March 2021

Distance-Based Formation Control using Decentralized Sensing with Infrared Photodiodes

Steven Williams

DISTANCE-BASED FORMATION CONTROL USING DECENTRALIZED SENSING WITH INFRARED PHOTODIODES

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

The Department of Mechanical Engineering

by
Steven Williams
B.S. Physics, State University of New York at Geneseo, 2018
May 2021

Acknowledgements

I would like to thank Dr. de Queiroz for giving me the opportunity to participate in this project, and for ending every meeting with a feeling of encouragement.

I would also like to thank all the graduate students of the iCORE lab for their friendship and kindness, I could not have picked a better group of people to work with.

Finally, I would like to thank my parents for all the support they have given me throughout my academic life.

Table of Contents

List of Figures

Abstract

This study presents an onboard sensor system for determining the relative positions of mobile robots, which is used in decentralized distance-based formation controllers for multi-agent systems. This sensor system uses infrared photodiodes and LEDs; its effective use requires coordination between the emitting and detecting robots. A technique is introduced for calculating the relative positions based on photodiode readings, and an automated calibration system is designed for future maintenance. By measuring the relative positions of their neighbors, each robot is capable of running an onboard formation controller, which is independent of both a centralized controller and a global positioning-like system (e.g., GPS-denied environments). This independence is referred to as decentralization. This was demonstrated with three different formation acquisition experiments, which were compared to equivalent experiments using a global positioning system and a centralized control station designed in prior studies. The centralized and decentralized experiments resulted in similar formation outcomes, but the steady-state error for the decentralized system increased. This result was an expected consequence of the uncertainty in decentralized localization measurements.

Chapter 1.
Introduction

1.1. Motivation

Multi-robot systems have been receiving more attention in recent years with the highly anticipated self-driving cars, drones, or any group of mobile sensors. Multi-robot systems offer substantial advantages over a single robot with swarming and position dependent task allocation, allowing them to accomplish tasks that could not be done with a group of uncoordinated individuals.

As with many of our technological endeavors, we find inspiration in the natural world, which is a humbling universe of unduplicatable complexities. We see formation control in schools of fish and flocks of birds, both of which exist as defense mechanisms from predators. Migratory birds fly in V-formation for increased aerodynamic efficiency and take turns pulling the draft in front. One of the most extreme examples of cooperative behavior are ant colonies which are known to organize themselves into bridges and even rafts in times of flood. An ant colony is considered a super organism, the individuals cannot survive independently from the group.

Formation control is the organization of a fleet of robots into a desired spatial pattern. Formation control has many applications, not the least of which is surveillance. A fleet of surveillance drones in formation can cover a broad area efficiently. Mobile robots with sensors can use their positions and sensor information to map out an approximate gradient of whatever they are sensing. For instance, a fleet of robots tracking a toxic gas could calculate the direction of the gas' diffusion. A more complex use of formation control is object manipulation, e.g. multiple drones picking up a package too large for an individual to pick up. The most common application of formation control would be in traffic control and collision avoidance. This has the obvious

application of self-driving cars but can also be applied to the movement of commodities in warehouses and fulfilment centers.

The positions of the robots during formation control can either be expressed in a global coordinate system or a local coordinate system. To use a global coordinate system, the robots must be within the framework of a central position sensing system where the absolute positions of each robot can be acquired. This can be accomplished with an external observer such as a camera that acquires positional data and communicates with the robots (indoor operation), or with a global positioning system (GPS) which allows each robot to obtain its own location and communicate that with its neighbor (outdoor operation). This prevents the robots from functioning in remote places where they cannot remain connected to a central positioning system. On the other hand, a decentralized positioning system can be constructed using only onboard sensors that measure the robots' relative positions.

The controller for the system can also either be centralized or decentralized. A centralized controller gathers all the position data, computes the control laws, and communicates the control outputs to the robots. The decentralized control system is one in which each of the robots gather position data and runs its own on-board control algorithms. Decentralized control can be easily scaled up, as opposed to a centralized system where the number of agents is limited by the computing power of the centralized control unit (CCU) and its ability to communicate with each individual.

1.2. Related Work

This section will review the references within the scope of this thesis and give credit to the prior work on which this project was based. The iCORE lab developed a centralized arena called

the TIGER square, the individual robots are referred to as TIGER robots. This arena was developed to serve as an experimental test bed for formation control algorithms within its framework.

The TIGER square project included an objective for future work, which was enabling the TIGER square project to operate in a decentralized fashion. The present project is focused on fulfilling this goal by calibrating onboard sensing systems, establishing a communication sequence to organize the robot's sensor usage, and implementing onboard control algorithms that utilize their sensor data to control inter-robot distances and acquire formation. The individual robots in this system are called TIGERBots.

1.2.1. The TIGER Square Project

The TIGERBot is a 9x9x7 cm two wheeled robot, powered by a 4.7V LiPo battery and propelled two stepper motors. Their frames are 3-D printed and they are supported by a ball bearing in the front a wheelie-bar in the back.

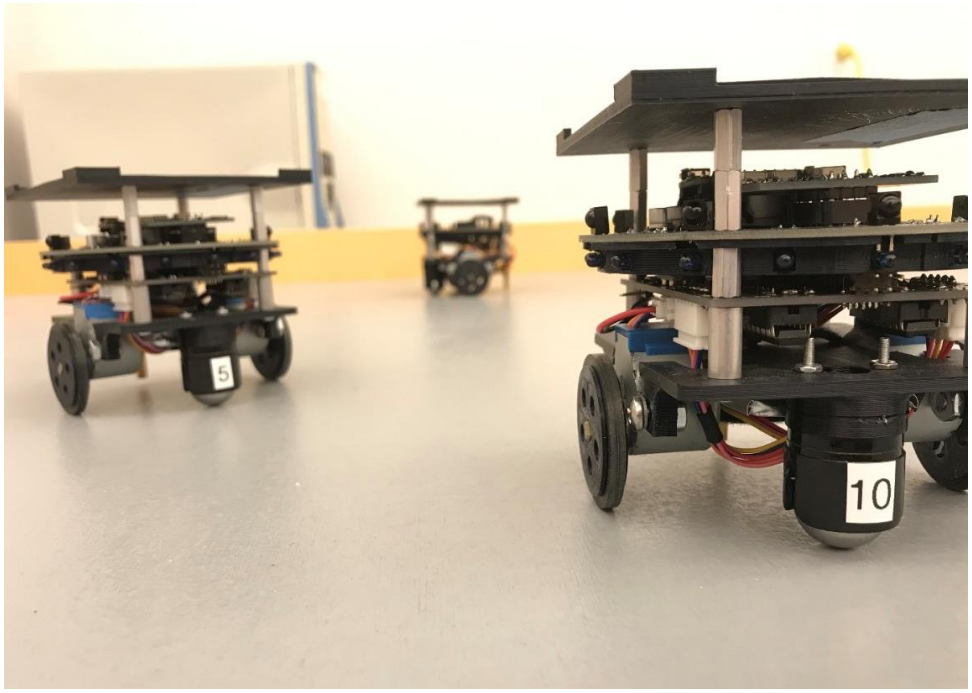

Figure 1. The TIGERBots with sensor boards mounted on them

Each TIGERBot has been equipped with a NodeMCU microcontroller, which uses an ESP8266 for network communication, as well as a separate board for controlling the stepper motors. This motor board is also equipped with a battery monitor and an inertial measurement unit (IMU). The performance of these robots has been thoroughly investigated[1], the key takeaway for the purposes of this experiment was that the velocities of the stepper motors should not surpass be set above 10 cm/s. The firmware for these microcontrollers came from a similar experimental multi-agent formation control system called "The Robotarium"[2].

The TIGER square setup is 1.6x1.6 m and uses a centralized positioning system. This positioning system is an overhead camera that identifies QR codes on the tops of the robots. This method had excellent accuracy (±5mm). The MATLAB controller would read these positions, run them through control algorithms, and send the resulting velocity outputs to the TIGERBots. The present project will still use this position sensing system for data collection.

1.2.2. Decentralized Sensing Projects

The most desirable solution for position sensing in experiments like this is computer vision. Although computer vision has been used in other projects such as [3] and [4], it is a very costly solution and isn't practical for small-scale robots such as the ones used in this project.

Decentralized sensing projects have also been designed using ultrasonic sensors, where one robot emits ultrasound and the other calculates the distance based on the time of flight (TOF)[5]. This works over long ranges (<100 m without obstructions) but it isn't ideal because of difficulties caused by reflected signals, especially indoors. There are also relative localization methods using ultra-wideband radio (UWB), which uses an array of antennas to compare the TOF of a UWB signal [6]. This has an extremely long range and does not require a direct line of sight because radio waves can travel through structures, but it's an impractical solution for a small-scale project

because it requires large antenna arrays (30x30x30cm) and has a resolution of 0.5 m, which is still quite impressive since it's measuring the TOF of light.

Other projects have used infrared localization methods [7-10] which is the sensing method used in this project. Using a circumferential array of photodiodes and LEDs, this method determines the distance from the emitting agent (range) based on the signal strength, and determines the direction to the emitting agent (bearing) based on the differing photodiode signals. Reference [9] describes the methods for converting multiple sensor signals into a single signal measurement with a bearing, this project borrows from these methods as described in section 2.1.2.

Bearing Measurement. A similar method was used for 3D localization using additional LEDs and photodiodes at different azimuthal angles on indoor flying robots [8] .

Infrared localization has the benefits of speed and small sensor size. It is limited to indoor use because direct sunlight would cause too much signal interference, and it relies on a direct line of sight between agents. It also has a limited range, which depends on the strength of the LEDs. Prior experiments [7] and [9] conducted performance tests for ranges up to 450 and 500 cm respectively, in both studies the range error stayed within ±4 cm for the first 100 cm and began to digress at greater distances.

Decentralized experiments require inter-agent communication. The speed and scalability of these experiments depend on the speed and scalability of their communication methods. Infrared localization requires the coordination of sensor usage to ensure that the infrared signals do not superimpose. Other studies have established communication systems using the infrared sensors themselves [9, 10], both of these experiments connected their photodiode signals to a radio-frequency demodulator to break down the photodiode signals into a coherent digital signal.

In the absence of demodulating hardware, the present experiment relies on a centralized communication system for sensor coordination. Decentralized agent to agent communication methods will be considered for future use in order to speed up loop time [7].

Chapter 2.
Experimental Methods

2.1. Sensor Board

The greatest challenge in setting up this system is designing sensors that accurately measure the distance between agents. Useful sensing requires the measurement of distance and direction, these are referred to as the range and bearing, respectively. The sensor board was designed with 8 IR photodiodes and 16 IR LEDs evenly spaced about its circumference as seen in Figure 2.

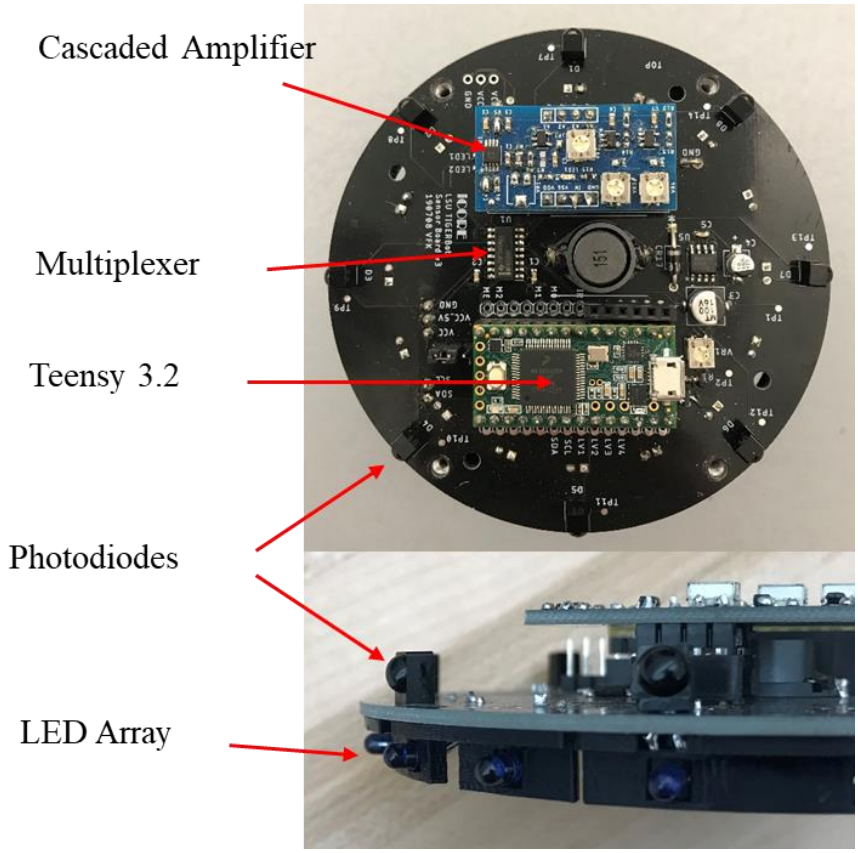

Figure 2. The sensor board top view and side view

This sensor system was proposed in [1], and minor changes were made here to maximize the effectiveness of the sensors. This section will be focused on the functionality of the sensor board and the interpretation of its signals in order to calculate range and bearing.

2.1.3. Hardware Layout

The sensor board is controlled by a Teensy3.2 microcontroller which is displayed in the block diagram below. Only one photodiode signal can be measured at once, and the Teensy controls which one is being measured with a multiplexer. The Teensy is also responsible for switching the LED array on and off.

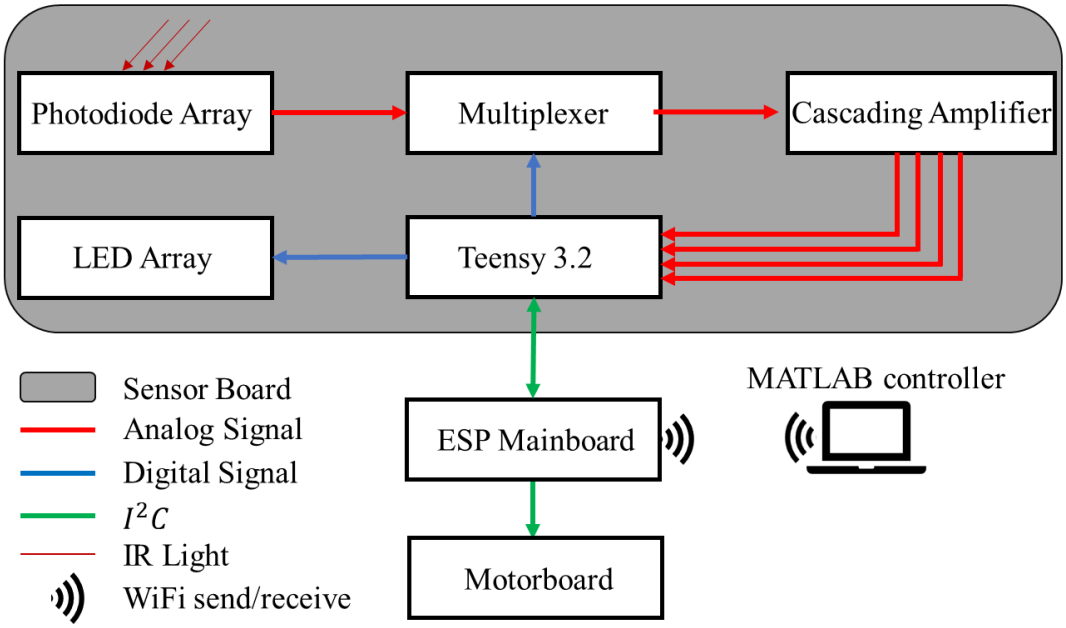

Figure 3. All the components in this block diagram are located on the sensor board except for the mainboard and the motorboard

The Teensy is connected to the Espressif (ESP) mainboard through inter-integrated circuit (I^2C) communication in which the ESP is the master and the Teensy is the slave. Through the I^2C, the ESP tells the Teensy when to acquire localization data, and when to turn on the LEDs. In turn, the teensy uses the I^2C to send the localization data to the ESP mainboard.

Photodiodes function as normal diodes in the dark, but when light shines on them they lose their ability to stop reverse-bias current flow. This reverse bias current flow serves as the photodiode's output signal, which is converted into a voltage by a transimpedance amplifier. The

cascaded amplifier is responsible for converting the current signals from the photodiodes into four distinct voltage signals, as a result only one photodiode's signal can be measured at a time. To accomplish this the photodiode is selected by the Teensy using a multiplexer. The first voltage signal from the cascaded amplifier comes from a transimpedance amplifier, which converts the photodiode's current signal into a voltage.

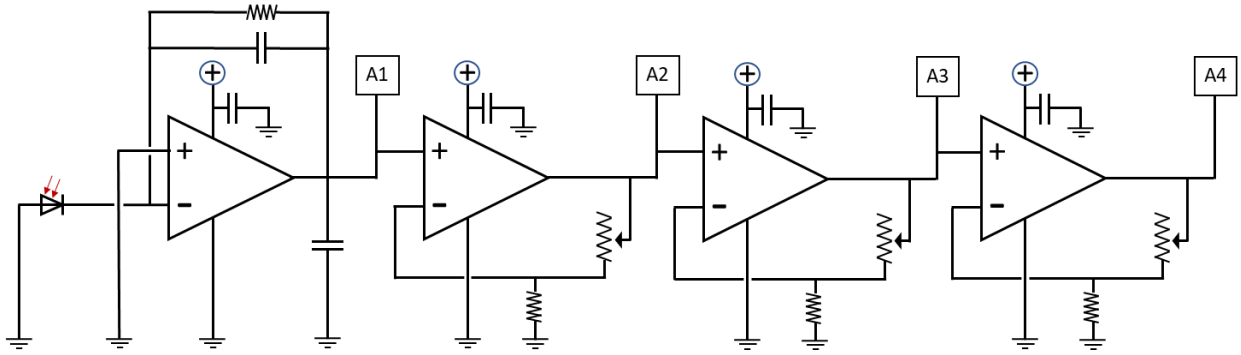

Figure 4. Circuit diagram for the cascaded amplifier

The signal A1 from the transimpedance amplifier has a value of $I_{pd}R$ where I_{pd} is the current induced by the photodiode. The subsequent signals (A2-4) have values of the previous signal multiplied by $\frac{(R_{tr}+R_{fixed})}{R_{fixed}}$, where the trimmers can be adjusted to control R_{tr} and the gains of these amplifiers. The object of controlling the gains of these amplifiers is to make their signals desaturate when the LEDs reach a specific distance away from the photodiodes. This is a part of the calibration method.

2.1.1. Range Measurement

The first objective is to determine the relationship between the signal strength and range. Figure 5. Amplified photodiode signals shows this relationship for the four signals from the cascaded amplifier.

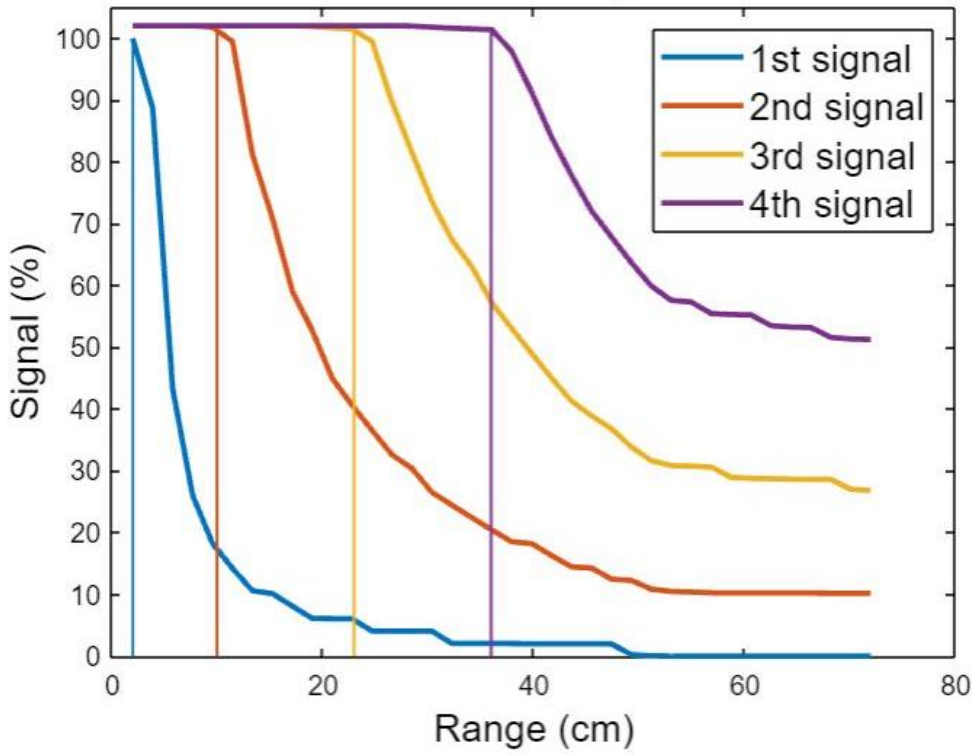

Figure 5. Amplified photodiode signals are tuned to desaturated at different range values

The signals resemble an exponential decay curve which continuously decreases and has a slope that flattens out and approaches zero. The unamplified signal can only be used to accurately detect range at short distances, after the 10cm mark the signal quickly flattens out and is no longer sensitive to changes in range. However, when the signal is amplified, it becomes sensitive to range changes at a greater distance. An amplified signal is only useful beyond the point where it de-saturates from its 5V maximum. This is where the cascading amplifier becomes useful, to maximize the usefulness of the signals, they were adjusted so the next one would de-saturate as the prior one began flattening out. To accomplish this, the points of de-saturation were loosely set to 10, 25, and 40 centimeters as seen in Figure 5. Amplified photodiode signals. The sum of these signals is used as a calibration curve to map the signal strengths to resulting range measurement. The most accurate way to use this calibration curve is to execute a lookup table and a linear

interpolation between the two data points that the measured value falls between. The methods for conducting these calibrations will be described more thoroughly in subsequent sections.

2.1.2. Bearing Measurement

The second measurement required for localization is the signal's bearing, for this the board is given an angular coordinate system as displayed in Figure 6a.

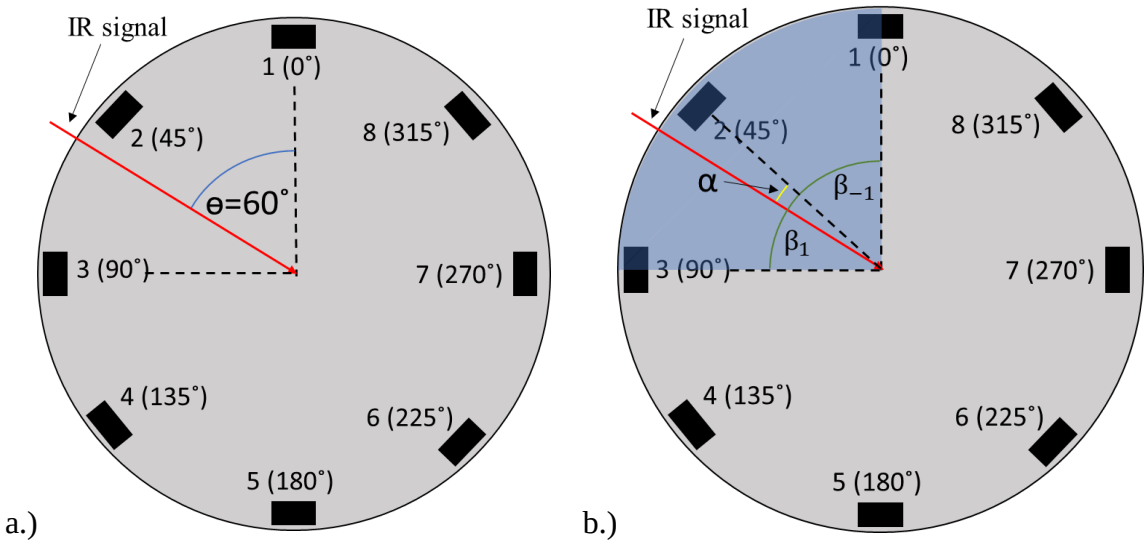

Figure 6. The bearing is taken as the angle between diode 1 and the signal. Signal bearing is calculated using the three photodiodes closest to the signal

The positive angular direction is counterclockwise, following mathematical convention and allowing for easier conversion between the board's coordinate system and the global coordinate system. The angle between the photodiode's normal vector and the incident IR signal affects the photodiode's ability to detect the signal, this is referred to as the angular response. For example, in Figure 6a the signal is offset from diode 1 by 60 degrees and only 15 degrees from diode 2, hence diode 2 will have a stronger response because the signal is facing it more directly. An individual photodiode would not be able to distinguish between a weak signal and a signal that

is on its peripheral. But using an angular response relationship and the signals from two or more photodiodes, both the bearing and the range can be calculated.

The most accurate model for angular response would simply be an interpolation of calibration measurements, which would have to include all combinations of angles and distances. This would require a massive amount of calibration data for each diode which would be time consuming to obtain and computationally costly to process. Instead, the angular response was modeled with

$$r' = r\cos(\theta) \tag{1}$$

where ϴ is the angle between the photodiode's normal vector and the incoming IR light, r' is the detected signal, and r is the signal that would be received if the photodiode were directly facing the IR source. Prior work [9] demonstrated the use of this angular response function in conjunction with trigonometric identities to derive r and ϴ from multiple photodiode measurements.

In theory, all photodiodes on the same side of the board as the signal direction could be used, which would be 4 photodiodes, but photodiode signals at such steep angles are so weak that they add more uncertainty than information. Instead, the three photodiodes with the highest signals are the only ones used. As seen in Figure 6b, α is the angle between the central photodiode and the IR signal, and β_i is the angle between the central photodiode and the i^{th} photodiode. $\beta_1 = \frac{\pi}{4}$ is the angle to the photodiode on the left, $\beta_{-1} = -\frac{\pi}{4}$ is the angle to the photodiode on the right, and $\beta_0 = 0$ is the angle between the center of the photodiode set and the center photodiode. The latter definition may seem trivial, but its application will be made apparent shortly.

The trigonometric identity

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \pm -\sin(\alpha)\sin(\beta) \tag{2}$$

is used to isolate $\cos(\alpha)$. This is accomplished by summing (2) for all β_i, which cancels out the sine terms, since $\sin(\beta_i)$ and $\sin(\beta_{-i})$ are of equal magnitudes with opposite signs. This sum also includes the central diode's position β_0 twice to preserve the symmetry for the cancellation of the second term. The central diode can be thought of as two diodes that approach the center from opposite sides. The resulting sum for this three-diode system is

$$\cos(\alpha+\beta_1)+\cos(\alpha+\beta_{-1})+2\cos(\alpha+\beta_0)=2\cos(\alpha)(\cos(\beta_1)+\cos(\beta_0)) \quad (3)$$

and all the terms on the left are then related to our angular response function. The relationship is rearranged to become

$$a=\frac{r'_{-1}+2r'_0+r'_1}{2+\sqrt{2}}=r\cos(\alpha) \quad (4)$$

which will be useful for finding r and α.

Since $\cos(\beta_1)$ and $\cos(\beta_{-1})$ are equal, the difference between the right and left photodiodes will cancel out the cosine terms and result in

$$b=\frac{r'_{-1}-r'_1}{\sqrt{2}}=r\sin(\alpha) \quad (5)$$

Now, all the tools are present to find r and α using

$$r=\sqrt{a^2+b^2} \quad (6)$$

$$\alpha=\left(\frac{b}{a}\right)\text{atan} \quad (7)$$

A similar method can be conducted using the angular response function

$$r'=r\sqrt{\cos(\theta)} \quad (8)$$

The signals must be squared before being substituted into (3) and (5) and the resulting relationships are

$$c = \frac{r'^2_{-1} + 2r'^2_0 + r'^2_1}{2 + \sqrt{2}} = r^2 \cos(\alpha) \tag{9}$$

$$d = \frac{r'^2_{-1} - r'^2_1}{\sqrt{2}} = r^2 \sin(\alpha) \tag{10}$$

$$r = \sqrt[4]{c^2 + d^2} \tag{11}$$

$$\alpha = \operatorname{atan}\left(\frac{d}{c}\right) \tag{12}$$

When performing this experimentally, each photodiode takes a sensor reading and the three greatest signals are chosen and categorized as the right, left, or center diode. Ultimately, this experiment will use whichever angular response function generates more accurate angular measurements.

2.2. Communication System

Communication methods are established to coordinate position sensing and distribute data. The communication in this experiment remains centralized, meaning there is two-way communication between the MATLAB control station and the robots, but not between individual robots. This method has the advantages of: allowing for the collection of the robot's sensor data, collecting position data from the overhead camera, aligning the start of the camera's data with the start of the decentralized distance-based controllers, and setting the robot's control parameters without re-installing their firmware.

2.2.1. Messaging Structure

The overhead camera is connected to the computer through a USB, the video feed is processed into individual images in MATLAB and the positions of the robots are collected by identifying QR codes on the top surfaces of the robots as in the prior centralized experiments[1].

The laptop is connected to the router's network through an ethernet cable, each robot occupies a port in this network and shares information through the user datagram protocol (UDP).

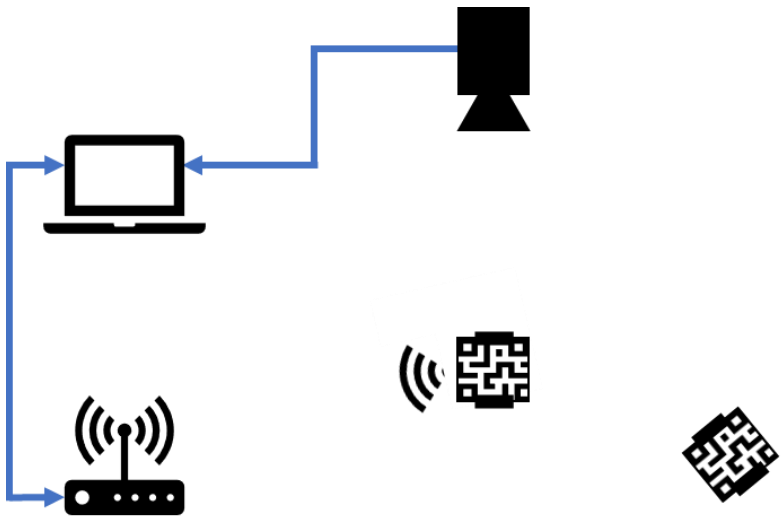

Figure 7. There is two-way communication between the robots and the MATLAB controller

The messages are structured with Java Script Object Notation (JSON). Within each message structure there is a collection of objects with names and values, which is encoded into an array of characters that is sent through the UDP. Both the MATLAB controller and the robots are responsible for composing JSON messages when they transmit data and decomposing JSON messages when they receive data. There are two variables that are contained in every message for this experiment: the destination's ID, and the message type. Since all parties involved are listening on the same network, the message will be received whether the ID is addressed to them or not, but if it does not match their ID they will not store the message's contents. An ID of 0 was used as the universal ID for broadcasting to all participants simultaneously instead of writing individual messages. The messaging time was the most time-consuming processes for this controller, taking between 100 and 200 ms to send a message to a robot. This sending time did not scale up with more robots. Oscilloscope testing showed that a message would reach all the robots within 1 ms

of each other, which suggests that sending messages is time consuming because of the time MATLAB takes constructing the UDP objects, and not because of the router's communication with its robot clients.

2.2.2. Decentralized Localization Coordination and Data Acquisition

Performing decentralized positioning experiments with IR proximity sensors requires coordination to avoid the interference of IR signals. Only one robot can be transmitting an IR signal at once, otherwise their signals will overlap, making position measurements impossible.

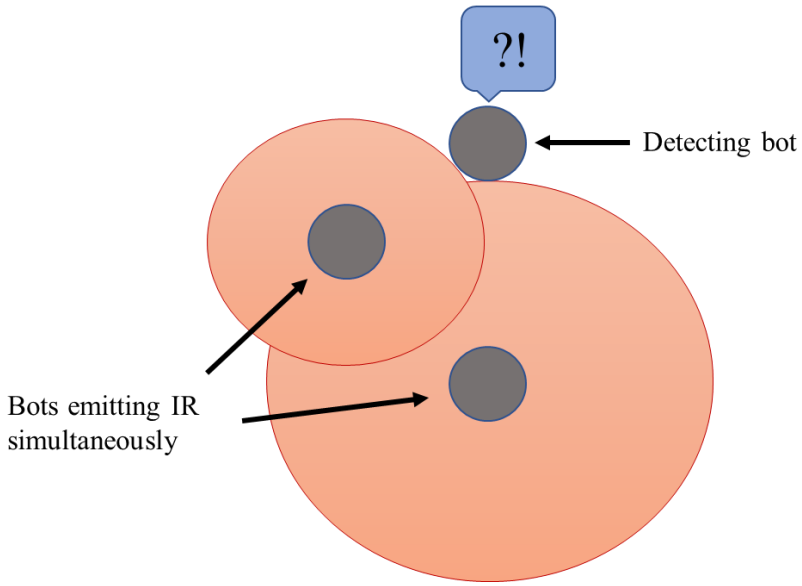

Figure 8. Superimposing IR signals make positions measurements impossible

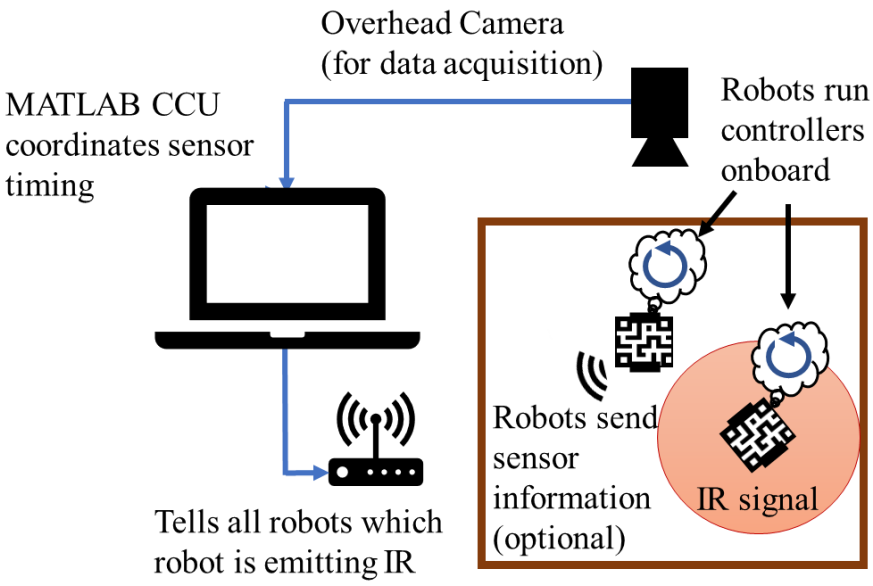

Figure 9. Decentralized localization is coordinated by the MATLAB CCU

The MATLAB controller is responsible for coordinating the use of IR localization, to do this a variable representing the ID of the emitting robot is sent to all the robots. If this variable's value is equal to a robot's ID, that robot will emit IR from its LEDs, otherwise the robots will sense the relative position of the emitting robot and store that robot's position.

Chapter 3.
Calibration

Since this controller's behavior primarily relies on the accuracy of the decentralized position sensing, calibrating these sensors becomes the most crucial process of the entire experiment. The calibration process is categorized into three segments, a thorough preliminary calibration on a track, a time efficient intermittent calibration executed on the tiger square with the guidance of MATLAB, and the tuning of the LEDs output to ensure consistency in the robot's localization. The quality of the bearing calculations will be compared using different angular responses functions and the uncertainty in the error in the range calculations will be assessed at different bearing values.

3.1. Calibration Methods

3.1.1. Preliminary Calibration

The track for the preliminary calibration was conducted for all sensor boards and special care was taken to ensure ambient light was consistent throughout all calibrations.

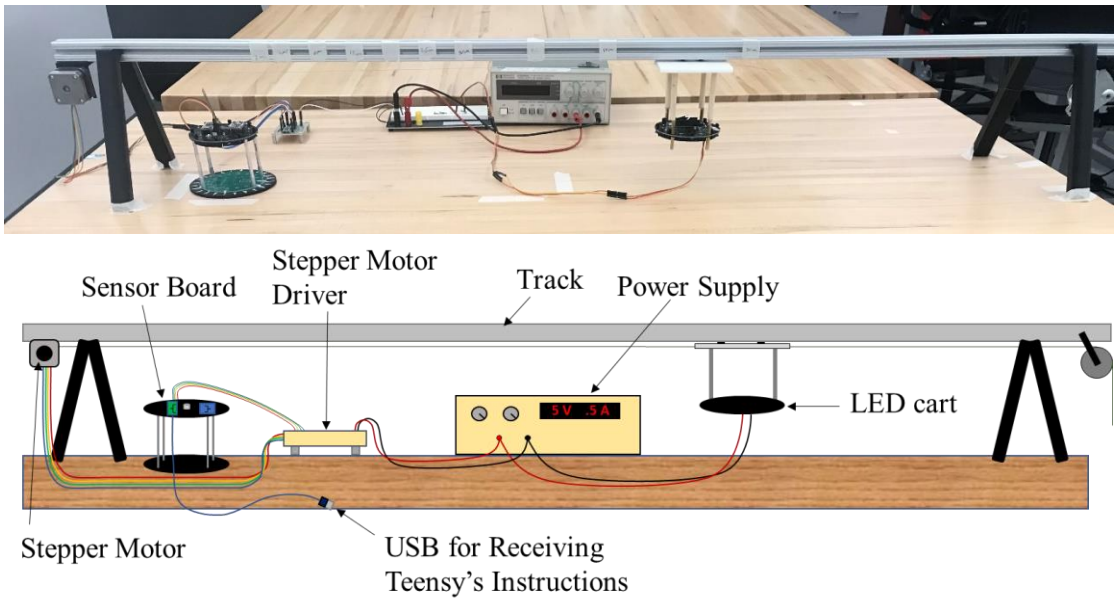

Figure 10. The preliminary calibration setup

The Teensy guides the user through the steps of this calibration process through the serial monitor on a PC. The Teensy is also responsible for moving the LED cart by controlling the stepper motor. Before calibrations runs, the user can enter an integer into the serial monitor and the stepper motor will move the LED cart by that many centimeters, positive values move it away from the sensor board and negative values move it closer. The user can also press the "enter" key to take a sensor measurement, and the measurement from each of the four amplified photodiode signals will be displayed. In order to maintain a consistent LED output, the intensity of the LEDs are adjusted until the LED cart draws 0.5 A at 5 V.

As explained in Section 2.1.1. Range, the amplified photodiode signals must be set to desaturate at specific distances to maximize the range and sensitivity of the sensors. The user is instructed to move the cart to the 10 cm point and adjust the amplification of the second signal until it exits saturation. The signals range from 0 to 3.3V, which is normalized to a percent value

from 0 to 100, so the signal has left saturation when it drops below 100. This is repeated for the third signal at 25 cm and the fourth at 40 cm.

Each diode is then calibrated individually, starting with diode 1. The user is instructed to move the LED cart to the 2cm mark, align diode 1 on the sensor board, and enter "go" when ready.

The bearing and range calculations operate under the assumption that all photodiodes have an identical response to the IR signals. Of course, this is not the case, and in order to account for this the photodiode signals are normalized to match the average response of all the diodes. This normalization process occurs during the preliminary calibration and will be explained further in 3.2.1. Preliminary Calibration.

During a calibration run the sensor board takes measurements at 2 cm increments, once the LED cart reaches 70 cm it returns to the 2 cm mark and the process is repeated for the subsequent diode until all the diodes have been calibrated. When the calibration is complete the calibration data is stored in the Teensy's permanent memory.

3.1.2. Automated MATLAB Calibration

Over time the behavior of the sensor boards might digress, and repeated calibration will be required. The preliminary calibration process required a painstaking amount of time, instead it can be automated by the MATLAB controller. The centralized MATLAB controller collects position information from the overhead camera and controls the robots' velocities to reach the desired positions for calibration. During the calibration, the robots form into a polygon with the number of vertices matching the number of robots being calibrated.

Figure 11. MATLAB controlled sensor calibration. The five robots at the points of the pentagon are being calibrated, the robot in the center is serving as the beacon

In the center of the polygon there is a robot which acts as a beacon, serving the same purpose as the LED cart in the preliminary calibration. Sensor measurements are taken for distances from 60 cm to 14 cm. Eight measurements are taken at each distance, at each of the measurements the robot rotates to allow a different photodiode to face the beacon. These eight measurements are averaged and stored for their corresponding distance in the calibration array.

Replicating the calibration data inside the Teensy would require extensive I2C communication between the mainboard and the Teensy to relay information to and from the MATLAB controller. Instead, the Teensy behaves much as it would during a normal range detection, combining readings from all the diodes, only in this case the Teensy sends the mainboard its sensor reading instead of the calculated distance. The mainboard processes and stores this data along with position data from the MATLAB controller to create its own calibration array.

Additional communication functions are added to the system to accomplish this calibration method. The robots' default is to request position information from the Teensy (the Teensy uses its original calibration), this can be changed from the MATLAB controller, setting the robots to request sensor information from the Teensy instead. After each sensor reading is taken it is summoned from the mainboard by the MATLAB controller, which passes the reading through a filter to ensure that the measurement was taken correctly. Based on this, MATLAB either instructs the sensor boards to retake the reading or move on to the next calibration position. These readings are stored in the MATLAB controller for future analysis.

The consistent output of the robot's LEDs is just as important as the calibration of the photodiodes. The LED outputs can be tuned with a trimmer on the top of the sensor board. In order to tune them to consistent outputs, a MATLAB script was created that behaves similarly to the automated MATLAB calibration method, the robots arrange into a symmetrical polygon with 40 cm edges, similar to Figure 11. MATLAB controlled sensor calibration. The five robots at the points of the pentagon are being calibrated, the robot in the center is serving as the beacon, and each robot faces its neighbor in the clockwise direction. The robots take a range reading which is displayed on the MATLAB console, and the user adjusts the output of the LEDs on the clockwise neighbor until the range reading matches the edge length of 40 cm. The user indicates when the correct LED output is reached, and the process is repeated for the next robot until they have all been calibrated.

3.2. Calibration Results

3.2.1. Preliminary Calibration

The primary purpose of the preliminary calibration was to set the gain values for the cascading amplifier, and to normalize the photodiode signals. To normalize the signals, each photodiode's signal is integrated over the span of the calibration distance and the normalization constant for each diode is taken to be the average of all the photodiode integrals divided by the diode integral for that photodiode. Whenever a sensor measurement is taken from that photodiode in the future, that measurement is multiplied by the photodiode's normalization constant. The outcome is photodiode readings that are more comparable and better suited for range and bearing calculations.

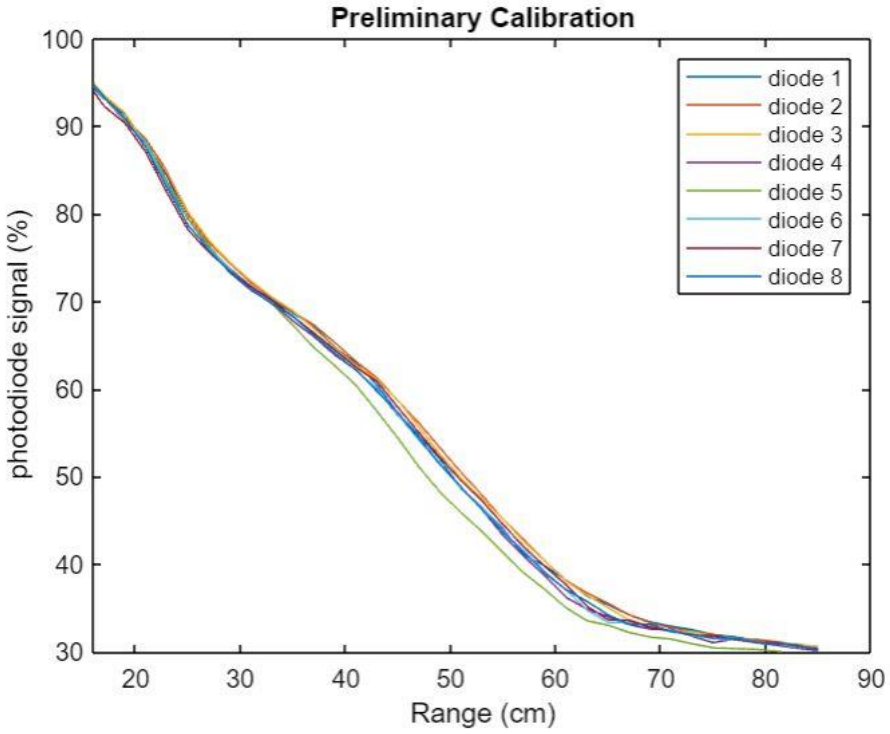

Figure 12. The photodiode responses are consistent during the preliminary calibration, notice that photodiode 5 has a weaker signal than the rest

The low sensitivity of diode 5 on this sensor board is accounted for with a higher normalization as displayed in .

Table 1. Normalization constants for each photodiode on a sensor board

Photodiode	1	2	3	4	5	6	7	8
Norm	1	.99	.99	1	1.02	1	1	1

It can also be noted that the responses of the photodiodes start to flatten out after 65 cm, which leads to an increased uncertainty in distance measurements for distances within the flattened region. To maximize the quality of distance measurements, the robots will have target distances of less than 60 cm.

3.2.2. Bearing Calculation Tests

The preliminary calibration is a well-controlled experiment, ensuring consistent LED output and only taking measurements from one photodiode at a time, resulting in precise and consistent outcomes as seen in Figure 11Figure 12. The photodiode responses are consistent during the preliminary calibration, notice that photodiode 5 has a weaker signal than the rest. On the other hand, the intermediate MATLAB calibration uses the same sensing techniques used in the final experiment, where each photodiode acquires a signal, and those signals are combined into the direct signal r using equations (6) and (11). As a reminder, r is the value calculated from the measured signals which represents the signal that would be measured by a photodiode directly facing the source, and equations (6) and (11) use the relationship between the photodiode's signal response and the incident angle of the LED source to calculate r.

The angular response function used in the range and bearing calculations is either $Cos(\theta)$ or $\sqrt{Cos(\theta)}$. Before continuing to the MATLAB calibration, the most accurate angular response function was determined by comparing the accuracies of the bearing calculations from these two angular response functions. This was accomplished using the MATLAB control station to direct the robots to designated measurement positions, coordinate and request their measurements, and plot the results. This test was conducted at night in order to avoid the possibility of ambient light interfering with the bearing calculation. The bearing was calculated from 0 to 350 degrees in 10-degree increments, these measurements were taken at distances of 20, 40, and 60 cm from the IR beacon robot.

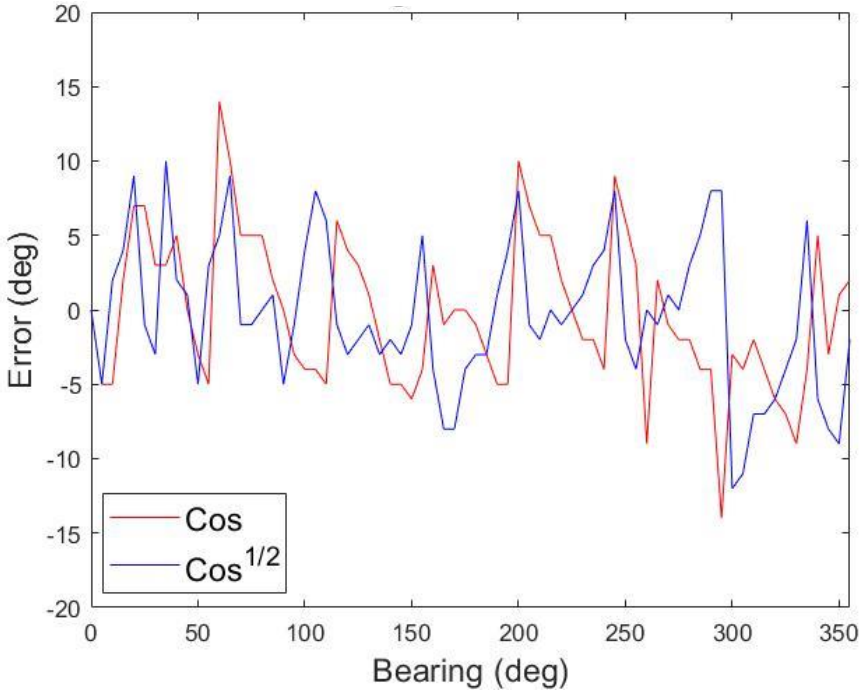

Figure 13. Bearing error at a distance of 20 cm

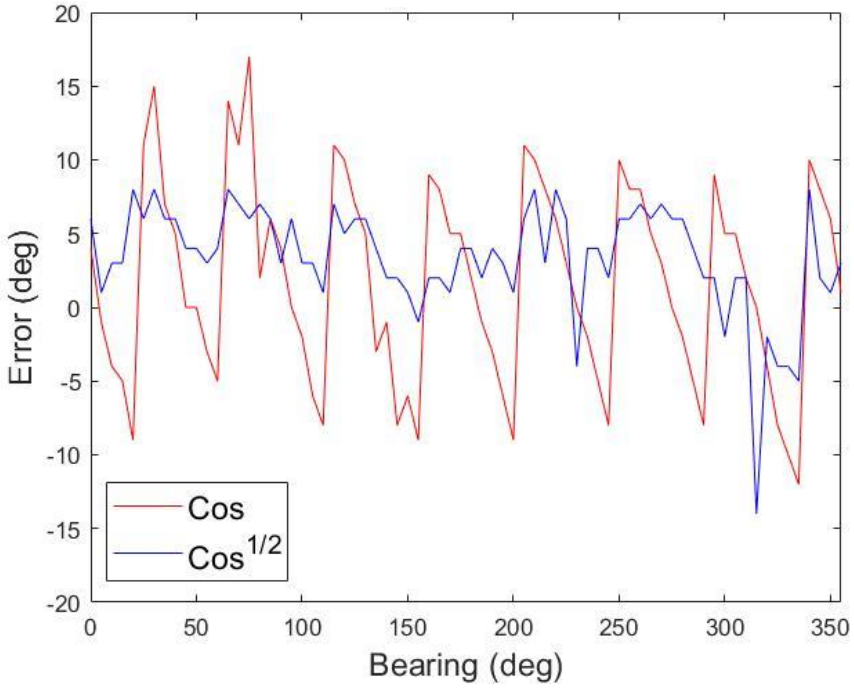

Figure 14. The bearing error at 40 cm was substantially lower for the $\sqrt{Cos(\theta)}$ function

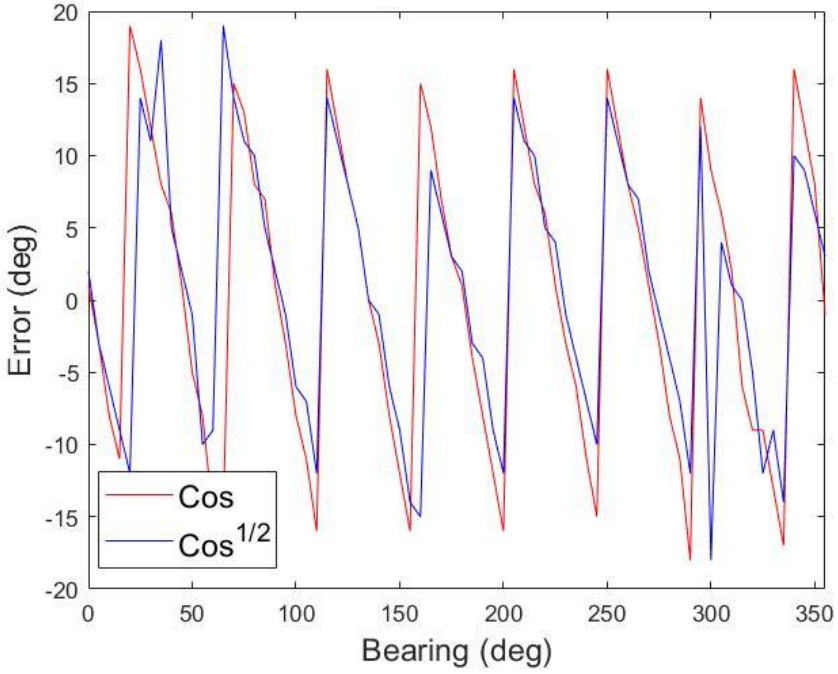

Figure 15. Bearing error at a distance of 60 cm

From these plots it can be seen that the two angular response functions exhibit similar behaviors at distances of 20 cm and 60 cm, but at 40 cm the $\sqrt{Cos(\theta)}$ function substantially outperformed its $Cos(\theta)$ counterpart. The errors in the bearing calculations are periodic with respect to the bearing's alignment to the photodiodes, which is to be expected because the orientation of three closest photodiodes repeats itself every 45 degrees. It can also be observed that the error is offset towards the positive direction, and more so at greater distances, this may be due to asymmetries inherent in the photodiodes. The key takeaway is the overall performance, which is summarized in the table below.

Table 2. The mean error and the mean of the absolute value of the error

	Mean Error (deg)	Mean Absolute Error (deg)
$Cos(\theta)$	0.62	6.38
$\sqrt{Cos(\theta)}$	3.27	5.33

The mean absolute error is the most important indicator for quality of outcome, which is simply the average distance from the true value. $\sqrt{Cos(\theta)}$ outperformed $Cos(\theta)$, and will be used in all tests and experiments going forward. The mean error resulting from using the $Cos(\theta)$ response was 3.27 degrees; this value is now added to the bearing calculations made by the robots.

3.2.3. MATLAB Calibration Results

During the MATLAB calibration the photodiodes were directly facing the beacon robot, but the calculated r values were still slightly greater that those acquired by the single diodes during the preliminary calibration. This is because the calculated signals rely on the angular responses of the photodiodes.

Eight signals are captured during this calibration, each with a different photodiode facing the LED beacon robot. The inconsistency between these signals is used to calculate the uncertainty of the signal readings. The figure below displays each sensor board's signal response averaged over all photodiodes. The error bars represent the standard deviation of the 8 signals.

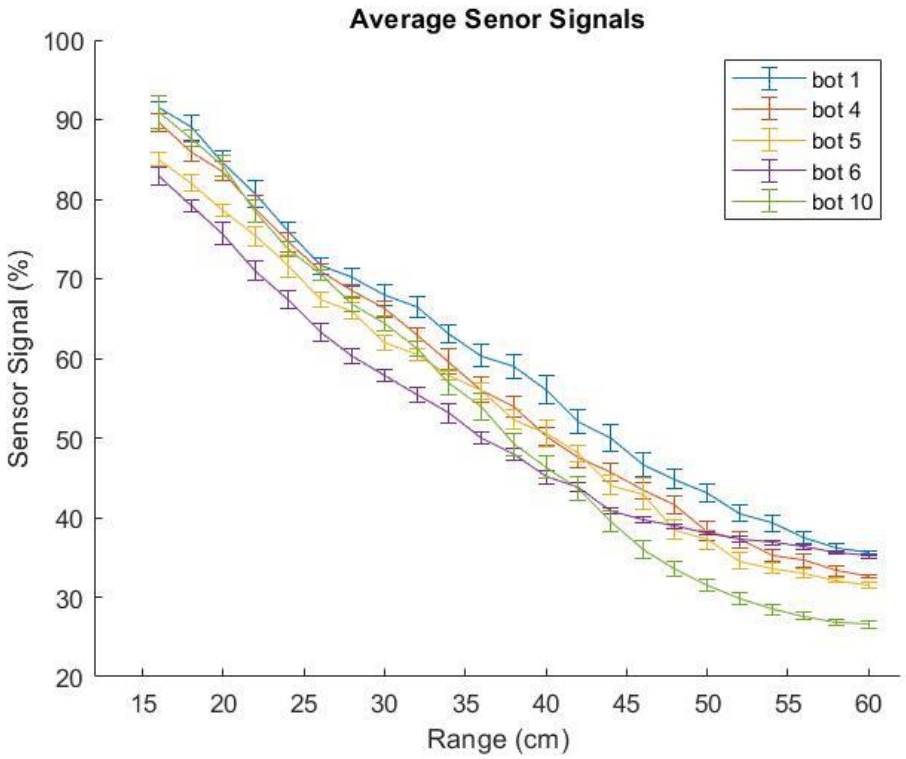

Figure 16. The calibration curves for each sensor board using the MATLAB calibration technique

Since distance calculations rely on sensor readings, the uncertainties in the sensor signals are useful for determining the precision of the distance calculations through the propagation of uncertainty.

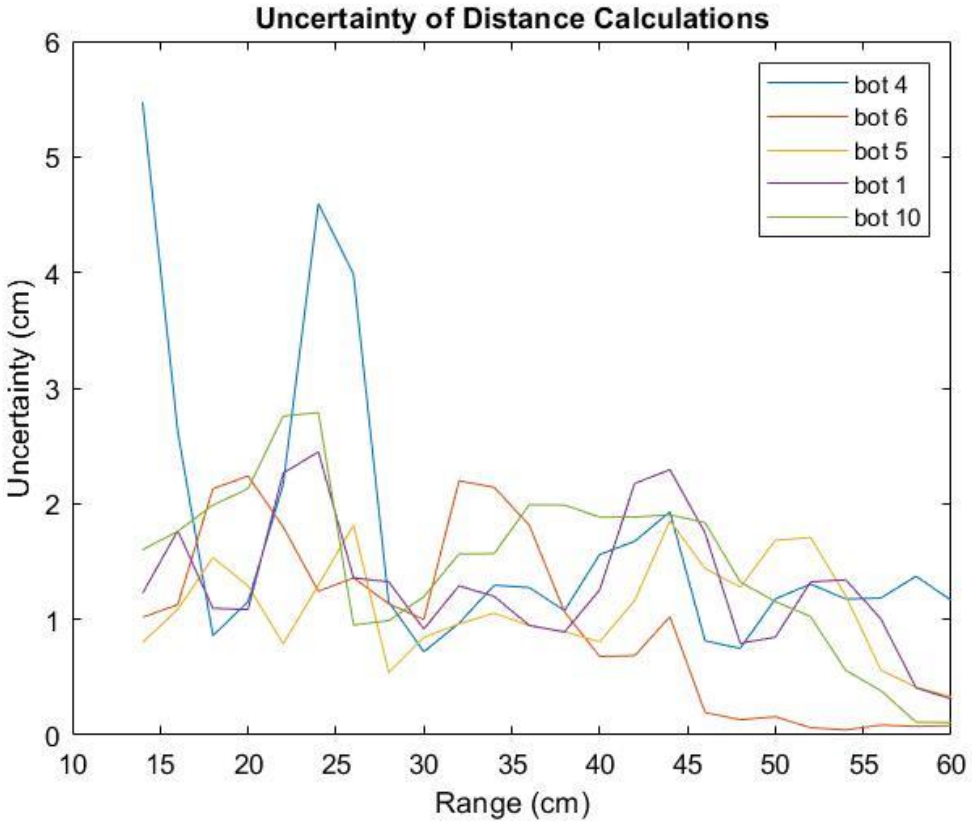

Figure 17. The uncertainty in distance calculations is based on the uncertainty in sensor measurements

Uncertainty is propagated from one quantity to another by multiplying the uncertainty of the initial quantity by the second quantity's derivative with respect to the first. For example, if we have a function $f(x)$, the uncertainty in f based on the uncertainty in x is

$$\Delta f(x) = \frac{df(x)}{dx} * \Delta x \tag{13}$$

Since distance calculations are based on a linear interpolation of the sensor signals in Figure 16, the derivative for distance with respect to sensor signal can be found using a central difference approximation. This was then multiplied by the uncertainty in the sensor outputs to obtain the results displayed in Figure 17. The uncertainty in distance calculations is based on the uncertainty in sensor measurements. There are a few outliers in this dataset, but overall a generous value to place on this uncertainty would be $\pm$2.5 cm.

The intermediate MATLAB calibration is likely going to be the one used more frequently because it can be conducted quickly in order to account for changing ambient light and photodiode behavior. Ultimately, the option with superior accuracy will be used in the distance controller.

A range accuracy test was conducted from 14 to 70 cm in 3 cm increments, and with bearings of $\theta = 0$, $\frac{\pi}{8}$, and $\frac{\pi}{16}$. These bearing were chosen because the photodiodes are separated by $\frac{\pi}{4}$ radians, and as a result these bearings represent orientations where the IR source is directly facing a photodiode ($\theta = 0$), is evenly split between two photodiodes ($\theta = \frac{\pi}{8}$), and between two photodiodes but closer to the central photodiode ($\theta = \pi/16$).

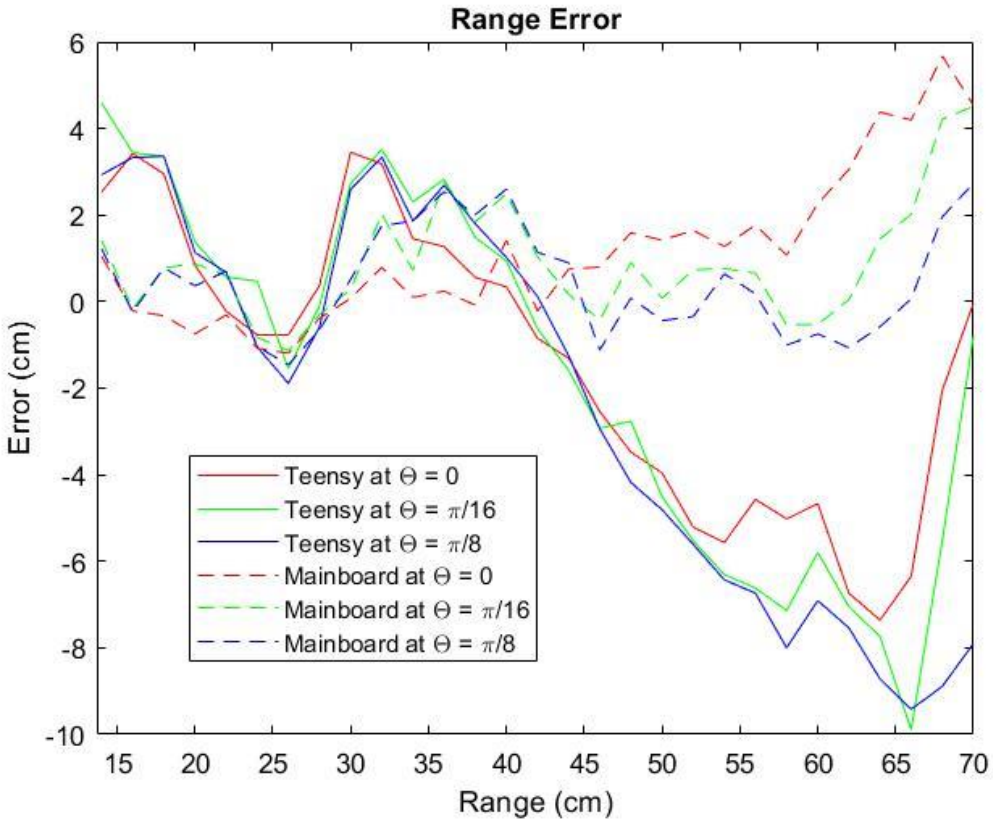

Figure 18. The range error for different calibration methods

From this data we can see that the mainboard calibration data created superior distance measurement results, and its error remains within the photodiode uncertainty until the range surpasses 60 cm. This is not surprising because the MATLAB calibration used the same combined signals as this range test, as opposed to the Teensy's experimental data which is based on individual photodiode measurements. While the Teensy's preliminary calibration was crucial for setting up signal amplifications and acquiring normalization constants, it will no longer be used for range calculation. Now that the sensors have been thoroughly calibrated and characterized, they can be used to provide the robots with bearing and range information for distance-based controllers.

Chapter4.
Formation Control Algorithm

Ultimately, the purpose of this project is to acquire a formation by controlling inter-agent distances. Thus far this study has discussed the methods for inter-agent position sensing, this chapter will describe the controllers used for implementing that information.

In order to acquire a specific polygonal formation, the lengths of at least $2n-3$ edges must be controlled. An edge is just the distance between two agents, and n is the number of agents in formation. A graph with at least $2n-3$ edges is considered rigid and a graph with exactly $2n-3$ edges is considered minimally rigid. For example, consider a square with $n=4$ agents.

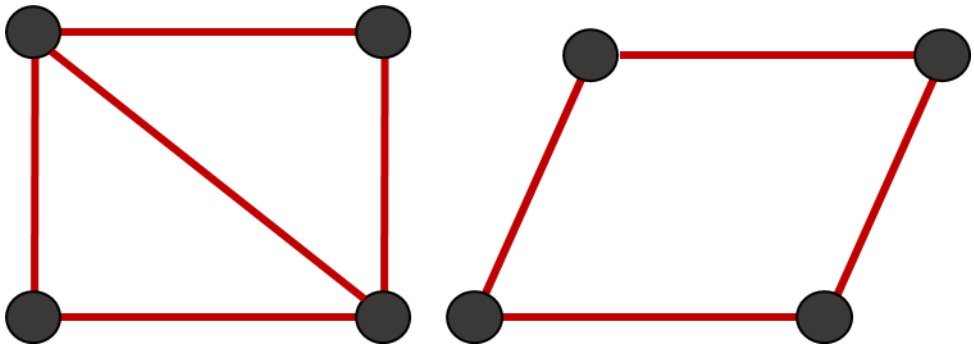

Figure 19. A minimally rigid graph (left) and a non-rigid graph (right)

If there are only four edges you cannot control the overall shape, as it can just as easily become a rhombus. With a fifth edge you can completely control the overall shape. It should also be noted that there are two categories for graphs used in formation control, directed and undirected. In the context of formation control this describes which agent controls which edge. In the directed case, an edge length is controlled by one agent controlling its distance from the agent on the other side of the edge. In the undirected case, both agents of an edge are controlling the edge length.

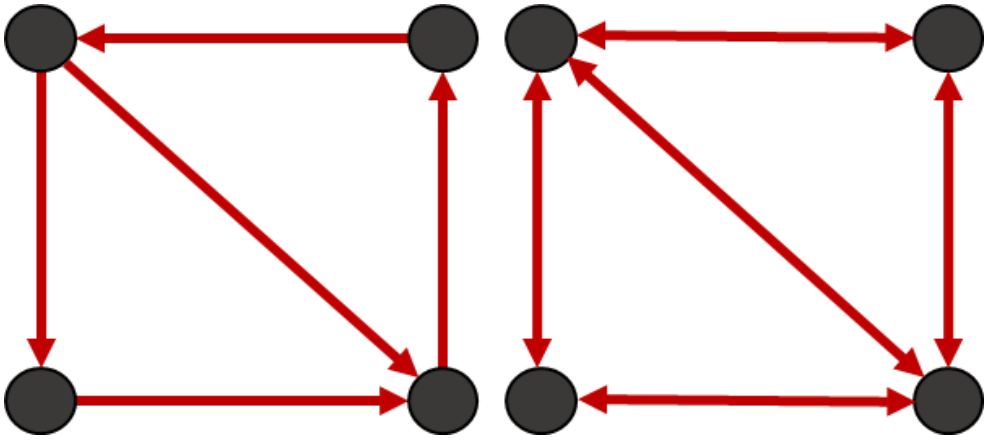

Figure 20. A directed graph (left) and an undirected graph (right), only minimally rigid directed graphs are used in this experiment

4.1. Global Holonomic Distance-Based Controller

The distance controller used in this experiment is the same as the one presented in [11] and [12], and implemented in centralized experiments [1]. The control law is inherently decentralized because it only depends on the relative positions of the robots and can be expressed in any coordinate system. The remainder of this chapter will describe this distance controller, its application to a non-holonomic robot, and its manifestation in the robot's local coordinate system, which matches the robot's onboard sensor data.

*note: going forward, all vector quantities will be in **bold** and all scalar quantities will be in normal type

In a global coordinate system, the positions of the robots are defined by the vectors $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ and the relative position between two robots is denoted as

$$\tilde{\boldsymbol{p}}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j \tag{14}$$

where $\tilde{\boldsymbol{p}}_{ij}$ is the vector from Bot j to Bot i and θ_j is the angle between $-\tilde{\boldsymbol{p}}_{ij}$ and the x-axis.

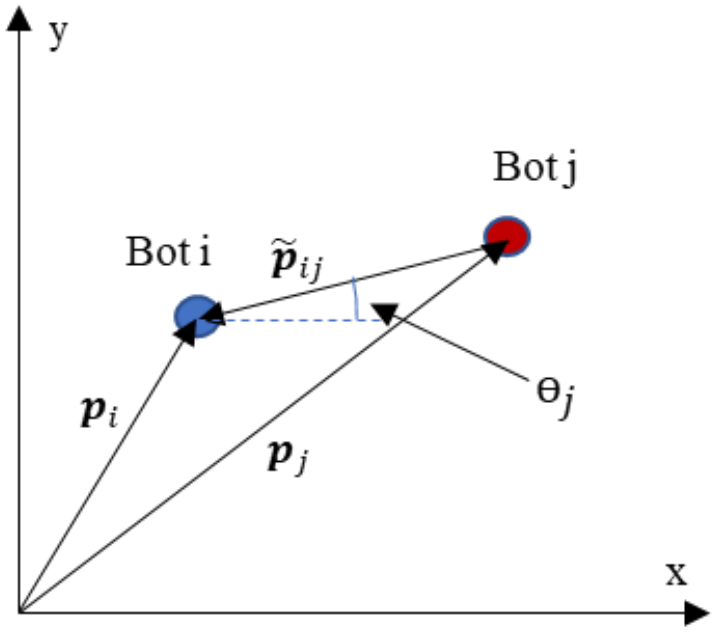

Figure 21. Two robots in the global coordinate system

Let d_{ij} be the desired distance between the two robots and let

$$z_{ij} = \left\|\tilde{\boldsymbol{p}}_{ij}\right\|^2 - d_{ij}^2 \tag{15}$$

In the case where the i[th] robot is attempting to maintain a distance of d_{ij} between itself and the j[th] robot, the control law for the i[th] robot becomes

$$\boldsymbol{u}_i = -k\,\tilde{\boldsymbol{p}}_{ij}\,z_{ij} \tag{16}$$

where k is a positive constant. $\boldsymbol{u}_i$ can be separated into its components and expressed as

$$\begin{bmatrix} u_{ix} \\ u_{iy} \end{bmatrix} = k\left\|\tilde{\boldsymbol{p}}_{ij}\right\| z_{ij} \begin{bmatrix} Cos\ \theta_j \\ Sin\ \theta_j \end{bmatrix} \tag{17}$$

where θ_b is the angle between $-\tilde{\boldsymbol{p}}_{ij}$ and the x-axis.

This control law can be applied to multiple targets simultaneously by summing (16) over the set of neighbors (Bot j's) that Bot i is directed to ($j \in dir.$).

$$\boldsymbol{u}_i = -k \sum_{j \in dir.} \tilde{\boldsymbol{p}}_{ij}\, z_{ij} \tag{18}$$

This controller assumes that the robot can independently control each degree of freedom (holonomic), for example a hamster ball can control its x and y velocities on an x-y plane (assuming a cat doesn't get involved). But this does not apply to most situations, including the robots used in this experiment.

4.2. Non-Holonomic Conversion

The robotic vehicle has two controllable motion outputs, $[v_i, \omega_i]$ where v_i is the vehicle's speed and ω_i is the vehicle's angular velocity about the z axis. These are both considered scalar quantities. Implementing the current controller (18) under these constraints is demonstrated in [12] and [13], these methods are summarized in this section.

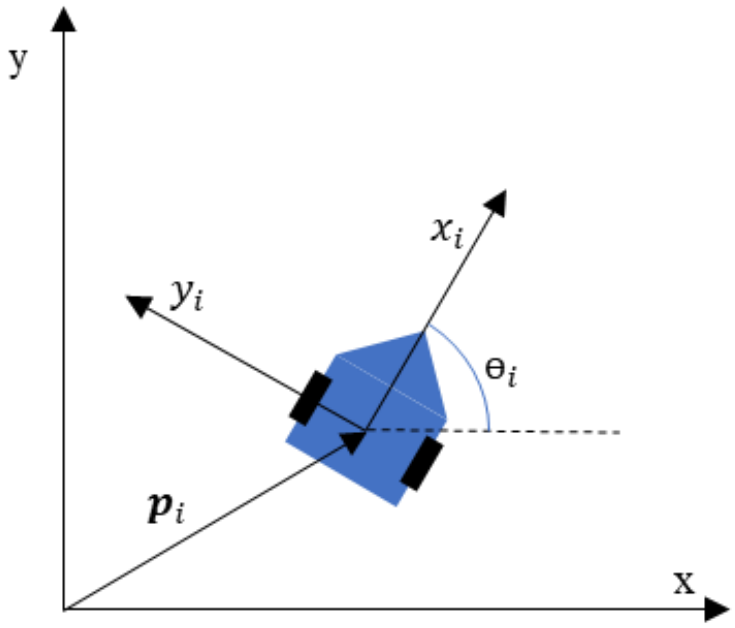

Figure 22. The coordinate system of the robot is expressed within the global coordinate system

These constraints on the robot's motion result in

$$\dot{\boldsymbol{p}}_i = \begin{bmatrix} v_i Cos\ \theta_i \\ v_i\ Sin\ \theta_i \end{bmatrix} \tag{19}$$

$$\dot{\theta}_i = \omega_i \tag{20}$$

where θ_i is the angle between the robot's coordinate system and the global coordinate system.

The desired global velocities are

$$\boldsymbol{u}_i = \begin{bmatrix} u_{ix} \\ u_{iy} \end{bmatrix} \tag{21}$$

which can only be achieved by bringing θ_i to the desired value of

$$\theta_{di} = \begin{cases} 0\ , & if\ u_{iy} = u_{ix} = 0 \\ \text{atan}\left(\dfrac{u_{iy}}{u_{ix}}\right)\ , & otherwise \end{cases} \tag{22}$$

The error in the orientation of the robot then becomes

$$\tilde{\theta}_i = \theta_i - \theta_{di} \tag{23}$$

which leads to the simple angular controller

$$\omega_i = -\beta\tilde{\theta}_i \tag{24}$$

where β is a positive constant. The speed output for the robot is the scalar product between the desired velocity output and the direction the robot is facing

$$v_i = \boldsymbol{u}_i \cdot \hat{\boldsymbol{x}}_i \tag{25}$$

which can be a negative value, representing the robot moving in reverse.

Thus far, the kinematic control law for the robots can be expressed as

$$v_i = k \sum_{j \in dir.} \left(\left\| \tilde{\boldsymbol{p}}_{ij} \right\| z_{ij} \begin{bmatrix} Cos\ \theta_j \\ Sin\ \theta_j \end{bmatrix} \right) \cdot \begin{bmatrix} Cos\ \theta_i \\ Sin\ \theta_i \end{bmatrix} \tag{26}$$

$$\omega_i = -\beta \left(\theta_i - \text{atan} \left(\frac{\sum_{j \in dir.} \left(\left\| \tilde{\boldsymbol{p}}_{ij} \right\| z_{ij} sin\ \theta_j \right)}{\sum_{j \in dir.} \left(\left\| \tilde{\boldsymbol{p}}_{ij} \right\| z_{ij} Cos\ \theta_j \right)} \right) \right) \tag{27}$$

Finally, this controller is expressed in terms of the robot's local coordinate system in order to apply it based on the robot's onboard sensor data.

4.3. Formation Control Based on Robot's Local Coordinate System

In a decentralized sensing mechanism, the robots have no connection to a global coordinate system, but instead operate within their own local coordinate system. This can be expressed in the same fashion as the global coordinate system with a few key modifications.

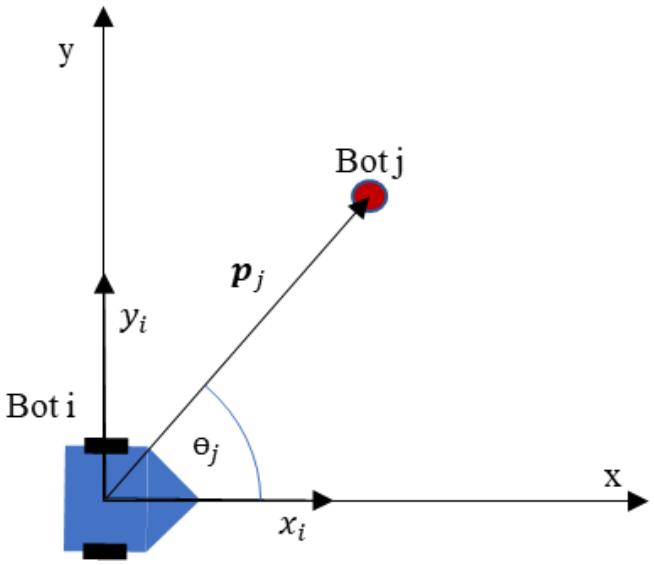

Figure 23. The robot's local coordinate system can be expressed as a special version of the global coordinate system

The local coordinate system becomes coincident on the global coordinate system when $\boldsymbol{p}_i = 0$ and $\theta_i = 0$.Given $\boldsymbol{p}_i = 0$,

$$\tilde{\boldsymbol{p}}_{ij} = -\boldsymbol{p}_j \tag{28}$$

which represents the distance measurements made by the sensor boards, and (15) becomes

$$z_{ij} = \|\boldsymbol{p}_j\|^2 - d_{ij}^2 \tag{29}$$

Substituting $\theta_i = 0$ and (28) into (26) yeilds

$$v_i = u_{ix} = k \sum_{j \in dir.} \left(\|\boldsymbol{p}_j\| \, z_{ij} Cos\, \theta_j \right) \tag{30}$$

as the control output for the robot's speed. The control output for the angular velocity becomes

$$\omega_i = \beta \text{ atan} \left(\frac{\sum_{j \in dir.} \left(\|\boldsymbol{p}_j\| \; z_{ij} sin\, \theta_j \right)}{\sum_{j \in dir.} \left(\|\boldsymbol{p}_j\| \; z_{ij} Cos\, \theta_j \right)} \right) \tag{31}$$

where θ_j is represented by the bearing values calculated by the sensor board.

This is the final version of the distance-based controller using decentralized sensor inputs.

Chapter 5.
Formation Acquisition Experiments

So far, the onboard sensors have been calibrated and their behavior characterized, communication protocols have been established for sensor usage, and that sensor information has been applied to a distance-based control law. Now these methods will be used in decentralized formation acquisition experiments. Identical experiments will be conducted using both the centralized and decentralized methods and their performances will be compared.

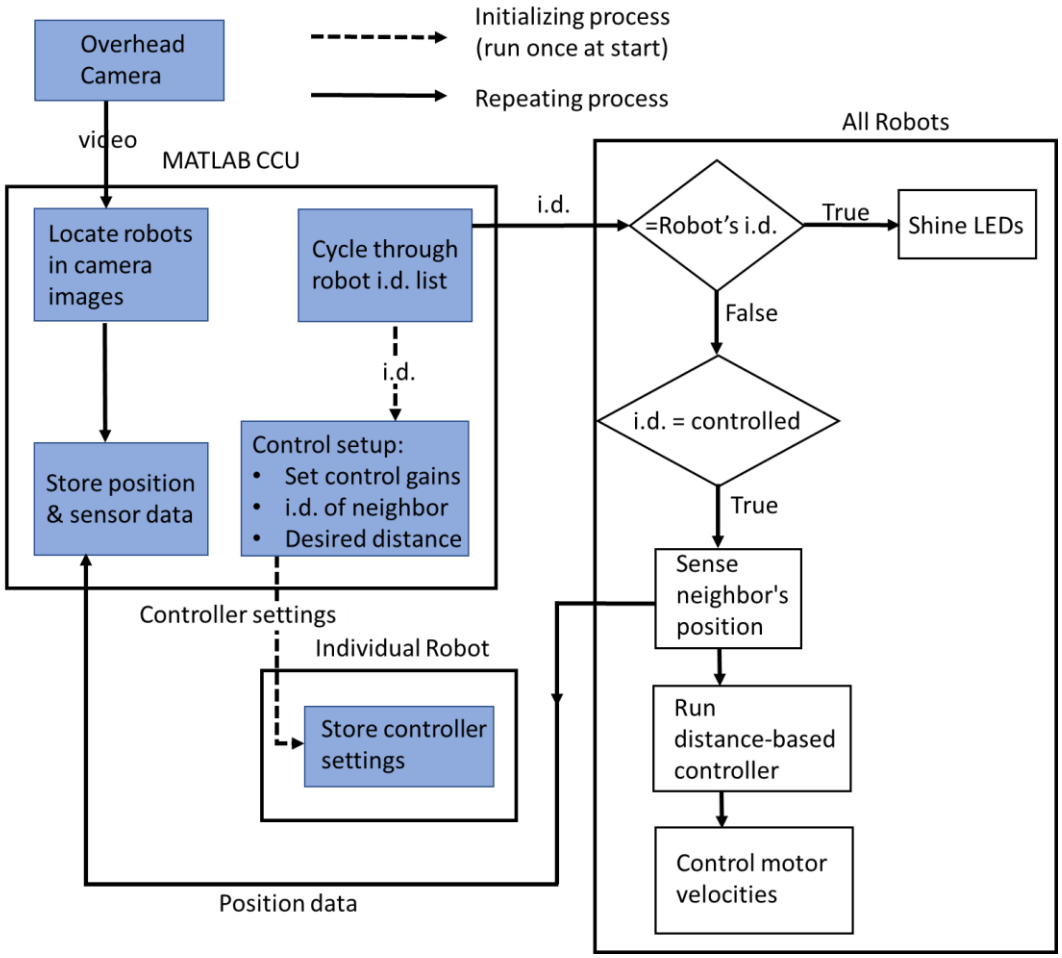

Figure 24. The decentralized controller begins by sending controller settings to all the individual robots, then cycling through the list of robots for LED use. Sending sensor position data back to the CCU is optional

Table 3. Summary of experiemnts

Number of Robots	Control method	Loop time (ms)
3	Centralized	180
4	Centralized	200
3	Decentralized	600
4	Decentralized	800
3	Decentralized with sensor data communication	1200

The experiments performed are summarized in

Table 3. Summary of experiemnts, and include two formation schemes and three different methods of implication. All experiments used the same linear and angular velocity control gains (k and β). All three and four robot experiments had the same directing scheme (which neighbor each mot is directed to) and the same initial conditions. The directing scheme is displayed in Figure 25 and the initial conditions are displayed in Figure 23.

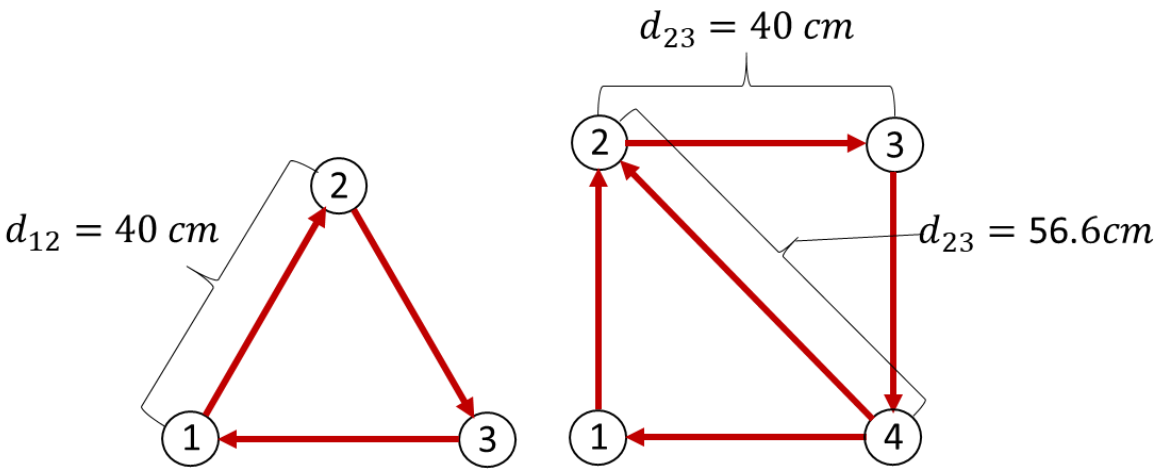

Figure 25. The directing scheme displays which neighbor each robot is directed to, the desired distances of each perimeter edge is 40 cm

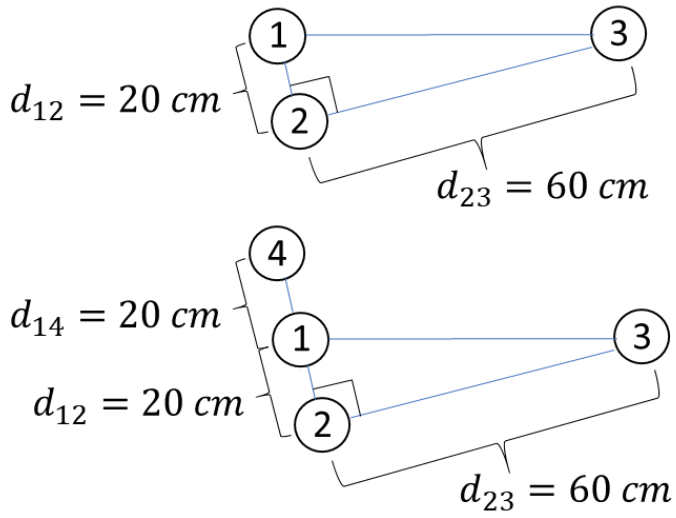

Figure 26. The initial conditions of the 3 and 4 robot systems are configured into right triangle segments

The performances of the control systems are based on the behaviors of the controlled distance errors. Metrics of evaluation include the steady state error averaged over all the edges, and the range of that average steady state error, this describes the quality of the final formation and the variance in that quality. The settling time and the integral of the absolute value of the error over the testing period will both serve to characterize the transient behavior of the formation controllers. These tests were each run several times to ensure that their behavior was repeatable.

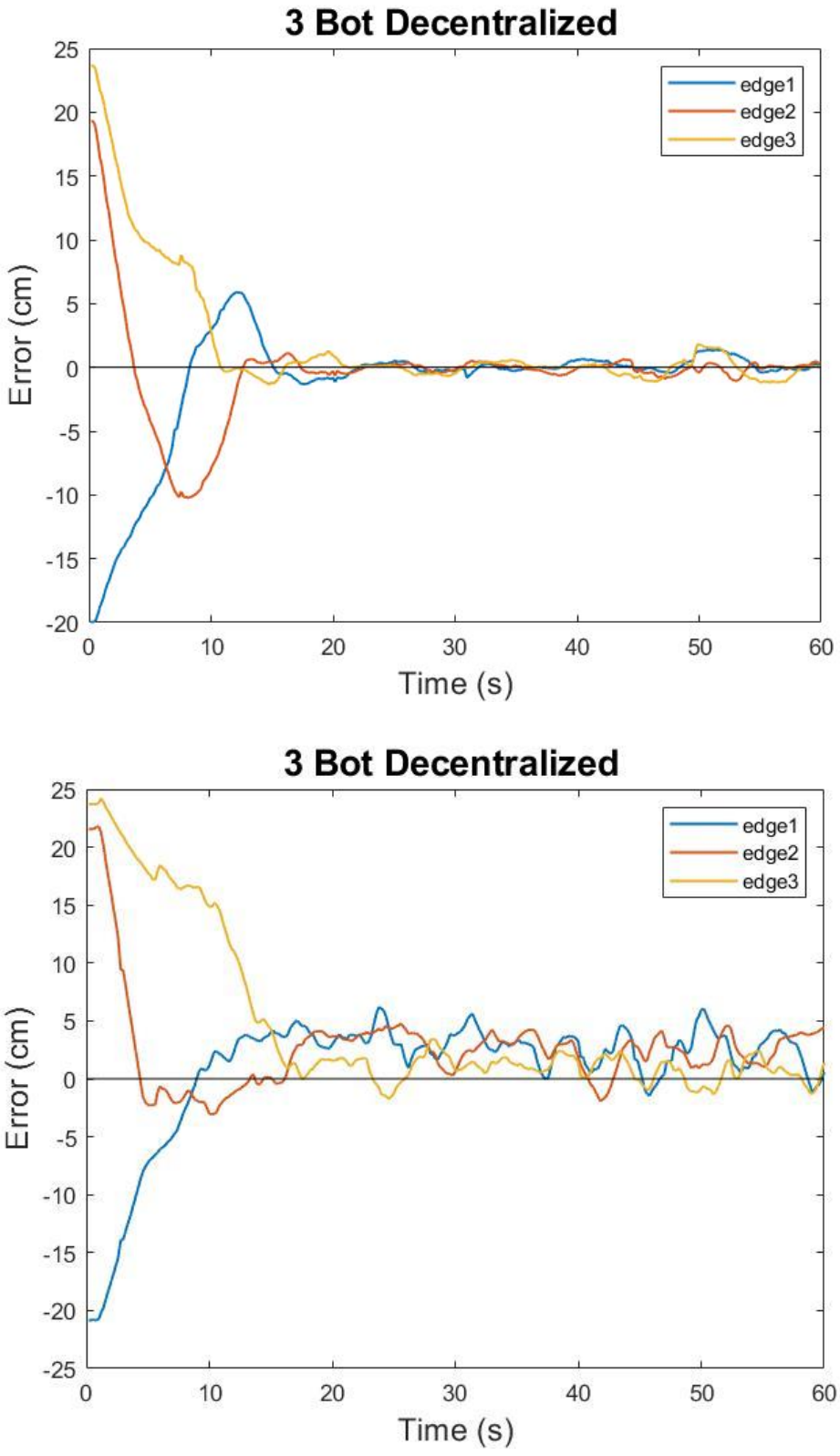

Figure 27. Triangle acquisition experiments with centralized and decentralized controllers

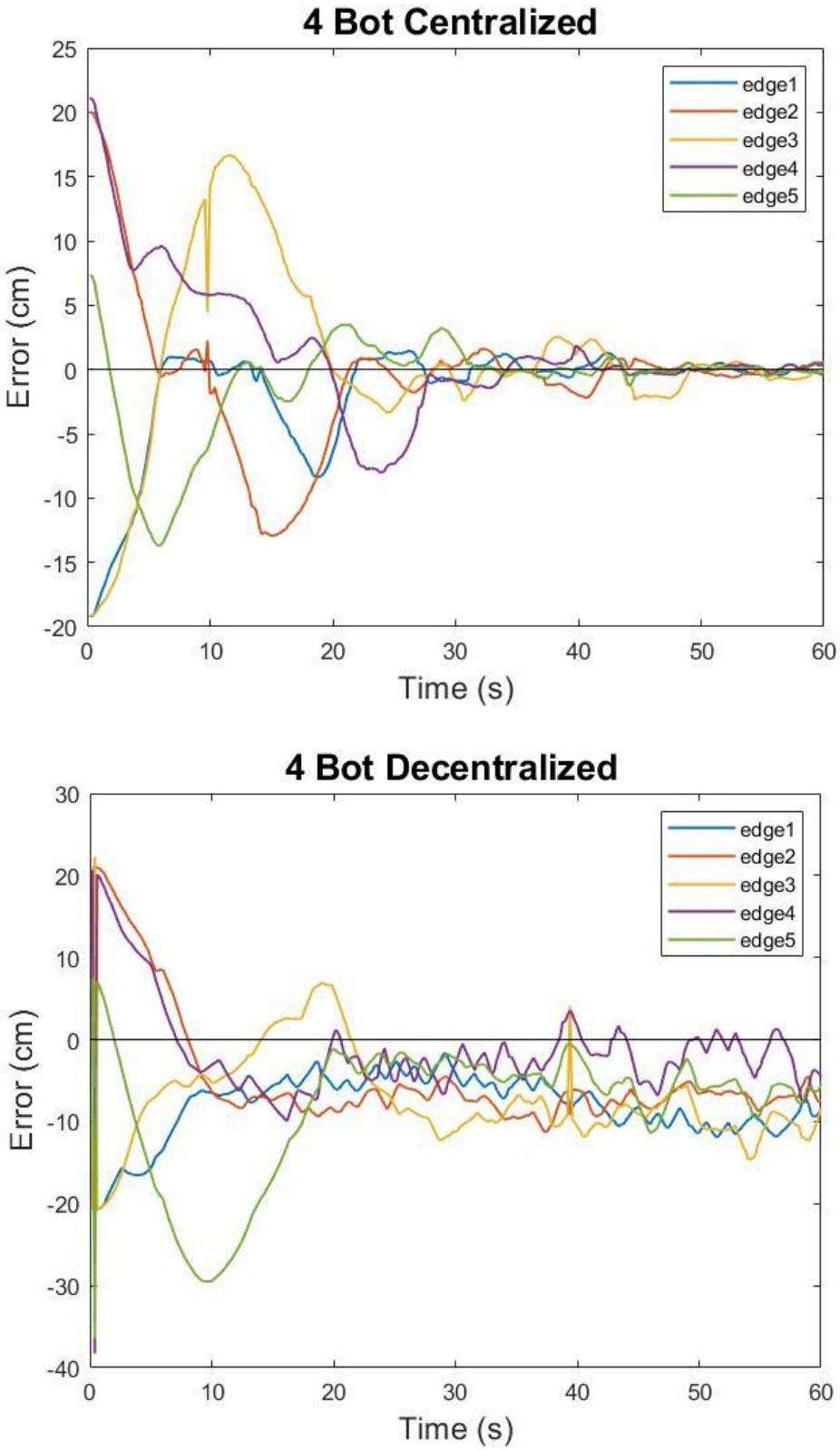

Figure 28. Square acquisition experiments with centralized and decentralized controllers

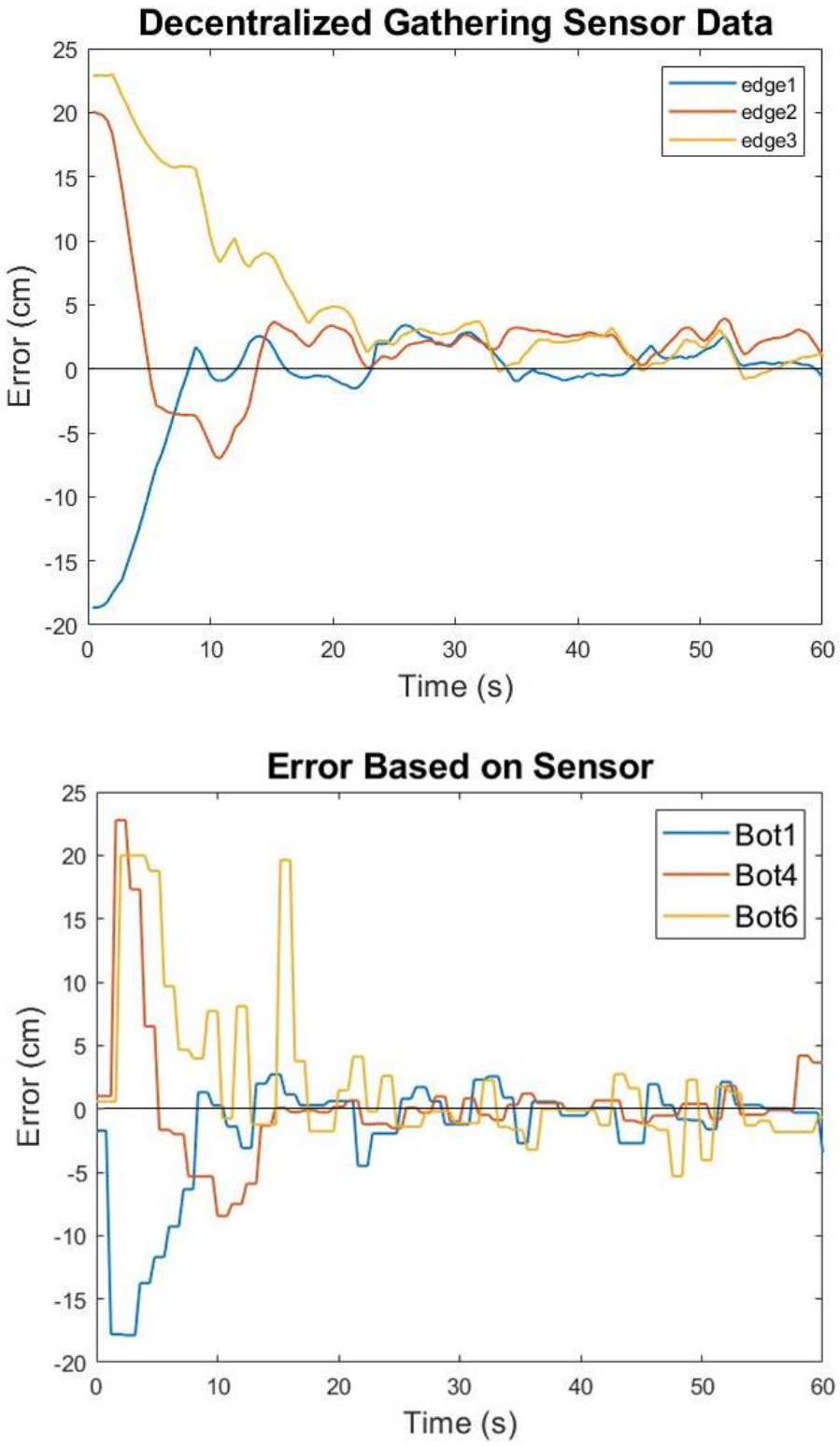

Figure 29. A decentralized triangle acquisition experiments was run while sending sensor data to the MATLAB station

The steady state error values differ from conventional examples of steady state errors in which a constant state is reached, and that state differs from the desired state, but remains constant because the controller's output matches a force acting on the system. In some respects, one could claim that this system never reaches steady state, but is in a constant transient state. However, the key distinction is that this lack of stability is not caused by varying external forces acting on the system, but by the uncertainty in the sensor measurements. With this in mind, the system can be considered to be in a steady state when its primary transient actions have been completed and it maintains the lowest error it is capable of. The settling time is the time at which steady state is reached, which is based on observational approximations.

It should also be noted that the errors of these controllers are interdependent because correcting a distance error requires moving the robot controlling that distance, and then creating a new distance error for the next robot. This interdependence is likely the reason for the periodic behavior in the errors since these errors ripple through the formation and are directed back around. This also means that inconsistent sensor measurements from any robot will perturb the entire system by creating new errors for the rest of the robots. This is the one of the challenges facing the scalability of formation controllers, it is not specific to decentralized sensing systems.

Table 4. The performance metrics for the controllers include the settling time, and quantities based on the steady state error averaged over all the edge errors

	3 Robots Centralized	4 Robots Centralized	3 Robots Decentralized	4 Robots Decentralized	3 Robots Long Step Time
Steady State Time (s)	22.5	49	22	26	23.5
Average Steady State Error (cm)	0.01	-0.03	1.45	-5.22	1.17
Max Average Steady State Error (cm)	0.83	0.16	2.39	-1.76	2.25
Min Average Steady State Error (cm)	-0.53	-0.36	-0.15	-7.60	0.12
Average Integrated Absolute Value of Error (cm·s)	97.98	149.01	183.50	373.18	172.15

The most surprising result from these experiments was that the decentralized 3 robot system with the longer step time outperformed the normal decentralized robot system by every metric except for the settling time. Recall that the step time was doubled in order for the robots to send their sensor data back to the MATLAB station. This means that the already lengthy cycle time of 600 ms was doubled, to a cycle time of 1200 ms. Despite this time increase, the average steady state error decreased from 1.45 cm to 1.17 cm, the range in the average steady state error decreased and the average integrated absolute value of the error decreased by 6 %. This is particularly dismaying because up until now it has been assumed that increased cycle time would be the greatest limitation on the scalability of this project. One possible explanation is that a loop time of this length allowed the controller to overshoot their desired distance enough for the sensors

to detect the overshoot and resulted in smoother behavior. This also explains why edge 2 exhibits more of an overshoot during the slower time step experiment.

A prominent difference between the 3- and 4-robot decentralized experiments is their average steady state errors. The three-robot experiment had an average steady state error of 1.45 cm and the 4-robot experiment had an average steady state error of -5.22 cm. This is puzzling because they were trying to acquire the same distances and robots 1 and 2 were directed to the same neighbors in both experiments. The issue wasn't caused by differing amounts of ambient IR light since these experiments were all taken under the same lighting conditions. This difference could be due to inconsistent output between the LEDs since the robots were facing different LEDs during the different experiments. Another possibility is that varying battery voltage changed the LED outputs between experiments, however this is unlikely because the robots were charged between every experiment in order to eliminate this possibility.

Compare the plots for the centralized and decentralized experiments in Figure 27Figure 28 and notice that there was overshoot in the centralized experiments but not in the decentralized experiments. The distance controllers used were identical for both experiments, and the increased step time for the decentralized controller would produce a delayed response and greater overshoot if all else was equal. The reason for this difference is the inconsistency in the sensing system. As the error value enters the range of uncertainty for the sensor board, an outlying sensor reading will have a more drastic effect on the resulting control output. For instance, if the error value is great, the uncertainty in sensor measurements isn't enough to change the overall direction of the control output, but if the error value is small enough, an outlying sensor measurement could reverse the robots control output and damp its progress as it approaches its target. This is especially true in the case of a unicycle robot because a change in the direction of

the controller's output results in a heading change for the robot which takes a longer time to correct. This can be seen in the error curves, at greater error values the robots moved with more certainty but as the error values begin to approach the range of sensor uncertainty the curves become jagged as the control output changes directions, damping the progress of the robots before they overshot.

A final observation is that the second plot in Figure 29. A decentralized triangle acquisition experiments was run while sending sensor data to the MATLAB station has error values centered around zero. This plot represents the error values based on the robot's sensor readings, which are the error values that the controller is based on. This demonstrates that the controller does an effective job bringing the steady state error down to zero based on its sensor readings, but the final performance of this controller is ultimately limited by the accuracy of the sensors.

Chapter 6.
Conclusions and Future Work

6.1. Conclusions

As mobile robotics applications expand, so must their region of operation. This project expanded the region of operation for the TIGERBots by making their operation independent of a centralized sensor and controller. This was accomplished with an onboard relative position sensor, an onboard controller, and a centralized communication method.

The relative position sensors were the key focus throughout this experiment because the quality of the experiment depended on their accuracy. Their hardware was set up to maximize the sensitivity of their signal responses. Methods for interpreting these signals were laid out and investigated using rigorous calibration and testing methods. A preliminary calibration setup was designed to set the gains for the cascading amplifier, and to normalize the signals of the photodiodes in order to make them interchangeable for future bearing and error calculations. An automated calibration system was designed to conduct calibration quickly and effectively. All of the necessary communication protocols were established to facilitate the robot's use of their sensors, and to transfer information between the robots and the MATLAB station.

A layout for the directed formation controller was given in the beginning of chapter 4, and it was applied to the information provided by the sensor board for use on a robot with unicycle-dynamics. Five different formation acquisition experiments were conducted, as listed in Table 3. Summary of experiemnts

Surprisingly, the triangle acquisition experiment behaved just as well after its loop time was doubled. Having such a long loop time added regularity to the oscillation of the steady state error, as the system had time to overshoot to an extent measurable by the IR sensors. The sensors

were inconsistent in their measurements, the variability in their steady state errors indicated an uncertainty value greater than the ±2.5 cm based on precision of the calibration. The controllers did a good job bringing the error to zero based on their sensor data, which solidifies the conclusion that the steady state errors in the formation distances are largely due to the inaccuracy of the sensors and not the system's loop time.

6.2. Future Work

This project is not yet fully decentralized, because its communication methods still depend on a centralized communication system. The last step in fully decentralizing this experiment would be to establish robot to robot communication and make the experiment completely independent of any external systems. Free from the time-consuming MATLAB control station, decentralized communication methods could also allow for much higher loop frequencies. This cannot correct inaccuracy in the sensor measurements, but it can reduce the effects of imprecision with the use of a moving average filter. This would likely result in smoother motion and smoother steady state error curves.

Future projects could also include other control schemes. Adding redundancies such as additional edges or undirected edges could help reduce the effect of sensor inaccuracies. Another redundancy that could prove useful is area-based controllers, where the system not only controls edge lengths, but also controls the angles between edges. For example, a triangular formation could be acquired by controlling the angles for two of its corners and the length of one of its edges. However, the inaccuracies for bearing measurements were very high in some cases, it would be a good idea to minimize this by using angular measurements at distances and bearing values that yield the greatest accuracy. For example, the bearing calculations performed the best at distances of 40 cm and when he photodiodes were facing the targets directly. Another possibility for sensor

redundancy is to use the onboard IMUs for dead-reckoning calculations in conjunction with the sensor board information [14]. This would be especially effective if the robots could communicate dead-reckoning information for relative position calculations.

An alternative use for the sensor boards would be obstacle avoidance. This could be done by using photodiodes and LEDs simultaneously to detect reflective obstacles, or by giving those obstacles their own infrared source.

References

1. Fernandez-Kim, V., *A Low-Cost Experimental Test Bed for Multi-Agent System Coordination Control*. 2019, Louisiana State University Gaton Rouge, LA.

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

3. I. Rekleitis, G.D., and E. Milios, *"Multi-robot collaboration for robust.* Ann. Math. Artif. Intell, 2001. **31**: p. 7–40.

4. Kwansik Cho, H.Y.S.a.J.P., *Accurate localization in short distance based on computer vision for mobile sensors.* International Multiconference on Computer Science and Information Technology, 2008.

5. F. Rivard, J.B., F. Michaud, and L. Dominic, *"Ultrasonic Relative Positioning for*

   Multi-Robot Systems. IEEE International Conference on Robotics and Automation, 2008.

6. A. Kohlbacher, J.E., K. Acres, H. Chung and J. C. Barca, *A low cost omnidirectional relative localization sensor for swarm applications*, in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. 2018: Singapore.

7. James F. Roberts, T.S.S., Jean-Christophe Zufferey and Dario Floreano, *2.5D Infrared Range and Bearing System for Collective Robotics*, in *9 IEEE/RSJ International Conference on*

   Intelligent Robots and Systems. 2009, IEEE: St. Louis

8. Roberts, J.F., et al., *3-D relative positioning sensor for indoor flying robots.* Autonomous Robots, 2012. **33**(1-2): p. 5-20.

9. Pugh, J., et al., *A Fast Onboard Relative Positioning Module for Multirobot Systems.* IEEE/ASME Transactions on Mechatronics, 2009. **14**(2): p. 151-162.

10. Martinoli, I.D.K.a.A., *A scalable, on-board localisation and.* Sens. Rev., 2004. **24**(2): p. 167-180.

11. M. de Queiroz, X.C., and M. Feemster, *Formation Control of Multi-Agent Systems:*

    A Graph Rigidity Approach. 2019, Hoboken, NJ

    Wiley.

12. L. Krick, M.E.B., and B.A. Francis, *Stabilization of infinitesimally rigid formations of multi-robot networks.* Intl. J. Contr., 2009. **83**(3): p. 423-439.

13. P. Zhang, M.d.Q., M. Khaledyan, and T. Liu, *Control of Directed Formations Using Interconnected Systems Stability.* ASME Journal of Dynamic Systems, Measurement, and Control, 2019. **141**.

14. S. Thurn, W.B., D. Fox, *Probabilistic Robotics*. Vol. 141. 2005, Cambridge, MA: MIT Press.

Vita

Steven Williams grew up in the foothills of the Adirondack Mountains of upstate New York. After graduating for Clinton High School in 2014, he enrolled in the State University of New York at Geneseo, where he earned his Bachelor of Science degree in Physics in Spring 2018, with a minor in Mathematics. In Fall 2018 he began pursuing a Ph.D. in Mechanical Engineering at Louisiana State University as a researcher in the Composite Manufacturing Lab. He left Material Science in Spring 2019 to join the iCORE lab and pursue a Master of Science degree in Mechanical Engineering with research in Robotics and Control Theory. He plans to receive his Master's degree in May 2021 and pursue a career in IoT or imbedded systems development.