Scalable Control Strategies and a Customizable Swarm Robotic Platform for Boundary Coverage and Collective Transport Tasks

by

Sean Thomas Wilson

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2017 by the
Graduate Supervisory Committee:

Spring Berman, Chair
Panagiotis Artemiadis
Thomas Sugar
Armando Rodriguez
Jesse Taylor

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Swarms of low-cost, autonomous robots can potentially be used to collectively perform tasks over large domains and long time scales. The design of decentralized, scalable swarm control strategies will enable the development of robotic systems that can execute such tasks with a high degree of parallelism and redundancy, enabling effective operation even in the presence of unknown environmental factors and individual robot failures. Social insect colonies provide a rich source of inspiration for these types of control approaches, since they can perform complex collective tasks under a range of conditions. To validate swarm robotic control strategies, experimental testbeds with large numbers of robots are required; however, existing low-cost robots are specialized and can lack the necessary sensing, navigation, control, and manipulation capabilities.

To address these challenges, this thesis presents a formal approach to designing biologically-inspired swarm control strategies for spatially-confined coverage and payload transport tasks, as well as a novel low-cost, customizable robotic platform for testing swarm control approaches. Stochastic control strategies are developed that provably allocate a swarm of robots around the boundaries of multiple regions of interest or payloads to be transported. These strategies account for spatially-dependent effects on the robots' physical distribution and are largely robust to environmental variations. In addition, a control approach based on reinforcement learning is presented for collective payload towing that accommodates robots with heterogeneous maximum speeds. For both types of collective transport tasks, rigorous approaches are developed to identify and translate observed group retrieval behaviors in *Novomessor cockerelli* ants to swarm robotic control strategies. These strategies can replicate features of ant transport and inherit its properties of robustness to different environments and to varying team compositions. The approaches incorporate dynamical models of the swarm that are amenable to analysis and control techniques, and therefore provide theoretical guarantees on the system's performance. Implementation of these strategies on robotic swarms offers a way for biologists to test hypotheses about the individual-level mechanisms that drive collective behaviors. Finally, this thesis describes Pheeno, a new swarm robotic platform with a three degree-of-freedom manipulator arm, and describes its use in validating a variety of swarm control strategies.

To my friends for constantly distracting me from my work, my sister Mary for keeping me grounded, my parents Helen and Thomas for always facilitating my interests, and my wife Samantha for her continued love and support no matter the distance.

ACKNOWLEDGMENTS

I would first like to thank my advisor Professor Spring Berman whose hard work, patience, and guidance has provided me with opportunities and experiences that have pushed me to excel during this process. I would also like to thank Prof. Panagiotis Artemiadis, Prof. Thomas Sugar, Prof. Armando Rodriguez, and Prof. Jay Taylor for teaching me inside and outside the classroom, serving on my dissertation committee, and providing me valuable feedback that improved the work presented in this thesis. I'd like to extend my gratitude to Prof. Ted Pavlic and Dr. Ganesh Kumar for mentoring me as big brother figures during my first years as a graduate student as well as Prof. Stephen Pratt and Dr. Aurèlie Buffin for introducing me to the fascinatingly complex world of social insects.

I would like to credit my friends old and new who have provided me with chances to relax and step away from the stress caused during graduate work. I'd like to thank all my lab mates who made working in the lab extremely entertaining and who have had conversations with me that have spawned great and terrible ideas.

Finally I want to thank my family; my parents, Thomas and Helen, my sister, Mary, and my in-laws, Oleg and Karen, for always providing me with warm loving homes whenever it was needed. Most of all, I'd like to thank my wife. Since we were going through the process of getting our doctoral degrees at the same time, she has provided me with advice, love, and patience that pushed me through this process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Swarm Robotics

In recent years, there has been an increasing focus on the development of robotic swarms [28] for performing tasks that require high degrees of parallelism, redundancy in system components and behaviors, and adaptability to changes in environmental conditions and failures. These systems would be composed of hundreds or thousands of relatively expendable, resource-constrained robots that operate with little-to-no human supervision.

A major challenge in controlling robotic swarms is the synthesis of robot controllers in a *scalable* manner; i.e., the controller design does not become more complex for larger numbers of robots. One approach to controlling smaller groups of robots is a centralized, deterministic control scheme [29, 71]. In a centralized strategy, one component (a computer, robot, etc.) takes on the majority of the computational load to plan a deterministic series of actions to be executed by each agent while keeping track of the entire swarm's state. This plan is then repeatedly communicated to each robot in the swarm. For example, in a formation control problem, a central computer would plan a path for each member of the swarm such that the robots do not collide with one another or with obstacles while establishing and maintaining a formation. However, as the swarm increases in size, this method becomes infeasible due to limited computational power, limited communication bandwidth, and delay times in communication.

This challenge has motivated the development of control frameworks for swarm robotic systems with a specific set of requirements. In 2004, Erol Şahin proposed the following definition of swarm robotics:

Definition 1.1.1. *Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment [121].*

Since this paper, many swarm control strategies have been developed with several common features taking influence from Şahin's early criterion [20, 45, 103, 104]:

Autonomous Robots: The system should be comprised of physical robotic platforms that are capable of interacting with one another and world around them.

Scale: The number of agents in the system should be "large." This size will vary among applications. However, in any scenario, the strategy should be scalable.

Composition: A swarm robotic system should be composed of large groups of homogeneous robots. For example, a group of robots where each individual has different capabilities is considered less "swarm-like" than a group of robots with a mixture of two types of capabilities.

Individual Capability: Each robot in the swarm should be relatively ill-suited to perform the desired task on its own. That is, either an individual is incapable of performing the desired task alone (e.g., a single robot is not strong enough to transport an object), or a group of robots is more capable of performing the desired task than an individual (e.g., a single robot cannot explore an area as effectively as a group of ten robots).

Limited Sensing and Communication: Each robot in the swarm has limited sensing and/or communication abilities, which constrains the swarm control strategies to be distributed rather than centralized.

Outcome: A robotic swarm should exhibit an desired group behavior. In other words, a swarm of robots with programmed individual behaviors should collectively produce an expected outcome when enough of them operate concurrently.

By these criteria, swarm robotic systems are redundant in composition, capable of performing tasks in parallel, and potentially able to adapt to their surroundings. These properties enable swarms to excel at solving problems in large, unknown, possibly hazardous environments. This thesis presents swarm control strategies with these beneficial characteristics that are designed to solve problems of boundary coverage and collective payload transport.

1.2 Swarm Behaviors in Biological Systems

Swarms in nature provide examples of systems where complex macroscopic tasks can be achieved through the interaction of large numbers of relatively simple individuals. Notable examples are the self-organization behaviors of social insects [22]. Colonies of bees, termites, ants, and other insects are able to perform tremendous feats such as nest construction [54, 57, 72], colony emigration [56, 112, 136], foraging [59, 89, 133], and transport [39, 95] through the interaction of several behavior archetypes, using local sensing and interactions with their environment.

These biological swarms exemplify the criteria set forward by Şahin. Thus, there is an inherent benefit to understanding the underlying behavioral mechanisms of these systems in order to translate them into a swarm robotic framework. Two methods of investigation, both of which are used in this thesis, can be applied toward this end.

The first method involves observing a biological system to infer behaviors that could be implemented on a robotic system. In this case, the focus is on designing robot controllers that are "inspired" by the biological system, in the sense that they inherit certain properties that are desirable for the robotic system. The second method uses robotic systems to test hypotheses about the biological system in order to gain insight into the natural phenomenon being observed. Simple robots can be used to help elucidate the mechanisms that drive collective animal behaviors [58, 60, 75].

1.3 Validation of Swarm Controllers Outside of Simulation

Advances in computing, sensing, actuation, power, communication, and control technologies are currently enabling the production of affordable robots that are designed to act in collectives, both in research and education [23, 44, 64, 97, 120]. In the past few years, the miniaturization of these technologies has led to a plethora of novel platforms for swarm applications, including micro quadrotors [79] and flapping-wing micro aerial vehicles (MAVs) [143]. At even smaller scales, advances in MEMS, low-power VLSI, and nanotechnology are facilitating the development of sub-millimeter self-powered robots [10, 11]. However, many swarm applications continue to be simulated rather than tested on hardware due to the high cost of existing robot platforms or their inability to perform the required tasks. Hence, there is a need for an affordable yet capable robot platform that can be readily customized by users for desired swarm applications. Inexpensive, reliable robots can also be used in education and outreach activities to spark students' interest in STEM fields and demonstrate physical applications of mathematics and physics. Robots that are constructed from components with intuitive interfaces and large user communities are highly beneficial in these contexts.

To address these applications and assist in the validation of the control strategies described in this thesis, a new mobile robot platform named *Pheeno* has been developed with several design criteria in mind. Pheeno is small and affordable, sophisticated enough for multi-robot research experiments, accessible to students and others who are new to robotics, and modular to suit the requirements of different robotic tasks. Chapter 3 describes the development and capabilities of this platform in more detail as presented in Wilson et al. [142].

1.4 Boundary Coverage Tasks for Robotic Swarms

Many potential swarm applications will require the self-organization of robots into groups of different sizes around various regions or objects in their environment (see Figure 1.1). For instance, a swarm may be tasked to transport multiple payloads that are each too heavy for a single robot to retrieve, which would necessitate that enough robots aggregate around each load to move it to a target destination, possibly at a desired speed. However, simply allocating robots to saturation on each payload may be inefficient. Similarly, surveillance tasks may require a swarm to surround different types of regions, such as structure perimeters, to achieve particular degrees of sensor coverage. Other possible applications include environmental monitoring and mapping, automated construction and manufacturing, and disaster response tasks such as cordoning off a hazardous area or extinguishing a fire. At the micro- and nano-scale, applications include micro-object

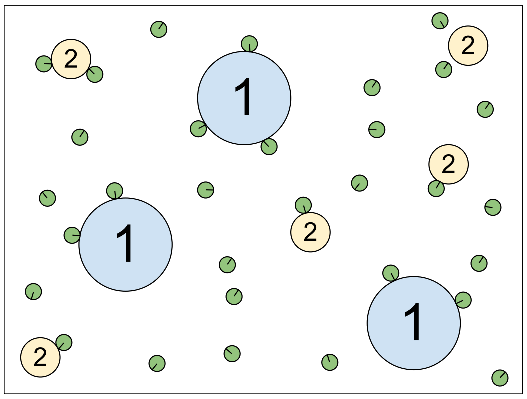


Figure 1.1. Example scenario with two types of disk-shaped regions, labeled 1 and 2. The unlabeled circles are robots that are allocating themselves to the region boundaries.

manipulation, microfactories and nanofactories, and medical monitoring, diagnosis, and treatment. For example, nano-scale robots could collect in desired proportions around objects that are transparent to macroscopic sensing technologies. If the proportions of nano-scale robots were detectable, then the presence of the objects could be inferred.

In order for robotic swarms to reliably carry out these tasks, a rigorous methodology is needed for synthesizing individual robot behaviors that provably result in target robot allocations around boundaries. The swarm control framework must be scalable to arbitrary robot population sizes and accommodate possibly extreme limitations on each robot's sensing, communication, and computation abilities. It must also account for stochasticity arising from noise due to sensor and actuator errors; inherent randomness in robot encounters with each other and with environmental features; and, for nanorobots, the effects of Brownian motion and chemical interactions at scales below tens of micrometers [43].

Chapter 4 develops a control framework with the properties in Section 1.1 for the problem of allocating a robotic swarm in target group sizes around the boundaries of disjoint, stationary objects or regions of different types in an unknown environment. The objective is to achieve a desired percentage of boundary coverage by the swarm (e.g., 50% of the boundary length of a certain type of region is covered). For simplicity, only disk-shaped regions are considered, which are referred to as disks, but this formulation can be extended to other region shapes. The robots have no prior information about the disks and they use only local sensing and local communication, encountering the disks during the course of random walks. Disk types may be categorized according to physical or subjective properties; for instance, size or weight if the disks are payloads to be transported, or relative surveillance value if they are areas to be monitored. Figure 1.1 depicts an example scenario in which the objective is to attain an average allocation of three robots per type-1 disk and one robot per type-2 disk. Stochastic binding and unbinding behaviors of the robots will result in fluctuations around these target allocations, as illustrated by the variation in number of robots bound to each disk type.

A top-down approach is employed to synthesize robot control policies that produce target allocations among the disks with probabilistic guarantees on performance. The stochastic robot interactions with disks and with each other are represented as a well-mixed chemical reaction network (CRN). The robot-to-robot interactions consist of an enzyme-inspired behavior, implemented at the disk boundaries, that greatly reduces the dependence of the allocation strategy on the encounter rates, the difficult-to-characterize probabilities per unit time that a robot encounters an occupied or unoccupied section of a disk boundary. This behavior also decouples allocation tasks that may be occurring in parallel. The CRN formulation allows the abstraction of the system to a macroscopic population model, a set of ordinary differential equations (ODEs) that is amenable to analysis and control. This approach can be implemented by using a supervisory agent to design the model parameters for a particular global objective and broadcast them to the robots. The robots use these parameters to define their stochastic decision-making policies, and the resulting collective behavior follows the macroscopic prediction in expectation.

This work is presented as in Pavlic et al. [109].

Division of Work: Dr. Theodore P. Pavlic and I designed the stochastic controller. Dr. Pavlic developed the CRN model and associated ODEs, and corrected for spatial effects around the boundaries. Dr. Ganesh P. Kumar developed the closed-form solution to the CRN model and approximated the nonlinear ODEs by a linear, large-population approximation. I created the agent-based simulation in NetLogo [139], analyzed the stability of the system, and ran simulations to validate the method.

1.5 Collective Transport Tasks for Robotic Swarms

Cooperative manipulation and transport of heavy payloads will be required in various potential swarm robotic applications, including automated construction, manufacturing, and warehouses; disaster response and search-and-rescue missions; assembly of ships and aircraft; and manipulation, assembly, and construction tasks in inhospitable space and marine environments. In such applications, robots will be tasked to form a team around a payload and coordinate their motion and applied forces to transport the load to a predefined destination. Although various approaches to this task have been developed [14, 33, 52, 62, 119, 130, 132], there remains a need for a rigorous swarm control framework that can produce reliable transport in a wide range of scenarios with arbitrary payloads, unstructured and possibly hazardous environments, and lack of prior information about the loads and environment.

Group food retrieval in ants (Figure 1.2) is a valuable source of design inspiration for multi-robot transport. This phenomenon is a striking example of a cooperative manipulation strategy that is (a) fully decentralized and scalable in the number of transporters, (b) conducted without specialized end effectors, and (c) successful for a wide range of payloads in environments with uneven terrain and obstacles. The behavioral mechanisms underlying group retrieval remain poorly understood [38, 95], but coordination likely depends on indirect interactions through the load itself, known as *stigmergy* [61, 77], although more direct interactions and signaling among transporters may play a role as well. Coordination also depends on some or all of the ants knowing the heading to the nest and being able to maintain this direction using visual or other cues.

This thesis considers a collective transport problem in which a group of robots

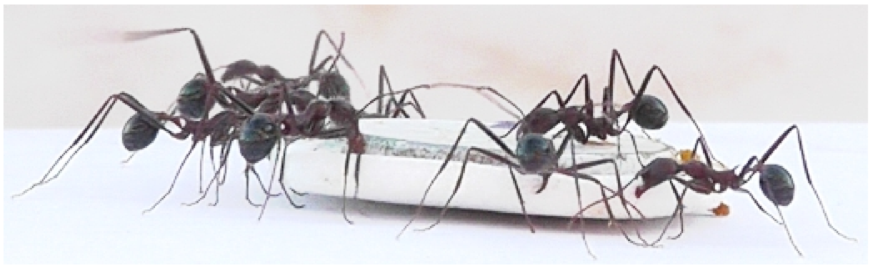


Figure 1.2. Image, from Kumar et al. [78], of a team of *Novomessor cockerelli* ants carrying a weighted circular foam disk along a leftward direction. For size and weight reference, the object on top of the foam load is a U.S. dime (2.3g, 1.8cm diameter).

must move one or several payloads that are too heavy or cumbersome for an individual robot to move. The goal of the transport is to move the load(s) from a starting location to a previously defined destination at a desired velocity. The robots can differentiate between different load types, but they have no prior information on the load quantities and locations. This problem is addressed from two different perspectives, as described below, in Chapter 5 and Chapter 6.

Chapter 5 presents a method that uses the boundary coverage strategies developed in Chapter 4 to maintain multi-robot transport teams of desired sizes around multiple payloads while moving them in a desired direction. Robots can join the team or leave to recharge or perform other tasks, causing the team composition to change dynamically. The swarm control strategy is based on observed attachment and detachment behaviors that are exhibited during group food retrieval in *Novomessor cockerelli* ants. This work focuses on the synthesis of robot control policies for collective transport that: (a) are derived from a dynamical model of the system, enabling theoretical analysis of the transport behavior; (b) allow robots to detach from the payload, dynamically allocating themselves to the transport team; and (c) do not require a centralized coordinator for any information other than high-level task specifications, such as the types of payloads to be retrieved. This work is presented as in

Wilson et al. [141].

Chapter 6 describes an approach to testing hypotheses about ant-like collective towing behaviors using the Pheeno robot platform. Specifically, robots are used to replicate the ant behaviors in order to better understand the factors that produce an observed decrease in steady-state transport speed with increasing team size in *N. cockerelli* ants. This approach develops models of one-dimensional collective towing and applies order statistics to previously obtained ant towing data from [31] to determine possible individual ant behaviors. These ant behaviors are then translated into robot controllers that incorporate a reinforcement learning algorithm in order to replicate the observed group behavior. The work focuses on determining whether the observed decrease in transport speed can be reproduced using the assumed capabilities and prior knowledge of *N. cockerelli* ants. While the main objective of this work is to understand the biological system, robot controllers that are based on the hypothesized ant behaviors can be further analyzed and optimized for specific applications while retaining beneficial properties of the biological strategies.

1.6 Contributions of the Thesis

The contributions of this thesis can be summarized as follows. First, scalable control strategies are developed that provably allocate a swarm of robots among a heterogeneous set of tasks/roles that are associated with the boundaries of distinct spatial regions. These strategies account for spatially-dependent effects on the robots' physical distribution and are designed in such a way that they are largely robust to environmental variations. Second, a control approach is presented for collective payload towing that accommodates robots with heterogeneous capabilities. Third, rigorous approaches are developed for identifying and translating collective transport behaviors that are observed in ant colonies to swarm robotic control strategies. These strategies can replicate features of the biological transport process and inherit its beneficial properties of robustness to different environments and to individual failures and errors. The approaches incorporate dynamical models of the swarm that are amenable to analysis and control techniques, and therefore provide theoretical guarantees on the system's performance. Implementation of these strategies on robotic swarms offers a way for biologists to experimentally test hypotheses about the individual-level mechanisms that drive collective transport behaviors. Fourth, this thesis presents a new type of low-cost, customizable swarm robotic platform with a three degree-of-freedom manipulator arm and describes its use in validating a variety of swarm control strategies.

Chapter 2

LITERATURE REVIEW

This chapter summarizes relevant literature on affordable robotic platforms for swarm experiments; modeling and control of robotic swarms with stochastic behaviors, including behaviors that are similar to adsorption processes; and collective transport strategies by robotic swarms.

2.1 Existing Swarm Robotic Platforms

In recent years, various robot platforms have been developed for multi-robot research and education. Table 2.1 compares several existing platforms that have capabilities similar to the robot described in this thesis, Pheeno.

The r-one robot [96] is an open source platform designed for multi-robot experiments and education. It has a large sensor array for communication and localization. A gripper attachment [98] allows the robot to drag payloads along the ground, although it does not enable 3D manipulation. The platform is not readily expandable to users who lack significant experience with electronics.

The WolfBot [19] is an open source platform designed for distributed sensing and education. It has a sensor suite for communication, localization, and on-board image processing. The platform incorporates the BeagleBone Black computer, which allows the robot to be modular but has less community support than Pheeno's processors, the Raspberry Pi 2 and the Arduino Pro Mini. The cost of parts for the WolfBot is \$550, which is twice the cost of the Pheeno base.

The Khepera IV is an expandable, commercially available research and educational platform. Additional modules have been developed for the robot, including a two

Robot	Processor	Sensing	Communication	Manipulator	Cost
Pheeno	AT-mega328P and ARM Cortex-A7	3D accelerometer, 3D magnetometer, wheel encoders, IR, camera	Serial, WiFi, Bluetooth	RPR serial linkage	\$270 for base (core), \$80 for gripper†
r-one [96, 98]	ARM Cortex-M3	3D accelerometer, 2D gyroscope, wheel encoders, IR, bump, ambient light	Radio, IR	Omni-directional gripper	\$220 - base, \$70 - gripper†
WolfBot [19]	ARM Cortex-A8	3D accelerometer, 3D magnetometer, IR, camera, microphone, ambient light	WiFi, Zigbee	None	\$550†
Khepera IV	ARM Cortex-A8	3D accelerometer, 3D gyroscope, wheel encoders, IR, ultrasonic, camera, ambient light	WiFi, Bluetooth	RR serial linkage	\$3,180 - base, \$2,900 - gripper^{*1}
marXbot [24]	ARM11	3D accelerometer, 3D gyroscope, IR, omni-directional camera, front camera, RFID reader, 2D force	WiFi, Bluetooth	Three-fingered attachment device	N/A
e-puck [100]	dsPIC 30F6014A	3D accelerometer, IR, camera, microphones, range and bearing turret$^{\oplus}$, three-camera turret$^{\oplus}$, omni-directional camera$^{\oplus}$	Zigbee	None	\$1,000 - base, battery, and charger^{*2}
Thymio II [117]	PIC24FJ128GI	3D accelerometer, IR, microphone, temperature, touch	IR	None	\$199 - base^{*3}
Scribbler 2 [9]	P8X32A-Q44	wheel encoders, microphone, IR, ambient light, camera$^{\oplus}$	Serial, Bluetooth$^{\oplus}$	None	\$150 - base^{*4}, \$100 - IPRE Fluke2 Board^{*5}

† Cost of parts. * Retail price. $^{\oplus}$ Additional extensions not included with base robot.

^{1} Available for purchase at http://www.k-team.com/mobile-robotics-products/khepera-iv

^{2} Available for purchase at http://www.gctronic.com/shop.php

^{3} Available for purchase at http://www.techykids.com

^{4} Available for purchase at https://www.parallax.com/product/28136

^{5} Available for purchase at http://www.betterbots.com

Table 2.1. Comparison of currently available multi-robot platforms. Figure from Wilson et al. [142]

degree-of-freedom revolute, revolute (RR) serial linked manipulator capable of lifting 50 g. The robot and gripper attachment retail for approximately $6,000, making them expensive to use in educational curricula and multi-robot research.

The marXbot [24] is a highly capable, modular, open source platform designed for multi-robot experiments. It has a large array of sensors for communication, localization, and onboard image processing. It also has a three-pronged attachment mechanism that can connect to a *hand-bot* [44] for manipulation tasks. Its large sensor suite makes it an expensive platform for multi-robot applications.

The e-puck [100] is a commercially available robot designed for education at the university level. The platform is equipped with sensors for odometry and communication, and its capabilities can be increased with various extensions. However, the robot is fairly expensive for multi-robot applications, retailing for about $1,000.

The Thymio II [117] is an open source platform designed for users with little or no previous experience in robotics. It has a variety of sensors but lacks basic odometry sensors like wheel encoders, a gyroscope, or a magnetometer, although its capabilities can be expanded with accessories such as LEGO components.

The Scribbler 2 [9] is a commercially available, open source educational robot that is suitable for users with a range of programming experience. Additional devices, sensors, and servos can be attached to the robot through a hacker port, and an add-on board can be plugged in to provide a camera and wireless communication with a computer.

There are other robot platforms that can be used for multi-robot research and education, although they are too dissimilar to Pheeno to be included in the comparison table. The Kilobot [118] and GRITSbot [111] are very small, affordable robots that have been developed for swarm robotic experiments. The Pololu 3pi [90] is a small hobby platform with very limited sensing. The LEGO Mindstorms [13] and iRobot Create 2 [90] are designed for education and can be given augmented capabilities through accessories and expansion packs. The LEGO Mindstorms kit has a specific set of sensors and chassis components, allowing up to four sensors and four motors to be driven at once. This inherently limits its sensing and actuation abilities. The iRobot Create 2 is suitable for users with more experience in robotics. Its main chassis contains bump sensors, drop sensors, and encoders, which enable the robot to perceive its environment through collisions but do not allow it to predict and avoid encounters.

2.2 Modeling and Control of Robotic Swarms with Stochastic Behaviors

Much existing work on understanding and controlling robotic swarm behaviors relies on developing an accurate macroscopic model of the population dynamics. The model's dimensionality is independent of the swarm size, which facilitates quick simulation and a scalable control approach. Previous work has addressed stochastic approaches to swarm robotic task allocation, in which the robot task-switching rates are optimized using non-spatial macroscopic models that describe the time evolution of the robot population in each state [15, 36, 85, 93, 106]. Non-spatial swarm models have also been used to optimize stochastic robot behaviors in problems of robotic assembly of parts into products and self-assembly via binding through random collisions [50, 74, 94, 102]. Spatial macroscopic models of swarms that describe robot deterministic and random motion in addition to stochastic task switching have also been developed recently [40, 47, 65, 113].

The utility of the macroscopic model in these works hinges on the ability to accurately determine the non-tunable components of the model parameters. Specifically, in applications where the model captures random interactions between entities in the system, these components are the corresponding encounter rates. However, encounter rates can often be determined only through simulation [63, 68] because the robot motion pattern and sensor footprint and the environment configuration can induce unpredictable spatially dependent effects, or simply spatial effects, on the frequency of robot encounters with other system entities. Encounter-rate formulas based on geometric parameters have been used in previous work on macroscopic swarm modeling of systems in which robots encounter objects that are small [80, 92] or large (and elongated) [37] relative to them. However, these formulas can be applied only for environments with low densities of robots and objects in which robots are uniformly spatially distributed at all times, which is ensured when robots execute random walks and the objects do not bias the robots' movements. The scenario presented in Chapter 4 and Chapter 5 violates these assumptions in that the encountered objects are adjacent to one another and thus not distributed at low density. The implausibility of deriving analytical solutions of encounter rates for this scenario provides motivation to find a way of controlling the swarm without knowledge of these rates while still using a macroscopic model.

A variety of approaches have been developed for controlling task allocation, assembly, and self-assembly in robotic swarms with stochastic behaviors [15, 36, 37, 85, 93, 94, 102, 106, 107]. These control strategies achieve target equilibrium populations by making use of internal timer events that may be tuned to specific parameters of the environment. For example, mobile robots that perform random walks throughout a domain will randomly encounter certain features (e.g., other robots, regions of interest, assembly components, payloads) at an average rate determined by environmental parameters. A stochastic task-allocation strategy will allocate a robot to a task according to some designed probability. Once allocated, the robot will continue to perform the task until an internally generated timeout event occurs and the robot returns to its search. If the allocation probability and return-to-search timeout are properly tuned, then the swarm of robots will converge to some desired allocation across task types in the environment. At that equilibrium, the flux of robots into each task type will match the flux of robots timing out of that task type. However, if the task-type encounter rates change in the environment (eg, due to battery-related speed changes or a change in task distribution), then the timeouts will also have to be changed in order to maintain the same allocation levels. Thus, unlike the control design method presented in Chapter 4 and Chapter 5, these stochastic control strategies for robotic swarms are not robust to changes in the environment.

2.2.1 Relationship to Adsorption Processes

Adsorption describes the process of particles binding to a surface for an amount of time that varies with thermodynamic parameters of the system (i.e., temperature, density, and pressure). The resulting equilibrium distribution of particles on the surface also varies with these thermodynamic properties. In the boundary coverage strategy presented in this thesis, the mechanism by which robots allocate to region boundaries is patterned in part on both reversible processes characterizing Langmuir adsorption [82] and irreversible processes characterizing *random sequential adsorption (RSA)* [49, 134]. If the stochastic strategy relied on internal timer events, then the result would be an artificial adsorption process in which the equilibrium team sizes vary with the swarm size and number of regions. The work in Chapter 4 and Chapter 5 shows that these density-dependent effects can be eliminated by mimicking far-from-equilibrium irreversible processes rather than depending on timeouts.

In Langmuir adsorption, finite-sized particles collide with a surface, bind (adsorb) to the surface with some probability, and then spontaneously unbind (desorb) later after some mean residence time. The system reaches thermodynamic equilibrium, where the flux of binding particles is balanced by the flux of unbinding particles. As the thermodynamic properties of the free particles change (e.g., as their temperature or density increases), the equilibrium allocation of particles on the surface changes. Langmuir processes have been used to design advanced drug delivery devices [137] which selectively target some tissues (e.g., cancers) but have negligible allocations around others (e.g., normal tissue). However, because such allocations are achieved by thermodynamic equilibrium, there must be tight control over the number of devices and knowledge of the thermodynamic variables around the tissues.

In the case of RSA, the desorption process happens at such a relatively large time scale that adsorption is considered to be irreversible. That is, particles adsorb rapidly but then take much longer to desorb, yielding the appearance or approximation that only adsorptions are occurring on random locations on the surface. Consequently, the modeled RSA system is always far from its thermodynamic equilibrium, where adsorptions and desorptions happen on a closer timescale, and instead saturates at some suboptimal packing around the surface. Rényi [116] originally showed that the limiting fraction of a line covered with such sequentially attaching particles is a mathematical constant near 0.75 that has since been called the parking constant [53, 127]. Likewise, the approach in Chapter 4 and Chapter 5 cannot be used for target allocation ratios above 0.75. However, by implementing classical RSA with a second irreversible process that catalyzes desorption (which is otherwise absent), it is possible to stabilize any target allocation ratio between 0 and the parking constant. As a result, the strategy can achieve a continuum of dynamical equilibria without the limitations of a thermodynamic equilibrium. This modification of RSA is trivial to implement in robotic systems with a modicum of agency.

2.3 Collective Transport Strategies for Robotic Swarms

Various collective transport approaches have been designed to assemble large robot teams around objects that are difficult to move. However, the team sizes in such approaches are generally unpredictable, and there is the possibility of a deadlock occurring when no loads accumulate enough robots for movement but the pool of free robots is completely depleted. Thus far, collective transport strategies that are designed for robotic swarms have not yet addressed scenarios with all of the properties that are considered in Chapter 4 and Chapter 5. Some prior approaches require information about the payload and each robot's location on the load, and they utilize a central controller, a leader, or explicit communication between robots [84, 105, 131, 138]. Rubenstein et al. [119] and Christensen et al. [35] derive and experimentally validate a physics-based model for transport of arbitrarily shaped rigid objects by a fixed group of robots. A dynamical model is also derived by Stilwell and Bay [130] for the control of robots that can arrange themselves in a fixed team underneath a palletized load and use a leader-follower transport strategy. Other collective-transport strategies for fixed robot teams have employed algorithms for negotiating the direction of transport in environments with obstacles [52] and for tuning the parameters of neural network controllers using artificial evolution [62].

In several strategies, load transport is performed by a subset of the swarm that changes composition over time; these strategies are investigated through simulations and experiments. O'Grady et al. [107] consider the case in a self-assembly application where broken robots themselves become immobile loads that must be removed by functional teammates. Those teammates must accumulate around immobile loads until a large enough team can move the object. Moreover, the robots must randomly leave these teams to prevent deadlock conditions. An approach to box pushing is described by Kube and Bonabeau [77] in which robots can leave the transport team and reposition themselves on the load, switching between simple behaviors in response to locally sensed cues. Similarly, Chen et al. [33] organize a set of pushing robots behind a tall object by taking advantage of the object's ability to block visual contact with a destination marker. In other approaches, the robots do not deliberately approach the load and propel it toward a specific location but rather affect transport through their contact with the load under the influence of an external control input. The mechanism of granular convection is exploited by Sugawara et al. [132] to drive a load to a target destination through a swarm of randomly moving robots that experience a repulsive force from the destination. Becker et al. [14] use a broadcast control input to steer the swarm around the environment and push an object encountered by the mass of robots to a target configuration.

Chapter 3

PHEENO ROBOTIC PLATFORM

Source: Wilson et al. [142]

ABSTRACT

Swarms of low-cost autonomous robots can potentially be used to collectively perform tasks over very large domains and time scales. Novel robots for swarm applications are currently being developed as a result of recent advances in sensing, actuation, processing, power, and manufacturing. These platforms can be used by researchers to conduct experiments with robot collectives and by educators to include robotic hardware in their curricula. However, existing low-cost robots are specialized and can lack desired sensing, navigation, control, and manipulation capabilities. This chapter presents a new mobile robot platform, Pheeno, that is affordable, versatile, and suitable for multi-robot research, education, and outreach activities. Users can modify Pheeno for their applications by designing custom modules that attach to its core module. This chapter describes the design of the Pheeno core and a three degree-of-freedom gripper module, which enables unprecedented manipulation capabilities for a robot of Pheeno's size and cost. Several preliminary experiments are presented that demonstrate Pheeno's ability to fuse measurements from its onboard odometry for global position estimation and use its camera for object identification in real time. Finally, an experiment demonstrates that groups of two and three Pheenos can act on commands from a central controller and consistently transport a payload in a desired direction.

To implement and further validate swarm robotic strategies, like those discussed in Chapter 4, Chapter 5, and Chapter 6, a robotic platform is required. However, current commercially available platforms are either too specialized and/or lack desired features. Thus, the objective of this research is to create a low-cost robot that can be customized by users to conduct a variety of multi-robot research experiments. Toward this end, Pheeno has been designed to be a modular platform that is constructed from commercially available components, including low-cost processors, an array of basic sensors, and 3D printing plastic filament. The list of robot components, schematics for PCB boards, CAD designs for the core and gripper modules, and guides to assembly, calibration, and programming of the platform are publicly available in an online repository [4]. Section 3.1 and Section 3.2 were first presented in Wilson et al. [142].

The robot is composed of a core module, described in Section 3.1.1, that users can interface to their own custom-designed modules for desired applications. Section 3.1.2 describes one such module, a three degree-of-freedom gripper module that enables the robot to manipulate and transport objects either individually or in cooperation with other Pheeno platforms. Pheeno's sensing, navigation, processing, communication, and cooperative manipulation capabilities are demonstrated in a series of experiments presented in Section 3.2 and Section 3.3. Finally, recent upgrades to the original platform and its capabilities are discussed in Section 3.4. Pheeno has been used at many outreach events to spark interest in STEM fields, providing K-12 students the opportunity to learn from programming and controlling the platform.

3.1 Design

3.1.1 Core Module

Figure 3.1 shows the fully assembled core module of Pheeno with an exploded SolidWorks rendering. The cylindrical core has a diameter of 12.7 cm and a height of 11.1 cm. Most of the components are 3D printed using standard ABS plastic, which allows for easy replication and modification of the core. The only components that are not 3D printed are the motor mounts as well as the circular base and cap of the housing. These are standard robotic chassis parts sold by Pololu Robotics & Electronics [6], used to reduce the printing time of the robot. Currently, a Pheeno core takes about 5 hours to print with a MakerBot Replicator 2X 3D printer (MakerBot Industries) using an infill of 12%.

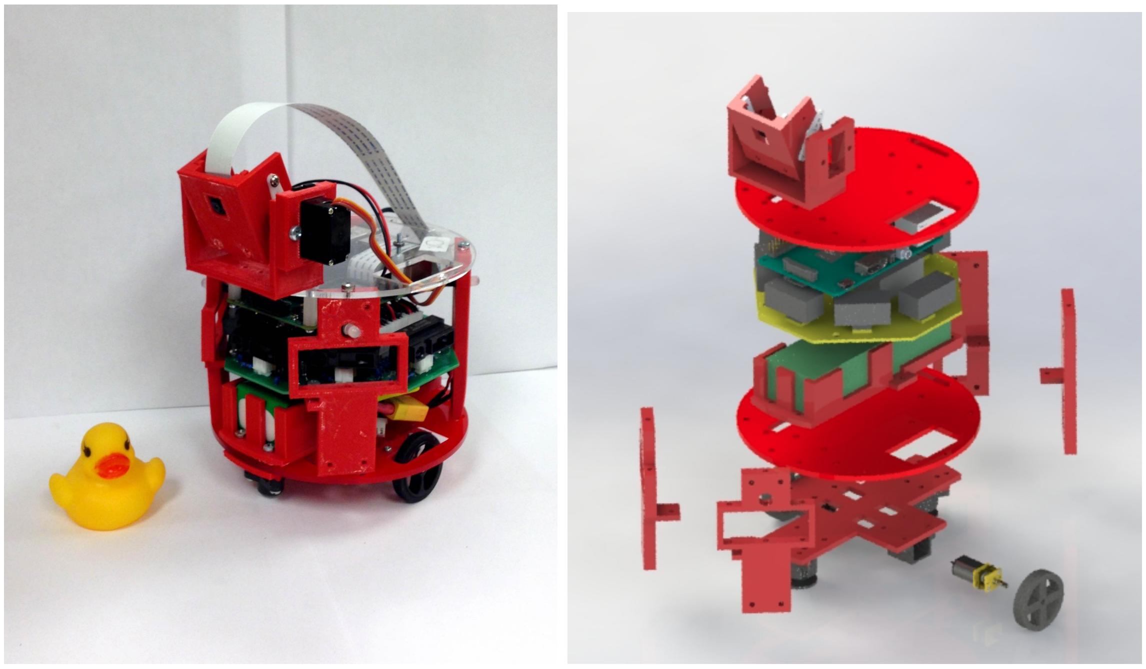


Figure 3.1. *Left:* The Pheeno robot platform with the ICRA 2016 *duckie* [3] for scale. *Right:* Exploded SolidWorks rendering of Pheeno.

The core module is a differential drive platform that is actuated by two standard micro-metal gear motors with extended back shafts and supported by two caster wheels to maintain the balance of the robot. Motors with a 51.45:1 gear ratio are currently used, but standard micro-metal motors with higher or lower gear ratios can be substituted, allowing Pheeno to be quicker or more powerful to suit the user's needs. With the current gear ratio and 32-mm-diameter wheels, Pheeno can move at controllable speeds between 4 cm/s and 42 cm/s. The extended back shafts enable the attachment of magnetic quadrature wheel encoders, which have a linear resolution of 0.163 mm/tick. The robot can measure its acceleration and heading using an STMicroelectronics LSM303D that contains a 3D accelerometer and a 3D magnetometer.

The main processors onboard Pheeno are the Raspberry Pi 2 Model B microprocessor [5] and the Arduino Pro Mini microcontroller (3.3V model) [1]. These boards were chosen for their accessibility to new users and the large user communities supporting them. The Raspberry Pi is a credit-card sized Linux computer that enables users to program the robot in a range of languages. The Arduino Pro Mini is a small (1.8 cm $\times$ 3.3 cm) member of the widely used Arduino microcontroller family. These two boards interact through serial communication. Pheeno uses the Raspberry Pi for high-level control and image processing and the Arduino Pro Mini for control of low-level actuation and sensor data processing for accurate navigation. All of the GPIO pins on the Arduino Pro Mini are occupied by connections to the accelerometer, magnetometer, wheel encoders, H-bridge motor driver, and infrared (IR) sensors on the core module. The Raspberry Pi has 26 open GPIO pins that can be used for sensor inputs and actuator control, as well as 3 open USB ports that allow the use of various USB adaptors such as WiFi and Bluetooth for communication.

The core module is equipped with six IR proximity sensors for enabling collision

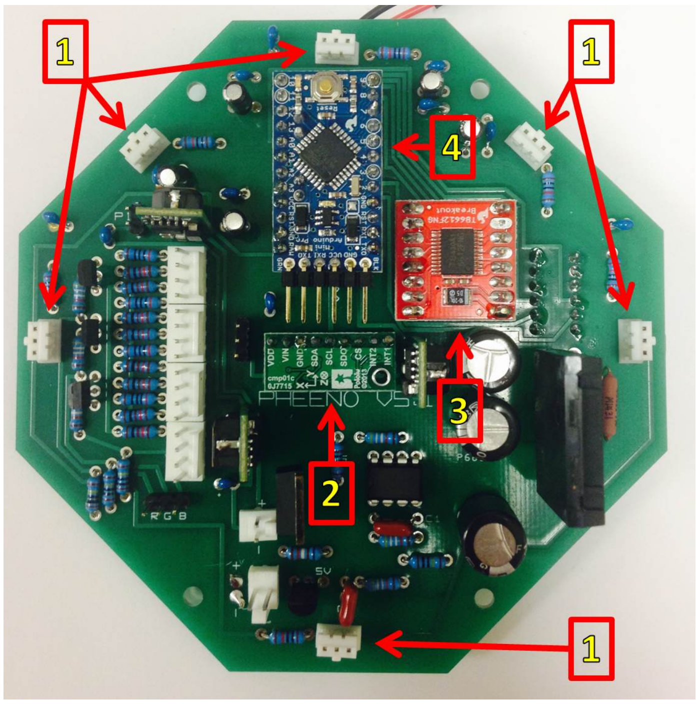


Figure 3.2. Top-down view of the main Pheeno circuit board with the major components numbered. **1)** Infrared (IR) sensor mounts able to interface with any 3-pin JST connector. **2)** 3D accelerometer and magnetometer. **3)** Motor control board (H-Bridge). **4)** Arduino Pro Mini microprocessor.

avoidance, a Raspberry Pi camera for vision-based object detection, and four RGB LEDs for displaying the robot's state. Five of the IR sensors are evenly spaced along the front perimeter of the robot, and one sensor is placed on the back to detect nearby objects. Currently, Pheeno is equipped with six Sharp GP2Y0A41SK0F IR sensors with a range of $4-30$ cm. IR sensors with different ranges could be substituted if they interface with 3-pin JST PH connectors on the custom printed circuit board (PCB), shown in Figure 3.2. The Raspberry Pi camera, a 5MP Omnivision 5647 sensor in a fixed focus module, is mounted on top of the core. A servomechanism tilts the pitch angle of the camera within a 180° range. The LEDs are evenly spaced around the perimeter of the core.

The robot is powered by a 11.1V 3000 mAh LiPo battery that is secured to the bottom of the chassis. The current draw of the robot is 120 mA when idle and 410 mA during typical use. When the motors are stalled and image processing is performed onboard, the current draw can spike to 670 mA. During demonstrations at outreach events, where visitors remotely controlled the robot through a graphical user interface on a laptop computer [67], the robot operated continuously for 5 hours while retrieving small objects and streaming 1080 × 720p video to the laptop.

3.1.2 Gripper Module

In order to give Pheeno manipulation capabilities, a gripper module (Figure 3.3 and Figure 3.4) was designed that consists of a standard 3-degree-of-freedom (DOF) revolute, prismatic, revolute (RPR) serial arm with an end-effector capable of grasping an object. The joints of the arm are driven by three standard servos, which provide the arm with the ability to lift an object up to 6.2 cm, roll it up to 180° about the core's radial axis, and rotate it up to 180° about the core's central vertical axis.

Most of the gripper components are 3D printed using standard ABS plastic, al-

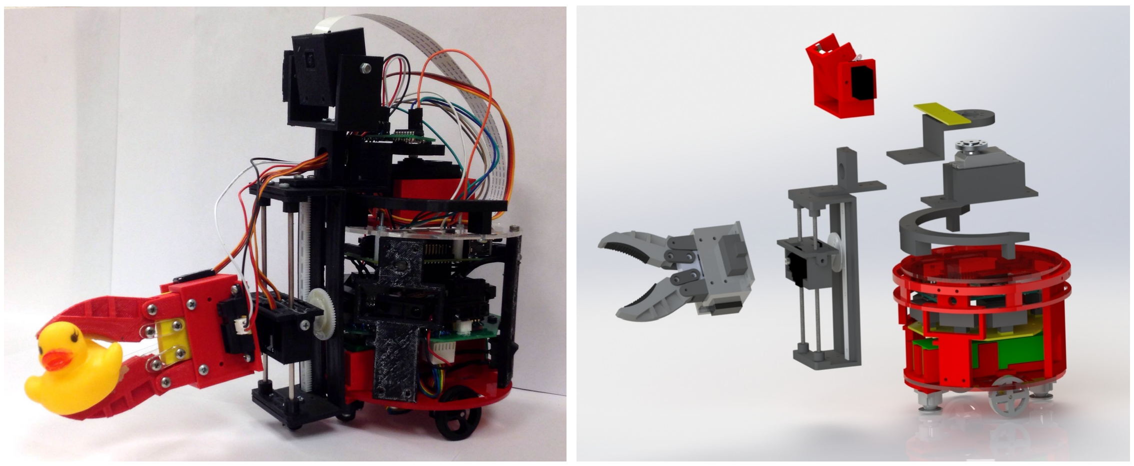


Figure 3.3. *Left:* The Pheeno robot platform with gripper module holding an ICRA 2016 *duckie* [3] for scale. *Right:* Exploded SolidWorks rendering of the Pheeno gripper module.

lowing for easy modification and replication. The rack gearing that provides the prismatic motion and two shafts that stabilize the lift are available commercially from [2, 7]. The gripper jaws are composed of a rigid structure with molded urethane rubber pads that deform while grasping an object. The gripper is underactuated, driven by a single servo through a yoke mechanism. The Raspberry Pi camera from the core module is affixed to the top of the gripper module, allowing visual servoing of the gripper if desired. Additional sensors included in the gripper module are a front-mounted IR sensor and a potentiometer to give feedback on grasping.

Another PCB is included in the module to provide power to the servos and take in

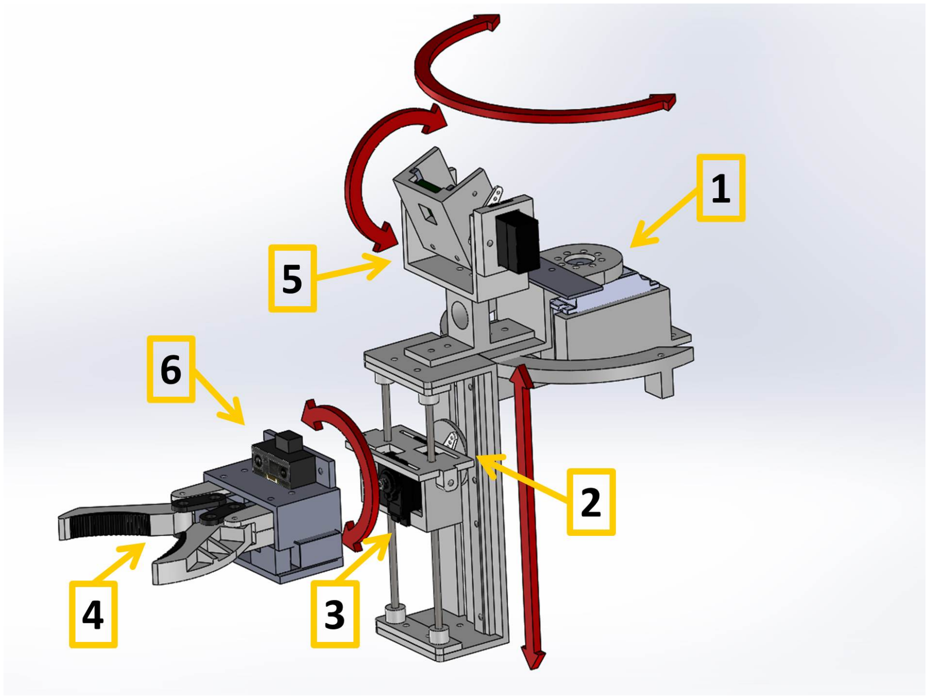


Figure 3.4. A SolidWorks rendering of the gripper module with degrees of freedom and components shown. **1)** Yaw servo, which enables 180° rotation about the central vertical axis of the Pheeno core module. **2)** Gear rack servo, which enables 6.2 cm of prismatic motion. **3)** Wrist servo, which enables 180° rotation about the radial axis of the Pheeno core. **4)** Underactuated gripper with potentiometer feedback. **5)** Core module camera. **6)** IR distance sensor.

sensor inputs. An 8-channel 10-bit ADC allows these analog sensors to be connected to the Raspberry Pi, enabling closed-loop control of the gripper through image processing and sensor feedback. The PCB has been designed with two open channels for the ADC, which allows additional analog sensors to be interfaced with the gripper. When Pheeno is driving and moving its gripper during typical use, its current draw is 640 mA. In situations where the gripper servo and drive motors are stalled and the lift servo is strained while image processing is being performed onboard, the current draw jumps to 1.08 A. A large battery capacity was chosen to accommodate these high power demands.

The arm is capable of lifting and manipulating a weight of about 400 g. This allows Pheeno to manipulate light objects independently or cooperate in a team to transport heavier objects. When a robot manipulates an object by itself, its yaw servo can be actively controlled to rotate the object about the core module's central vertical axis independently of the rotation of the wheel base. During collective transport tasks, each robot can turn off its yaw servo, which allows its drive train to backdrive the yaw servo and rotate to a desired heading within a 180° range. This enables the robots to simultaneously grasp a load while driving in a common direction at different angles relative to their manipulator arms. Once a robot detaches from the load, its yaw servo can become active again to return the gripper to its forward configuration.

3.2 Experimental Tests of the Pheeno Robot's Capabilities

Three types of experiments were conducted to evaluate Pheeno's ability to localize using its onboard odometry, identify objects by color from its camera images, communicate through WiFi, and cooperatively manipulate an object as a team. The experimental results are discussed in this section.

Dead Reckoning A path-following algorithm was implemented in two experiments in order to test Pheeno's ability to determine its (x, y) coordinates in a global reference frame using only onboard odometry. Pheeno's heading is defined as θ, its angular speed as ω, and its translational speed as v. The robot's motion can be modeled as unicycle dynamics:

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \omega$$

A standard PID controller was used to drive the robot to a desired position (x_d, y_d) at a constant speed while tracking a desired heading,

$$\theta_d(t) = \arctan\frac{y_d - \hat{y}(t)}{x_d - \hat{x}(t)},$$

where $(\hat{x}(t), \hat{y}(t))$ is the robot's estimate of its position at time t. This state estimate was updated at a rate of 15 Hz. In the first experiment, Pheeno estimated its global position and orientation using only measurements from its encoders. In the second experiment, a set of complementary filters was applied to the platform's encoder, accelerometer, and magnetometer measurements in order to correct for errors due to wheel slipping, which can occur when the robot accelerates and makes fast turns (Appendix B). A general block diagram of a complementary filter is shown in Figure 3.5. For this sensor fusion, the encoder measurements were high-pass filtered and the accelerometer and magnetometer readings were low-pass filtered using a first-order filter.

The target trajectory and actual robot trajectory in the first and second experiments are compared in Figure 3.6a and Figure 3.6b, respectively. In both experiments, drift in the robot's position from the target trajectory is unavoidable. In the second experiment, the inclusion of the accelerometer and magnetometer measurements produces a more accurate estimate of the robot's global position compared to the first

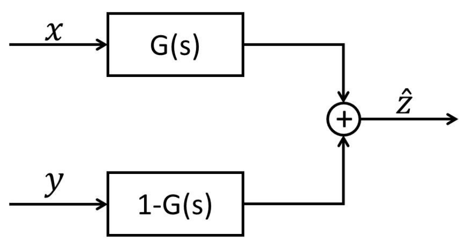
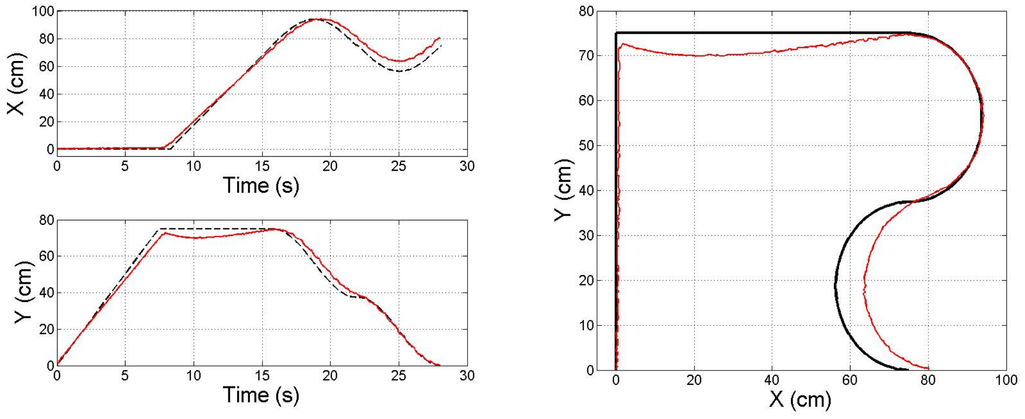
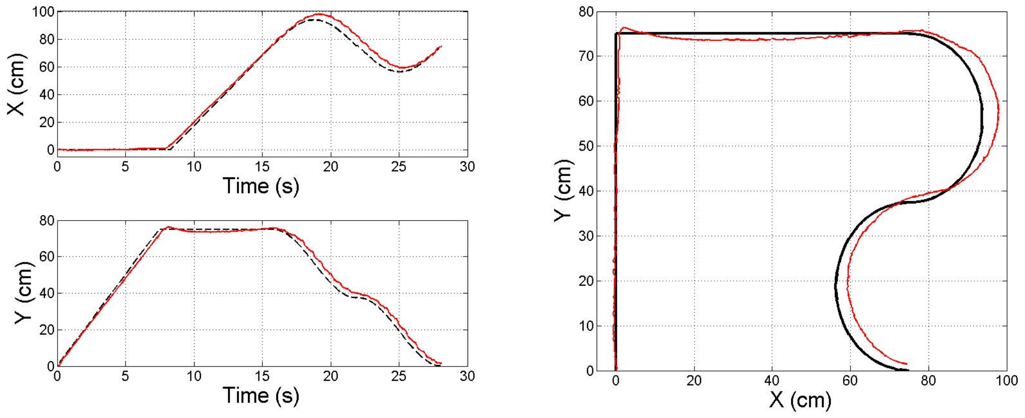


Figure 3.5. A block diagram of a complementary filter. Here, x and y are measurements of a state z, and $\hat{z}$ is the estimate of this state by the filter.

experiment, in which only encoder measurements are used. However, the resulting improvement in tracking performance is relatively small; moreover, the localization approach in the first experiment is easier to implement and explain in an educational setting.



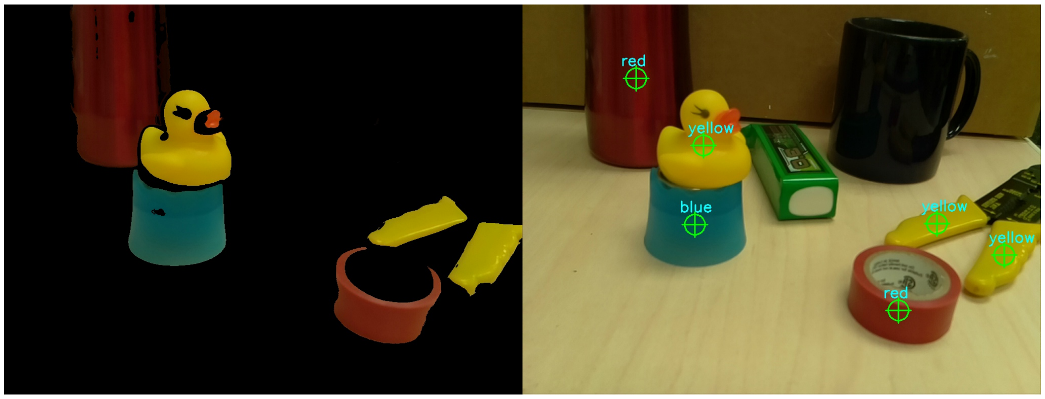
(a)



(b)

Figure 3.6. Pheeno's actual trajectory (red line) as it tracks a predefined trajectory (black line) while localizing using (a) only encoder measurements, or (b) a fusion of encoder, accelerometer, and magnetometer measurements.

Image Processing To evaluate Pheeno's onboard image processing capabilities, measurements were taken of the time required for the robot to (1) acquire image frames from its camera for processing, and (2) perform contouring to identify the centers of mass of shapes with different colors in the image. These tests were run for the image in Figure 3.7 at three different resolutions and used image processing algorithms from the OpenCV library [27]. The images were transformed to the HSV color space, thresholded for red, yellow, and blue, and contoured using the standard transform, thresholding, and Canny contouring functions from OpenCV 2.7 in Python 2.7.10. Figure 3.7 shows an output of the color tracking algorithm with the color blob centers of mass identified. Table 3.1 lists the minimum, average, and maximum times to acquire and process 200 frames at different image resolutions. The data shows the expected tradeoff between resolution and processing speed and demonstrates that the image processing routines are performed with reasonable sampling times at lower resolutions.



(a) (b)

Figure 3.7. (a) A masked image from Pheeno's camera after contouring for red, yellow, and blue. (b) Identification of each color blob's center of mass.

Resolution (pixels)	**Image Acquisition** (sec)	**Contouring** (sec)
320×240	[0.010, 0.023, 0.040]	[0.170, 0.172, 0.220]
640×480	[0.050, 0.063, 0.090]	[0.660, 0.672, 0.680]
1024×768	[0.130, 0.139, 0.180]	[1.760, 1.781, 1.800]

Table 3.1. Time to acquire an image and perform image processing routines at different resolutions. The results are from 200 captured frames. The data take the form *[minimum, average, maximum]* time to process a frame in seconds.

3.2.1 Collective Transport

Algorithm 1 was implemented on several Pheenos equipped with gripper modules in order to test the platform's ability to perform cooperative manipulation tasks. The experiments were performed in the UCLA Applied Mathematics Laboratory, directed by Prof. Andrea Bertozzi. Three robots were placed in a 1.5 m $\times$ 2.1 m arena with a single circular payload of height 8 cm, diameter 20 cm, and weight 150 g. The robots and payload are marked with 2D binary identification tags to enable real-time tracking of their positions and orientations. The tags are tracked using two overhead Imaging Source DMK 21F04 1/4" Monochrome CCD cameras with a resolution of 640 $\times$ 480 pixels at a frame rate of 30 FPS. The robots and load are identified from their tags using the thresholding, boxpoint, and contouring OpenCV libraries on a Windows computer. A *control computer* serves as a pseudo-GPS, path planner, and communication hub for the robots. The control computer and the robots communicate with each other using WiFi. The overall control architecture of each robot is shown in Figure 3.8.

Robot path planning The control computer selects a *goal position* a set distance away from the load perimeter for each robot (see Figure 3.9). The first is designated

Algorithm 1: Transport strategy for a single robot

```
while not at goal position around load do
    move to next waypoint
end
while entire team not at goal positions around load do
    wait
end
if gripper IR sensor measurement $> 8$ cm then
    move forward
else
    rotate core to target transport direction
    rotate gripper back to load
    grasp load
end
while entire team not grasping load do
    wait
end
lift load
carry load in target transport direction
```

as the position of a robot that will pull the load backward in the desired direction of transport. The control computer assigns this position to the robot that is closest to it. The remaining goal positions are evenly spaced around the load according to the number of robots in the team. Starting from the first goal position, the control computer sweeps the image in the counterclockwise direction and assigns the next detected robot to the next goal position around the load.

The control computer plans each robot's path as a series of points from its initial position to the goal position. A waypoint algorithm is used to design robot paths that circumvent the load, avoiding robot-load collisions. The control computer sends each robot its next waypoint and acts as a pseudo-GPS, updating the robot's global position and orientation at a rate of 3 Hz to correct for errors in the onboard state estimates described in Section 3.2. When a robot determines that it has reached its waypoint, it requests the next one from the control computer. If it has reached its final location around the load, it waits for all robots to communicate to the control

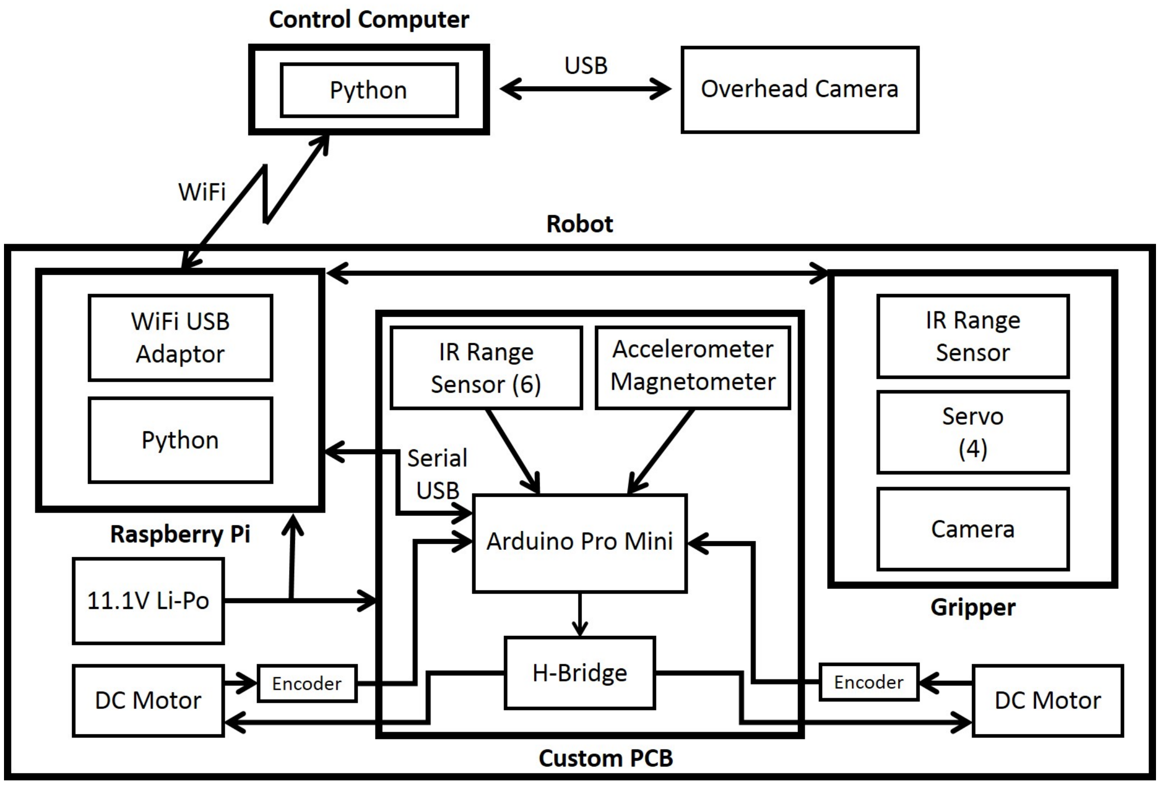


Figure 3.8. Robot control architecture for collective transport.

computer that they have reached their goal locations.

Load grasping and transport The control computer notifies the robots once they all have reached their assigned positions around the load. Next, the robots must orient themselves in the desired direction of transport and grasp the load, as shown in Figure 3.10. In the *Approach* phase, each robot drives forward until the reading from the IR proximity sensor on its gripper drops below 8 cm, indicating that the load is within its gripper pincers. The robot then records its current orientation and enters the *Core Rotation* phase, during which its core module rotates so that its heading aligns with the direction of transport. If the robot ends up facing the desired direction, then it will drive forward during transport; if it is facing the opposite direction, then

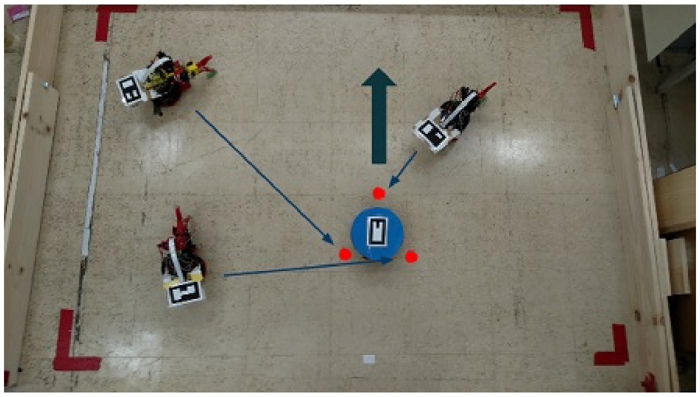


Figure 3.9. Goal position assignments for three Pheenos prior to transport in the direction of the large arrow.

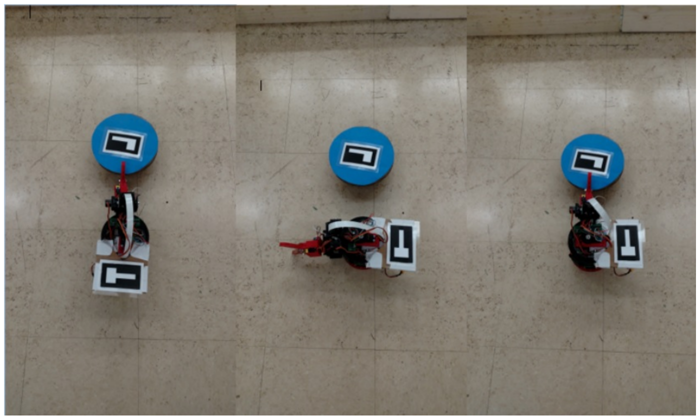


Figure 3.10. Pheeno reorienting itself and grasping the load in preparation for transport. From left to right: the *Approach* phase, *Core Rotation* phase, and *Gripper Rotation* phase.

it will drive backward. The robot calculates the angular difference $\Delta\theta$ between its initial and final headings using its onboard magnetometer, and in the subsequent *Gripper Rotation* phase, its gripper rotates an angle $-\Delta\theta$ back toward the load.

A robot's onboard camera is used to correct any misalignment of its gripper with respect to the load due to noise in the magnetometer readings and error in the rotation of the core and gripper. The Raspberry Pi uses an algorithm similar to the one described in Section 3.2 to determine the center of the load, which is colored blue. Two PI controllers regulate the yaw angle of the gripper and the pitch angle of the camera to align the center of the thresholded image with the center of the frame. Once this alignment is achieved and the load is gripped, the robot communicates to the control computer that it is ready to lift the load and waits for confirmation.

After all robots have communicated they are ready to lift, the control computer sends them a command to start the transport. At this point, the control computer no longer serves as a pseudo-GPS or communication hub. The robots simultaneously lift the load, turn off their yaw servos to allow passive yaw rotation of their grippers, and drive in the direction of transport while maintaining their headings with a PI controller that acts on each robot's magnetometer readings. The robots continue transporting the load until they exit the arena defined by the overhead cameras' view.

Transport Results Transport experiments were performed with teams of two and three robots, with the same transport task repeated five times for each team size. The initial positions and orientations of the load and robots were chosen using a random number generator. Figure 3.11 plots the robot trajectories during the path planning phase of the experiments. The robots follow very similar trajectories during each trial, demonstrating that they can reliably communicate with a central computer hub to localize and receive commands. The slight discrepancies in robot paths across trials are likely due to errors by the control computer in reading the fiducial tags and variations in each robot's determination of whether or not it has reached a particular waypoint along its path. Figure 3.12 plots the load trajectories during the transport phase of the experiments. For each team size, the load follows approximately the same trajectory and travels in the desired direction, indicating that the robots are

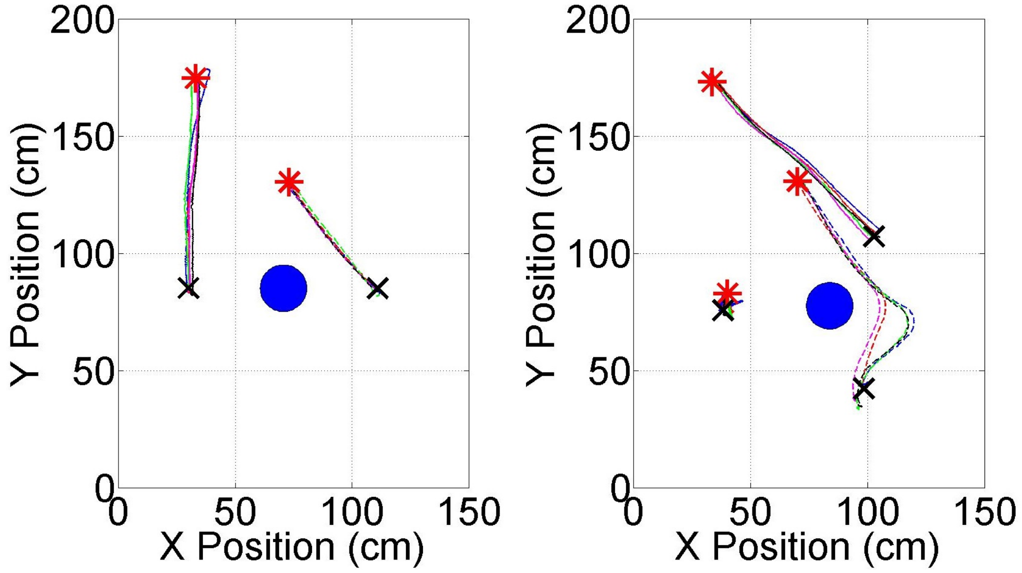


Figure 3.11. The individual robot trajectories during the path planning phases of five transport trials with a team of (*left*) two robots and (*right*) three robots. Trajectories with the same line style correspond to the same robot, and trajectories with the same color correspond to the same transport trial. Robots begin at the red stars and move to the black ×'s, and the blue circle represents the load.

able to consistently achieve stable transport of the load in a target direction without communication. Small variations in the initial load position are due to human placement error, and discrepancies in the load trajectories are likely due to noise in the magnetometer readings of each robot.

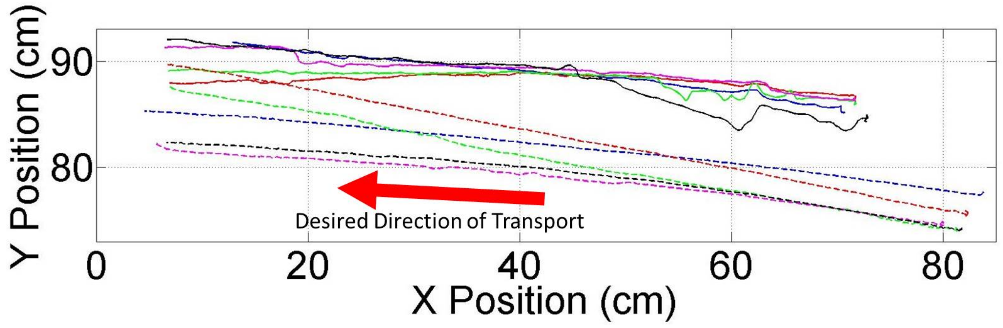


Figure 3.12. The load trajectory during five transport trials with a team of two (solid lines) and three (dashed lines) robots.

3.3 Experimental Validation of Novel Control Strategies

Pheeno has also been used to validate other mobile robot controllers developed in the Autonomous Collective Systems Lab. This section will discuss them briefly.

3.3.1 Confinement Control of Double Integrators using Partially Periodic Leader Trajectories (Elamvazhuthi et al. [48])

This work uses an open-loop oscillatory leader strategy to confine a follower or group of followers. The confinement is done via interaction potentials that are of gravitational type but repulsive in nature. A system is considered with a single *leader agent*, whose position at time t is given by $[x_l(t), y_l(t)]^T$, and N *follower agents*, whose positions at time t are $[x^i(t), y^i(t)]^T$, $i = 1, ..., N$. The x and y velocity components of follower agent i at time t are denoted by $[v_x^i(t), v_y^i(t)]^T$. The dynamics of the leader and followers are defined by the following system of equations:

$$\begin{aligned} \dot{x}_l &= u_x \\ \dot{y}_l &= u_y \\ \dot{x}^i &= v_x^i \\ \dot{v}_x^i &= \frac{A(x^i - x_l)}{[(x^i - x_l)^2 + (y^i - y_l)^2]^\alpha} - kv_x^i \\ \dot{y}^i &= v_y^i \\ \dot{v}_y^i &= \frac{A(y^i - y_l)}{[(x^i - x_l)^2 + (y^i - y_l)^2]^\alpha} - kv_y^i \end{aligned} \tag{3.1}$$

where $A, k, \alpha \in \mathbb{R}^+$ and $i = 1, ..., N$. This model produces one-way interaction between the leader and the followers for any non-zero value of A.

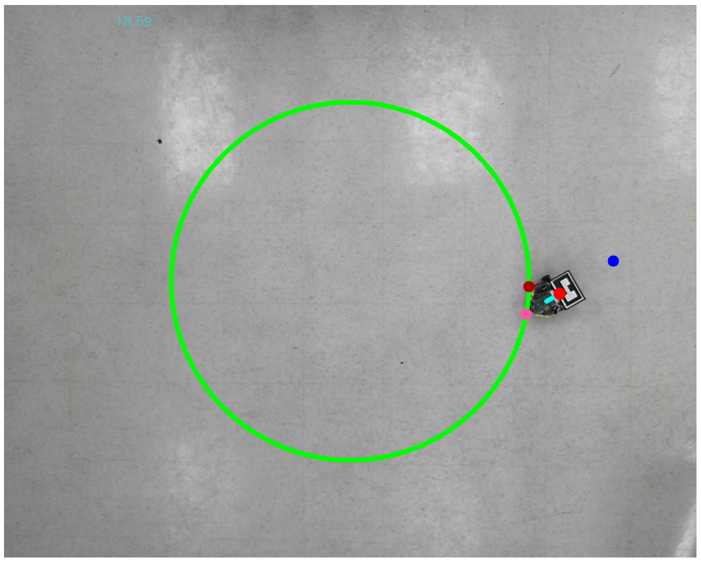
This confinement control approach was validated with an experiment in which Pheeno emulated a follower agent that was confined by a virtual leader. One robot was placed in a 1.5 m $\times$ 2.1 m arena and marked with a 2D binary identification tag to allow for real-time position and orientation tracking. The tags were tracked using one overhead Microsoft LifeCam Studio Webcam with a resolution of 1920 $\times$ 1080 pixels at a frame rate of 30 FPS. The 2D binary tags were identified using standard OpenCV algorithms on a Windows computer.

To accommodate the nonholonomic constraint of the robot platform, the robot used a PI controller to keep its heading facing away from the virtual leader at all times. This heading control produces approximate holonomic motion in a differential-drive platform. To facilitate this motion, the leader's oscillation frequency was maintained below the bandwidth of the robots' motors. A desktop computer updated and transmitted the virtual leader's position and the robot's global position to the robot via WiFi at a frequency of 3 Hz while the robot updated its position onboard at a frequency of 10Hz.

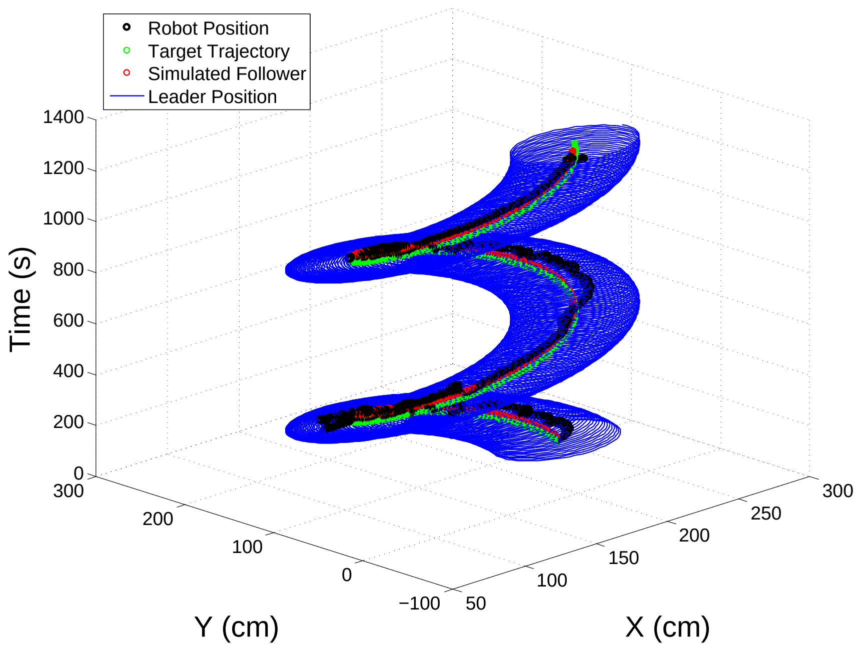
A circular target trajectory was defined, shown in Figure 3.13a, for the robot to track in the experiment. The trajectory was given by $\gamma_E(t) = [R_p \cos(\omega_p t) \;\; R_p(\sin \omega_p t)]^T$, where $R_p = 67$ cm and $\omega_p = 0.02$. The robot was controlled using the parameters $A = R_L^{2\alpha-1}$, $R_L = 38$ cm, $k = 1$, $\alpha = 4$, and $\omega = 2$.

The trajectory of the robot during the experiment was compared with the corresponding trajectory of a simulated follower agent, indicated by a dark red dot in Figure 3.13a, that moves under the influence of the same leader agent. Note that while the robot is nonholonomic, the simulated follower is holonomic. Figure 3.13b plots the trajectories of the virtual leader, the robot, and the simulated follower agent during the experiment. The robot was initially placed inside the region of confinement bounded by the leader's path, and the simulated follower was initialized at the center of the leader's orbit. The figure illustrates that the robot was successfully kept inside the region of confinement for the duration of the experiment.

Figure 3.13b shows that the trajectory of the robot deviates farther from γ_E than the trajectory of the simulated follower. This is likely due to the nonlinear effects of friction on the shafts of the robots' motors. The motors require a minimum input voltage to drive the robot's motion, which can be controlled at speeds above 2 cm/s. These speeds are generated by the repulsive interaction potential in the robot's controller when the distance between the robot and the leader is below a threshold value. Hence, the robot moves away from the leader only when it is within this distance; otherwise, it is stationary. In contrast, the simulated follower does not have such a constraint on its speed and will move when the leader is farther away, causing its trajectory to adhere more closely to γ_E. Thus, the experiment shows that when the confinement strategy is implemented in practice, the controller design must account for unmodeled dynamics in the physical platform by allowing only those velocity control inputs that exceed an appropriate lower bound.



(a)



(b)

Figure 3.13. (a) The robot being herded by a virtual leader along a target trajectory γ_E, shown in green. The blue dot is the virtual leader's position, the pink dot is the center of the leader's orbit, and the dark red dot is the position of a simulated follower agent. The light blue line on the robot indicates its orientation. (b) Time evolution of the positions of the leader agent (blue), simulated follower agent (red), and robot (black) for an experiment with the target trajectory γ_E (green).

3.3.2 Decentralized Sliding Mode Control for Autonomous Transport by Multi-Robot Systems (Farivarnejad et al. [51])

In this work, a sliding mode control law was defined for a fixed group of robots to transport a payload in a desired direction, ϕ_{des}, at a desired speed, v_{des}. The control laws for each robot's heading, ϕ, and speed, v, are defined as:

$$\dot{\phi} = -k_{\phi} sgn(s_{\phi}) \tag{3.2}$$

and

$$\dot{v} = -k_{v} sgn(s_{v}) \tag{3.3}$$

where,

$$s_{\phi} = \phi - \phi_{des} \qquad \text{and} \qquad s_{v} = v - v_{des}. \tag{3.4}$$

A unicycle model transformation was applied to these controllers to enable their implementation on the robots.

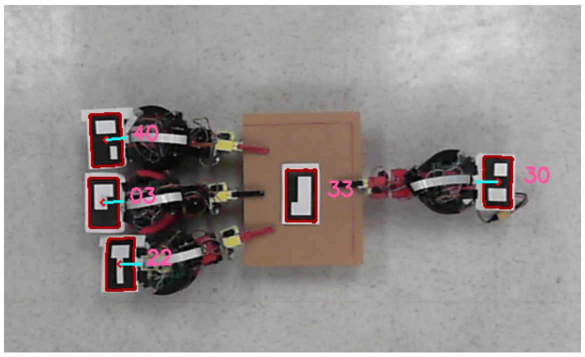


Figure 3.14. A zoomed-in screenshot from the overhead camera that tracked the load and robots during the experiment.

To validate the control strategies, five experimental trials of collective transport were conducted with four Pheeno robots and a rectangular load. The robots and load were marked with 2D binary identification tags to enable real-time tracking of their positions and orientations by an overhead camera. The robots were initially placed in the configuration shown in Figure 3.14. This configuration was chosen to minimize unwanted effects such as wheel slip and unnecessary stress on the central servo, which controls the yaw angle of the manipulator arm about the central axis of the robot. Each robot updated its state estimate using a basic complementary filter acting on its onboard encoders, compass, and accelerometer. The controllers Equation 3.2 and Equation 3.3 were implemented on the robots with the parameters set to $k_\phi = 0.01$, $k_v = 0.05$ and a sliding mode boundary layer parameter $\epsilon_b = 0.01$.

The robots were tasked with transporting the load at a desired velocity of $v_{des} = 10$ cm/s along the x-axis of the global frame defined by the overhead camera. Each trial was run for 30 s. Figure 3.15 shows the paths of the load and transporting robots during a single experiment, and Figure 3.16 plots the average and standard deviation of the load's velocity, heading, and trajectory over the five experiments. These plots show that the sliding mode controllers are fairly successful at achieving the control objectives. The slight rotation of the load and its deviation from the desired path in Figure 3.15, as well as the increasing standard deviations in the plots in Figure 3.16, are due to unavoidable drift in the onboard odometry caused by wheel slip, sensor noise, and model error, among other factors. Sensor noise can result in discrepancies in the robots' velocities, causing the robots to exert torques on each other through the load, which produces wheel slip and error in the odometry.

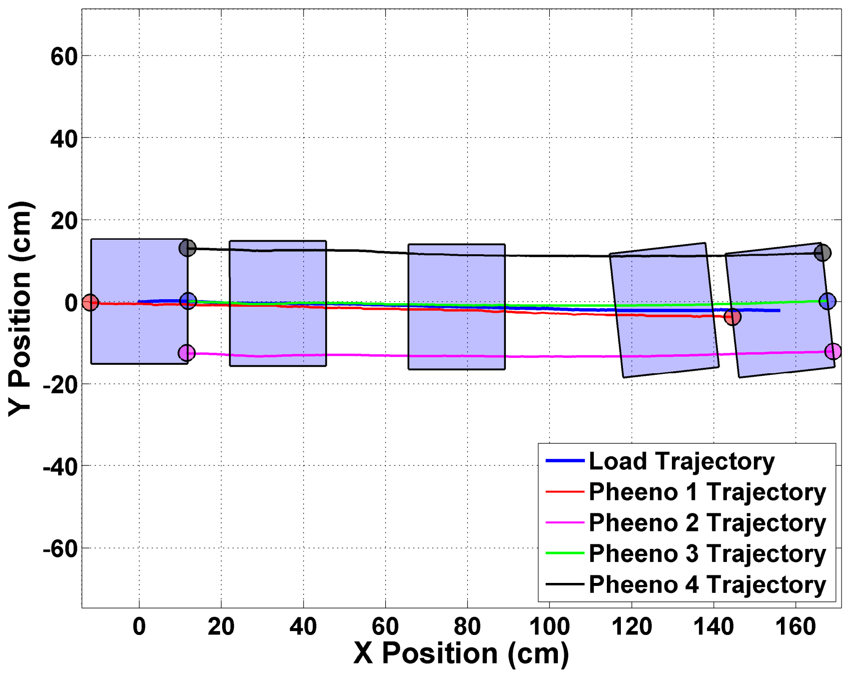


Figure 3.15. The trajectory of the Pheenos and the transported load during one experiment. The rectangle shows the orientation of the load at several time points. The colored circles mark the robot attachment points at the beginning and end of the experiment.

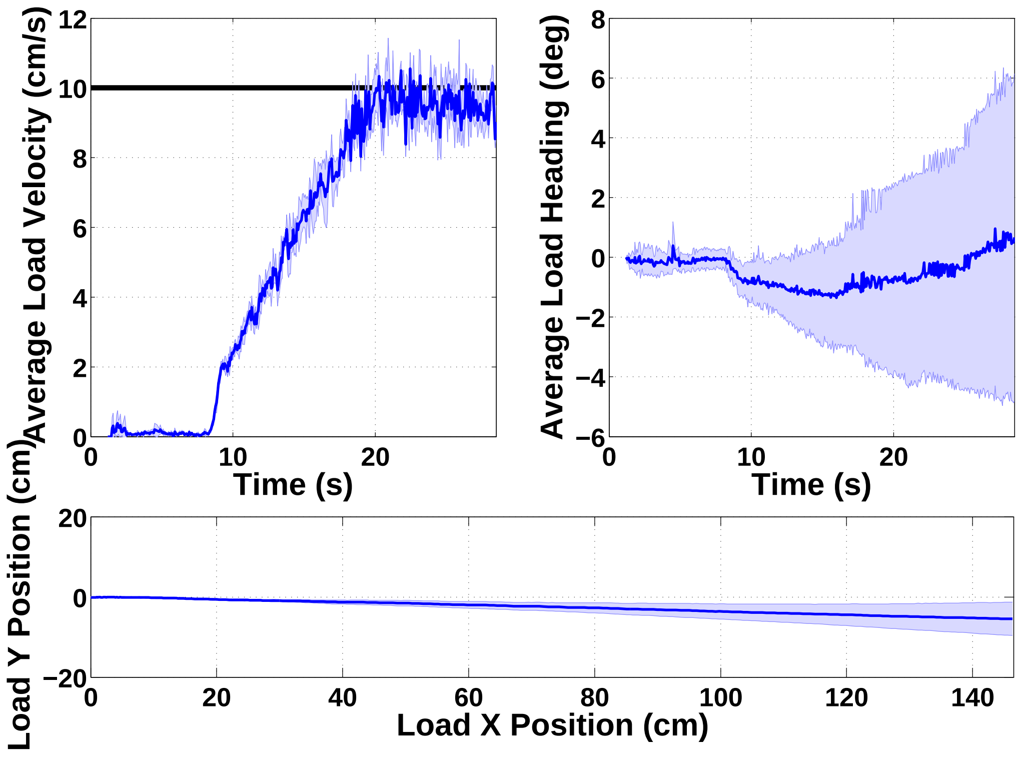
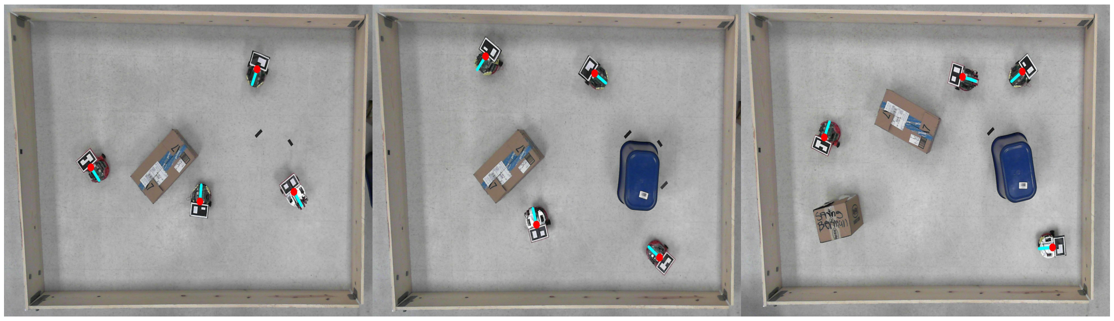


Figure 3.16. The states of the load during the transport experiments. The dark blue lines are the mean states averaged over five trials, and the light blue area shows the standard deviation. *Top Left*: The average velocity of the load during transport. *Top Right*: The average heading of the load during transport. *Bottom*: The average trajectory of the load during transport.

3.3.3 A Probabilistic Approach to Feature Identification and Automated Construction of Topological Maps using a Stochastic Robotic Swarm (Ramachandran et al. [114, 115])

This work quantifies the number of topological features of an unknown environment using a swarm of robots with local sensing and limited or no access to global position information. The robots randomly explore the environment and record a time series of their estimated position and the covariance matrix associated with this estimate using an extended Kalman filter (EKF). Tools from topological data analysis, in particular the concept of persistent homology, are applied to a subset of the point cloud to construct barcode diagrams, which are used to determine the numbers of different types of features in the domain. This procedure was used to construct topological maps to identify collision-free trajectories for robots to navigate through the environment.

To experimentally validate this procedure beyond simulation, four Pheeno robots were deployed in an environment with one to three features. These experiments tested the robustness of the procedure to hard-to-simulate noise and disturbances in the system that are contributed by wheel slip, sensor noise, and controller action on non-ideal feedback. The robots were initially placed at random locations in a 1.5×2.1 meter rectangular arena that was bounded by wooden walls, as shown in Figure 3.17. The robots were controlled to move at 10 cm/sec with an avoidance radius of 10 cm. Whenever a robot detected a feature, wall, or another robot, it avoided a collision by moving according to a specular reflection from the detected object and then continued in a straight line. The robots were marked with 2D binary identification tags to enable real-time tracking of their positions and orientations by an overhead camera (Microsoft Life Cam, resolution of 1920×1080 pixels). A control



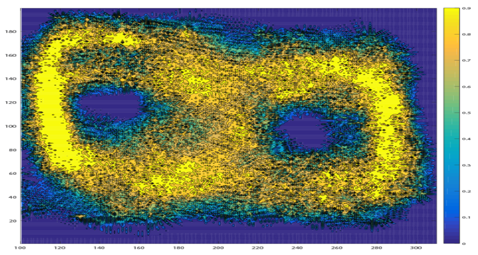
(a) **(b)** **(c)**

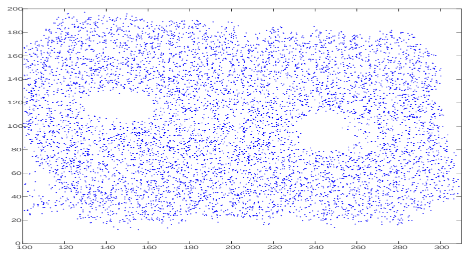
Figure 3.17. The experimental arena with four Pheeno robots and (a) one feature, (b) two features, or (c) three features. At the start of the experiment, the control computer identifies the robots' positions and orientations, indicated by the red dots and cyan lines, from the robots' 2D binary identification tags. This identification is done using the thresholding, boxpoint, and contouring OpenCV libraries on a Windows computer.

computer broadcast each robot's initial state $\mathbf{x} = [x, y, \phi]^T$ over WiFi, where x and y are the robot's position coordinates in the arena and ϕ is its heading. Each robot used an Extended Kalman Filter (EKF) to estimate its state at intervals of 200 ms. This state was updated according to a kinematic unicycle model and a measurement state vector, $\mathbf{z} = [\Delta d_e,\ \Delta\phi_e,\ \phi_c]^T$, where Δd_e is the encoders' measurement of the linear distance traveled, $\Delta\phi_e$ is the change in heading angle measured by the encoders, and ϕ_c is the orientation of the robot in the global frame measured by the compass. The state error covariance matrix $\mathbf{P}$, process covariance matrix $\mathbf{Q}$, and measurement covariance matrix $\mathbf{R}$ were initially set to $\mathbf{P} = diag(0.2, 0.2, 0.1)$, $\mathbf{Q} = diag(2, 2, 4)$, and $\mathbf{R} = diag(0.1, 5, 0.4)$. These matrices were chosen to favor the robot's measurements over the kinematic motion model. The initial state estimate covariance was chosen to reflect errors in tag placement on the robots and camera discretization error. The EKF was implemented on Pheeno's Arduino Pro Mini microcontroller (3.3V 8MHz), while the state data and covariance matrices were stored onboard its Raspberry Pi 2

Model B.

Figure 3.18 shows several representations of the robots' estimated locations during an experiment with two features. These plots clearly show two distinct regions where obstacles must lie, since the robots could not drive within those regions. The plots in Figure 3.19 and Figure 3.20 show that given a sufficiently long deployment time and a sufficiently large number of robots, the approach produces accurate counts of the numbers of connected components and features in environments with one, two, and three features. There is a trade-off between the robots' deployment time and the reliability of their position data, since the EKF state estimates will drift due to the robots' wheel slip and sensor noise. These factors cause the covariances of the position estimates to eventually grow larger than the environment and thus yield no useful information for mapping. This uncertainty can be reduced by correcting the drift with direct GPS measurements or with estimates of global position using local measurements of known objects in the environment. From these experiments, it is evident that larger numbers of robots yield more accurate mapping results, since there is a higher chance of robots exploring small gaps between features before the covariances of their position data grow too large to provide useful information. Finally, Figure 3.21 shows that the procedure is effective at building the topological map of the experimental arena.


(a) Contour plot of the probability that each grid cell is free, over all grid cells of the discretized domain; colorbar values range from 0 to 0.9


(b) Point cloud

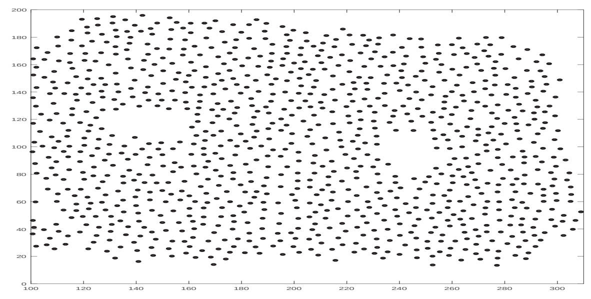

(c) Landmark points

Figure 3.18. Experimental results from an environment containing two features.

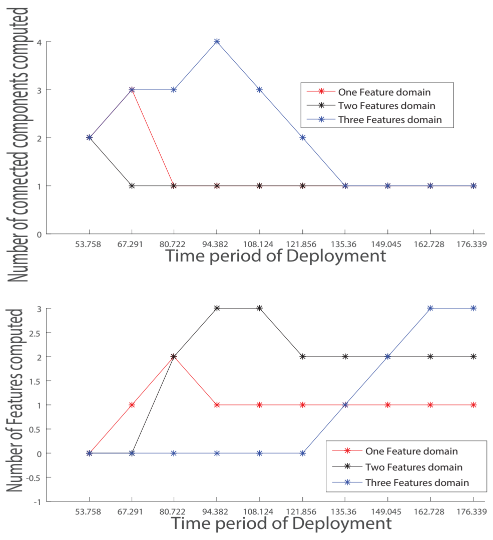


Figure 3.19. Computed numbers of connected components (*top*) and features (*bottom*) versus swarm deployment time period T (in seconds) for experiments with four robots on the domains shown in Figure 3.17.

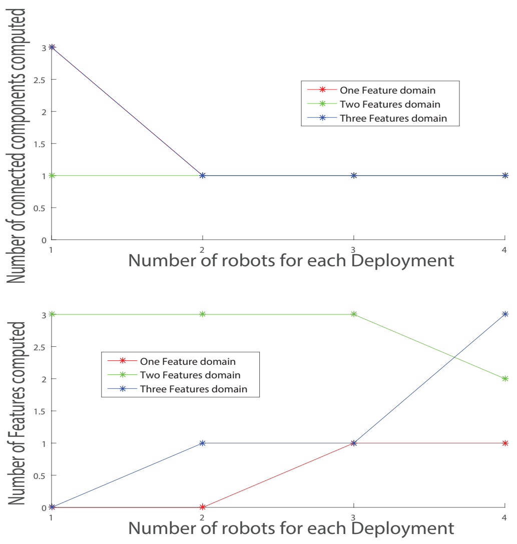


Figure 3.20. Computed numbers of connected components (*top*) and features (*bottom*) versus number of deployed robots N for experiments with $T = 180$ sec on the domains shown in Figure 3.17.

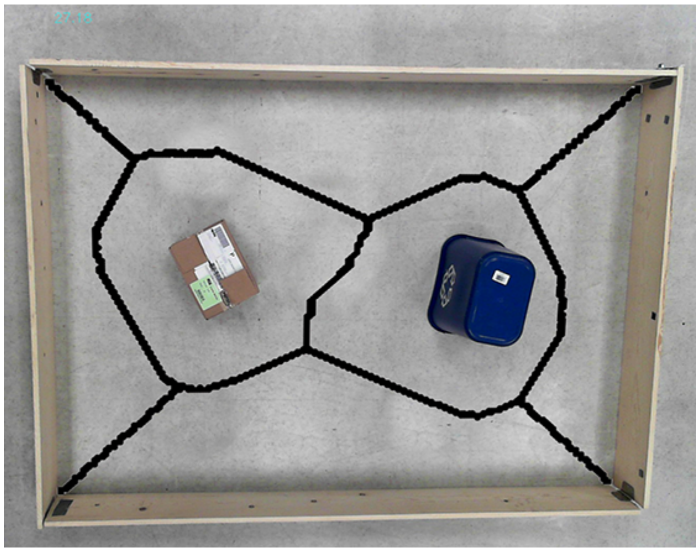


Figure 3.21. Topological map overlaid on the experimental arena with two obstacles.

3.4 Updates to the Pheeno Design

To take advantage of advances in microelectromechanical systems, microcontroller, and microprocessor technology and improve Pheeno's design, updates were made to the Pheeno core module to increase its capabilities while maintaining the same price point. The original and updated versions of Pheeno will be referred to as *Pheeno 1.0* and *Pheeno 2.0*, respectively. Although Pheeno 2.0 operates on a different main processor, the same code will work on both versions of the robot (with some GPIO pin numbers changed), i.e., they are forward and backward compatible.

Figure 3.22 shows the Pheeno 2.0 core module with an exploded rendering. By changing the locations of some of the larger electronic components and optimizing the wire routing, the height of the cylindrical core was shortened to 6.9 cm while maintaining the 12.7 cm diameter. This reduction in height moved the center of mass of the platform lower to the ground, making it more stable. The robot is still mostly composed of 3D-printed parts, which have previously been fabricated from either Acrylonitrile butadiene styrene (ABS) or Polylactic acid (PLA) plastics. The top and bottom caps of the core module are now custom-made from laser-cut acrylic sheets to allow easier access to the increased number of GPIO pins. It is still possible to use the standard robotic chassis parts sold by Pololu Robotics and Electronics, but some of these pins would become much harder to access on the new design.

Some other improvements on the original design were also incorporated into this update. An additional switch was included with an indicator LED to allow the robot to operate without powering the Raspberry Pi, and to allow the user to see whether the Raspberry Pi is powered on without having to look inside the robot's core module. On many occasions, especially in outreach event settings, the robot has been operated without requiring the use of the Raspberry Pi. Thus, this switch was

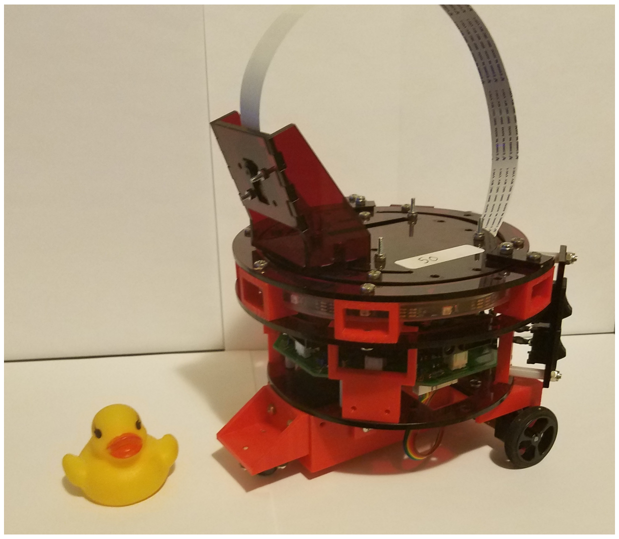
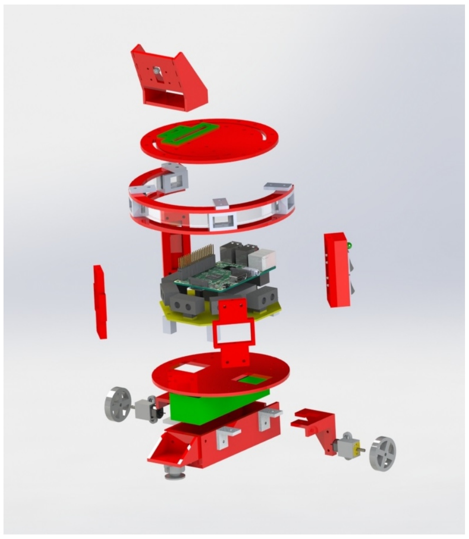



Figure 3.22. *Left:* The Pheeno 2.0 robot core module with the ICRA 2016 *duckie* [3] for scale. *Right:* Exploded SolidWorks rendering of the Pheeno 2.0 core module.

added to save power and to avoid damaging the Raspberry Pi when turning off the robot without properly powering down the Pi. The four LEDs were replaced by a strip of 8 NeoPixels [32], which use one digital GPIO pin to uniquely address each LED along the strip. In contrast to the original design, in which all four LEDs could only display the same commanded color, each of the 8 NeoPixels can be commanded different colors simultaneously. This capability can be used for an expanded range of user feedback and visual signals between robots.

The main circuit board of the robot was also updated to increase the capabilities of the robot. Figure 3.23 shows a top and bottom view of the main circuit board with the major components labeled. The major changes to the circuit board are the replacement of the Arduino Pro Mini with a Teensy 3.2 [8] microcontroller, an upgrade of the Rasperry Pi 2 to the Raspberry Pi 3 microprocessor, the addition of a second motor control board (H-bridge), and the addition of a new IMU board that includes an accelerometer, magnetometer, and gyroscope. A summary of the

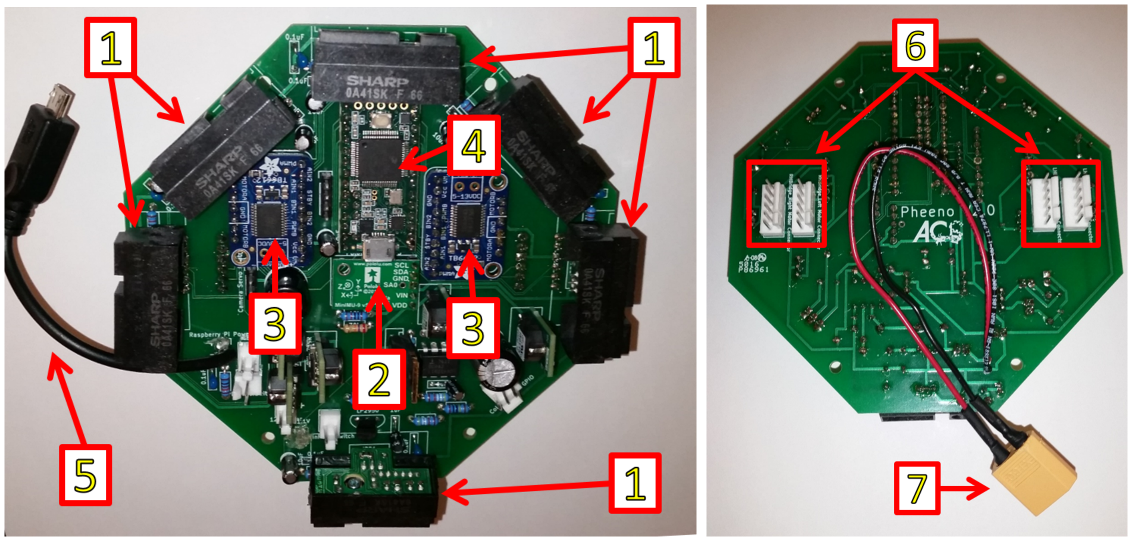


Figure 3.23. Top-down view of the *(left)* top and *(right)* bottom of the main Pheeno 2.0 circuit board with the major components numbered. **1)** Infrared (IR) sensor mounts able to interface with any 3-pin JST connector. **2)** 3D accelerometer, magnetometer, and gyroscope. **3)** Motor control board (H-Bridge). **4)** Teensy 3.2 microprocessor. **5)** Power cable for the Raspberry Pi 3. **6)** Motor and encoder connections. **7)** Main power cable.

differences can be found in Table 3.2.

The updated microcontroller and microprocessor enable a reduction in the sampling time of control schemes that are run on the robot, since the robot's sensors can be sampled at a higher frequency and more complex arithmetic can be performed at a faster rate. The larger storage capacity of 256K on the Teensy also allows more complex code like Kalman filters to be run onboard the robot without running out of memory. There are also 19 more available analog/digital GPIO pins on the Teesny in Pheeno 2.0 than on the Arduino-Pro Mini in Pheeno 1.0. Four of these pins are unused by standard sensors and actuators onboard the robot and allow users to connect additional electronic components to the core module, facilitating expanded customizability of the platform. The rest of the available pins are used for the ad-

	Microcontroller	Microprocessor	Sensing	Cost
Pheeno 1.0	Arduino Pro Mini (3.3V, 8MHz)	Raspberry Pi 2	3D accelerometer, 3D magnetometer, wheel encoders, IR, camera	$270
Pheeno 2.0	Teensy 3.2	Raspberry Pi 3	3D accelerometer, 3D magnetometer, wheel encoders, IR, camera, 3D gyroscope	$260
Difference	**Clock Speed:** 8MHz to 96MHz **Flash Memory:** 32K to 256K **RAM:** 2K to 64K **GPIO Pins:** 20 to 39	**CPU:** 900GHz to 1.2GHz **Architecture:** 32-bit to 64-bit **Communication:** Built-in Bluetooth and WiFi (No need for USB adaptor)	3D gyroscope	Comparable

Table 3.2. Differences between the original and updated Pheeno core design.

ditional motor control board. With two motor control boards, up to four motors with encoder feedback can be controlled at the same time by the robot. This design allows the integration of holonomic drivetrains, like the one pictured on the left of Figure 3.24. The use of a holonomic drivetrain enables testing of swarm control schemes without the added complexity of including nonholonomic motion constraints or approximating holonomic motion, which is required when the robot has a differential drivetrain such as the wheeled drivetrain or the tank tread drivetrain shown in Figure 3.24. All three drivetrains in Figure 3.24 were developed in the Autonomous Collective Systems Lab.

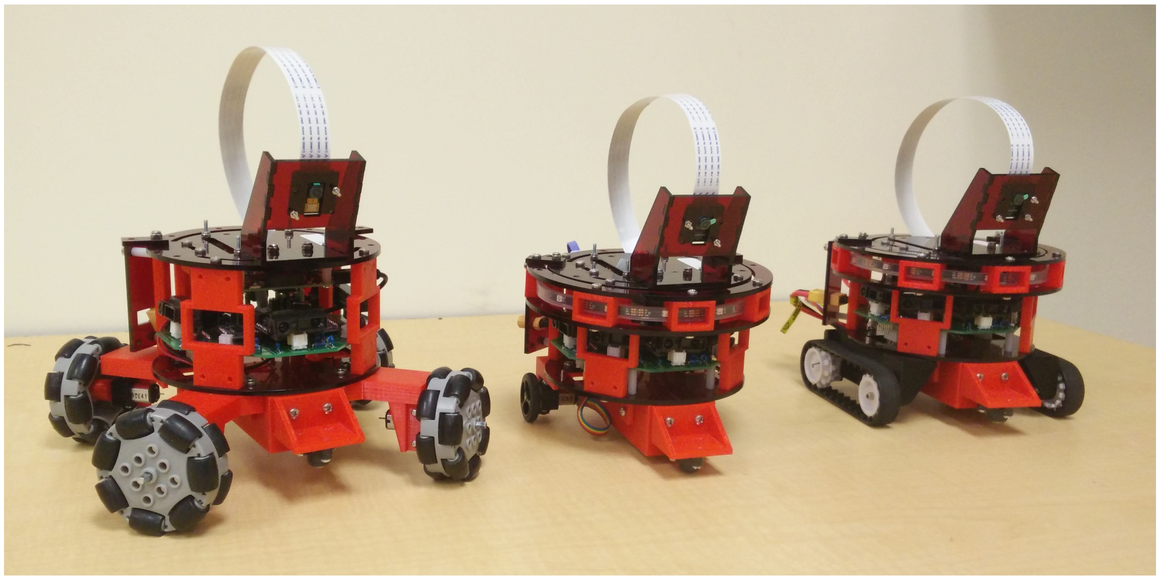


Figure 3.24. Three Pheeno 2.0 robot core modules with different drivetrains: (left) holonomic drivetrain with omnidirectional wheels, (center) differential drivetrain, (right) tank treads.

Chapter 4

DESIGN OF STOCHASTIC CONTROL STRATEGIES FOR BOUNDARY COVERAGE

Source: Pavlic et al. [109]

ABSTRACT

This work presents a novel control approach for allocating a robotic swarm among boundaries. It represents the first step toward developing a methodology for encounter-based swarm allocation that incorporates rigorously characterized spatial effects in the system without requiring analytical expressions for encounter rates. The proposed approach utilizes a macroscopic model of the swarm population dynamics to design stochastic robot control policies that result in target allocations of robots to the boundaries of regions of different types. The control policies use only local information and have provable guarantees on the collective swarm behavior. The relationship between the stochastic control policies and target allocations are analytically derived for a scenario in which circular robots avoid collisions with each other, bind to boundaries of disk-shaped regions, and command bound robots to unbind. This relationship is validated in simulation and is shown to be robust to environmental changes, such as a change in the number or size of robots and disks.

This chapter presents a control framework with the properties in Section 1.1 for allocating a robotic swarm in target group sizes around the boundaries of disjoint, stationary disks of different types. This problem is illustrated in Figure 1.1. The robots have no prior information about the disks and only use local sensing and local communication, encountering the disks while performing random walks. A top-down approach is employed to design stochastic robot control policies that produce target allocations among the disks at steady-state, with probabilistic guarantees on performance. These control policies include an enzyme-inspired behavior, implemented at the disk boundaries, that greatly reduces the dependence of the strategy on the robots' encounter rates with occupied and unoccupied sections of the disk boundaries. The stochastic robot interactions with boundaries and with other robots are represented as a well-mixed chemical reaction network (CRN). The resulting collective behavior follows the prediction of a corresponding *macroscopic* population model, a set of ordinary differential equations (ODEs), in expectation.

Agent-based NetLogo [139] simulations of a *microscopic* model of the strategy confirm that the simulated system retains all of the qualitative features of the predictions of the macroscopic model and is robust to environmental parameter variations. That is, a single control strategy can be implemented based on the geometric properties of a single robot and a single disk, and the equilibrium occupancy levels of robots around disks will be invariant to changes in the total numbers of robots, disks, and disk types, as well as robust to changes in robot speed (e.g., due to battery decay). Hence, the control strategy need not be re-tuned if the robots' environment changes over time.

4.1 Robot Controller

It is assumed that each robot has a small sensing and communication radius. Even when communication is possible, it may be difficult for a robot to identify the location of the source of a message. Consequently, a control strategy is proposed that achieves a desired average allocation around each type of disk using only local robot-robot and robot-disk interactions. The controller for each robot, shown in Figure 4.1, incorporates:

Robot movement: Robots move according to a *correlated random walk* (CRW) in order to achieve approximately uniform distributions throughout empty space. Each robot moves straight ahead in a short segment and then turns to a random angle before repeating. If this assumption of spatial homogeneity is violated, then different regions of space may approach equilibrium at faster rates than others. However, the limiting average allocations will be robust to inhomogeneity of robot density.

Robot-disk interactions: Each robot can identify the type of disk that it encounters. The robot then chooses to bind to that disk with probability p_b that depends on disk type; otherwise, the robot ignores the encounter and continues its CRW.

Robot-robot interactions: Upon encountering another robot, a robot can identify whether it is an unbound robot or a robot that is bound to a disk. If it encounters an unbound robot, the robot executes a collision avoidance maneuver. If it encounters a bound robot, the robot chooses to command that robot to unbind based on a probability p_u that depends on the disk type; otherwise, the robot ignores the encounter and continues its CRW.

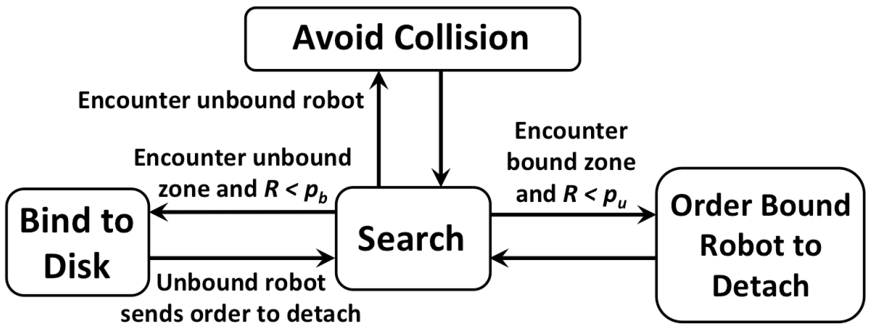

Avoid Collision
Encounter unbound robot
Encounter unbound zone and $R < p_b$
Encounter bound zone and $R < p_u$
Bind to Disk
Search
Order Bound Robot to Detach
Unbound robot sends order to detach

Figure 4.1. Diagram of control flow. Robots randomly cycle through searching and encountering robots and unbound zones. On encountering those disk regions, they probabilistically choose to bind to unbound regions or tell bound robots to unbind. On encountering unbound robots, they avoid collisions. Probabilities can be implemented with a pseudo-random number $R \in \text{unif}(0, 1)$.

A behavior in which robots unbind spontaneously at a certain probability rate is not specified here. Robots either probabilistically choose to bind to encountered disks or stochastically detach from free robot encounters. If a bound robot is never encountered by a free robot, it will never unbind from the disk.

4.2 Microscopic Model: Enzymatic Chemical Reaction Network

The robot-disk system described previously resembles a gas made up of species of *free robots* and disk zones that are either *bound* or *unbound*. For simplicity, only one disk type is considered here; however, it will be shown how these results naturally extend to an arbitrary number of disk types. The corresponding well-mixed chemical reaction network (CRN) for the single-type case is:

$$r + U \xrightarrow{p_b e_u} B \tag{4.1a}$$

$$r + B \xrightarrow{p_u e_b} U + 2r \tag{4.1b}$$

where r represents the free-robot species, B represents bound zones, and U represents unbound zones. The mass-action rate constants $p_b e_u$ and $p_u e_b$ include:

e_u: The probability per unit time that a single free robot will encounter a single unbound zone. This encounter rate is an environmental parameter.

e_b: The probability per unit time that a single free robot will encounter a single bound zone. This encounter rate is an environmental parameter.

p_b: The probability that a free robot will bind to an unbound zone given that it has just encountered it. This parameter is under the control of the designer.

p_u: The probability that a free robot will command a bound robot to unbind given that the free robot has just encountered the bound zone. This parameter is under the control of the designer.

Reverse reactions are necessary to stabilize unique non-trivial equilibria. Without a reverse reaction, the reaction in Equation 4.1a would cause the system to reach trivial saturation of bound zones. In other examples in stochastic robotics [74, 101, 16, 18, 94, 15], event-driven forward reactions like Equation 4.1a are accompanied with delay-driven reverse reactions – robots have a tendency to *decay* back into earlier behavioral modes. Instead of implementing the reverse reactions as decay processes, the reverse direction is implemented with the event-driven enzymatic reaction in Equation 4.1b. The free robot that encounters the bound zone in Equation 4.1b is not consumed by the reaction; it is analogous to an enzyme which rapidly increases the decay rate of bound zones. Since both reactions are event driven, the expected equilibrium distributions will vary with the ratio e_b/e_u as opposed to the absolute encounter rates. In general, the absolute encounter rates e_b and e_u will change with robot density, total number of zones, and robot speed (which itself can change over time with battery fatigue). However, the ratio e_b/e_u, and thus the equilibrium distribution, will be invariant to these changes.

4.3 Macroscopic Model: Concentration Fields of Zones and Robots

From the theory of mass-action kinetics in well-mixed gases, a smooth concentration--field approximation of the CRN in Equation 4.1 for large populations is the multi-affine system

$$\begin{cases} \dot{r} = p_u e_b r B - p_b e_u r U \\ \dot{U} = p_u e_b r B - p_b e_u r U \\ \dot{B} = p_b e_u r U - p_u e_b r B \end{cases} \tag{4.2}$$

where r, U, and B represent the number of free robots, unbound zones, and bound zones, respectively, for a given arena (i.e., concentrations in a fixed volume size). The system clearly has a continuum of trivial equilibria characterized by $r = 0$, which represents the total depletion of free robots. Moreover, because this system is continuous, the set $\{r : r > 0\}$ is positively invariant; if the initial concentration of free robots is positive, then the concentration will remain positive indefinitely. Thus, assuming non-zero mass-action rate constants, there is an additional equilibrium $(r, U, B) = (r^*, U^*, B^*)$ where $r^* > 0$ and

$$\frac{B^*}{U^*} = \frac{p_b e_u}{p_u e_b} \quad \text{or,} \quad \frac{B^*}{B^* + U^*} = \frac{p_b e_u}{p_b e_u + p_u e_b} = \frac{\frac{p_b}{p_u}}{\frac{p_b}{p_u} + \frac{e_b}{e_u}}. \tag{4.3}$$

Let B_0, U_0, and r_0 represent the initial number of bound zones, unbound zones, and free robots, respectively. Noting that $\dot{U} = \dot{r}$ and $\dot{B} = -\dot{r}$, it must be the case that $B^* + U^* = B_0 + U_0$. Additionally, the system of three differential equations in

Equation 4.2 can be re-written as a single differential equation

$$\begin{aligned}\dot{r} &= p_u e_b r(\overbrace{B_0 - (r - r_0)}^{B}) - p_b e_u r(\overbrace{U_0 + (r - r_0)}^{U}) \\ &= ((B_0 + r_0)p_u e_b - (U_0 - r_0)p_b e_u)r - (p_b e_u + p_u e_b)r^2\end{aligned} \tag{4.4}$$

representing the dynamics of the number of free robots. Under the assumption of non-zero mass-action rate constants, the non-trivial equilibrium at $r = r^* > 0$ is such that

$$r^* = (B_0 + r_0)\frac{p_u e_b}{p_b e_u + p_u e_b} - (U_0 - r_0)\frac{p_b e_u}{p_b e_u + p_u e_b}.$$

So, by Equation 4.3 and because $B^* + U^* = B_0 + U_0$,

$$r^* = (B_0 + r_0)\left(1 - \frac{B^*}{B_0 + U_0}\right) - (U_0 - r_0)\frac{B^*}{B_0 + U_0}$$

which is positive and asymptotically stable so long as $B_0 + r_0 > B^*$. That is, so long as the total number of free and bound robots $r_0 + B_0$ is larger than the predicted equilibrium number of bound robots B^*, the (r^*, U^*, B^*) equilibrium will be asymptotically stable with $r^* > 0$. In other words, from Equation 4.3 and the condition that $B_0 + r_0 > B^*$, the system in Equation 4.1 has an asymptotically stable equilibrium described by

$$(r, B, U) = \begin{cases} (r^*, B^*, U^*) & \text{if } r_0 + B_0 > B^*, \\ (0, U_0 - r_0, B_0 + r_0) & \text{otherwise.} \end{cases} \tag{4.5}$$

So the system is driven by the imbalance between fluxes to and from bound and unbound zones; it comes to rest when enough free robots are converted into bound zones to restore flux balance or when the pool of free robots is totally depleted.

4.3.1 High-Population Linear Approximation

Due to its quadratic structure, Equation 4.4 can be solved explicitly. For any $t > 0$,

$$r(t) = r^* \frac{r_0}{r_0 + (r^* - r_0)\exp(-r^*(p_b e_u + p_u e_b)t)},$$

which, for $r_0 \gg 0$, is essentially constant. That is, $r(t) \approx r^* \approx r_0$ for $r_0 \gg 0$. Consequently, for $r_0 \gg 0$, the multi-affine system in Equation 4.2 that approximates the bimolecular CRN in Equation 4.1 can be viewed as the linear system

$$\begin{cases} \dot{U} = p_u e_b r_0 B - p_b e_u r_0 U \\ \dot{B} = p_b e_u r_0 U - p_u e_b r_0 B \end{cases} \quad \text{that models the } \textit{unimolecular} \quad \begin{matrix} U \xrightarrow{p_b e_u r_0} B \\ B \xrightarrow{p_u e_b r_0} U \end{matrix} \tag{4.6}$$

In this $r_0 \gg 0$ regime, the number of free robots scales the per-zone encounter rates. Moreover, although the linear system in Equation 4.6 has an equilibrium in Equation 4.3 that is independent of r_0, the time constant of the system is

$$\frac{1}{(p_b e_u + p_u e_b)r_0}. \tag{4.7}$$

Thus, increasing r_0 increases the total speed of the system but has no impact on the equilibrium allocation of robots to disks.

4.3.2 Multiple Disk Types: Decoupled Analysis and Control

For any number of zones, there is some sufficiently large initial number of free robots r_0 that satisfies the condition that $B_0 + r_0 > B^*$ and thus guarantees the stability of a non-trivial equilibrium zone concentration described by Equation 4.3, which is invariant to changes in r_0. So if the pool of free robots is sufficiently large, the equilibrium analysis of a system with multiple disk types can be performed independently for each disk type.

For example, if there are n disk types and the number of free robots r_0 is initially greater than the total number of unbound zones across all disk types, then the concentration of bound zones B^i and unbound zones U^i on disk type $i \in \{1, 2, \ldots, n\}$ at equilibrium is such that $B^i/U^i = (e_u^i/e_b^i)(p_b^i/p_u^i)$ where e_u^i, p_b^i, e_b^i, and p_u^i are the encounter rates and reaction probabilities specialized for type i. That is, the equilibrium analysis for any type is decoupled from the analysis of any other type.

Although the multiple types have a coupled effect on convergence rate and transient dynamics in general, the equilibrium allocations can be predicted in isolation. So for the remainder of this chapter, a sufficiently large pool of robots is assumed to meet subjective convergence time constraints for an arbitrary number of disk types. Moreover, only design for a single disk type will be explicitly discussed; it is implied that the process is identical for multiple coexisting types.

4.3.3 Corrections for Spatial Effects on Boundaries

In principle, robots can be organized around a disk to reach 100% allocation ($B/(B + U) = 1$). However, in practice, it is likely that two robots interacting stochastically with a disk will bind with non-zero inter-robot space between them that is nevertheless too small for another robot to encounter. So although the amount of unbound space on a disk may be large, the actual number of unbound zones available for additional binding may be small. Thus, the maximum value of $B/(B+U)$ will be less than unity; even a $(p_b, p_u) = (1, 0)$ policy will saturate with free space remaining on disks. Moreover, even well before saturation, some amount of free space will be inaccessible for free robots to bind to because it will be too close to existing bound robots. Thus, a theory is needed to model the nonlinear reduction in remaining unbound space as robots bind to disks.

In the following, assume that all linear distances are given in units of the arc length occupied by a robot when bound to a disk. That is, each robot binds to 1 unit of arc length, and so the theoretical maximum number of robots bound to a disk is equal to the disk's circumference. However, because robots are not equipped with the ability to cluster together, the actual maximum number of bound robots will be much lower. Consider:

- A disk with a single robot bound to it. An incoming unbound robot will not be able to discover disk space adjacent to a bound robot unless its center is at least 0.5 units away from the edge of the bound robot. So the bound robot effectively occupies both its own 1 unit of arc space plus an additional 1 unit of arc length adjacent to it. Additionally, if the incoming unbound robot maintains a distance a between itself and every other robot (to avoid collisions), then the additional space occupied by the bound robot increases to $\delta_{\max} \triangleq 1 + 2a$ units because there are $0.5 + a$ units of additional occupation on both sides of the bound robot.

- A disk with many robots bound to it. If two robots have less than $1 + 2a$ unbound arc length between them, an incoming unbound robot will not be able to discover it. However, these small distances can be no smaller than the avoidance distance $\delta_{\min} \triangleq a$.

With this in mind, the space between robots is partitioned into sections no longer than $\delta_{\max} \triangleq 1 + 2a$, as shown in Figure 4.2. The quantity δ is defined to be the mean size of the partitioned inter-robot spaces. Thus, although truncation of an inter-robot space that is only slightly larger than $1 + 2a$ can create a truncated space smaller

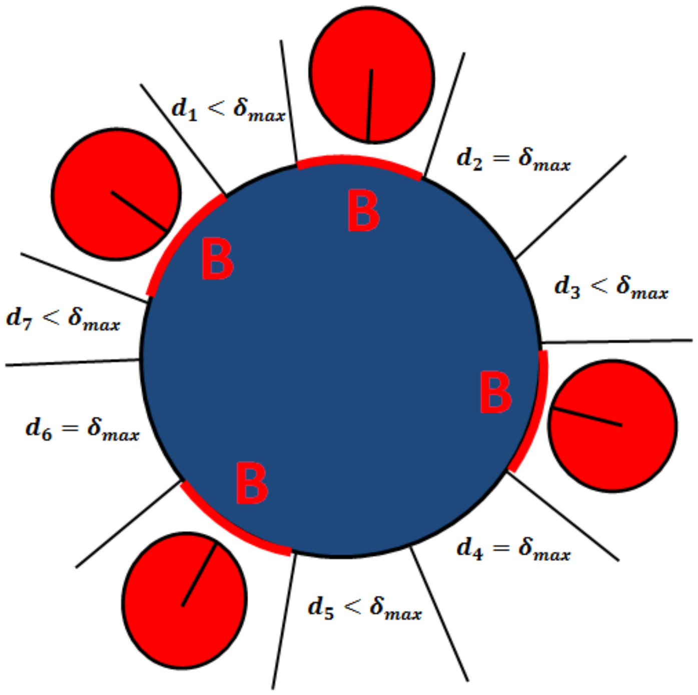


Figure 4.2. Partitioning of space between robots. Here, four robots are shown connected to a single disk. Each robot with its corresponding disk sector, shown with a "B", constitutes a bound zone. The four spaces between each pair of robots have been partitioned into smaller spaces no larger than $\delta_{\max} = 1 + 2a$, which represents the maximum additional arc length that a bound robot can interfere with due to spatial effects.

than a, the mean δ is bounded above and below such that

$$\delta_{\min} = a \ \leq \ \delta \ \leq \ 1 + 2a = \delta_{\max}.$$

The statistic δ is actually a function $\delta : [0, 1] \to [a, 1 + 2a]$ that maps an allocation ratio $B/(U+B)$ to the mean additional arc occupancy per bound zone $\delta(B/(U+B))$, which is abbreviated to δ here for convenience. At low allocation ratios, $\delta \approx 1 + 2a$ because the space between bound robots is large. Consequently, there will be more encounters with bound zones and fewer encounters with unbound zones than otherwise expected. Similarly, at high allocation ratios, $\delta \approx a$. So although bound zones are still magnified, this magnification decreases with added allocation ratio.

To model the effective increase in B by δB and the corresponding effective decrease in U by δB, the substitutions $B^* \mapsto (1+\delta)B^*$ and $U^* \mapsto U^* - \delta B^*$ are applied to the equilibrium condition in Equation 4.3. Consequently, the actual (B^*, U^*) equilibrium will be such that

$$\overbrace{\frac{(1+\delta)}{1-\delta\frac{B^*}{U^*}}}^{\text{Correction factor}} \frac{B^*}{U^*} = \frac{e_u p_b}{e_b p_u} \tag{4.8}$$

where the overbraced expression is a correction factor for the spatial effects in a physical robot scenario. For comparison, the corrected allocation can be related to the idealized allocation by

$$\frac{B^*}{U^*+B^*} = \frac{B^*}{(U^*-\delta B^*)+(1+\delta)B^*} = \frac{\frac{B^*}{(1+\delta)B^*}}{\frac{U^*-\delta B^*}{(1+\delta)B^*}+1} = \frac{1}{1+\delta} \overbrace{\frac{\frac{p_b}{p_u}}{\frac{p_b}{p_u}+\frac{e_b}{e_u}}}^{\text{Idealized allocation}} \tag{4.9}$$

where the overbraced expression matches the idealized allocation ratio in Equation 4.3. So the ideal and actual allocations are predicted to be related by a $1/(1+\delta)$ gain. For low allocations, this gain will be $1/(1+2a)$; for high allocations, this gain will be determined by the saturated value of $\delta > \delta_{\min} = a$.

4.4 Shape of δ Function

An important future direction is to develop theory to predict the precise shape of the δ function. However, as discussed later, simulations suggest that δ is only determined by the value of a. Moreover, δ appears to be a cosine of the form $\delta(r) = A\cos(2\pi(r/T)+c)$ that pierces $\delta(0) \approx 1+2a$ and $\delta(1/(1+a)) = a$ subject to the constraints $A \geq (1+a)/2,\ T \geq 2/(1+a)$, and $c \geq 0$.

4.5 Control of Equilibrium Allocations

From the equilibrium described by Equation 4.8, a (p_b, p_u) control policy can be synthesized using the rule

$$\frac{p_b}{p_u} = \frac{e_b}{e_u}\frac{B^*}{U^*}\frac{(1+\delta)}{1-\delta\frac{B^*}{U^*}} \tag{4.10}$$

where B^*/U^* is the desired bound-unbound allocation ratio of zones at equilibrium. Equivalently, if the desired robot-to-boundary-space *allocation ratio* is $B^*/(B^*+U^*)$, then (p_b, p_u) should chosen according to Equation 4.9. Thus, for any given allocation ratio, there is a continuum of (p_b, p_u) pairs that will achieve the desired equilibrium.

The control policy in Equation 4.10 has one degree of freedom over which some feature of the system can be optimized. For example, by Equation 4.7, the convergence rate of the system can be maximized by making the sum $p_b + p_u$ as large as possible. So, for fastest convergence for to a desired allocation (B^*, U^*), p_b and p_u can be chosen so that

$$(p_b, p_u) = \begin{cases} \left(\frac{e_b}{e_u}\frac{B^*}{U^*}\frac{(1+\delta)}{1-\delta B^*/U^*}, 1\right) & \text{if } e_b B^*(1+\delta) < e_u U^*(1-\delta B^*/U^*), \\ \left(1, \frac{e_u}{e_b}\frac{U^*}{B^*}\frac{1-\delta B^*/U^*}{(1+\delta)}\right) & \text{otherwise.} \end{cases} \tag{4.11}$$

However, optimization criteria other than maximal convergence rate may suggest other choices of (p_b, p_u). For example, there will be fewer temporal variations in the number of robots bound to each disk if $p_b + p_u$ is reduced. Similarly, the variance in allocation across disks may be reduced for certain (p_b, p_u) combinations. Furthermore, if the e_b/e_u ratio can be artificially shifted (by asymmetrically changing the relative distance that sensors react to bound and unbound zones) or the avoidance distance a changed, it is possible to shift the p_b/p_u control policy for a desired B^*/U^* allocation ratio. Thus, there are mechanisms that can further adjust the $p_u + p_b$ sum without changing the equilibrium allocation ratio.

4.6 Model Validation in Simulation

To test the macroscopic model of stochastic allocation to circular boundaries, simulation trials were conducted using NetLogo [139]. The framework allowed for simulating hundreds of mobile robots randomly interacting with each other and with disks of different types.

4.6.1 Variations due to Encounter-rate Ratio

For many robot motion behaviors, the encounter-rate ratio e_b/e_u may be approximated, for example, by dividing the sum of the areas of a robot and an unbound zone sector by the area of an unbound zone sector alone, where zone sectors are slices of each disk with arcs that are the length of the interaction region with the robot. In general, it can be estimated from equilibrium allocation data. If, for example, it is incorrectly assumed that the e_b/e_u ratio is unity, the equilibrium allocation will shift in a predictable way based on the correct e_b/e_u ratio, as shown in Figure 4.3. Consequently, if the e_b/e_u ratio is not well known, it can be estimated by measuring this curve during system testing. Inferring this encounter-rate *ratio* is empirically much simpler than inferring the actual encounter rates. Also shown in Figure 4.3 is the effect of the inter-robot space $\delta(1.0)$ being both non-zero and yet smaller than required to fit any additional robots. Thus, disks saturate at a level less than full occupancy.

To validate these predictions, trials were conducted using 500 simulated mobile robots moving along correlated random walks in a space with 6 disks with circumference capacity for 28.27 robots per disk. By increasing the so-called *turning angle* of the CRW, the robot motion became less directional and more Brownian. Conse-

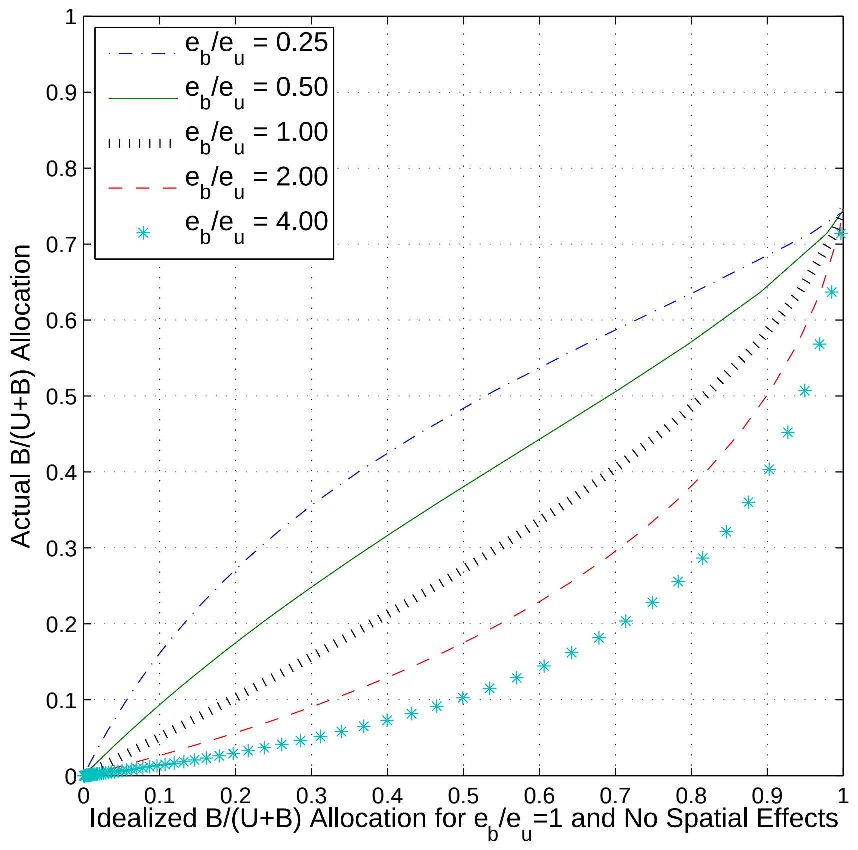
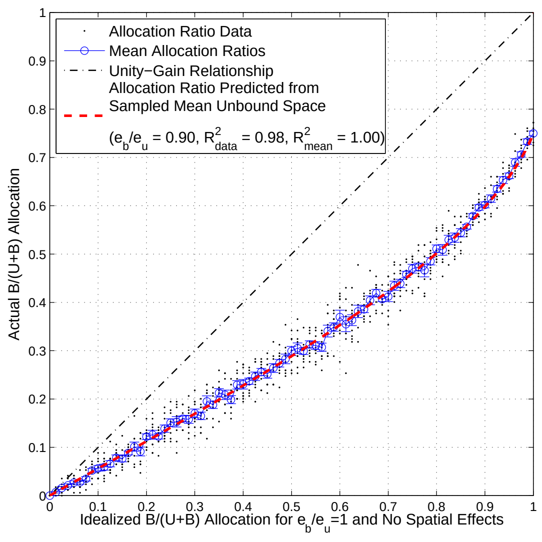


Figure 4.3. Effect of encounter ratio. For each e_b/e_u ratio, a plot comparing the idealized allocation ratio to the actual allocation ratio is shown according to Equation 4.9. Here, $\delta(r) = \cos(2\pi(r/4.6) + 0.2)$, which is consistent with an $a = 0$ case. Allocations saturate near an actual ratio of 0.75 because the mean slack space $\delta(1)$ is both non-zero and too small to accommodate additional binding.

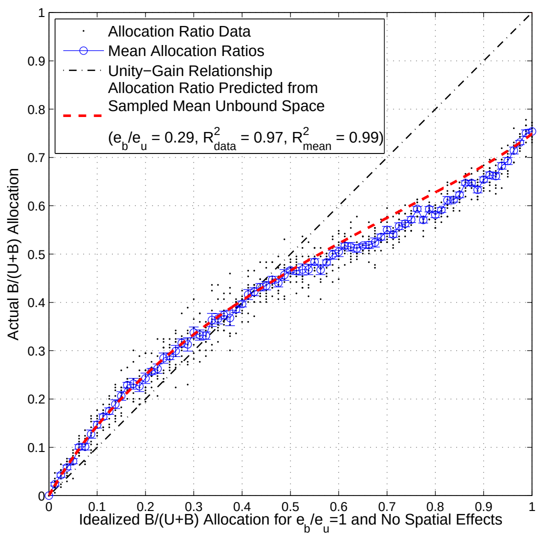
quently, robots with higher turning angle are more likely to re-encounter a disk and re-bind immediately after choosing to unbind. This decrease in unbinding efficacy decreases the effective e_b/e_u ratio, as shown in Figure 4.4 which matches Figure 4.3 for different e_b/e_u ratios.

4.6.2 Estimation of δ Function

The δ function used in Figure 4.3 is based on an avoidance range of $a = 0$ and a circumference of 28.27 robots. As shown in Figure 4.5, this function fits mean data



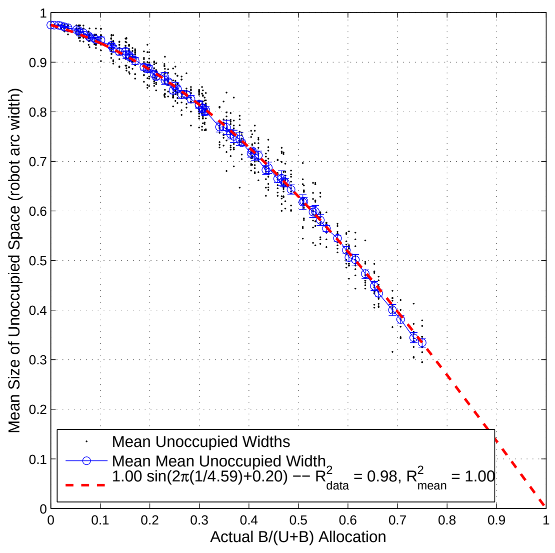
(a) Low CRW turning angle ($e_b/e_u \approx 0.9$)



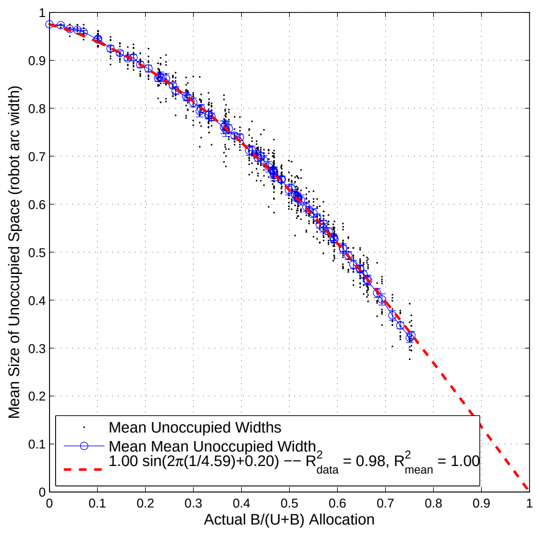
(b) High CRW turning angle ($e_b/e_u \approx 0.29$)

Figure 4.4. Effect of encounter ratio in simulation. For each e_b/e_u ratio, a plot comparing the idealized allocation ratio to the actual allocation ratio is shown according to Equation 4.9. The particular e_b/e_u ratios corresponding to the two motion primitives (low and high turning angle) were fit to the observed data. Additionally, $\delta(r) = \cos(2\pi r/4.6 + 0.2)$, which is consistent with predictions from an $a = 0$ case with a disk circumference of 28.27 robot widths. Allocations saturate near an actual ratio of 0.75 because the mean slack space $\delta(1)$ is both non-zero and too small to accommodate additional binding. The slight deviations from prediction in (b) can be improved with better understanding of the derivation of the δ function. Small dots show outcomes of individual simulation runs. Open circles show means across ten trials of each allocation ratio. Error bars show ± 1 standard error of the mean (SEM). Each trial uses 500 simulated robots and 6 disks.

from simulated scenarios regardless of CRW parameters and effective encounter-rate ratio. The empty-occupancy $\delta(0) < 1 + 2a$ because the circumference does not divide evenly by $1 + 2a$. That is, the partitioned space of the empty disk includes a residual sub-unity partition, and so the mean across those partitions is less than 1.



(a) Low CRW turning angle



(b) High CRW turning angle

Figure 4.5. Simulated effect of encounter ratio on δ. The single δ function from Figure 4.3 accurately predicts mean inter-robot space across different encounter ratios and motion primitives. Each small dot shows a result from an individual simulation run. Open circles show means for different allocations. Error bars show SEM. Ten trials were run per allocation ratio.

4.6.3 Robustness to Environmental Variations

Empirical studies show that the relationship between idealized and actual allocation is not sensitive to environmental variations. For example, Figure 4.6 shows statistics taken from simulation runs with several combinations of robot population, robot size, number of disks, and disk size. As shown, varying the size and number of disks and robots does not change the actual allocation ratio. A single control strategy leads to the same equilibrium allocation ratio in every case.

4.6.4 Robustness to Multiple Disk Types

In Section 4.3.2, it was argued that binding and unbinding probabilities for each disk type can be designed in isolation so long as the pool of robots is sufficiently large.

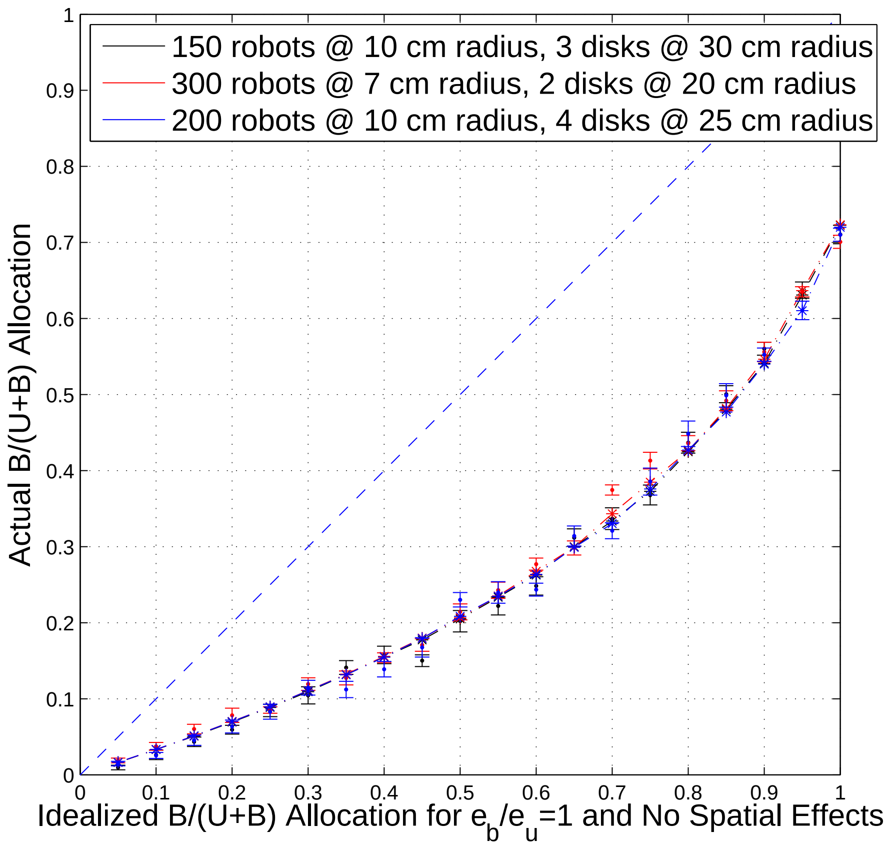
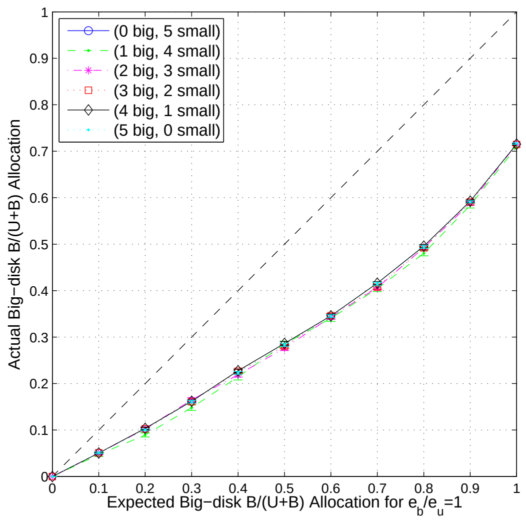
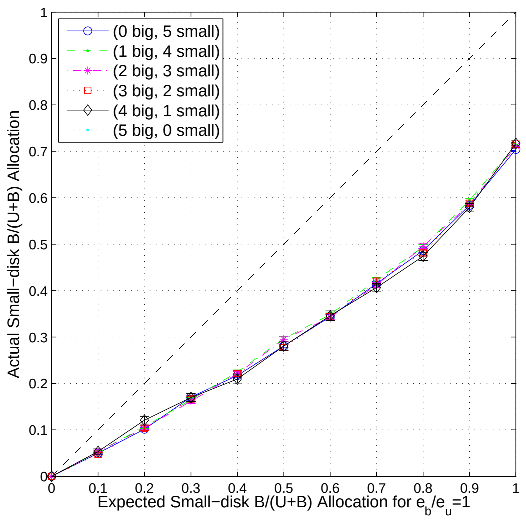


Figure 4.6. Effect of varying environmental parameters. Ten trials were generated for each disk size, and the average across the trials are shown with error bars indicating ± 1 standard error of the mean. A dashed line of unity slope is shown for reference. The solid line represents the predicted curve based on the avoidance distance a, which is non-zero for these cases.

This characteristic is confirmed in Figure 4.7, which shows results from simulations that each generate 500 robots in arenas with varying numbers of disks of different sizes and target allocation ratios. As predicted, the equilibrium allocation for either disk type is independent of the presence of other disk types with different sizes and different target allocation ratios.



(a) Big-disk allocations



(b) Small-disk allocations

Figure 4.7. Effect of varying disk mixture. Ten trials were generated for each experimental treatment (number of big, number of small disks, big allocation ratio, and small allocation ratio); the big-disk type is twice as large as the small-disk type. The averages shown for each big-disk allocation are taken across the pool of 110 trials that include all eleven small-disk allocation ratios for the corresponding big-disk allocation ratio and big–small mixture. Error bars show ± 1 standard error of the mean. Linear interpolation lines are shown for clarity. A dashed line of unity slope is shown for reference.

4.6.5 Effects Due to Interactions Between Free Robots

When robots can obstruct the path of other robots, there is an opportunity for repeated disk encounters because of the inability of recently unbound robots to escape. To investigate this effect, simulations were run of collision-avoiding robots that would ignore encounters with a disk after unbinding from it until either a new disk was encountered or a programmed time delay had elapsed. Figure 4.8 displays plots of expected-vs-actual allocations for different values of time delay. For low delay values, recently unbound robots are likely to re-encounter the same disk after taking evasive

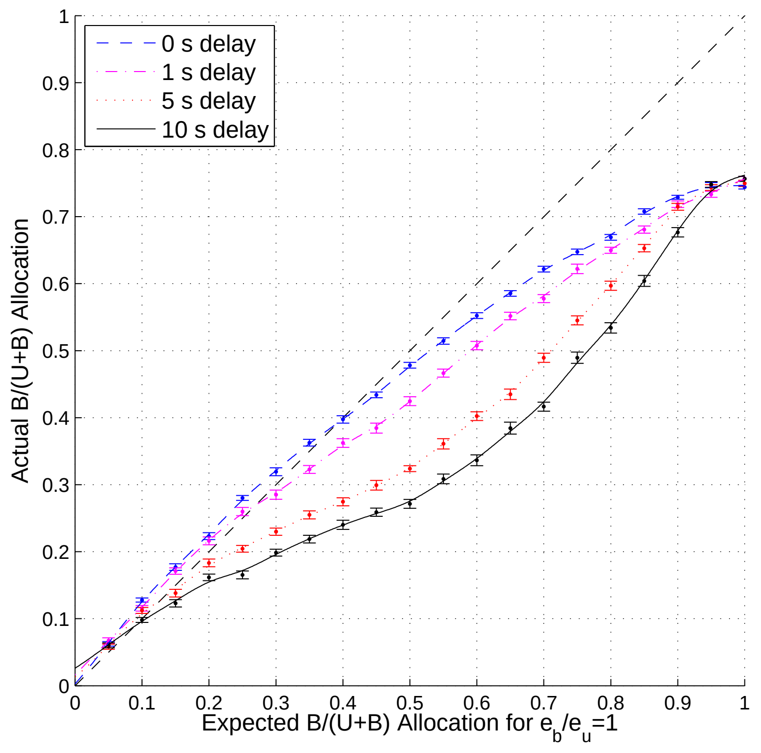


Figure 4.8. Effect of varying the delay before a recently unbound robot is allowed to rebind to the same disk. Each data point is an average over 30 simulation runs at the corresponding set of delay parameters shown in the legend; error bars show ± 1 standard error of the mean. Arbitrary smoothing splines have been added for clarity. A dashed line of unity slope is shown for reference.

maneuvers to avoid collisions with surrounding free robots. Consequently, at low delay values, the encounter rate with unbound space is effectively increased and the expected-vs-actual curve shifts upward as in the low e_b/e_u cases in Figure 4.3. With higher delay values, recently unbound robots are more likely to escape into free space where their encounters with disks will be randomized. Consequently, increased re--binding delay leads to curves that approach the idealized boundary-avoidance-only case in Figure 4.4a.

Free-space collision-avoidance behavior does not reduce the environmental robust-

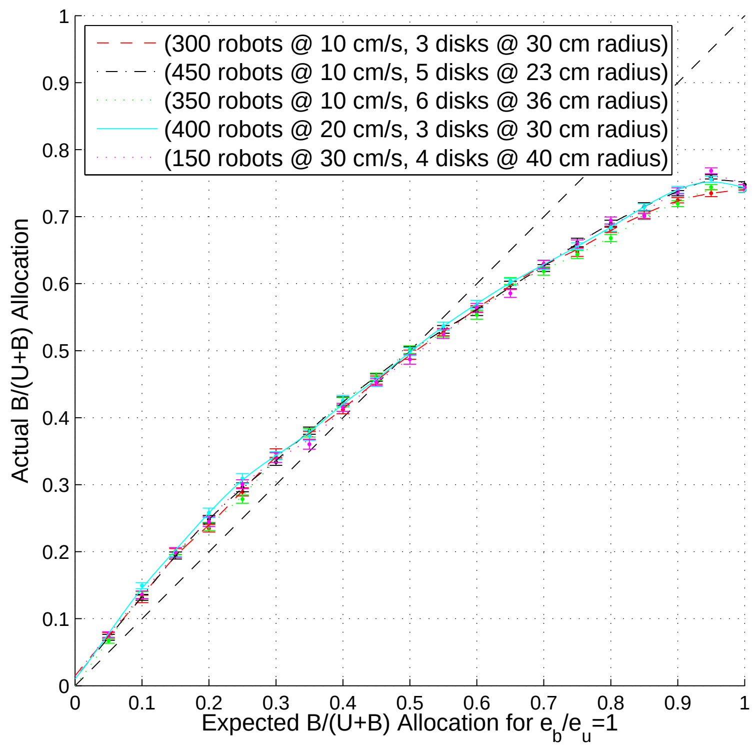


Figure 4.9. Effect of varying environmental parameters. Each data point is an average over 30 simulations runs at the corresponding set of parameter choices shown in the legend; error bars show ±1 standard error of the mean. Arbitrary smoothing splines have been added for clarity. A dashed line of unity slope is shown for reference.

ness that was shown in Section 4.6.3 and 4.6.4 for the idealized boundary-avoidance-only case. Figure 4.9 displays plots of expected-vs-actual allocations for a time delay of 0 s with four random combinations of feasible environmental parameter values, both controllable (robot number and speed) and uncontrollable (number and size of disks). The proximity of the curves show that the relationship between actual and target allocation is invariant to these parameter changes. Hence, they provide a robust guideline for specifying stochastic policies that lead to a desired allocation.

Chapter 5

DESIGN OF STOCHASTIC CONTROL STRATEGIES FOR COLLECTIVE PAYLOAD TRANSPORT

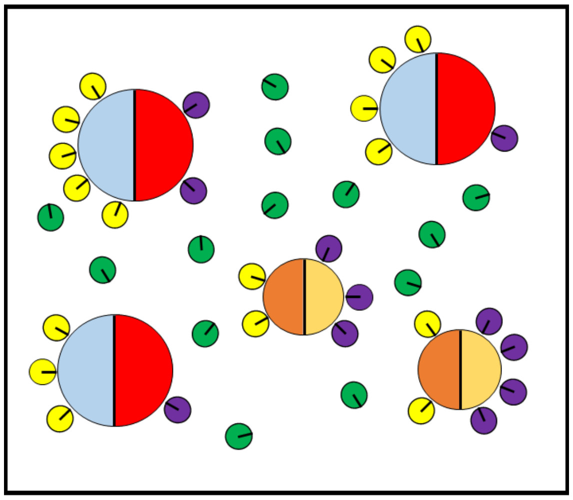
Source: Wilson et al. [141]

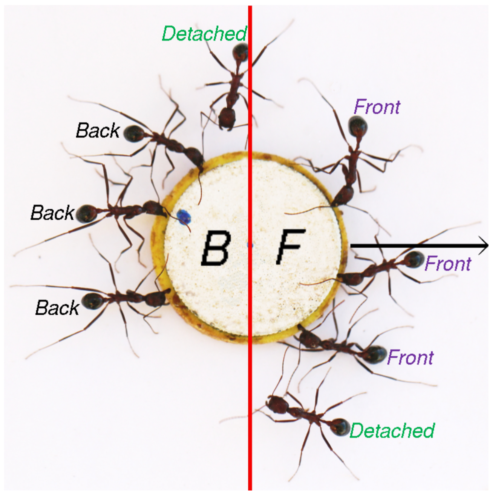
ABSTRACT

This work presents an approach to designing decentralized robot control policies that mimic certain microscopic and macroscopic behaviors of ants performing collective transport tasks. In prior work, a stochastic hybrid system (SHS) model was used to characterize the observed team dynamics of ant group retrieval of a rigid load. Macroscopic population dynamic models have also previously been used to design enzyme-inspired stochastic control policies that allocate a robotic swarm around multiple boundaries in a way that is robust to environmental variations. This approach builds on this prior work to synthesize stochastic robot attachment–detachment policies for tasks in which a robotic swarm must achieve non-uniform spatial distributions around multiple loads and transport them at a constant velocity. Three methods are presented for designing robot control policies that replicate the steady-state distributions, transient dynamics, and fluxes between states that have been observed in ant populations during group retrieval. The equilibrium population matching method can be used to achieve a desired transport team composition as quickly as possible; the transient matching method can control the transient population dynamics of the team while driving it to the desired composition; and the rate matching method regulates the rates at which robots join and leave a load during transport. The model predictions are validated in an agent-based simulation to verify that each controller

design method produces successful transport of a load at a regulated velocity. The advantages and disadvantages of each method are compared.

In this chapter, the boundary coverage algorithm presented in Chapter 4 is extended to a collective transport scenario. It addresses the ant-inspired cooperative transport problem summarized in Figure 5.1. The problem is to design stochastic control policies for individual robots that, when implemented on a sufficiently large swarm, will produce desired allocations of robots around each load and maintain those team sizes while the load is being carried toward its destination. In the work


(a) Multi-robot transport arena.



(b) Top view of ant transport team.

Figure 5.1. Multi-robot and multi-ant collective transport scenarios. In (a), an arena is depicted with five loads surrounded by a swarm of robots, including some that have attached to the loads and some that are moving freely between the loads. The three large loads are of type 1, and the two smaller loads are of type 2. Although the size of a load may not be observable by an individual robot, the load type is assumed to be measurable (by the color of the load or its surface texture). The loads are divided into a *Back* (left) half and a *Front* (right) half. Based on the type of the object, a certain number of robots is desired on each of the two halves. Robots bound to the *Back* (left) halves of the loads are shown in yellow, robots bound to the *Front* (right) halves are shown in purple, and the freely moving *Detached* robots are shown in green. This scenario is based on the observations of ants in (b) from [78]. There, the circular load is moving to the right, and ants are characterized as grasping the *Back* (left) or *Front* (right) of the load; otherwise, they are *Detached* ants.

presented in Chapter 4, robot attachment-detachment strategies were designed for allocating target-sized populations that are uniformly randomly distributed around regions at equilibrium. Here, that approach is extended to achieve non-uniform distributions of robots around loads. In particular, it is desired to match the observations of [78] that more ants accumulate on the leading side of a transported object and appear to interact differently with the load based on their attachment position with respect to the direction of motion. Since this is a stochastic allocation strategy, the equilibrium distributions will be dynamic: there will be continual fluctuations above and below the desired allocation levels, but the mean allocations will converge to these levels. Moreover, as long as there are enough robots available to reach the desired allocation levels on the loads, but not so many that robot crowding impedes individual robot motion and load transport, the equilibrium mean allocation levels will be insensitive to the size of the swarm and the density of robots in the arena.

5.1 Robot Controller Architecture

The robot controller design is illustrated by the state-transition diagram in Figure 5.2. As was discussed previously for the application of boundary coverage, the catalytic detachment process allows allocation policies for multiple load types to be decoupled. Consequently, without loss of generality, only one type of load is considered here, which is define as disk-shaped. Although there is only one type of load, there are two types of subregions: the leading (*Front*) and trailing (*Back*) sides of the load with respect to its transport direction (Figure 5.1b). An *unbound zone* is defined as a load sector with an arc length equal to the linear distance that a robot can occupy along the load perimeter. A *bound zone* is comprised of a robot attached to the load along with the adjacent load sector.

Matching the description of the ant behaviors presented in [78], robots switch between *Front*, *Back*, and *Detached* states. Each robot is initially *Detached* and executes a correlated random walk (CRW) in order for the population to disperse approximately uniformly throughout the arena. In a CRW, robots iterate through short straight paths that are each punctuated by a turn to a random angle. However, other motion patterns that achieve similar dispersal are also valid choices. When a *Detached* robot encounters another robot, it executes maneuvers to avoid a collision. Alternatively, when a *Detached* robot encounters an unbound zone on the *Front* (or *Back*) of a load, it attaches with probability p_{bF} (or p_{bB}). If a robot is attached to the *Front* (or *Back*) of a load and encounters a *Detached* robot nearby, the attached robot will detach with probability p_{uF} (or p_{uB}). In the absence of encounters, an attached robot will never detach from a load. In summary, the four parameters that characterize the attachment-detachment policy of each robot are:

- p_{bF}, the probability to attach (bind) to an encountered unbound zone on the *Front* of a load
- p_{uF}, the probability to detach (unbind) from a zone on the *Front* of a load after encountering a *Detached* robot
- p_{bB}, the probability to attach (bind) to an encountered unbound zone on the *Back* of a load
- p_{uB}, the probability to detach (unbind) from a zone on the *Back* of a load after encountering a *Detached* robot

Once attached to a load, a robot lifts with a small force F_ℓ. A *Front* robot will additionally pull in the desired direction with a time-varying force described by the control law $K(v_L^d - v_L(t))$, where K is a proportional gain, v_L^d is a set point for the load velocity, and $v_L(t)$ is the measured load velocity at time t. Thus, robots have

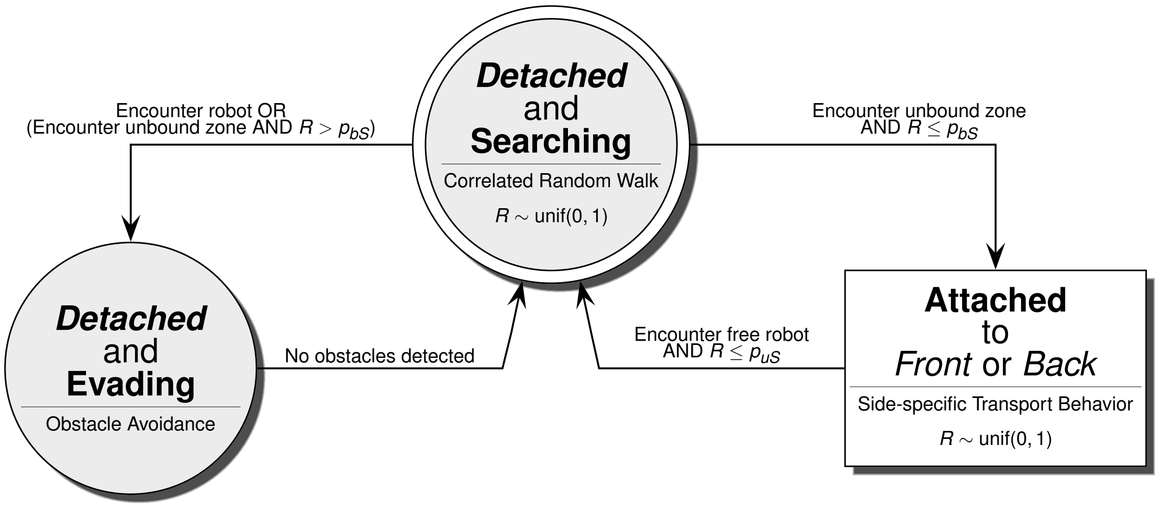

Encounter robot OR
(Encounter unbound zone AND $R > p_{bS}$)
Detached and Searching
Correlated Random Walk
$R \sim \text{unif}(0,1)$
Encounter unbound zone AND $R \leq p_{bS}$
Detached and Evading
Obstacle Avoidance
No obstacles detected
Encounter free robot AND $R \leq p_{uS}$
Attached to Front or Back
Side-specific Transport Behavior
$R \sim \text{unif}(0,1)$

Figure 5.2. State-transition diagram of robot controller for collective transport. The diagram outlines a program that would run on a single robot. The states shown as circles represent *Detached* robots that are unbound and free to move. The state shown as a rectangle represents either *Front* or *Back* robots that are attached to the leading or trailing edge of a load with respect to its motion and executing a side-specific transport behavior (*Front* robots lift and pull, and *Back* robots only lift). The *Detached* robots execute a space-filling random motion primitive, such as a correlated random walk, and avoid obstacles as necessary. The robots employ a simple obstacle-avoidance algorithm: if a robot senses an object that it will not attach to, then it chooses another direction until it does not sense any interference. The subscript $S \in \{F, B\}$ represents the identified side of the load, *Front* or *Back*.

three additional parameters that characterize the transport of the load:

- F_ℓ, the lifting force exerted by a robot that is attached to a load
- v_L^d, the desired velocity of the load
- K, the proportionality constant between the pulling force of a robot attached to the *Front* of a load and the error between the load velocity and v_L^d

5.2 Load Dynamical Model

As done in [78], the load dynamics are formulated using a double-integrator model that describes the relationship between the time-varying force inputs applied by the robots and the load acceleration. The model is fully characterized by two parameters:

- m_L, the mass of the load
- μ, the kinetic coefficient of linear sliding friction

To prevent robots from switching directly between the *Back* to *Front* states due to load reorientation, it is assumed that rotational friction is sufficiently high to prevent load rotation. Even with the possibility of rotation, theoretical models of fixed teams of robots [119] show that rotation is only transient. Moreover, observational evidence of ants and simulated ant models [17, 78] show very little load rotation once smooth persistent load transport has begun. Thus, between attachment and detachment events, the time evolution of the load position x_L and load velocity v_L is modeled as linear translational motion:

$$\begin{cases} \dot{x}_L = v_L \\ m_L \dot{v}_L = \eta_F K(v_L^d - v_L) - \mu\left(m_L g - (\eta_F + \eta_B)\, F_\ell\right) \end{cases} \tag{5.1}$$

where η_F denotes the number of *Front* robots and η_B denotes the number of *Back* robots.

5.3 Robot Controller Design for Mimicking Ant Behaviors during Collective Transport

In previous experimental work with *N. cockerelli* [78], data was obtained on the mean transport team dynamics and fit to a stochastic hybrid system (SHS) model in which ants switch between *Back*, *Front*, and *Detached* states at constant rates that signify probabilities per unit time. The state-transition system parameterized by these rates is a Markov process on a fully connected graph. Thus, the time evolution of the states is described by a third-order linear time-invariant (LTI) system, which consists of three coupled linear ordinary differential equations with constant coefficients. The system converges at an exponential rate to the following mean numbers of *Front* and *Back* ants at equilibrium:

$$[\mathit{Front}]^* \approx 5.78 \text{ ants} \quad \text{and} \quad [\mathit{Back}]^* \approx 3.54 \text{ ants} \quad \text{with time constant} \quad \tau \approx 52.08\,\text{s}. \tag{5.2}$$

From this data, three controller design methods are created for swarm-robotic mimicry of different aspects of the collective transport behavior exhibited by the ants. In Section 5.3.1, the *equilibrium population matching method (EPMM)* is described, which reproduces the average steady-state *Front* and *Back* populations from the ant data in the multi-robot scenario but does not necessarily reproduce the transient population dynamics. Section 5.3.3 introduces the *transient matching method (TMM)*, which shows how the transient population dynamics can also be controlled when the average distributions of environmental features around the swarm do not change. Finally, Section 5.3.4 describes the *rate matching method (RMM)*, which reproduces the ant *Back*-*Front* transition rates rather than the equilibrium allocations or system convergence rate. Results are provided in Section 5.4.2 that compare the outcomes of using these different methods.

In the EPMM, a control policy can be designed using predicted or experimentally determined values of the ratios of encounter rates, as opposed to the absolute values of these rates. These ratios will be invariant to changes in parameters such as robot density or speed, and so the equilibrium population distributions around loads will also be invariant to such changes. However, the transient dynamics of the system will be sensitive to changes in the absolute encounter rates. Thus, in both the TMM and RMM, the absolute encounter rates will need to be estimated in order to control the transient dynamics. Toward this end, Section 5.3.2 demonstrates how the EPMM can be used as a tool for inferring the encounter rates.

5.3.1 Equilibrium Population Matching Method (EPMM)

Here, a modeling and control approach is outlined that simplifies the design of the attachment-detachment probabilities discussed in Chapter 4, so that it is possible to guarantee that the mean *Front* and *Back* robot populations match those observed in the ant data. Adapting the work described in Chapter 4, the *Detached-Front* robot transitions and the *Detached-Back* robot transitions are modeled as two parallel chemical reaction networks (CRNs). The species in the CRNs are defined as r, a free (*Detached*) robot; U_F (U_B), an unbound zone on the *Front* (*Back*) side of a load; and B_F (B_B), a bound zone on the *Front* (*Back*) side of a load. Here, e_{uF} (e_{uB}) are defined as the mean per-robot rate of encounters between a single free robot and a single unbound zone on the *Front* (*Back*) side of a load, and e_{bF} (e_{bB}) are defined as the mean per-robot rate of encounters between a single free robot and a robot that is attached to the *Front* (*Back*) side of a load. A reaction $r + U_F \xrightarrow{p_{bF}e_{uF}} B_F$ signifies the following:

- The notation $r + U_F$ represents the event of a *free robot* r encountering an *unbound zone* U_F on the *Front* side of a load.
- The notation $\xrightarrow{p_{bF}e_{uF}} B_F$ represents how often such encounter events occur and result in the free robot r binding to the unbound zone U_F to produce a new *bound zone* B_F.

The full CRN model is given by:

$$\underbrace{\begin{array}{l} r + U_F \xrightarrow{p_{bF} e_{uF}} B_F \\ r + B_F \xrightarrow{p_{uF} e_{bF}} U_F + 2r \end{array}}_{\textit{Front} \text{ side}} \quad \text{and} \quad \underbrace{\begin{array}{l} r + U_B \xrightarrow{p_{bB} e_{uB}} B_B \\ r + B_B \xrightarrow{p_{uB} e_{bB}} U_B + 2r \end{array}}_{\textit{Back} \text{ side}} . \tag{5.3}$$

Strictly speaking, the *Front* and *Back* CRNs are coupled by the shared pool of free robot (r) reactants. However, for catalytic allocation to multiple task types, the equilibrium population for each type can be determined independently so long as there are more total robots than required for the desired equilibrium allocation as shown in Section 4.3. So in this application, the equilibrium *Front* and *Back* populations are independent of each other so long as there are more total robots than the desired number of attached robots. Consequently, *Front* and *Back* control strategies can be designed independently. The desired mean equilibrium quantities of unbound zones and bound zones on side $S \in \{F, B\}$ are denoted as U_S^* and B_S^*, respectively, and define the *target allocation ratio* as $B_F^*/(U_F^*+B_F^*)$. For example, if the *Front* boundary of every load is desired to be half covered with robots, then $U_F^* = B_F^*$ and the target allocation ratio is 0.5.

Due to the fact that the loads considered have a continuum of unbound zones (robots can attach anywhere on the loads), two attached robots on a load can have too little room between them to allow another robot to connect. This leads to a macro-scale model complication in which the presence of a unit of unbound space in the system does not necessarily imply that there is space available for further robot attachment. Consequently, the equilibrium of the well-mixed CRN model in Equation 5.3 must be appropriately adjusted to account for this non-well-mixed effect in the same fashion as described in Section 4.3.3. The relationship between the control pair (p_{bS}, p_{uS}) and the target equilibrium bound-to-unbound zone ratio B_S^*/U_S^* is:

$$\frac{p_{bS}}{p_{uS}} = \frac{1+\delta_S^*}{1-\delta_S^* \frac{B_S^*}{U_S^*}} \frac{e_{bS}}{e_{uS}} \frac{B_S^*}{U_S^*} \qquad \text{with} \qquad \delta_S^* \triangleq \delta_S\left(\frac{B_S^*}{U_S^*+B_S^*}\right) \tag{5.4a}$$

where the function $\delta_S : [0,1] \mapsto [0,1]$ is given by

$$\delta_S(r) = A\cos\left(2\pi\frac{r}{T_S}+c\right) \tag{5.4b}$$

and the parameter T_S is the largest value that ensures that $\delta_S(\gamma) = (1-\gamma)/\gamma$ for the *parking constant* $\gamma \approx 0.7476$ [53, 127]. The value of $\delta_S(r)$ represents the mean length of the unbound zones along sides S as a function of the target allocation ratio. The extremes of this function are $\delta_S(1) = 0$ and $\delta_S(0) = L_S/\lceil L_S \rceil \approx 1$. The quantity $\lceil L_S \rceil$ represents the total number of unbound zones, including the slack zone, on an empty side S. Thus, $L_S/\lceil L_S \rceil \approx 1$ is the average length of an unbound zone on an empty load. The parameter T_S is fixed by the value of $L_S/\lceil L_S \rceil$ and can be solved for during the design process. In general, $T_S \in [4.1, 4.77]$, which can be verified by solving for T_S over a range of $L_S \in [1, \infty)$. When robots are very small relative to the load (i.e, $L_S \approx \infty$), then $T_S \approx 4.76$.

Equation 5.4 constitutes a control law that maps the target bound-to-unbound zone ratio, B_S^*/U_S^*, to the ratio of probabilities, p_{bS}/p_{uS}, which can be set by the control designer. Equivalently, Equation 5.4 maps the target allocation ratio, $B_S^*/(U_S^* + B_S^*)$, to the ratio $p_{bS}/(p_{uS}+p_{bS})$, which will be referred to as the *control factor*. Each control factor has an extra degree of freedom that can be used for optimization. For the EPMM, $p_{uS} = 1$ is chosen for control factors less than 0.5 and $p_{bS} = 1$ otherwise. This enforces a one-to-one mapping between control factors and (p_{bS}, p_{uS}) pairs.

The relationship between control factors and allocation ratios is plotted in Figure 4.3. The figure shows that for a given encounter rate ratio e_{bS}/e_{uS}, this relationship is described by a monotonically increasing curve that is anchored by zero on the left and the parking constant on the right. If it is easier for a robot to detect a bound zone than an unbound zone (due to the extra size that a bound robot adds to an unbound zone), then allocations will be lower for the same control factor due to the increased frequency of robot-catalyzed unbinding. Conversely, if unbound zones are easier to detect than bound zones (because the motion primitive of recently unbound robots makes it very likely that they immediately re-encounter unbound zones), then the allocations will be higher for the same control factor. Hence, the relationship in Figure 4.3 is governed by the motion and sensing characteristics of the robots and not by environmental parameters that can change over time, including the robot density and the number and sizes of loads. The encounter rate ratio can be estimated in two ways:

- If robots detaching from a load are no more likely to re-encounter that load than other robots in the nearby vicinity, then the ratio e_{bS}/e_{uS} may be estimated as the ratio of the area of a bound zone to the area of an unbound zone. That is, for sides S that are shaped like half-circles,

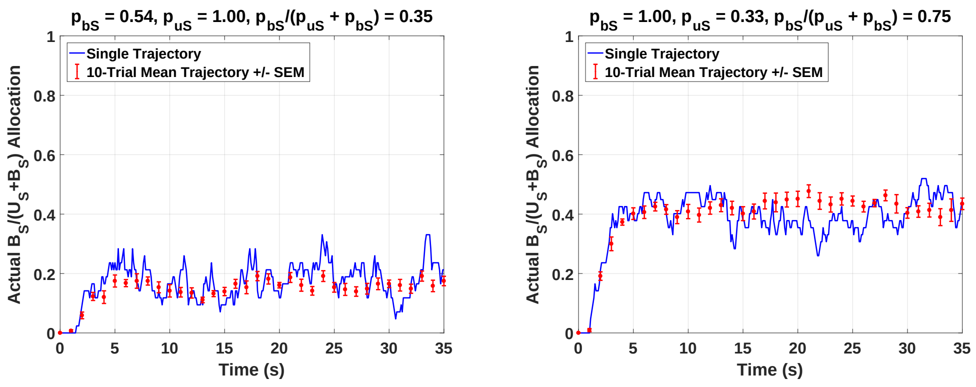
$$\frac{e_{bS}}{e_{uS}} \approx \frac{\frac{\pi R^2}{2M} + A}{\frac{\pi R^2}{2M}} = 1 + \frac{2AM}{\pi R^2}, \tag{5.5}$$

  where R is the radius of the load, A is the area of a circular robot, and M is the length of a side after normalizing to the arc length occupied by a single robot. However, it is often the case that robots that have recently detached from a load will have a higher encounter rate with unbound zones than robots that are randomly searching. Under these circumstances, estimation of the ratio e_{bS}/e_{uS} would require the method described below.

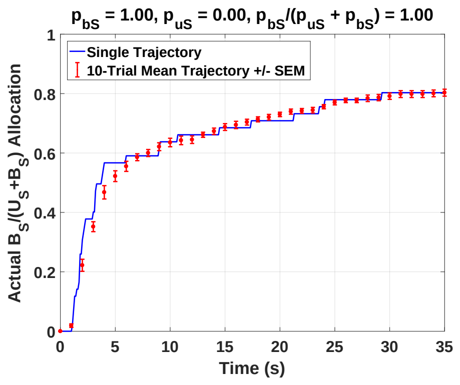
- An empirical estimate of e_{bS}/e_{uS} can be fit to a sampled version of Figure 4.3 that consists of data from robot simulations or experimental trials in which the control factor is varied. For example, Figure 5.3 shows the actual allocation ratio of a side S over time for $(p_{bS}, p_{uS}) = (1, 0.11)$. The data were generated in NetLogo from simulations of 300 robots in an arena with three loads. The solid line plots the allocation ratio of the side over time during one trial. The asterisks show the mean allocation ratio across 10 trials, with bars showing one standard error of the mean. The mean converges to the allocation ratio $B_S^*/(U_S^* + B_S^*)$ corresponding to the control factor $p_{bS}/(p_{uS} + p_{bS}) = 1/1.11 = 0.90$. In this particular case, the control factor 0.90 results in an equilibrium allocation ratio of 0.4. In Figure 4.3, the pair (0.90, 0.4) lies on the curve for $e_{bS}/e_{uS} \approx 4$. This process can be repeated for additional control factors to gain more confidence in the e_{bS}/e_{uS} estimate.

5.3.2 *Using the EPMM to Estimate Absolute Encounter Rates*

As discussed at the beginning of Section 5.3, the TMM and RMM approaches described in Section 5.3.3 and 5.3.4 require the estimation of absolute encounter rates in order to precisely control the transient dynamics and transition rates of the robotic swarm. Analytical predictions of encounter rates based on geometry are not feasible for scenarios where robots are significantly smaller than loads, and estimating encounter rates from empirical data is typically very difficult [68, 63]. There are typically unpredictable, nonlinear effects from motion primitives of the robots and the non-uniformly randomly distributed spacing of attachment points on the loads through the environment. However, as shown in Section 5.3.1, swarms that follow



(a) **(b)**



(c)

Figure 5.3. Time evolution of the allocation ratio on a side $S \in \{F, B\}$ from simulation trials in NetLogo (for simulation details, see Section 5.4.1). The solid line shows the trajectory of the allocation ratio for a single trial and the dot with error bars show the mean ± 1 standard error of the mean (SEM) across ten trials with the same control pair. Each simulation used 300 initially unbound robots in an arena with three loads.

the control policies encoded in the CRN (Equation 5.3) will converge in the mean to allocations which are only sensitive to the *ratio* e_{bS}/e_{uS} of the encounter rates e_{bS} and e_{uS}. This section discusses how measuring the convergence rate of a system designed using the EPMM allows for indirect inference of the encounter rates.

If the initial number r_0 of free robots is significantly larger than the number of robots needed to attain the desired allocation levels on the loads, then the likelihood of a free robot finding an unbound zone or bound zone will not vary over time, as described in Section 4.3.1. In other words, if the fraction of robots that are in the *Detached* state does not change appreciably over time, then the bimolecular CRNs in Equation 5.3 can be replaced by reversible unimolecular CRNs:

$$\underbrace{U_F \underset{p_{uF}e_{bF}r_0}{\overset{p_{bF}e_{uF}r_0}{\rightleftharpoons}} B_F}_{\textit{Front}\text{ side}} \qquad \text{and} \qquad \underbrace{U_B \underset{p_{uB}e_{bB}r_0}{\overset{p_{bB}e_{uB}r_0}{\rightleftharpoons}} B_B}_{\textit{Back}\text{ side}}\,. \tag{5.6}$$

That is, because the system effectively has a buffered capacity of r_0 robots, the free robot species r in Equation 5.3 has been replaced with the scalar r_0, which is included in the rate constants in Equation 5.6. So although all reactions in Equation 5.3 are irreversible, they approximate the reversible system in Equation 5.6. It is easy to show [15, 106] that the mean-field dynamics of a set of unimolecular reversible reactions are linear and time-invariant (LTI). In particular, for each side $S \in \{F, B\}$, the set of two LTI ordinary differential equation for the $U_S \rightleftharpoons B_S$ CRN has two eigenvalues,

$$\lambda_0 = 0 \qquad \text{and} \qquad \lambda_1 = -\left(p_{bS}e_{uS}r_0 + p_{uS}e_{bS}r_0\right),$$

where λ_0 represents the conservation of the sum $U_S + B_S$, and $-\lambda_1$ is the sum of the two mass-action rates. Moreover, the time constant $\tau_S = -1/\lambda_1$ for each side

$S \in \{F, B\}$ is

$$\begin{aligned}
\tau_S &= \frac{1}{(p_{bS} e_{uS} + p_{uS} e_{bS})\, r_0} && \text{(5.7a)}\\
&= \frac{1}{\left(p_{bS} + p_{uS} \frac{e_{bS}}{e_{uS}}\right) e_{uS} r_0} && \text{(5.7b)}\\
&= \frac{1}{\left(\frac{p_{bS}}{p_{uS}} + \frac{e_{bS}}{e_{uS}}\right) e_{uS} p_{uS} r_0} \overset{\text{(by Equation 5.4a)}}{=} \frac{U_S^* - \delta_S^* B_S^*}{U_S^* + B_S^*} \frac{1}{e_{bS} p_{uS} r_0} && \text{(5.7c)}
\end{aligned}$$

where three equivalent forms have been provided in Equation 5.7 for application--specific convenience. In general, the relationship between τ_S and the control factor $p_{bS}/(p_{uS} + p_{bS})$ depends on the absolute encounter rates e_{uS} and e_{bS}. However, as shown in Equation 5.7, the shape of the τ_S curve is fixed by the e_{bS}/e_{uS} ratio and the absolute encounter rates only scale that relationship. Some example τ_S curves are shown in Figure 5.4 for five different encounter-rate ratios. Hence, for each of several (p_{bS}, p_{uS}) pairs, a mean step response like Figure 5.3 can be generated to determine the equilibrium allocation ratio $B_S^*/(U_S^* + B_S^*)$ and time constant τ_S for that pair. By the methods described at the end of Section 5.3.1, the allocation ratio data will determine the effective e_{bS}/e_{uS} ratio. Then, Equation 5.7 can be solved to yield an estimate of encounter rate e_{uS} (and thus e_{bS} as well). Using this method, encounter rates do not need to be known or solved for prior to simulating the system, but rather can be inferred from the observed behavior of the swarm.

Small-swarm case: If, initially, there are few free robots, the transient response to the sudden introduction of several unoccupied loads will be better described by Equation 5.3 than Equation 5.6. Moreover, the step response will rise with a logistic as opposed to an exponential shape. However, if the resulting equilibrium allocations are perturbed by numbers of bound and unbound zones that are low enough to not appreciably change the number of free robots when the system restores equilibrium,

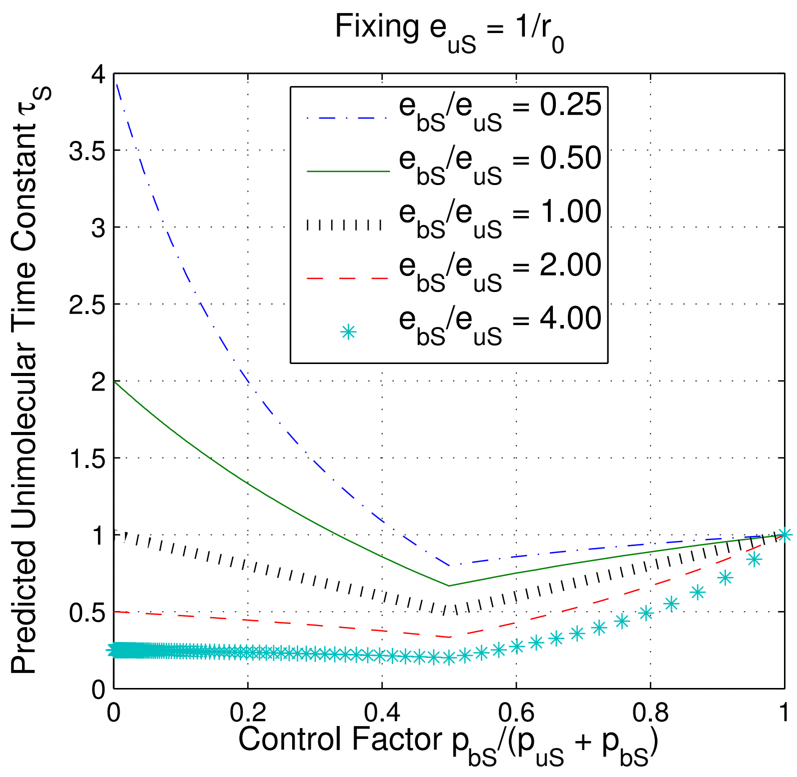


Figure 5.4. Effect of encounter-rate ratio on the relationship between the control factor $p_{bS}/(p_{uS} + p_{bS})$ and the approximate time constant τ_S for convergence to the equilibrium allocation on side $S \in \{F, B\}$ (*Front* or *Back*). The solution of a first--order linear time-invariant system (an exponential curve) can be fitted to the mean step response in Figure 5.3 to predict the time constant τ_S. Here, the predicted relationship between the control factor $p_{bS}/(p_{uS} + p_{bS})$ and the time constant τ_S is shown, assuming that the system can be approximated as a unimolecular CRN. Although the actual convergence rate depends on the absolute encounter rates e_{bS} and e_{uS}, the shape of the time constant curve is fixed by the encounter-rate ratio e_{bS}/e_{uS}. As in Figure 4.3, $p_{uS} \equiv 1$ is set for control factors less than 0.5 and $p_{bS} \equiv 1$ otherwise.

then the transient response will be approximately exponential and the LTI-based encounter rate estimates above can still be applied.

5.3.3 Transient Matching Method (TMM)

In Section 5.3.1, a method is shown to choose the p_{bS}/p_{uS} ratio such that a desired bound-to-unbound zone ratio B_S^*/U_S^* at equilibrium on a side $S \in \{F, B\}$ is achieved. This control policy provides one degree of freedom to the designer. Specifically, there is a line of (p_{bS}, p_{uS}) pairs that achieve the same B_S^*/U_S^* ratio at equilibrium. Consequently, the control policy can be optimized over this space. In Section 5.3.2, it was shown how convergence rate measurements can be used to infer the absolute encounter rates e_{bS} and e_{uS}. Once these encounter rates are known, the precise (p_{bS}, p_{uS}) pair can be chosen that most accurately reproduces the transient dynamics exhibited by an ant collective transport team. In particular, the values of the robot swarm size r_0 and the probability p_{uS} (or, equivalently, p_{bS}) can be chosen to guarantee that the system converges with a desired time constant.

Here, it is assumed that the swarm is sufficiently large to allow the use of the unimolecular approximation in Equation 5.6 with the time constant in Equation 5.7. For any desired bound-to-unbound zone ratio B_S^*/U_S^*, Equation 5.4 specifies the corresponding p_{bS}/p_{uS} ratio. Moreover, Equation 5.7c provides an expression for the time constant τ_S with free parameters e_{uS}, p_{uS}, and r_0. As described in Section 5.3.2, the rate e_{uS} can be inferred from the EPMM approach. So, if τ_S^* is a desired time constant, the product $r_0 p_{uS}$ should be chosen so that $p_{uS} \in [0, 1]$ and

$$r_0 p_{uS} = \frac{1}{\left(\frac{p_{bS}}{p_{uS}} + \frac{e_{bS}}{e_{uS}}\right) e_{uS} \tau_S^*}. \tag{5.8}$$

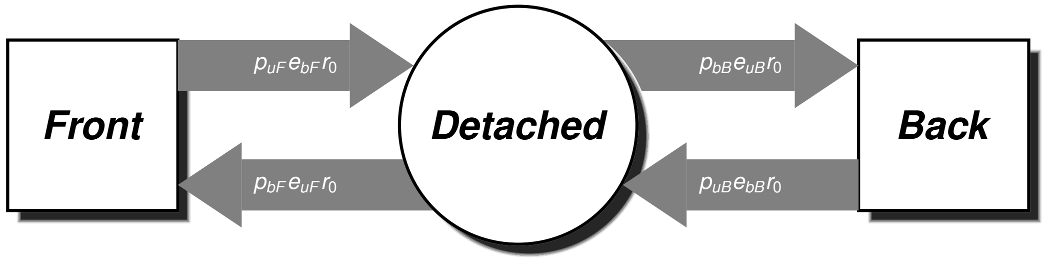
For any swarm size r_0 and p_{bS}/p_{uS} ratio, the probability p_{uS} (or, equivalently, p_{bS})

can be chosen to scale the convergence rate, with the fastest convergence at $p_{uS} = 1$ and very slow convergence at $p_{uS} \approx 0$. So, to achieve any time constant τ_S^*, the number of robots r_0 must be sufficiently high so that p_{uS} can then be used to scale the convergence rate to the desired value.

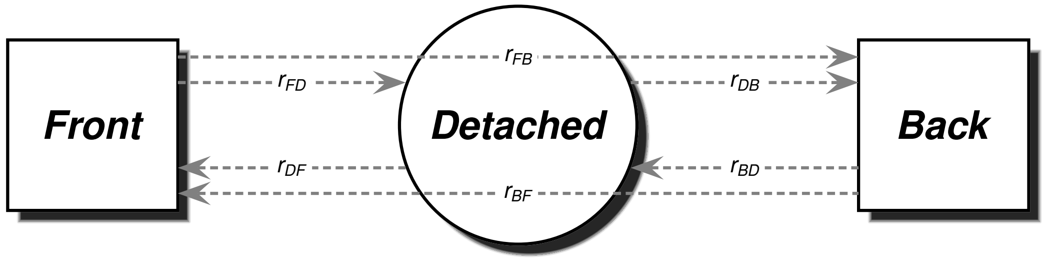
5.3.4 Rate Matching Method (RMM)

In Section 5.3.1 and 5.3.3, a method to achieve a desired equilibrium mean allocation at a desired convergence rate is described. Alternatively, there may be some applications where the priority is to match the fluxes of ants and robots switching between states, as opposed to their equilibrium allocation levels. Toward this end, a method for matching the *Front-Back* transition rates of the ant and robot systems is presented.

Under the assumption of a relatively large robotic swarm with respect to the number of bound zones at equilibrium, the robot state transitions can approximated as a set of two unimolecular reversible reactions like Equation 5.6 with LTI dynamics. Because it is not physically possible in this system for a robot to directly transition from *Back* to *Front* and vice versa, transitions to an intermediate *Detached* state are necessary. Figure 5.5a shows a Markov chain representing the transitions that a robot can execute. The system has four rate constants: two for the *Front-Detached* transitions and two for the *Back-Detached* transitions. A complication arises from the fact that the ant data were sampled at 5-second intervals [78], and so some recorded ants appear to transition directly from one side of the load to the other. Because of this, the SHS model of the ant collective transport dynamics has six rate constants: the same four as in the multi-robot scenario (r_{FD}, r_{DF}, r_{BD}, r_{DB}), plus two more for the *Back-Front* transitions (r_{BF}, r_{FB}). The resulting Markov chain followed by the observed ants is shown in Figure 5.5b. Comparing Figure 5.5a and 5.5b, it is clear



(a) Markov chain of implemented robot



(b) Markov chain of sampled ant

Figure 5.5. Single-ant and single-robot Markov chains. Each agent, ant or robot, transitions among *Front*, *Detached*, and *Back* states. In a robotic implementation with random attachment and catalytic detachment, direct transitions between *Back* and *Front* are not possible. Consequently, in (a), only four transition rates are shown. However, in data collected from ants, some ants appear to transition directly between the two attached states. Consequently, the model fit to the ant data introduces two additional transition rates, r_{FB} and r_{BF}, which are shown in (b) as transitions cutting directly across the *Detached* state. To reconcile (a) and (b), the r_{FB} rate can be added to both r_{FD} and r_{DB}. Similarly, the r_{BF} rate can be added to both r_{BD} and r_{DF}. In other words, in order to observe a stable r_{FB} (or, similarly, r_{BF}) rate in sampled data, there must be a hidden flux of *Front*-to-*Detached* ants that matches a flux of *Detached*-to-*Back* ants.

that each direct transition from *Front* to *Back* could be alternatively recorded as a transition from *Front* to *Detached* followed by a transition from *Detached* to *Back*. Thus, this strategy incorporates the r_{FB} and r_{BF} rates from the ant model into the robot control policies by adding them to the the *Front-Detached* and *Back-Detached* rates in the multi-robot CRN:

$$\underbrace{U_F \underset{r_{FD}+r_{FB}}{\overset{r_{DF}+r_{BF}}{\rightleftharpoons}} B_F}_{\textit{Front}\text{ side}} \qquad \text{and} \qquad \underbrace{U_B \underset{r_{BD}+r_{BF}}{\overset{r_{DB}+r_{FB}}{\rightleftharpoons}} B_B}_{\textit{Back}\text{ side}}, \tag{5.9}$$

Then, equating the rate constants in Equation 5.6 and 5.9 yields the programmed robot attachment-detachment probabilities

$$\begin{aligned} p_{uF} &= \frac{r_{FD}+r_{FB}}{e_{bF}r_0}, & p_{bB} &= \frac{r_{DB}+r_{FB}}{e_{uB}r_0}, \\ p_{uB} &= \frac{r_{BD}+r_{BF}}{e_{bB}r_0}, \quad \text{and} & p_{bF} &= \frac{r_{DF}+r_{BF}}{e_{uF}r_0} \end{aligned} \tag{5.10}$$

where e_{bS} and e_{uS} can be determined using the methods in Section 5.3.2 for each side $S \in \{F, B\}$. To ensure that these probabilities take values in the interval $[0, 1]$, the swarm size r_0 must be chosen sufficiently large so that

$$r_0 \geq \max\left\{\tfrac{r_{FD}+r_{FB}}{e_{bF}},\ \tfrac{r_{DB}+r_{FB}}{e_{uB}},\ \tfrac{r_{BD}+r_{BF}}{e_{bB}},\ \tfrac{r_{DF}+r_{BF}}{e_{uF}}\right\}.$$

Expected RMM results and caveats:

Under the conditions above, the robotic swarm will converge to the *Front* and *Back* allocation ratios

$$\begin{aligned}\frac{B_F^*}{U_F^* + B_F^*} &= \frac{1}{1+\delta_F^*}\frac{r_{DF}+r_{BF}}{r_{FD}+r_{FB}+r_{DF}+r_{BF}} \quad \text{and} \\ \frac{B_B^*}{U_B^* + B_B^*} &= \frac{1}{1+\delta_B^*}\frac{r_{DB}+r_{FB}}{r_{BD}+r_{BF}+r_{DB}+r_{FB}},\end{aligned} \tag{5.11a}$$

which can be derived from the combination of Equation 5.10 and Equation 5.4. Moreover, from the combination of Equation 5.10 and Equation 5.7a, the system will converge with *Front* and *Back* time constants

$$\tau_F = \frac{1}{r_{FD}+r_{FB}+r_{DF}+r_{BF}} \quad \text{and} \quad \tau_B = \frac{1}{r_{BD}+r_{BF}+r_{DB}+r_{FB}}.$$

Although the RMM is the most direct attempt at replicating the microscopic behaviors of the ants on robotic platforms, it is the least likely to match macroscopic transport properties such as equilibrium team sizes and convergence rates. At a basic level, there are zero degrees of freedom in Equation 5.11 and Equation 5.12. Thus, the goal of the RMM to match rate constants across systems eliminates any ability to control the equilibrium of the system or its convergence rate. Moreover, fundamental differences between the SHS ant model and the multi-robot stochastic implementation make equilibrium or convergence-rate matching only possible by unlikely coincidence. The unimolecular CRN used to model the populations dynamics of the multi-robot system relies on the assumption of a sufficiently large swarm size r_0 that is buffered against changes due to robot attachments and detachments. However, the ant population sizes associated with the SHS model data were on par with the eventual transport team sizes. Consequently, the number of free ants was appreciably decreased as teams formed around loads, and thus the attachment rate was limited by the fewer available free ants for further attachment. Therefore, it is generally expected that the TMM will produce a closer match in the population dynamics than the RMM when the reference ant swarm is small. The TMM allows for implementation-level dissimilarities in order to better match desired macroscopic characteristics.

Although the TMM will better match macroscopic allocation properties, there can be cases where fluxes from one region to another are more relevant than coverage around regions. In these cases, a RMM-like approach may be more appropriate. For example, consider a case like the one described by [145] where robots have been programmed with an algorithm to track the shapes, sizes, and movements of puddles in an area by skirting the perimeter of each puddle. Similar to the collective transport problems presented here, the boundaries of the puddles are analogous to the boundaries of the loads with the robots “attaching” to their boundaries in each case. If a central server is using swarm robots to find these puddles and sample their continuous time evolution, having a high concentration of robots on the boundaries corresponds to having a high spatial resolution of each sample. However, it may instead be of critical importance to return frequently enough to the central location to meet sampling-rate requirements (temporal resolution may be prioritized over spatial resolution). In this latter case, controlling the transition rate between the boundary and the central server allows for controlling the sampling rate. By using stochastic attachment-detachment policies designed by the RMM, ingress and egress fluxes (as opposed to boundary coverage at any given time) can be controlled, thus setting the sampling rate of the distributed swarm sensor system.

5.4 Validation of Collective Transport Control Strategies in Simulation

5.4.1 Simulation Setup

The control policies computed by the EPMM, TMM, and RMM were tested with simulations of multi-robot collective transport in the agent-based model simulation software NetLogo [139]. Each of these three methods generated a different set of attachment and detachment stochastic control policies, (p_{bF}, p_{uF}) and (p_{bB}, p_{uB}); all other parameters were held constant across the simulations. To demonstrate that the control policies are effective even with a single load, a scenario depicted in Figure 5.6 is considered. In the simulations, $r_0 = 200$ robots with radius 1 cm are initially placed randomly throughout an arena of size 125 cm $\times$ 50 cm. A rigid circular load with radius 8 cm and mass 2.3 g is placed in the arena with the robots. During the simulations, each robot runs the controller shown in Figure 5.2 with one of the three sets of control policies. The *Detached* robots perform correlated random walks with a turning angle of 45°. Using experimentally measured values of ant speeds and ant forces applied to a vision-based force sensor as guidelines [17], the robot speed is set to 10 cm/s and the maximum robot pulling force to 10.5 mN.

5.4.2 Results

Following the procedure discussed in Section 5.3, the EPMM, TMM, and RMM is used to design transport behaviors for individual robots that replicate different aspects of previously collected ant data [78]. As described in Section 5.3.1 for the EPMM, the encounter rate ratio e_{bS}/e_{uS} for each side $S \in \{F, B\}$ was estimated by running a multi-robot transport simulation with different values for the control factor $p_{bS}/(p_{uS} + p_{bS})$. In principle, this calibration process need only be done for a single control factor. A range of control factor values are used here to show how well the

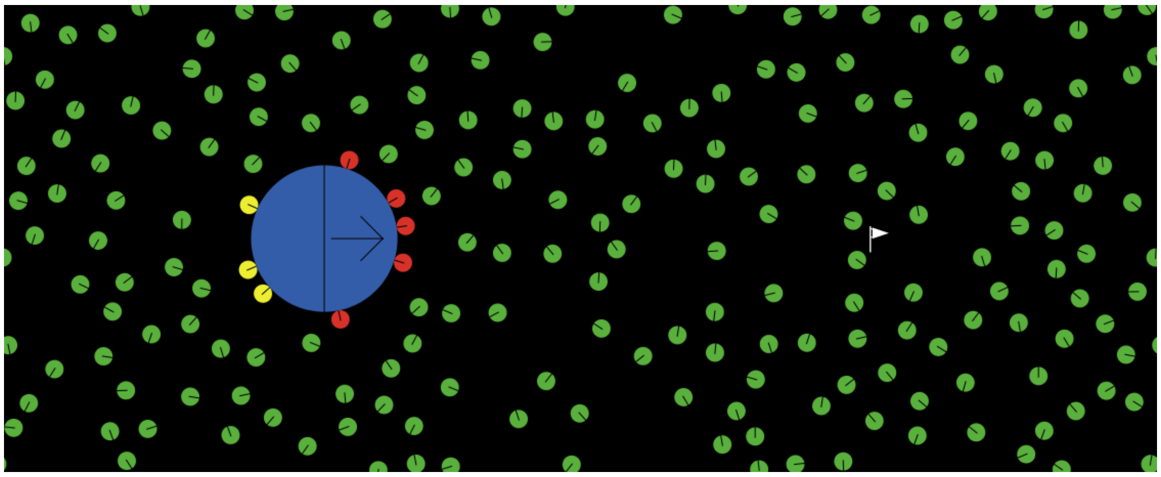
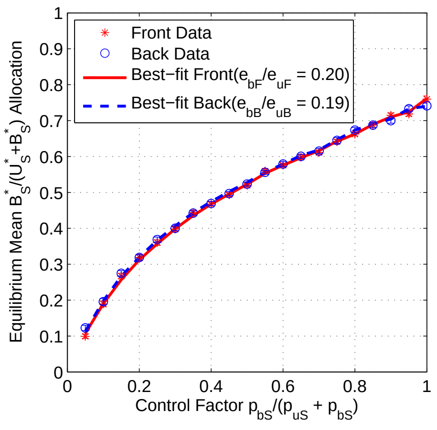


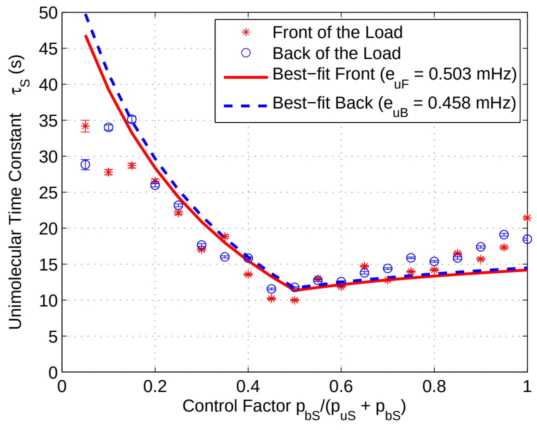
Figure 5.6. A screen shot from the NetLogo simulation of collective transport. The large blue circle represents the load, red circles are *Front* robots, yellow circles are *Back* robots, and green circles are *Detached* robots. The arrow on the load indicates its direction of motion during transport, and the white flag indicates its goal position.

microscopic multi-robot simulation matches macroscopic predictions. Like the ant experiments, the multi-robot simulations were run with a rigid circular load. Because the *Front* and *Back* sides of the loads have identical geometries, it is expected that the encounter rate ratios for both sides will be identical. Figure 5.7a confirms that this is indeed the case: $e_{bF}/e_{uF} = 0.20$ and $e_{bB}/e_{uB} = 0.19$.

For the TMM and RMM, the absolute encounter rates e_{uS} and e_{bS} must be estimated for sides $S \in \{F, B\}$ according to the procedure in Section 5.3.2. So, for each ensemble of trials used to generate Figure 5.7a, the average step responses similar to the one in Figure 5.3 were also plotted and their rise times were estimated. The resulting time constants are shown in Figure 5.7b. Using the encounter rate ratios already estimated for the EPMM, the relationships in Figure 5.4 are fit to these data to determine that $e_{uF} = 0.503$ mHz and $e_{uB} = 0.458$ mHz, which could then be used with the encounter rate ratios to compute e_{bF} and e_{bB}. These encounter rates are small because they represent the rate at which an individual robot (in an empty



(a) Estimation of e_{bS}/e_{uS} from equilibria.



(b) Estimation of e_{bS}, e_{uS} from convergence rates.

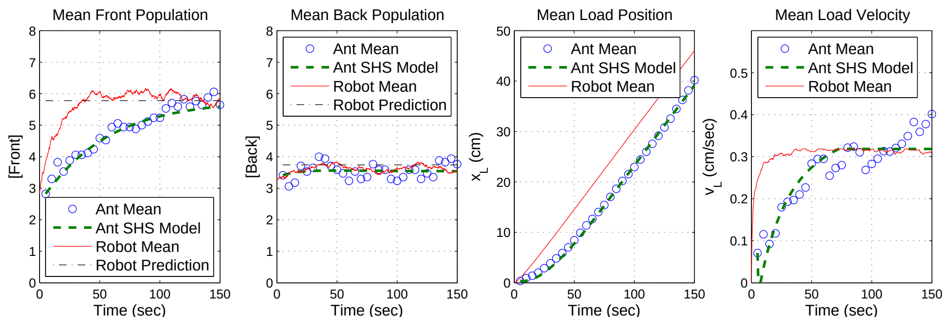
Figure 5.7. Encounter rate estimation results. As in Figure 5.3, a set of 100 identical multi-robot simulation trials were performed in NetLogo for each control factor value. Averaging across the set produced the *Front* and *Back* mean allocation ratios plotted in (a) with error bars showing standard error of the mean (SEM). The expected relationships shown in Figure 4.3 were then fit to these empirical curves, yielding a e_{bS}/e_{uS} ratio of approximately 0.2 for each side $S \in \{F, B\}$. The *Front* and *Back* time constants estimated from the average step response for each control factor value are plotted in (b). Using the $e_{bS}/e_{uS} \approx 0.20$ relationship from Figure 5.4, these time constants were used to infer the encounter rates e_{bS} and e_{uS} for each side $S \in \{F, B\}$.

arena) would encounter a single specific zone of a load. This event happens rarely per robot, but the number of encounters that happen across a swarm is significantly larger.

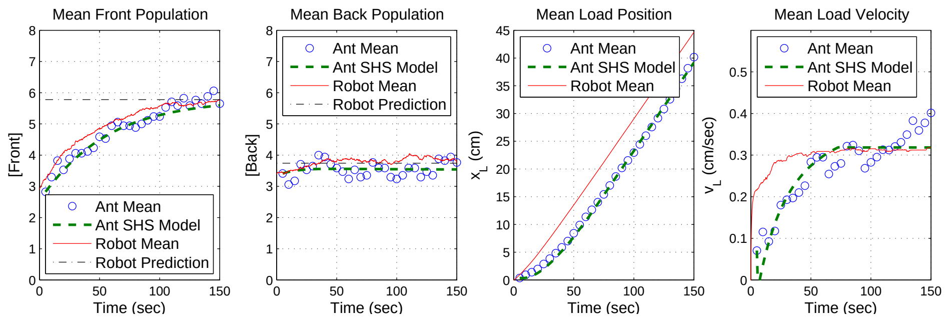
Now equipped with e_{bS}/e_{uS}, e_{bS}, and e_{uS} for each side $S \in \{F, B\}$, the control law Equation 5.4 is applied to, 5.8, and 5.10 to find the (p_{bS}, p_{uS}) control pairs using the EPMM, TMM, and RMM. The simulation results are shown in Figure 5.8 and are overlayed with the ant data, the SHS ant model trajectories, and the predicted equilibrium allocations. In all three cases, the mean equilibrium numbers of *Front* and *Back* robots are lower than the predicted values by less than 1 robot. This small discrepancy may be corrected by better estimating the e_{bS}/e_{uS} encounter rate ratio. Moreover, because each side of the load can only support 12 attached robots that are optimally packed (zero free space between attached robots on a load), quantization error can be a significant accuracy limitation in this scenario. For larger loads or smaller robots, more accurate mean allocations can be achieved. As shown by Figure 5.8a and Figure 5.8b, the mean equilibrium allocations from the 100 EPMM simulations agree with those from the 100 TMM simulations. Consequently, although the theoretical predictions can be slightly improved, the allocation results in general are highly repeatable.

5.4.3 Discussion

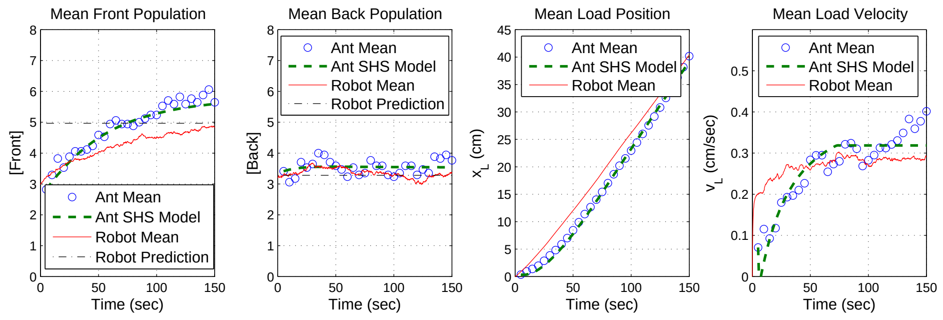
This chapter has presented three methods of designing stochastic encounter-based controllers that drive robotic swarms to transport payloads along linear trajectories. In the EPMM used to generate Figure 5.8a, the extra degree of control freedom was used to maximize the convergence rate to equilibrium. As a result, the time constants τ_F and τ_B differ from those of the ant population dynamics, and the τ_F constant results in a much faster rise in the mean *Front* robot population than in the ant data. Because the transport team forms much faster, it achieves velocity regulation more quickly. Because of this, the position of the simulated load takes an early but bounded lead with respect to the position of the load in the ant data. These results show that the EPMM is most suited for cases where it is necessary to achieve steady transport of the load in the minimum possible amount of time. The EPMM control policies can produce rapid changes in the *Front* population, leading to quick fluctuations of the pulling force on the load and thus to a jerky start to the load's motion. However, the method guarantees the fastest convergence to the load velocity set point and target equilibrium allocations.



(a) Equilibrium Population Matching Method ($B_F^* = 5.78$, $B_B^* = 3.74$; expected $\tau_F = 7.68$ s, $\tau_B = 13.96$ s)



(b) Transient Matching Method ($\tau_F^* = \tau_B^* = 52.08$ s)



(c) Rate Matching Method (rates from [78]; expected $B_F^* = 5.09$, $B_B^* = 3.35$, $\tau_F = 20.62$ s, $\tau_B = 11.27$ s)

Figure 5.8. Population dynamics and load position and velocity during collective transport by ants and robots. The plots show results from 100 simulated trials with robot control policies designed using each of the three methods described in Section 5.3. The left two columns show the time evolution of the mean numbers of *Front* and *Back* individuals; the right two columns show the trajectories of the mean position and speed of a single non-rotating load moving along a line. In each graph, the reference data from the ant trials are shown as circles, trajectories from the SHS model fit to the ant data are shown as dashed curves, and the multi-robot simulation data is shown as a solid line. The horizontal dash-dotted lines in the left two columns show the predicted equilibrium allocations based on the control approach.

The TMM reduces the discrepancy between the transient dynamics of the ant and robot transport teams, as evidenced by the results in Figure 5.8b. The TMM results show the same equilibrium team sizes as the EPMM, as expected, but the rise time of the *Front* robot population more closely matches that of the *Front* ant population. Moreover, the rise time of the load speed is increased, and the lead of the load position with respect to the ant team's load trajectory is thereby decreased. However, although the TMM is able to better match the transient population dynamics of the ants, the load velocity still rises too quickly compared to the ant data. This disparity may be due to differences in how nonlinearities are implemented in the SHS and NetLogo friction models. In particular, the SHS model appears to show a delay before its initial rise that is not reflected in the NetLogo simulation. Another factor could be differences in the effectiveness and degree of coordination between the robot and ant transport teams. The simulated *Front* robots always pull the load in the desired direction with a force defined according to a prescribed control law. However, the ants could be counteracting each other's efforts by pulling in different directions, and they may not be exerting their maximum possible forces. Unlike the EPMM, the TMM requires information on the absolute encounter rates, but it allows for control over the transient population dynamics. This control can effect a slower growth of the transport team if desired, which can produce a smoother acceleration of the load.

Finally, the RMM results in Figure 5.8c show that control design based on rate constant matching does not reproduce any macroscopic properties of the ant teams in the simulated multi-robot transport system. The RMM leaves no degrees of freedom to control the allocation ratio, and so the predicted allocation ratios differ from the EPMM and TMM cases. However, this method is useful when it is necessary to control the fluxes of robots that join and leave the load boundary. In the EPMM and TMM cases, there is no control over the frequency of attachment and detachment events that occur during transport. The RMM controls the rate at which these events occur and can be useful when frequent robot attachment and detachment is risky to the transport dynamics or to the object being transported (if the load is brittle).

Chapter 6

DESIGN OF BIOMIMETIC CONTROL STRATEGIES FOR COLLECTIVE PAYLOAD TOWING

Source: Wilson et al. [140]

In a recent experimental study of collective towing by *Novomessor cockerelli* ants [31], a species of desert ant that is capable of highly coordinated, stable transport of large food items in teams [66], the steady-state load transport speed was observed to decrease with increasing team size when the load weight per ant was held constant. The objective of the work in this chapter is to develop hypotheses about the ant behavioral mechanisms that produce this effect and test these hypotheses through experiments on a multi-robot testbed. It is assumed that all the ants in the transport team know the direction to the goal (the ants' nest) and can navigate even while moving backwards [124], but that they have no knowledge of the load characteristics, the number of teammates, or their location on the load relative to its center of mass.

Another experimental study [30] on group transport by *N. cockerelli* also demonstrated that individual ants move loads faster than teams even when the ant was carrying the same amount of weight as the teams. However, other factors besides the payload weight could have contributed to this drop in speed. The ants were able to grip the load anywhere along its perimeter, and ants at different positions around the load may have varied in their posture and behavior, and hence their applied force and speed, depending on their orientation with respect to the transport direction. For instance, in a previous dynamical model of these observations [78], ants on the leading edge of the load pulled and lifted, while ants on the other half of the load only lifted. In the more recent study [31], this phenomenon is re-investigated for a more constrained scenario in which the ants could only transport a rectangular load by pulling on strings attached to one of its sides.

In this work, several Pheeno robots are used to test hypotheses about the ant behaviors in order to gain insight into the cause of the load speed decrease with respect to team size found in [31]. Section 6.1 describes the ant experimental trials conducted in [31] and the multi-robot testbed that was designed to mimic the ant experiments. Section 6.2 summarizes the data from [31] on the decrease in ant transport speed with increasing team size, and Section 6.3 presents two candidate models of collective towing behavior in an effort to reproduce this observed trend. The first model assumes a team of ants with heterogeneous maximum speeds that pull on the load with continuous, variable forces, requiring the team to move at the speed of the slowest member for stable transport to occur. The second model assumes a homogeneous team of ants that pull on the load with intermittent, identical forces as they take uncoordinated steps backward toward the nest. For both models, the average steady-state transport speed is predicted and compared to the ant data. Section 6.4 proposes a decentralized robot controller for cooperative towing that can be implemented on a team of robots with heterogeneous maximum speeds. The controller is based on a reinforcement learning algorithm that uses only stigmergic feedback, similar to the type of information that would be available to the ants. This controller was implemented on teams of two to four Pheeno robots that cooperatively towed a rectangular payload, along with a second controller that produced intermittent, uncoordinated pulling forces. Section 6.5 discusses the steady-state transport speeds in these robot experiments and compares them to the transport speeds observed in the ant experiments.

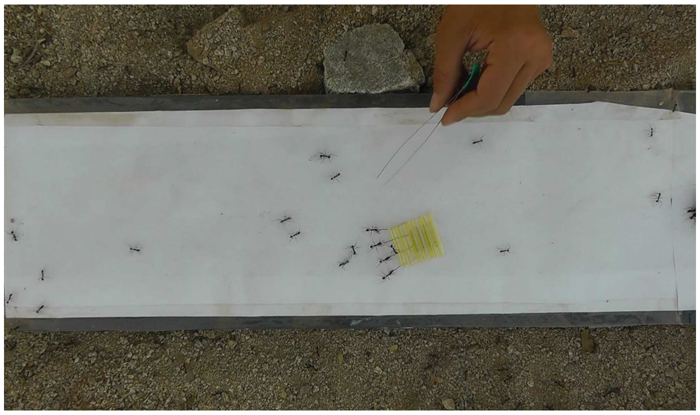


Figure 6.1. Experimental setup for tracking ant group transport of an artificial payload.

6.1 Materials and Methods

6.1.1 Experiments with N. cockerelli *Ants*

Three colonies of *N. cockerelli* were filmed in South Mountain Park in Phoenix, Arizona during collective transport of an artificial payload. Experiments were carried out in June 2013 during the cooler early morning hours (05:30-07:30), when foragers were active outside the nest. The experimental arena, shown in Figure 6.1, was a 61 cm $\times$ 46 cm $\times$ 0.5 cm Plexiglas$^{®}$ sheet covered in white paper. This sheet was leveled and placed 20 cm south of the main nest entrance. To control the size of the transport team, four artificial loads were constructed with one, two, three, and four attachment points for the ants. The loads were fabricated from segments of drinking straws and had dimensions of 3 cm $\times$ 4 cm. Along the 3 cm edge, 1 cm-long silk threads were attached at 1 cm intervals. Pieces of ethylene vinyl acetate (EVA) foam coated in fig paste were affixed to the free ends of the threads to attract the ants for transport. The loads weighed 0.3 g per attachment point (e.g., a load for two ants to transport weighed 0.6 g). This load design constrained the ants to grip only at the free ends of the threads and pull the load while walking backward.

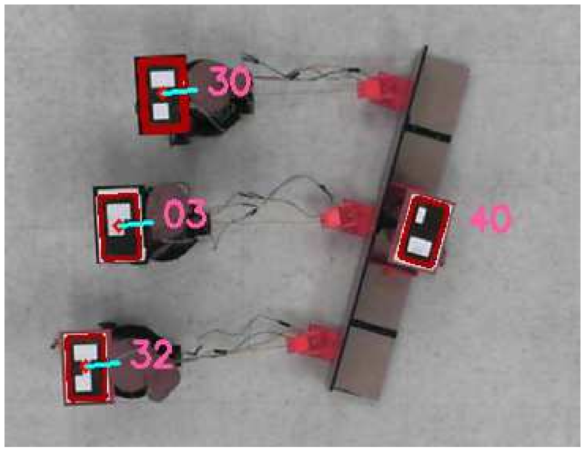
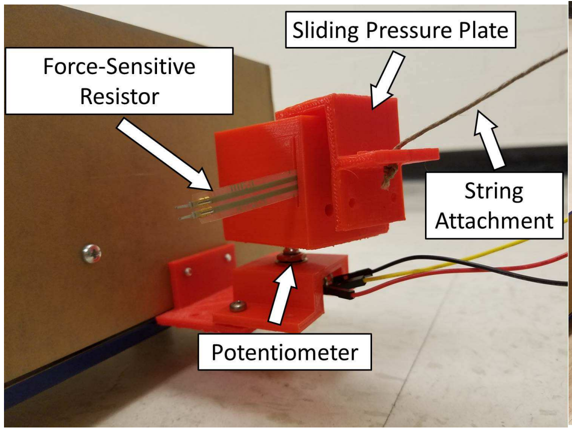


Figure 6.2. Experimental setup for tracking a group of Pheeno robots during transport of a payload.

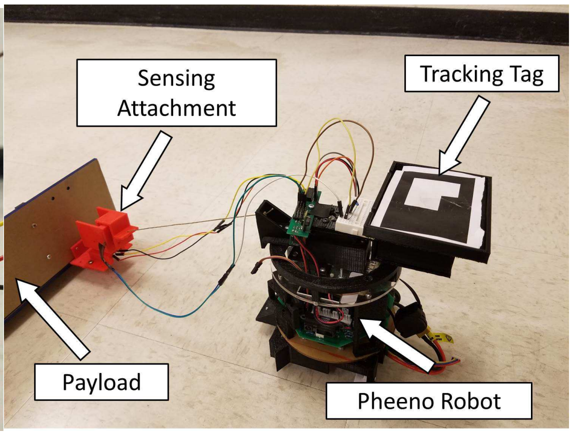
The transport was filmed with a Canon G12 camera mounted on a Manfrotto 055 Series tripod. The camera's field of view was 1280 pixels × 720 pixels, centered on the sheet. Foragers were recruited to a whole fig placed on the edge of the Plexiglass® sheet furthest from the nest. Once around 10 foragers were feeding on the fig, the fruit was replaced with an artificial load. For each of the three colonies, ten experimental trials per team size with transport teams of one to four ants were recorded.

6.1.2 Experiments with Multi-Robot Transport Teams

Collective towing experiments were conducted with teams of two, three, and four Pheeno robots. Figure 6.2 shows an overhead snapshot of the experimental setup with three robot transporters. The payload in these experiments was a 76.2 cm × 10.2 cm × 10.2 cm L-shaped acrylic frame weighing 500 g. A 3D-printed plastic basket attached to the frame allowed different masses to be added. A sensor suite was designed for each robot to measure the force vector that it applied to the load throughout the transport process. The sensor suite consisted of a 3D-printed sliding pressure plate mounted on a potentiometer that allowed 270° of rotation. A circular force-sensitive



(a)



(b)

Figure 6.3. Details of the multi-robot towing experimental setup. (a) A close-up, labeled view of a sensor suite attached to the payload. (b) A single robot attached to the payload.

resistor was used to measure the magnitude of the force that the pressure plate exerted due to the robot's pulling force, and the potentiometer determined the direction of the force with respect to the load's orientation. The sensor suites were affixed to the payload, and a hemp cord connected each robot to a sensor suite, as shown in Figure 6.3. The experiments were filmed with an overhead camera (Microsoft Life Cam, resolution of 720p) at a rate of 30 frames per second. The robots and payload were marked with 2D binary identification tags to enable real-time tracking of their positions and orientations by the overhead camera.

6.2 Observations of Ant Team Size versus Steady-State Payload Speed

From the video recording of each *N. cockerelli* experimental trial, the longest recorded segment in which the ants were smoothly transporting the load and the transport team size remained stable was extracted. In these segments, the position of the center of the load was tracked at 10-second intervals using ImageJ [123] and the Mtrack plugin [99]. The transport path was assumed to be straight between

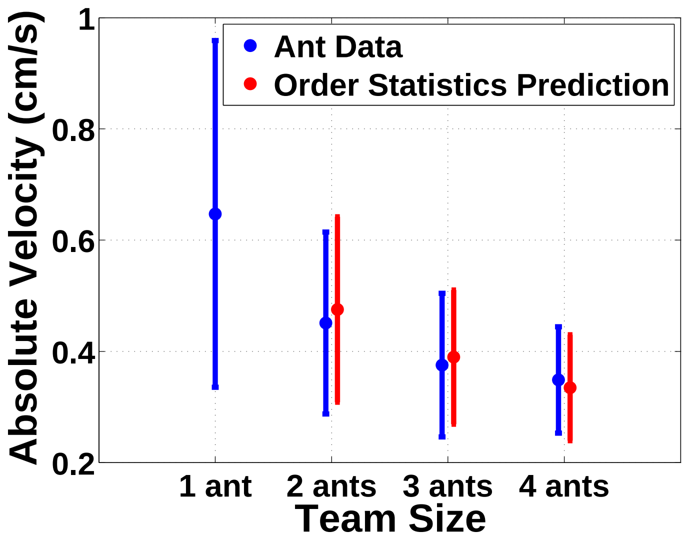


Figure 6.4. *Blue plots:* Steady-state speed of the artificial load (0.3 g per ant) with varying team size [31]. The circles with error bars represent the average ± standard deviation across 30 experimental trials (10 trials per ant colony). *Red plots:* First order statistics of a normal distribution with mean and standard deviation fit to individual ant data on collective towing.

measurements. The steady-state speed of the load during each segment was estimated by dividing the total transport path length during the segment by the duration of the segment. For all segments with the same transport team size, the average and standard deviation of the load speed estimates were calculated. Figure 6.4 plots these statistics versus the team size. The plot shows a significant decrease in steady-state load speed with increasing team size, even though each ant was pulling the same amount of weight. In [17], the average pulling force of a single *N. cockerelli* ant on an elastic load was measured to be 10.5 mN; therefore, one ant would likely have been capable of overcoming static friction and pulling the heaviest load in the experimental trials (0.12 g, with a weight of 1.2 mN). Hence, the observed decrease in speed indicates that a transport team is less powerful than the sum of its members' efforts.

6.3 Two Models of Ant Behaviors and the Resulting Payload Speed

6.3.1 Ants with Continuous, Adaptive Pulling Forces

Figure 6.4 shows a large variance in the steady-state speed of a load that is towed by a single ant. This suggests that an ant transport team can be modeled as a heterogeneous group whose members are capable of different maximum speeds and pulling forces. Such a team would be able to cooperatively tow the load only as fast as the speed of its slowest member, requiring coordination among the teammates.

In this model, it is assumed that an ant's maximum towing speed is distributed according to a Gaussian probability density function (pdf) $\phi(v)$ with corresponding cumulative density function (cdf) $\Phi(v)$ (Appendix A). Under this assumption, a Gaussian distribution was fitted to the data in Figure 6.4 on the steady-state towing speed of a single ant ($\mu = 0.7$ cm/s, $\sigma = 0.36$ cm/s). The expected speed of the slowest member of a transport team with N members was then computed using order statistics [41]. The expected first moment of the r^{th} order statistic from n samples of the distribution $\phi(v)$, where $0 < r \leq n$, is given by:

$$E(r,n) = \frac{n!}{(r-1)!(n-1)!} \int_{-\infty}^{\infty} v[1-\Phi(v)]^{n-r}\Phi(v)^{r-1}\phi(v)dv. \tag{6.1}$$

The variance of the r^{th} order statistic is:

$$Var(r,n) = \int_{-\infty}^{\infty} (v - E(r,n))^2[1-\Phi(v)]^{n-r}\Phi(v)^{r-1}\phi(v)dv. \tag{6.2}$$

The speed of the slowest member of a team is distributed according to the first order statistic ($r = 1$), for which Equation 6.1 and Equation 6.2 simplify to:

$$E(1, n) = n \int_{-\infty}^{\infty} v[1 - \Phi(v)]^{n-1} \phi(v) dv, \tag{6.3}$$

$$Var(1, n) = \int_{-\infty}^{\infty} (v - E(1, n))^2 [1 - \Phi(v)]^{n-1} \phi(v) dv. \tag{6.4}$$

To enforce realistic constraints on the range of possible ant speeds, the support of $\phi(v)$ was truncated to $[0, v_{max}]$, where $v_{max} = 1.2$ cm/s is slightly higher than the fastest towing speed measured during experiments with a single ant. The integrals Equation 6.3 and Equation 6.4 were evaluated over the range of $v \in [0, v_{max}]$, after being renormalized to have a total probability of 1, for team sizes $n = 2, 3, 4$. The computed values of $E(1, n) \pm Var(1, n)^{1/2}$ for $n = 2, 3, 4$ are plotted alongside the statistics of the ant data in Figure 6.4. The figure shows that the average and standard deviation of the measured towing speed for each team size are very close to those of the corresponding first order statistics. This similarity supports the hypothesis that ant transport teams move at the speed of the slowest teammate, which implies that the ants can coordinate transport in a decentralized fashion without explicit communication, information about the payload, or knowledge of the team size and configuration.

6.3.2 *Ants with Intermittent, Constant Pulling Forces*

Besides the ants' variability in speed, their unsynchronized gaits during transport could have contributed to the decrease in load speed with increasing team size. The ants in a transport team were observed to step backward while towing the load, and their out-of-phase stepping could have coincided with their application of intermittent, uncoordinated forces on the load. In this section, a dynamical model is developed to investigate the effect of this hypothetical behavior on the steady-state transport speed.

Collective transport behaviors have previously been studied with the Kilobot robotic platform [119] and the Bristlebot, Hexapod, and μTug platforms [35]. Rubenstein et al. [119] found that the transport speed remains the same regardless of team size when the payload mass per robot is kept constant. However, this paper did not take into account out-of-phase stepping by the robots. Christensen et al. [35] investigated the effect of uncoordinated stepping on effective force per robot during group towing. They found no effect for the case where the robots' gait is non-impulsive; e.g., the case where each robot's contact time with the ground, during which it exerts a pulling force on the load, is not extremely short compared to the stride period of its gait. This section combines the dynamic model presented in [119] with the gait consideration discussed in [35] to predict the effect of out-of-phase stepping on the steady-state transport speed.

During the experiments, the ants transported the load along an approximately straight path and produced very little load rotation. Hence, the load dynamics can be modeled as translation in the plane using Newton's second law of motion:

$$\sum_{i=1}^{N} \mathbf{F}_i + \mathbf{F}_f = m\mathbf{a}, \tag{6.5}$$

where $\mathbf{F}_i \in \mathbb{R}^2$ is the force applied by ant i, $\mathbf{F}_f \in \mathbb{R}^2$ is the kinetic frictional force on the load, m is the load's mass, and $\mathbf{a} \in \mathbb{R}^2$ is the load's acceleration. Using an ideal motor assumption to relate force to velocity, the force applied by ant i can be modeled by the linear relation

$$\mathbf{F}_i = K(v_{max} - \mathbf{v}_i \cdot \hat{\mathbf{x}})\hat{\mathbf{x}}, \tag{6.6}$$

where $K > 0$ is a constant gain, v_{max} is the maximum ant speed under no load, $\mathbf{v}_i \in \mathbb{R}^2$ is the velocity of ant i, and $\hat{\mathbf{x}} \in \mathbb{R}^2$ is the direction of the load transport. Under the assumption that the load is in static equilibrium in the vertical direction, the frictional force is given by,

$$\mathbf{F}_f = -\mu_k m g \hat{\mathbf{x}}, \tag{6.7}$$

where μ_k is the coefficient of kinetic friction of the load on the ground, and g is the acceleration due to gravity.

In this model, all ants move at the same speed. Since the ants pull on strings to tow the load, they are rigidly attached to the load when applying force (e.g., the ants may not move faster than the load). This model assumes that a transporting ant will not accumulate enough slack in its string such that its pulling effort does not affect the payload. During transport, the load stops immediately when the ants stop pulling it. This allows for the use of the quasi-static motion assumption [110, 135], implying that the load velocity $\mathbf{v}_L \in \mathbb{R}^2$ is in the same direction as the net force applied by the ants. These assumptions simplify Equation 6.6 to,

$$\sum_{i=1}^{N} \mathbf{F}_i = NK(v_{max} - \|\mathbf{v}_L\|)\hat{\mathbf{x}}. \tag{6.8}$$

Solving Equation 6.5 for $\|\mathbf{v}_L\|$ at steady state ($\mathbf{a} = \mathbf{0}$) yields,

$$\|\mathbf{v}_L\|_{ss} = v_{max} - \frac{\mu_k m g}{NK}. \tag{6.9}$$

To include the effect of asynchronous ant gaits, the model can incorporate a probability $p_f = t_c/t_s$ that an ant applies a pulling force at any given time, where t_c is its contact time with the ground and $t_s > t_c$ is the period of its gait, as defined in [35].

In the case where N ants pull in parallel with their steps beginning at independent, uniformly random drawn starting times between 0 and t_s, the number of transporters $n \leq N$ that apply force at the same time is described by the binomial distribution, $n \sim B(N, p_f)$. Then, the average total force exerted by the ants on the load is given by:

$$\sum_{i=1}^{N} \mathbf{F}_i = p_f N K (v_{max} - \|\mathbf{v}_L\|)\hat{\mathbf{x}}. \tag{6.10}$$

Inserting this total applied force into Equation 6.5 and solving for the steady-state load velocity results in an equation similar to Equation 6.9:

$$\|\mathbf{v}_L\|_{ss} = v_{max} - \frac{\mu_k m g}{p_f N K}. \tag{6.11}$$

What is important to note in Equation 6.11 is that the term producing a slowing effect, $\frac{\mu_k m g}{p_f N K}$, includes the load mass m in the numerator and the team size N in the denominator. Therefore, if the ratio m/N of the load mass to the team size is kept constant, as was done in the ant experiments, then the steady-state load speed should remain the same regardless of the team size, as observed in [119]. This contradicts the observed trend in load speed from the ant experiments.

6.4 Design of Robot Controllers for Adaptive, Continuous Pulling

One possible method for a heterogeneous transport team to adjust to the speed of its slowest member is through learning based on *stigmergy*, a mechanism by which agents communicate indirectly through modifications to their environment (in this case, the payload). As stated previously, it is assumed that the ants all know the direction to their nest and that they tow the load in this direction along a straight line. Under these assumptions, the ants' configuration on the load during the experiments (see Figure 6.1) would have resulted in a net zero moment on the load if all ants pulled with identical forces. Any disparities in towing force would produce differences in the ants' maximum possible steady-state towing speeds as well as rotation of the transported load. However, significant rotation of the payload was not observed in the ant experiments (e.g., the load oscillated slowly but did not rotate indicating one ant was moving faster than the others).

Ants could have implicitly communicated their differences in towing speed by measuring the collective effect of these differences on the load's orientation and rotational dynamics during transport. To illustrate, Figure 6.5 shows a cooperative towing scenario with two robots. Robot R_1 is moving faster than robot R_2, causing the load to rotate in the clockwise direction. If robot R_2 were moving faster than robot R_1, then the load would rotate in the counterclockwise direction. By making an association between the load's direction of rotation and the disparity in robot speeds, and by measuring its own pulling force, each robot should be able to learn the conditions under which it should speed up or slow down in order to move at the same speed as the rest of the transport team. These conditions do not require the robot to know its location on the load. When there is an incentive for each robot to move as fast as possible, the competing objectives of preventing load rotation and transporting the load quickly will cause the speeds of the faster team members to oscillate around the speed of the slowest member when it is moving at its maximum speed.

A real-time reinforcement learning algorithm was developed to implement this behavior on robots. The algorithm drives a team of robots with heterogeneous maximum speeds to transport a load at the speed of its slowest member. Each robot runs the two-layer neural network shown in Figure 6.6. There is no distinction between a learning phase and an exploitation phase; instead, an ϵ-greedy algorithm is chosen to adapt to the unpredictable and changing interaction forces during transport. This is done to allow learning errors that can be made during the start of the transport to

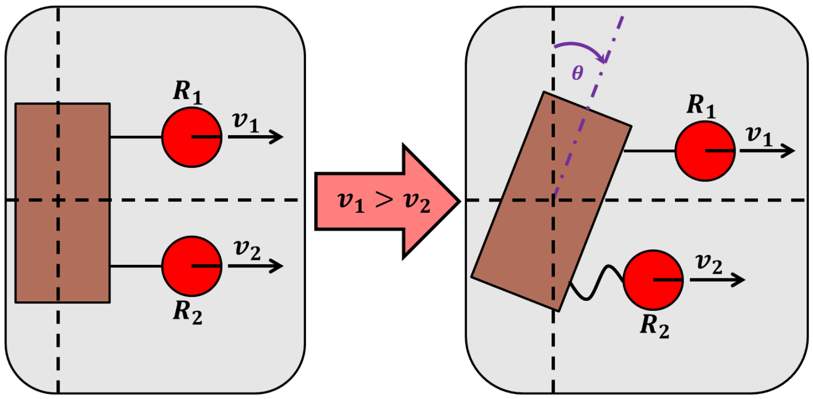


Figure 6.5. A two-robot towing scenario. Robot R_1 is moving faster than robot R_2, causing the load to rotate by angle θ and the towing string of robot R_2 to go slack.

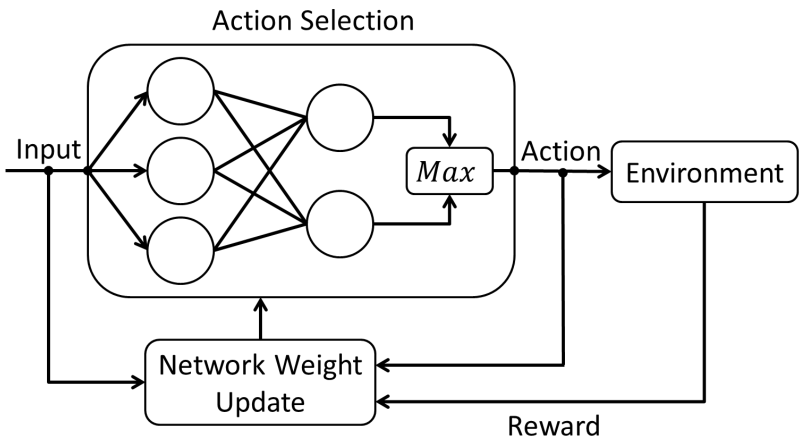


Figure 6.6. Block diagram of the neural network.

be corrected .

For this application, the goal is for the robots to learn the association between the direction of the load's rotation and the necessary adjustments to the individual speeds of the transporters. This learning is done at discrete times, where, for simplicity, the times are defined in unit increments. The input vector to the neural network of robot k at time t, $\mathbf{p}_k(t) = [p_{k,1}\ p_{k,2}\ p_{k,3}\ p_{k,4}]^T = [sign(\theta)\ sign(\dot{\theta})\ sign(\ddot{\theta})\ sign(\dot{v}_k(t))]^T$, contains the directions of the load's orientation θ, angular velocity $\dot{\theta}$, and angular acceleration $\ddot{\theta}$, as well as the direction of the change in robot's velocity $\dot{v}_k(t)$ at time

t. The output vector of the neural network of robot k is defined as $\mathbf{a}_k = [a_{k,1} \ \ a_{k,2}]^T = [v_{k+} \ \ v_{k-}]^T$, whose entries indicate a binary decision for the robot to speed up or slow down. Each robot k computes the value $a_{k,j}(t)$ of its output neuron $j \in \{1, 2\}$ at time t as:

$$a_{k,j}(t) = \sum_{i=1}^{4} w_{k,ij}(t) p_{k,i}(t), \tag{6.12}$$

where $w_{k,ij}(t)$ is the weight value of the connection between the robot's i^{th} input neuron and j^{th} output neuron.

An ϵ-greedy algorithm is applied to determine the action taken by the robot. With probability ϵ, the robot randomly chooses to speed up or slow down without using the neural network to make a decision. Otherwise, the output neuron with the largest value is chosen as the winner. If $v_{k+} \geq v_{k-}$, then robot k speeds up; otherwise, it slows down, according to the following controller:

$$\dot{v}_k = \begin{cases} K_a \frac{|\theta|}{\theta_{max}} + K_v (1 - \frac{v_k}{v_{k,max}}), & v_{k+} \geq v_{k-} \ \text{ or } \ F_k = 0 \\ -K_a \frac{|\theta|}{\theta_{max}} - K_v (1 - \frac{v_k}{v_{k,max}}), & \text{otherwise,} \end{cases} \tag{6.13}$$

where K_a and K_v are constant gains, $\theta_{max} = \pi$, $v_{k,max}$ is the robot's maximum possible towing speed, and F_k is the magnitude of the pulling force applied by the robot. When the robot measures $F_k = 0$ during some time period, it speeds up but does not update its weights $w_{k,ij}$ ("learn") during that period.

This velocity controller is used to avoid identical decisions by the robots to speed up or slow down by a constant value. If robots were allowed to change their velocities by the same amount, they would sometimes get stuck in loops of repeatedly making the same decision as their teammates. In this case, the differences between the robots' velocities would not change, and nothing useful would be learned about their latest actions (e.g., identical actions by robots would not alter their difference in speeds).

The proposed velocity controller enables all robots in the team to make the same decisions while still learning. Equation 6.13 drives inherently faster robots with higher maximum speeds to be more agile than slower robots, since the quantity $(1 - \frac{v_k}{v_{k,max}})$ is larger for faster robots for a given robot speed v_k, producing a higher acceleration. The controller also makes larger adjustments to the robots' accelerations when the payload is far from its initial orientation. The gains K_a and K_v weight the competing objectives of maintaining the load in its original orientation and moving the load as fast as possible.

After taking an action of speeding up or slowing down, each robot computes a reward that is based on the resulting change in the load's orientation. The reward function at time t for robot k is defined as,

$$E_k(t) = \begin{cases} \frac{\theta_{max}-|\theta|}{\theta_{max}} + \frac{v_k}{v_{k,max}}, & sign(\theta\dot{\theta}) <= 0, \\ -|\frac{\theta}{\theta_{max}}|, & \text{otherwise.} \end{cases} \tag{6.14}$$

The difference in reward values $E_k(t-1)$ and $E_k(t)$ at times $t-1$ and t, respectively, determines the adjustments to the neural network weights according to the Instar Rule [42]. Reward constants $r_{k,j}(t)$ are defined at time t such that only the output neuron associated with the chosen action is rewarded or penalized:

$$r_{k,j}(t) = \begin{cases} 1, & E_k(t) - E_k(t-1) \geq \tau \text{ and } a_{k,j} > a_{k,l},\ l \neq j \\ -1, & E_k(t) - E_k(t-1) \leq -\tau \text{ and } a_{k,j} > a_{k,l},\ l \neq j \\ 0, & \text{otherwise,} \end{cases} \tag{6.15}$$

where τ is a threshold value chosen to differentiat between sensor noise and a signifi-

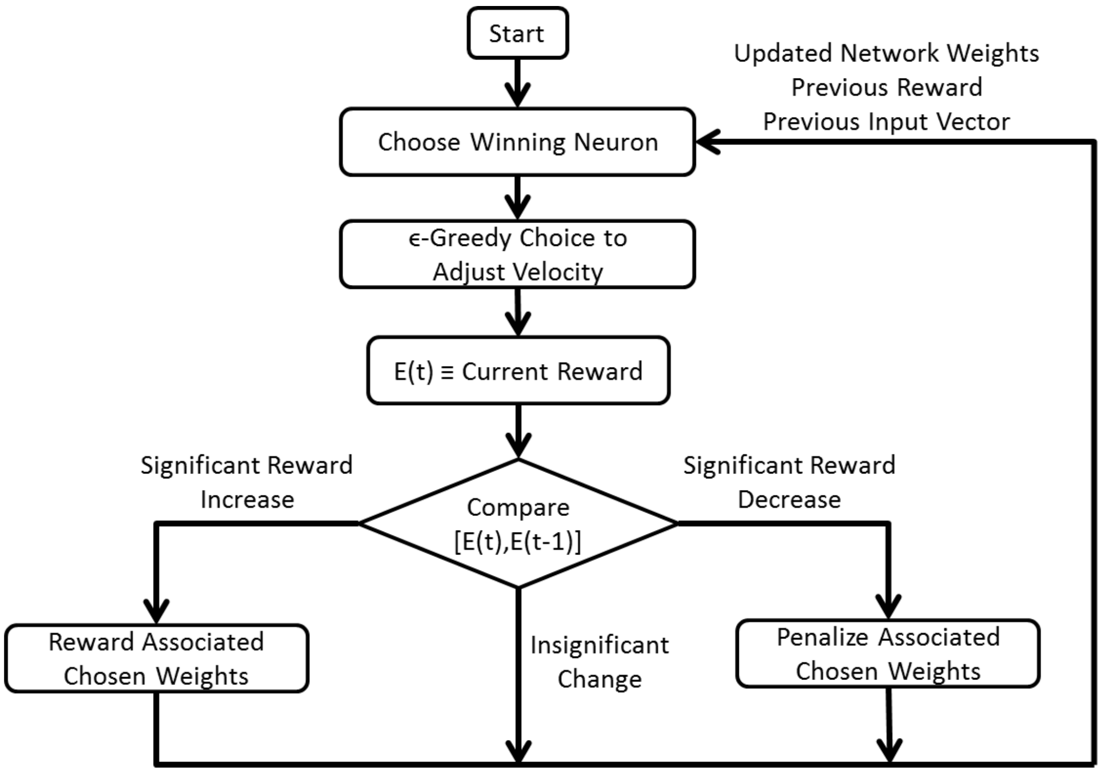

Start
Updated Network Weights
Previous Reward
Previous Input Vector
Choose Winning Neuron
ϵ-Greedy Choice to Adjust Velocity
E(t) ≡ Current Reward
Significant Reward Increase
Significant Reward Decrease
Compare [E(t),E(t-1)]
Reward Associated Chosen Weights
Insignificant Change
Penalize Associated Chosen Weights

Figure 6.7. Flowchart of an individual robot's decision process during transport.

cant reward change. Then the weights are updated as follows:

$$w_{k,ij}(t) = (1 - \gamma r_{k,j}(t))w_{k,ij}(t-1) + \alpha r_{k,j}(t)p_{k,i}(t), \tag{6.16}$$

where $\gamma \in [0,1]$ is a rate of forgetting and $\alpha \in [0,1]$ is a rate of learning. These rates are chosen to bound the maximum weight values of the update matrix to,

$$w_{k,ij}^{max} = \frac{\alpha}{\gamma} \tag{6.17}$$

After updating the weights $w_{k,ij}$, the process is repeated. A flowchart of the robot behavior is shown in Figure 6.7.

6.5 Multi-Robot Experimental Results

6.5.1 Continuous, Adaptive Pulling

This learning algorithm described in Section 6.4 was implemented on teams of 2 to 4 Pheeno robots to investigate whether learning through stigmergic feedback could cause a transport team to tow the payload at the speed of the slowest member. In these experiments, the mass of the load was maintained at 500 g, which a single robot is capable of towing. This was done to ensure that the decrease in payload speed could be attributed entirely to the learning algorithm, not to the experimental setup. To imitate the ants' directionality of transport toward the nest, all the robots were assigned to drive in the same direction. The robots changed their speed according to the learning algorithm at a rate of 0.5 Hz. Each robot was programmed with a probability $\epsilon = 0.2$ of choosing a random action, a forgetting rate $\gamma = 0.02$, a learning rate $\alpha = 0.1$, a significance threshold value $\tau = 0.5$, and controller gains $K_a = 1$ and $K_v = 0.8$. Ten trials were run for each team size. Teams of 2, 3, and 4 robots consisted of members with maximum speeds of $[4, 12]$ cm/s , $[4, 8, 12]$ cm/s, and $[4, 6, 9, 12]$ cm/s, respectively. The locations of the robots on the load were chosen randomly for each trial.

The load and robot velocities during the experiments are plotted in Figure 6.8. These results show that the robots adjust their speed to the speed of the slowest transporter in the team. There are slight discrepancies between the robots' reference velocities and their actual velocities, which were measured from the robot locations tracked by the overhead camera. These differences were caused by camera error and tracking tag placement error which, combined with controller action that drives the robots in the same direction, causes rotational corrections by the robots to be detected as additional linear velocity.

6.5.2 Intermittent, Constant Pulling

To simulate out-of-phase stepping during transport as described in Section 6.3.2, towing experiments with 2-4 Pheeno robots were run in which the robots' motors were periodically turned on and off to mimic the contact and swing phases of an ant's gait. The robots' motors were turned on for 1.5 s with a total step time of 3 s. The robots' start times for each step were randomly drawn from a uniform distribution. Each robot "steps" with the same maximum velocity. The load mass was scaled with the number of robots in the team, with a constant mass of 500 g per robot. Ten trials were run for each team size.

Figure 6.9 shows that the average steady-state payload speed during these experiments matches the value predicted by the model Equation 6.11. Thus, it can be concluded that out-of-phase stepping is not the primary cause of the decrease in steady-state transport speed with respect to team size that is observed in *N. cockerelli*.

In addition, out-of-phase stepping was tested on teams of Pheeno robots with different maximum speeds. Under these conditions, the robot teams exhibited uncoordinated transport in which the load was pulled into the slowest robot, which was then dragged along by the efforts of the other robots. This type of behavior was never observed during the ant towing experiments. Wheel slip ultimately stalled the robots or made them uncontrollable and prevented further transport of the load.

6.6 Discussion

The statistical analysis in Section 6.3.1 of data on collective towing by *Novomessor cockerelli* ants and the results of the multi-robot towing experiments described in Section 6.5.1 support the conclusion that ants adjust their speeds during collective transport to accommodate the slowest member of the transport team. Hence, robotic implementation of ant-like transport strategies could enable adaptive, decentralized transport by teams of robots with heterogeneous capabilities, manufacturing variations, differences in charge state, and other dissimilar properties.

Previous studies have addressed other scenarios where group efficiency is a sublinear function of group size. Krieger et al. found that in a group of robots mimicking an ant foraging behavior, the contribution of each individual to the foraging process decreased as the group size increased due to more time spent avoiding collisions [76]. Lerman et al. also discovered this sublinear relationship in foraging behaviors and predicted an optimal group size for foraging before interference among individuals begins to lower the net group performance [83]. In contrast to these group foraging examples, the sublinear relationship between transport speed and team size that has been investigated here appears to be caused by heterogeneity of teammate capabilities rather than interference arising from uncoordinated individual behaviors (e.g., out-of-phase stepping, unaligned pulling angles, or avoidance of fellow transporters).

The differences in individual *N. cockerelli* speeds could be caused by physical variations among ants that are related to age, fatigue, or general fitness. It is also possible that the differences are behavioral, as observed in other species [69]. These behavioral disparities may arise from varying genotypes within the colony that resist the spread of disease, environmental differences which can naturally cause developmental or learned differences, or social differences which cause discrepancies in behavior through interactions within the shared living environment [70]. Previous work has shown that colonies with a mixture of aggressive and passive members are more fit than colonies comprised of entirely one behavior type. This is because a particular behavior type can be more productive in some scenarios than others: for instance, aggressive members benefit in confrontational settings (e.g., foraging and defending the colony), but their behavior is a detriment when they interact in social settings (e.g., hiding from predators and brood care) [126]. Differences in speed among individual *N. cockerelli* ants could be a consequence of a similar behavior dichotomy.

Cooperation among ants allows colonies to harvest large food items, thus requiring less time and energy to be spent searching for individually transportable food items [30]. Other ant species, such as *Eciton burchellii* (army ants), exhibit collective transport that is *superefficient* in that groups can lift more weight per ant than solitary transporters [38, 55]. Unlike *E. burchellii*, *N. cockerelli* forage in low-density groups and are monomorphic, i.e., they do not have major variations in physiology. *N. cockerelli* also must compete with ants from the *Forelius* genus that use mass-raiding strategies to acquire food items. This competition may not afford *N. cockerelli* the opportunity to wait for the fastest foragers to retrieve a food item before it is claimed by other ants.

A heterogeneous transport team that adapts to its slowest member will move the load more slowly as the team size increases; however, this strategy also has various benefits. If this adaptation did not occur, the fastest individuals would need to overwhelm the efforts of the slowest members and handle a heavier portion of the load, possibly dragging the slower teammates during the transport. By accommodating the slowest member, large transport teams can utilize the strength of all teammates, not just the fastest ones, and thus can transport heavier loads. Another advantage may arise when the load must be transported over different types of terrain. A large transport team could apply a high net force to a load in order to pull it up a hill or over an obstacle. The presence of slow members in the team could potentially stabilize this maneuver, since from a control-theoretic perspective, less aggressive systems are easier to control. Thus, an ant-like strategy could potentially increase the robustness of the transport to changing, unanticipated environmental conditions.

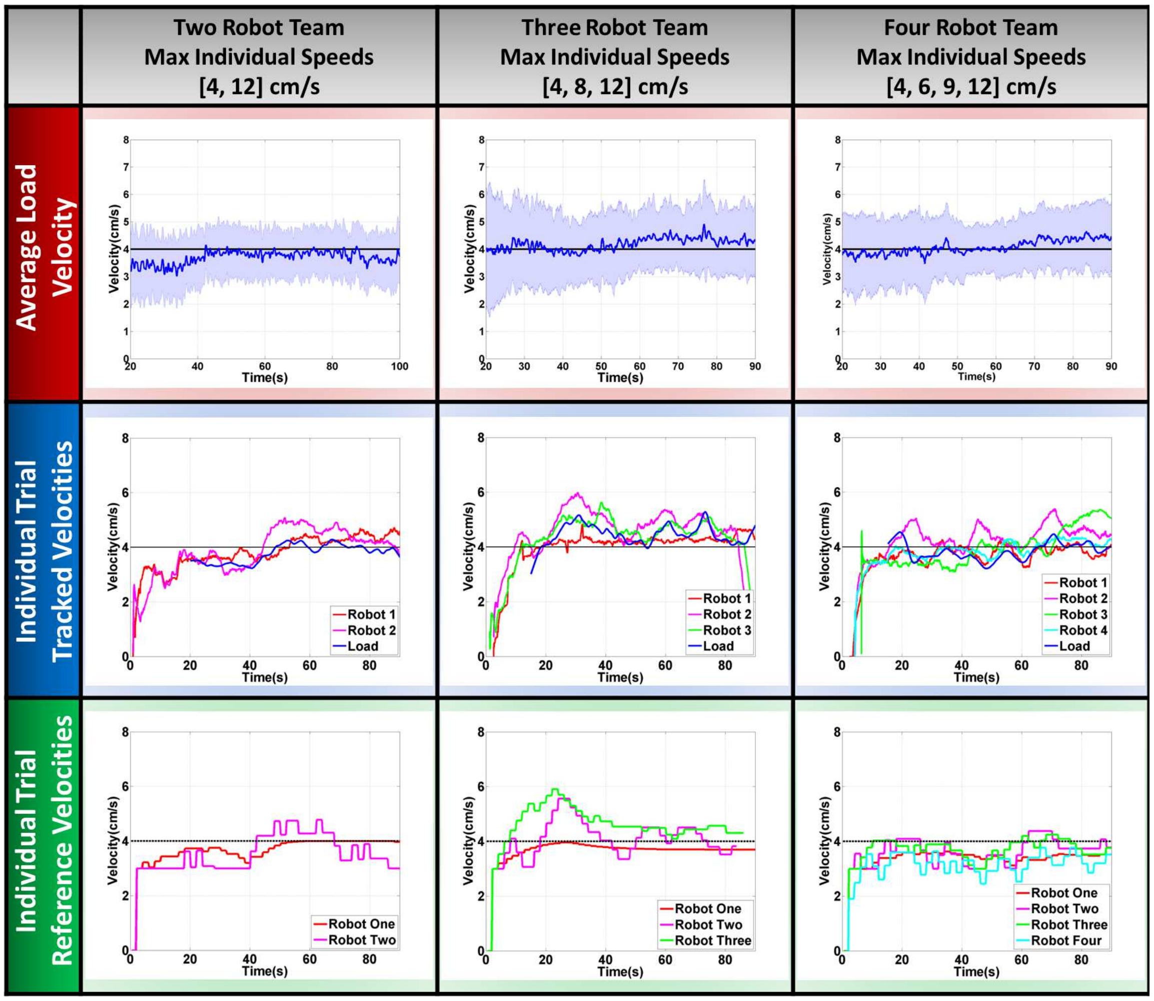


Figure 6.8. Results of towing experiments with the reinforcement learning algorithm. The plots display the time evolution of the load and robot velocities for each team size. In all plots, the black line is the maximum reference speed of the slowest team member. The first row shows the average load velocity across ten trials, with the solid blue line and shaded area representing the mean and standard deviation, respectively. The second row shows the measured velocities of the robots and the load during a single experimental trial, and the third row shows the corresponding reference velocity that was calculated by each robot using the learning algorithm.

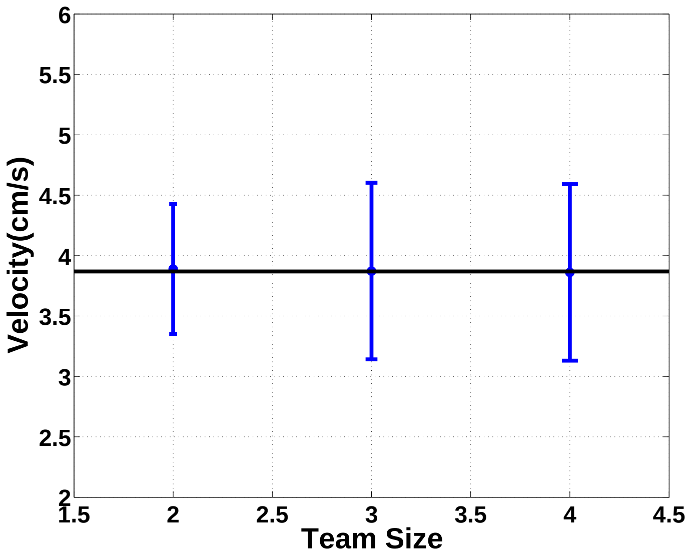


Figure 6.9. Results of towing experiments with out-of-phase robot stepping. Blue dots with error bars are the mean ± standard deviation of the average speed of the payload across ten trials. The black line is the prediction of model Equation 6.11 of the average steady-state transport speed.

Chapter 7

CONCLUSION AND FUTURE WORK

This thesis has presented approaches to designing controllers for stochastic boundary coverage and collective payload transport by robotic swarms that respect the constraints on such systems, outlined in Chapter 1. Chapter 4 described a decentralized stochastic strategy to solve a boundary coverage problem, and Chapter 5 extended this strategy to solve a collective transport problem in a way that reproduces the previously modeled dynamics of group retrieval observed in *Novomessor cockerelli* ants. Finally, Chapter 6 develops a real-time reinforcement learning-based controller that experimentally replicates the observed payload speed drop-off with respect to ant transport team size that was observed in colonies of *N. cockerelli.* To validate these swarm robotic control strategies and others, a low-cost, open-source, customizable robotic platform *Pheeno* was designed and tested, as described in Chapter 3. The following sections describe conclusions and possible future research directions that pertain to the work in Chapter 3 to Chapter 6.

7.1 Coverage Tasks for Robotic Swarms

Chapter 4 presents a rigorous methodology for designing control policies that distribute a swarm of robots among a set of boundaries. The control policies rely only on local information obtained by the robots and have probabilistic guarantees on steady-state performance. This work investigates the effect of robot interactions on the actual steady-state allocations that are achieved with the use of control policies derived from an abstraction of the swarm population. The simulation results illustrate that the actual system behavior follows predictable and controllable trends when robot interactions at the boundaries and in the free space are introduced. In this way, a foundation is built for further work on encounter-based swarm robotic allocation that obviates an analytical characterization of encounter rates.

In future work, these investigations will be repeated for more general boundary shapes and for time-dependent boundaries. Analysis is also needed to gain a better understanding of the relationship between actual allocation ratio and the mean space between robots. This control approach can be extended to other scenarios in which robotic swarms must allocate among both static and dynamically moving regions, such as surveillance and target-tracking applications. Finally, this control scheme will be experimentally validated on a physical multi-robot testbed.

7.2 Collective Transport Tasks for Robotic Swarms

Chapter 5 extends and applies work presented in Chapter 4 on stochastic allocation of robotic swarms around boundaries to mimic collective-transport data obtained from *Novomessor cockerelli* ant teams [78]. This design framework allows for controlling the equilibrium demographics of multi-robot transport teams (EPMM) as well as how fast they converge to this equilibrium (TMM). Although the presented scenario focuses on two-sided loads, the approach could be used for achieving other robot distributions that can be approximated by further dividing a load into different regions with region-specific desired allocations. Alternatively, this design approach also allows for equilibrium robot fluxes to be controlled (RMM) instead of macroscopic demographic properties. In all three cases, the equilibrium allocations of robots are robust to environmentally-induced changes to the encounter rates, including changes in swarm size and number of loads. This robustness allows the development of a calibration method that provides a novel way to estimate robot encounter rates without using geometrical parameters or measurements of times between events across an ensemble of robots.

One possible criticism of the current approach is that the attachment-detachment process never “turns off.” Even after reaching equilibrium team demographics, free robots continue to attach and detach at random. A feature of this approach is that the system continues to adapt to environmental changes. For instance, if more loads are added, then teams will continue to form around those loads. However, in scenarios where this is not a concern, it seems more efficient for the system to detect when the desired allocation has been reached and transition to another system-level mode (e.g., from “allocation phase” to “transport phase”). Chemical reaction networks (CRNs) have already been constructed to implement oscillators [81, 129], counters [128, 129], and algorithms which are sensitive to the macroscopic state of the system [129]. Therefore, it may be possible to augment this approach to either detect a time when the system has likely converged or to directly detect the equilibrium condition and transition to a more efficient transport behavior. In fact, there are existing CRNs implemented with irreversible reactions that achieve distributed consensus [34, 125]. These CRNs are able to detect which chemical species is in the majority and switch all entities in the mixture to that species. Because they are implemented with irreversible reactions, the equilibrium of each trial is a true fixed point of the system (i.e., it is not a thermodynamic equilibrium that fluctuates around a mean). Although the presented method is built using intuition from reversible processes that reach dynamic equilibria, the underlying implementation presented here uses irreversible reactions. Consequently, a future direction could similarly leverage these irreversible reactions to reach truly static equilibria similar to the consensus cases.

In future work, these methods can be extended to control two-dimensional motion of non-circular loads that can rotate. Whereas this study focused on the allocation dynamics of robots around mobile loads, future studies will focus on other behaviors that are needed during multi-robot transport. A particularly interesting case is decentralized multi-robot collision avoidance for multiple transport teams that are surrounded by free robots. It would also be useful to develop analytically tractable models of systems with attachment-detachment probabilities that depend on the speed of the load. Using speed-dependent probabilities, it may be possible to replace the proportional velocity controller used in this paper with a simpler constant-force behavior. Such an implementation may be more suitable for microrobots and nanorobots that derive their applied forces via externally generated fields. Finally, experimental validation of these transport control strategies will be implemented on a physical multi-robot testbed to eliminate any numerical artifacts of imperfect simulation tools.

7.3 Biomimetic Swarm Control Strategies for Collective Transport

Chapter 6 investigated the observation that the transport speed of a load towed by several *Novomessor cockerelli* ants decreases as a function of the team size, even with the same load mass per ant. A control approach in which a homogeneous team of robots pull on the load with intermittent, constant forces was tested as a possible ant towing behavior. A dynamical model was used to predict the steady-state speed of a load that is transported in this manner by a team of known size. The predicted load speed was independent of team size, as long as the mass per transporter remained constant, and the prediction was verified through experiments with teams of 2, 3, and 4 Pheeno robots. Using order statistics, it was found that the load speed decrease can be attributed to the heterogeneous abilities of individual ants. Since ants within a given colony may move at a range of possible speeds, as a result of differences in age, energy, ability to orient during transport, and other factors, it was hypothesized that an ant transport team must move at the speed of the slowest member for successful load transport. Due to their biological limitations, the ants would need to identify the slowest member without explicit communication or global information. To implement a multi-robot towing strategy with these constraints, a real-time reinforcement learning algorithm was developed that relies on implicit communication through the load and local measurements by each robot. This algorithm was tested on teams of 2 to 4 Pheeno robots with significantly different individual maximum speeds. The experimental results supported the hypothesis that the ants are towing the load at the speed of the slowest team member.

The experiments on collective transport described in this thesis have focused on one-dimensional payload transport by both ants and robots in flat environments with no obstacles. In the future, experiments and analyses should be conducted for transport along two-dimensional trajectories through more complex environments. To more closely emulate the ant behaviors, multi-robot experiments can be performed using legged robots that are closer in design to the ants' anatomy. In addition, a rigorous analysis of the reinforcement learning algorithm presented in this work is needed to characterize the existence and stability of equilibrium payload speeds. This analysis would provide theoretical guarantees on the transport dynamics, and thus more confidence in the algorithm's effectiveness in a wide range of scenarios.

7.4 Validation of Swarm Control Strategies with the Pheeno Robotic Platform

Pheeno is a new mobile robot platform that is designed to be accessible to students for educational use, while still incorporating sensing and manipulation capabilities that are sophisticated enough for multi-robot research experiments. The robot's modular design allows users to develop custom attachments that suit their specific applications. Chapter 3 focuses on the design of the core robot module and a gripper module that enables Pheeno to manipulate objects, either individually or in groups. The robot's capabilities were demonstrated with proof-of-concept experiments on

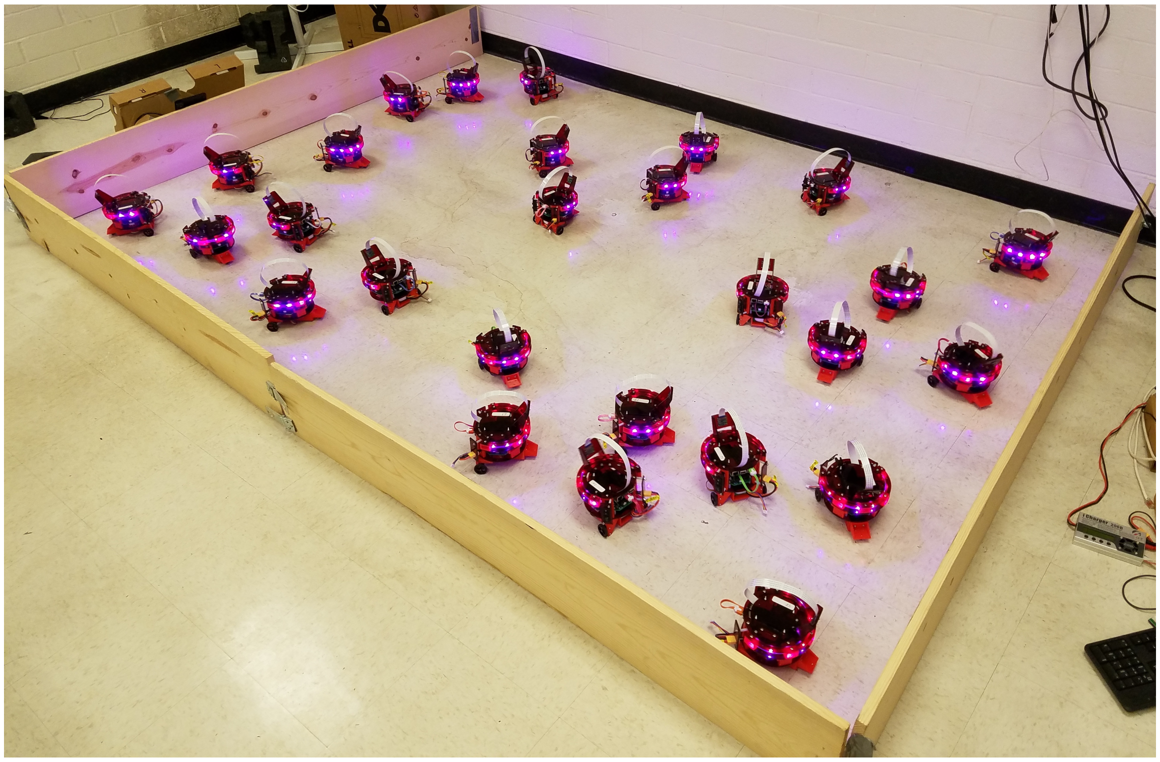


Figure 7.1. Group of 30 Pheeno robots operating concurrently.

trajectory tracking, image processing, and collective transport, and the robot was used to validate new strategies for confinement control, topological mapping, and collective transport based on sliding mode control.

In the future, Pheeno will be used for larger-scale swarm robotic experiments on performing mapping, coverage, manipulation, construction, and excavation tasks. Currently, 30 Pheeno platforms have been fabricated and deployed at the same time (Figure 7.1). The Pheeno fabrication process must be optimized in the future for efficient mass production of the robot. Although at this time the Pheeno robots must each be programmed individually, work is currently in progress on programming a large group of Pheenos over Wi-Fi or Bluetooth. Communication-based strategies can currently be emulated through a central router. To test these strategies in more realistic scenarios (outside of a lab), local wireless communication will be implemented between robots without the use of a central router. For experiments that last longer than Pheeno's battery life, autonomous recharging capabilities will be developed for the robot. In addition, new modules are currently being designed for Pheeno to enable its operation in different types of environments.

REFERENCES

[1] Arduino Pro Mini. URL `https://www.arduino.cc/en/Main/ArduinoBoardProMini`. Accessed: 2015-01-12.

[2] Gizmoszone. URL `http://www.gizmoszone.com`. Accessed: 2015-01-12.

[3] ICRA 2016 trailer theme: ducks in the loop & duck-robot interaction. URL `http://trailer.icra2016.org`. Accessed: 2015-08-30.

[4] Pheeno Robot GitHub Repository. URL `https://github.com/ACSLab/PheenoRobot`. Accessed: 2015-01-12.

[5] Raspberry Pi 2 Model B. URL `https://www.raspberrypi.org/products/raspberry-pi-2-model-b/`. Accessed: 2015-01-12.

[6] Pololu Robotics & Electronics. URL `https://www.pololu.com/`. Accessed: 2015-01-12.

[7] ServoCity. URL `https://www.servocity.com/`. Accessed: 2015-01-12.

[8] Teensy usb development board. URL `https://www.pjrc.com/teensy/teensy31.html`. Accessed: 2017-03-11.

[9] Scribbler 2, 2012. URL `https://www.parallax.com/product/28136`. Accessed: 2016-01-12.

[10] A Roadmap for U.S. Robotics: From Internet to Robotics, 2013 Edition, 2013. URL `http://robotics-vo.us/sites/default/files/2013%20Robotics%20Roadmap-rs.pdf`. Sponsored by Robotics Virtual Organization (Robotics VO).

[11] A Roadmap for U.S. Robotics: From Internet to Robotics, 2016 Edition, 2016. URL `http://www.robotics-vo.us/sites/default/files/A%20Roadmap%20for%20US%20Robotics%20From%20Internet%20to%20Robotics%202016%20Edition%2011.07.2016%20.pdf`. Sponsored by Robotics Virtual Organization (Robotics VO).

[12] Iman Anvari. Non-holonomic differential drive mobile robot control & design: Critical dynamics and coupling constraints, 2013.

[13] D. Baum. *Definitive Guide to Lego Mindstorms*. Apress, 175 Fifth Avenue, New York, NY, 10010, 2nd edition, Nov. 2002. ISBN 1590590635.

[14] Aaron Becker, Golnaz Habibi, Justin Werfel, Michael Rubenstein, and J. McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013. IEEE.

[15] Spring Berman, Ádám Halász, M. Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Trans. Robot.*, 25(4):927–937, August 2009. doi: 10.1109/TRO.2009.2024997.

[16] Spring Berman, Vijay Kumar, and Radhika Nagpal. Design of control policies for spatially inhomogeneous robot swarms with application to commercial pollination. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9–13, 2011.

[17] Spring Berman, Quentin Lindsey, Mahmut Selman Sakar, Vijay Kumar, and Stephen C. Pratt. Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proc. IEEE*, 99(9):1470–1481, September 2011. doi: 10.1109/JPROC.2011.2111450.

[18] Spring Berman, Radhika Nagpal, and Ádám Halász. Optimization of stochastic strategies for spatially inhomogeneous robot swarms: a case study in commercial pollination. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3930, San Francisco, CA, USA, September 25–30, 2011. doi: 10.1109/IROS.2011.6094771.

[19] Joseph Betthauser, Daniel Benavides, Jeff Schornick, Neal O'Hara, Jatin Patel, Jeremy Cole, and Edgar Lobaton. WolfBot: A distributed mobile sensing platform for research and education. In *Zone 1 Conf. of the American Society for Engineering Education (ASEE Zone 1)*, pages 1–8. IEEE, 2014.

[20] Mauro Birattari. Automatic design of robot swarms: achievements and challenges. *Frontiers in Robotics and AI*, 3:29, 2016.

[21] C Bohn and DP Atherton. A simulink package for comparative studies of pid anti-windup strategies. In *Computer-Aided Control System Design, 1994. Proceedings., IEEE/IFAC Joint Symposium on*, pages 447–452. IEEE, 1994.

[22] Eric Bonabeau, Guy Theraulaz, Jean-Louls Deneubourg, Serge Aron, and Scott Camazine. Self-organization in social insects. *Trends in Ecology & Evolution*, 12(5):188–193, 1997.

[23] Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4187–4193, Taipei, October 18–22, 2010. doi: 10.1109/IROS.2010.5649153.

[24] Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *Proc. IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems (IROS)*, pages 4187–4193, 2010.

[25] George EP Box. Robustness in the strategy of scientific model building. *Robustness in statistics*, 1:201–236, 1979.

[26] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964.

[27] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 25 (11):120–126, 2000.

[28] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell.*, 7 (1):1–41, March 2013. doi: 10.1007/s11721-012-0075-2.

[29] Alexandre Santos Brandão and Mário Sarcinelli-Filho. On the guidance of multiple uav using a centralized formation control scheme and delaunay triangulation. *Journal of Intelligent & Robotic Systems*, 84(1-4):397–413, 2016.

[30] A. Buffin and S. C. Pratt. Cooperative transport by the ant novomessor cockerelli. *Insectes Sociaux*, pages 1–10, 2016. ISSN 1420-9098. doi: 10.1007/s00040-016-0486-y. URL `http://dx.doi.org/10.1007/s00040-016-0486-y`.

[31] Aurélie Buffin, Takao Sasaki, and Stephen C. Pratt. Scaling of speed with group size in cooperative transport by ants. In preparation.

[32] Phillip Burgess. The magic of neopixels. URL `https://learn.adafruit.com/adafruit-neopixel-uberguide/overview`.

[33] Jianing Chen, Melvin Gauci, and Roderich Groß. A strategy for transporting tall objects with a swarm of miniature mobile robots. In *Proceedings of the 2013 International Conference on Robotics and Automation*, pages 863–869, Karlsruhe, Germany, May 6–10, 2013. doi: 10.1109/ICRA.2013.6630674.

[34] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nat. Nanotechnol.*, 8:755–762, 2013. doi: 10.1038/nnano.2013.189.

[35] David L Christensen, Srinivasan A Suresh, Katie Hahm, and Mark R Cutkosky. Let's all pull together: Principles for sharing large loads in microrobot teams. *IEEE Robotics and Automation Letters*, 1(2):1089–1096, 2016.

[36] Nikolaus Correll. Parameter estimation and optimal control of swarm-robotic systems: a case study in distributed task allocation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 3302–3307, Pasadena, CA, USA, May 19–23, 2008. IEEE. doi: 10.1109/ROBOT.2008.4543714.

[37] Nikolaus Correll and Alcherio Martinoli. Modeling and optimization of a swarm-intelligent inspection system. In *Proceedings of the Seventh International Symposium on Distributed Autonomous Robotics Systems (DARS 2004)*, pages 369–378, Toulouse, France, 2004. doi: 10.1007/978-4-431-35873-2_36.

[38] Tomer J. Czaczkes and Francis L. W. Ratnieks. Cooperative transport in ants (Hymenoptera: Formicidae) and elsewhere. *Myrmecol. News*, 18:1–11, 2013.

[39] Tomer J Czaczkes and Francis LW Ratnieks. Cooperative transport in ants (hymenoptera: Formicidae) and elsewhere. *Myrmecol. News*, 18:1–11, 2013.

[40] Karthik Dantu, Spring Berman, Bryan Kate, and Radhika Nagpal. A comparison of deterministic and stochastic approaches for allocating spatially dependent tasks in micro-aerial vehicle collectives. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 793–800, Vilamoura, Portugal, October 7–12, 2012. doi: 10.1109/IROS.2012.6386233.

[41] H. A. David and H. N. Nagaraja. *Expected Values and Moments*, pages 33–58. John Wiley & Sons, Inc., 2005. ISBN 9780471722168. doi: 10.1002/0471722162.ch3. URL `http://dx.doi.org/10.1002/0471722162.ch3`.

[42] Howard B. Demuth, Mark H. Beale, Orlando De Jess, and Martin T. Hagan. *Neural Network Design*. Martin Hagan, USA, 2nd edition, 2014. ISBN 0971732116, 9780971732117.

[43] E. Diller and M. Sitti. Micro-scale mobile robotics. *Foundations and Trends in Robotics*, 2:143–259, 2013.

[44] M. Dorigo et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, Dec 2013. ISSN 1070-9932. doi: 10.1109/MRA.2013.2252996.

[45] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.

[46] Elisabeth Eitel. Basics of rotary encoders: Overview and new technologies. `http://machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0`, 2014. Accessed: 2017-03-01.

[47] Karthik Elamvazhuthi, Chase Adams, and Spring Berman. Coverage and field estimation on bounded domains by diffusive swarms. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2867–2874. IEEE, 2016.

[48] Karthik Elamvazhuthi, Sean Wilson, and Spring Berman. Confinement control of double integrators using partially periodic leader trajectories. In *Proceedings of the 2016 American Control Conference (ACC)*, Boston, MA, USA, July 06–08, 2016.

[49] J. W. Evans. Random and cooperative sequential adsorption. *Rev. Mod. Phys.*, 65(4):1281–1329, October 1993. doi: 10.1103/RevModPhys.65.1281.

[50] William C. Evans, Grégory Mermoud, and Alcherio Martinoli. Comparing and modeling distributed control strategies for miniature self-assembling robots. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 1438–1445, Anchorage, AK, USA, May 3–7, 2010.

[51] H. Farivarnejad, S. Wilson, and S. Berman. Decentralized sliding mode control for autonomous collective transport by multi-robot systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1826–1833, Dec 2016. doi: 10.1109/CDC.2016.7798530.

[52] E. Ferrante, M. Brambilla, M. Birattari, and M. Dorigo. Socially-mediated negotiation for obstacle avoidance in collective transport. In *Proc. Int'l. Symp. on Distributed Autonomous Robotic Systems (DARS)*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 571–583, Naples, Italy, 2013. Springer.

[53] Steven R. Finch. *Mathematical Constants*, volume 94 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2003. ISBN 978-0521818056.

[54] D. Fouquet, A. M. Costa-Leonardo, R. Fournier, S. Blanco, and C. Jost. Coordination of construction behavior in the termite procornitermes araujoi: structure is a stronger stimulus than volatile marking. *Insectes Sociaux*, 61 (3):253–264, 2014. ISSN 1420-9098. doi: 10.1007/s00040-014-0350-x. URL `http://dx.doi.org/10.1007/s00040-014-0350-x`.

[55] Nigel R Franks. Teams in social insects: group retrieval of prey by army ants (eciton burchelli, hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 18(6):425–429, 1986.

[56] Nigel R Franks, Stephen C Pratt, Eamonn B Mallon, Nicholas F Britton, and David JT Sumpter. Information flow, opinion polling and collective intelligence in house–hunting social insects. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 357(1427):1567–1583, 2002.

[57] NR Franks, Andy Wilby, Bernard Walter Silverman, and C Tofts. Self-organizing nest construction in ants: sophisticated building by blind bulldozing. *Animal behaviour*, 44:357–375, 1992.

[58] Simon Garnier. From ants to robots and back: How robotics can contribute to the study of collective animal behavior. In Yan Meng and Yaochu Jin, editors, *Bio-Inspired Self-Organizing Robotic Systems*, volume 355 of *Studies in Computational Intelligence*, pages 105–120. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20759-4. doi: 10.1007/978-3-642-20760-0_5. URL `http://dx.doi.org/10.1007/978-3-642-20760-0_5`.

[59] Simon Garnier, Aurélie Guérécheau, Maud Combe, Vincent Fourcassié, and Guy Theraulaz. Path selection and foraging efficiency in argentine ant transport networks. *Behavioral Ecology and Sociobiology*, 63(8):1167–1179, 2009.

[60] Simon Garnier, Maud Combe, Christian Jost, and Guy Theraulaz. Do ants need to estimate the geometrical properties of trail bifurcations to find an efficient route? A swarm robotics test bed. *PLoS Comput. Biol.*, 9(3):e1002903, Mar 2013. doi: 10.1371/journal.pcbi.1002903. URL `http://dx.doi.org/10.1371/journal.pcbi.1002903`.

[61] Pierre-P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *Bellicositermes natalensis* et *Cubitermes sp.* la théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6(1):41–80, 1959. doi: 10.1007/BF02223791.

[62] R. Groß and M. Dorigo. Towards group transport by swarms of robots. *Int'l. Journal of Bio-Inspired Computation*, 1(1-2):1–13, 2009.

[63] Eliezer Gurarie. *Models and analysis of animal movements: From individual tracks to mass dispersal.* PhD thesis, University of Washington, 2008.

[64] Golnaz Habibi, K Zachary, William Xie, Mathew Jellins, and James McLurkin. Distributed centroid estimation and motion controllers for collective transport by multi-robot systems. In *Proc. IEEE Int'l. Conf. on Robotics and Automation (ICRA)*, 2015.

[65] Heiko Hamann and Heinz Wörn. A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intell.*, 2(2–4):209–239, December 2008. doi: 10.1007/s11721-008-0015-3.

[66] B. Hölldobler, R. C. Stanton, and H. Markl. Recruitment and food-retrieving behavior in Novomessor (Formicidae, Hymenoptera: I. Chemical signals). *Behav. Ecol. Sociobiol.*, 4:163–181, 1978.

[67] Gregory Hutchins. Development of graphical user interfaces and algorithms for controlling a robotic swarm. Barrett honors undergraduate thesis, Arizona State University, 2016.

[68] John M. C. Hutchinson and Peter M. Waser. Use, misuse and extensions of "ideal gas" models of animal encounter. *Biol. Rev.*, 82(3):335–359, August 2007. doi: 10.1111/j.1469-185X.2007.00014.x.

[69] Jennifer M Jandt, Sarah Bengston, Noa Pinter-Wollman, Jonathan N Pruitt, Nigel E Raine, Anna Dornhaus, and Andrew Sih. Behavioural syndromes and social insects: personality at multiple levels. *Biological Reviews*, 89(1):48–67, 2014.

[70] Raphaël Jeanson and Anja Weidenmüller. Interindividual variability in social insects–proximate causes and ultimate consequences. *Biological Reviews*, 89(3): 671–687, 2014.

[71] Kelin Jose and Dilip Kumar Pratihar. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems*, 80:34 – 42, 2016. ISSN

0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2016.02.003. URL `http://www.sciencedirect.com/science/article/pii/S0921889016000282`.

[72] Anaïs Khuong, Jacques Gautrais, Andrea Perna, Chaker Sbaï, Maud Combe, Pascale Kuntz, Christian Jost, and Guy Theraulaz. Stigmergic construction and topochemical information shape ant nest architecture. *Proceedings of the National Academy of Sciences*, 113(5):1303–1308, 2016.

[73] Evgeni Kiriy and Martin Buehler. Three-state extended kalman filter for mobile robot localization. *McGill University., Montreal, Canada, Tech. Rep. TR-CIM*, 5:23, 2002.

[74] Eric Klavins, Samuel Burden, and Nils Napp. Optimal rules for programmed stochastic self-assembly. In *Proceedings of Robotics: Science and Systems II*, Philadelphia, PA, USA, August 16–19, 2006.

[75] Jens Krause, Alan FT Winfield, and Jean-Louis Deneubourg. Interactive robots in experimental biology. *Trends in Ecology & Evolution*, 26(7):369–375, 2011.

[76] Michael JB Krieger, Jean-Bernard Billeter, and Laurent Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[77] C. Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robot. Auton. Syst.*, 30(1–2):85–101, January 2000. doi: 10.1016/S0921-8890(99)00066-4.

[78] Ganesh P. Kumar, Aurélie Buffin, Theodore P. Pavlic, Stephen C. Pratt, and Spring M. Berman. A stochastic hybrid system model of collective transport in the desert ant *Aphaenogaster cockerelli*. In *Proceedings of the 16th ACM International Conference on Hybrid Systems: Computation and Control*, pages 119–124, Philadelphia, PA, April 8–11, 2013. doi: 10.1145/2461328.2461349.

[79] Aleksandr Kushleyev, Vijay Kumar, and Daniel Mellinger. Towards a swarm of agile micro quadrotors. In *Proceedings of Robotics: Science and Systems VIII*, Sydney, NSW, Australia, July 9–13, 2012.

[80] T. H. Labella, M. Dorigo, and J.-L. Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Trans. Auton. Adapt. Syst.*, 1(1):4–25, 2006. ISSN 1556-4665. doi: 10.1145/1152934.1152936.

[81] Michael Lachmann and Guy Sella. The computationally complete ant colony: global coordination in a system with no hierarchy. In *Proceedings of the Third European Conference on Artificial Life*, pages 784–800, Granada, Spain, June 4–6, 1995.

[82] Irving Langmuir. The adsorption of gases on plane surfaces of glass, mica and platinum. *J. Am. Chem. Soc.*, 40(9):1361–1403, 1918. doi: 10.1021/ja02242a004.

[83] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002. ISSN 1573-7527. doi: 10.1023/A:1019633424543. URL `http://dx.doi.org/10.1023/A:1019633424543`.

[84] Quentin J Lindsey, Michael Shomin, and Vijay Kumar. Cooperative quasi-static planar manipulation with multiple robots. In *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1351–1360. American Society of Mechanical Engineers, 2010.

[85] Wenguo Liu and Alan F. T. Winfield. Modeling and optimization of adaptive foraging in swarm robotic systems. *Int. J. Robot. Res.*, 29(14):1743–1760, December 2010. doi: 10.1177/0278364910375139.

[86] Lennart Ljung. System identification toolbox^{TM}, 2017. URL `https://www.mathworks.com/help/pdf_doc/ident/ident.pdf`. Accessed: 2017-03-07.

[87] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, June 2011. doi: 10.1109/ICORR.2011.5975346.

[88] R. Mahony, T. Hamel, and J. M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5): 1203–1218, June 2008. ISSN 0018-9286. doi: 10.1109/TAC.2008.923738.

[89] Anne-Catherine Mailleux, Jean-Louis Deneubourg, and Claire Detrain. Regulation of ants' foraging to resource productivity. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(1524):1609–1616, 2003.

[90] Louis Major, Theocharis Kyriacou, and O Pearl Brereton. Systematic literature review: Teaching novices programming using robots. *IET Software*, 6(6):502–513, 2012.

[91] Kumar Malu and Sandeep Jharna Majumdar. Kinematics, localization and control of differential drive mobile robot. *Global Journal of Research In Engineering*, 14(1), 2014.

[92] Alcherio Martinoli, Kjerstin Easton, and William Agassounon. Modeling swarm robotic systems: a case study in collaborative distributed manipulation. *Int. J. Robot. Res.*, 23(4–5):415–436, April 2004. doi: 10.1177/0278364904042197.

[93] T. William Mather and M. Ani Hsieh. Distributed robot ensemble control for deployment to multiple sites. In *Proceedings of Robotics: Science and Systems VII*, Los Angeles, CA, USA, June 27–30, 2011.

[94] Loïc Matthey, Spring Berman, and Vijay Kumar. Stochastic strategies for a swarm robotic assembly system. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 1953–1958, Kobe, Japan, May 12–17, 2009.

[95] Helen F. McCreery and Michael D. Breed. Cooperative transport in ants: a review of proximate mechanisms. *Insectes Sociaux*, 61(2):99–110, May 2013. doi: 10.1007/s00040-013-0333-3.

[96] James McLurkin, Andrew J Lynch, Scott Rixner, Thomas W Barr, Alvin Chou, Kathleen Foster, and Siegfried Bilstein. A low-cost multi-robot system for research, teaching, and outreach. In *Distributed Autonomous Robotic Systems (DARS)*, pages 597–609. Springer, 2013.

[97] James McLurkin, Joshua Rykowski, Meagan John, Quillan Kaseman, and Andrew J. Lynch. Using multi-robot systems for engineering education: teaching and outreach with large numbers of an advanced, low-cost robot. *IEEE Trans. Educ.*, 56(1):24–33, February 2013. doi: 10.1109/TE.2012.2222646.

[98] James McLurkin, Adam McMullen, Nick Robbins, Golnaz Habibi, Aaron Becker, Alvin Chou, Hao Li, Michael John, Nnena Okeke, Jarogniew Rykowski, et al. A robot system design for low-cost multi-robot manipulation. In *Proc. IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems (IROS)*, pages 912–918, 2014.

[99] Erik Meijering, Oleh Dzyubachyk, and Ihor Smal. Chapter nine - methods for cell and particle tracking. In P. Michael conn, editor, *Imaging and Spectroscopic Analysis of Living CellsOptical and Spectroscopic Techniques*, volume 504 of *Methods in Enzymology*, pages 183 – 200. Academic Press, 2012. doi: http://dx.doi.org/10.1016/B978-0-12-391857-4.00009-4. URL `http://www.sciencedirect.com/science/article/pii/B9780123918574000094`.

[100] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proc. 9th Conf. on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[101] Nils Napp and Eric Klavins. A compositional framework for programming stochastically interacting robots. *Int. J. Robot. Res. [Special Issue Stochasticity in Robot. Bio-Systems Part 2]*, 30(6):713–729, May 2011. doi: 10.1177/0278364911403018.

[102] Nils Napp, Samuel Burden, and Eric Klavins. Setpoint regulation for stochastically interacting robots. In *Proceedings of Robotics: Science and Systems V*, Seattle, WA, USA, June 28 – July 1, 2009.

[103] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *ISRN Robotics*, 2013, 2012.

[104] Iñaki Navarro and Fernando Matía. A survey of collective movement of mobile robots. *International Journal of Advanced Robotic Systems*, 10(1):73, 2013.

[105] Michael A Neumann, Matthew H Chin, and Christopher A Kitts. Object manipulation through explicit force control using cooperative mobile multi-robot systems. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, 2014.

[106] Lael U. Odhner and Harry Asada. Stochastic recruitment control of large ensemble systems with limited feedback. *J. Dyn. Syst., Meas., Control.*, 132(4): 041008, July 2010. doi: 10.1115/1.4001706.

[107] Rehan O'Grady, Carlo Pinciroli, Roderich Groß, Anders Lyhne Christensen, F. Mondada, Michael Bonani, and Marco Dorigo. Swarm-bots to the rescue. In *Proceedings of the 10th European Conference on Artificial Life*, volume 5777 of *Lecture Notes in Computer Science*, pages 165–172, Budapest, Hungary, September 13–16, 2009. Springer-Verlag. doi: 10.1007/978-3-642-21283-3_21.

[108] Talat Ozyagcilar. Calibrating an ecompass in the presence of hard- and soft-iron interference. `http://cache.freescale.com/files/sensors/doc/app_note/AN4246.pdf`, 2015. Accessed: 2017-03-01.

[109] Theodore P Pavlic, Sean Wilson, Ganesh P Kumar, and Spring Berman. An enzyme-inspired approach to stochastic allocation of robotic swarms around boundaries. In *16th International Symposium on Robotics Research (ISRR)*, pages 16–19, Singapore, December 16–19 2013.

[110] Michael A Peshkin and Arthur C Sanderson. Minimization of energy in quasi-static manipulation. *IEEE Transactions on Robotics and Automation*, 5(1): 53–60, 1989.

[111] Daniel Pickem, Myron Lee, and Magnus Egerstedt. The GRITSBot in its natural habitat - a multi-robot testbed. In *Proc. IEEE Int'l. Conf. on Robotics and Automation (ICRA)*, 2015.

[112] Stephen C Pratt, Eamonn B Mallon, David J Sumpter, and Nigel R Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant leptothorax albipennis. *Behavioral Ecology and Sociobiology*, 52(2):117–127, 2002.

[113] Amanda Prorok, Nikolaus Correll, and Alcherio Martinoli. Multi-level spatial modeling for stochastic distributed robotic systems. *Int. J. Robot. Res.*, 30(5): 574–589, April 2011. doi: 10.1177/0278364910399521.

[114] Ragesh K. Ramachandran, Sean Wilson, and Spring Berman. A probabilistic topological approach to feature identification using a stochastic robotic swarm. In *The 13th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2016.

[115] Ragesh K Ramachandran, Sean Wilson, and Spring Berman. A probabilistic approach to automated construction of topological maps using a stochastic robotic swarm. *IEEE Robotics and Automation Letters*, 2(2):616–623, 2017.

[116] Alfréd Rényi. On a one-dimensional problem concerning random space-filling. *Publ. Math. Inst. Hung. Acad. Sci.*, 3:109–127, 1958.

[117] F. Riedo, M. Chevalier, S. Magnenat, and F. Mondada. Thymio II, a robot that grows wiser with children. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 187–193, Nov 2013. doi: 10.1109/ARSO.2013.6705527.

[118] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *Proc. IEEE Int'l. Conf. on Robotics and Automation (ICRA)*, pages 3293–3298, 2012.

[119] Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James McLurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, Saint Paul, Minnesota, USA, May 6–10, 2013. URL `http://dl.acm.org/citation.cfm?id=2484920.2484932`.

[120] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. doi: 10.1126/science.1254295. URL `http://www.sciencemag.org/content/345/6198/795.abstract`.

[121] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[122] RM Sakia. The box-cox transformation technique: a review. *The statistician*, pages 169–178, 1992.

[123] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. Nih image to imagej: 25 years of image analysis. *Nat Meth*, 9(7):671–675, July 2012. ISSN 15487091. URL `http://dx.doi.org/10.1038/nmeth.2089`.

[124] Sebastian Schwarz, Michael Mangan, Jochen Zeil, Barbara Webb, and Antoine Wystrach. How ants use vision when homing backward. *Current Biology*, pages –, 2017. ISSN 0960-9822. doi: http://dx.doi.org/10.1016/j.cub.2016.12.019. URL `//www.sciencedirect.com/science/article/pii/S096098221631466X`.

[125] Ehud Shapiro and Tom Ran. DNA computing: molecules reach consensus. *Nat. Nanotechnol.*, 8:703–705, 2013. doi: 10.1038/nnano.2013.202.

[126] Andrew Sih, Alison Bell, and J Chadwick Johnson. Behavioral syndromes: an ecological and evolutionary overview. *Trends in ecology & evolution*, 19(7):372–378, 2004.

[127] Herbert Solomon and Howard Weiner. A review of the packing problem. *Commun. in Stat. - Theory Methods*, 15(9):2571–2607, 1986. doi: 10.1080/03610928608829274.

[128] David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Nat. Comput.*, 7: 615–633, 2008. doi: 10.1007/s11047-008-9067-y.

[129] David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proc. Natl. Acad. Sci. USA*, 107(12):5393–5398, March 23, 2010. doi: 10.1073/pnas.0909380107.

[130] D.J. Stilwell and J.S. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 766–771, May 1993. doi: 10.1109/ROBOT.1993.292070.

[131] Ashley Stroupe, Terry Huntsberger, Avi Okon, and Hrand Aghazarian. Precision manipulation with cooperative robots. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 235–248. Springer, 2005.

[132] K. Sugawara, D. Reishus, and N. Correll. Object transportation by granular convection using swarm robots. In *Distributed Autonomous Robotic Systems*, STAR, Baltimore, MD, 2012. Springer-Verlag. URL `http://spot.colorado.edu/~dure7259/papers/Sugawara12.pdf`.

[133] David JT Sumpter and Madeleine Beekman. From nonlinearity to optimality: pheromone trail foraging by ants. *Animal behaviour*, 66(2):273–280, 2003.

[134] J. Talbot, G. Tarjus, P. R. Van Tassel, and P. Viot. From car parking to protein adsorption: an overview of sequential adsorption processes. *Colloids Surfaces, A: Physicochem. Eng. Asp.*, 165(1–3):287–324, May 30, 2000. doi: 10.1016/S0927-7757(99)00409-4.

[135] Jeffrey C Trinkle. A quasi-static analysis of dextrous manipulation with sliding and rolling contacts. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 788–793. IEEE, 1989.

[136] P Kirk Visscher. Group decision making in nest-site selection among social insects. *Annu. Rev. Entomol.*, 52:255–275, 2007.

[137] Shihu Wang and Elena E. Dormidontova. Selectivity of ligand-receptor interactions between nanoparticle and cell surfaces. *Phys. Rev. Lett.*, 109:238102, Dec 2012.

[138] Zijian Wang and Mac Schwager. *Multi-robot Manipulation Without Communication*, pages 135–149. Springer Japan, Tokyo, 2016. ISBN 978-4-431-55879-8. doi: 10.1007/978-4-431-55879-8_10. URL `http://dx.doi.org/10.1007/978-4-431-55879-8_10`.

[139] Uri Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, USA, 1999. URL `http://ccl.northwestern.edu/netlogo/`.

[140] Sean Wilson, Aurélie Buffin, Stephen C. Pratt, and Spring Berman. Multi-robot replication of ant collective towing behaviors. In preparation.

[141] Sean Wilson, Theodore P. Pavlic, Ganesh P. Kumar, Aurélie Buffin, Stephen C. Pratt, and Spring Berman. Design of ant-inspired stochastic control policies for collective transport by robotic swarms. *Swarm Intelligence*, 8(4):303–327, December 2014. doi: 10.1007/s11721-014-0100-8.

[142] Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, a versatile swarm robotic research and education platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.

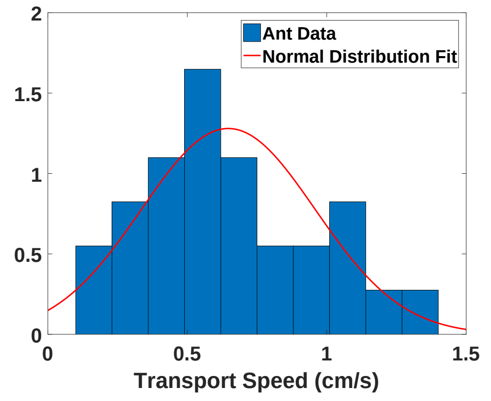
[143] R. J. Wood, B. Finio, M. Karpelson, K. Ma, N. O. Pérez-Arancibia, P. S. Sreetharan, H. Tanaka, and J. P. Whitney. Progress on 'pico' air vehicles. *Int. J. Robot. Res.*, 31(11):1292–1302, September 2012. doi: 10.1177/0278364912455073.

[144] Tae Suk Yoo, Sung Kyung Hong, Hyok Min Yoon, and Sungsu Park. Gain-scheduled complementary filter design for a mems based attitude and heading reference system. *Sensors*, 11(4):3816–3830, 2011.

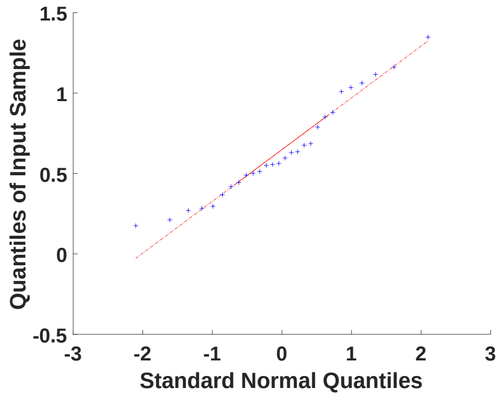
[145] Guoxian Zhang, G.K. Fricke, and D.P. Garg. Spill detection and perimeter surveillance via distributed swarming agents. *IEEE/ASME Transactions on Mechatronics*, 18(1):121–129, Feb 2013. ISSN 1083-4435. doi: 10.1109/TMECH.2011.2164578.

APPENDIX A

STATISTICAL ANALYSIS OF LOAD TOWING SPEEDS IN *NOVOMESSOR COCKERELLI* ANTS



(a)

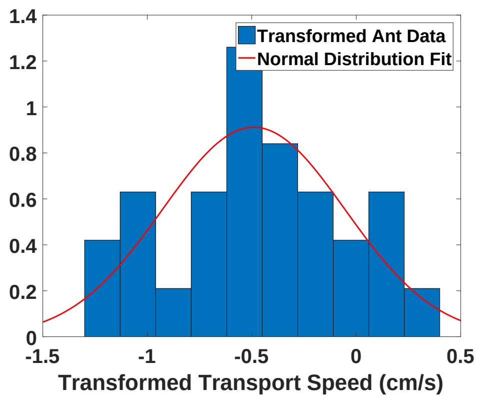


(b)

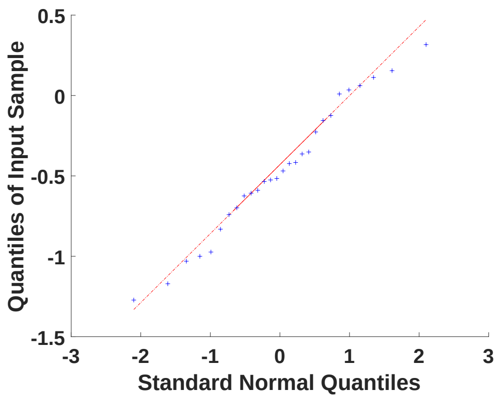
Figure A.1. *Left*, histogram (blue) of data of the individual transport speeds from *N. cockerelli* experiments, with a Gaussian fit (red). *Right*, quantile-quantile plot of the same data.

This appendix provides a justification for the assumption that individual transport speeds observed in *Novomessor cockerelli* ants are normally distributed, as stated in Chapter 6. Figure A.1a shows a histogram of data on the transport speeds of individual ants. Both the Anderson-Darling and Kolmogorov-Smirnov tests for normality find no significant departure from normality ($p = 0.05$). The quantile-quantile (q-q) plot in Figure A.1b also shows a relatively linear relationship. However, this data has a skewness value of 0.4985, which is evident in both Figure A.1a and Figure A.1b. This is present due to the inherent lower bound of the data (e.g., the magnitude of the transport speed cannot be negative.).

The order statistics analysis presented in Chapter 6 relies on the assumption that the data is normally distributed. Although this assumption is not violated, it is important to check whether the order statistics prediction still holds when the data does not have this skew present. To eliminate the skew in the data, a Box-Cox transformation [26, 122] is performed on the data. This reduces the skew of the data to -0.0499. The transformed data with a Gaussian fit is shown in Figure A.2a, with the associated q-q plot shown in Figure A.2b. The figures confirm that the transformed data is no longer subject to the inherent lower bound of the original data. The first order statistic is applied to the Box-Cox transformed Gaussian fit of the ant transport data. This is compared to the Box-Cox transformed ant transport data for each team size in Figure A.3. The order statistics prediction still exhibits the trend of reduced steady-state transport speed with increasing team size. However, analysis using the transformed speeds is not as intuitive as the original data, and thus the original data are presented in Chapter 6.



(a)



(b)

Figure A.2. *Left*, histogram (blue) of data of the individual transport speeds from *N. cockerelli* experiments after Box-Cox tranformation, with a Gaussian fit (red). *Right*, quantile-quantile plot of the same data.

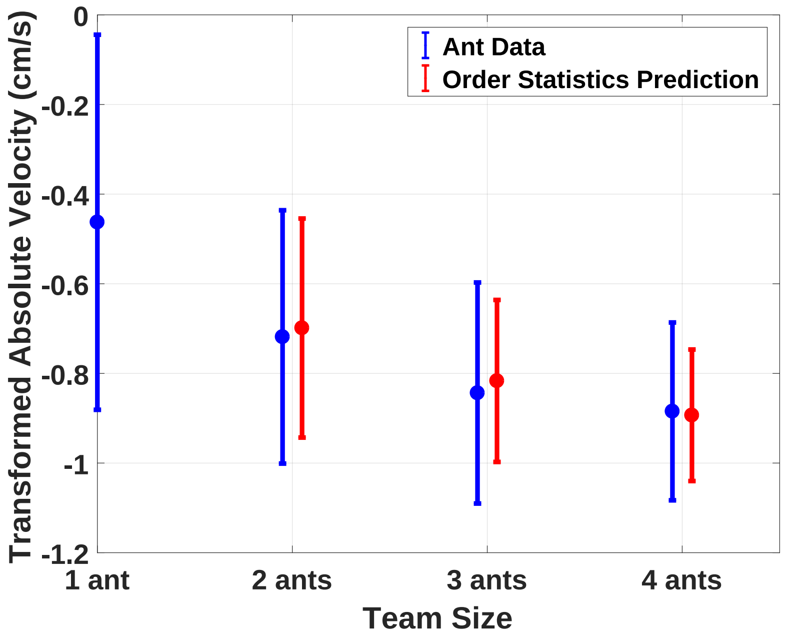


Figure A.3. *Blue plots:* Steady-state speed of the artificial load (0.3 g per ant) with varying team size [31] after applying the Box-Cox transformation. The circles with error bars represent the average ± standard deviation across 30 experimental trials (10 trials per ant colony). *Red plots:* First order statistic for each team size of a normal distribution with mean ± standard deviation fit to individual *N. cockerelli* transport speed data after applying the Box-Cox transformation.

APPENDIX B

DESCRIPTION, CALIBRATION, AND FUSION OF THE PHEENO ROBOT'S SENSOR SUITE FOR RELIABLE FEEDBACK

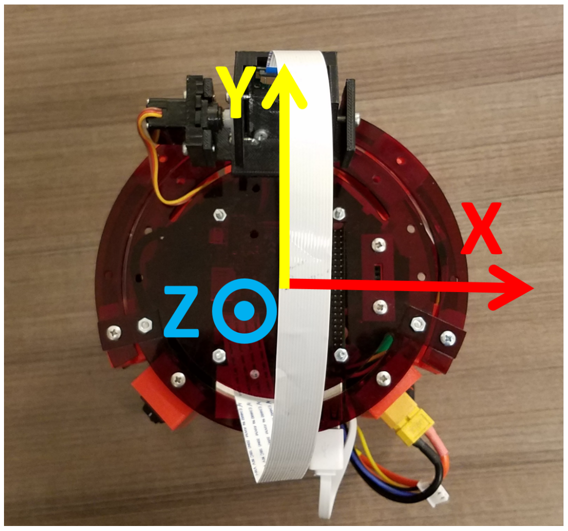


Figure B.1. The reference frame of Pheeno. Ideally, this should be the same reference frame as the inertial measurement unit.

This appendix is meant to help new robotics users and specifically users of the Pheeno robotic platform get used to onboard sensing typically used as feedback in control theory. Typically when learning the theory behind controllers the feedback term is assumed to be perfect and known. In reality, these feedbacks are based on sensor measurements that must be calibrated properly with limitations that should be known. There are many different types of sensors with large ranges of precision and cost. Here, the focus will be put on the sensors onboard the Pheeno platform and fusing their measurements to produce reliable feedback.

Section B.1 will provide a high level introduction to the sensors on board the Pheeno robotic platform, their strengths and weaknesses, and how they should be used. Section B.2 will go into how and why to calibrate sensors on board the robot. Section B.3 will introduce different methods to combine sensor measurements into reliable, robust position and orientation estimates for the robot. It will also briefly go into the pros and cons of each fusion.

B.1 Getting to Know the Onboard Sensors.

Pheeno is equipped with a MiniIMU-9 v5 made by Pololu. This little board contains a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer. Figure B.1 shows the ideal reference frame of the MiniIMU-9 v5 aligned with Pheeno's reference frame. Pheeno is also equipped with 6 *infrared* (IR) distance sensors, a quadrature encoder on each motor, and an RGB camera. Each of these sensors alone are very powerful, however, they each have critical weaknesses which will be explored below. Together, with the help of a little mathematics, these sensors make up for each other's weakness and allow Pheeno to figure out where it is in space and what its surroundings are like.

One thing that will not be covered in this document is converting raw data/voltages provided by the MinIMU-9 v5 or any of the sensors on the robot. For reference, sensors typically output analog or digital voltages which must be converted to digital signals that may then be transformed to standard units by a micro processor. Other, more complex sensors, have micro chips on board that use communication protocols like *Inter Integrated Circuit Communications* (I^2C) and *Serial Peripheral interface* (SPI).

A typical *inertial measuring unit* (IMU) like the MiniIMU-9 v5 will not output raw data in any units that are useful to the operator. IMUs like this typically have different operating ranges that can be set by the user. For whatever range is chosen by the user, a conversion factor is typically given in the datasheet associated with the IMU. It is important to note however, all sensors on a robot are quantized at some resolution (meaning whatever continuous source they measure will be represented at discrete increments). Typically there is a trade off between the resolution of measurements and the range of values that can be measured. For example, the standard setting for the gyroscope on Pheeno is set to measure an angular rotation rate range of $\pm 245°/s$. This allows for a measurement resolution of $0.00875°/s$. This is fine for determining rotations on a small robotic platform like Pheeno but this concept should not be ignored. The resolution and range of any sensor on board should be chosen as appropriate to the application.

B.1.1 Accelerometer

An accelerometer measures its linear acceleration along several principle directions. In Pheeno's case the MiniIMU-9 v5 has a 3-axis accelerometer allowing the user to know how the robot's linear acceleration in all three possible dimensions. From this sensor alone the user can infer the direction the robot is accelerating as well as its pitch and roll angle (if you are unfamiliar with roll and pitch angles, refer to Figure B.12, they will be further explained later). If Pheeno drives off a cliff, it will be very apparent from the accelerometer's sensor measurements that it is tumbling to its doom.

To explain what an accelerometer should be used for, raw accelerometer data from a resting and rotating Pheeno will be analyzed. First, Pheeno was placed onto a table top and 60 seconds of accelerometer data was recored at a rate of 100 Hz (every 10 ms). Figure B.2 shows the raw time series data of the the acceleration felt along each axis of the accelerometer. This data shows the readings are noisy but very stable. The accelerometer is currently just measuring gravity which should be only in the z-direction. However, from this plot it is apparent there are slight measurements in the x and y direction as well. This is due to slight misalignment between the IMU board orientation and Pheeno's resting orientation caused during manufacturing of the robot. A misalignment similar to this should be present in every robot. From this data, it is possible to calibrate the accelerometer and determine how tilted the board is relative to the robot, thus aligning their reference frames.

Figure B.3 shows the acceleration along the Y-Direction of the accelerometer after the time averaged bias has been subtracted. this plot shows most of the noise in the raw sensor measurements (black line) is around $\pm 1 cm/s$. If the sensor is moved around quickly, the accelerations of the motion will be measured and the vibrations during the motion will amplify the noise. It should be noted how the 100 sample running mean of the signal (red line) provides a much more stable signal. This shows the noise content of the signal is at a high frequency and a *low pass filter* can be used to get reliable measurements from the accelerometer. However, a low pass filter will not track quick acceleration changes well by its design. This will be addressed later.

Figure B.4 shows the velocity approximated from the accelerometer if the accelerometer measurements are integrated. Over small time intervals this works reasonably well but over long time periods *integration error* overwhelms the approximation. Recall, this is from an accelerometer at rest on a table. During the 60 second interval pictured, the at rest accelerometer yields non zero velocities. Note this happens even when the signal is averaged and the noise is not as severe.

Figure B.5 shows the position approximated from integrating the accelerometer measurements twice. As expected this leads to compounded integration error which causes this position approximation to be 7.5 cm from its original location in 60 seconds. If left to rest for a longer period of time, the accelerometer's estimation of position will only get worse.

From these results, an accelerometer should really only be used to determine the angle of orientation of the robot (pitch and roll), the acceleration of the robots motion, and sometimes the velocity of the robot over short time periods. These become tricky when the robot is actually being driven around since acceleration from the motion of the robot and interaction between the robot and its environment is measured by the accelerometer. If the robot is in a terrain where the ground changes the orientation of the robot drastically this challenge becomes even more difficult but not impossible. Later the accelerometer's readings will be combined with other on-board sensors to give an accurate measurement of the robot's orientation and linear acceleration. An accelerometer should almost never (really never) be used to approximate the robot's position through integration, especially over longer time scales. Integration error adds up very quickly and throws off a robot's localization significantly.

B.1.2 Gyroscope

Gyroscopes measure the rotational velocity about several principle axes. In Pheeno's case, the MiniIMU-9 v5 has a 3-axis gyroscope allowing the user to know how fast the robot is rotating about each axis. This sensor is typically used in parallel with the magnetometer and accelerometer to determine the angular orientation of the robot at all times. Unlike the accelerometer and magnetometer which typically have very accurate but noisy signals that cannot detect fast motions, gyroscopes are great at capturing fast rotations without being affected by the accelerations, but their orientation estimates will drift over time due to integration error.

To better understand a gyroscope's output and limitations Pheeno was manually rotated 90 degrees (~ 1.57 radians) back and forth about its z-axis for 60 seconds. The raw data is shown in Figure B.6a. Due to the misalignment of the MiniIMU-9 v5 board with the robot's reference frame, there is noticeable rotation about the z-axis (yaw) and y-axis (roll) and x-axis (pitch). This can be corrected in calibration which will be discussed in Section B.2.2. The corrected data are shown in Figure B.6b. This data shows that the gyroscope is very good at capturing consistent angular rate data at high frequencies with a relatively small amount of noise. Figure B.7a shows the data from Figure B.6a integrated once to determine the Pheeno's orientation.

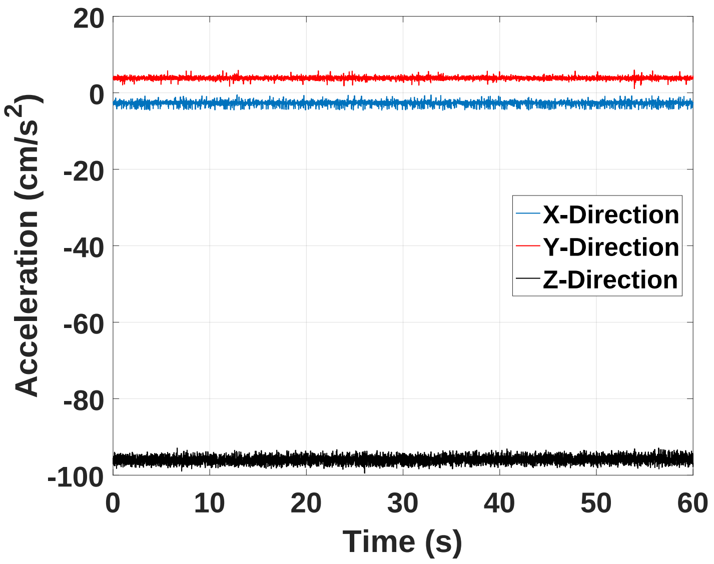


Figure B.2. The time evolution of raw accelerometer data along each principle axis measured when Pheeno is resting on a table. The blue line is along the x-direction of the IMU, the red line is along the y-direction of the IMU, and the black line is along the z-direction of the IMU.

Again, this data is uncalibrated and the errors from the misalignment and offset add up quickly. After calibration this runaway integration decreases as shown in Figure B.7b. The yaw orientation estimate is fairly consistent showing approximately 90 degrees of rotation back and forth at two different rates. Due to the nature of the rotation (manually turning the robot back and forth and eyeballing a 90° rotation) there is slight deviation in the periods of the rotations as well as the magnitudes. The calibration here is not perfect as some of the rotation is captured in the roll estimate but this is small and with more exact and thorough calibration can be suppressed further.

The weakness of gyroscope measurements is commonly referred to as *drift*, where the integration error causes the estimate to " *drift*" in one direction. Figure B.7b shows this as the yaw estimate is slowly becoming more and more negative. Gyroscopes are great at estimating rotations over small time scales but suffer if this estimate is done for long periods of time. While frustrating, it is important to note this is the opposite of the accelerometer measurements which are not reliable over short time spans but very reliable over long ones because their orientation estimate does not rely on integration and thus will not drift. However, accelerometers are only able to recover two angles of orientation, which have been chosen to be pitch (rotation about the x-axis) and roll (rotation about the y-axis). In order to determine the robots yaw angle (rotation about the z-axis) for long time spans, the robot requires an additional sensor, the magnetometer.

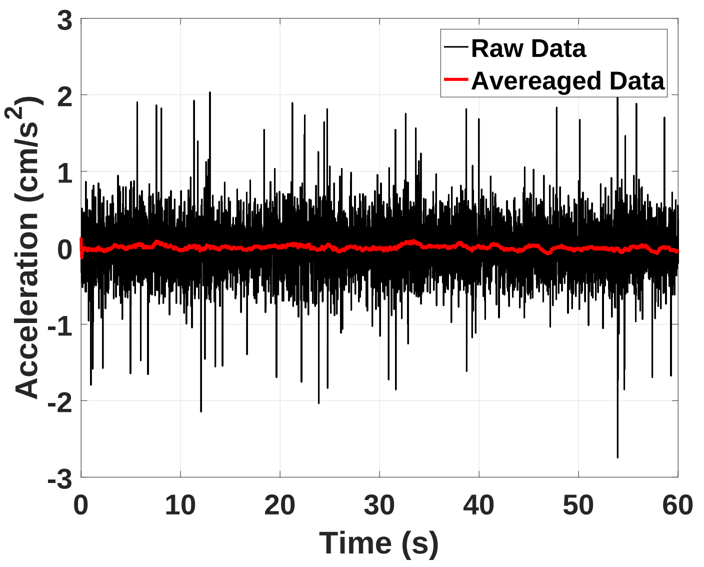


Figure B.3. The time evolution of raw accelerometer data (black) and averaged accelerometer data (red) along the y-direction of the IMU after the time averaged bias is subtracted. The average line is a running 100 sample average.

B.1.3 Magnetometer

The magnetometer is very similar to an accelerometer. However, in stead of measuring a gravity vector it measures the direction of the magnetic field surrounding it. The magnetometer on the MiniIMU-9 v5 represents the magnetic field vector along the same axis as the accelerometer. Its readings do not drift (given there are no magnetic anomalies) but are noisy like the accelerometer.

Typically the magnetic field being read is dominated by the earth's magnetic field. This field can easily be influenced by other magnetic sources like wires carrying large currents in buildings, large motors onboard the robot, and large metal beams in buildings. Luckily Pheeno is a small robot that uses low current micro metal gear motors that do not produce large enough magnetic fields to really influence the magnetometer readings. In larger robots with bigger motors, the magnetometer's proximity to the motors should be accounted for. Since Pheeno is typically used indoors, it is important for the user to determine if the room the robot is being used in has a consistent magnetic field. If the field changes too drastically in some areas those areas must be avoided or the magnetometer cannot be used reliably.

Assuming the area the robot is in has a consistent magnetic field, when the robot is rotated the measured magnetic field vector should point to the surface of a sphere centered at the origin with radius equal the strength of the present magnetic field. However, without calibration the readings from a rotating robot will look like Figure B.13a. This plot is 2D planar slices of the 3D ellipsoid showing the resulting measurements are not a sphere centered at the origin. Typically the offset of this

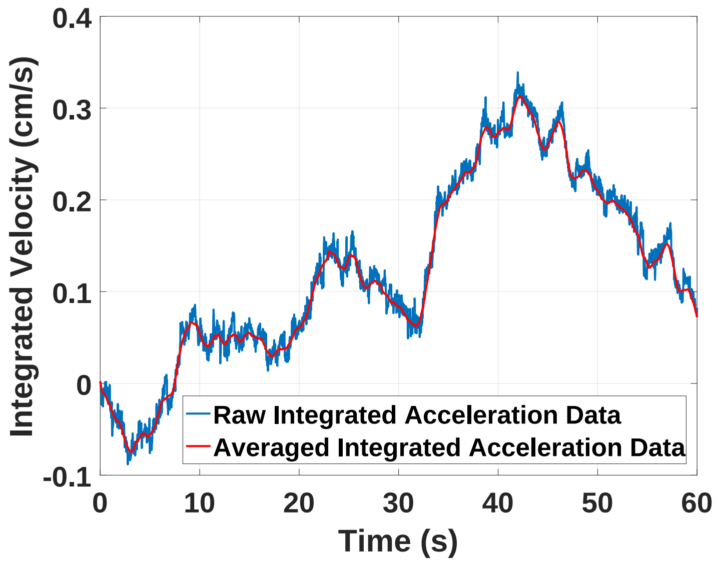


Figure B.4. The time evolution of the approximation of velocity after integrating accelerometer measurements from Figure B.3. The blue line is from integrating the raw accelerometer data, the red line is from integrating the running 100 sample averaged accelerometer data.

ellipsoid from the origin is referred to as *hard iron* error and the directional scaling in each direction is called *soft iron* error. It is crucial the magnetometer is calibrated properly to correct both these errors. If the magnetometer is not calibrated correctly, Pheeno will be unable to reliably determine its yaw angle and thus heading angle. This will be further discussed in Section B.2.3.

B.1.4 Motor Encoders

Motor encoders are used to count the number of shaft rotations that have occurred. Figure B.8a is a cartoon of a simple optical encoder. The wheel that is attached to the shaft blocks the light from hitting the sensor at set increments which creates a signal that is able to be measured by a micro controller. This is the foundation of all encoders. An emitter's signal (the light source in this case) is interrupted by a shaft attachment creating known patterns (e.g. 12 per rotation, a known pattern at a set angle, etc). However, there are many variations of encoders which use different emitters (e.g. magnetic, optical, electrical contact, resistance, etc.), have different number of sensors, and should be used in different scenarios.

Encoders typically fall into two classes, *incremental* and *absolute*. Incremental encoders, shown in the top of Figure B.8a, are only able to determine whether a rotation of the shaft has taken place and increment or decrement their rotation count. Absolute encoders, shown in the bottom of Figure B.8a, are able to determine the orientation of the shaft at any time (within some angular resolution) due to a specific

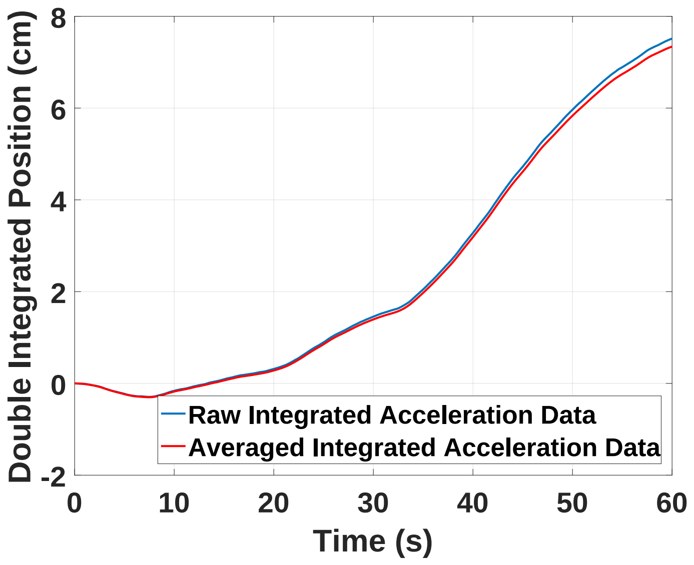


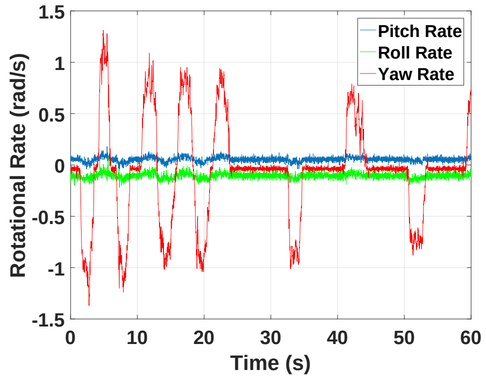
Figure B.5. The time evolution of the approximation of position after integrating accelerometer measurements from Figure B.3 twice. The blue line is from integrating the raw accelerometer data, the red line is from integrating the running 100 sample averaged accelerometer data.

feedback at each orientation. Typically incremental encoders are less expensive than absolute encoders but are susceptible to errors if counts are missed due to power loss or other factors. This can be a catastrophic problem in a robotic arm assembling a car frame but is less consequential in a small robotic platform like Pheeno.

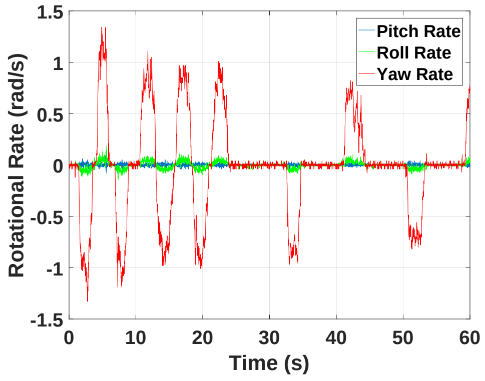
Mobile platforms like Pheeno use encoders to count the number of rotations each wheel has made. This can then be used as feedback to determine how fast the motor is actually rotating the platforms wheels and how far the robot's body has traveled or rotated. However, this position and orientation estimates are extremely vulnerable to non level surfaces and wheel slip. Both of these problems cannot be remedied by absolute encoders and thus Pheeno uses incremental encoders, specifically quadrature encoders.

Unlike standard (one sensor) encoders, which can only determine speed and displacement, quadrature encoders can determine velocity and direction. The major difference here is quadrature encoders can determine the direction of rotation of a shaft. It is possible to use a normal encoder on a motor and trust the encoder is rotating in the direction commanded, however, it is a horrible idea to assume your input direction translates to your output direction. In a small robot like Pheeno if the motors are turned off and allowed to rotate passively, this rotation will be captured correctly by quadrature encoders and will not necessarily be captured correctly by normal encoders.

Pheeno uses magnetic quadrature encoders that use hall effect sensors to detect rotational motion of its motors. Hall effect sensors are used because the emitter and



(a) Uncalibrated



(b) Calibrated

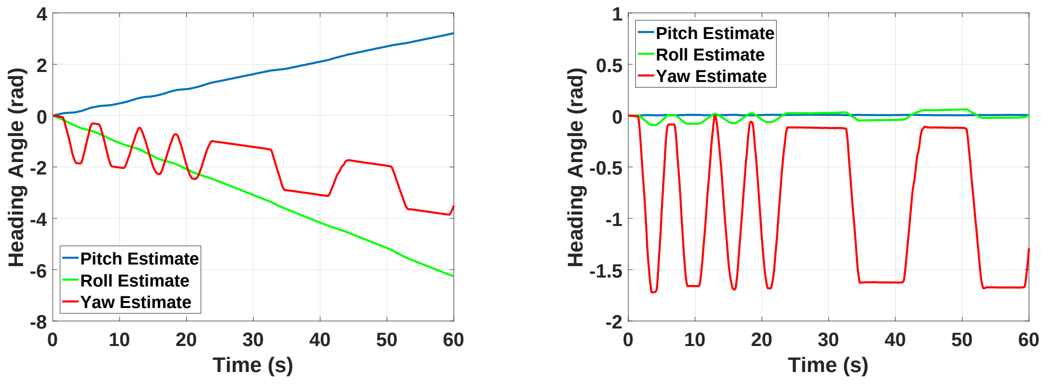
Figure B.6. The time evolution of the (a) uncalibrated and (b) calibrated rotational velocity measurements about each axis measured by the gyroscope on board Pheeno. The robot was rotated 90 degrees back and forth over a 60 second period.

sensor do not need to be well aligned and are not effected by lighting conditions like optical encoders. The encoder is mounted on an *extended back shaft* of a micro metal gear motor which rotates at the speed of the motor before the rotation is geared down to the wheel shaft. This allows for higher resolution measurement of the wheel rotation.

B.1.5 Infrared Distance Sensors

Pheeno uses six Sharp GP2Y0A41SK0F analog infrared distance sensors to sense linear distance of objects around the robot. These sensors were chosen due to their price instead of an expensive *Light Detection And Ranging* (LIDAR), sometimes referred to as *Light Imaging, Detection, And Ranging*, sensors. They also do not have issues with *ghost echoes* like *Sound Navigation And Ranging* (SONAR) sensors. Five are uniformly distributed radially along the front 180° of the robot with one in the rear. These are made to measure 4 cm to 30 cm distances in front of the sensor. However, they can be exchanged with any other IR sensor with JST mounts for different distance ranges (given the power, signal, and ground connections are the same).

Figure B.9 shows a cartoon representation of how a Sharp IR distance sensor works. A *light emitting diode* (LED) emits a beam of light through a lens which reflects off a surface and hits a different location on the *position-sensible photo detector* (PSD). Based on the location struck by the reflected beam, a distance measurement can be produced. This method of measuring distance has a few drawbacks that should be recognized. The first is, small or very curved surfaces (e.g. a chair leg) can not always be observed by the IR sensor because the initial beam will miss the object or the reflection will miss the PSD. The second reason is, light absorbing surfaces (e.g. darker surfaces) will absorb a lot of the sensing beam and thus will not cause a reflected beam. The final is a common problem in most light based distance



(a) Uncalibrated **(b)** Calibrated

Figure B.7. The time evolution of Pheeno's estimation of orientation from integration of (a) uncalibrated and (b) calibrated gryoscopic rate measurements in (a) Figure B.6a (b) Figure B.6b. The robot was rotated 90 degrees back and forth over a 60 second period.

sensors. The beam is very thin and thus the sensor must be aligned both in height and orientation with the object to create a reliable measurement.

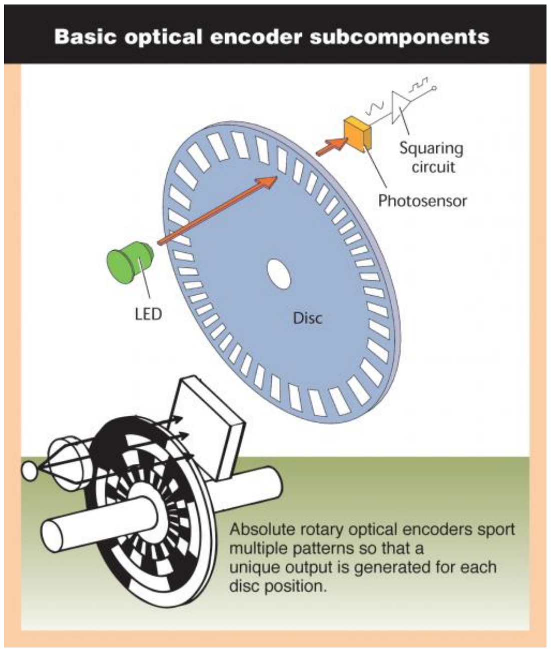
B.1.6 Camera

Pheeno has an optional *Raspberry Pi Camera* attached to the top of the robot which allows visual information to be captured, processed, and/or transmitted to other devices. This camera is a five mega-pixel fixed-focus camera that is able to capture 1080p resolution at 30 frames per second (fps), 720p resolution at 60 fps, or lower resolutions at 90 fps. These frame rates are possible in theory but in naive practice these frame rates drop due to the robot's processing capabilities (on board the raspberry pi). When performing further processing on the robot, these frame rates drop even further unless more advanced techniques beyond the scope of this document are implemented.

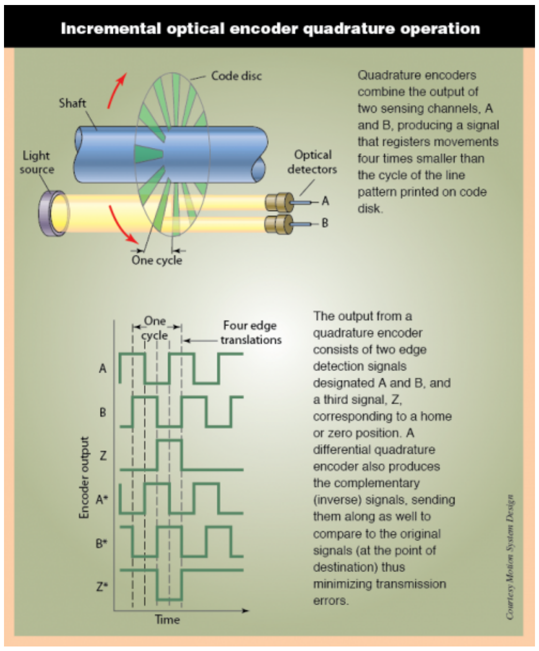
B.2 Calibration

This section will go through the calibration required to use Pheeno's on-board sensor suite correctly. This is meant as a high level calibration, meaning the misalignment of the sensors and their sensing ranges will be corrected. Typically these sensors should be calibrated to deal with temperature changes and other factors but that will not be covered.

Without calibration the on-board sensors could still be used but would yield information with systematic errors constantly. Calibration really only needs to be done to the IMU. The accelerometer calibration will be covered in Section B.2.1, gyroscope calibration will be shown in Section B.2.2, and the magnetometer calibration will


Basic optical encoder subcomponents
Squaring circuit
Photosensor
LED
Disc
Absolute rotary optical encoders sport multiple patterns so that a unique output is generated for each disc position.

(a) Diagram of an incremental (top) and absolute (bottom) optical encoder.


Incremental optical encoder quadrature operation
Code disc
Shaft
Light source
Optical detectors
A
B
One cycle
Quadrature encoders combine the output of two sensing channels, A and B, producing a signal that registers movements four times smaller than the cycle of the line pattern printed on code disk.
One cycle
Four edge translations
A
B
Z
A*
B*
Z*
Encoder output
Time
The output from a quadrature encoder consists of two edge detection signals designated A and B, and a third signal, Z, corresponding to a home or zero position. A differential quadrature encoder also produces the complementary (inverse) signals, sending them along as well to compare to the original signals (at the point of destination) thus minimizing transmission errors.
Courtesy Motion System Design

(b) Diagram of an incremental quadrature optical encoder (top) and the produced signal (bottom).

Figure B.8. Encoder diagrams. Figure from Eitel [46].

be done in Section B.2.3. Encoders typically work out of the box. Any calibration needed to detect the changes in magnetic or optical field created by the encoder has already been done before purchasing the sensor. Due to this, Section B.2.4 will discuss common problems that can be avoided with counting incremental encoders and how to transform counts to linear distances traveled by the wheels of the robot. IR distance sensors typically have a conversion factor available in the data sheet that transforms the voltage read to distance, however, it is a good idea to not blindly trusting data sheets and fit distance data to readings on their own. Cameras, like the one on Pheeno, are typically plug and play. However, it should be noted there are auto calibration functions constantly going on in the background of these web-cam like cameras which are actively focusing the picture, letting in an ample amount of light, and auto-balancing the colors. There are ways to do this manually but that will not be covered here.

B.2.1 Accelerometer Calibration

The data from the accelerometer at rest in Section B.1.1 suggested the orientation of the IMU was not the same as the robot due to manufacturing errors. If this slight misalignment was ignored, the accelerometer would pass biased acceleration information to the robot every measurement. This would cause errors in the orientation calculations as well as acceleration measurements of the robot. A typical mistake

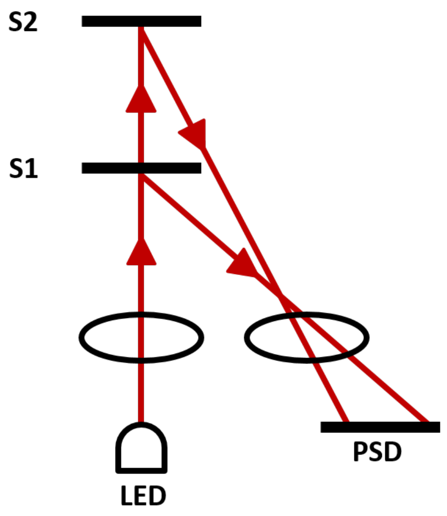


Figure B.9. A cartoon representation of how an infrared distance sensor determines distance of obstacles. A *light emitting diode* (LED) emits a beam of light through a lens which reflects off a surface (S1, S2) and hits a different location on the *position-sensible photo detector* (PSD).

made by new robotic users is to simply subtract off this bias measured when the robot is at rest. This does not solve the problem as any acceleration will not be measured in the same reference frame as the robot's. Even worse, because the gyroscope measures rotational velocity and the magnetometer measures the magnetic field about the same axes, the angular rotation and magnetic field measurements would be in the wrong reference frame as well.

To correct this, the same resting accelerometer data from Section B.1.1 will be used. First, the bias in the data must be identified. Here, it is assumed the sensor has *additive white Gaussian noise* (AWGN). This assumption means any noise in the sensor is uniform across frequencies, can be described by a Gaussian (normal) distribution, and is added to the true signal. To back up this assumption, the resting accelerometer signal in the x-direction from Section B.1.1 is analyzed. Figure B.10 shows the single-sided amplitude spectrum of the resting accelerometer data along the x-axis after a fast Fourier transform. This shows the expected spike at zero because the measurement is just a static signal as well as spikes along the rest of the frequencies from noise. This supports the white noise assumption as there are no significant peaks besides the signal. Figure B.11 displays the histogram of the measurements along the x-axis from the accelerometer at rest. While there is a small bump on the left tail of the distribution, this still supports the Gaussian noise assumption.

From the AWGN assumption, the bias in each direction is the average acceleration along each axis. This bias vector allows the user to identify the roll and pitch angle differences between the reference frame of the IMU and the resting Pheeno's reference frame. Figure B.12 is a cartoon representation of Pheeno in orientations that would create a roll, pitch, and yaw angles. From our single at rest accelerometer measurement it is impossible to calibrate all three angles. Any gravity measurement will wind up on a unit gravity sphere which only requires two angle parameters to describe fully. Another way to think about this is if the z-axis of the robot's reference frame and the IMU's reference frame were aligned, any yaw rotation would cause the same measurement of $1g$ along the z-axis. However from this it is still possible to calibrate the roll and pitch offset between the IMU and the robot reference frames.

With a bit more mathematical rigor, Pheeno's IMU oriented in Earth's gravitational field $\vec{g}$ undergoing a linear acceleration $\vec{a_e}$ in the earth's reference frame will produce a reading $\vec{M}$ of,

$$\vec{M} = \mathbf{R} * (\vec{a_e} + \vec{g}) \tag{B.1}$$

where $\mathbf{R}$ is the rotation matrix that relates the IMU's reference frame to the Earth's. Since this scenario considers orientation data where the robot is at rest on a surface parallel to the Earth's surface, the z-axis of the robot's reference frame is aligned with the z-axis of the reference frame of the Earth, this equation simplifies to,

$$\vec{M} = \mathbf{R} * \vec{g}$$

.

The rotation matrices that describe the roll, pitch, and yaw rotations shown in Figure B.12 are described as,

$$\boldsymbol{R_y}(\phi) = \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{B.2}$$

$$\boldsymbol{R_x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{B.3}$$

$$\boldsymbol{R_z}(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

Using these rotation matrices there are six unique rotation orders that can be done to produce the same rotation; $\boldsymbol{R_y}(\phi)\boldsymbol{R_x}(\theta)\boldsymbol{R_z}(\psi)$ (Roll, Pitch, Yaw), $\boldsymbol{R_y}(\phi)\boldsymbol{R_z}(\psi)\boldsymbol{R_x}(\theta)$ (Roll, Yaw, Pitch), $\boldsymbol{R_x}(\theta)\boldsymbol{R_y}(\phi)\boldsymbol{R_z}(\psi)$ (Pitch, Roll, Yaw), $\boldsymbol{R_x}(\theta)\boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\phi)$ (Pitch, Yaw, Roll), $\boldsymbol{R_z}(\psi)\boldsymbol{R_x}(\theta)\boldsymbol{R_y}(\phi)$ (Yaw, Pitch, Roll), $\boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\phi)\boldsymbol{R_x}(\theta)$ (Yaw, Roll, Pitch). These rotation orders are not commutative, like most matrix multiplications, and multiplying them out yields different matrices. To solve for roll and pitch, expanding the pitch, roll, yaw matrix multiplication yields,

$$\begin{bmatrix} \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\psi) & -\sin(\phi) \\ \cos(\psi)\sin(\phi)\sin(\theta) - \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\theta) \\ \cos(\theta)\cos(\psi)sin(\phi) + \sin(\theta)\sin(\psi) & \cos(\theta)\sin(\phi)\sin(\psi) - \cos(\psi)\sin(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

Using this matrix in Equation B.1 with $\vec{g} = [0\,0\ -1]^T$ yields,

$$\vec{M} = \begin{bmatrix} \sin(\phi) \\ -\cos(\phi)\sin(\theta) \\ -\cos(\phi)\cos(\theta) \end{bmatrix} \tag{B.5}$$

which is only dependent on the pitch (θ) and roll (ϕ) angles. Using the average measurement vector, it is possible to solve Equation B.5. It should be noted that the vector on the right side of Equation B.5 always has a length of 1, thus $\vec{M} = [M_x\, M_y\, M_z]^T$ should be normalized. Solving Equation B.5 for the pitch and roll angles yields,

$$\tan(\theta_{xyz}) = \frac{-M_y}{-M_z} \tag{B.6}$$

$$\tan(\phi_{xyz}) = \frac{M_x}{\sqrt{M_y^2 + M_z^2}} \tag{B.7}$$

Note that the pitch angle has two negatives that could be canceled. These are left in purposefully so that when solving for the angle and using and atan2() function the user does not get the wrong angle.

Now to determine the yaw angle offset, Pheeno is pitched at a known angle like in Figure B.12b. The new measured gravity vector should now be $g = [0\ -\sin(\theta_d)\ -\cos(\theta_d)]$ where θ_d is the known inclination of the robot. Substituting this into Equation B.1 yields,

$$\frac{\vec{M_p}}{\|M_p\|} = \boldsymbol{R_z}(\psi)\boldsymbol{R_x}(\theta_{xyz})\boldsymbol{R_y}(\phi_{xyz})\vec{g} \tag{B.8}$$

where M_p is the accelerometer measurement vector when Pheeno is pitched at angle θ_p. In this equation there is no $\boldsymbol{R_z}(\psi_{xyz})$. This is because previously ψ_{xyz} was not able to be solved for. Thus, it can be chosen arbitrarily. If chosen to be $\psi_{xyz} = 0$, $\boldsymbol{R_z}(\psi_{xyz})$ is a 3x3 identity matrix. To simplify this equation, substitute,

$$\vec{v} = \boldsymbol{R_x}(\theta_{xyz})\boldsymbol{R_y}(\phi_{xyz})\vec{g}$$

.

This allows for an explicit solution for ψ,

$$\psi = \arcsin\left(\frac{\frac{M_{px}}{\|M_p\|}v_y - \frac{M_{py}}{\|M_p\|}v_x}{v_x^2 + v_y^2}\right). \tag{B.9}$$

The rotation matrix which aligns the IMU's reference frame with Pheeno's is then,

$$\begin{aligned} \boldsymbol{R_{Calibration}} &= (\boldsymbol{R_z}(\psi)\boldsymbol{R_x}(\theta_{xyz})\boldsymbol{R_y}(\phi_{xyz}))^{-1} \\ &= \boldsymbol{R_y}(-\phi_{xyz})\boldsymbol{R_x}(-\theta_{xyz})\boldsymbol{R_z}(-\psi). \end{aligned} \tag{B.10}$$

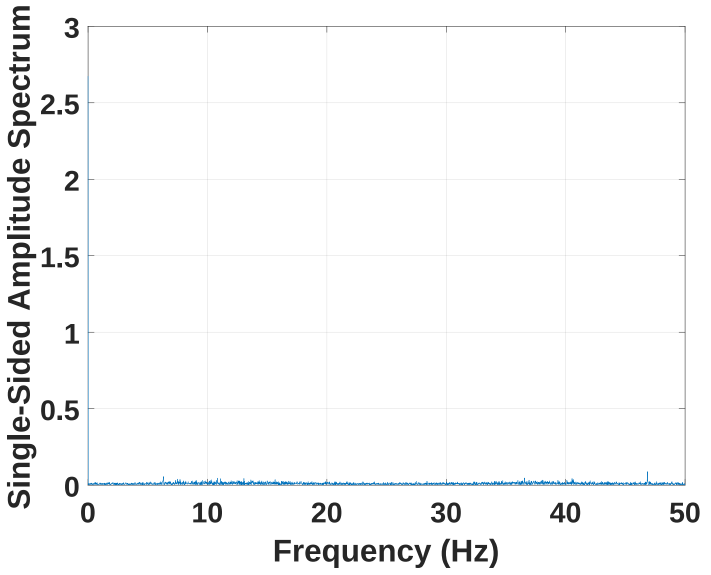


Figure B.10. The single-sided amplitude spectrum of the resting accelerometer data along the x-axis.

These angles should be saved. To use this calibration properly, any measurement should be rotated through this matrix.

B.2.2 Gyroscope

Gyroscopes typically don't need much calibration. Again, an assumption about the sensor noise should be made. The gyroscope is assumed to have AGWN like the accelerometer. The analysis supporting this assumption is the same as provided in Section B.2.1 and will not be shown here. Calibrating a gyroscope only requires subtracting a resting bias. This involves simply averaging the gyroscope measurements in each direction when Pheeno is at rest on a level surface, then saving this information and subtracting the average from any reading.

It should be noted that the axis the gyroscope measures rotation about are the same as the accelerometer. This means any measurement made by the gyroscope should be transformed to Pheeno's reference frame through the rotation matrix found in Section B.2.1. It is up to the user whether to subtract an average of the transformed measurements from transformed measurements or subtract the average untransformed measurements from untransformed measurements then transforming the result. The latter is preferred but both are valid.

B.2.3 Magnetometer

Magnetometers are typically the sensor that requires the most frequent calibration. Like the accelerometer, the magnetometer measures a theoretically constant magnetic field vector with respect to the magnetometer's orientation. However, these readings

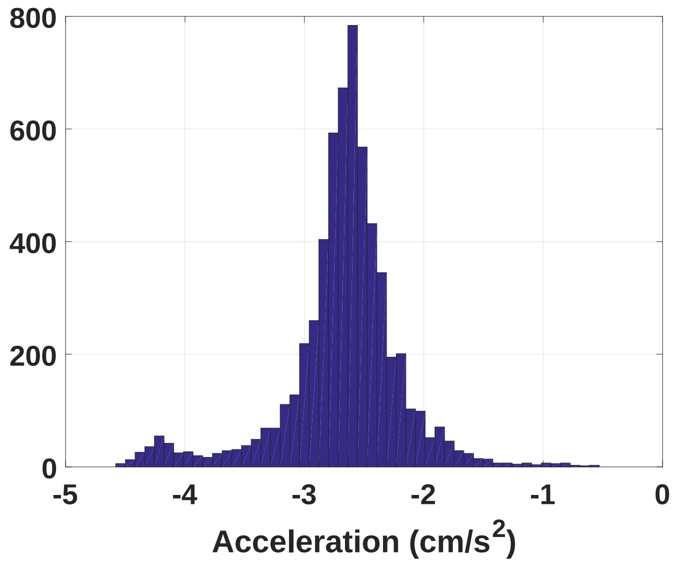


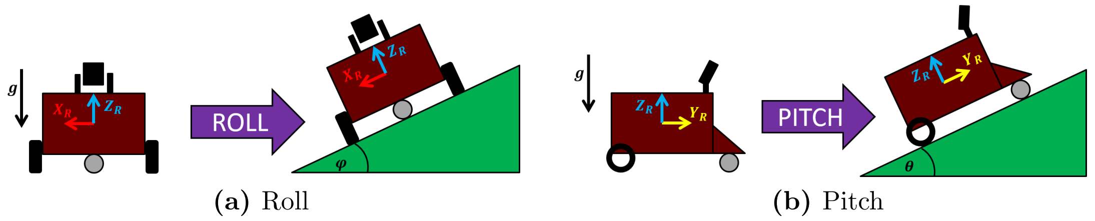
Figure B.11. Histogram of the resting accelerometer data along the x-axis.

can easily be thrown off by large metal beams or wires with large electrical currents in buildings. For larger robots, the motors can cause throw off large magnetic anomalies, however, Pheeno uses very small low current motors so their effects can be ignored. In an ideal scenario, the magnetometer would just read the magnetic field of the earth, which is different depending on the user's location around the globe. With that said, even if the magnetometer were calibrated at the factory it was produced in, those calibration values would be invalid other places. For this section, it will be assumed the magnetometer is only reading the Earth's magnetic field. Ways to overcome or recognize anomalies in the magnetic field are possible but will not be covered.

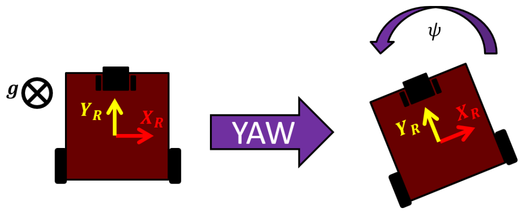
The ideal response surface for a 3-axis magnetometer is a sphere centered at the origin. This means if the user rotates the magnetometer while taking readings, a well calibrated magnetometer will produce point readings on a sphere with a radius equal to the magnitude of the magnetic field present. Figure B.13a shows uncalibrated data taken from the magnetometer on-board Pheeno. The data was taken at 1Hz increments while the sensor was slowly rotated about each axis. The plot is a various 2D slices from the 3D sphere to illustrate the fact that the slices are not centered at the origin and the response sensitivity is different along each axis (they are not equal radius circles). These are often referred to as *hard iron* and *soft iron* errors or biases, respectively.

Hard iron biases are typically the largest source of error and usually the easiest to account for. To correct this, record the maximum and minimum field measurements along each axis while rotating the magnetometer. Once the user is satisfied the magnetometer has taken sufficient measurements in each orientation, the average between the max and min magnetometer reading along each axis is equal to the hard iron bias in each direction.

To correct for the soft iron bias correctly, the response surface from the raw



(a) Roll **(b)** Pitch



(c) Yaw

Figure B.12. Cartoon of Pheeno at different orientations representing the roll, pitch, yaw convention.

measurements of the magnetometer should be deconstructions into their elliptical principle axis to create a 3×3 correction matrix to transform the general ellipsoid to a sphere. This is pretty involved and more importantly can be approximated in a much easier way. An explanation of the mathematics behind the full calibration method can be found at [108].

A decent approximation for this process is to simply scale the response along each axis with the maximum and minimum measurements already calculated previously. First a scale factor, s, is calculated,

$$s = \frac{M_{xL_{avg}} + M_{yL_{avg}} + M_{zL_{avg}}}{3}$$

where,

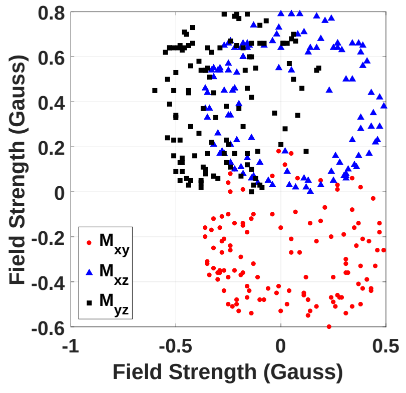
$$M_{xL_{avg}} = \frac{M_{x_{max}} - M_{x_{min}}}{2} \qquad M_{yL_{avg}} = \frac{M_{y_{max}} - M_{y_{min}}}{2} \qquad M_{zL_{avg}} = \frac{M_{z_{max}} - M_{z_{min}}}{2}$$

This average scale factor is then projected onto each axis as a gain,

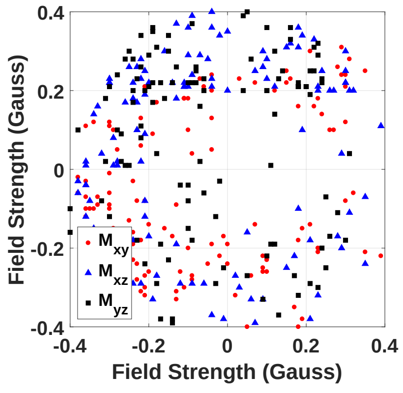
$$s_x = \frac{s}{M_{x_{avg}}} \qquad s_y = \frac{s}{M_{y_{avg}}} \qquad s_z = \frac{s}{M_{z_{avg}}}$$

This approximation of the full calibration is a simple orthogonal rescaling; equivalent to a diagonal 3×3 calibration matrix.

The calibrated data is then found by subtracting the hard iron bias from the raw measurement in each axis and scaling the difference. For example the calibrated magnetometer reading,$\vec{M_{cal}}$, of a raw measurement, $\vec{M_{raw}} = [M_x \; M_y \; M_z]$, with hard iron bias vector $\vec{b_{HI}} = [b_x \; b_y \; b_z]$ and scaling matrix $\boldsymbol{G} = diag(sx, sy, sz)$ would be,



(a) Uncalibrated



(b) Calibrated

Figure B.13. Several slices of the magnetometer measurements along the principle plains.

$$\vec{M_{cal}} = \boldsymbol{G}(\vec{M_{raw}} - \vec{b_{HI}}) \tag{B.11}$$

Figure B.13b shows data taken after calibration. Compared to the uncalibrated data, the circles are now concentric and approximately circular.

B.2.4 Motor Encoders

Motor encoders do not require any sort of calibration. However, the data (which is the number of counted rotations) has to be transformed to some usable units. For generality, the encoders will be assumed to count n times per rotation of the extended back shaft and there will be a g gear ratio from the measured extended back shaft to the wheel shaft. From this assumption, the numbers of radians the drive shaft has traveled per count, x, would be,

$$2\pi \frac{x}{ng}. \tag{B.12}$$

From this transformation, the linear distance a wheel would travel (assuming no slipping) and the rotational velocity of the wheel shaft can be calculated.

An important note that can cause some issues is these counts become very large very quickly. Usually these counts are stored in an integer variable on board your robot's processor which only allows n-bit ranges (2^n). In Pheeno's case signed integers are stored in a 16-bit variable. This means signed integers can only be stored from $[-32768, 32767]$ once this range is surpassed, the number will *roll over*. This means if the count were supposed to increment to 32768 (outside the range) it would actually go to -32768. This can cause velocity and position estimates to go haywire if not expected. Standard conditional statements inside the robot's code can alleviate this issue but users should still be aware of this issue.

B.3 Sensor Fusion for State Feedback

This section will go through complementary filters that are used on board Pheeno to fuse the sensor measurements to determine the robot's state (position, velocity, orientation, etc.). It will also briefly go into using the sensors that sense the robot's surroundings (IR distance sensors and camera) to determine reliable information about the environment.

Section B.3.1 will go over *complementary filtering* of sensors in a very basic sense with limited mathematics to give the user an intuition. Section B.3.2 will introduce complementary filters in a more mathematically rigorous sense. Section B.3.3 briefly describes the complementary filters used on board Pheeno.

B.3.1 Complementary Filter (Basic)

A complementary filter is an easy to implement sensor filter that joins two state estimates together. These estimates are required to be accurate on different time scales. Meaning that one must be able to capture fast and aggressive changes while the other maintains a consistent reading that will stay correct for long periods of time after the aggressive maneuver (the measurement does not drift). For example, to determine the robot's orientation about one axis, gyroscope measurements can be combined with magnetometer or accelerometer measurements. The gyroscope is able to pick up quick motions well but after long periods of time will drift and its angle estimates will become wrong, while the accelerometer and magnetometer will determine the correct orientation when the motions are less aggressive.

Complementary filters are very similar to *proportional, integral, derivative* controllers in nature. They are easy and computationally inexpensive to implement on micro controller while still yielding extremely accurate measurements. Their major drawback is they may only fuse two measurements and do not give any intuition about how wrong their measurement estimates may be. The measurements are also required to have strengths in opposite frequency domains which is not always possible. Complementary filters are used on board Pheeno to estimate the robot's roll, pitch, and yaw angle as well as body linear velocity estimates. For this small, relatively slow robotic platform, complementary filters are found to be just as effective as more advanced Kalman filters at a fraction of the computational expense.

These filters work by using *high pass* and *low pass* filters simultaneously. High pass filters allow high frequency signals while suppressing low frequency signals (such as drift of the gyroscope). Low pass filters act the opposite way by allowing low frequency signals while suppressing high frequency contents (like vibrational noise picked up by the accelerometer). In its most simple form, a first order complementary filter takes the form,

$$m_{filter} = a m_{fast} + (1-a) m_{slow} \tag{B.13}$$

where, m_{filter} is the filtered measurement, m_{fast} is the measurement that is accurate over short timescales, m_{slow} is the measurement that is accurate over long timescales, and $a \in (0,1)$ is the filter gain that is to be chosen.

Choosing a properly requires a bit of mathematics to fully understand but can be chosen and tweaked based on some intuition as well. Equation B.13 can be looked at naively as an average. Two measurements are being averaged based on the users confidence in them during short time periods. The higher a is chosen, the more the filtered measurement will rely on the fast measurement and will take longer to return to the slow measurement (which will be true when the aggressive maneuver has ended). The lower a is chosen, the more the filtered measurement will rely on the slow measurement and will be more prone to short term noise. While this is not how these filters were formulated (the idea behind them was not averaging in this sense), it is good intuition to design these filters. The optimal choice for a in a scenario will result in a filtered measurement that is able to capture very fast changes in the measurements as well as not have that measurement drift.

As a more concrete example, consider an accelerometer and gyroscope measurement being fused to determine a roll angle estimate. Using the first order complementary filter, the roll estimate at time t after a time step of Δt could be determined by,

$$rollAngle(t) = 0.9(rollAngle(t - \Delta t) + gyroRollRate\Delta t) + 0.1accRollAngle.$$

This example essentially updates the new roll angle estimate, *rollAngle*, by combining 90% of the gyroscopes update, *gyroRollRate* with 10% of the accelerometer's update, *accRollAngle*. This combination will ensure the measurement won't drift due to the accelerometer limiting the integration error and will still be accurate in short term estimates due to the majority of the updated estimate coming from the gyroscope.

B.3.2 Complementary Filter (Advanced)

While Section B.3.1 gives a basic understanding of the complementary filter, this section looks at it with slightly more mathematical rigor. Figure B.14 shows an example block diagram of a complementary filter fusing a gyroscopic angle measurement with an accelerometer measurement. The gyroscopic measurement is integrated once to yield an angle then high pass filtered to avoid drift. The accelerometer measurement is low pass filtered to avoid the high frequency noise that plagues accelerometer measurements during fast rotations. When added these measurements complement each other's weaknesses.

Using a first-order high pass and low pass filter, the transfer function in continuous time is,

$$\theta = \frac{1}{1+Ts}\theta_{acc} + \frac{Ts}{1+Ts}\frac{1}{s}\dot{\theta}_{gyro} = \frac{a+Tw}{1+Ts}. \tag{B.14}$$

where T determines the cut-off frequencies. This now must be transformed to discrete time, as robots do not operate in continuous time. Using backwards difference, $s = \frac{1}{\Delta t}(1 - z^{-1})$, in Equation B.14 leads to the final equation,

$$\theta(k+1) = \alpha(\theta(k) + \dot{\theta}_{gyro}\Delta t) + (1-\alpha)\theta_{acc}, \tag{B.15}$$

where, $\alpha = \frac{T}{T+\Delta t}$. Note, this is the same as Equation B.13.

This still begs the question, how should the cut off frequency be chosen? The answer is an optimization problem which is beyond the scope of this paper and usually

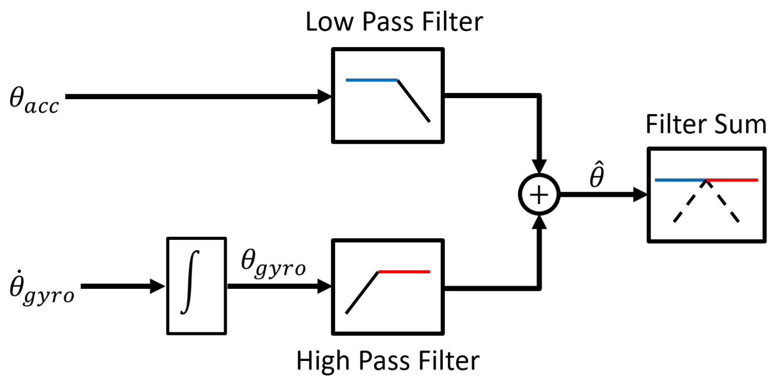


Figure B.14. Block diagram of a complementary filter fusing a gyroscope angular measurement with an accelerometer angular measurement.

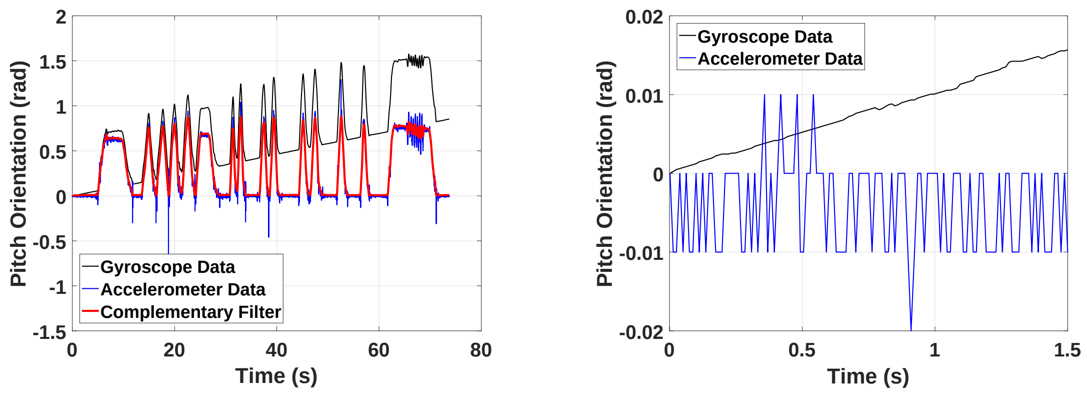
needs to be adjusted in application if the optimization is done. The filter needs to be designed such that there are constant amplification and small phase loss of all measurements. More specifically, this means setting the cut off frequency high enough such that the largest range of frequencies is measured by the accurate but noisy sensor with slow dynamics (accelerometer, magnetometer, etc.). This avoids the drift typical in faster sensors. When motions occur that are at higher frequencies than the dynamics of the slow sensor, the cut off frequency should be set low enough such that the expected phase loss of the slower sensor is compensated by the faster sensor (gyroscope, encoders, etc.).

It should be noted on a small slow robotic platform like Pheeno, it is possible to set only one cut off frequency and get reliable measurements. However, in faster more agile systems like quad rotors, more advanced techniques like a gain-scheduled complementary filter are required. This filter switches it cut off frequency or other design parameters depending on how aggressive a measured action is (acceleration measurements). There is also another representation of a second order complementary filter based on the Mahoney and Madgwick filter for more agile systems that can be used over a first order filter to capture more advanced dynamics [87, 88, 144].

B.3.3 Complementary Filter Design for Pheeno

Pheeno uses complementary filters to determine its orientation angles (roll, pitch, yaw) as well as its linear velocity. This involves fusing the robot's sensors with slow dynamics (accelerometer, magnetometer) with its fast drifting sensors (encoders, gyroscope). Using the same convention established in Section B.2.1, the accelerometer's measurements is combined with the gyroscopes measurements to determine roll and pitch angles of the robot. The magnetometer angular measurements are combined with the gyroscopic measurements to determine the yaw angle of the robot (heading). The accelerometer is combined with encoders to determine the robot body's velocity.

This design mostly comes from experience working with the sensors and in application the best way to determine filter coefficients is to tune on a data set and gain intuition about the sensors from their data outputs. It is possible to model the sensor



(a) Pitch Angle Estimate **(b)** Zoomed in Pitch Angle Data

Figure B.15. Orientation angle estimates from several sensors and a filtered combination while Pheeno was manually pitched about 45° at different rates. (a) Accelerometer (blue), gyroscope (black), and complementary filter (red) estimate of Pheeno's pitch angle. (b) A zoomed in section of (a).

and determine these cut off frequencies in a more mathematically rigorous fashion but, when working with inexpensive robots with readily accessible parts, this method is faster with just as valid results.

Figure B.15 shows the time evolution of the angular orientation estimate for Pheeno about its x-axis (pitch) as it was manually pitched from a level table to about 45° over different periods. The filter take the form of Equation B.15. To determine the parameter of the complementary filter equation, $\alpha = \frac{T}{T+\Delta t}$, refer to Figure B.15b. This data was sampled at 100Hz so $\Delta t = 0.01$. This leaves T, the time constant of the system. A rule of thumb used here is to determine the time when the fast sensor, in this case the gyroscope, drifts out of the error of the slow sensor, in this case the accelerometer, when the system is at rest and the slow sensor's measurement is correct. For this case Figure B.15b shows the gyroscope's pitch estimate drifting outside the 0.01 rad error envelope of the accelerometer after 1s. Plugging this into the equation for α yields, $\alpha = 0.99$. A similar process is done for each axis of rotation on the robot.

Further analysis of Figure B.15 gives a good idea of how this fusion is performing. It is very apparent the pitch estimate from the gyroscope (black line) is drifting away from the true rotation range but is still capturing the rotation rate correctly especially during higher frequency rotations like between $t = 65$s and $t = 70$s. The accelerometer's estimate (blue line) is not drifting but there are very apparent spikes when the robot makes contact with the table again and during the high frequency rotation between $t = 65$s and $t = 70$s. The complementary filter with $\alpha = 0.99$ captures the best of both of these sensors. There is no apparent spiking or noise from the accelerometer measurements and the estimate is not drifting.

APPENDIX C

CONTROL OF THE PHEENO ROBOT FOR NAVIGATION

This appendix describes the controller design for the motors that drive Pheeno as well as the higher level controller structure that navigates the robot from one location to another. To create these controllers accurate models of the motors and robot are needed. The choice of models and fitting will be discussed in Section C.1 and Section C.3. The controller design sections will address which sensors are used to calculate the desired feedback but the details are discussed in the observer design document and thus, will not be talked about here.

The simplest possible models are used to design controllers for Pheeno. To understand this choice, refer to the passage from George Box's 1978 paper:

> *Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law $PV = RT$ relating pressure P, volume V and temperature T of an "ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules.*
>
> *For such a model there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?"*[25].

The better known section header of this passage is,

> *All models are wrong but some are useful.*

There is a need for the model to capture the properties of the system that are useful without overparamertizing or overelaborating the model. Using more intricate models than are needed makes the controller design more complicated and, more importantly, over fits the data used to parameterize the system which can cause this model to describe a certain data set more than the system it is meant to represent. With the use of feedback control, even simple models that do not describe the system completely can be controlled in desirable fashion.

C.1 Modeling the Motors

The first and arguably most important step towards controlling Pheeno's motion is designing a fast controller for the motors. To do this an accurate model of the motors must be created. Typically, this would be done by parameterizing the standard second order transfer function for a direct current (DC) motor,

$$P(s) = \frac{\dot{\Theta}(s)}{E_a(s)} = \frac{K_T}{(Js+b)(Ls+R)+K_BK_T}. \tag{C.1}$$

where K_T is the gain relating the armature current to armature torque, K_B is the gain relating the rotational velocity of the rotor to the back EMF in the armature circuit, J is the inertial of the rotor, b is the viscous friction acting on the rotor, L

is the inductance in the armature circuit, and R is the resistance in the armature circuit.

However, for micro-metal gear motors, like the ones used on Pheeno, $L << R$. This allows for the second order system to be approximated by a first order system by setting $L = 0$ yielding,

$$P(s) \approx \frac{K_T}{R(Js+b)+K_BK_T} = \frac{K}{Ts+1} \tag{C.2}$$

where,

$$K = \frac{K_T}{Rb+K_TK_B} \qquad T = \frac{RJ}{Rb+K_TK_B}.$$

To fit these parameters, the motors will be black box modeled using MATLAB's system identification toolbox [86]. Two motors attached to Pheeno were given step voltages and sinusoidal voltages of frequencies between 0.1 and 50 Hz. The max sinusoidal frequency was chosen following the Nyquist sampling criterion which states a signal may only be recreated if it is sampled twice as fast as its highest frequency component. The rotor rotational velocities were measured with its encoders at a rate of 100Hz. The input and output time plots are shown in Figure C.1. From these plots, it is apparent there is a large magnitude drop off as the frequency of the input signal increases. This is expected in a natural system like the DC motor.

This data is given to the system identification toolbox to fit a first and second order continuous model of the motor. As expected, the first and second order transfer functions yield equal "goodness of fit" to the data ($\sim 80\%$). This fit is a little low but acceptable as a linear model is unable to represent nonlinear friction effects on the motor rotor. The first and second order system both have a pole at ~ 36 rad/s with the second order system also having a pole at $\sim 60,000$ rad/s. From this fitting it is obvious the dynamics of the motor are dominated by the single pole. The location of the pole fit also makes sense as the magnitude of the sinusoidal response begins to decay around 30 rad/s as is expected with roll off caused by a pole.

This leaves the first order approximation of the system.

$$P(s) = \frac{2.9876}{s+36.07} \tag{C.3}$$

The model compared to a set of validation data can be seen in Figure C.2.

C.2 Controlling the Motors

C.2.1 Continuous Consideration

From this plant it is possible to design a controller so that the motor responds in a desired way. For this application it is desirable to make the motors follow step commands (and rejects step disturbances) as fast as possible with no overshoot. From the internal model principle, a proprtional integral (PI) controller is enough to satisfy these requirements.

This formulation leads to a standard feedback problem that must be solved. Figure C.3 shows a block diagram of standard negative feedback loop with the symbols

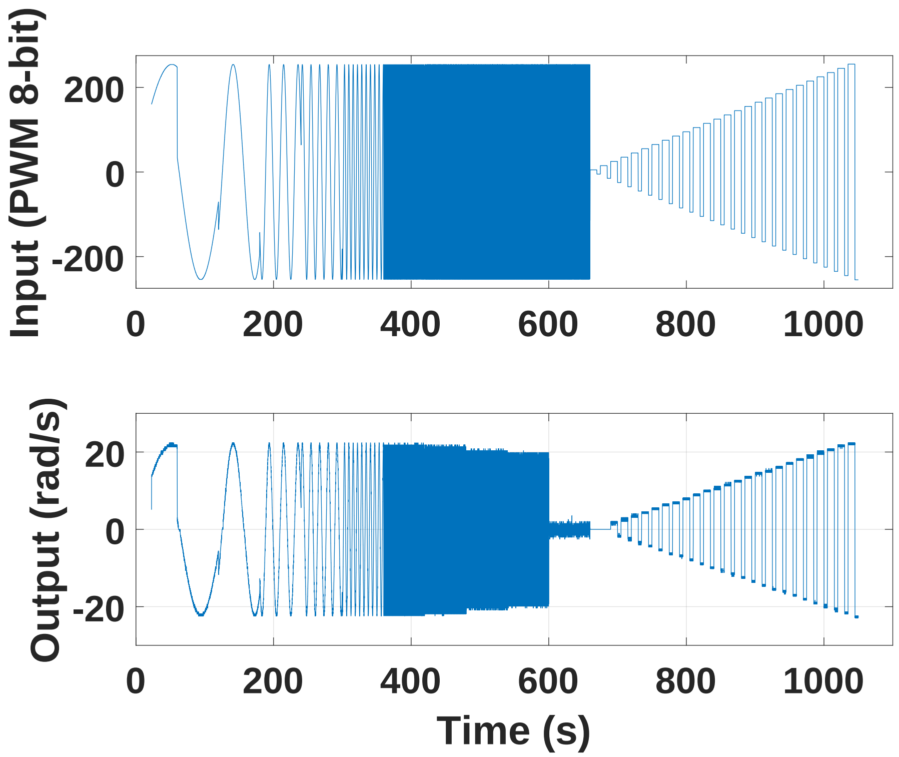


Figure C.1. The voltage input (*top*) and rotor velocity output (*bottom*) signals for a DC motor.

that will be used here. The plant, $P(s) = \frac{2.9876}{s+36.07}$, was found previously, a PI controller, $K = g\frac{s+a}{s}$, needs to be designed, and a pre-filter, $W = \frac{a}{s+a}$, needs to be incorporated to reduce the overshoot caused by the zero of the controller. Here, the sensor dynamics, H, are assumed to be ideal.

Typically, in a classroom setting, a pole placement method would be employed to match the nominal closed loop system to some desired closed loop system. In this case, a slightly different method will be employed. The plant here is a nominal representation of the system. This means there are higher order dynamics present in the system that are not modeled so the gain chosen cannot be very high and the dominant pole modeled here can be slightly or significantly off. First, a will be chosen with these real world constraints in mind, then g will be chosen such that there is no overshoot and the motors spin up as fast as possible.

First, notice the pre-filter is a stable low pass filter (presuming $a > 0$). This means it will create a natural delay on the reference signal depending on how small a is chosen. Thus a should be chosen to be large such that this delay does not significantly slow down the motor's response to low frequency reference commands and does not restrict the magnitude of desired higher frequency signals. However, it should be chosen low enough that the integral term of the controller does not become too large and exacerbate the integral windup problem caused by motor saturation (which will be addressed later). Secondly, the proposed open loop system's (KP) root locus will result in two different looking root locus depending on how a is chosen. Figure C.4 shows an example of both root locus. Note choosing a larger than the dominant pole of the modeled motor has two critical damping points. The left critical dampin point requires a much larger gain and thus will not be considered. The right critical point will shift left as a is chosen closer to the pole of P making the closed loop system's response faster. To make the system as fast as possible, choose $a = 36.07$ so it cancels the pole of the motor. Typically, this is not a good idea because of modeling errors that cause this cancellation to not be true in reality. However, because it is a very stable pole, this choice does not have such severe consequences as model error will result in one of the two root locus represented in Figure C.4.

The gain, g, is chosen such that the rise time of the system is < 0.5s. Figure C.5 shows a simulated step response of this controller and modeled plant which behaves as expected.

C.2.2 Dealing with Integral Windup

Integral windup is a problem that occurs in controllers containing integral terms when a large change in the reference command occurs. For example, if Pheeno is suddenly commanded to go from rest to full speed there will be a large error initially that will get smaller as Pheeno begins to reach its top speed. However, during this time the integral term of the controller will be compounding this error and cause an overshoot until enough error has occurred in the opposite direction to offset it. This gets worse when saturation of actuators are considered. In this case, a naive controller can require an output larger than what can be produced by the actuator. This causes integration error to continue to compile without knowledge that the error it is trying to rectify is beyond the capabilities of the system.

There are several ways to address this issue. In Pheeno, this issue is addressed by adding a secondary feedback loop that limits the integral term when motor saturation has occurred. Figure C.6 shows this feedback loop in block diagram form. This feedback loop only kicks in if the controller's desired output is higher or lower than the actuator can output. When saturation occurs, the feedback loop keeps the integral term from compounding when the error cannot be reduced. Typically the tracking gain, K_t, is chosen to be equal to the integral gain, K_i, but higher values can cause better performance [21].

C.2.3 Discrete Time Adaptation

It would be naive to just throw this controller onto the robot and assume the motors will respond as they were designed to. If the controller were designed in the continuous domain without accounting for the delays created by the sampling of the microcontroller, it is very likely the control would be unstable at worse or not exhibit the designed properties at best. The Teensy microcontroller can easily perform control loops at 100Hz. Thus a sampling time of 0.01 is chosen to design the motor control around.

First, the plant should be transformed from the continuous domain to the discrete domain. There are many options to transform a continuous plant represented in the s-domain to the discrete time z-domain. For the plant, a zero-order hold (ZOH) conversion is chosen as that best represents the type of hold circuit that will be used in sampling the motor. The controller designed in the continuous case is transformed using the bilinear transformation to better approximate the continuous behavior of the controller in the discrete space.

This control case is ideal as the sampling time is very fast compared to the desired rise time. Thus, the continuous system is very close to the discrete system. Figure C.7 shows the step response of the discrete designed system with the continuous designed system. However, in general this will not be the case and the continuous system would need to be augmented with an additional gain to get the desired response characteristics or redesigned entirely.

To validate this control, several known commands are given to two different motors on two different robots. The model's prediction is compared to the motor outputs in Figure C.8. For large jumps in the reference command, like the last two step commands, the integral windup overshoot is apparent but not overwhelming. This could be remedied with a less aggressive controller or putting a more restrictive low pass pre-filter on the reference commands. This would slow down the response considerably which is undesirable.

C.3 Modeling the Robot

Pheeno is by default a differential drive robot. This means each wheel can be controlled independently to produce desired motion. However, this makes the robot a coupled system resulting in a multi input multi output system which can be tricky to control properly. To simplify this, a decoupled kinematic model is used to represent Pheeno's motion and control its position and orientation in a global reference frame. This can be done since Pheeno is so light and its motion is dominated by the motor torques. An extremely in depth analysis about when this assumption can be used is done by Anvari [12] in his master's thesis.

Consider Pheeno in an inertial reference frame $\{X_o, Y_o\}$ as shown in Figure C.9. Pheeno's basic motion model is, what is commonly referred to as, the unicycle model.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{C.4}$$

This model transforms the robot's linear velocity, v, and rotational velocity, w, in Pheeno's reference frame to velocity states in the inertial frame. However, the robot's linear and rotational velocity cannot be controlled directly so another transformation is needed linking the rotational velocity of the wheels, v_R and v_L, to v and w. This relation is derived more thoroughly in Malu and Jharna Majumdar [91].

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{L}{2r} \\ \frac{1}{r} & \frac{-L}{2r} \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{C.5}$$

Here, r is the wheel radius and L is the axle length of the differential drive robot. Combining Equation C.4 and Equation C.5, yields the final relation between the wheel speeds of the robot and the velocity states in the inertial reference frame.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \\ \frac{r}{L} & \frac{-r}{L} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \tag{C.6}$$

However, it is much more intuitive to use the unicycle model (Equation C.4) thus control will be done to create reference linear velocities, v, and rotational velocities, w, for the robot to follow. These will then be transformed to motor velocity commands using Equation C.5.

In discrete time, this unicycle model takes the form,

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k} + \begin{bmatrix} \Delta t\cos(\theta_k + \frac{\Delta\theta_k}{2}) & 0 \\ \Delta t\sin(\theta_k + \frac{\Delta\theta_k}{2}) & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{C.7}$$

This unicycle model has slight changes to the orientation model that can be found in a paper by Kiriy and Buehler [73]. Using this model over the usual one showed vast improvements in dead reckoning navigation for Pheeno.

C.4 Controlling the Robot's Motion

The approach to using the unicycle model to navigate Pheeno from an initial position to a goal position described here is using a layered architecture. This means using a high level planner to design way points for the robot to pass through, which are then translated to linear and rotational velocities of the robot, which are finally put through the fast PI controller of the motors. This section focuses on the middle component which decides the set points for the linear and rotational velocities.

Assume the high level planner has given an initial desired position $\vec{u} = [u_x \ u_y]^T$. From a Lyapunov stability analysis in [91] the controllers which produce stable global position tracking are,

$$v = K_p \rho \cos\alpha \tag{C.8}$$

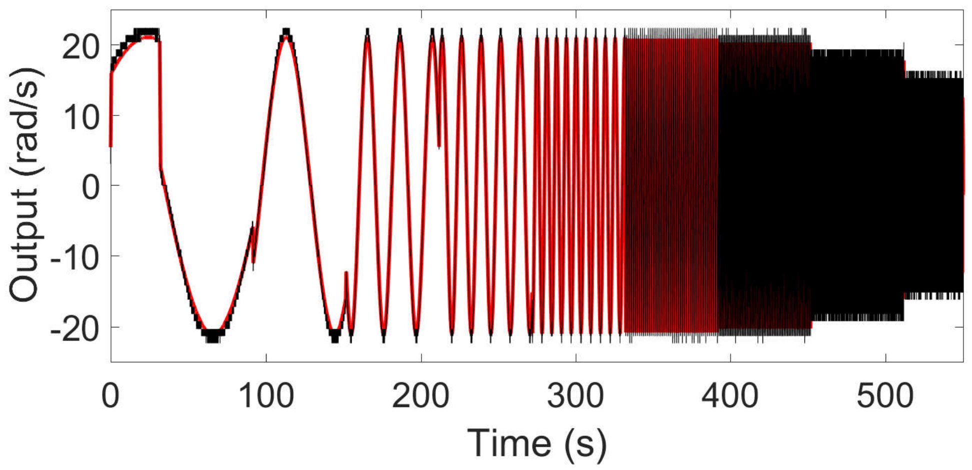
$$w = K_p \sin\alpha\cos\alpha + K_\alpha \alpha \tag{C.9}$$

where $K_p > 0$ is a gain associated with radial distance error from the goal location, $\rho = \sqrt{(u_x - x)^2 + (u_y - y)^2}$, and $K_\alpha > 0$ is a gain associated with the robot's heading error from the goal direction vector, $\alpha = atan2(\frac{u_y - y}{u_x - x})$. It should be noted the heading error, α, is bounded $(-\pi, \pi]$ which limits how large w can get. However, the radial distance error, ρ, is unbounded. Thus, it is typical in application to either know the bounds of ρ when designing K_p as a constant or choosing K_p to be of the form,

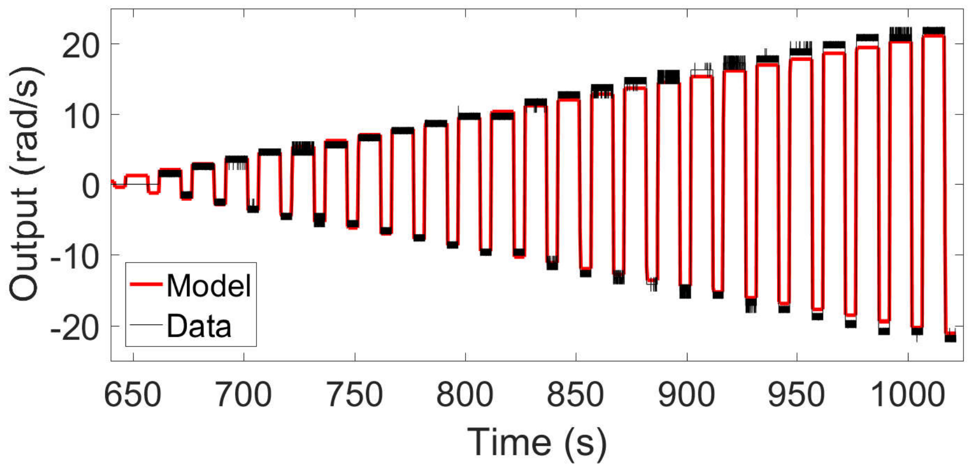
$$K_p = \frac{v_0(1 - e^{-a\rho^2})}{\rho} \tag{C.10}$$

which limits the maximum linear velocity of the robot to a designed v_0. When designing for any application, the gains should be chosen carefully to avoid wheel slip caused by high accelerations. The controllers should also operate slower than the rise time of the motor controller ($\sim$ 0.1s) so the motors have a chance to produce the desired linear and rotational velocities demanded by the higher order controller.

Alternatively, a PID controller can be substituted for this non-linear controller to track high level headings and velocities. For this, the unicycle model should be linearized as a series of short straight motions to design around. As with all applied PID controllers, integral windup needs to be addressed and the controller should be designed with actuator bounds in mind.



(a)



(b)

Figure C.2. The first order DC motor model (*red*) compared to validation data (*black*).

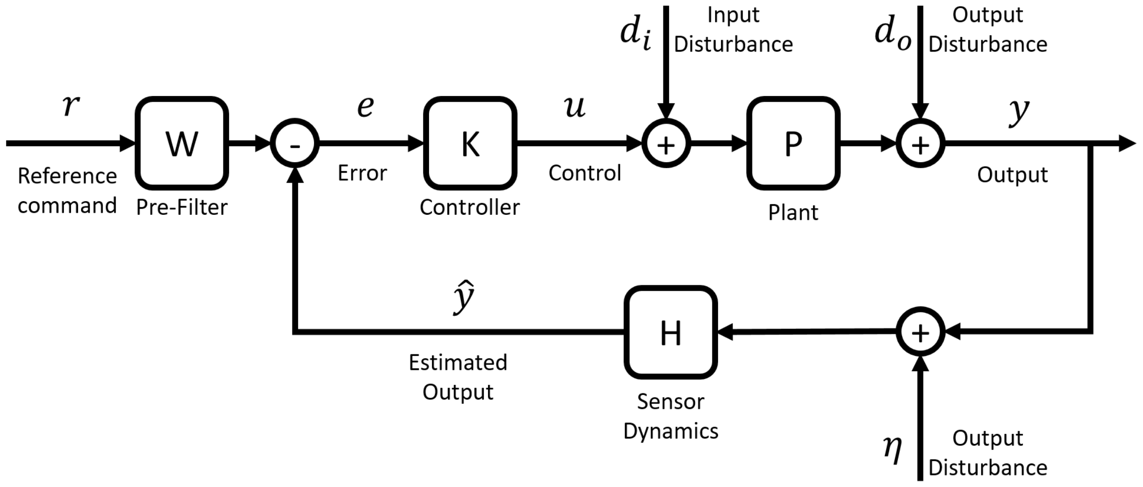
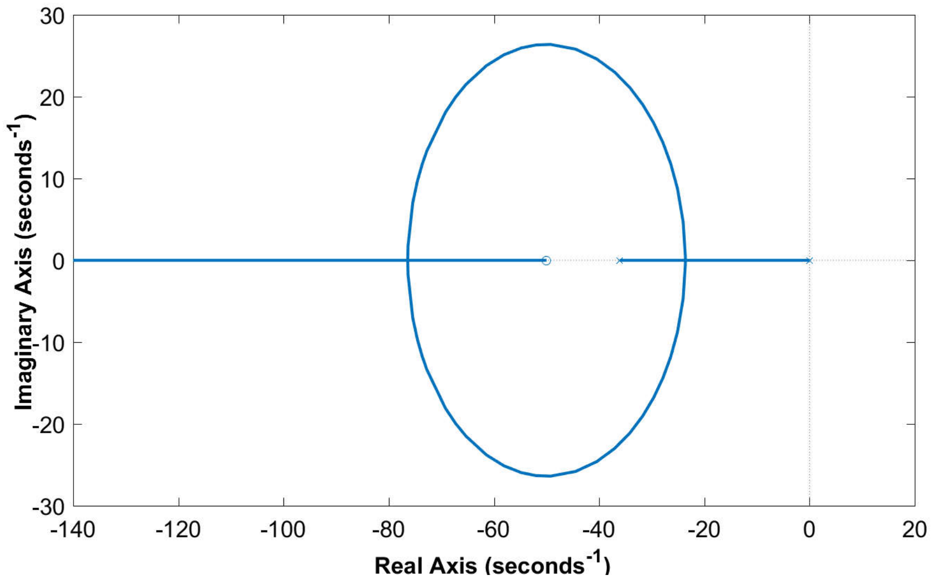
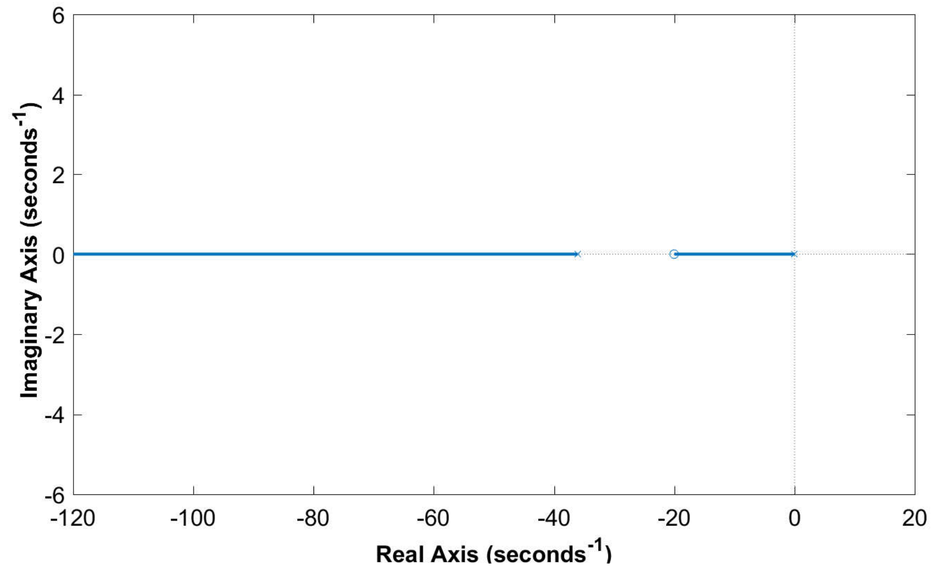


Figure C.3. A block diagram representation of a standard negative feedback loop.



(a)



(b)

Figure C.4. The root locus of Pheeno's motors where the zero of the controller is chosen larger (*top*) and smaller (*bottom*) than the modeled pole of the motor.

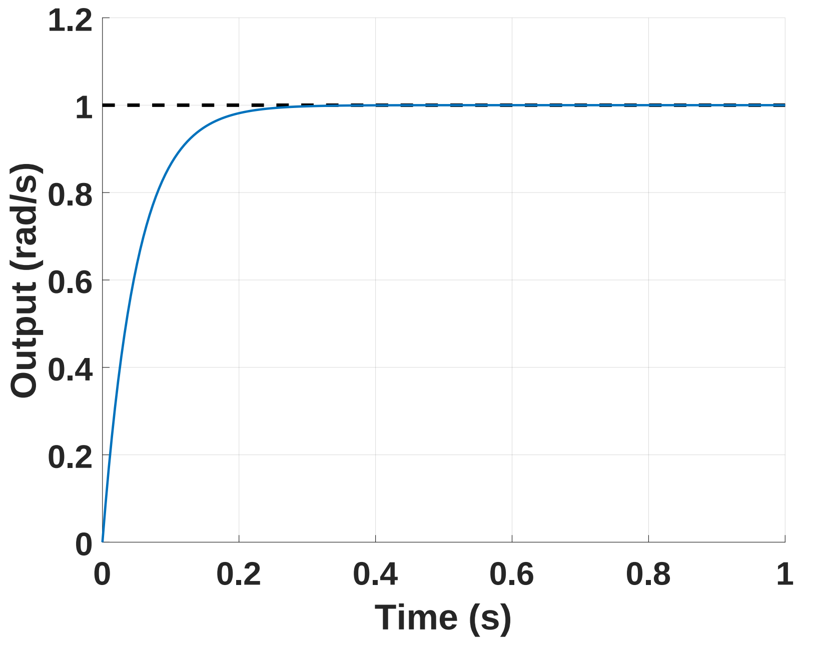


Figure C.5. The simulated step response of the designed continuous motor control loop.

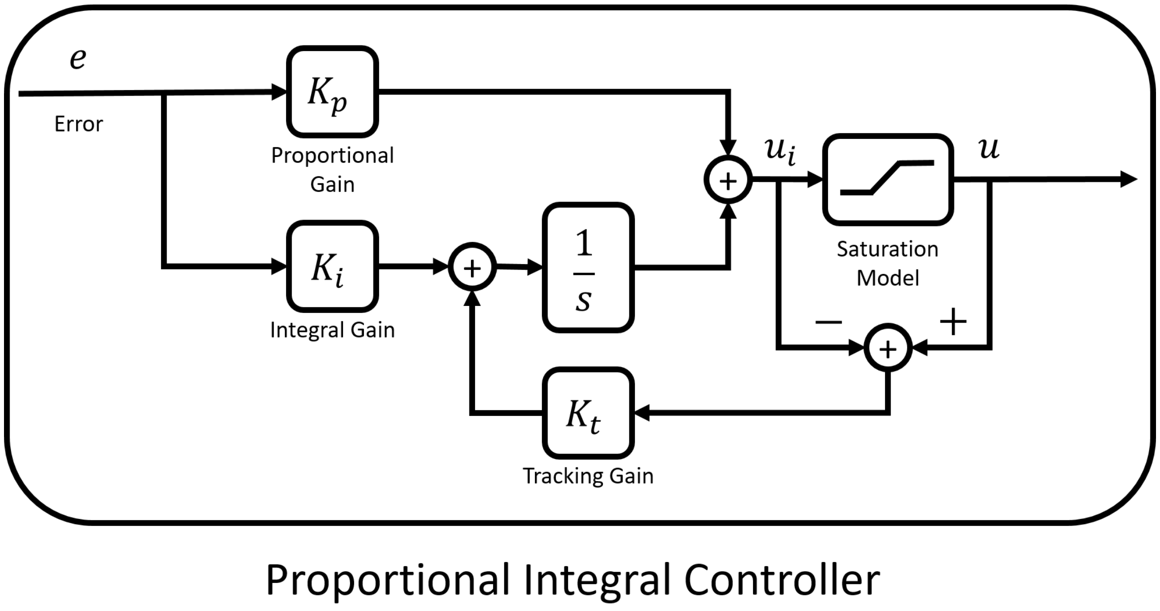


Figure C.6. Comparison of model to motor output for several reference commands.

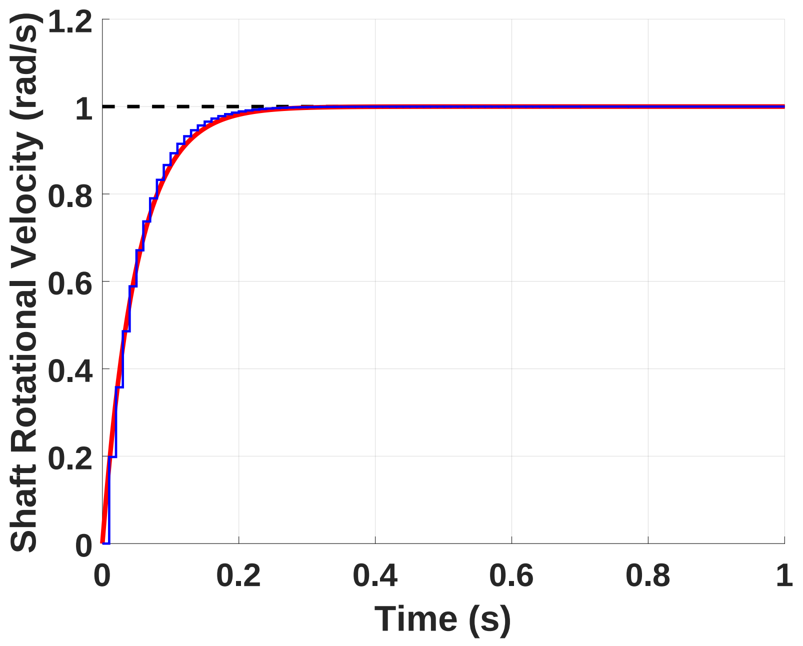


Figure C.7. Comparison of simulated step response of the designed continuous and discrete motor control loop. The *blue* line shows the discrete time step response and the *red* line shows the continuous step response.

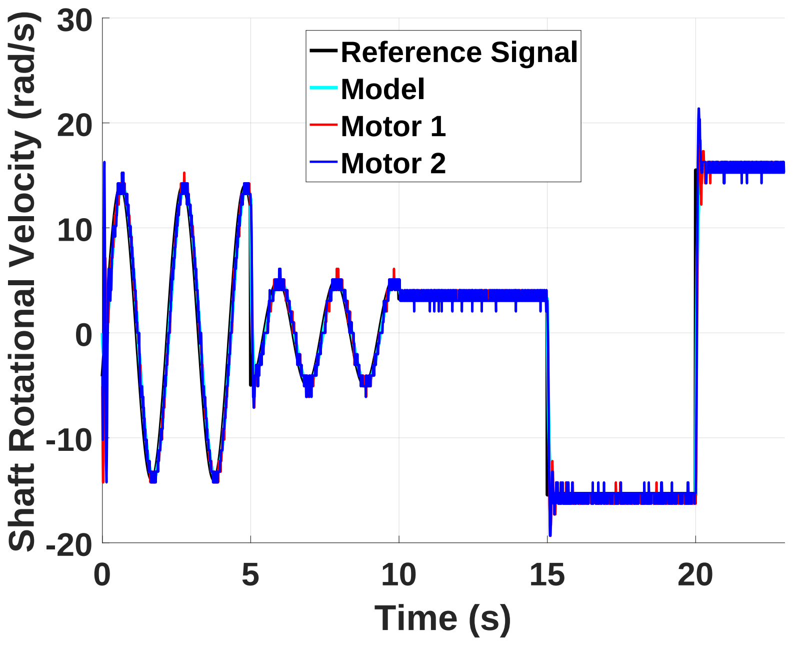


Figure C.8. Comparison of model to motor output for several reference commands.

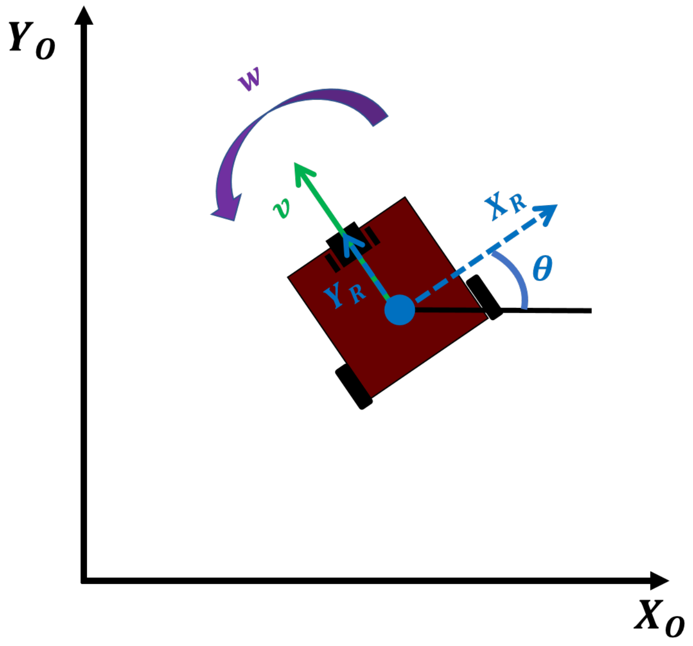


Figure C.9. Cartoon representation of Pheeno in a Cartesian coordinate frame.

I am pleased to have worked with and gained permission from my co-authors to include the material in this dissertation from my collaborators Dr. Spring Berman, Dr. Theodore Pavlic, Dr. Stephen Pratt, Dr. Andrea Bertozzi, Dr. Matthew Haberland, Dr. Ganesh Kumar, Dr. Aurélie Buffin, Karthik Elamvazhuthi, Ruben Gameros, Ragesh Ramachandran, Hamed Farivarnejad, Kathryn Dover, Matthew Lin, Michael Sheely, and Robert Gevorkyan.