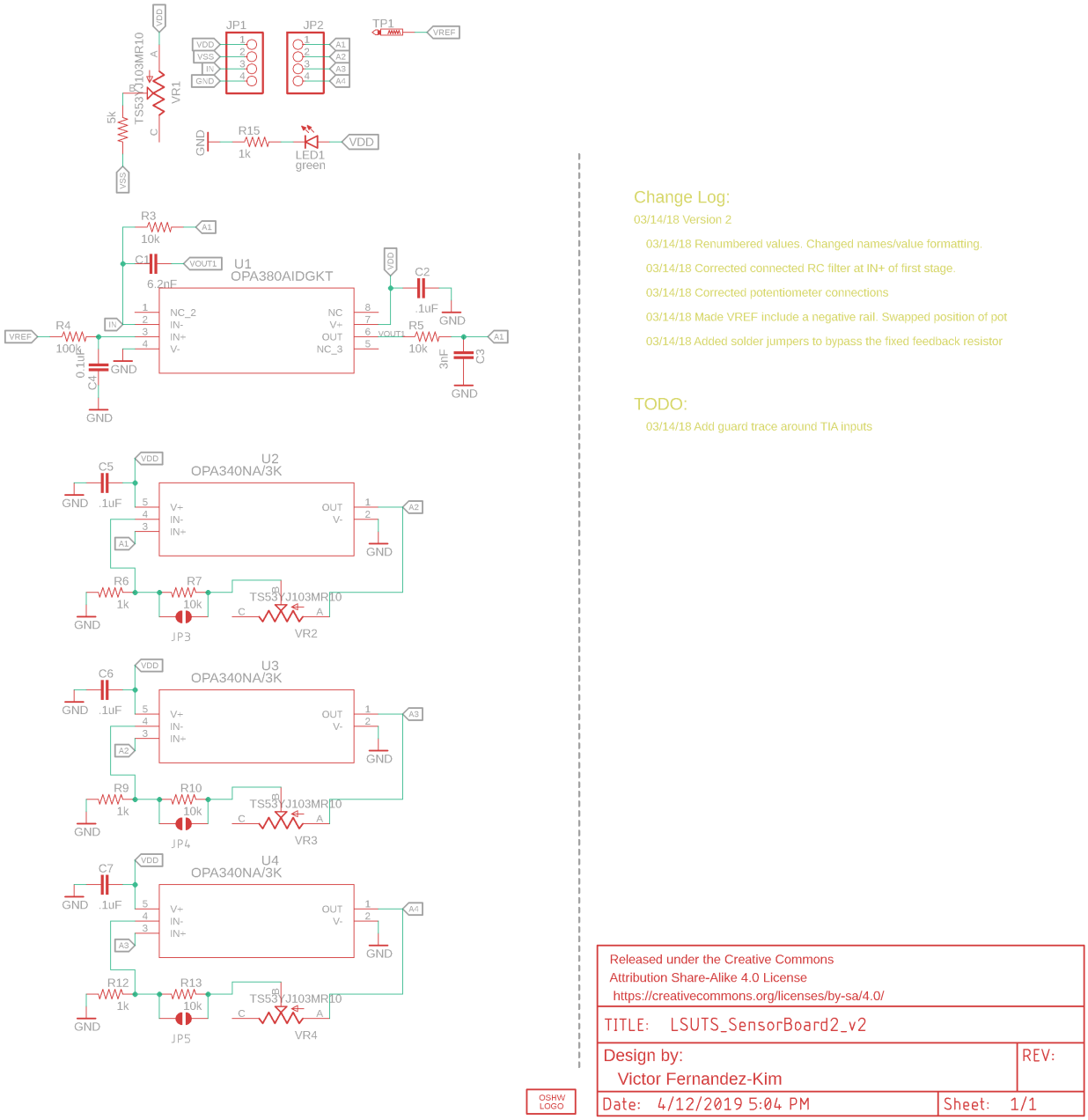

Figure A.6. TIGERBot Cascaded Amplifier Breakout Schematic.

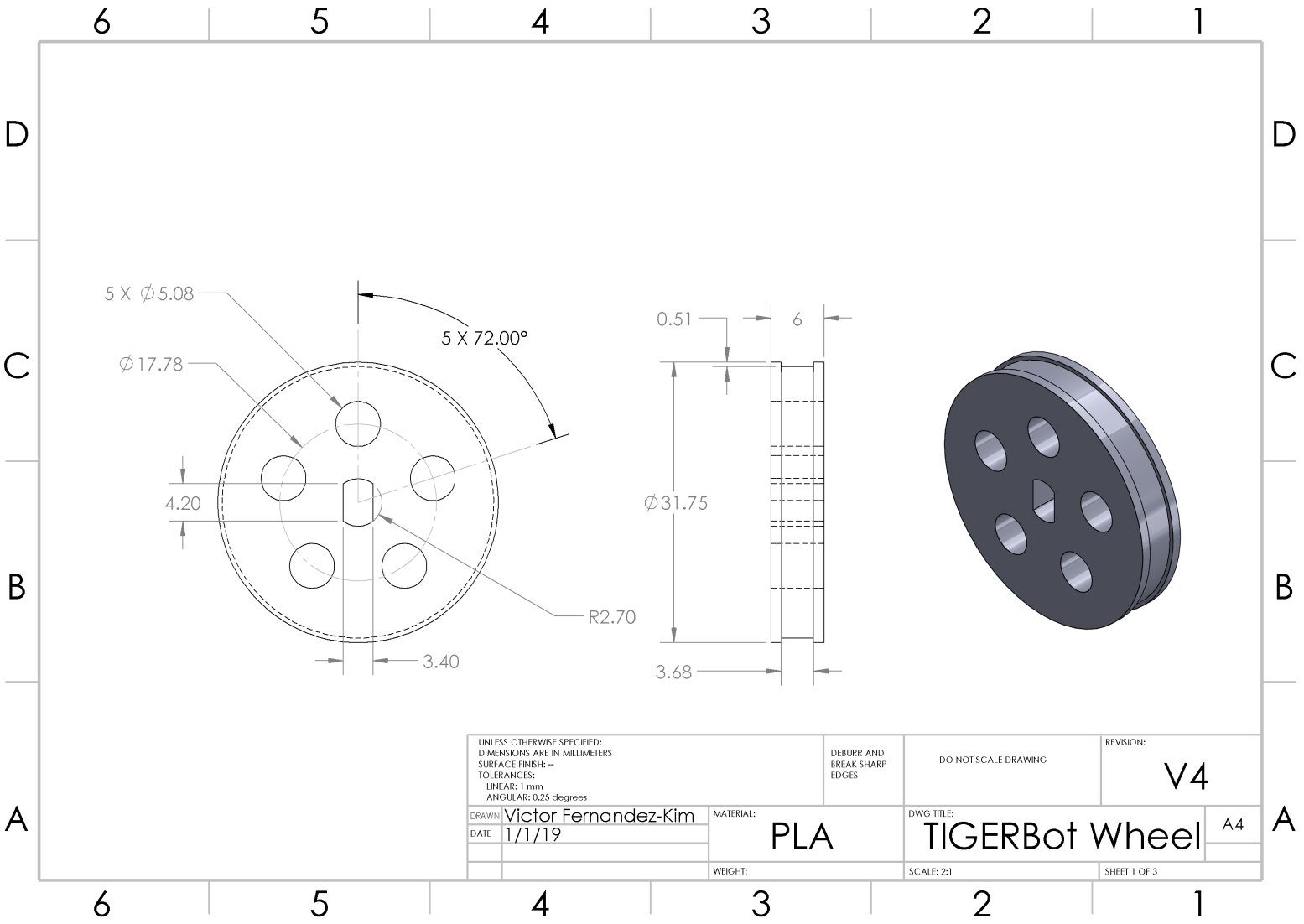

Figure A.7. TIGERBot Wheel Mechanical Drawing.

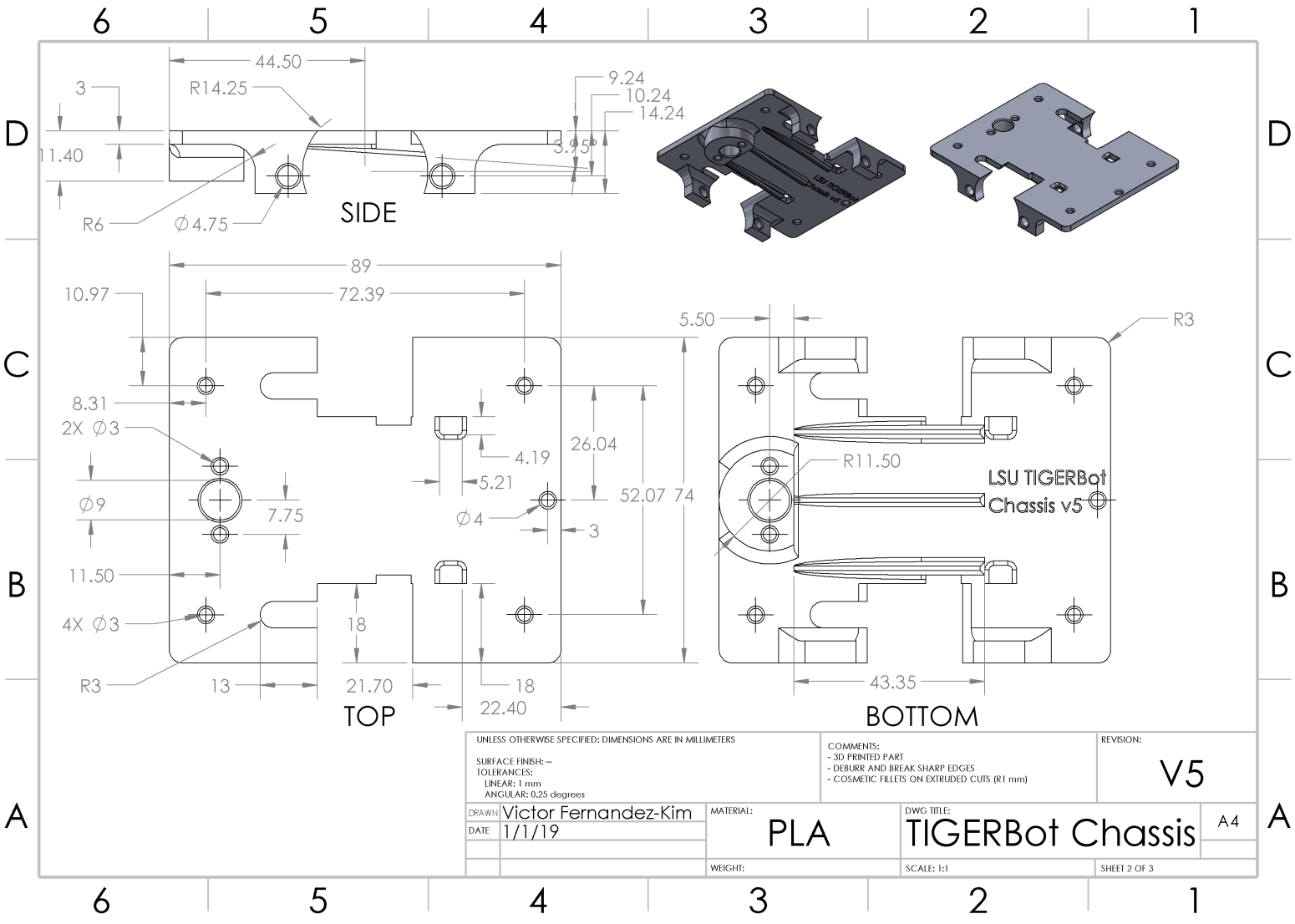

Figure A.8. TIGERBot Chassis Mechanical Drawing.

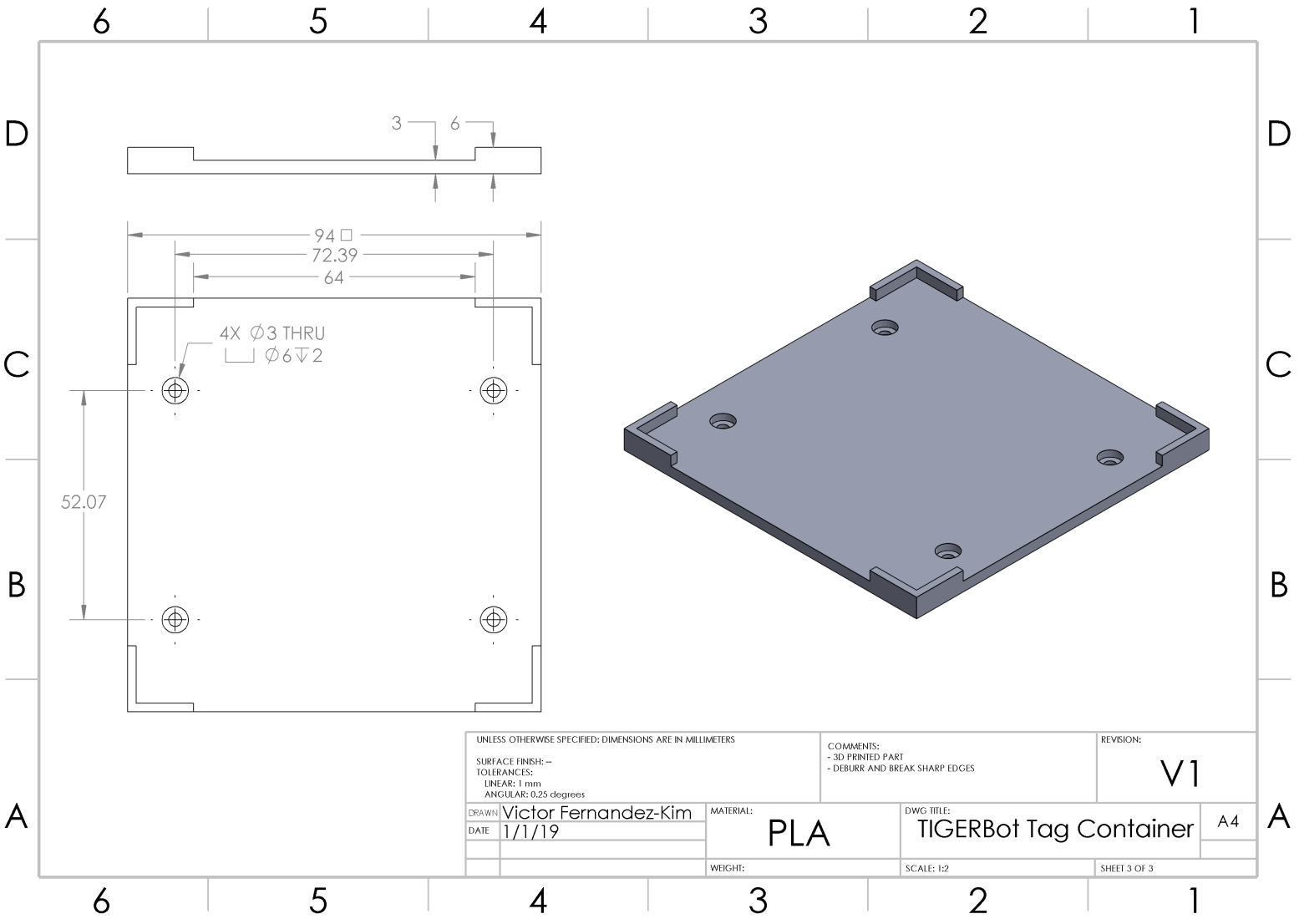

Figure A.9. TIGERBot Tag Container Mechanical Drawing.

Table A.1. TIGERBot Bill of Materials

	Description	Distributor	Catalog Number	Qty/ Robot	Unit Cost	Subtotal/ Robot
1	Pololu Ball Caster with 3/4" Plastic Ball	RobotShop	RB-pol-94	1	$ 3.59	$ 3.59
2	3D Solutech Black PLA Filament	Amazon	–	0.032	$ 17.99	$ 0.58
3	NodeMCU	Amazon	–	1	$ 8.79	$ 8.79
4	0.1" Straight Female Headers (120pcs)	Amazon	–		$ 14.75	$ -
5	0.1" Male Headers (48pcs)	Amazon	–		$ 7.99	$ -
6	(5) 28BYJ48 5 V Motors and Drivers	Amazon	–	0.4	$ 13.99	$ 5.60
7	Square-Profile Oil-Resistant Buna-N O-Rings (50 pcs)	McMaster	4061T24	0.04	$ 8.22	$ 0.33
8	Female Threaded Round Standoff, 1/4" OD, 9/16" Long, 4-40 Thread Size (100 pcs)	McMaster	93330A571	4	$ 0.39	$ 1.56
9	Steel Pan Head Phillips Screws, 4-40 Thread, 1/4" Long (100 pcs)	McMaster	90272A106	0.04	$ 1.46	$ 0.06
10	18-8 Stainless Steel Pan Head Phillips Screw 8-32 Thread, 3/8" Long (100 pcs)	McMaster	91772A192	0.08	$ 6.88	$ 0.55

Table A.1 cont'd

	Description	Distributor	Catalog Number	Qty/ Robot	Unit Cost	Subtotal/ Robot
11	Low-Strength Steel Hex Nut, Zinc-Plated, 8-32 Thread Size (100 pcs)	McMaster	90480A009	0.04	$ 1.60	$ 0.06
12	REG SW 5-15V 2.4A HORZ T/H	Digikey	296-20498-ND	1	$ 16.23	$ 16.23
13	IC MONITOR PWR/CURR BID SOT-23-8	Digikey	296-27898-1-ND	1	$ 2.45	$ 2.45
14	IC CONTROLLR LI-ION 4.2V SOT23-5	Digikey	MCP73831T-2ACI/ OTCT-ND	1	$ 0.58	$ 0.58
15	IC MCU 8BIT 16KB FLASH 28DIP	Digikey	556-ATMEGA168P -20PU	1	$ 1.60	$ 1.60
16	IC CURRENT MONITOR 1% SOT23-3	Digikey	ZXCT1009FCT-ND	2	$ 0.88	$ 1.75
17	CONN IC DIP SOCKET 28POS TIN	Digikey	ED3050-5-ND	1	$ 0.32	$ 0.32
18	CONN IC DIP SOCKET 16POS TIN	Digikey	AE9992-ND	2	$ 0.18	$ 0.35
19	Switch	Digikey	401-2016-1-ND	1	$ 0.91	$ 0.91
20	RES SMD 1K OHM 1% 1/10W 0603	Digikey	P1.00KHCT-ND	4	$ 0.06	$ 0.25
21	RES 0.1 OHM 1% 2W 2512	Digikey	CRM2512-FX- R100ELFCT-ND	3	$ 0.41	$ 1.23
22	RES SMD 2K OHM 1% 1/10W 0603	Digikey	311-2.00KHRCT-ND	1	$ 0.02	$ 0.02

Table A.1 cont'd

	Description	Distributor	Catalog Number	Qty/ Robot	Unit Cost	Subtotal/ Robot
23	RES SMD 10K OHM 1% 1/10W 0603	Digikey	541-10.0KHCT-ND	6	$ 0.02	$ 0.10
24	RES SMD 4.7K OHM 1% 1/10W 0603	Digikey	A106050CT-ND	2	$ 0.02	$ 0.04
25	CAP CER 33UF 25V JB 1206	Digikey	445-11710-1-ND	1	$ 1.37	$ 1.37
26	CAP CER 0.1UF 10V X7R 0603	Digikey	399-1095-1-ND	3	$ 0.01	$ 0.04
27	CAP CER 4.7UF 25V X5R 0603	Digikey	1276-1900-1-ND	1	$ 0.26	$ 0.26
28	CAP ALUM 100UF 20% 16V SMD	Digikey	493-2105-1-ND	2	$ 0.23	$ 0.46
29	CAP CER 10UF 6.3V X5R 0603	Digikey	490-7316-1-ND	2	$ 0.14	$ 0.27
30	JST-PH Male Header Conn	Digikey	455-1719-ND	1	$ 0.16	$ 0.16
31	CONN HEADER XH TOP 5POS 2.5MM	Digikey	455-2270-ND	2	$ 0.24	$ 0.47
32	BATTERY LITHIUM 3.7V 2AH	Digikey	1528-1857-ND	1	$ 12.50	$ 12.50
33	LED RED CLEAR 0603 SMD	Digikey	350-2029-1-ND	2	$ 0.33	$ 0.65
34	LED YELLOW-GRN CLEAR 0603 SMD	Digikey	350-2034-1-ND	2	$ 0.33	$ 0.65
35	9DOF SENSOR STICK	Digikey	1568-1424-ND	1	$ 14.95	$ 14.95
	Total per Robot					$ 78.73

Table A.2. TIGER Square Arena Bill of Materials

	Description	Distributor	Catalog Number	Qty	Unit Cost	Subtotal
1	Heavy Duty Leg Leveler	Amazon		1	$ 24.99	$ 24.99
2	1.25” x 5’ SCHE 40 PVC Pipe	Lowes	23982	1	$ 3.93	$ 3.93
3	1” x 5’ SCH 40 PVC Pipe	Lowes	23977	3	$ 3.37	$ 10.11
4	1” SCH 40 Elbow	Lowes	22706	2	$ 1.28	$ 2.56
5	1” SCH 40 Adapter	Lowes	23864	2	$ 0.77	$ 1.54
6	1” Galv Split Ring Hanger	Lowes	302038	2	$ 1.87	$ 3.74
7	8 oz Oatey Handy Pack PVC cement	Lowes	150887	1	$ 8.98	$ 8.98
8	1” x 0.5” SCH 40 Tee	Lowes	188243	1	$ 1.78	$ 1.78
9	2” X 2” X 1/8” X 20’ A-36 ANGLES	Brecheen Pipe & Steel	2218	2	$ 18.00	$ 36.00
10	1/8” X 8” X 20’ HR STRIP	Brecheen Pipe & Steel	188	1	$ 41.00	$ 41.00
11	White Metal Primer	Lowes	99059	1	$ 3.98	$ 3.98
12	1” x 6’ x 10’ Whitewood Board	Lowes	952	2	$ 11.37	$ 22.74
13	Birch Russian 18mm 5’ x 5’	Brazos	815-315813-00	1	$ 25.78	$ 25.78
14	3/8 ” Galv Washers 25 ct	Lowes	41706	1	$ 5.51	$ 5.51

Table A.2 cont’d

	Description	Distributor	Catalog Number	Qty	Unit Cost	Subtotal
15	Painters Tape	Lowes	957863	1	$ 9.48	$ 9.48
16	3M Paint Respirators	Lowes	39277	1	$ 5.97	$ 5.97
17	6 mL Threadlocker Blue	Lowes	42539	1	$ 6.58	$ 6.58
18	Phillips #3 Driver Bit 2	Lowes	567355	1	$ 2.03	$ 2.03
19	4 fl oz Gorilla Glue	Lowes	116178	1	$ 6.58	$ 6.58
20	3/8” - 16 x 1” Bolt	Lowes	61907	4	$ 0.46	$ 1.84
21	3/8” -16 Nut	Lowes	67341	4	$ 0.23	$ 0.92
22	Rust-Oleum High Performance 2-Part Gray Gloss Garage Floor Epoxy Kit	Lowes	16697	1	$ 45.54	$ 45.54
23	#8 Washers 36 ct	Lowes	58122	1	$ 1.28	$ 1.28
24	# 10 Washers 24 ct	Lowes	58123	1	$ 1.28	$ 1.28
25	8-32 x 1.5” Machine Screws	Lowes	67797	1	$ 5.78	$ 5.78
26	10-32 x 1” Machine Screws	Lowes	70052	1	$ 5.78	$ 5.78
27	Routing Clamp 1 1/4”	McMaster	3039T17	2	2.09	4.18
	Total					$ 289.88

Appendix B
User Manual

B.1. Running Experiments

B.1.1. Setting Up

Before running an experiment, a few steps must be taken to ensure the system is properly initialized. A summary of these steps are as follows

1. Connect the router ethernet cable, power cable, and USB webcam cable to the control laptop. These connection ports are all found on the left side of the control laptop (Figure B.1)
2. Turn on the WiFi router.
3. Open MATLAB on the control laptop.
4. Open and run the script "addPaths.m". This file is located in the root directory of TIGERSquare.
5. Open the script "runExperiment.m" and make changes to the object configuration, robot numbers, or algorithm script as necessary.
6. If video recording will be used, plug in the second webcam cable to an secondary computer and open the recording software.

When the router is turned on and the ethernet cable is plugged in, a web page may appear. This is normal and may simply be exited. When running experiments, be sure to keep the power cable connected. The processing speed is slightly faster when the power cable is connected. There are two webcam cables corresponding to the primary (localization) camera and secondary (video recording) camera (Figure B.2). Be sure that it is the primary camera connected to the control laptop.

The "addPaths.m" script simply adds all the relevant file paths in that the MATLAB control software will use. This script must be run before any TIGER Square operation, including video processing. Directories can be appended in the future if necessary. As shown in Figure B.3, "runExperiment.m" is the top-level script used in any experiment

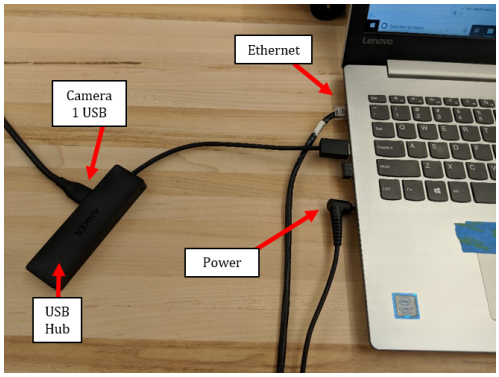

Figure B.1. Control Laptop Connections.

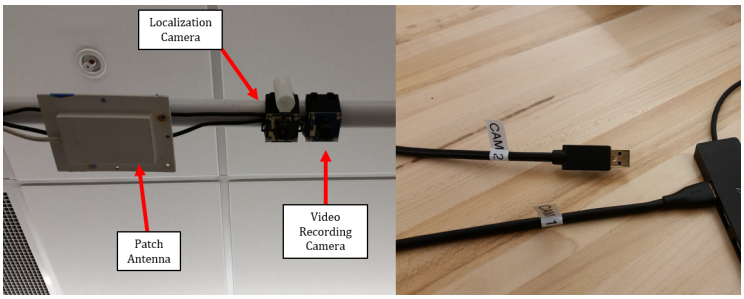

Figure B.2. Camera Connections.

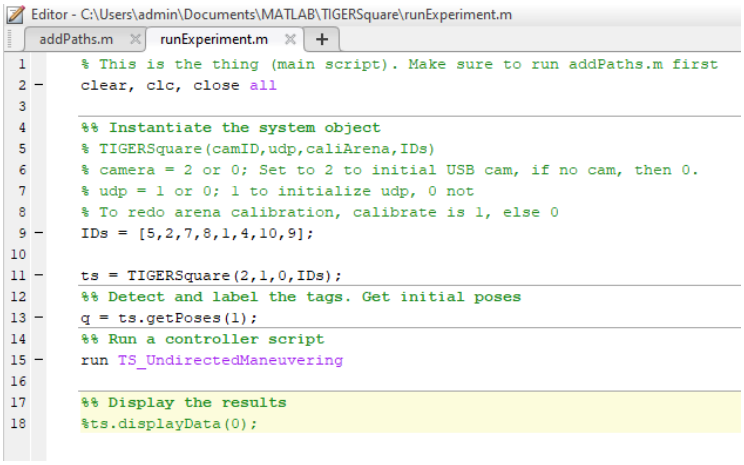

Figure B.3. Screenshot of runExperiment Script.

run on TIGER Square. From this script the user can configure (1) the robots used in an experiment, (2) the TIGERSquare object settings (e.g. initialize camera or router), and (3) the algorithm being executed.

B.1.2. Importing User Algorithms

In line 15 of the "runExperiment.m" script the algorithm to be executed is called by the file name. Technically, the algorithm file can be placed anywhere within the path domains defined by "addPaths.m", but it is recommended to place the algorithm file in the "algorithms" folder of TIGERSquare. Make certain that the user-defined algorithms follow the syntax provided in the "algorithmSkeleton.m" script. This script provides an example of how the algorithm should be structured and what TIGER Square functions are available.

B.1.3. During an Experiment

When running an experiment, always execute from the "runExperiment.m" script. This will reinitialize the TIGERSquare object and avoid overlapping data sets. Once executed, the TIGERSquare object will be instantiated based on the configuration settings defined in the constructor. This process may take up to a minute. Once complete, the program will move on to running the algorithm. During an experiment, it may be possible that detection error messages appear in the MATLAB command window. These are indications

that the robot detector has failed to accurately localize the position of a specific robot at a specific experiment iteration. These messages are non-critical and will not halt an execution. However, the frequency of these messages may be an indication of some larger issues (e.g. robot is not in the scene, two robots are frequently being mixed).

Currently, the collision avoidance algorithm occasionally fails. Due to the slower robot movement speeds, this is not a critical issue, but may result in 2 or more robots "stuck" in place as they cannot negotiate around their positions. In this case, the robots can be picked up and moved. However, when doing so, make certain that the ID tag is not obstructed from the camera.

B.1.4. After an Experiment

Once an experiment is complete, a few TIGERSquare macros are available to quickly plot results and assess the system performance. Each of the following commands must be preceded with the TIGERSquare object handle (typically "ts.").

- showAllCommands: Lists all the available functions in the object.
- displayData(1 or 0): Plots the robot trajectories and input velocities, outputs the saved poses and input velocities for each iteration.
- performanceReport(): Reads out the occurrences of robot detector errors and corrections. Also plots the step time data during the experiment
- getPoses(1 or 0): Plots the current robot poses.
- dock(): Commands the robots to move to the arena edge closest to the control laptop.
- getStatus(22): Reads out the current battery voltage of each robot.

If the experiment data will be saved for later use, be sure to save the entire workspace as a MAT file. If testing is complete, disconnect the cables from the computer, shut off the router, and check to make sure all the robots are turned off and back on the table. If any robots are low on power, connect them to a 5 V power supply for charging.

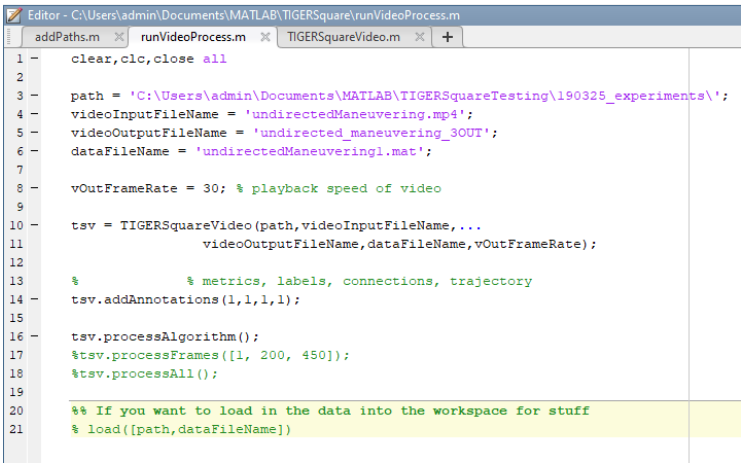

Figure B.4. Screenshot of runVideoProcess Script.

B.1.5. Processing a Video Recording

Video recordings of an experiment can be processed to undistort (due to the wide-angle lens), crop, and annotate each video frame. This can be done through the "runVideoProcess.m" script, Figure B.4. Similar to the "runExperiment.m" script, the video process script calls on a class file that contains predefined macros. The "runVideoProcess.m" script requires the experiment video file and workspace data (MAT file). In the script, the file path and names must be defined, as well as the output video frame rate. This frame rate affects the playback speed of the experiment. For example, if the output video frame rate is set to 30 Hz when the experiment refresh rate is 5 Hz, then playback speed will be 6 times as fast.

Annotations are defined by the addAnnotations function, where each input field indicates what annotation to include to the processed frame. Metrics annotations include the playback speed (or timestamp) and reference scale. The labels annotation places the robot number labels on each frame. The connections annotation draws the framework graph of robots on each frame. These drawn connections are based on the algorithm adjacency matrix. If no adjacency matrix exists, no connections will be drawn. Lastly, the trajectory annotation draws a tail for each robot indicating it's trajectory taken. The length and width of the trajectory tails can be modified in the class file.

Three processing methods can be used. Methods 1 and 2 will process frames based on the inputs of the addAnnotations function. Method 3 simply undistorts and crops the full video. The details of these methods are as follows

1. processFrames([frames]): Process and show each frame defined in a 1xN array. This can be used to test the appearance of each frame before processing the entire video.
2. processAlgorithm(): Process the input video and output only the algorithm section. In other words, only the section of the input video with corresponding data points from the workspace will be processed. All other frames will be cut. This method will annotate each frame based on the inputs of the addAnnotations function.
3. processAll(): This method simply undistorts and crops the full video.

It is important to note that the input video is recorded at 1020x720 with the wide-angle camera. If there are any changes (e.g. resolution, viewing position) to the input video, the calibration properties within the class file must be remade.

B.2. Maintenance

B.2.1. File Structure

Figure B.5 summarizes the current file structure on the Control Station Laptop. Currently, all design documents, notes, and data are saved within a TIGER Square Google Drive. The firmware and TSCS code are saved in local directories of the computer. Backups of these files are also found within the Google Drive. As mentioned in Chapter 7, a repository or file server will be implemented to maintain a unified set of files for future users to access.

B.2.2. Arena

Over time the arena surface collects dust, therefore every month the surface must be dusted off. This will avoid any excessive dust collecting on the wheels of the robots. Regularly check PVC pipe clamp screws, making certain that they are tightened so that the structure does not slip. Lastly, every month check that the arena is level using the digital level. Adjustments can be made using the feet located at each corner of the arena.

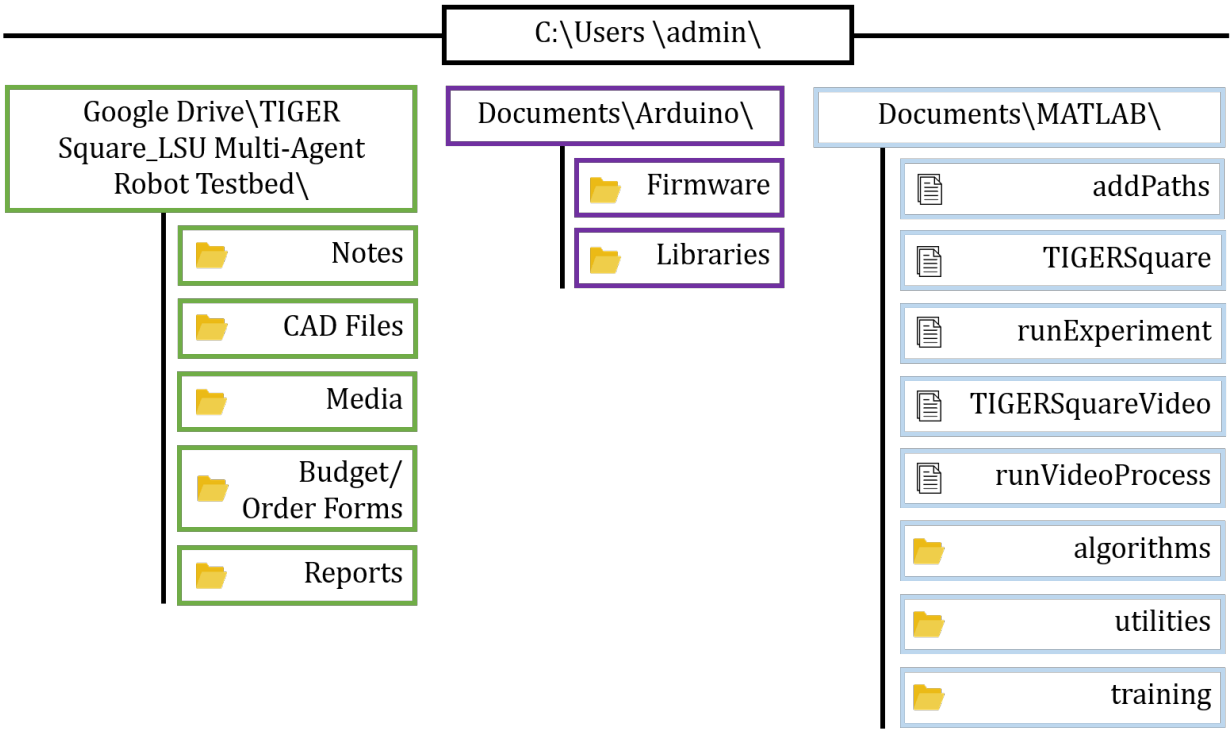

Figure B.5. TIGER Square File Structure

These feet can be turned using a 5 mm hex key.

B.2.3. TIGERBot

In the "Notes" folder of the file server there is a document detailing the assembly procedure of the robot. This can be used as a reference should any of the robots require assembly/disassembly. Similarly, the CAD files for the body components can be modified/reprinted should they need replacing. Finally, the Bill of Materials spreadsheet can be referenced to find any off-the-shelf components that need to be replaced. The robot firmware is downloaded to the robots via the Arduino IDE. In the IDE, the board selected and settings must be set to reflect the microcontroller being programmed. The NodeMCU is directly programmed using a USB micro cable. However, the ATmega chip is programmed using a USBTiny ISP AVR programming board interfaced to the 6-pin male header located at the head of the robot. The robots can be charged with any 5 V power supply. The male charging pins are located on the right side of the robot (when viewed from above and robot is facing away), next to the power switch. Pay careful attention to the polarity of the

Figure B.6. Charging Connection to TIGERBot.

charging pins. In the current Main Board design, there is no reverse-polarity protection, therefore the charging chip can very easily be burned out. Four female-to-male jumper wire pairs have been designated for use as the robot charging cables, see Figure B.6. To avoid the chance of plugging in the charging cable incorrectly, a "sleeve" has been hot glued to the female side of the cable so that it may only be plugged in with the correct polarity. Obviously, the male side polarity must be also be correctly connected to the power supply. The charging chip will regulate the power flow to allow up to 500 mA of current. As the battery begins to reach a full charge, the current flow will ramp down until eventually shutting off.

Vita

Victor Fernandez-Kim is a native of Baton Rouge, Louisiana. After graduating from Baton Rouge Magnet High School in 2012, he enrolled in Louisiana State University in Baton Rouge, where he earned a Bachelor of Science degree in Mechanical Engineering in Spring 2017, with a minor in Robotics. Upon graduation, he returned the following semester to pursue a Master of Science degree in Mechanical Engineering. He plans to pursue an engineering research and development career in the robotics industry upon completion of his degree.