2015

Robot Localization Obtained by Using Inertial Measurements, Computer Vision, and Wireless Ranging

William Baker
University of Vermont, willcbaker@gmail.com

Robot Localization Obtained by Using Inertial Measurements, Computer Vision, and Wireless Ranging

A Thesis Presented

by

William Baker

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fullfillment of the Requirements
for the Degree of Master of Science
Specializing in Electrical Engineering

October, 2015

Defense Date: May 15, 2015
Thesis Examination Committee:

Jeff Frolik, Ph.D., Advisor
Joshua Bongard, Ph.D., Chairperson
Stephen Titcomb, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

Abstract

Robots have long been used for completing tasks that are too difficult, dangerous, or distant to be accomplished by humans. In many cases, these robots are highly specialized platforms - often expensive and capable of completing every task related to a mission's objective. An alternative approach is to use multiple platforms, each less capable in terms of number of tasks and thus significantly less complex and less costly. With advancements in embedded computing and wireless communications, multiple such platforms have been shown to work together to accomplish mission objectives. In the extreme, collections of very simple robots have demonstrated emergent behavior akin to that seen in nature (e.g., bee colonies) motivating the moniker of ”swarm robotics” - a group of robots working collaboratively to accomplish a task. The use of robotic swarms offers the potential to solve complex tasks more efficiently than a single robot by introducing robustness and flexibility to the system.

This work investigates localization in heterogeneous and autonomous robotic swarms to improve their ability to carry out exploratory missions in unknown terrain. Collaboratively, these robots can, for example, conduct sensing and mapping of an environment while simultaneously evolving a communication network. For this application, among many others, it is required to determine an accurate knowledge of the robot's pose (i.e., position and orientation). The act of determining the pose of the robot is known as localization. Some low cost robots can provide location estimates using inertial measurements (i.e., odometry), however this method alone is insufficient due to cumulative errors in sensing. Image tracking and wireless localization methods are implemented in this work to increase the accuracy of localization estimates. These localization methods complement each other: image tracking yields higher accuracy than wireless, however a line-of-sight (LOS) with the target is required; wireless localization can operate under LOS or non-LOS conditions, however has issues in multipath conditions. Together, these methods can be used to improve localization results under all sight conditions. The specific contributions of this work are: (1) a concept of 'shared sensing' in which extremely simple and inexpensive robots with unreliable localization estimates are used in a heterogeneous swarm of robots in a way that increases the accuracy of localization for the simple agents and simultaneously extends the sensing capabilities of the more complex robots, (2) a description, evaluation, and discussion of various means to estimate a robot's pose, (3) a method for increasing reliability of RSSI measurements for wireless ranging/localization systems by averaging RSSI measurements over both time and space, (4) a process for developing an in-field model to be used for estimating the location of a robot by leveraging the existing wireless communication system.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Dr. Jeff Frolik for his continuous support, patience, motivation, and great wisdom. His guidance helped me throughout the duration of my research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Stephen Titcomb and Dr. Joshua Bongard, for their insightful comments and encouragement, but also for their challenging questions which intended me to widen my research from various perspectives.

Additionally, I would like to thank The Vermont Space Grant Consortium for providing financial support and the funds required to fulfill the research discussed in this thesis.

Last but not the least, I would like to extend my gratitude to the staff members of the graduate college who's work behind the scenes ensured the proper delivery of this thesis.

I would like to dedicate this work to my family and friends who have continuously supported me in my research efforts. I would like to thank my mother, who has provided me with courage, my grandfather, who has encouraged me to think outside the box, my father, who has helped me develop the practical skills required to leverage knowledge as a tool, my siblings, for their comfort and entertainment, my grandmothers, who have always shown me the true meaning of ‘welcome home’, and all of my friends who have marched beside me along the way.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The use of unmanned spacecraft, such as mobile robots, has long been a key component of NASA's space exploration program. Of recent note are the successful 2004 Mars explorations by the *Spirit*, *Opportunity*, and *Curiosity* (2012) rovers along with NASA's involvement with the upcoming European Space Agency's (ESA) *ExoMars Rover* (expected 2018). However, developing and launching such complex robotic platforms comes at a significant cost. NASA's Mars Exploration Rover Mission (MER) that operated the *Spirit* and *Opportunity* craft was an ~$1B venture, while the Mars Science Laboratory (MSL) program, which resulted in *Curiosity*, cost twice that. Ensuring the robust and reliable operation of these platforms and their instruments results in much of this cost. Still, these are single objects prone to design errors or other failures (e.g., Mars Climate Orbiter, ne *Surveyor*).

An alternative approach for exploration is to use multiple platforms, each significantly less complex and less costly. With advancements in embedded computing and wireless communications, multiple such platforms have been shown to work together to accomplish mission tasks [1]. In the extreme, collections of very simple robots (e.g., Bristlebots [2, 3] and Kilobots [50]) have demonstrated emergent behavior akin to that seen in nature (e.g., bee colonies) motivating in the moniker of *swarm robotics*.

Additionally, a single robot can only sense its environment from a single viewpoint, even when it is equipped with a large array of different sensing modalities, whereas a team of robots can perceive its environment from multiple disparate viewpoints and exchange sensor information [42]. Distributed sensing can result in increased coverage area or increased resolution of said area. A clear example of this is surveillance and/or the monitoring of an environment (i.e., region-of-interest). The usefulness of a sensor (e.g., camera) or array of sensors each with individual operating constraints (e.g., resolution, field-of-view, etc.) are often limited by their physical location; for example, a camera operating in a cluttered environment is unable to provide a full description of the environment due to occlusions from objects. Conversely, a multi-robot system equipped with the same sensors could offer simultaneous coverage over a larger area due to their spacial distribution. There are many other tasks for which the concept of distributed viewpoints may be advantageous. For example, the researchers in [47] use a network of swarm robots equipped with gas sensors to identify the source of an odor plume that converges more quickly to a solution than that of a single robot. The spacial distribution between swarm robots in this example is crucial to detecting the concentration of gas at various locations, thus enabling the swarm to find the source of the plume. Note, in order to accomplish this task with a single robot efficiently, it would require the robot to be very large such that sensors could be placed at distances far apart. Platforms that can leverage swarm behavior are already in development for space exploration (e.g., the intended small rovers on *Asteroid Explorer Hayabusa 2* [4] and a recent deployment from the International Space Station of a 28 CubeSat network, *Flock 1*, by Planet Lab [5]).

In mobile robotics, a common requirement is to determine an accurate knowledge of the robot's pose. The robot's pose is defined as the robot's orientation and position [7]. The orientation of the robot includes the physical state (e.g., the state of an extended actuator or arm) as well as the heading, or direction, the robot is traveling. The robot's position describes where the robot is in space, either relative to an object or defined by some set of global coordinates (i.e. absolute). Determining the pose of the robot is often referenced as: *the localization problem in robotics* [8].

Knowledge of the robots pose, either absolute and/or relative, is critical for distributed robotic applications that require robots to share sensor information (e.g., exploration, mapping, surveillance, etc.). Without this knowledge, it becomes impossible to interpret the sensor data geospatially and integrate with data obtained by other swarm members [45]. For example, NASA's Magnestospheric Multiscale Mission (MMS) uses four identical spacecraft flying in a tetrahedral formation [37] to study the Earth's magnetosphere. The relative distance and formation between these satellites is required to correctly interpret the data collected from the individual crafts.

1.2 Localization Methods

Robot localization is the act of determining the robot's pose in a defined coordinate system (e.g., x-y-z, roll-pitch-yaw, coordinates in a room). Techniques of localization include odometry, inertial navigation, magnetic compasses, active beacons, global positioning systems (GPS), landmark navigation, and model matching; summarized by the researchers in [27]. Localization of multi-robot systems can be roughly classified into two main categories: absolute and relative positioning systems [34].

Absolute localization determines the robot's position in a global coordinate framework, where the world-frame is fixed and the robot's pose can be defined through translations and rotations in reference to the world's origin. This technique offers robust solutions for scenarios in which *a priori* knowledge (e.g., a map) of the environment is provided. Alternatively, a relative positioning system may be used to define a robot's pose in reference to another robot, object, or its previous self. In this scenario, each robot views itself as the fixed reference frame and all measurements are relative to its own pose (either initial, previous, or current). Relative positioning is often prone to both systematic errors (e.g., unequal wheel diameters, uncertainty about the effective wheel-base, limited encoder resolution, etc.) and non-systematic errors (e.g., wheel slippage, irregularities in the floor, etc.). Relative position measurements often use the method of *dead reckoning.* Dead reckoning is the process of calculating the pose of a robot by using a previously determined position. This is most simply accomplished by use of odometry, i.e., the integration of incremental motion information over time. Most simply, physical odometry is often based on the assumption that wheel revolutions can be translated into linear displacement relative to the floor [55]. Absolute positioning addresses some of these errors, in that it is not susceptible to error accumulation, however may still suffer from the effects of sensor resolution and is often more computationally expensive and less energy efficient than relative positioning systems. It should also be noted that one may simply derive a set of relative pose estimates given a set of absolute estimates.

The practical usefulness of a localization system depends on the context in which it will be used; different techniques must be considered in order to match the desired accuracy, cost, and environmental conditions for each application. For instance, GPS is the accepted standard for positioning on Earth. GPS requires a line of sight (LOS) communication with satellites to calculate position. However, due to signal attenuation/blockage from large objects in built or natural environments, GPS is not suitable for all applications. In this work, the context is that GPS is not available, for example in dense cities with tall buildings, indoors, underground, or on lunar or planetary surfaces.

The use of computer vision is commonly used for localization indoors [10]. The researchers in [35] use a fixed and distributed multi-camera system to achieve an absolute indoor positioning system for multiple robots. An overhead camera system has also been shown to produce similar results [36]. Computer vision is often coupled with other sensors to increase accuracy in localization estimates. The researchers in [26] use a combination of vision and odometry to better estimate the location of a mobile robot using an extended Kalman filter. The researchers in [28] tested computer vision coupled with inertial measurement devices as an alternative for GPS denied environments. While these computer vision techniques may offer robust localization where GPS is not accessible, they require line-of-sight (LOS) to the target. Thus, the work presented in this paper aims to explore a suitable method for localization in both LOS and non-line-of-sight (NLOS) scenarios.

1.3 Context of Research

The research uses low-cost, commercial-off-the-shelf (COTS) components to build a heterogeneous robotic swarm capable of localization in GPS denied environments for both LOS and NLOS scenarios. This work presents such a localization method by using inertial measurements, computer vision, and wireless ranging.

This work leverages two tiers of robotic platforms, where one tier can control and manipulate another tier for the purpose of environmental exploration and navigation. These two tiers are referred to as high-level and low-level robots, indicating their degree of on-board computational power and sensing capabilities. The first platform, the high-level robot, is a four-wheeled rover (Figure 1.1) with sufficient energy and computation resources to enable computer vision for obstacle avoidance, target seeking, remote sensing, etc. and the capability to command/control a fleet of less sophisticated robots. The second category of platforms, the low-level robots, represent these lesser-capable robots of the swarm and are only capable of rudimentary sensing, wireless communications, and imprecise movement due to their compact form factor. These robots are shown with both spherical and cylindrical bodies in Figure 1.1. Together, the robotic swarm is tasked with navigating an environment. The localization techniques presented in this work are used to define a coordinate system among the robots relative to one another for navigational purposes.

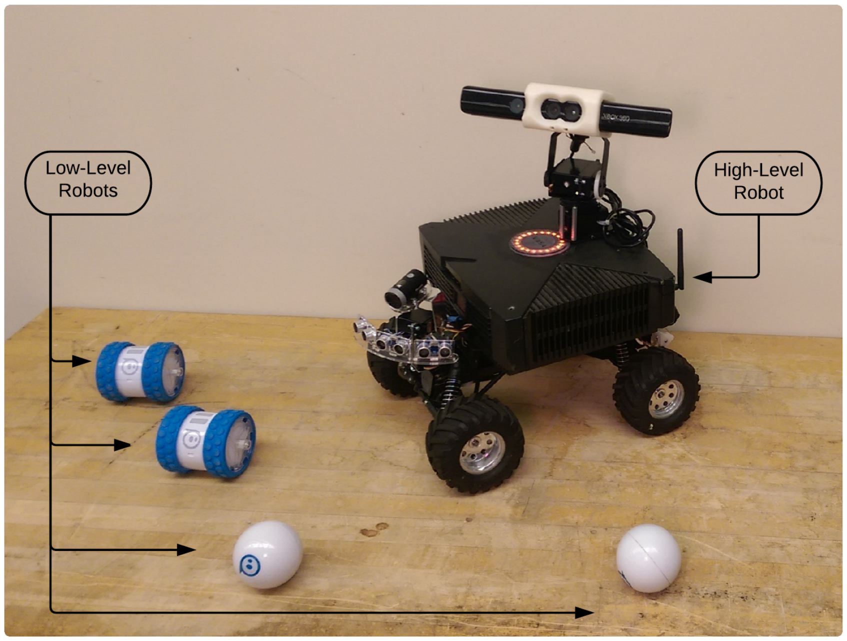

Figure 1.1: The heterogeneous swarm of robots used in this research. High-level robot (top right) and simple, commercialy available low-level robots (left and bottom)

1.4 Thesis Contributions

The contributions of this work are fourfold as described below.

1. This work introduces a concept of 'shared sensing' in which extremely simple and inexpensive robots with unreliable localization estimates are used in a heterogeneous swarm of robots in a way that increases the accuracy of localization for the simple agents and simultaneously extends the sensing capabilities of the more complex robots. This method also enables the swarm to investigate hazardous and unknown terrain by leveraging the mobility of a low cost, potentially expendable robot instead of endangering a significantly more complex, and thus expensive robot.

2. This work outlines various means to estimate a robot's pose. The accuracy of each

method is evaluated. The work also addresses the benefits of using multiple sensors simultaneously.

3. This work introduces a method for increasing reliability of RSSI measurements for wireless ranging/localization systems by averaging RSSI measurements over both time and space. This method addresses large fluctuations in RSSI measurements due to the natural propagation of radio signals in multipath environments as well as the non-ideal communication system performances that can be largely attributed to the orientation of the robot.

4. The final contribution of this work is a process for developing an in-field model to be used for estimating the location of a robot by leveraging the existing wireless communication system. This process provides a way to calibrate a model to be used in an environment where prior knowledge of the wireless channel performance is not required.

1.5 Thesis Outline

This introduction is followed in Chapter 2 by a presentation on using a group (i.e., swarm) of heterogeneous robots working collaboratively to achieve a common goal. The physical hardware and software used for each type of robot as well as the role of each agent are then detailed and described.

In Chapter 3, a method of localizing a robot using only inertial measurements is introduced. This method is used on the low-level robotic platforms to test the accuracy of the default localization method available to the low-level robot.

Localization using a vision sensor is discussed and explored in Chapter 4. We present a method of localization using the vision sensor to locate and track a low-level robot(s) as it moves about an environment. The high-level robot is used to relatively localize the low-level robots and other swarm members within the environment.

In Chapter 5, we present a wireless localization technique using Bluetooth. We discuss the chosen model for determining distance from radio signal strength over a wireless channel and present known limitations and new mitigation techniques. These techniques are implemented in test, results are analyzed and discussed. Included in this chapter is an outline for a multi-sensor localization process that combines the three aforementioned localization techniques.

The thesis is concluded in Chapter 6, where we discuss and summarize the results from the three different localization modalities presented in this work and outline future work and suggestions to continue research in this area.

Chapter 2

Swarm Robotics

2.1 Introduction to Swarms

In this chapter, the concept of using multi-robot systems or swarms is discussed. A robotic swarm can be defined as a group of agents (i.e., robots) working collaboratively towards accomplishing a common goal. As will be discussed shortly, a system comprised of multiple robotic platforms enables each agent to be significantly less complex and less costly. The swarm is a distributed system that introduces robustness, flexibility, and scalability into robotic missions. The swarm robotic system should be able to continue to operate, although at a lower performance, despite failures in the individuals, or disturbances in the environment [6] (e.g., natural disaster resulting in spatial separation of the individual agents). This requirement ensures the team members are able to respond robustly and reliably to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention [45]. Therefore, a robotic swarm should allow for members to be added, modified, or replaced as requirements and tasks change over time.

Multi-agent systems offer considerable advantages in comparison to single unit systems. Physically separate entities permit simultaneous sensing and actuation in spatially different locations [41], reconfigurability of the system, redundancy, and sometimes the ability to achieve superlinear performance (i.e., the whole is greater than the sum of its parts) via division of labor [43, 44]. For example, multiple agents have been shown to work cooperatively to manipulate or carrying large objects [46] and scale obstacles [48]. This tactic enables the load to be distributed over several robots, enabling each robot to be much smaller, lighter, and less expensive [45].

The robotic swarm in this work consists of heterogeneous agents. A heterogeneous structure allows for individual specialization in completing a task or subset of tasks. Specialization is achieved by exploiting the nature of a specific robot type. Instead of equipping every robot with every sensor and computation or communication capabilities, each robot may instead be built for a particular aspect of the task such that they are more adept to fulfill certain roles - such as transportation, resource extraction, energy harvesting and storage, physical manipulation, material identification/testing, environmental sensing, high resolution photography, extended communication, navigation planning, etc. The various actuation and sensing capabilities that make up each robot type may be redundant or complementary to other members; this allows the robots to work together in such a way that the efficiency of the swarm is greater than if the different robot types worked independently and in parallel without mutual cooperation [48]. For several tasks such as distributed search [38], distributed coverage [39] (e.g., mine sweeping, floor cleaning, harvesting, lawn mowing, etc.), and movement in formation [40], knowledge of one's own location and/or that of neighboring swarm members is required [41]. As discussed later in this work, the requirement to localize can be carried out by means of various sensing techniques which may be custom tailored to fit the architecture of each robot.

The structure of the robotic swarm presented in this work leverages two tiers of robotic platforms, where the first tier (i.e., high-level robots) is able to command/control a fleet of less sophisticated robots which make up the second tier (i.e., low-level robots). By nature, these low-level agents are small, mobile robots with limited capabilities, and less computational complex, allowing them to be equipped with smaller processors and motors that in turn consume less power. The size and mobility of these robots enables them to travel to places that the other robots cannot. Furthermore, the simplicity of their hardware enables these robots to be potentially expendable. Simply stated, these low-level robots act as a distributed sensor platform that are remotely controlled by a leader (i.e., the high-level robot) who performs the high-level mission planning. By constructing a tiered (i.e., heterogeneous) system, the average size and cost per robot can be minimized, thus many robots can be built and deployed in large numbers to achieve dense sensing coverage, and fault tolerance [42] that would otherwise require many expensive and larger robots.

When a team of robots is composed of different platforms, carrying different types or resolutions of sensors and thus having different capabilities and accuracies for self-localization, the quality of the localization estimates will vary significantly across the individual members [36]. The collaboration (i.e., flow of information) among the swarm members establishes a form of 'sensor sharing' and can thus enable the swarm to overcome the limitations of individual agents and improve the overall accuracy of localization within the swarm. This increase in accuracy will allow the swarm to accomplish more difficult tasks that depend on accurate localization, such as surveillance, mapping, and/or exploration.

A block diagram of the swarm is shown in Figure 2.1. The diagram shows that the computationally difficult tasks, such as objective planning, collaborative decision making, and object tracking are completed by the high-level robots. The low-level robots are simply commanded and controlled by the high-level robots, as if an extension of a high-level robot. The low-level robots may be shared among multiple high-level agents in order to increase sensing capabilities. The rudimentary sensing and positional awareness of the low-level robots are relayed back to the high-level agents to be assessed and improved. Any high level agent may subscribe to information from any low-level agent, this information is in turn used to determine an improved estimate of location for the low-level agent. This estimate allows the swarm to localize all of its members, including the initially inaccurate low-level agents. This information is used by the high-level agents to aid the swarm as it steps through through the various stages of the mission objective until completion.

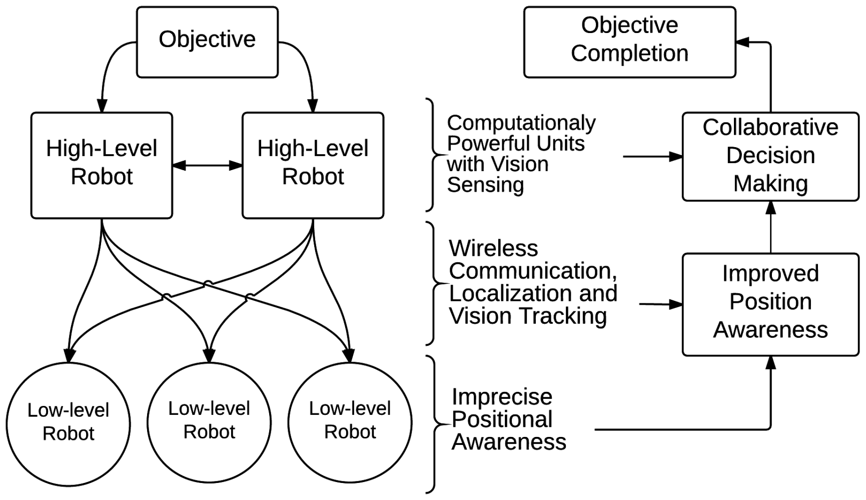
Objective
High-Level Robot
High-Level Robot
Low-level Robot
Low-level Robot
Low-level Robot
Objective Completion
Computationaly Powerful Units with Vision Sensing
Collaborative Decision Making
Wireless Communication, Localization and Vision Tracking
Improved Position Awareness
Imprecise Positional Awareness

Figure 2.1: Functional overview of the robotic swarm

2.2 High-Level Robots

For this research, the first platform is a four-wheeled rover (Figure 2.2) with sufficient energy and computation resources to enable computer vision for obstacle avoidance, target seeking, remote sensing, etc. and the capability to command/control a fleet of less sophisticated robots which make up the swarm.

2.2.1 Hardware

The hardware on-board the high level robots features a customized mobile processing unit (MPU) from Logic Supply Inc. The processor is a 64-bit Intel Core i5 operating at 3.2 GHz

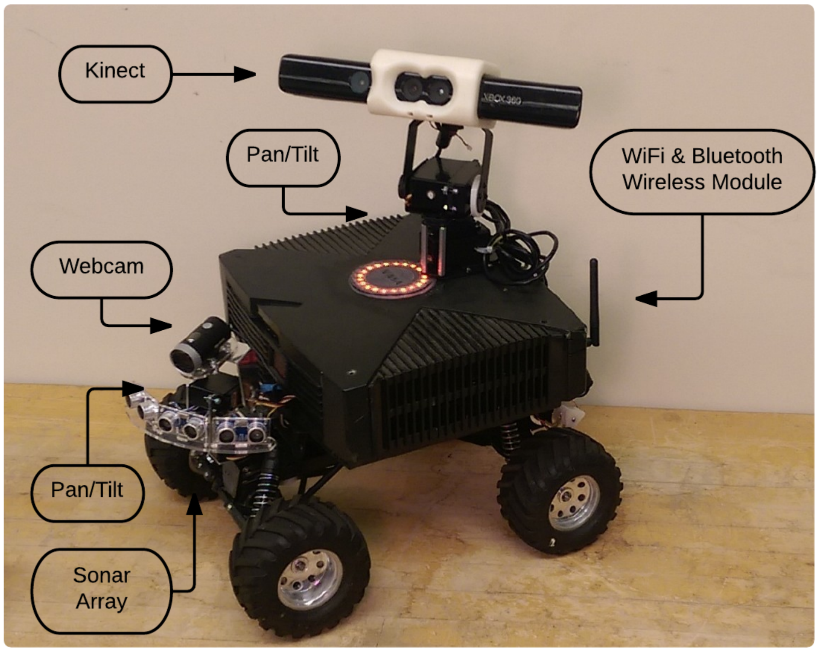

Figure 2.2: The high-level robot in this swarm

and is supported by 8GB of RAM.

As outlined in Figure 2.3, the high level robot caries a suite of sensors that collect data about the environment. The primary vision sensor is a Microsoft Kinect which provides the robot with RGB-Depth images. The robot is also outfitted with a Logitech USB Webcam for secondary video input. These vision sensors are controlled and managed directly by the MPU. The high level robot is also equipped with a power monitor that measures the amount of energy consumed as well as an array of ultrasound range sensors, an IMU, and a temperature sensor. These sensors are controlled by an Arduino microcontroller and data collected is forwarded to the MPU for processing. The MPU passes any motor control commands to the Arduino which controls all hardware-level drivers. The hardware level drivers include a L293D motor controller for driving the robot forward/revers, and a PCA9685 PWM driver which drives the pan-tilt mechanisms and steering servos. The robot is powered from a single 6S LiPo battery pack and all voltage regulation is performed by the battery monitor.

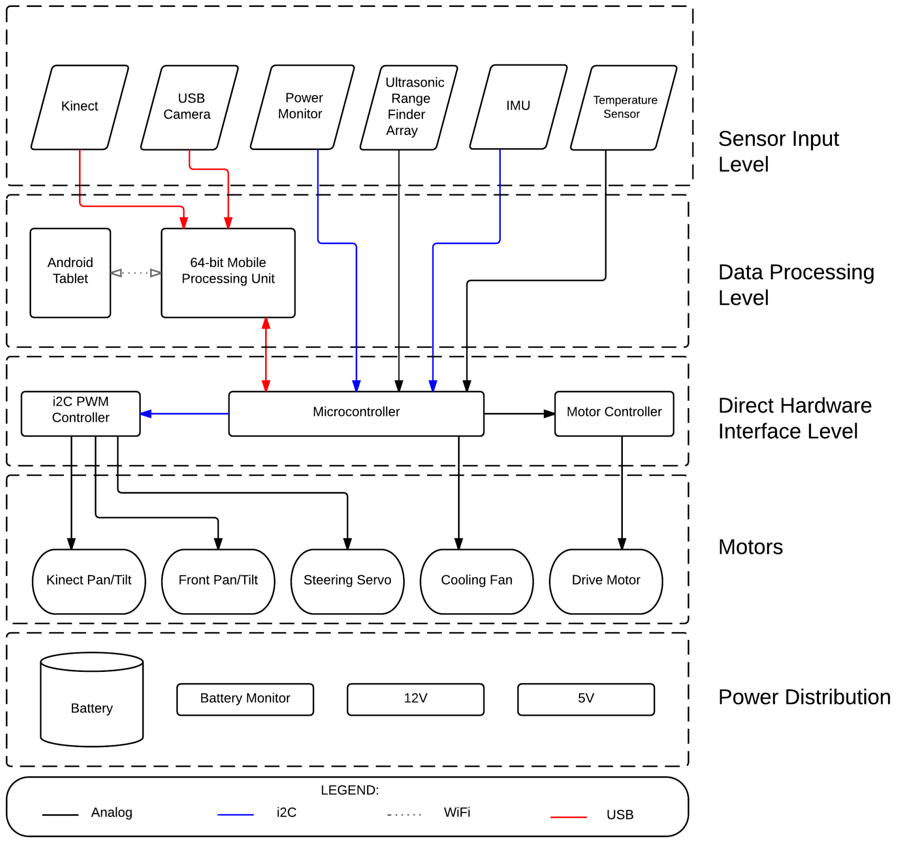

Figure 2.3: Hardware overview of the high-level robot.

2.2.2 Software

The software for high-level robots is built on Ubuntu 14.04 running the Indigo Igloo distribution of Robot Operating System (ROS) [54]. ROS is used for handling all sensory inputs and output; the ROS packages are written in Python and C++. The ROS node graph of the high-level robot is shown in Figure 2.4.

The ROS launch file for the high-level robot initiates the robot at start-up. The *OpenNI*

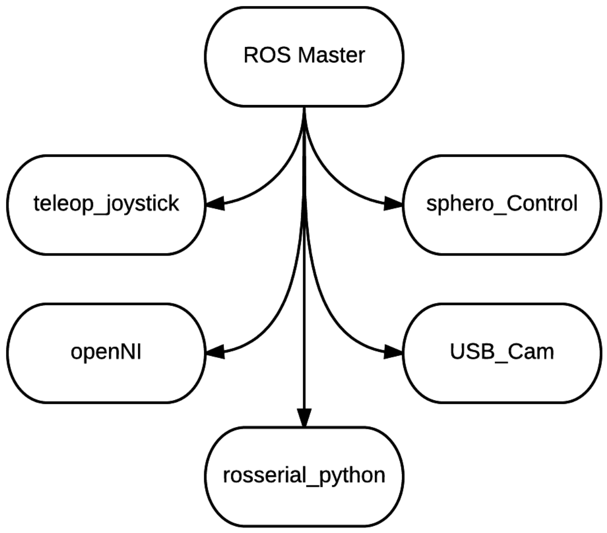

Figure 2.4: ROS node graph depicting all running nodes.

node uses Freenect drivers to interface with the Microsoft Kinect; the point clouds generated by this node are used for 3D object recognition described in Section 4.3. The *sphero_Control* node handles all Bluetooth communication to interface with the low-level Sphero robots. The USB webcam node publishes raw and compressed images from the webcam at 30 Hz. The *teleop_joystick* node is used to translate joystick messages into motor controls for remote operation. The *rosserial_python* node bridges the ROS platform to the Arduino microcontroller using UART serial communication. The Arduino is running a *rosserial* node at 100 Hz that polls the IMU, power monitor, ultrasound array, and temperature sensors and publishes the values to the ROS Master. This node also subscribes to all motor commands and outputs these values to the desired motor.

2.3 Low-Level Robots

The low-level robots are commercially available Sphero Ollie (Figure 2.5-left) and Sphero 2.0 (Figure 2.5-right). The Sphero robots are unmodified COTS units and equipped with an internal inertial measurement unit (IMU), Bluetooth communication module, and internal batteries with supportive circuitry for inductive charging. The Sphero 2.0 robot features a compact spherical form factor with native Bluetooth 2.0 communications. The Sphero Ollie has a similarly compact form-factor but features a differential drive system and Bluetooth 4.0. The Sphero robots' compact packaging allows the device to navigate complex terrains, survive high falls, and access tight locations that the wheeled high-level robot cannot; however, this pacakging also leads to non-uniform communications performance as will be detailed in Chapter 5.

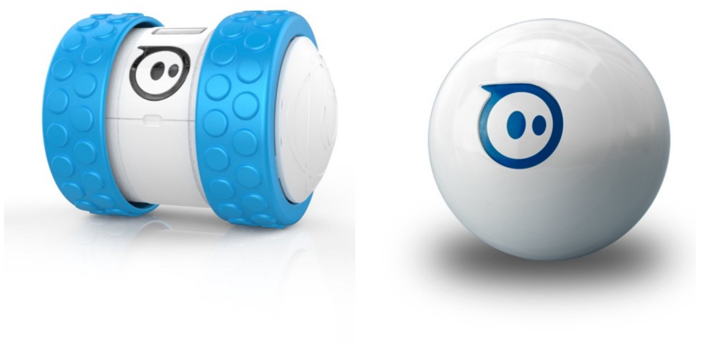

Figure 2.5: The low level robots. Left: Sphero Ollie, Right: Sphero 2.0

The low-level robots are controlled using a custom developed Android application that utilizes the official Sphero SDK to facilitate communication with the robot and couples the low-level robot to the ROS Master running on the high-level robot. The SDK allows for basic input command functions such as setting the robot's heading, velocity, and color. The control of the color may be used to communicate the state of the robot, as well as distinguish it from otherwise identical robots. These commands are published by the ROS Master on the high-level robot, thus essentially rendering the low-level robot as a remote appendage of the high-level robot. The low-level robot in turn provides rudimentary localization estimates using an inertial navigation system; these estimates are published to the ROS Master for processing. An overview of the hardware enclosed in the low-level robots are shown in Figure 2.6.

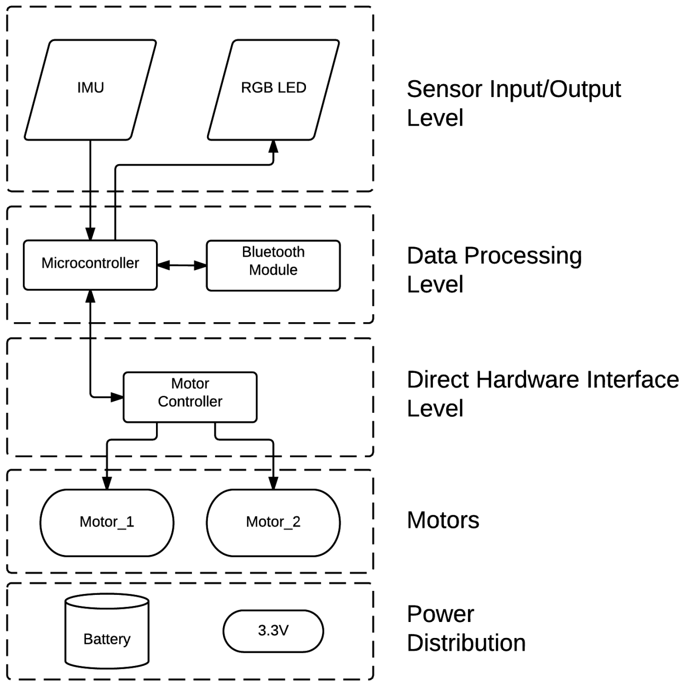

Figure 2.6: Hardware overview of the low-level robot

The inside of the low-level robot is shown in Figure 2.7. The main processor on the low-level robot is a small unmarked microcontroller. The robot features an RGB LED that is mounted vertically to illuminate portions of the chassis that are visible from the outside. The Bluetooth module is used for communication and has a large plate antenna mounted vertically on the left side of the robot. The microcontroller interprets data received by the Bluetooth module and drives the LED and motors accordingly. The IMU collects inertial

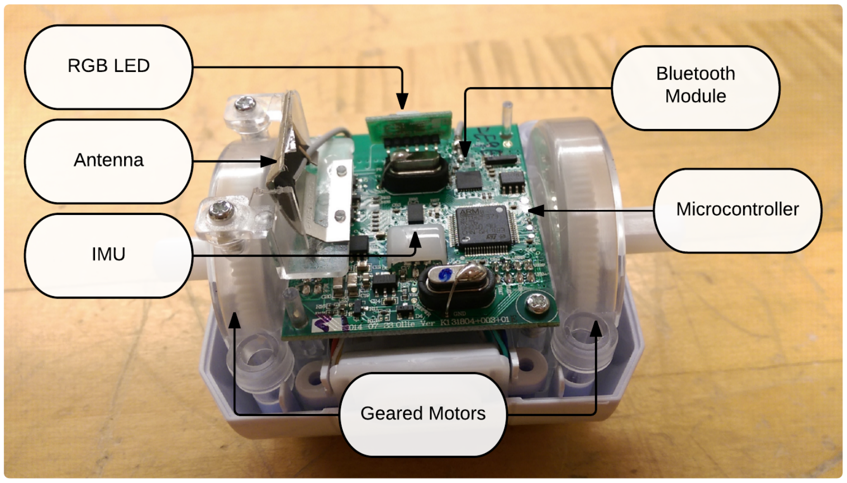

Figure 2.7: Hardware enclosed inside the low-level robot

measurements and and passes this data to the microcontroller to be used as feedback in the motor control loop and can also be sent over the Bluetooth connection.

2.4 Conclusions

For multi-robot systems, as pointed out by the researchers in [49], a valid approach to solve the problem of localizing N robots would be to localize each robot independently. However, if the robots are able to detect each other, there is opportunity to increase the accuracy, resolution, or efficiency of the system. For example, if all robots only have rudimentary sensing capabilities, then the introduction of a robot with high-resolution localization sensors could vastly improve the resolution or accuracy of positional estimates by sharing information between the agents. Conversely, if all robots have high precision sensors they may be producing redundant, excessive, or even competing positional estimates (i.e., two individual estimates are contradictory to each other); the same or similar results could alternatively be achieved using fewer sensors, thus decreasing the computational requirements and energy efficiency of the system. The localization techniques presented in this research draw on the assumption that the high-level robots employ sufficient localization techniques to self-localize within the environment, i.e., their estimated pose is interpreted as truth. Therefore, the focus of the localization techniques in this work is to estimate the relative pose of the low-level robots within the environment. The techniques used to localize these robots are discussed in the chapters to follow.

Chapter 3

Inertial Odometry

3.1 Overview

In this chapter, we examine the use of inertial odometry to determine a robot's location. Inertial odometry of a vehicle may be obtained by use of the Inertial Navigation System (INS). An INS is a system comprised of a processor inertial measurement unit (IMU). The INS integrates the measurements obtained by the IMU and performs dead-reckoning to determine the location and orientation of the system. In this work, each robot would be equipped with an INS/IMU pair that includes both a gyroscope and an accelerometer. A gyroscope provides angular rate, while accelerometers measure the linear accelerations, thus the INS is capable of determining the robot's orientation and velocity.

Inertial rate sensor (i.e., gyroscope) data must be integrated to yield position data. Accelerometer data must be integrated twice to yield position. Thus even very small errors in the information provided by inertial sensors can cause an unbounded growth of the error in integrated measurements [52]. Additionally, compounded errors in orientation will cause large lateral position errors, which increase proportionally with the distance traveled by the robot [27].

Another problem is that accelerations under typical operating conditions can be very small, on the order of 0.01 g; yet, fluctuations of this magnitude already occur if the sensor tilts relative to a perfectly horizontal position by only 0.5° [51]. This scenario is likely to occur, for example, when a robot travels over uneven terrain. These effects can be mitigated by routinely resetting the orientation and location of the robot to a known pose. This can be achieved by physically moving the robot back to the origin, which does not fit the purpose of this research, or alternately, by determining its true pose using some other means of localization (e.g., computer vision, GPS, etc.).

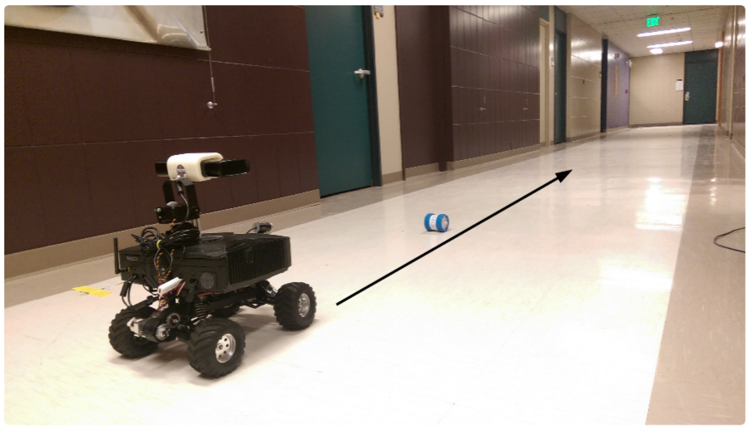

Figure 3.1: Robot setup for inertial navigation system experiment.

3.2 Test and Analysis Methods

The low-level robots used in this work has software that includes an INS that reports the robot's estimated position and controls the robot to maintain a desired heading angle. To demonstrate this capability, a test was developed to control the Sphero Ollie robot based on the positional estimates provided by the embedded INS. The robot was commanded to drive at a fixed velocity of 300 cm/s with a zero heading for an operator commanded duration of time. After the robt came to a stop, the ground-truth position and the output of the INS system were recorded. This process was then repeated to obtain a total of ten position estimates and measurements. The positions were converted to distance from the origin and compared to the ground truth measurements. The setup of this experiment is shown in Figure 3.1.

3.3 Results and Discussion

The estimated distance traveled (provided by the INS), measured ground truth distance, and absolute positional error determined from the test are shown plotted in Figure 3.2. Note that the error between estimated and actual distance traveled increases as distance increases, representing the *drift*, or ever-increasing difference between the estimated and actual positions of the system.

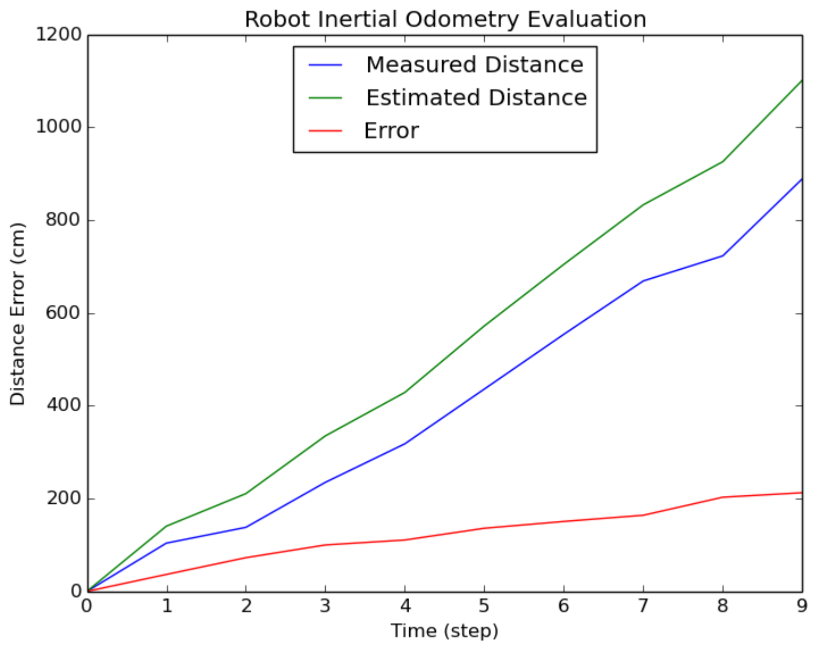

Figure 3.2: Distance traveled (blue), estimated distance traveled provided bt the INS (green), and absolute estimation error (red)

It should be noted that these results were obtained in an ideal environment (i.e., a smooth, flat surface), representing potentially the best-case scenario for inertial localization. For the applications of interest (e.g., environmental exploration) the terrain is expected to be uneven, with grade and to have a loose surface. The worst case percent error of the data collected was 52% at 138.0 cm and the average percent error over all results was 33%.

3.4 Conclusions

In this chapter we introduced the concept of using an inertial navigation system to provide positional estimates of a robot. Although the instantaneous rate information (i.e., raw data) provided by the sensors are consistently accurate over long periods of time, the small errors accumulated during integration grows indefinitely. One can infer that an inertial navigation system operating in a non-ideal environment (i.e. rough, sloped terrain) would yield even larger errors in position as time progresses. Therefore, this method of localization, alone, is not sufficient means of estimating the pose of a robot. One way of overcoming this problem is to periodically reset inertial odometry estimates with other absolute sensing mechanisms and so eliminate this accumulated error [52]. The next chapters will introduce localization methods that can be used in parallel to an INS to provide better accuracy.

Chapter 4

Vision Based Localization

4.1 Overview

This chapter discusses methods for localizing the low-level robots in an environment using a computer vision system comprised of a single RGBD sensor. An RGBD sensors is a sensor capable of providing in realtime, a color image (red, green, and blue, i.e., RGB channels) of the scene, as well as an image in which each pixel value corresponds to the distance to the object, i.e., depth image (D channel) [56]. The advent of low cost RGBD consumer devices, such as the Microsoft Kinect or Asus Xtion devices, have enabled a significant growth in the use of RGBD sensors for object detection and pose estimation. The localization strategy presented in this work leverages the high-level robot's RGBD sensor, a Microsoft Kinect. The following sections discuss the methodologies of object recognition in both RGB (i.e., 2-dimensional) and RGBD (i.e., 3-dimensional) scenarios.

4.2 2D Object Recognition

Object recognition is achieved by comparing the input image (i.e., scene), to a database of training samples (i.e., models). In the scenario presented in this work, the pose of robot has an impact on the ability to recognize the robot in the scene. Therefore, multiple orientations must be included in the database in order to ensure the robot is able to be detected. The process of 2D object recognition is outlined in Figure 4.1.

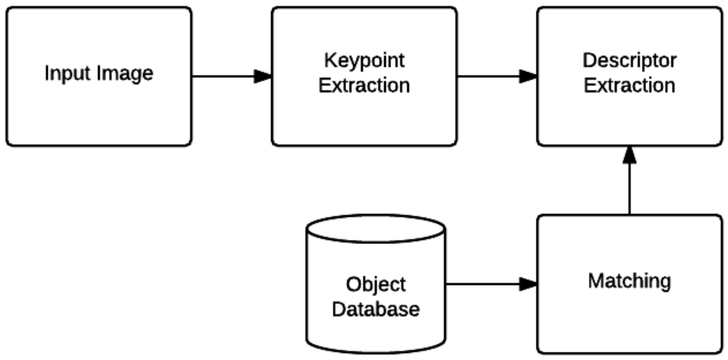

Figure 4.1: 2D Object recognition overview. Object keypoints/features are extracted from the input source, these features are described using a descriptor and then compared to features in the object database during the matching process.

Before the object recognition process begins, an object database must be constructed using images containing the object at multiple orientations and distances. Next, areas of interest (i.e., keypoints or features) are computed for each image in the database, this process is known as *Keypoint Extraction* or *Feature Detection.* Features are often classified into four main types: edges, corners, blobs, and ridges. Once features are detected, their position, orientation, and/or scale are described using a *descriptor.* Descriptors work by encoding information about the surrounding pixels (i.e., neighborhood) of a keypoint, to be used later for comparison and matching. Many different feature detection algorithms exist; some include a descriptor operator, while others must have descriptors computed externally. Some commonly employed keypoint detectors/descriptors are outlined below:

Scale Invariant Feature Transform (SIFT)

SIFT is one example of a keypoint detector that includes a descriptor. SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes [62]. However, SIFT is computationally expensive and is often not suitable for real-time applications.

Speeded Up Robust Features (SURF)

SURF is based on the same principles and steps as SIFT but each step is slightly different. For instance, SIFT recursively calculates the difference of Gaussians (i.e., a feature enhancement algorithm used to increase visibility of edges) on rescaled images whereas SURF uses square-shaped filters on the integral image (i.e., summed area table) to approximate Gaussian smoothing. These small functional differences allows SURF to be a common alternative to SIFT due to its ability to perform several times faster, however comes at the cost of poorer keypoint description approximations [61].

Features from Accelerated Segment Test (FAST)

FAST is an extremely computational efficient corner detector. The detector works by evaluating the brightness (i.e., intensity) of pixels in a 16 pixel circle; A corner is detected if a set of contiguous pixels in the circle are all brighter than the intensity of candidate pixel or all darker than the intensity of candidate pixel. The speed and efficiency of of the algorithm is what makes FAST the method of choice for finding keypoints in real-time systems that match visual features [59]. FAST does not include a built-in descriptor.

Binary Robust Independent Elementary Features (BRIEF)

BRIEF is a feature descriptor that uses simple binary tests between pixels in a smoothed image patch [60]. Its performance is similar to SIFT in many respects, including robustness to lighting, blur, and perspective distortion. However, it is very sensitive to in-plane rotation [59].

Oriented FAST and Rotated BRIEF (ORB)

ORB is an extension and combination of FAST and BRIEF that adds a fast and accurate orientation component to FAST and method of computing oriented BRIEF features. ORB is found to outperform SIFT/SURF in nearest-neighbor matching over large databases of images, however does not adequately address scale invariance.

ORB is the keypoint detection and description method chosen in this research due to its computational performance and accuracy. Once the keypoints and descriptors are computed for all objects in the database, the model is considered complete and it is now possible to compare an input scene to the model.

To localize the low-level robot in the scene using 2D object recognition, features and their descriptions are continuously calculated on the current color image obtained from the RGBD sensor. The feature descriptions in the scene are compared to those stored in the object database by calculating the Euclidean distance, sum of square differences, or by using nearest-neighbor matching algorithms; this process is known as *matching*. If a match is found between the model and the scene, the orientation of the robot is computed by calculating the rotational transform between the closest matching image from the database and the current scene image. To complete the full pose of the robot, the matching pixels from the color-image are converted to their corresponding pixels to the depth-image, this reveals the positional vector of the robot.

However, while 2D object detection uses only the color image to sufficiently detect an object in a scene, a method that includes the addition of depth information during object detection can increase recognition rates and accuracies in orientation [57]. To accomplish this feat, a specific 3D image type is used, called a *point cloud*.

4.3 3D Object Recognition using Point Clouds

In the 3D scenario, scenes and models are represented as arrays of three-dimensional points each consisting of color and position data (i.e., R,G,B and x-y-z values). These arrays of RGBXYZ points are called point clouds. Point Cloud Library (PCL) offers the tools required to manipulate, measure, and compare point cloud data. Acccording to the researchers who developed PCL, PCL presents an advanced and extensive approach to the subject of 3D perception, and it's meant to provide support for all the common 3D building blocks that applications need [63]. The library contains state-of the art algorithms for: filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation [63].

As shown in Figure 4.2, when objects have visually similar colors and textures but are different sizes, it may be difficult to correctly identify the object using only the color image. Additionally, even the same object may appear to be completely different when viewed under various lighting conditions, as shown in In Figure 4.2 (a) .In Figure 4.2 (b) the two visually similar objects are perfectly scaled in a way that could cause a computer vision system looking for features in RGB space to identify both objects as the same object at different distances. The addition of depth information can be used to increase reliability, however, also increases computational complexity and time required to achieve successful object recognition.

The same concepts of 2D recognition hold true for object recognition in 3D; a similar process involving training, feature detection, description, and matching takes place in order to detect the object using three dimensions. This process is the same as outlined in Figure 4.1 except the input is now a point cloud.

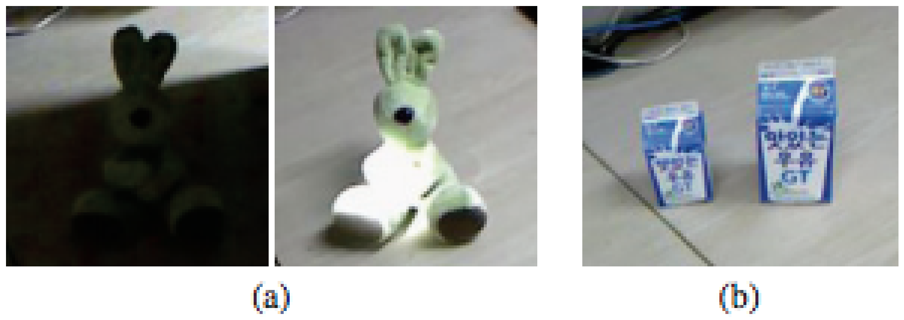

Figure 4.2: Examples where the sole use of an RGB image is unsuccessful in detection: (a) a target object is under different lighting conditions; (b) two objects have almost the same shapes and textures but their sizes are different [57].

4.3.1 Training

Just as required for object recognition in 2D, a database of 3D models is required to determine if the target object is present in the scene. There exist two methods for creating a 3D model for recognition: ray-tracing and frame-stitching. These processes are outlined in Figure 4.3 and detailed below.

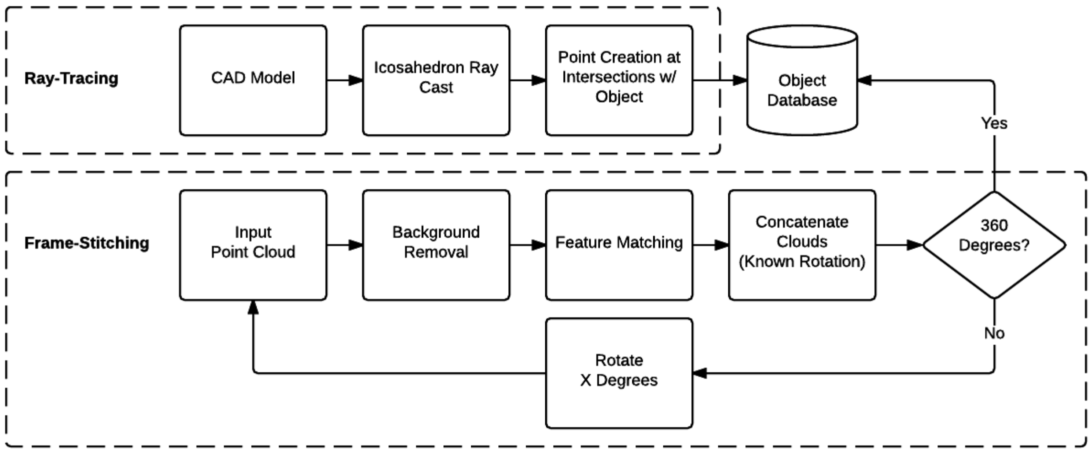

Figure 4.3: Two methods for creating a point cloud representation of the model. Ray-tracing (top) uses a 3D CAD model to generate the model, whereas frame-stitching (bottom) uses the 3D sensor to reconstruct a digital projection (i.e., model) of the physical object. Both methods can be used to add models to the database of recognizable objects.

The ray-tracing method creates a point cloud from a computer generated (i.e., CAD) 3D model. This method generates a sphere or icosahedron that is tesselated (i.e., divided in polygonal regions) that surrounds the virtual object. Then, a ray is cast from each of the vertices pointing to the origin. Finally, a point is generated at the location of intersection for each ray; this creates a 3D point cloud from points that lie on the surface of the object model.

The benefit of this technique is that, assuming the CAD model was correct, the point cloud model should perfectly represent the object we are seeking to detect. However, this can be non-ideal in some cases where the input scene contains high levels of *noise* (i.e., random points in the coud that do not correctly correspond to pints on the model). The researchers in [69] classify this noise produced by the Kinect sensor as having both axial and lateral noise distributions as a function of distance and angle from the sensor to observed surface. As a result, the point cloud produced by the RGBD sensor contains a non-perfect projection of the object, thus matching a non-deal object to an ideal model can be challenging.

Conversely, frame-stitching (i.e., reconstruction) is a method that generates a point cloud of an object by using the same sensor that will be used for detection, therefore there is a comparable amount of noise already built-in to the model. This process works by capturing multiple snapshots at various orientations based on a known rotational transform. This can be achieved by placing the object that is to be recognized on checker-pattern atop a lazy susan, as shown in Figure 4.4. The checker pattern is used by the camera to subtract the background from the object, as well as compute the rotational transform between snapshots. Once the object has been rotated a full 360 degrees, the point cloud snapshots are concatenated together by using the rotational transform and matched features of successive orientations to create a full point cloud projection of the object.

Unlike ray-tracing, this model is not ideal. This too can have a non advantageous effect if the noise is too great, it is possible that the noise will cause errors during reconstruction,

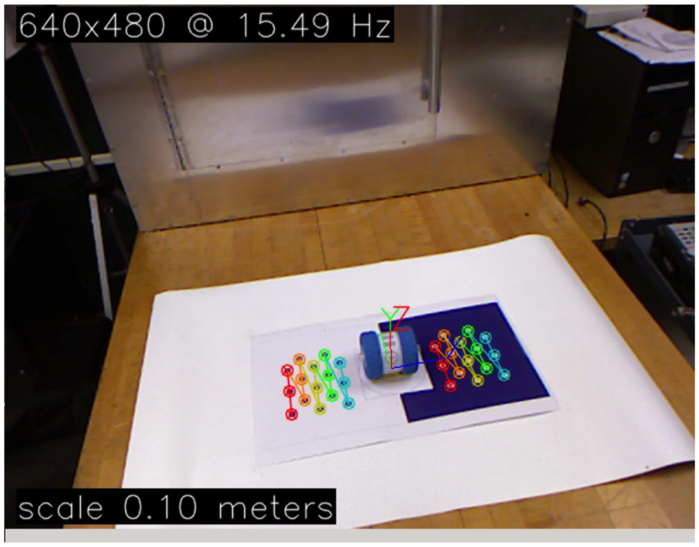

Figure 4.4: Setup to perform frame stitching for point cloud reconstruction

resulting in a significantly distorted model of the object. Figure 4.5 depicts an example of a point cloud generated using this method.

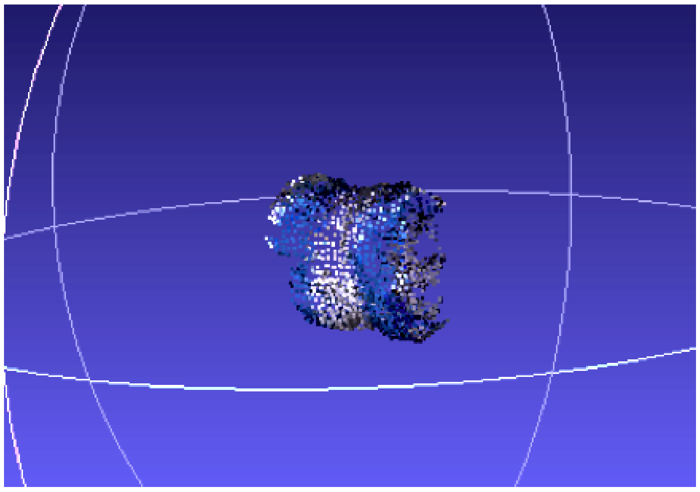

Figure 4.5: Point cloud model generation using frame stitching for reconstruction

4.3.2 Feature Detection and Description

Similar to object detection in 2D, an object must be described using 3D features. There again exist many different types of 3D features, each with their own specific strengths and weaknesses. As with all heuristic based searching (e.g., locating or matching 3D features), there are big differences in terms of recognition performance, size and time requirements for each available descriptor. The researchers in [58] perform a full comparison of available 3D features and their descriptors available in PCL. Their study concluded that the SHOTCOLOR descriptor presents a good balance between recognition performance and time complexity The Signature of Histograms of Orientations (SHOT) works by describing topological (i.e., surface) traits by taking points within a spherical support structure. The 3D descriptor used for the research presented in this paper is SHOT, chosen for its invariance to rotation and translation, as well as its robustness to noise (e.g., low quality image sensors, image down sampling) and clutter (i.e., occlusions and many objects in proximity).

4.4 Test Methodology

An experiment was devised to detect the position of the low-level robot in two dimensional space by using the RGB (i.e., color) image obtained by the Kinect and determine the 3D position of the robot by using the corresponding depth data. First, information about the low-level robot was input to the object database by manually obtaining multiple RGB photos of the low-level robot at 0.5m distance increments and 30 degree increments in orientation. Each image was cropped such that the robot took up the majority of the frame. Keypoint detection and feature description was performed on each image using the ORB feature detector/descriptor pair; the resulting datapoints were used to create the object database of recognizable objects. The low-level robot's model was then placed in front of the high-level robot and the object detection sequence described in Figure 4.1

was performed. When the low-level robot was detected in the RGB image, the estimated distance between the robots was obtained by recording the pixel intensity value of the depth image for the corresponding pixel in the RGB image (i.e., the center of the bounding box that encloses the recognized object). Next, multiple low-level objects were placed in front of the high-level robot at different orientations to evaluate the ability to detect multiple robots at different orientations.

Next, an experiment to explore object recognition with 3D point clouds was designed to test the ability of the robot to detect objects using the high-level robot's Kinect sensor. Again, the low-level robot was input into the object database using both techniques described in Figure 4.3. The keypoint detection method used in this experiment was uniform sampling (i.e., filtering the point cloud into uniformly spaced 3D points), and the descriptor used to describe each keypoint was SHOT. The low-level robot was then placed in front of the high-level robot and the object detection sequence described in Figure 4.1 was run on the input cloud using the same keypoint detection and descriptors as used to create the database.

4.5 Results and Discussion

The two experiments outlined in the previous section were performed to detect the low-level Sphero Ollie robot shown in Figure 2.5. Results from the object recognition experiment using the 2D (i.e., RGB) detection method are describe in Section 4.5.1. Results of 3D object detection are detailed in Section 4.5.2.

4.5.1 2D Object Recognition

The 2D object recognition experiment was able to successfully identify the low-level robot using the RGB image obtained by the high-level robot. Figure 4.6 shows a bounding box drawn around the identified object. Note the box is skewed to represent the rotational transform corresponding to the orientation of the low-level robot.

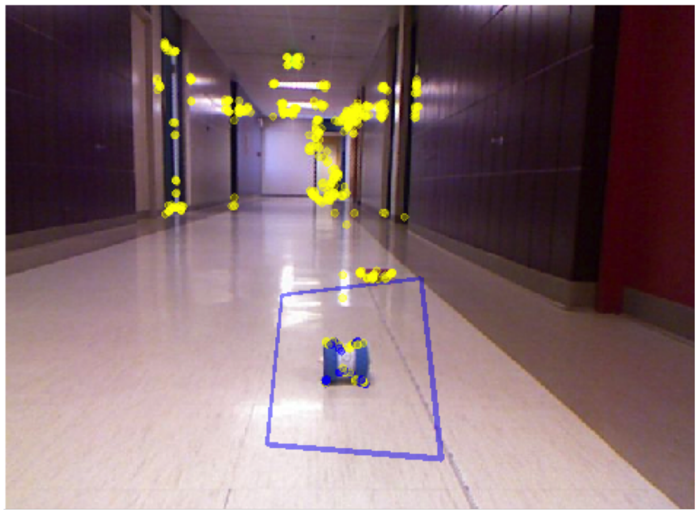

Figure 4.6: 2D object recognition on low-level robot with bounding box drawn at object's center. Features detected in the scene are marked by circles.

The 2D object recognition system was also able to simultaneous detection multiple low-level robots at different orientations. Figure 4.7 shows both low-level robots with ther corresponding bounding boxes. As the robot(s) are recognized in the RGB image, the depth value is found by calculating the intensity of the depth image for the corresponding pixels. Figure 4.8 shows the depth image side-by-side to the RGB image to depict the dualities in the images. The depth image value that corresponds to the location of the low-level robot was compared to the actual distance measured and plotted in Figure 4.9.

The absolute error between the estimated distance (i.e., depth image value) and actual measured distance is less than 10 cm for all positions where object identification was successful. Note that object detection was not successful for all positions, thus an estimated distance is not obtainable for this datapoint. The inability to detect the low-level robot may be, in part, due to the non-scale invariant nature of ORB. This means that in order to efficiently detect the object at multiple distances, similar distances need to be reflected

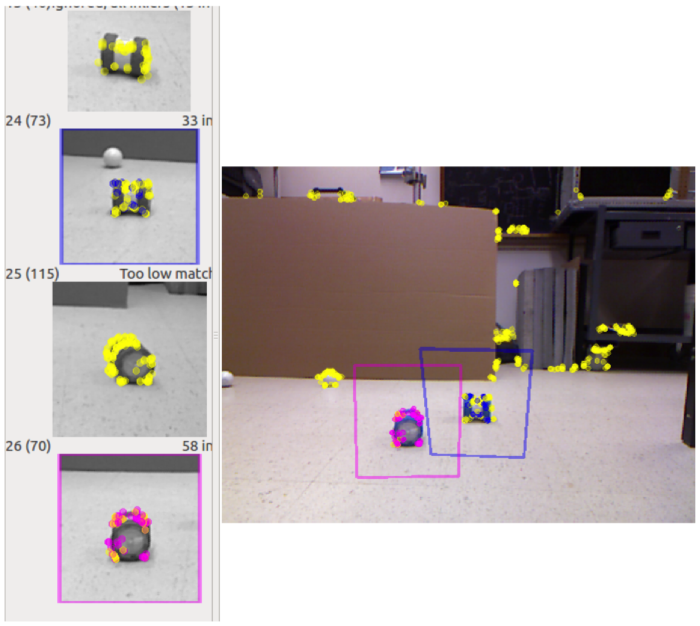

Figure 4.7: 2D object recognition on multiple low-level robots at different orientations, simultaneously (right). An excerpt of the object database is shown on the left.

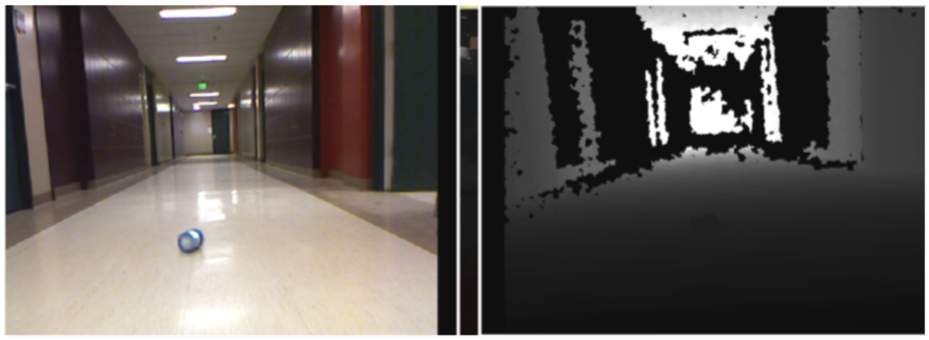

Figure 4.8: The Kinect RGB (left) and Depth (right) channels.

in the object database. Therefore, a simply method to improve recognition rates would be to simply append datasets to the database at set increments of distance (e.g., every 0.2 meters). However, it may not be ideal to increase the size of the recognition database, as more calculations will need to be performed at each time step, ultimately increasing the required time to complete the detection algorithm. Another viable method to increase recognition rate is to fuse this data with the odometry information based on previously

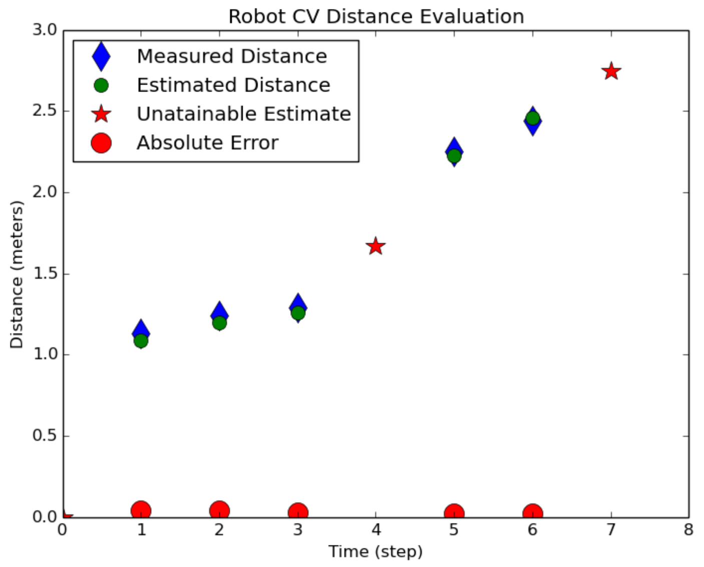

Figure 4.9: Measured distance of robot in hallway (blue), distance estimate using RGBD values (green) from Microsoft Kinect and 2D object detection, and absolute error (red). Locations where the robot was unable to be detected are shown by a star.

identified distances. Furthermore, an object tracking system could be designed such that when a successful image detection is recorded, the known dynamics of the low-level robot can be used to guide the vision system to maintain an esitamted representation of where the robot is.

When the 2D image detection system succeeded, the high-level robot is able to process the estimated pose of the low-level robot and insert this pose into the robot's virtual representation of the world around it. This pose estimate is used to locate the object in the environment as shown in Figure 4.10

The relative pose of the low-level robot is shown by the long arrow pointed towards the high-level robot model. The red, yellow, and blue (partially hidden) axial arrows represent the orientation of the low-level robot, thus fully describing the pose of the low-level robot in relation to the high-level robot.

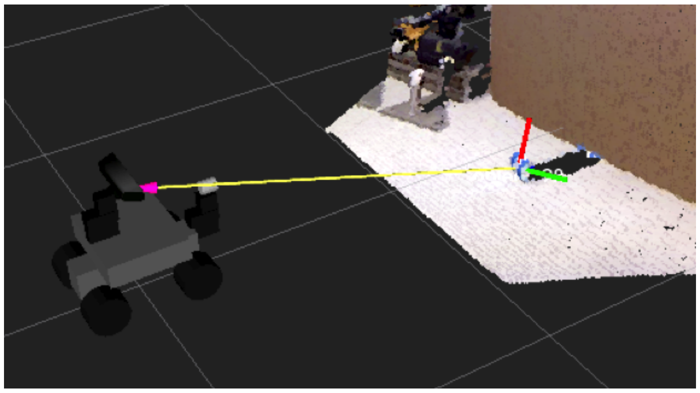

Figure 4.10: 3D Pose estimate of low-level robot relative to high-level robot.

4.5.2 3D Object Recognition

For the three dimmensional point cloud experiment, complete processing of the scene took over 20 seconds to complete on the high-level robot's processing unit. The scene under test is shown in Figure 4.11. The keypoints chosen by uniform sampling are represented by bold spheres, recall that descriptors were calculated for each of these points. Note that many of the descriptors were calculated on points that do not lie upon the object (i.e., low-level robot). These descriptors consumed processing time, thus future attempts should focus on removing as much of the background as possible before computing descriptors on the scene.

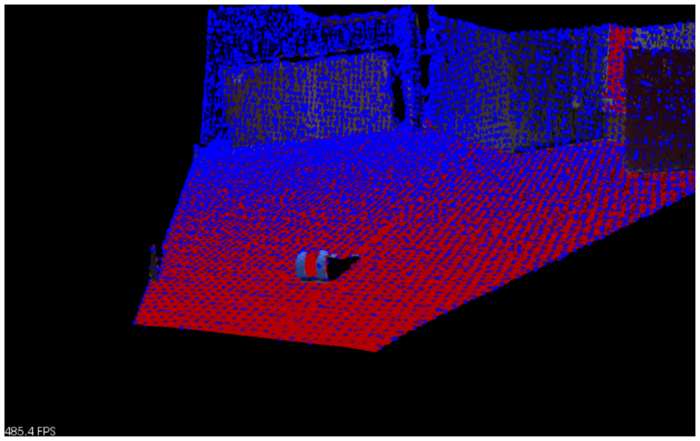

Figure 4.11: 3D Point cloud of experiment scene; Uniformly sampled kepyoints shown as large blue spheres.

Note that the low-level robot was unable to be identified in the scene using the chosen 3D points and descriptors. To debug the failed results, the algorithm was tested against sample dataset that included a point cloud scene and model. Recognition was successful when using the sample dataset and the model was correctly identified in the scene with the correct pose. For this reason, it is believed that the failure to identify the object in the scene could be due to the following reasons: 1) The construction methods (refer to Section 4.3.1) used in this work were performed incorrectly, which resulted in an inaccurate model that was unable to be identified in the scene; and 2) The chosen feature detector/descriptor pair was insufficient for the desired application. It is possible that the shape of the low-level object was not sufficiently captured by the SHOT descriptor.

4.6 Conclusions

Object detection in two dimensions was fast and estimated pose was accurate when available. However, 2D object detection is prone to lighting effects that are often uncontrollable for the given environment. These lighting effects can be mitigated by the addition of depth information into the algorithm. For exmample, the researchers in [57] use a depth sensor in addition to a RGB camera to improve object detection and pose estimations. Additionally, they were able to achieve real-time results by parallelizing the intensive computations required for 3D detection on a graphics processing unit (GPU). Therefore, although 2D object recognition was able to produce successful matching results in real-time, the use of 3D object recognition should not be discounted.

As evident from the experiments discussed in this chapter, computer vision is not guaranteed to always be able to identify and localize the object, this is specifically true in scenarios where line-of-sight with the object is not possible due to obstructions (e.g., difficult to navigate terrain). For such scenarios, it would be advantageous to implement a localization strategy that functions in parallel and successfully in non-line-of-sight scenarios. In the following chapter, we will introduce such a method that uses the robot's existing wireless communication system as a means for localization.

Chapter 5

Wireless Localization

5.1 Overview

This chapter presents a localization strategy that leverages the existing wireless (or radio frequency - RF) signal used for communicating between swarm robots. The primary advantage of this system is the ability to estimate location without a direct line-of-sight of the object. In addition, RF localization can be used in as long as the robots maintain a communication link. RF localization techniques include ranging, or estimating the distance or proximity to an object [22]; triangulation, which performs ranging from at least three different access points to geometrically determine the position [9]; or fingerprinting, which involves matching current signal strength measurements to a set of previously recorded measurements over all possible positions (similar to determining a position by comparing to a map) [22]. The work presented in this paper leverages the latest low-energy wireless communication protocol, Bluetooth 4.0 (Low Energy) and propose a new signal measurement strategy to account for practical hardware and channel constraints.

5.2 Received Signal Strength Indicator

Localization based on RF signals is one of the most used approaches (alongside vision-based systems) for indoor applications. Wireless localization techniques have focused on wireless protocols such as WiFi [23, 31, 32] as well as Bluetooth [11, 12, 13, 14, 15]. A summary of RF localization experiments sorted by wireless protocol are listed in Table 5.1.

Table 5.1: A Summary of RF Localization Experiments

Experiment	Protocol	Method	Line-of-Sight	Accuracy
[32]	WiFi	Fingerprinting	NLOS	1.2 m
[23]	WiFi	Fingerprinting	NLOS	2.57 m (0.4 m w/ odometry)
[11]	Bluetooth	Multiple Neural Networks	NLOS	0.5 m
[12]	Bluetooth	Fingerprinting	LOS	3 m
[15]	Bluetooth	Ranging	LOS	2.08 m
[22]	Bluetooth	Ranging	LOS	<5 m
[18]	Zigbee	Ranging	LOS,NLOS	1.8 to 3.5 m
[17]	Theory only	Triangulation	Not specified	Not available

Many RF localization techniques have been developed using a radio's received signal strength indicator (RSSI) [16, 17, 18]. Another signal measurement parameter used in localization is the Link Quality Index, or LQI. In [18], the experimenters used both RSSI and LQI to derive better location estimates. It is noted that LQI is correlated to RSSI and signal to noise ratio (SNR), as well as bit error rate (BER). For most Bluetooth modules, LQI is derived from the average BER seen at the receiver, and is constantly updated as packets are received. However, the exact mapping from BER to LQI is device-specific [20]. The localization experiment in [18] used 802.15.4 Zigbee radios, where LQI is well defined in the protocol's specification. Unfortunately, the LQI value is not well defined in the Bluetooth specification, thus it is not explored in this research.

5.3 Bluetooth

Bluetooth is widely accepted as the primary short-distance wireless transfer protocol. Communication range varies by chip manufacturer; most chips can achieve distances of 30 to 50 meters, but some advertise a range of over 100 meters [29]. The widespread adoption of Bluetooth in commercial systems (e.g., phones, laptops, and cars) has resulted in the availability of Bluetooth chips at low prices. Bluetooth's low power consumption (typically between 1 and 10 mW) makes this technology more appealing to use than similar technologies such as Wi-Fi (between 30 and 100 mW) [30].

The availability of RSSI has been included in the Bluetooth specification since Version 1.1 in 2002, however this value was often deemed ambiguous because of an open-loop power control scheme. In 2009, Bluetooth 3.0 introduced an Enhanced Power Control (EPC) requirement. EPC implements a closed loop power control scheme that dynamically adjusts the transmitter power to maximize bandwidth and minimize power consumption.

For all Bluetooth 3.0 devices, the RSSI measurement compares the received signal power with two threshold levels (determined by the chip manufacturer), which define the Golden Receive Power Range (GRPR). Positive values of RSSI indicate that the signal strength was above the upper threshold while negative values signifies the signal strength fell below the lower limit. A value of zero represents the optimal condition, when the signal strength is between the two thresholds [11]. The EPC scheme is used to position the RSSI within the GRPR. In order to use the RSSI value for ranging purposes while implementing a dynamic power control scheme, the transmitter power must be known. Consequentially, the only time an RSSI measurement could be reliably interpreted was during device discovery scan - as it was known that the device was then transmitting at maximum power. For this reason, most devices did not provide access to RSSI readings during an active connection (i.e., when EPC is enabled).

The advent of Bluetooth 4.0 (also known as Bluetooth Smart) in 2010 introduced many new improvements that merged three Bluetooth protocols: Classic (legacy support), high speed (based on Wi-Fi), and Low Energy (LE or BLE). Among these improvements was the implementation of the Proximity Profile (PXP) which provides an optional service called TX Power Service. This service is used to normalize the client's RSSI value by subtracting the RSSI from the transmitter's power level [24]. This enables BLE devices to report accurate RSSI values during an active connection. These RSSI measurements can be used to obtain an estimated distance of the device as it moves within the environment.

The use of Bluetooth for location-based services is increasing due to the recent introduction (2010) of the revised Bluetooth Low Energy protocol. BLE devices consume significantly less power (between 1% to 50% the power of Bluetooth classic technology [29]) and feature an Ultra-Low-Power (ULP) protocol that boasts operational energy use of merely 100 mAh per day - enabling the device to operate on a single coin cell battery for up to four years [24]. The low power consumption makes this technology appealing for mobile applications. For this reason, Bluetooth LE devices are commonly found in new consumer electronics such as smartphones, tablets, and laptops.

In 2013, Apple introduced a location-based service called iBeacon that utilize Bluetooth LE to estimate a user's proxim,y to an iBeacon accessory [13]. While iBeacon was only targeted towards iOS 7+ compatible devices, the company, Radius Networks, designed an open source library to interact with iBeacon accessories across multiple platforms including Linux and Android [14]. Original iBeacon devices have since been recreated using off the shelf Bluetooth LE adapters and Linux machines or low cost microcontrollers such as Raspberry Pi and Arduino. The Apple iBeacons, and similar devices, operate by broadcasting a PXP profile that nearby devices can link to and estimate their distance from the beacon. A single locating beacon may be used to determine if a target is within proximity to the device, or multiple devices can be used to triangulate the user within an environment.

5.4 Limitations

The accuracy of localization based on Bluetooth RSSI strongly depends on the techniques used when taking RSSI measurements. In general, the accuracy of RF localization is limited by many factors; precision is significantly reduced due to characteristics of or objects within the environment, as well as the pose of the receiver.

Complex environments have many factors that contribute to variable attenuation of a radio signal, i.e., signal fading. For example, indoor environments often contain reflective materials that affect the propagation of radio frequency signals in non-trivial ways, causing severe multipath effects that can produce dead-spots [18]. These effects may in addition cause the power received to decay at a faster rate than in a free-space model [19]. Radio waves also lose energy when they collide or pass through an object [22], which leads to an effect known as scattering. Both multipath and scattering influence the RSSI value, thus decreasing the accuracy and consistency of the measurement.

Fading effects may be also frequency selective, i.e., different frequency components experience uncorrelated fading [25]. In order to mitigate effects of frequency selective fading, the Bluetooth specification has implemented an adaptive frequency hopping (AFH) technique [29]. However, this may introduce uncertainty in an RSSI measurement due to the fact that there is no indication as to what frequency is being used when an RSSI measurement is being taken.

The relative orientation between the antennas of a sender and receiver has a direct influence on the signal strength measured by the receiver [21]. The effect of orientation is due to the non-uniform radiation pattern emitted from the antenna of the device. A non-uniform radiation pattern implies that signal strength is a function of both distance and angle between the signal source and receiver [22]. An experiment performed in [22] shows that merging different antenna orientations, i.e., calculating the mean RSSI from from four different orientations at each distance, improves the accuracy of distance estimates. This is a concept we leverage in our presented experiments (Section 5.6).

RF localization techniques that involve a user carrying a ranging device must also be aware of signal attenuation due to power absorption from the user's body. During a Bluetooth localization experiment using a fingerprinting technique, the researchers in [11] designed a neural network (NN) algorithm to mitigate the effects of the power absorption from the carrier's body. In a similar RF localization experiment that used WiFi as the radio protocol, a combination of fingerprinting and an artificial neural network algorithm was used on a robot carrier. The experimenters in [23] concluded it was possible to determine the position (and sometimes orientation) of the robot from only the RSSI measurements by comparing them to a prerecorded wireless fingerprint (i.e., map).

Thus multipath effects, antenna orientation, and blockage by a robot's structure can all potentially alter a RSSI measurement. Due to these factors, two robots at the same distance from a source can measure vastly different RSSI values. It is this variability that we strive to mitigate in the presented work.

5.5 Modeling the Wireless Channel

In order to make sense of the available RSSI data, it is important to first understand the relationship between radio signal power and distance. This relationship is used to model the wireless channel such that distance estimations can be extracted from RSSI data. The following sections detail this relationship.

5.5.1 Log-Distance Pathloss Model

Under ideal propagation effects, the power received is proportional to the squared distance between the transmitter and receiver. The equation used to relate power to distance using the Friis model is shown in Equation 5.1

$$\overline{RSSI} = P_{TX} + G_{TX} + G_{RX} + 20 \times log_{10} \left(\frac{\lambda}{4\pi R} \right)^2 \tag{5.1}$$

Where $\overline{RSSI}$ is the median radio signal strength indicator in dBm, P_{TX} is the transmit power in dBm, G_{TX} is the gain of the transmitter antenna in dBi, G_{RX} is the gain of the receiver antenna in dBi, λ is the wavelength in meters, and R is the distance from the receiver in meters.

This model is sufficient for a free-space environment, however, complex environments have many factors that contribute to attenuation of a radio signal which may cause the power received to decay at a different rate than in a free-space model.

5.5.2 Log-Normal Model

To better represent the effects of signal fading, we introduce the concept of a *pathloss coefficient*. This coefficient is used to describe the rate at which the wireless signal decays and brings about a new equation used to model a wireless channel, shown in Equation 5.2.

$$\overline{RSSI} = -10 \times n \log_{10}(d) + A \tag{5.2}$$

Where $\overline{RSSI}$ is the median radio signal strength indicator in dBm, n is the pathloss coefficient (<2 to 4, where $n = 2$ is ideal for free-space environments), d is the relative distance between the communicating nodes relative to 1 meter, and A is a reference received signal strength in dBm at a distance of 1 meter. As there are many environmental factors that contribute to the attenuation of the radio signal, the pathloss coefficient is often found empirically by measuring the RSSI at various distances and calculating the rate at which RSSI decreases relative to the distance from the transmitter.

However, this model does not account for non-trivial propagation of radio signals under multipath conditions or the effects of shadowing (i.e., blockage of signals due to objects).

5.5.3 Log-Normal Shadowing Model

To account for the multipath and scattering effects discussed in Section 5.4, a term that represents a uniformly distributed random variable is appended to the Log-Normal pathloss model to obtain the equation shown in Equation 5.3.

$$\overline{RSSI} = -10 \times n \log_{10}(d) + A + X_{\sigma} \tag{5.3}$$

Where X_{σ} is a normally distributed random variable (in dB) with zero mean, representing the variable attenuation caused by a non-uniform environment.

The Log-Normal Shadowing (LNS) model is used in this work to estimate the distance between robot swarm members. The model parameters, n, A, and X_{σ} are found during a calibration phase, described in Section 5.7.

5.6 Mitigation of Undesired Effects

In the presented work, we leverage Bluetooth LE hardware to explore techniques to increase the reliability of RSSI data even under the operational constraints discussed in Section 5.4. Orientation effects can be caused by *i*) the antenna not having a uniform omni-directional pattern, *ii*) the robot's structure impedes the antenna radiation in some directions, or *iii*) the local environment creates different multipath conditions based on orientation. All three of these effects can be modeled as being random, thus to mitigate these effects we introduce the concept of having the robot spin about the vertical axis while RSSI is collected. By collecting RSSI data under this operational condition, the effects of orientation are "averaged" out. For typical mobile systems, it is not practical to have users do this maneuver and in wireless sensors networks, devices lack the mobility. However, robotic systems, by nature of their highly maneuverable designs, have this distinct attribute that we leverage for improving RSSI data measurements.

5.7 Test and Analysis Methods

As discussed in the Section 5.4, the orientation of the target devices strongly influences the measured RSSI. To illustrate this concept,the low-level robot is placed in an ideal (i.e., anechoic) communication environment and RSSI data is collected at fixed rotational intervals. The robot was placed in the anechoic chamber 0.75 meters away from a Android Nexus 7 tablet used for collecting the RSSI data. To control the orientation of the robot, the robot was mounted onto a stepper motor that was driven by a microcontroller. 400 RSSI samples were collected at 60 Hz for each orientation (16 steps/revolution). The motor was automatically commanded to go to the next orientation only after the desired number of samples were taken. The setup of this experiemnt is shown below in Figure 5.1.

Next, the low-level robot was moved to a non-ideal environment and placed at fixed distances from the high-level bot. The low-level robot was commanded to spin about its vertical axis and 600 RSSI samples were taken at 60 Hz while the robot spins at approximately 5 revolutions per second. The setup of this experiment are shown in Figure 5.2. The median RSSI data is calculated and used to calibrate a LNS model of the wireless channel (i.e., environment). This now calibrated model is used for calculating distance estimates. Finally, the robot was placed at various distances from the source and the distance estimates obtained from the now-calibrated wireless model were compared to the ground truth location of the robot.

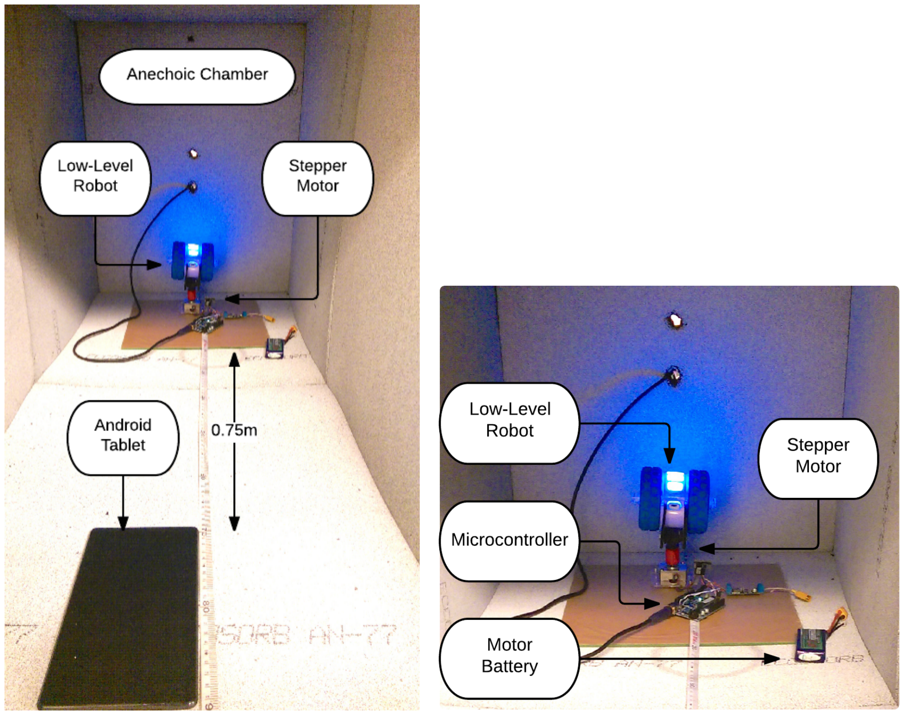

Figure 5.1: The low-level robot setup in the anechoic chamber (Left). The mechanism for setting the orientation of the low-level robot in the anechoic chamber (Right).

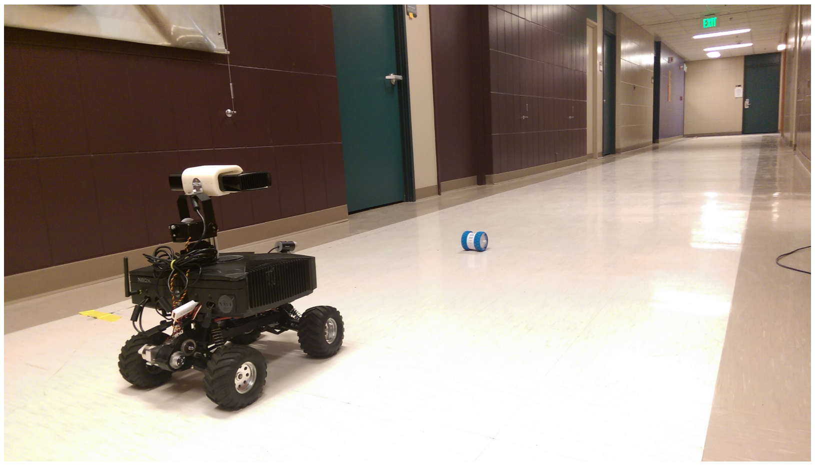

Figure 5.2: Robot setup for multipath (hallway) RSSI sampling experiment.

5.8 Results and Discussion

The two experiments outlined in the previous section were performed using the low-level Sphero Ollie robot shown in Figure 2.5. The communication performance of the ideal environment at multiple orientations is discussed in Section 5.8.1. To test the accuracy of RF-localization using RSSI measurements in non-ideal environments, the robot was moved into a hallway (i.e., multipath environment). A LNS model was calibrated and tested for accuracy, the results of this experiment are described in Section 5.8.2.

5.8.1 Anechoic Chamber Measurement

As the robot rotates about the vertical axis, the received radio strength greatly fluctuates. The resulting RSSI samples in reference to the robot's orientation are illustrated in Figure 5.3. The median RSSI value is indicated by a circle and the error bars represent the range in the recorded RSSI data. Upon observing four complete revolutions of RSSI data, a noticeably cyclical pattern is revealed. This pattern represents the non-ideal communication performance of the robot at various orientations under idealized RF propagation conditions.

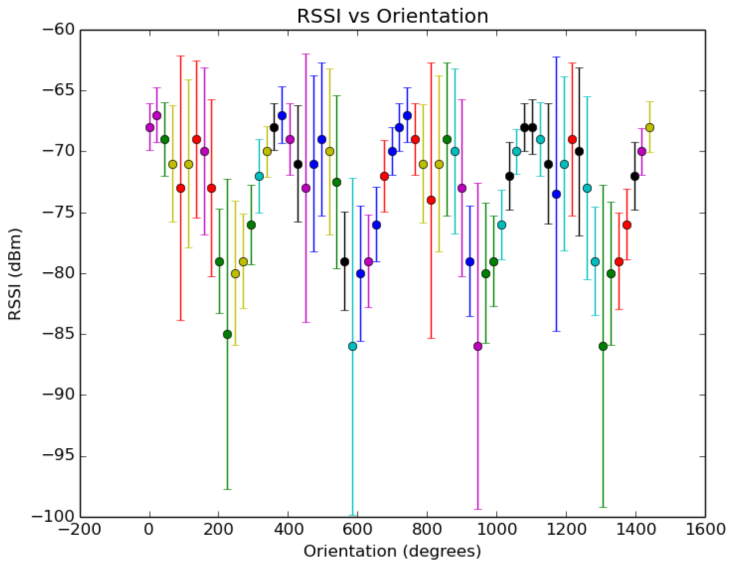

Figure 5.3: RSSI vs orientation of rotating robot in hallway environment.

The communication performance may be due to a non-ideal radiation pattern of the robot's antenna in combination with the robot's physical structure. The orientation and configuration of the robot's antenna is shown in Figure 5.4 (also refer to Section 2.3 and Figure 2.7 for details of the robot's internal hardware).

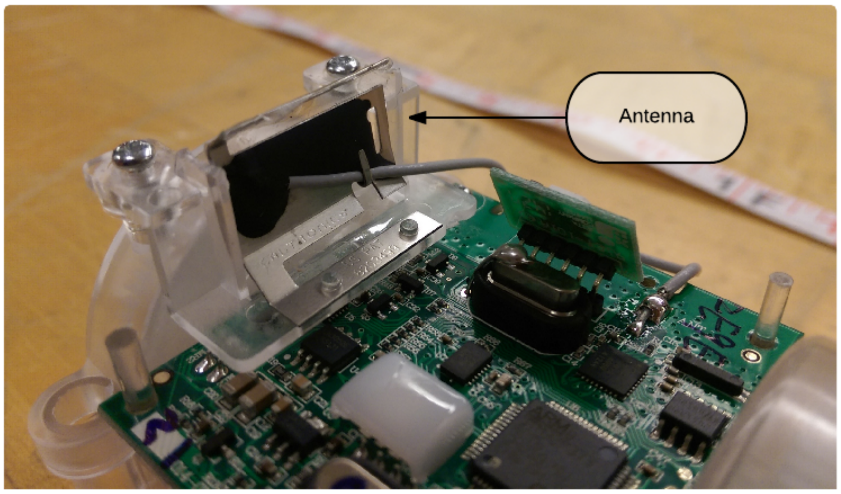

Figure 5.4: The antenna and supporting structure within the low-level robot

Note that the plot in Figure 5.3 is a function of the robot's orientation and that there is a significant amount of variation between measurements at a particular angle. For these reasons it can be presented that we can obtain a more accurate RSSI measurement by averaging the samples over of space (i.e., rotational orientation), and average over time (i.e., median filter).

5.8.2 Hallway Measurement

In order to mitigate the effects of orientation, the robot is commanded to spin about the vertical axis and a median filter is used to 'average' the RSSI over the various orientations the robot travels through during rotation.

The results of this experiment are shown in Figure 5.5. The RSSI values obtained during this experiment are used to calibrate the LNS model of the environment. From this data

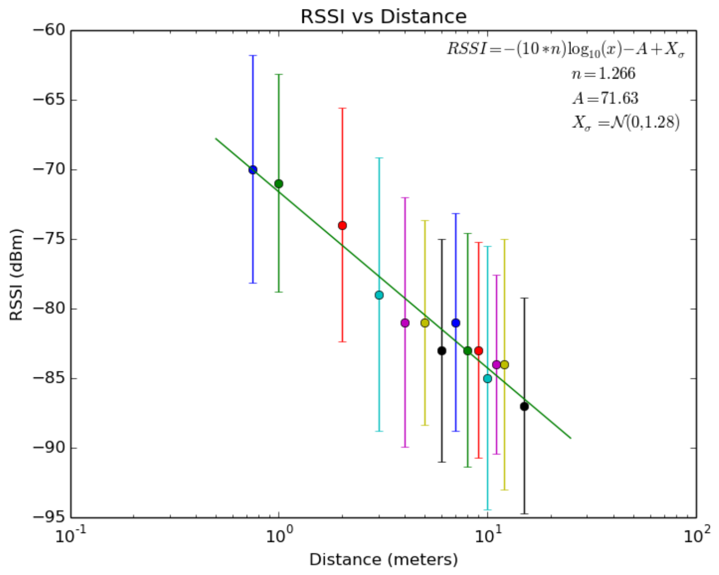

Figure 5.5: RSSI data used to fit a LNS model

we are able to determine a best fit with a pathloss exponent, $n = 1.266$, a reference value, $A = 71.63$ dBm, and $X_\sigma = \mathcal{N}(0, 1.28)$ dB. Note that a pathloss exponent of $n = 2$ is not uncommon for indoor and hallway type environments; the researchers in [22] obtained similar pathloss coefficients ranging between 1.4 and 1.55.

5.8.3 Multipath (Hallway) LNS Test

The previous experiment was then repeated, this time sampling RSSI measurements at various distances to be included into a test dataset. This experimental dataset was then tested against the LNS model calibrated from the previous experiment. The distances and corresponding RSSI values are shown in Figure 5.6. The LNS model is then used to estimate the distance between the robots and yields accuracies under 2.15 meters. The greatest percent error of these measurements was 28% and the average percent error over all results was 11.5%.

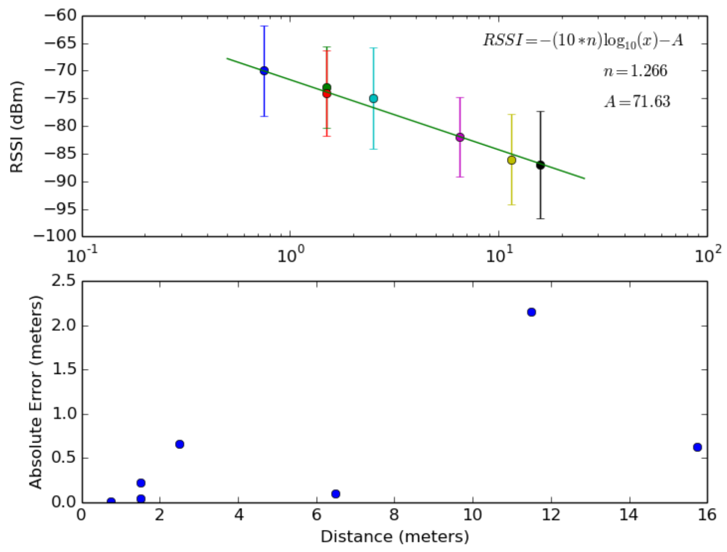

Figure 5.6: The top plot shows the recorded RSSI data from the test dataset with the median RSSI marked by a circle. The line represents the previously calibrated LNS model for this environment. The bottom plot depicts the absolute error of distance estimate using RSSI values from calibrated LNS model.

5.8.4 In-Field Calibration

Note that the LNS model used to estimate the distance between the robots had to be manually calibrated by measuring the distance of the robot at various distances; however, this process may be automated by comparing the RSSI value to estimated distances obtained by the localization methods mentioned previously in this work. An example of how this system would function is shown in Figure 5.7.

The robot is commanded to move a small distance (e.g., 0.5 m) away from the robot. The vision system attempts to locate the robot (Chapter 4). Note, in this scenario the processing time of the vision system is a non-crucial parameter as high accuracy is preferred. If vision acquisition is unattainable, inertial estimates (Chapter 3) can be used instead. The robot is then commanded to spin about its vertical axis to mitigate the effects of orientation. Next, the RSSI is measured over a duration of time (e.g. 90 seconds) and the data is added to the

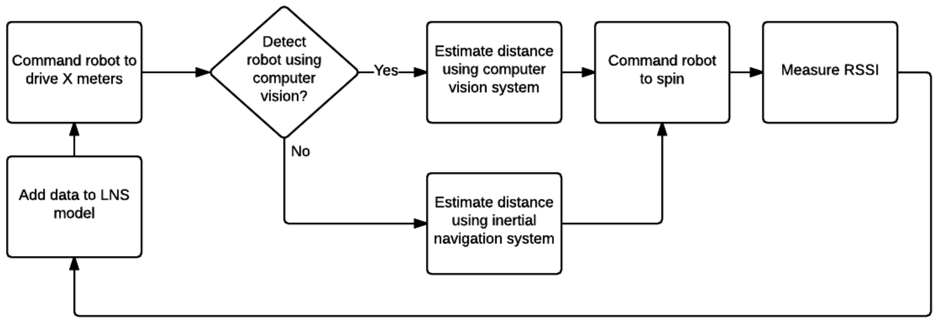
Command robot to drive X meters
Detect robot using computer vision?
Yes
No
Estimate distance using computer vision system
Command robot to spin
Measure RSSI
Estimate distance using inertial navigation system
Add data to LNS model

Figure 5.7: Process to calibrate LNS model using in-field calibration method.

LNS model. This process is then repeated at various distances to fully calibrate the LNS model.

5.9 Conclusions

In this chapter we introduced the concept of wireless ranging using received radio signal strength. The experiment depicted in Figure 5.3 supports the notion that RSSI is highly dependant on orientation of the device. We then introduced a method for increasing the reliability of wireless RSSI measurement by rotating the robot about the vertical axis and applying a median filter. This method effectively 'averages' out effects from orientation as well as variations over time. The localization estimates produced using this method offer greater accuracy than inertial estimates alone, and has a significantly larger maximum operating range than the computer vision method described in Chapter 4. Finally, we proposed a method to automatically calibrate the LNS model; this process can be used to ensure an accurate wireless model is developed without the need for operator intervention or prior information about the performance of the wireless channel.

Chapter 6

Conclusion and Extensions to Work

In this work we have introduced the concept of using a swarm of heterogeneous robots to achieve a goal. The heterogeneity of the group of robots is exploited to utilize extremely low-cost, virtually expendable robots as a means to increase sensing and actuating capabilities of the swarm in entirety. However, to make use of their sensory data their locations must be estimated and therefore must implement a method of localization. This thesis has discussed and presented three different modalities for localizing a robot within an environment: inertial, computer vision, and wireless ranging. The experiments performed in Chapter 3 show that, alone, the low-level robots are unable to localize over extended distances. However, if coupled to a robot with more sufficient sensing/localization capabilities, the low-level robot is able to be more accurately localized within the environment. The approach will enable high-level robot with sufficiently greater computational abilities and a RGBD sensor to locate and track the low-level robots as they move about the environment. This localization method is shown to achieve accuracy under 2 cm, however is not available if the object is unrecognizable (e.g., occlusion, lighting conditions).

The RF-based localization method utilizes only the robots' existing wireless communication system and is available in both LOS and NLOS scenarios. This method was shown to achieve accuracy between zero and 2.5 meters, however requires a environment-specific calibrated model. This model can be developed in-field by utilizing the alternative localization methods discussed in Chapters 3 and 4. Together, these localization methods can be used to cover multiple scenarios and localize swarm members within the environment.

6.1 Significant Contributions of Work

The contributions of this work are fourfold as described below.

1. In Chapter 2, this work introduces a concept of 'shared sensing' in which extremely simple and inexpensive robots with unreliable localization estimates are used in a heterogeneous swarm of robots in a way that increases the accuracy of localization for the simple agents and simultaneously extends the sensing capabilities of the more complex robots. This method also enables the swarm to investigate hazardous and unknown terrain by leveraging the mobility of a low cost, potentially expendable robot instead of endangering a significantly more complex, and thus expensive robot.

2. This work outlines various means to estimate a robot's pose. A method composed of an inertial navigation system is discussed in Chapter 3, localization using computer vision is detailed in Chapter 4, and a localization technique using the robot's existing wireless communication system is described in Chapter 5. The accuracy of each method is evaluated. The work also addresses the benefits of using multiple sensors simultaneously.

3. This work introduces in Chapter 5 a method for increasing reliability of RSSI measurements for wireless ranging/localization systems by averaging RSSI measurements over both time and space. This method addresses large fluctuations in RSSI measurements due to the natural propagation of radio signals in multipath environments as well as the non-ideal communication system performances that can be largely attributed to the orientation of the robot.

4. The final contribution of this work is a process for developing an in-field model for estimating the location of a robot by leveraging the existing wireless communication system. This process, described in Chapter 5, provides a way to calibrate a model to be used in an environment where prior knowledge of the wireless channel performance is not required.

6.2 Future Work

This work has highlighted three methods for localizing a robot within an environment and identified each method's strengths and weaknesses; these modalities are shown to complement each other and can be configured to work simultaneously. Moving forward, these systems must be implemented in such a way that each modality is a combined into a system that outputs a single positional estimate based on the perceived accuracy at any given time during various scenarios. For example, the proposed system should be able to interpolate where a robot is based on inertial and RSSI measurements when a visual detection is unavailable. Likewise, if a robot travels into an occluded area and experiences a large step in inertial data, it would be encouraged to rely primarily on wireless ranging data until visual detection is restored.

Multi-sensor fusion is in itself is a very large research topic; many efforts have been focused on ways to increase reliability of estimates using multiple sensors. As mentioned previously, a Kalman Filter is a common approach to fuse multiple sensory inputs together [26, 28]. In order to achieve such a system, it is important to realize that these localization techniques are operating in parallel and at different rates (i.e., data from each sensor is provided with different time resolutions). Therefore, a multi-rate solution should be employed such that the maximum data rate of each sensor can be utilized. The researchers in [64] describe a multi-rate system fusion technique and compare two different Kalman-based algorithms: the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). The

EKF and UKF are two widely used algorithms for sensor fusion due to their ability to handle non-linear systems.

The researchers in [65] developed a framework based on an Iterative Extended Kalman Filter (IEKF) for multi-sensor fusion that is able to process delayed, relative and absolute measurements from a theoretically unlimited number of different sensors and sensor types. Their system was successfully demonstrated on a micro aerial vehicle (MAV) where not all sensors were available at any time.

The ROS system that is used as a backbone of the heterogeneous robotic swarm in this research has a package that was designed to estimate the pose of a robot using an EKF. This package, *robot_pose_ekf* [66], is a strong starting point for someone wishing to continue research efforts in this area. The package provides a means to estimate the 3D pose of a robot, based on (partial) pose measurements coming from different sources. The package was designed to combine measurements from wheel odometry, IMU sensor and visual odometry, however it may be modified to operate with different sensors.

A more recent and further generalized ROS package that can be used to estimate a robot's location using multiple sensor inputs is *robot_localization* [67]. This package provides a collection of state estimation nodes, each of which is an implementation of a nonlinear state estimator for robots moving in 3D space. The intended use of this software package is outlined in Figure 6.1.

The robot *robot_localization* package is designed to handle input from multiple sensors, including an IMU, a camera, GPS, 3D depth sensors (e.g., Microsoft Kinect) and wheel odometry. The sensor inputs are fed into an estimator that provides a single 3D state estimate.

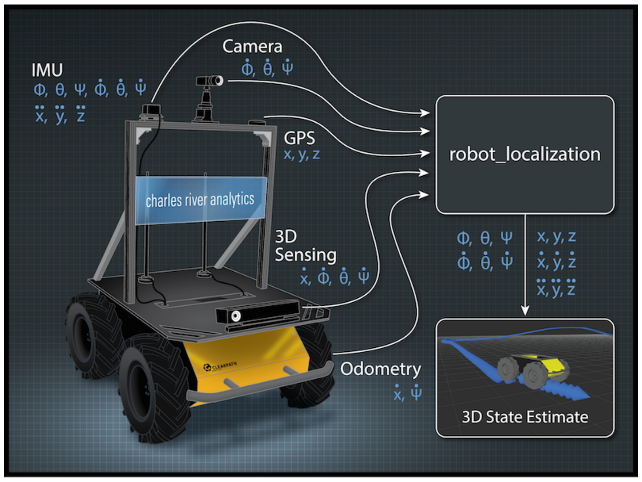
IMU
$\Phi, \theta, \Psi, \dot{\Phi}, \dot{\theta}, \dot{\Psi}$
$\ddot{x}, \ddot{y}, \ddot{z}$
Camera
$\dot{\Phi}, \dot{\theta}, \dot{\Psi}$
GPS
x, y, z
charles river analytics
3D Sensing
$\dot{x}, \dot{\Phi}, \dot{\theta}, \dot{\Psi}$
Odometry
$\dot{x}, \dot{\Psi}$
robot_localization
Φ, θ, Ψ
$\dot{\Phi}, \dot{\theta}, \dot{\Psi}$
x, y, z
$\dot{x}, \dot{y}, \dot{z}$
$\ddot{x}, \ddot{y}, \ddot{z}$
3D State Estimate

Figure 6.1: Functional overview of the robot_localization package for ROS [68].

6.3 Final Comments

Techniques of localization are not limited to the modalities described in this thesis; in fact, there exist many alternative approaches to localizing objects or vehicles within an environment (e.g., GPS, sonar echolocation, etc.). It is important to understand that chosen localization methods should cater to the environment in which they are employed as well as the budget and scope of the project. This research focused primarily on the use of low-cost robotics systems, which is why a consumer RGBD device was chosen as the vision sensor. However, better results may be achieved using more expensive, rotating 2D or 3D LIDAR systems, as well as time-of-flight or stereo cameras. Additionally, for the context of this research it is stated that GPS is unavailable; however it may be the case that GPS signal is intermittent (e.g., dense cities, forests, or inside buildings), thus localization cannot depend on GPS data however it may be used to increase reliability when available. Alternatively, there may be a positioning system that offers global localization measurements and can therefore be used in addition to or instead of the aforementioned methods to increase the reliability and accuracy of positional estimates.

The methodologies discussed in the paper are meant to explore the combinatorial effects of multi-sensor systems. Most notably, multi-sensor systems do not require all sensors to be present on a single body; sensors may be unevenly distributed, spatially or numerically, and can be used to achieve increased accuracy of positional estimates.

Bibliography

[1] F. Mondada, L. Gambardella, D. Floreano, S. Nolfi, J. Deneuborg, and M. Dorigo, *The Cooperation of Swarm-bots: Physical Interactions in Collective Robotics*, IEEE Robotics & Automation Magazine, Vol. 12, No. 2, 2005.

[2] Bristlebots, online: www.bristlebots.org, accessed: March 6, 2013.

[3] Giomi, L. N. Hawley-Weld, L. Mahadevan, *Swarming, Swirling and Stasis in Sequestered Bristle-Bots*, http://arxiv.org/abs/1302.5952, accessed: March 6, 2013.

[4] Japan Aerospace Exploration Agency (JAXA), *Asteroid Explorer Hayabusa2*, online: www.jaxa.jp/pr/brochure/pdf/04/sat33.pdf (accessed: March 8, 2014).

[5] D. Werner, Planet Labs Cubesats deployed from ISS with many more to follow, Space News, February 11, 2014.

[6] E. Sahin, *Swarm robotics: From sources of inspiration to domains of application.* Swarm robotics. Springer Berlin Heidelberg, 2005. 10-20.

[7] C. M. Wang, *Location estimation and uncertainty analysis for mobile robots*, in I.J. Cox and G.T. Wilfong, editors, Autonomous Robot Vehicles. Springer-Verlag, Berlin, 1990.

[8] I. J. Cox, *Blanche-An experiment in guidance and navigation of an autonomous robot vehicle*, IEEE Trans. Robotics and Automation, April 1991.

[9] P. Bahl, V. N. Padmanabhan, *RADAR: A, In-building RF-based User Location and Tracking System*, Proceedings of the IEEE Infocom 2000, vol.2, pp. 775-784, Tel Aviv, Israel, March 2000.

[10] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, S. Shafer, *Multi- camera, Multi-Person Tracking for Easy Living*, 3rd IEEE International Workshop on Visual Surveillance, Piscataway, NJ, 2002, pp. 3-10.

[11] M. Altini, D. Brunelli, E. Farella, L. Benini, *Bluetooth Indoor Localization with Multiple Neural Networks*, 5th International Symposium on Wireless Pervasive Computing (ISWPC). 2010.

[12] J. G. Castano, M. Svensson, M. Esktrom, *Local Positioning for Wireless Sensors Based on Bluetooth*, in Proceeding of IEEE Radio and Wireless Conference, 2004.

[13] Apple Knowledge Base, iOS: *Understanding iBeacon*, accessed May 2014 at http://support.apple.com/kb/HT6048

[14] Radius Networks, *Android Beacon Library*, accessed May 2014 at http://developer.radiusnetworks.com/ibeacon/android/

[15] S. Feldmann, K. Kyamakya, A. Zapater, Z. Lue, *An Indoor Bluetooth-Based Positioning System: Concept, Implementation And Experimental Evaluation*, in International Conference on Wireless Networks. 2003.

[16] C. Alippi, G. Vanini, *A RSSI-based and Centralized Localization Technique for Wireless Sensor Networks*, in Proceeding of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW), 2006.

[17] C.-C. Pu, C.-H. Pu, H.J. Lee, *Indoor Location Tracking Using Received Signal Strength Indicator*, Emerging Communications for Wireless Sensor Networks, Anna Foerster and Alexander Foerster (Ed.), ISBN: 978-953-307-082-7, 2011.

[18] S. J. Halder, J.-G. Park, W. Kim, *Adaptive Filtering for Indoor Localization using ZIGBEE RSSI and LQI Measurement, Adaptive Filtering Applications*, Dr Lino Garcia (Ed.), ISBN: 978-953-307-306-4. 2011.

[19] Q. Dong, W. Dargie, *Evaluation of the reliability of RSSI for Indoor Localization*, International Conference on Wireless Communications in Unusual and Confined Areas (ICWCUCA), 2012.

[20] A. K. M. M. Hossain, W.-S. Soh, *A Comprehensive Study Of Bluetooth Signal Parameters For Localization*, 2007. The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2007.

[21] B. J. Dil, P.J.M. Havinga, *Rss-based localization with different antenna orientations*, in Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian, pages 13-18, November 2010.

[22] D. Scheerens, *Practical Indoor Localization using Bluetooth*, 2012.

[23] M. Ocaña, L. M. Bergasa, M. A. Sotelo, J. Nuevo, R. Flores, *Indoor Robot Localization System Using WiFi Signal Measure and Minimizing Calibration Effort*, EEE ISIE 2005, June 20-23, 2005.

[24] *Bluetooth Specification and Glossary*, accessed May 2014 at http://www.bluetooth.com/Bluetooth/Technology

[25] J. Frolik, *A case for considering hyper-Rayleigh fading channels*, IEEE Trans. Wireless Communications, Vol. 6, No. 4, April 2007.

[26] F. Chenavier, J. Crowley, *Position Estimate for a Mobile Robot using Vision and Odometry*, in Precedings of the 1992 IEEE International Conference on Robotics and Automation (ICRA), 1992.

[27] J. Borenstein, H. R. Everett, L. Feng, D. Wehe, *Mobile Robot Positioning - Sensors and Techniques*, in an invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots. Vol. 14 No. 4, pp. 231 - 249, 1997.

[28] T. Oskiper, Z. Zhu, S. Samarasekera, R. Kumar, *Visual Odometry System Using Multiple Stereo Cameras and Inertial Measurement Unit*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007.

[29] *A Look at the Basics of Bluetooth Technology*, accessed May 2014 at http://www.bluetooth.com/Pages/Basics.aspx

[30] E. Ferro, F. Potorti, *Bluetooth and Wi-Fi wireless protocols: a survey and a comparison*, IEEE Wireless Communications, Volume 12, No. 1, 2005.

[31] Z. Xiao, W. Hongkai, A. Markham, N. Trigoni, P. Blunsom, J. Frolik, *Identification and mitigation of non-line-of-sight conditions using received signal strength*, in IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2013.

[32] J. Biswas, M. Veloso, *WiFi Localization and Navigation for Autonomous Indoor Mobile Robots*, in IEEE International Conference on Robotics and Automation (ICRA), 2010.

[33] J. Frolik, T. Weller, S. DiStasi and J. Cooper, *A compact reverberation chamber for hyper-Rayleigh channel emulation*, IEEE. Trans. Antennas and Propagation, Vol. 57, No. 12, December 2009.

[34] J. Pugh, X. Raemy, C. Favre, R. Falconi, A. Martinoli *A Fast Onboard Relative Positioning Module for Multirobot Systems*. Proceedings of IEEE Transactions on Mechatronics. 14: 151-162. 2009.

[35] Y. Hada, K. Takase *Multiple Mobile Robot Navigation Using the Indoor Global Po-*

sitioning System. Proceedings of IEEE International Conference on Intelligent Robots and Systems. Maui Hawaii: USA. pp. 1005-1010. 2001.

[36] S. I. Roumeliotis, G.A. Bekey *Distributed Multirobot Localization.* IEEE Transactions on Robotics and Automation, 2002, Vol. 18. pp. 781-795.

[37] NASA. *MMS Spacecraft and Instruments.* http://www.nasa.gov/mission_pages/mms/spacecraft/inde

[38] A. T. Hayes, A. Martinoli, and R.M. Goodman, *Distributed Odor Source Localisation*, Special Issue on Artificial Olfaction, Nagle H. T., Gardner J. W., and Persaud K., editors, IEEE Sensors Journal, 2002, Vol. 2, No. 3, pp. 260-271.

[39] H. Choset, *Coverage for Robotics –A Survey of Recent Results*, Annals of Mathematics and Artificial Intelligence, 2001, Vol. 31, pp. 113-126.

[40] Fredslund, J. and Matari'c, M. J. *General Algorithm for Robot Formations Using Local Sensing and Minimal Communication*, Special Issue on Advances in Multi-Robot Systems, Arai T., Pagello E., and Parker L. E., editors, IEEE Trans. on Robotics and Automation, 2002, Vol. 18, No. 5, pp. 837-846.

[41] J. Pugh and A. Martinoli, *Relative localization and communication module for small-scale multi-robot systems*, in Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006), 2006, pp. 188-193.

[42] R. Grabowski, et al. *Heterogeneous teams of modular robots for mapping and exploration*, Autonomous Robots, 2000, Vol. 8, No. 3, pp. 293-308.

[43] R.C. Arkin and T.R. Balch, *Cooperative Multiagent Robotic Systems AI-based Mobile Robots: Case Studies of Successful Robot Systems.* Kortenkamp, D., Bonasso, R.P. and Murphy, R. (eds). MIT Press. 1998.

[44] M. Mataric, 1995. *Issues and Approaches in the Design of Collective Autonomous Agents.*Robotics and Autonomous Systems, 16(2-4), Dec. 1995. pp. 321-331.

[45] L. E. Parker, 1999. *Adaptive Heterogeneous Multi-Robot Teams*, Neurocomputing, special issue of NEURAP 1998: Neural Networks and Their Applications 1999, vol. 28, pp. 75-92.

[46] D. Rus, B.R. Donald, and J. Jennings, 1995. *Moving Furniture with Teams of Autonomous Mobile Robots*, in Proc. IEEE/Robotics Society of Japan International Workshop on Intelligent Robots and Systems, (IROS). Pittsburgh, PA.

[47] A. Marjovi and L. Marques, *Optimal spatial formation of swarm robotic gas sensors in odor plume finding*, Autonomous Robots, vol. 35, pp. 93-109, 2013.

[48] M. Dorigo, D. Floreano, et al., *Swarmanoid: a novel concept for the study of heterogeneous robotic swarms*, IRIDIA, Brussels, Belgium, Tech. Rep. 14-11, July 2011.

[49] D. Fox, et al. *A probabilistic approach to collaborative multi-robot localization.* Autonomous Robots, 2000 Vol 18.1, p 325-344.

[50] M. Rubenstein, C. Ahler, R. Nagpal, *Kilobot: A Low Cost Scalable Robot System for Collective Behaviors*, IEEE International Conference on Robotics and Automation (ICRA), 2012.

[51] J. Borenstein and L. Feng. *Measurement and correction of systematic odometry errors in mobile robots*, IEEE Transactions on Robotics and Automation, 1996.

[52] B. Barshan and H. F. Durrant-Whyte. *Inertial navigation systems for mobile robots*, IEEE Transactions on Robotics and Automation, 1995.

[53] R. Cassinis, *Landmines Detection Methods Using Swarms of Simple Robots*, Proceedings of The 6th International Conference on Intelligent Autonomous Systems (IAS2000), Venice, Italy, July 25-27, 2000.

[54] M. Quigley, B. Gerkey, et al. *ROS: an open-source Robot Operating System*, in Proc. Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), 2009.

[55] J. Borenstein and L. Feng. *Gyrodometry: A new method for combining data from gyros and odometry in mobile robots.* In Proceedings of the IEEE International Conference on Robotics and Automation, pages 423-428, Mineapolis, Minnesota, April 1996.

[56] J. P. Lima, V. Teichrieb, H. Uchiyama, and E. Marchand. *Object Detection and Pose Estimation from Natural Features Using Consumer RGB-D Sensors: Applications in Augmented Reality.* In IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'12, Atlanta, Georgia, November 2012.

[57] W. Lee, N. Park, and W. Woo. *Depth-assisted real-time 3D object detection for augmented reality.* In ICAT '11, pages 126-132, Osaka, Japan, 2011.

[58] L. Alexandre, *3D Descriptors for Object and Category Recognition: a Comparative Evaluation.* IROS Workshop on Color-Depth Camera Fusion in Robotics. 2012.

[59] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. *ORB: an efficient alternative to SIFT or SURF.* In Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV), volume 13, 2011.

[60] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. *Brief: Binary Robust Independent Elementary Features.* In In European Conference on Computer Vision, 2010. 1, 2, 3, 5

[61] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*, 9th European Conference on Computer Vision, 2006.

[62] D. G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[63] R. Bogdan Rusu and S. Cousins, *3D is here: Point Cloud Library (PCL)*, IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China. May 9-13, 2011.

[64] L. Armesto, J. Tornero, and M. Vincze, *Fast Ego-Motion Estimation with Multi-Rate Fusion of Inertial and Vision*, Int. J. Robot. Res., vol. 26, pp. 577-589, 2007.

[65] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, *A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation*, in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013.

[66] *ROS: robot_pose_ekf Package*, accessed April 2015 at http://wiki.ros.org/robot_pose_ekf

[67] T. Moore, D. Stouch, *A Generalized Extended Kalman Filter Implementation for the Robot Operating System*, in International Conference on Intelligent Autonomous Systems,July 18, 2014.

[68] *ROS: robot_localization Package*, accessed June 2015 at http://wiki.ros.org/robot_localization

[69] C. V. Nguyen, S. Izadi, and D. Lovell. *Modeling kinect sensor noise for improved 3d reconstruction and tracking.* 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on. IEEE, 2012.