9-2016

A Framework for Collaborative Multi-task, Multi-robot Missions

John T. Shepard
Santa Clara University

A FRAMEWORK FOR COLLABORATIVE MULTI-TASK, MULTI-ROBOT MISSIONS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF SANTA CLARA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

John T. Shepard

September 2016

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

[illegible]

(Christopher A. Kitts, Ph.D.) Principal Advisor

[illegible]

(Yi Fang, Ph.D.)

[illegible]

(Ignacio Mas, Ph.D.)

[illegible]

(Timothy K. Hight, Ph.D.)

[illegible]

(Terry E. Shoup, Ph.D.)

Approved for the University Committee on Graduate Studies

Abstract

Robotics is a transformative technology that will empower our civilization for a new scale of human endeavors. Massive scale is only possible through the collaboration of individual or groups of robots. Collaboration allows specialization, meaning a multi-robot system may accommodate heterogeneous platforms including human partners.

This work develops a unified control architecture for collaborative missions comprised of multiple, multi-robot tasks. Using kinematic equations and Jacobian matrices, the system states are transformed into alternative control spaces which are more useful for the designer or more convenient for the operator. The architecture allows multiple tasks to be combined, composing tightly coordinated missions. Using this approach, the designer is able to compensate for non-ideal behavior in the appropriate space using whatever control scheme they choose. This work presents a general design methodology, including analysis techniques for relevant control metrics like stability, responsiveness, and disturbance rejection, which were missing in prior work

Multiple tasks may be combined into a collaborative mission. The unified motion control architecture merges the control space components for each task into a concise federated system to facilitate analysis and implementation. The task coordination function defines task commands as functions of mission commands and state values to create explicit closed-loop collaboration. This work presents analysis techniques to understand the effects of cross-coupling tasks. This work analyzes system stability for the particular control architecture and identifies an explicit condition to ensure stable switching when reallocating robots. We are unaware of any other automated control architectures that address large-scale collaborative systems composed of task-oriented multi-robot coalitions where relative spatial control is critical to mission performance.

This architecture and methodology have been validated in experiments and in simulations, repeating earlier work and exploring new scenarios and. It can perform large-scale, complex missions via a rigorous design methodology.

Acknowledgements

I am grateful for the support of many individuals and organizations that enabled me to complete this research program.

First, I must thank my advisor Chris Kitts, for his tenacity and patience with me during this long journey. This work extends his original idea for the cluster space technique and his guidance has been invaluable. Throughout my time at Santa Clara, I have appreciated his wit and golden insights. I also appreciate the guidance, recommendations and oversight of the members of my PhD committee.

I am grateful for the camaraderie in the members of the Robotic Systems Laboratory. My own research extends the earlier work of Ignacio Mas, Paul Mahacek, Jose Acain and Thomas Adamek (who additionally kept the robots running); I stand on their shoulders. Michael Neumann, Jasmine Cashbaugh, and Kamak Ebadi have shared the journey towards their own PhDs; I cheer them on as they finish. Anne Mahacek and Mike Rasay maintain order within the RSL; I respect and admire them both. I also thank Adwait Bhalerao, from whom I learned a great deal, and who endured many hours of testing.

I am thankful to CSA Engineering and Moog Inc, for their financial support and flexibility to accommodate my academics. I appreciate the encouragement, professional development, and technical guidance from my colleagues there, including: Christian Smith and Eric Anderson for their advocacy and career advice, Joe Maly for his constant encouragement, and Paul Keas, Pete Devlin, Tim Pargett, and Chris Oesch for their technical expertise and spirited discussions.

My family and friends have been essential for emotional support through this uncertain process. I am eternally grateful to my wife Kristen, for her patience and her encouragement in the form that I needed, not as I wanted. I value the guidance and suggestions of my parents, brother, parents-in-law, siblings-in-law, the extended clan, and Cooper and the Beans. Finally, I am appreciative of my friends and their frequent question to which I can finally respond, “Yes.”

Table of Contents

Table of Figures

Table of Tables

Nomenclature

Accents:

$\breve{x}$	Desired or commanded vector, also $\check{x}$
$\hat{x}$	Actual or estimated vector
$\dot{x}$	Time derivative of vector x

Spaces:

r	Robot space pose vector
c_j	Cluster space pose vector for task j
t_j	Task space pose vector for task j
m	Mission command vector

Transformations:

$KIN_i(x)$	Kinematic equations for space i
$J_{i_j}(x)$	Jacobian matrix for space i and task j

Allocation:

n	Total number of robots in the system
$\vec{n}$	Vector of robot allocation
n_i	Number of robots assigned to task or cluster i
S	Robot assignment matrix

Collaboration functions

Q	Task coordination function
P	Resource allocation function

Controllers:

$u_{i\,j}$	Control effort for space i and task j
$K_{i\,j}$	Control gain for space i and task j

Dynamics:

g_{i_j}	Transfer function for space i and state j
G_{i_j}	Diagonal transfer function matrix for space i and task j
p	switched state index corresponding to configuration
V	Candidate Lyapunov function

1 Introduction

1.1 Motivation

Robotics is a transformative technology that will empower our civilization for a new scale of human endeavors. These endeavors include scientific exploration, precision agriculture, military force, climate engineering, and planetary colonization. Massive scale is only possible through the collaboration of individual or groups of robots. Collaboration allows specialization, meaning a multi-robot system may accommodate heterogeneous platforms including human partners.

Multi-robot systems increase the scope and scale of tasks, both in quantity and quality. *More* robots incrementally improve tasks where quantity matters, like manufacturing or explorations. They also provide new capabilities like redundancy and distributed-ness, but most importantly multi-robot systems enable specialization through collaboration. Specialization allows robots to be *different* and thus better at particular tasks. Collaboration entails sharing resources, synchronizing efforts, and providing support services. It enables larger, multifaceted missions comprised of specialized coalitions of agents, like assembly, search and rescue or harvesting.

Today, robotics is a hot industry. Since the financial crisis of 2007-2008, industrial robot sales experienced a 17% compound annual growth rate (CAGR) between 2010 and 2014 and analysts predict 15% CAGR between 2015 and 2018 [**1**]. Consumer robots have had a more recent boom and analysts predict a 17% CAGR between 2014 and 2019 [**2**], thanks to robotic vacuums and consumer drones. Furthermore, the Internet of Things

(IoT) movement adds ubiquitous data and connected devices in the billions [**3**]. At the frontier of robotics is collaboration, in industry [**4**] [**5**] [**6**] as well as in research [**7**] [**8**] communities. Collaboration empowers diverse, multi-dimensional applications of robotics.

1.2 Vision

One goal for multi-robot research is synergy between man and multiple machines. The human operator can intuitively specify complex, multi-faceted goals with unspecified intermediate tasks and dependencies. The federated multi-robot controller decomposes the mission into efficient tasks, defining the necessary task dependencies, assigns coalitions of robots to accomplish each task, and manages changing environmental conditions and operator commands. Each task occurs quickly and precisely to accomplish the mission. This system can be designed in a straightforward, formulaic manner, has tangible performance metrics and is easy to implement and repurpose for new missions. The research summarized here is a small step towards this goal.

1.3 Example Collaborative Missions

A collaborative multi-robot system enables missions in addition to independent tasks. Missions are composed of multiple tasks, with each task performed by a coalition of robots. Some missions emphasize a primary task with auxiliary support tasks while other missions consist of many instances of the same task performed in parallel. Collaborative tasks may be performed by heterogeneous multi-robot systems, mixing platform capabilities (different sensing or actuation capabilities) or domains of operation (land, sea, air, and space). Listed below are examples of general categories of tasks with specific instances:

- Observation (exploration, scouting, data collection, reconnaissance)
- Transportation (harvesting, mining, forestry, oil & natural gas)
- Manipulation (manufacturing, construction, site clearing)
- Communication (long range, area coverage)
- Sensing augmentation (coverage, specialized or shared sensors)

- Protection (escort, guard, patrol)
- Relief (repair, recharging, refueling, unloading)

These tasks can be combined into missions such as:

- **Aircraft manufacturing**: Large mobile platforms move two aircraft sections into place for assembly. Multiple manipulators work together to fasten the sections together, with one robot hammering rivets into place while the other robot reacts forces into the bucking bar. Quality inspections can also be performed by another type of robot.
- **Planetary colonization**: Large soil-moving robots can prepare the terrain while construction robots can assemble buildings. Smaller aerial or inflatable robots can monitor work progress and provide overhead sensing capabilities to the ground crews.
- **Security**: Aerial vehicles provide situational awareness to ground teams. If a threat is detected, scouts are sent to identify their intention. If hostile, heavier vehicles are sent to engage.
- **Crop Harvesting**: Aerial vehicles observe fields to assess crops and decide where to harvest. Multiple specialized harvest robots are deployed to cut and collect the crop based on ripeness and weather conditions. Autonomous trucks coordinate with the harvesters to maximize throughput.
- **Science**: Specialized robots with a suite of instruments collect the relevant data. Other robots patrol the area to allow safe data gathering. Communications robots relay the data back to interested parties on the shore.

1.4 Literature Review

Given the broad topic of multi-agent systems control, the following subsections discuss our research in multi-robot systems within a larger body of research.

1.4.1 Control of Individual Task

For individual task-specific coalitions, researchers have demonstrated tasks such as foraging [**14**] [**15**], exploration [**16**] [**17**] [**18**] [**19**], field navigation [**20**] [**21**] [**22**], sensor coverage [**23**] [**24**] [**25**], and manipulation [**26**] [**27**] [**28**] [**29**] [**30**]. These applications are coordinated using algorithmic methods, decentralized strategies, implicit potential functions, or explicit space transformations depending on task complexity, state coupling, and performance requirements. Algorithmic search and symbolic techniques often do not consider system dynamics and thus have limited applicability to tightly coupled or high performance tasks. Decentralized strategies, like swarms, can be robust to robot failures and other unexpected behaviors but are difficult to analyze and design due to emergent phenomenon. Potential functions are simple but can require careful tuning to achieve the desired response for complex tasks.

1.4.2 Control of Multiple Tasks

For collaborative multi-task missions with federated coalitions, the primary challenges are task allocation, assignment of resources, and coordinated motion control.

On the topic of task allocation, robotics researchers are developing algorithms for decomposing and assigning tasks given constraints. For example, Parker, Zhang and Tang [**31**] [**32**] use behavior-based representations (schemas) of robots to identify candidate coalitions that are feasible for task execution. Their most recent developments use these representations during planning functions to enable autonomous capability sharing. Using a different approach, Vig and Adams [**33**] adapted the Shehory and Kraus distributed problem solving algorithm for multi-robot coalition formation. They address concerns specific to multi-robot systems, such as communications, computation and other resource constraints.

On the topic of resource allocation, robotics researchers are developing algorithms that allow sharing of capabilities and common resources. For example, Shiroma and Campos [**34**] use a bidding process and constraint functions to evaluate if resources, like operating space, communications channels, and processor capabilities, are sufficient to complete actions.

On the topic of coordinated motion control, we are unaware of any automated control architectures that address large-scale collaborative systems composed of task-oriented multi-robot coalitions where relative spatial control is critical to mission performance. Our goal is to address this missing piece.

1.4.3 Multi-Robot Control Taxonomy

Within the field of robotics, a generally accepted practice is to divide control into execution and planning functions. The execution component manages high speed, low complexity functions like state estimation, dynamic control, and actuation in real time. The planning component manages low speed, high complexity functions like task decomposition, command generation, and health management in non-real time. Multi-robot systems and systems of systems require additional functionality for cross-platform and cross-task collaboration

Collaboration is a broad topic with many proposed approaches. Seminal work by Gerkey and Mataric [**35**] [**36**] proposed a formal taxonomy of task allocation in multi-robot systems which was later extended by Korsah, Stentz and Dias [**37**]. Task allocation approaches are categorized as single task (ST) or multiple task (MT) assignments to each robots; tasks requiring single robots (SR) or multiple robots (MR); instantaneous assignments (IA) or time-extended assignments (TA) that plan for the future; and degree of utility interrelatedness, being no dependencies (ND), in-schedule dependencies (ID), cross-schedule dependencies (XD) or complex dependencies (CD).

Within the given taxonomy, the architecture presented herein uses single-task (ST) robots to perform multi-robot (MR) tasks with instantaneous (IA) assignments. The schedule dependencies depend on the task and resource allocation policies, which were not our focus, but can accommodate cutting edge algorithms. While valuable for comparison, this taxonomy does not consider factors of performance, our focus, which is another key attribute when selecting an architecture.

1.4.4 Systems-of-Systems

A particular instance of multi-task missions is in the field of systems of systems (SOS) engineering. DeLaurentis and Crossley state, "a system of systems arises when a set of needs are met through a combination of several systems. Each system can operate independently but each also must interact effectively with other systems to meet the specified needs" [**38**]. Many examples exist of systems-of-systems (SOS) in military, political, economic, civic, humanitarian, and agricultural environments, such as: advanced transportation management [**39**] [**40**] [**41**], satellite constellations [**42**], modern defense systems [**38**] [**39**] [**43**], integrated manufacturing [**38**], business enterprise resource planning (ERP) systems [**44**], health care [**38**], civic policy [**39**], and the Internet [**39**] [**44**].

To best address the needs of complex missions, systems within an SOS must work together in a collaborative way. This includes synchronizing motions and activities, sharing resources, and providing mutual support as required in order to respond quickly, maximize benefit, minimize expendables, manage complex trade-offs, fit within given constraints, and accommodate uncertainty. This is a matter of task coordination, resource allocation, and unified motion control.

As a simple example of coordinated motion control for a SOS, consider precision agriculture. When harvesting fruit, vegetables, and grains, the coordinated motion of harvesters and transportation equipment (typically trucks) influences speed, productivity, and safety [**45**]. Accurate tracking of harvesters by trucks reduces turn around time and swapping holding containers. Controlling multiple harvesters increases the throughput of a single operator. Maintaining separation distance avoids collisions and ensures safety. Joint control of harvesting and transportation equipment increases overall efficiency which is critical due to the large scale of commercial farms and the short harvest season.

In a completely separate SOS domain, highly collaborative control is cited as the future of disaster response [**46**]. In wildfires, for example, coordinated deployment of firefighting personnel and equipment enables rapid response, coverage of large areas, and management of resource constraints [**47**]. Rapid response with appropriate assets (fire engines, bulldozers, hand crews, helicopters) is key to minimizing fire size and intensity. Maintaining coverage helps manage the uncertainty of fire location. Redistribution of assets as the situation changes helps alleviate demands on operating bases. In these ways, coordinated control of assets is critical to wildfire suppression.

Systems-of-systems also exist within the field of robotics, as do similar challenges of cross-system control. For the case of multi-robot systems, the primitive system is the robot and the system-of-systems is the group or "coalition" of robots.

SOS engineering is a developing field. Researchers are exploring a number of key topics including: formalizing the SOS framework [**41**] [**44**] [**48**], developing strategies for design and performance analysis [**49**] [**50**] [**51**] [**52**], and creating integrated control architectures [**53**] [**54**]. This last topic, cross-system control, which we will discuss herein, is most critical to eliciting maximum performance of SOS and is the primary design space for SOS engineering. This architecture is applicable to SOS because it allows integration of different yet collaborative tasks and is able to analyze emergent behavior.

1.4.5 Complexity

Speaking more broadly, the field of complexity studies multi-agent systems, which can range from biological to software. "Complexity is a property of an open system that consists of a large number of diverse, partially autonomous, richly interconnected components, often called Agents…whose behaviour emerges from the intricate interaction of agents and is therefore uncertain without being random." [**55**] The distinguishing characteristics of complex systems are: connectivity, autonomy, emergence, non-equilibrium, nonlinearity, self-organization, and co-evolution [**55**]. Select characteristics of complexity exist within (or define) other fields such as self-organizing systems [**56**], complex adaptive systems [**57**], and systems of systems [**38**]. Our particular interest is in the control of collaborative multi-robot systems for which complex behaviors can be specified.

The complexity of multi-robot systems is determined by control hierarchy. Non-hierarchical control architectures have greater complexity due to increased autonomy of the individual agents. Individual autonomy increases robustness and adaptability with lower global communications requirements. Examples of non-hierarchical architectures include decentralized techniques [**33**] [**34**], symbolic reasoning [**30**] [**32**], and search methods [**16**] [**31**]. Hierarchical control architectures have lower complexity due to greater coordination between agents. Global coordination reduces uncertainty and can increase cross-agent performance metrics but has higher communications requirements which can reduce robustness. Examples of hierarchical control architectures include behavior based methods [**58**] [**25**] and control space transformations [**59**] [**19**] [**20**] [**22**] [**27**] [**29**].

1.4.6 Robot Control Perspectives

This research approaches robot control from the perspective of dynamic systems but many researchers take an algorithmic (computational) approach. Both are equally valid and appropriate for different applications, depending on system behavior and task complexity. Recent advances in machine learning techniques, like reinforcement learning [**60**], have allowed robotic control systems to learn complex behaviors without specific human instructions. One could argue that this capability diminishes the need for control architecture design and exhaustive analysis of system dynamics. However, well-designed architectures, like that presented herein, make the system "easier" to control, thus complimenting machine learning. Using an architecture like this will require fewer iterations to converge and allows less complex algorithms because it reformulating the states for simpler mapping between inputs and outputs.

1.5 Thesis statement & Contributions

This research developed and verified a formal, unified control architecture for collaborative missions comprised of multiple, tightly coupled tasks performed by coalitions of robots. The main contributions of this work are:

- Establishing a formal design process for creating multi-spatial control architectures, extending prior work in the cluster space to arbitrary spaces.
  Unifying the representation of multiple tasks performed in parallel to simplify analysis and implementation of collaborative missions
- Incorporating the capability for reallocating resources (robots) between tasks, including managing definitions of tasks and clusters as coalitions change size and establishing formal stability criteria for safely switching between configurations.
- Establishing a novel method for coordinating tasks enabling closed-loop collaboration and rapid re-tasking, deriving a dynamic model of the collaborative system for performance and stability analysis.
- Analyzing the stability and performance of multi-spatial control architectures typically used by our group, providing guidance for controller design. This includes both rigorous nonlinear Lyapunov analysis which is more general, and linear approximations which can be more convenient using standard design tools.

- Verifying the design process, by experiment for a communications task and by simulation for prior work by our research group, specifically: formation control, escorting, and adaptive navigation.
- Validating multi-task missions, by experiment for a simple mission (4 robots, 2 tasks) and by simulation for a complex mission (10 robots, 4 tasks)
- Adding new definitions for a cluster taking the form of a chain which is arbitrarily extensible, and a task of long-range communications relay.

1.6 Reader's Guide

The remaining document discusses details of the control space architecture. Chapter 2 examines individual tasks. It presents background material, the design technique, analysis approaches, and advantages. Simulations and experiments provide examples of the individual task architecture with results highlighting different features. Chapter 3 examines collaborative tasks. It presents a unified representation of multiple tasks, the method of resource allocation, and the method for task coordination. Simulations and experiments again provide examples of the multi-task architecture with results highlighting different features. Finally, Chapter 4 discusses conclusions of the research and directions for the future. Additionally, appendices discuss the multi-robot testbed, the communications relay testbed, and example applications.

2 Individual Task Space Control

2.1 Description of technique

The multi-robot control architecture is a series of cascaded control loops that each use alternative representations of the system state. Each layer defines the system using a complete set of states that are relevant to the scope of that layer, and kinematic transforms are used to convert between the layers and their associated state spaces. For a task-oriented multi-robot coalitions, we typically use pose descriptions in three different spaces: the global pose of the individual robots, termed the robot space; the geometric configuration of the robots, termed the "cluster space"; and the defining spatial parameters for the intended application, termed the task space.

As an example, consider an escorting task using three robots. Traditional controllers consider the individual robot positions; this is the robot space. Alternatively, the group geometrically forms a triangle; this is the cluster space. Still further, establishing an escorting perimeter can be described by centering and equalizing the triangle around a protectee with a specific radius and phase; this is the task space. These are three different descriptions of the same physical deployment of robots, each allowing specification and control from different points of view.

Pose states are mapped between spaces through a set of kinematic transformation equations. Velocity states and forces are mapped between spaces using Jacobian matrices. A general form of these transformations is presented below from space $\{X\}$

with pose $\vec{x}$, velocity $\dot{\vec{x}}$ and control effort $\vec{u}_x$ to space $\{Y\}$ with pose $\vec{y}$, velocity $\dot{\vec{y}}$ and control effort $\vec{u}_y$ for a system consisting of n robots of m degrees of freedom.

The kinematic transformation equations are:

$$\vec{y} = KIN(\vec{x}) \triangleq \begin{bmatrix} g_1(x_1\ , \dots , x_{nm}) \\ \vdots \\ g_{nm}(x_1\ , \dots , x_{nm}) \end{bmatrix} \tag{1}$$

$$\vec{x} = INVKIN(\vec{y}) \triangleq \begin{bmatrix} h_1(y_1\ , \dots , y_{nm}) \\ \vdots \\ h_{nm}(y_1\ , \dots , y_{nm}) \end{bmatrix} \tag{2}$$

The Jacobian matrices are:

$$\dot{\vec{y}} = J(\vec{x})\dot{\vec{x}} \triangleq \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_{nn}} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_{nm}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{mn}}{\partial x_1} & \frac{\partial g_{mn}}{\partial x_2} & \cdots & \frac{\partial g_{nm}}{\partial x_{nm}} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{nm} \end{bmatrix} \tag{3}$$

$$\dot{\vec{y}} = J^{-1}(\vec{y})\ \dot{\vec{x}} \triangleq \begin{bmatrix} \frac{\partial h_1}{\partial y_1} & \frac{\partial h_1}{\partial y_2} & \cdots & \frac{\partial h_1}{\partial y_{nm}} \\ \frac{\partial h_2}{\partial y_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial y_{nm}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_{mn}}{\partial y_1} & \frac{\partial h_{mn}}{\partial x_2} & \cdots & \frac{\partial h_{nm}}{\partial y_{nm}} \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_{nm} \end{bmatrix} \tag{4}$$

Assuming the use of a resolved-rate control approach of the type proposed in [**59**], which we typically use in practice, compensation commands are transformed:

$$\vec{u}_x = J^{-1}(\vec{y})\vec{u}_y \tag{5}$$

These layer-specific computations prescribed in (1)-(5) may be successively applied such as is shown in Figure 1. In this diagram, one set of transforms converts between the robot space and the geometrically-oriented cluster space. Then another converts between the cluster space and the application-oriented task space.

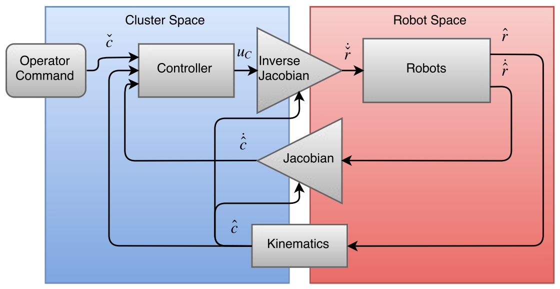


Figure 1: One formulation of the cluster space control architecture

2.2 Background: Cluster Space Control

Early work on this architecture focused on formation control which was often informally extended to task control. Using the cluster space technique, the multi-robot system is considered as virtual articulating mechanism which can be actuated along different degrees of freedom (separation distances, relative angles). The underlying goal of the cluster space technique is simple motion specification and control of multi-robot systems. This is accomplished by considering multiple robots as a single geometric entity rather than as individual robots. The pose of a cluster is described by its location and shape, which are related to individual robot positions through a set of kinematic transforms. The interested reader should consult [**61**] for the original description of this technique. Some of these previous applications are shown in Figures 2-4.

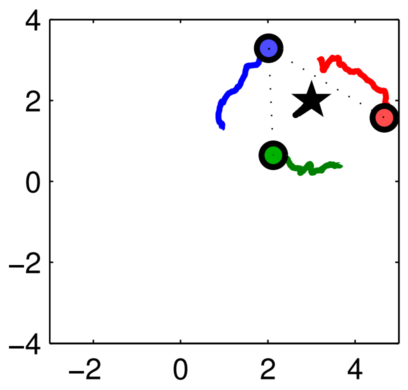


Figure 2: Target escorting and patrolling [62]

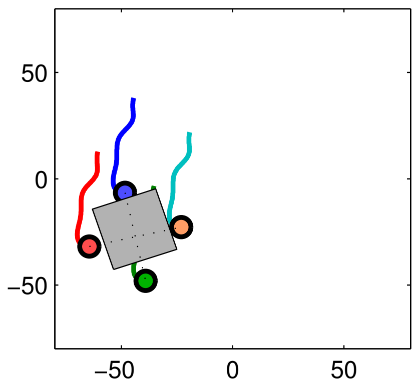


Figure 3: Object entrapment and manipulation [62]

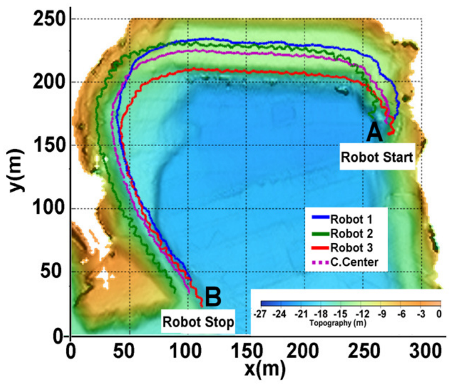


Figure 4: Adaptive navigation [22]

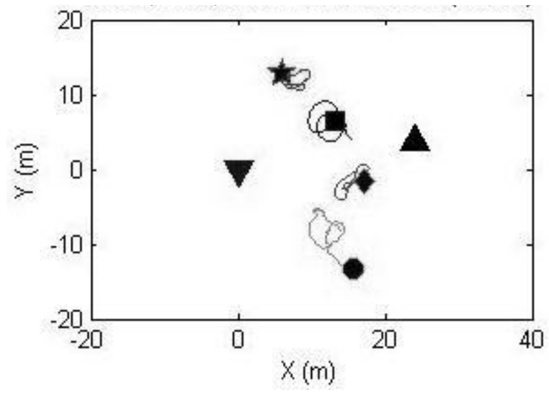


Figure 5: Dynamic guarding [63]

We observed a common approach in this prior work and formalized the extension of cluster space to the task space. Layering control spaces in this way can be extended arbitrarily. As an example, ongoing work extends [**22**] to follow higher order features of a field, like ridges and trenches, using a cluster of clusters in an additional layer of formation control.

2.3 Task Design Process

There is a systematic approach to constructing each control space:

1. Identify the key control spaces for the architecture and the spanning states for each space
2. Define the kinematic transformation equations to relate the pose state variables in adjoining spaces
3. Compute Jacobian matrices from the kinematic equations to relate the rates of change of the pose state variables
4. Design the space-specific controllers and evaluate their performance, integrating the components above

The following subsections provide detail on each step.

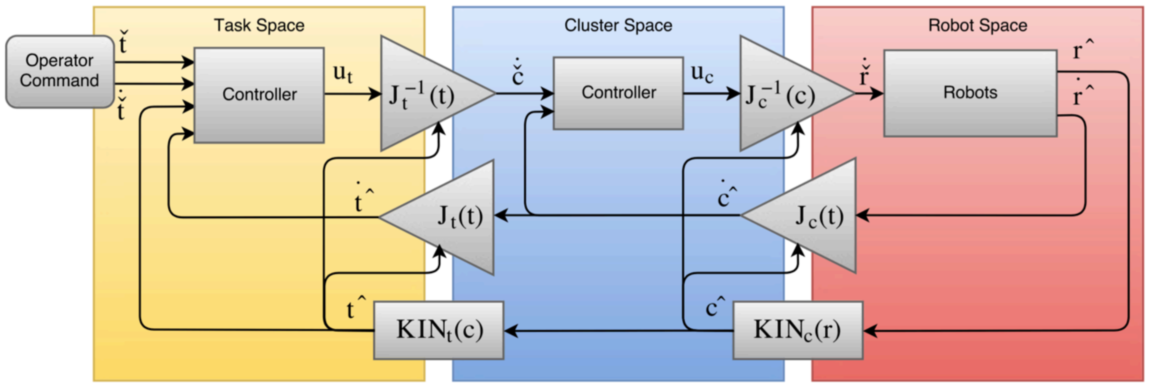


Figure 6: The layered, hierarchical control space architecture utilizing robot, cluster, and task spaces to perform a single task

2.3.1 Control Spaces & States

Each control space considers the system from an alternate perspective that will be more useful or beneficial to the designer or operator. The states within this new space must fully define all degrees of freedom of the system. These designations are up to the discretion of the designer but may correspond to operator inputs or where there are convenient distinctions in functionality (ex: different hardware). Designers can use these spaces to compensate for non-ideal behavior in appropriate spaces like friction in an actuator space or sensor behavior in a task space.

As an example, we typically use three spaces: robot space with states corresponding to the pose of the individual robots; cluster space with states corresponding to the formation parameters like centroid, separation distances and relative angles; and task space with states corresponding to the motion specific goals of the task.

2.3.2 Kinematic Transformations

Kinematic transformation equations algebraically map the system pose states (or degrees of freedom) between spaces. Forward kinematic equations (1) map the lower space states to the higher space states (ex: robot to cluster) and the inverse kinematic equations (2) map the higher space states to the lower (ex: cluster to robot). These equations may be based on geometry, modeled behavior, or any arbitrary function as desired by the designer or the operator.

2.3.3 Jacobian Matrices

The Jacobian matrices map system state velocities between spaces, per equations (3) and (4). These matrices are straightforward to derive from the kinematic equations but may be lengthy. If advantageous (ex: if the inverse kinematics are difficult to find), the Jacobian may be computed in one space and numerically inverted to compute the inverse Jacobian. A symbolic solver (ex: MATLAB symbolic toolbox) is highly recommended to pre-compute these equations

Because the Jacobians are generally a function of system state, they must be updated as the system changes pose. Certain configurations of the system may result in singular Jacobians corresponding to degenerate geometry or loss of degrees of freedom. Singularities can be calculated from the Jacobian determinant, below, and the designer should consider impacts on the system workspace.

$$\det\left(J(\vec{x}_{singular})\right) = 0 \tag{6}$$

2.3.4 Controller Design

Within each space (ex: robot, cluster, task, etc.), the architecture can accommodate any form of controller (ex: linear time invariant, state machine, optimal, adaptive). The previously defined control space transformations are assembled as shown by the block diagram in Figure 6. The kinematic transformations add coupling between the system states, but the Jacobians provide a degree of decoupling, allowing independent control of all states until nonlinear effects become appreciable. Model-based methods can completely cancel coupled dynamics as shown by [**64**]. With this structure, the state trajectories can be well behaved (exponentially decaying) with simple (ex: linear time invariant) controllers. Controllers can be empirically tuned or analytically designed as described in Section 2.4.

This layered architecture simplifies controller design, facilitates system interface and modularity, and can yield higher performance but may have some practical challenges.

Controller design is simplified by the construction of the layers. The different spaces are effectively a series of cascaded inner loops, which conditions and linearizes the system behavior. Non-ideal behavior, like disturbances, difficult dynamics (resonances, phase lag, roll-off) or nonlinearities (friction, deadbands, saturation, rate limits, state cross-coupling) can be managed within the appropriate control space. For example, if the robot space controller can compensate for wheel friction, the communication task space controller can focus on compensating for line-of-sight obstructions. In this way, the higher level system behavior (related to tasks) become independent of the lower level behavior (related to robots).

Control space abstraction facilitates interfaces for human operators and other systems. By constructing the architecture with the spaces relevant to a human operator, system states may be specified or monitored naturally in familiar terms. The abstraction due to the control space approach also benefits system integration. High-level analysis can make approximations of low-level behavior. This is especially important for systems-of-systems where the scope of integration can become prohibitive to analyze. This layering decouples the task from the actual hardware implementation, which allows resource sharing as will be discussed later. In some ways, abstraction facilitates heterogeneous coalitions, for the particular members are irrelevant so long as the task is accomplished.

Finally, this control architecture can improve performance metrics like speed, accuracy and robustness. Well-behaved (exponentially converging, decoupled) state trajectories are often naturally achieved in the control space through the use of simple linear controllers. The layered multi-space controllers linearize system responses and increase disturbance rejection through a combination of the controller design and Jacobian transformations. The transformations between control spaces explicitly encode model information in kinematic equations and Jacobian matrices. Model based control is grounded in the fundamental behavior of the system. The architecture may have some practical challenges, but so far these have been surmountable. The mathematically intensive nature of this approach can be concerning for scaling to larger numbers of robots. State updates and control calculations could burden real time computation and communication. Thus far, we have not been limited in our current experiments [**65**] which control up to 10 robots at 5Hz (totaling 9600 bits/sec) using non-optimized code (MATLAB) on commercial hardware (laptops, wireless modems). System hierarchy allows control computations to be partitioned and computed locally by each task coalition. Global information is only necessary for cross-task coordination (discussed in Chapter 3), which is typically at a lower rate than task control. By decentralizing the computations and limiting global communication, the architecture likely can be scaled to larger numbers of robots.

The following section presents rigorous analysis of the architecture which is only possible because of the formal mathematical basis of this method. This can be compared to implicit approaches like potential functions that can require careful tuning or swarm techniques where the resulting emergent behavior may be unintentional. The analytic rigor provides confidence during the design process to reduce system margins and increase system performance.

2.4 Analysis

Mathematical formalism is a key strength of this technique. It allows thorough analysis of the system behavior, the impact of the control space definitions and the control system interaction. This specific analysis assumes resolved-rate linear, time-invariant (LTI) controllers of the form presented in Figure 6; the approach may be followed to analyze different forms of controllers or architectures.

2.4.1 Control Space Transformation Stability Analysis

Let us consider the stability conditions for an architecture using transformations from space $\{X\}$ to control space $\{Y\}$. We define a candidate Lyapunov function of quadratic error in the control space $\{Y\}$ and assume the Jacobian is sufficiently far away from singularities:

$$V = \frac{1}{2}\vec{e}_Y^{\,T}\vec{e}_Y > 0 \qquad (7)$$

And finding the rate of change with respsect to time:

$$\frac{dV}{dt} = \vec{e}_Y^T \dot{\vec{e}}_Y = \vec{e}_Y^T \left(\dot{\vec{y}}_d - J \dot{\vec{x}} \right) \tag{8}$$

$$\frac{dV}{dt} \leq \|\vec{e}_Y\| \left(\|\dot{\vec{y}}_d\| - \|J\| \|\dot{\vec{x}}\| \right) \tag{9}$$

Then the Lyapunov rate of change is negative definite and thus stable in the Lyapunov sense if:

$$\frac{dV}{dt} \leq 0 \rightarrow \|\dot{\vec{y}}_d\| \leq \|J\| \|\dot{\vec{x}}\| \tag{10}$$

Hence the error remains bounded as long as the commanded rate in space $\{Y\}$ is less than actual rate in space $\{X\}$ as projected into the control space by the Jacobian. This conclusion is trivial, yet shows the influence of the Jacobians on system stability. Substituting our specific state spaces, the maximum robot rate limits the cluster rate command based on the cluster Jacobian and the maximum cluster rate limits the task rate command based on the task Jacobian.

2.4.2 Control Space Transformation Performance Analysis

Furthermore, exponential Lyapunov stability may be used to quantify the performance of an architecture using control space transformations by bounding the error with an exponential function with decay rate β:

$$\left| |\vec{e}| \right| \leq \alpha \|\vec{e}_0\| e^{-\beta t} \tag{11}$$

Starting with the condition for Lyapunov exponential stability, we again transform states to arrive at an expression for system responsiveness. We define a candidate Lyapunov function of quadratic error in the control space $\{\mathrm{Y}\}$:

$$V = \frac{1}{2} \vec{e}_Y^T \vec{e}_Y > 0 \tag{12}$$

$$\frac{dV}{dt} = \vec{e}_Y^T \dot{\vec{e}}_Y = \vec{e}_Y^T \left(\dot{\vec{y}}_d - J \dot{\vec{x}} \right) \tag{13}$$

Adding the condition for exponential stability:

$$\frac{dV}{dt} \leq -\beta V \tag{14}$$

$$\vec{e}_Y^T\left(\dot{\vec{y}}_d - J\dot{\vec{x}}\right) + \frac{1}{2}\beta \vec{e}_Y^T \vec{e}_Y \leq 0 \tag{15}$$

$$< \|\vec{e}_Y\|\left(\|\dot{\vec{y}}_d\| - \|J\|\|\dot{\vec{x}}\|\right) + \frac{1}{2}\beta\|\vec{e}_Y\|^2 \leq 0 \tag{16}$$

$$\rightarrow \beta \leq \frac{\|J\|\|\dot{\vec{x}}\| - \|\dot{\vec{y}}_d\|}{\frac{1}{2}\|\vec{e}_Y\|} \tag{17}$$

Hence the exponential decay rate is faster with smaller error, smaller command rate, faster speed, or a stronger relationship between spaces as defined by the Jacobian. Using this result, we can quantify the responsiveness of the system using the bounding exponential decay rate β

2.4.3 Linearized Transfer Function Analysis

We can approximate the robot behavior with a transfer function and compute the corresponding task-level transfer function. This allows us to design LTI feedback controllers within each control space using standard analysis techniques. The following analysis corresponds to the control architecture presented in Figure 6.

Starting with the robot space velocity transfer function, which we can assume as LTI given realistic (<10 Hz) bandwidth and slowly varying trajectories:

$$\dot{\hat{r}} = G_r \dot{\check{r}} \tag{18}$$

Transforming to cluster space and adding feedback control of cluster velocity gives the cluster space velocity transfer function:

$$J_c^{-1}\dot{\hat{c}} = G_r J_c^{-1} K_c\left(\dot{\check{c}} - \dot{\hat{c}}\right) \tag{19}$$

$$(J_c^{-1} + G_r J_c^{-1} K_c)\dot{\hat{c}} = G_r J_c^{-1} K_c \dot{\check{c}} \tag{20}$$

$$\dot{\hat{c}} = (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} G_r J_c^{-1} K_c \dot{\check{c}} = G_c \dot{\check{c}} \tag{21}$$

Transforming to task space and adding feedback control of task state gives the task space transfer function:

$$J_t^{-1} s\hat{t} = G_c J_t^{-1} K_t (\breve{t} - \hat{t}) \tag{22}$$

$$(J_t^{-1} sI + G_c J_t^{-1} K_t)\hat{t} = G_c J_t^{-1} K_t \breve{t} \tag{23}$$

$$\hat{t} = (J_t^{-1} sI + G_c J_t^{-1} K_t)^{-1} G_c J_t^{-1} K_t \breve{t} = G_t \breve{t} \tag{24}$$

where G_x represents a diagonal matrix of transfer functions in space x, K_x represents control gains in space x, and u represents control effort. The system pose is represented by r in robot space, c in cluster space, and t in task space. As subscripts, these letters associate the variable with a space. The hat ($\hat{x}$) and breve ($\breve{x}$) accents denote the actual and desired states respectively. The transfer functions at each layer can be approximated as LTI with proper tuning, maintaining diagonal dominance, and avoiding singularities. Equations (21) and (24) show the linearizing nature of multi-space control. A larger control gain K reduces the influence of the additional denominator term J^{-1} and minimizes the system dynamics.

2.4.4 Linearized Disturbance Rejection

Disturbances are most likely to occur at the robot (or platform) level from environmental effects (friction, traction, wind) or unmodeled phenomena (deadbands, saturation). We investigate the effects of robot-space disturbances δ_r on the cluster and task space states.

Starting with the robot space velocity transfer function subjected to a disturbance:

$$\dot{\hat{r}} = G_r\left(\dot{\breve{r}} + \delta_r\right) \tag{25}$$

Transforming to cluster space and adding feedback control of cluster velocity:

$$J_c^{-1}\dot{\hat{c}} = G_r J_c^{-1} K_c\left(\dot{\breve{c}} - \dot{\hat{c}}\right) + G_r \delta_r \tag{26}$$

$$(J_c^{-1} + G_r J_c^{-1} K_c)\dot{\hat{c}} = G_r J_c^{-1} K_c \dot{\breve{c}} + G_r \delta_r \tag{27}$$

$$\dot{\hat{c}} = (J_c^{-1} + G_r J_c^{-1} K_c)^{-1}\left(G_r J_c^{-1} K_c \dot{\breve{c}} + G_r \delta_r\right) \tag{28}$$

Assuming a regulating controller where $\dot{\check{c}} = 0$:

$$\dot{\hat{c}} = (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} G_r \delta_r \tag{29}$$

Transforming to task space and adding feedback control of task state:

$$J_t^{-1} s\hat{t} = (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t (\check{t} - \hat{t}) + G_r \delta_r) \tag{30}$$

$$\left(J_t^{-1} sI + (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t)\right)\hat{t} = (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t \check{t} + G_r \delta_r) \tag{31}$$

$$\hat{t} = \left(J_t^{-1} sI + (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t)\right)^{-1} (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t \check{t} + G_r \delta_r) \tag{32}$$

Assuming a regulating controller where $\dot{\check{t}} = 0$:

$$\hat{t} = \left(J_t^{-1} sI + (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} (G_r J_c^{-1} K_c J_t^{-1} K_t)\right)^{-1} (J_c^{-1} + G_r J_c^{-1} K_c)^{-1} G_r \delta_r \tag{33}$$

Much like in traditional cascaded control architecture, higher control gains K increase disturbance rejection. In a multi-spatial control architecture, the control space transformations also influence the system disturbance rejection as seen by the Jacobian inverse matrices J^{-1} in the denominator of equations (29) and (32). Large magnitude Jacobian inverses reduce the overall gain of the transfer function. In addition, these Jacobian inverses also add coupling between the original states which could benefit or impact the disturbance rejection of the system, depending on the space definition. At the task level, such as equation (33), the influence of the cluster layer is evident where control gains and Jacobian inverse matrices from both spaces are present.

2.5 Example Task: Long Distance Communications

As an example, consider the task of long-range communications between two exogenous end nodes using mobile relays. To maximize the link quality, robotic relay nodes will move to intermediate locations based on desired link characteristics.

2.5.1 Spaces & States

The problem can be divided into three spaces. The robot space describes the pose of the individual agents in the environment, defined by the global Cartesian position and orientation global position and orientation. The robot state vector is defined as:

$$\vec{r} \triangleq [x_1, y_1, \theta_1, \dots, x_n, y_n, \theta_n]^T \tag{34}$$

In the cluster space, the separation chain distances ρ_i and chain angles α_i are key due to their influence of the communication states, depicted in Figure 7. The cluster state vector is defined as:

$$\vec{c} \triangleq [x_c, y_c, \theta_c, \rho_1, \alpha_1, \phi_1, \dots, \rho_{n-1}, \alpha_{n-1}, \phi_{n-1}]^T \tag{35}$$

In the task space, the user is interested in maintaining sufficient communication quality of service (QoS) between two end nodes, with signals being relayed as needed. QoS proved impractical to quantify in real-time, so the system measures the link power between nodes using the received signal strength indicator (RSSI). For line-of-sight, the RSSI may be modeled as inversely proportional to the square of the distance between two points, hence:

$$s_i = \frac{k}{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} = \frac{k}{\rho_i^2} \tag{36}$$

where k is a constant associated with the antenna gain.

It is important to note that this quantity is measured directly; the mathematical model only guides the derivation of the kinematics *en route* to computing the Jacobians. Real world phenomenon, such as obstructions or directional antenna radiation patterns, are not captured by this simple model, but it proves sufficiently accurate to allocate control effort. Given a complex environment, such as non-planar terrain (hills, valleys) or obstructions (buildings, trees), this model would fail. More sophisticated models could be incorporated as appropriate, but that is beyond the scope of this dissertation.

As depicted in Figure 8, the quality of service between the end nodes is influenced by both the crosstrack error, e_{xt}, and the angles of alignment, γ_i. Given a line of sight model, the maximum total signal strength is achieved by minimizing the crosstrack error and the angles of alignment. The ratio or balance, B_i, of the link power in each segment is also important to avoid data rate bottlenecks or backup in homogeneous systems, or to allow for imbalanced transmission rates in nonhomogeneous systems. Lastly, the orientation of the robot, ψ_i, is included to define fully all degrees of freedom of the system. The communication state vector is defined:

$$\vec{t} \triangleq [\, B_1, \dots, B_n, e_{xt}, \gamma_1, \dots, \gamma_{n-1}, \psi_1, \dots, \psi_i]^T \tag{37}$$

2.5.2 Kinematic Equations

Robot states are transformed into the cluster states using kinematic equations derived from formation geometry presented in Figure 7:

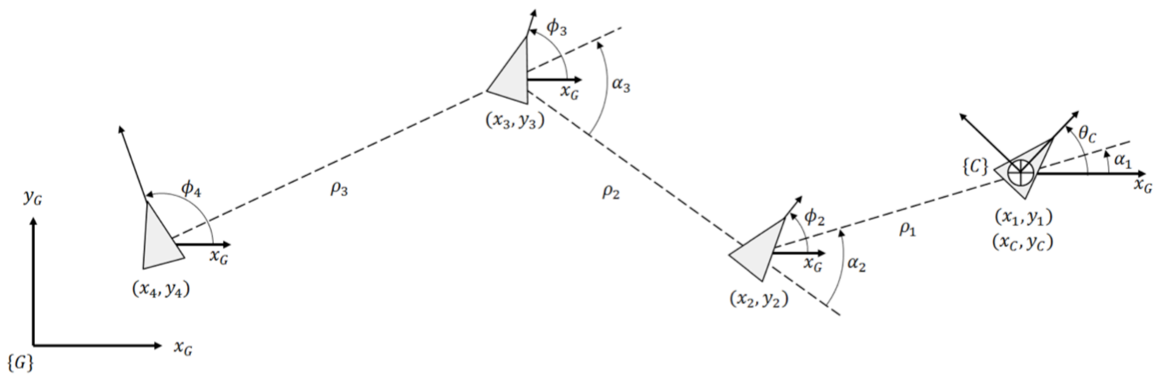


Figure 7: Serial Chain Cluster Diagram

Cluster frame:

$$x_c \triangleq x_1 \tag{38}$$

$$y_c \triangleq y_1 \tag{39}$$

$$\theta_c \triangleq \theta_1 \tag{40}$$

Chain length:

$$\rho_i \triangleq \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{41}$$

Chain angle:

$$\alpha_i \triangleq atan2(y_{i+1} - y_i, x_{i+1} - x_i) - \sum_{j=1}^{i-1} \alpha_j \tag{42}$$

Node orientation:

$$\phi_i \triangleq \theta_i \tag{43}$$

where $atan2(\dots,\dots)$ is the two-argument function that calculates a four-quadrant arc tangent with a range of $[\pi, -\pi]$.

These cluster states are transformed into the task states using the measured link states and system geometry as presented in Figure 8:

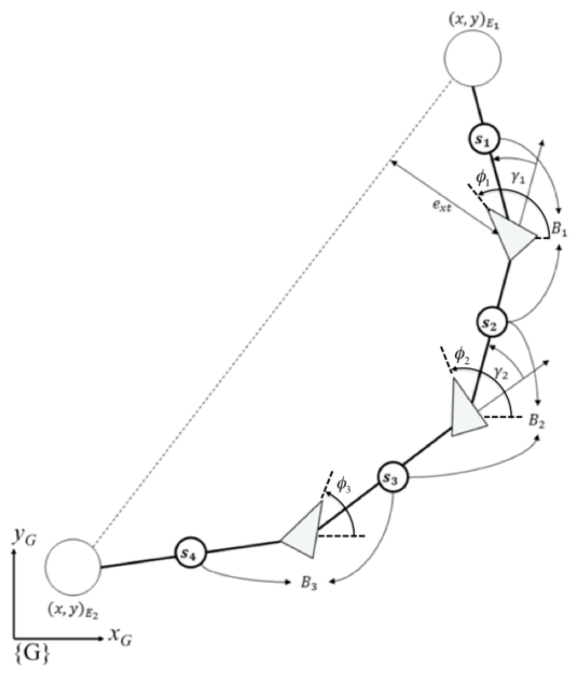


Figure 8: Long Range Communication Link State Diagram

Balance:

$$B_i \triangleq \frac{s_{i+1}}{s_i} = \frac{\rho_i^2}{\rho_{i+1}^2} \tag{44}$$

Crosstrack error:

$$e_{xt} = \sqrt{\frac{\left((x_{E_2}-x_{E_1})(y_{E_1}-y_c)-(x_{E_1}-x_c)(y_{E_2}-y_{E_1})\right)^2}{\left(x_{E_2}-x_{E_1}\right)^2+\left(y_{E_2}-y_{E_1}\right)^2}} \tag{45}$$

Angle of alignment

$$\gamma_i = \alpha_i \text{ for } i = 2, \dots, n \tag{46}$$

Orientation:

$$\psi_i = \phi_i \tag{47}$$

where (x_{E_1}, y_{E_1}) and (x_{E_2}, y_{E_2}) are the Cartesian positions of the base and end nodes that are being connected by the multi-robot system.

2.5.3 Jacobian Matrices

The Jacobian matrices are computed from the kinematic equations to map velocities between spaces. The solution is typically lengthy and so not shown here but easily computed.

2.5.4 Control Design

The cluster space control law utilizes proportional feedforward and feedback, shown below, for response time and error rejection respectively:

$$\vec{u}_c = K_f \dot{\vec{c}}_d + K_p(\dot{\vec{c}}_d - \dot{\vec{c}}) \tag{48}$$

where $\vec{u}_c$ denotes cluster space control effort, $\dot{\vec{c}}_d$ denotes desired cluster velocity, K_f denotes proportional feedforward gain, and K_p denotes proportional feedback gain.

The communication space uses proportional feedback control, shown below:

$$\vec{u}_a = K_p(\vec{a}_d - \vec{a}) \tag{49}$$

where K_p is the feedback gain and $\vec{a}_d$ is the desired state. This yields sufficient performance as the subsequent layers are well behaved.

2.5.5 Experimental Results

Two scenarios were examined with the single communications task: A) system response to environmental attenuation and B) system response to hardware configuration changes such as reductions in transmission power. Additional examples are provided in [**65**].

2.5.5.1 Simulated Attenuation

This scenario simulates system behavior from regional effects such as obstructions, fog, or foliage. A comparison of the trajectory of the system with and without these effects demonstrates its ability to adapt in unexpected environments.

A single overhead view is shown in Figure 9. with robot trajectories plotted from both the ideal and attenuated scenarios. A region of power attenuation has been created at $y > 40$, where any link involving a robot within this area is reduced by half. The remote node traverses an steady arc around the base node while the multi-robot system maintains link balance and maximizes transmission power as described before. In the ideal case, the robots spread evenly and follow the traverse in concentric arcs. In the non-ideal case, the multi robot system begins as before, but alters its trajectory to rebalance the links when it senses a drop in signal strength as nodes enters the region of attenuation.

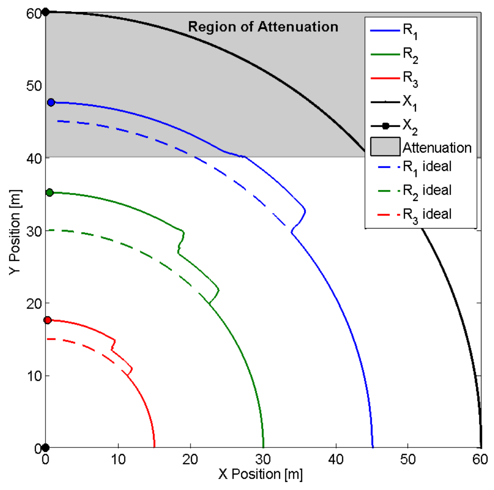


Figure 9: Overhead view of robot *R* position overlay comparing trajectories in ideal transmission environments (dashed) and trajectories responding to an encountered region of attenuation (solid)

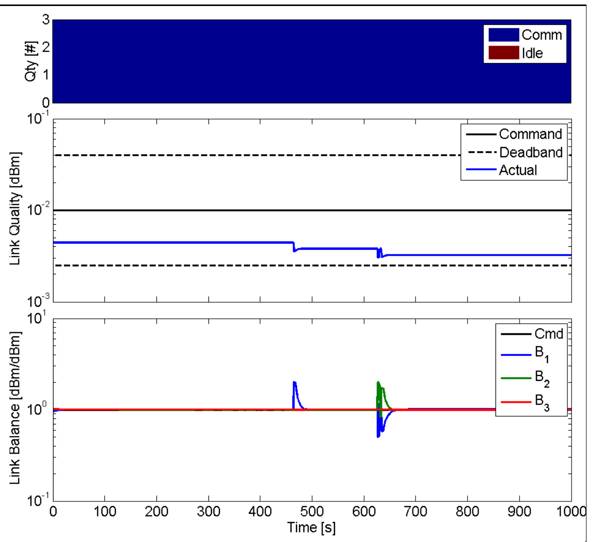


Figure 10: Time history of key system states for simulated attenuation scenario. All robots remain in the communication task and so quantity is constant. The robots enter the region of attenuation at time 475 and time 625 as shown by the decreases in link quality and link balance transients.

This example demonstrates the value of direct measurement of communication states and high-level task-space control. Sensing the signal strength allows the system to maintain the desired state despite unanticipated characteristics of the environment. In contrast, an open-loop, model-based approach would evenly distribute the nodes as shown in the first case which would yield lesser performance in non-ideal environments. Higher performance is achieved with simple high-level specification of the desired task with no additional input when encountering these localized effects.

2.5.5.2 Hardware Configuration Change Experiment

This scenario examines the control system response to internal events such as component failures or competing priorities like power reduction, using the test bed described in Appendix A. The system is allowed to reach equilibrium in its nominal configuration, then the power of transmitter 2 is reduced and the system achieves equilibrium.

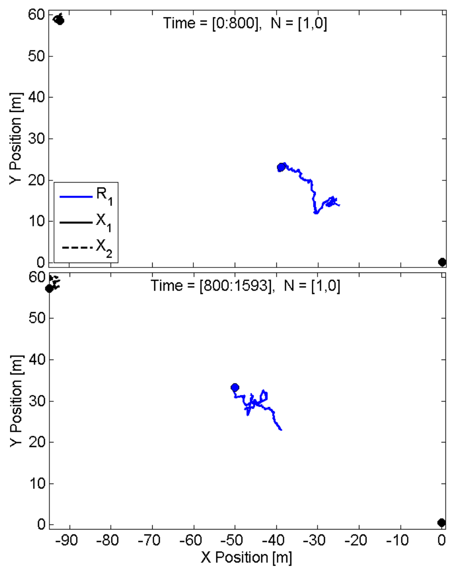


Figure 11: Overhead view of positions of robot *R* and exogenous nodes *X* at specified times during hardware configuration change experiment

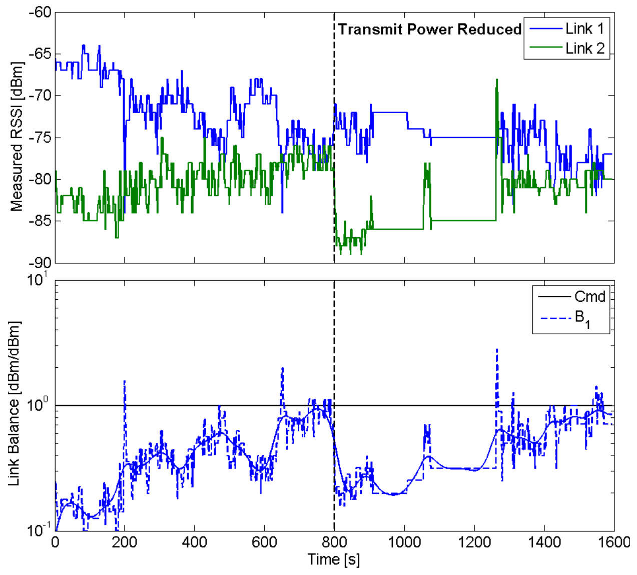


Figure 12: Link power and balance state time history during hardware configuration change experiment

An overhead view of robot position traces is shown in Figure 11, where each subplot corresponds to a different time window. The top plot for time t=[0:800] demonstrates link balancing and position cross track control for the nominal hardware configuration. From its initial position, the mobile relay robot turns around and moves toward a link equilibrium near the geometric midpoint. Figure 12 shows the raw received signal strength indication (RSSI) values and the balance ratio between them. In this first time period, the raw values converge and the balance moves towards commanded unity. At time t=800 seconds, the payload node transmitter power (link #2) is intentionally reduced, decreasing the measured RSSI value and altering the equilibrium position. As can be seen in the second overhead plot of Figure 11, for time t=[800:1600], the mobile relay compensates by moving closer to the end node with the reduced transmission power. This motion reestablishes link balance as indicated by the signal balance returning to unity in Figure 12. By directly measuring the parameters of interest, the system reacts to dynamic changes in the hardware and compensates by moving to maintain commanded parameters.

2.6 Chapter Summary

In summary, this chapter presents an architecture and design methodology for controlling multi-robot motion to perform a specific task. The original work is extended form the cluster space control technique to an arbitrary number of control spaces. Designing a task requires defining all spaces and states, relating these states through kinematic transformations and Jacobian matrices, and state controllers within each space. The system performance may be analyzed using classical and Lyapunov techniques.

3 Multi-Task Space Control

With the availability of a formal method to perform individual tasks, we now turn our attention to collaboration between multiple tasks. Each task is performed by multiple robots which we term a "coalition", and multiple tasks are performed by multiple coalitions which we term a "federation". To empower complex, motion-oriented missions through a federation of collaborating task-level multi-robot coalitions, we integrate multiple task-level controllers into a novel, formalized control architecture. First, a compact and integrated mathematical model of the task-level controllers is established. Second, re-allocation of robots among tasks is integrated through control logic that conserves the dimensionality of the federation's state space and kinematic transforms. Third, task coordination is modeled explicitly, facilitating federation-level analysis given the coupling of task-level coalitions.

3.1 Unified Multi-Task Representation

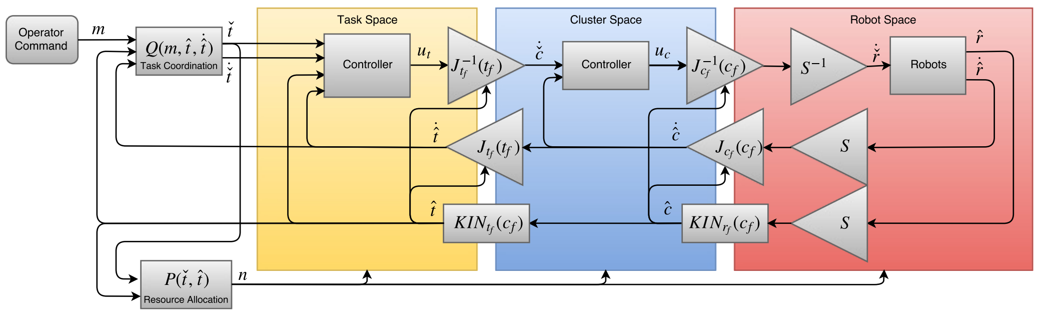


Figure 13: The unified control block diagram. The layered control space architecture utilizes robot, cluster, and task spaces. Task coordination and resource allocation functions enable collaboration between tasks

Consider a collection of the task-specific multi-robot coalition control systems shown in Figure 13. For a federation of o coalitions in this architecture, the unified task-level transfer function for multiple independently operating coalitions is:

$$G_{t_f} = \begin{bmatrix} G_{t_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & G_{t_o} \end{bmatrix} = \left(J_{t_f}^{-1} sI + G_{c_f} J_{t_f}^{-1} K_{t_f}\right)^{-1} G_{c_f} J_{t_f}^{-1} K_{t_f} \tag{50}$$

Where the federated versions of gains (K_{c_f}, K_{t_f}), position kinematics ($KIN_{c_f}(r_f)$, $KIN_{t_f}(c_f)$), and Jacobian matrices ($J_{c_f}, J_{t_f}, J_{c_f}^{-1}, J_{t_f}^{-1}$) are used. These quantities, as well as the internal signals within the systems, are concatenations or block diagonal quantities composed of the affiliated coalition quantities:

For the robot allocation vector:

$$\vec{n} = [n_1, n_2, \dots, n_o]^T \text{ where } n = \textstyle\sum_{i=1}^{o} n_i \tag{51}$$

For the federated pose vector concatenations:

$$r_f = [r_1^T \quad \dots \quad r_o^T]^T \tag{52}$$

$$c_f = [c_1^T \quad \dots \quad c_o^T]^T \tag{53}$$

$$t_f = [t_1^T \quad \dots \quad t_o^T]^T \tag{54}$$

For the federated velocity vector concatenations:

$$\dot{r}_f = [\dot{r}_1^T \quad \dots \quad \dot{r}_o^T]^T \tag{55}$$

$$\dot{c}_f = [\dot{c}_1^T \quad \dots \quad \dot{c}_o^T]^T \tag{56}$$

$$\dot{t}_f = [\dot{t}_1^T \quad \dots \quad \dot{t}_o^T]^T \tag{57}$$

For the federated kinematic transformation concatenations:

$$KIN_{c_f}(r_f) = \left[KIN_{c_1}(r_1)^T \quad \dots \quad KIN_{c_o}^T(r_o)\right]^T \tag{58}$$

$$KIN_{t_f}(c_f) = \left[KIN_{t_1}(c_1)^T \quad \dots \quad KIN_{t_o}^T(c_o)\right]^T \tag{59}$$

For the federated Jacobian matrix block-diagonalizations:

$$J_{c_f} = \begin{bmatrix} J_{c_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & J_{c_o} \end{bmatrix} \tag{60}$$

$$J_{t_f} = \begin{bmatrix} J_{t_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & J_{t_o} \end{bmatrix} \tag{61}$$

For the federated controller gain block-diagonalizations:

$$K_{c_f} = \begin{bmatrix} K_{c_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_{c_o} \end{bmatrix} \tag{62}$$

$$K_{t_f} = \begin{bmatrix} K_{t_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & K_{t_o} \end{bmatrix} \tag{63}$$

For the federated control effort concatenations:

$$u_{c_f} = \begin{bmatrix} u_{c_1}^T & \cdots & u_{c_o}^T \end{bmatrix}^T \tag{64}$$

$$u_{t_f} = \begin{bmatrix} u_{t_1}^T & \cdots & u_{t_o}^T \end{bmatrix}^T \tag{65}$$

where subscript f denotes federated elements; n_i is the number of robots assigned to task i, where there are a total of n robots in the mission; o is the number of tasks spanning the mission, r_i, c_i and t_i are the robot space, cluster space, and task space pose vectors for task i; $KIN_{c_i}(\vec{r}_i)$ and $KIN_{t_i}(c_i)$ are the cluster space and task space kinematic equations for task i; $J_{c_i}$and J_{t_i} are the cluster space and task space Jacobian matrices for task i; $K_{c_i}$and K_{t_i} are the cluster space and task space feedback matrices for task i; and u_{c_i} and u_{c_t} are the control efforts for cluster and task for task i. This approach maintains consistent dimensions of the control elements despite robot reassignments. Doing so allows the use of traditional dynamics and control system design techniques. The implementation must manage the changing coalition and task definitions, but the form remains the same.

3.2 Resource Allocation

3.2.1 Goals

Our architecture incorporates resource allocation as one function within a collaborative system. Although our work does not focus on innovations in resource allocation techniques, our architecture incorporates the use of such techniques in a novel manner and considers dynamic behavior which many existing techniques fail to address. In the context of integrated motion control across task-specific multi-robot coalitions, there are two allocation issues. The first is to determine how many robots to assign to each task's multi-robot group given a limited number of robots available within the federation. Given this, the second is to determine which robot should be assigned to the specific positions within each task as depicted in Figure 14.

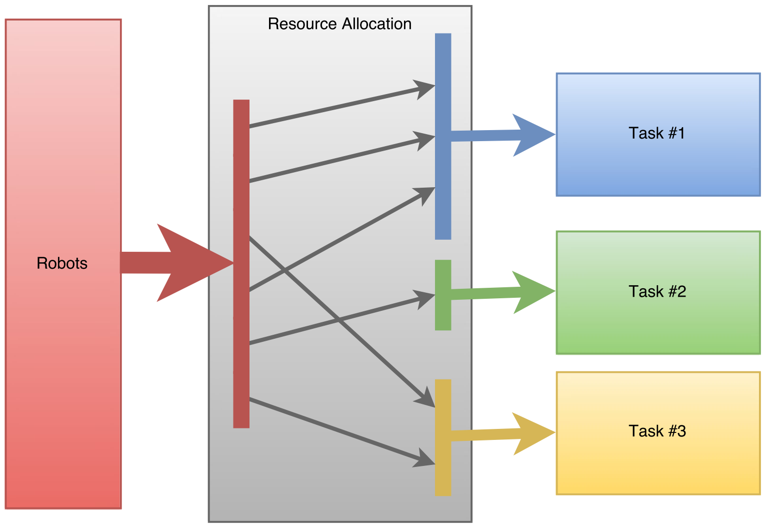


Figure 14: Resource Allocation Depiction

3.2.2 Method

3.2.2.1 Allocating the Number of Robots to Tasks

The first challenge, determining how many mobile robots to assign to each coalition, arises since it may be desirable to change this allocation over time. This may be due to the need to accommodate varying mission needs, new environmental conditions, or changes in the state of the federation. In our approach, we include all available robots in the federation. Because all robots may not be necessary to meet mission needs, some robots may be assigned to an idle task that maintains otherwise unused robots in a holding state.

For our work, we use a state machine to control the number of robots assigned to each task-specific coalition. Transition logic can be established to implement policies relevant to the mission at hand. In general, this logic may be a function of the system's state (e.g., poor performance for a task variable may necessitate an increase in robots assigned to that task), external variables, the priorities among task, and the nature of the tasks themselves (e.g., some may have a minimum number required in order to function). We define the allocation policy below:

$$\vec{n} = P(\check{t}, \hat{t}, x, \ldots) \tag{66}$$

The allocation policy P specifies the number of robots assigned to each task $\vec{n}$, based on desired task states $\check{t}$, actual task states $\hat{t}$, exogenous states x, and any other relevant factors.

As the allocation function changes the number of robots assigned to each task, it triggers control logic that loads new gains and kinematic transforms into each of the affected coalition controllers; in some sense, this may be considered to be an extended form of a gain scheduling adaptive control strategy. An attractive feature of the unified representation of the federation given by (50)-(65) is that the dimension and control architecture of the overall federation remains constant, which aids performance analysis and control implementation. Of course, transients can temporarily erode performance, and stability is certainly a concern for switched controllers; we address this in Section 3.2.3.

3.2.2.2 Assigning Specific Robots to Task Roles

The second challenge involves determining which specific robot should fulfill what role in each particular coalition. This may require a change over time and is a function of considerations such as robot capabilities/limitations, functional health, the state of consumables, and position.

For our work to date with homogeneous federations, we have adopted relatively simple assignment strategies ranging from arbitrary selection to the use of proximity tests to the minimization of errors. Again, our focus is not on innovations in resource allocation but in how to incorporate the resulting allocations into our integrated motion control architecture. For examples of state of the art methods, see [**31**] [**32**] [**33**].

From that perspective, we make a distinction between the robot hardware index and the actual robot assignment index to a role and coalition within the federation. An assignment matrix S is used to map between these two indices. This matrix maps the states of the numbered robots to the federation state vector; its inverse maps the robot space command vector to the numbered robots. The matrix takes the form of a permutation matrix consisting of quantity o identity matrices having a dimension equal to the degrees-of-freedom of that robot.

$$r' = Sr \text{ where } s_{m(i-1)+[1:m],m(j-1)+[1:m]} = I_m \text{ for robot I} \tag{67}$$

where r' represents the assigned robot vector, r represents the indexed robot vector, and S represents the assignment matrix. Agent index i is assigned to role j. I_m is the identity matrix of dimension m corresponding to the degrees of freedom of the robot. The lengthy subscript terms of s maintain consistent dimensions. The remaining analysis presented in this paper assumes these two vectors are equivalent for convienence but without loss of generality.

3.2.3 Analysis

We wish to establish conditions that will guarantee stability during robot reallocation. These conditions define when it is “safe” for a resource allocation algorithm to move robots between tasks without driving the system unstable. Even if every individual configuration of the system is stable, the system may be driven unstable through poor choices in switching [**66**]. Given a family of dynamic systems:

$$\dot{x} = f_p(x) \text{ where } p \in \mathcal{P} \tag{68}$$

where p denotes the switched-state index and $\mathcal{P}$ denotes the index set. From [**66**], a continuous positive definite function V is a common Lyapunov function if there exists a continuous positive definite function W such that:

$$\frac{\partial V}{\partial x} f_p(x) \leq -W(x) \tag{69}$$

then the switched system is stable in the Lyapunov sense. This approach requires that equilibria do not change with switched state and there are no instantaneous changes in state at switches (impulse effects). To meet these criteria, we select a quadratic function of robot velocity, where the hat accent, $\hat{x}$, represents a state value and the breve accent, $\breve{x}$, represents a desired value:

$$V = \frac{1}{2}\dot{\hat{r}}^T \dot{\hat{r}} \tag{70}$$

Finding the rate of change of the common Lyapunov function:

$$\dot{V} = \dot{\hat{r}}^T \ddot{\hat{r}} \tag{71}$$

Introducing the robot dynamics, assumed to be second order with mass matrix M, damping matrix B, and a proportional velocity feedback control loop with gain K_r:

$$M\ddot{\hat{r}} + B\dot{\hat{r}} = K_r\left(\dot{\breve{r}} - \dot{\hat{r}}\right) \rightarrow \ddot{\hat{r}} = M^{-1}\left(K_r\left(\dot{\breve{r}} - \dot{\hat{r}}\right) - B\dot{\hat{r}}\right) \tag{72}$$

$$\dot{V} = \dot{\hat{r}}^T M^{-1}\left(K_r\left(\dot{\breve{r}} - \dot{\hat{r}}\right) - B\dot{\hat{r}}\right) \tag{73}$$

Introducing the cluster (or formation) space control with a velocity feedback control loop with gain $K_{c,p}$, noting that it is a function of switched state:

$$\dot{\breve{r}} = J_{c,p}^{-1} K_{c,p}\left(\dot{\breve{c}}_p - \dot{\hat{c}}_p\right) = J_{c,p}^{-1} K_{c,p}\left(\dot{\breve{c}}_p - J_{c,p}\dot{\hat{r}}\right) \tag{74}$$

$$\dot{V} = \dot{\hat{r}}^T M^{-1}\left(K_r\left(J_{c,p}^{-1} K_{c,p}\left(\dot{\breve{c}}_p - J_{c,p}\dot{\hat{r}}\right) - \dot{\hat{r}}\right) - B\dot{\hat{r}}\right) \tag{75}$$

Introducing the task space control with a state feedback control loop with gain $K_{t,p}$:

$$\dot{\breve{c}} = J_{t,p}^{-1} K_t(\breve{t} - \hat{t}) = J_{t,p}^{-1} K_t e_t \tag{76}$$

$$\dot{V} = \dot{\hat{r}}^T M^{-1}\left(K_r\left(J_{c,p}^{-1}K_{c,p}\left(J_{t,p}^{-1}K_{t,p}e_{t,p} - J_{c,p}\hat{r}\right) - \hat{r}\right) - B\dot{\hat{r}}\right) \tag{77}$$

Expanding and adding the condition W for a common Lyapunov function:

$$\dot{V} = \dot{\hat{r}}^T M^{-1}\left(K_r J_{c,p}^{-1}K_{c,p}\left(J_{t,p}^{-1}K_{t,p}e_{t,p} - J_{c,p}\hat{r}\right) - (K_r + B)\dot{\hat{r}}\right) \leq -W \tag{78}$$

We can set the function W to bound the Lyapunov function based on the robot dynamics to cancel one of the terms.

$$W = \dot{\hat{r}}^T M^{-1}(K_r + B)\dot{\hat{r}} \tag{79}$$

$$\dot{V} = \dot{\hat{r}}^T M^{-1} K_r J_{c,p}^{-1}K_{c,p}\left(J_{t,p}^{-1}K_{t,p}e_{t,p} - J_{c,p}\hat{r}\right) \leq 0 \tag{80}$$

Taking the norm of the equation allows further simplifications by canceling terms while maintaining conservative bounds of the inequality:

$$\leq \left\|J_{t,p}^{-1}\right\|\left\|K_{t,p}\right\|\left\|e_{t,p}\right\| - \left\|J_{c,p}\right\|\left\|\hat{r}\right\| \leq 0 \tag{81}$$

Finally, rearranging the terms yields a stability condition for the switched system:

$$\left\|J_{c,p}^{-1}\right\|\left\|J_{t,p}^{-1}\right\|\left\|K_{t,p}\right\|\left\|e_{t,p}\right\| \leq \left\|\hat{r}\right\| \; \forall p \in \mathcal{P} \tag{82}$$

The switched system is stable in the Lyapunov sense if the commanded rate (in any space) is less than or equal to the current rate (in the same space) for all configurations of the switched system. Intuitively, this will naturally converge. Practically, this expression provides a simple, analytic condition to ensure stable switching. If considering a switch, candidate configurations (i.e. coalitions composed of different robots) can now be evaluated. This result provides a rigorous basis for aggressive switching, which is far superior to naïvely or ignorantly waiting for transients to settle as in the case of slow switching or ad hoc methods.

Should the condition not be met, the expression also provides some suggestions. Switching preparation could occur by moving the robots to reduce the initial error in the new task. Switching could occur gradually, slowly transitioning to control gains of the new configuration to maintain low control effort even if the task error is high. Switching could occur near a singularity of the task Jacobian inverse so there is minimal authority, though this has other practical challenges and is not recommended. Finally, switching could change to a different configuration, allocating different quantities or assigning different robots that meet the condition.

3.3 Task Coordination

3.3.1 Goals

Coordination is a second facet of collaborative systems. Robust collaboration should be performed with explicit coordination and feedback between tasks. Explicit coordination provides system agility, where tasks can rapidly alter their goals. Feedback ensures synchronization for highly coupled missions that may otherwise fail due to disturbances or other non-ideal behavior.

3.3.2 Method

The final element in our integrated motion control architecture consists of a task coordination and collaboration function. The coordination aspect of this function pertains to assigning mission-level federation goals to individual task-specific coalitions. Collaboration implies coalition interaction, and this is accomplished by making coalition-specific goals a function of the output states of other coalitions.

Formally, this functionality is achieved by a function that has mission-level goal set-points and the federation state vector as an input; the output of the function is the set of goal set-points for every task-specific coalition within the federation. The function, shown in (83) leads to behavioral coupling between coalitions within the federation, which is the power of a collaborative multi-robot system.

In the most general case, the coordination function may include definition of the kinematic equations of the task, allowing task states to be defined relative to other states or external variables. The general case, depicted in Figure 15, provides complete freedom with state definition, which is powerful, though perhaps inelegant. This provides complete flexibility for the system to be designed in accordance with operator preference and intuitive behavior.

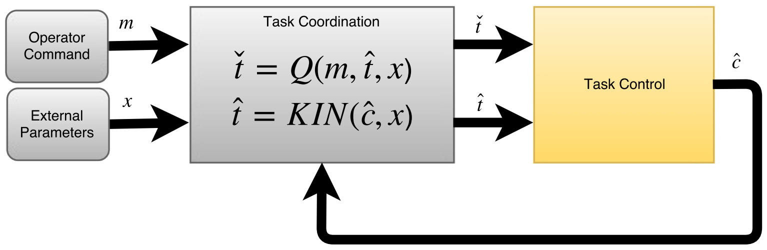


Figure 15: General task coordination block diagram. Task commands $\check{t}$ are a function Q of mission-level goal set points m, actual task states $\hat{t}$, and external parameters x. Task state $\hat{t}$ is a function of coalition state $\hat{c}$ and external parameters x. These functions add cross coupling between task states which results in explicit, closed-loop task coordination.

In many common cases, task coordination defines task set points exclusively as functions of other task states and the task states are defined exclusively by cluster states, avoiding redefining the kinematic transformations. This common case is depicted in Figure 16 and represented mathematically in (86).

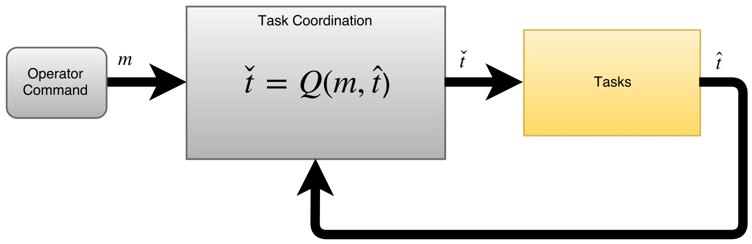


Figure 16: A common special case of the task coordination block diagram without modifications to task state definitions. Task commands $\check{t}$ are exclusively a function Q of mission-level goal set points m and actual task states $\hat{t}$.

3.3.3 Analysis

The resulting behavior can be formally characterized by development of the full dynamic model of the controlled federation, as shown in (85).

The general task coordination function, including state definition:

$$\check{t} = Q(m, \hat{t}, x) \tag{83}$$

$$\hat{t} = KIN(\hat{c}, x) \tag{84}$$

The coordinated dynamic model is:

$$\hat{t} = G_t Q(m, \hat{t}, x) \tag{85}$$

where G_t is defined as in (50). If the task coordination function is of the form (86), where task commands are decomposed to linear combinations of m and $\hat{t}$ and without external parameters x influencing the state definition, the coordinated dynamic model results in (87). This coordinated model shows cascading dynamics; the independent task dynamics influence the dependent task dynamics. It is possible that mutually dependent tasks, which would create a feedback loop, could destabilize the system. These equations allow analysis of system stability and performance.

The linear task coordination function is:

$$\check{t} = Q_I m + Q_D \hat{t} \tag{86}$$

The linear coordinated dynamic model is:

$$\hat{t} = (I - G_t Q_D)^{-1} G_t Q_I m \tag{87}$$

Mission-level task performance is determined by the task allocation function. The metrics may be identical to an independent task, such as transient rise time or steady state error, but the comparison to the ideal (i.e. performance) may be influenced by coupled states. For the given example, using the metric of following error, the following robot may have good performance with minimal following error for a stationary leading robot but poor performance with large following error for a leader that moves quickly or erratically. Preferably, the task coordination function should be designed in a way that allows intuitive specification of mission tasks and performance metrics by the operator.

3.3.4 Example: 1-DOF, 2 Task Following

As an example of the Q function, consider two multi-robot coalitions with one-dimensional motion in a simple mission in which the first coalition is commanded to go to a specific location m_1 and the second coalition is commanded to follow m_2 units behind.

$$t_d = \begin{bmatrix} m_1 \\ t_1 - m_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{88}$$

$$\hat{t} = \begin{bmatrix} g_1 & 0 \\ g_1 g_2 & -g_2 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} g_1 m_1 \\ g_1 g_2 m_1 - g_2 m_2 \end{bmatrix} \tag{89}$$

Using a first-order lag to represent the task dynamics, $g_i = \frac{1}{s+1}$, and mission commands of $m = [20 \quad 5]^T$, the system response below shows coupling of the tasks.

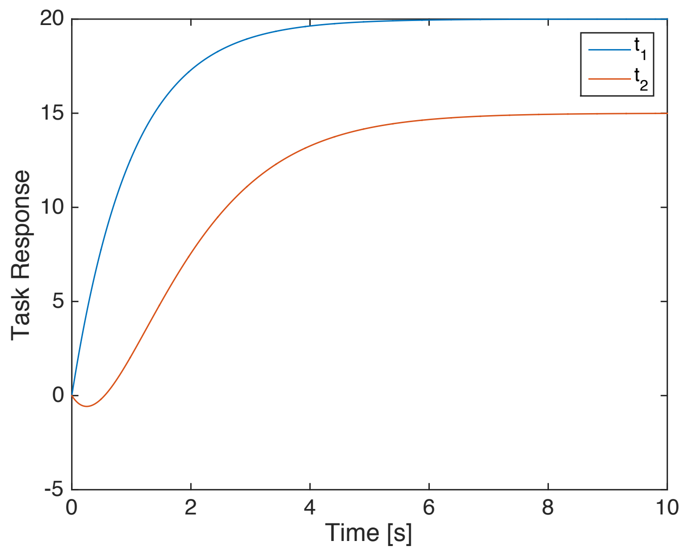


Figure 17: Time history of task states for a simple coordinated following mission. The second task has a higher order response because it follows the first task, which couples their dynamics.

3.4 Mission Examples:

3.4.1 Long Range Communications

These experimental results demonstrate the control of link quality and balance with a mobile endpoint, order to demonstrate performance of the control architecture given real-world challenges. The experimental testbed consists of multiple mobile terrestrial robots with onboard wireless modems capable of sensing communication link quality. A detailed description of the testbed is provided in Appendix A.

The experiment starts with the end stations near each other and directly communicating, with two relay robots in an idle position. As the mobile end station moves away, the two relay robots are sequentially added to the communication task in order to maintain the specified level of link quality and balance.

3.4.1.1 Control Space Definition

As a simple example of a mission comprised of multiple tasks, consider the following mission: maintain communication between two end points or otherwise return to idle parking position. A subset of the federated control space elements is shown below for two configurations of a $n = 3$ robot system: 1) one robot is allocated to the communications task and two robots are idle ($\vec{n} = [1,2]^T$) and 2) two robots are allocated to the communications task and one robot is idle ($\vec{n} = [2,1]^T$):

The federated Cluster State Vector:

$$\vec{c}_M = \begin{cases} \left[[x_c, y_c, \theta_c]\left[x_{I_1}, y_{I_1}, \theta_{I_1}, x_{I_2}, y_{I_2}, \theta_{I_2}\right]\right]^T \text{ for } \vec{n} = [1,2]^T \\ \left[[x_c, y_c, \theta_c, \rho_1, \alpha_1, \phi_1]\left[x_{I_1}, y_{I_1}, \theta_{I_1}\right]\right]^T \text{ for } \vec{n} = [2,1]^T \end{cases} \tag{90}$$

The federated Cluster Jacobian:

$$J_M = \begin{cases} \begin{bmatrix} \begin{bmatrix} \frac{\partial x_c}{\partial x_1} & \frac{\partial x_c}{\partial y_1} & \frac{\partial x_c}{\partial \theta_1} \\ \frac{\partial y_c}{\partial x_1} & \frac{\partial y_c}{\partial y_1} & \frac{\partial y_c}{\partial \theta_1} \\ \frac{\partial \theta_c}{\partial x_1} & \frac{\partial \theta_c}{\partial y_1} & \frac{\partial \theta_c}{\partial \theta_1} \end{bmatrix} & \\ & \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \end{bmatrix} \text{for } \vec{n} = [1,2]^T \\ \begin{bmatrix} \begin{bmatrix} \frac{\partial x_c}{\partial x_1} & \frac{\partial x_c}{\partial y_1} & \frac{\partial x_c}{\partial \theta_1} & \frac{\partial x_c}{\partial x_2} & \frac{\partial x_c}{\partial y_2} & \frac{\partial x_c}{\partial \theta_2} \\ \frac{\partial y_c}{\partial x_1} & \frac{\partial y_c}{\partial y_1} & \frac{\partial y_c}{\partial \theta_1} & \frac{\partial y_c}{\partial x_2} & \frac{\partial y_c}{\partial y_2} & \frac{\partial y_c}{\partial \theta_2} \\ \frac{\partial \theta_c}{\partial x_1} & \frac{\partial \theta_c}{\partial y_1} & \frac{\partial \theta_c}{\partial \theta_1} & \frac{\partial \theta_c}{\partial x_2} & \frac{\partial \theta_c}{\partial y_2} & \frac{\partial \theta_c}{\partial \theta_2} \\ \frac{\partial \rho_1}{\partial x_1} & \frac{\partial \rho_1}{\partial y_1} & \frac{\partial \rho_1}{\partial \theta_1} & \frac{\partial \rho_1}{\partial x_2} & \frac{\partial \rho_1}{\partial y_2} & \frac{\partial \rho_1}{\partial \theta_2} \\ \frac{\partial \alpha_1}{\partial x_1} & \frac{\partial \alpha_1}{\partial y_1} & \frac{\partial \alpha_1}{\partial \theta_1} & \frac{\partial \alpha_1}{\partial x_2} & \frac{\partial \alpha_1}{\partial y_2} & \frac{\partial \alpha_1}{\partial \theta_2} \\ \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial y_1} & \frac{\partial \phi_1}{\partial \theta_1} & \frac{\partial \phi_1}{\partial x_2} & \frac{\partial \phi_1}{\partial y_2} & \frac{\partial \phi_1}{\partial \theta_2} \end{bmatrix} & \\ & \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} \end{bmatrix} \text{for } \vec{n} = [2,1]^T \end{cases} \tag{91}$$

In this example, the robot allocation policy is dictated by the added state of communication link quality representing a characterization of the full communication chain, defined below:

$$Z \triangleq \frac{n_1+1}{\sum_{i=1}^{n_1-1}\frac{1}{s_i}} = \frac{k(n_1+1)}{\sum_{i=1}^{n_1-1}\rho_i^2} \qquad (92)$$

Allocation is done according to the policy defined below, much like the transition policy of a state machine:

Table 1: Example allocation policy for a communication + idle mission

Link Quality Policy	Current Allocation	Next Allocation
$Z < \frac{1}{4}Z_d$	$\vec{n} = [n_1, n_2]^T$	$\vec{n} = [n_1 + 1, n_2 - 1]^T$
$Z > 4\,Z_d$	$\vec{n} = [n_1, n_2]^T$	$\vec{n} = [n_1 - 1, n_2 + 1]^T$

The policy determines whether to add or subtract robots from the communication cluster. If the link quality falls below the lower threshold of this deadband, a robot is shifted from the idle cluster to the communications cluster. If the link quality rises above the upper threshold of this deadband, a robot is shifted from the communications cluster to the idle cluster. While simplistic, this yields acceptable system behavior and is easily accommodated by the control framework.

3.4.1.2 Experimental Results

3.4.1.2.1 Link Quality Command Response Experiment

This scenario demonstrates changing user requirements for better connectivity or higher throughput forcing a change in the cluster configuration. The communication endpoints are fixed and the link quality command is increased, prompting robots to be reallocated from the idle cluster to the communication relay cluster. Each newly incorporated robot moves from its idle position to the communication task, assisting with control of the commanded link quality and link balance states. Results were obtained using the test bed described in Appendix A.

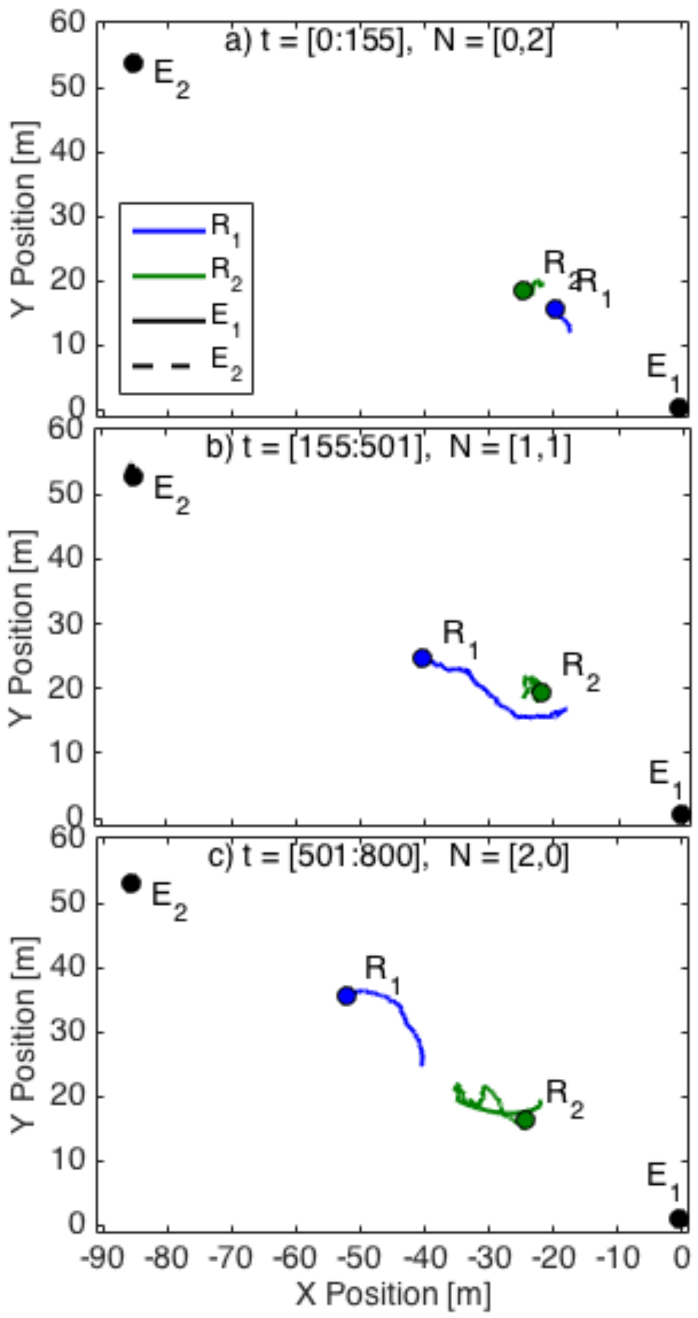


Figure 18: Overhead view of positions of robots *R* and exogenous nodes *E* at specified times while evaluating the link quality command response

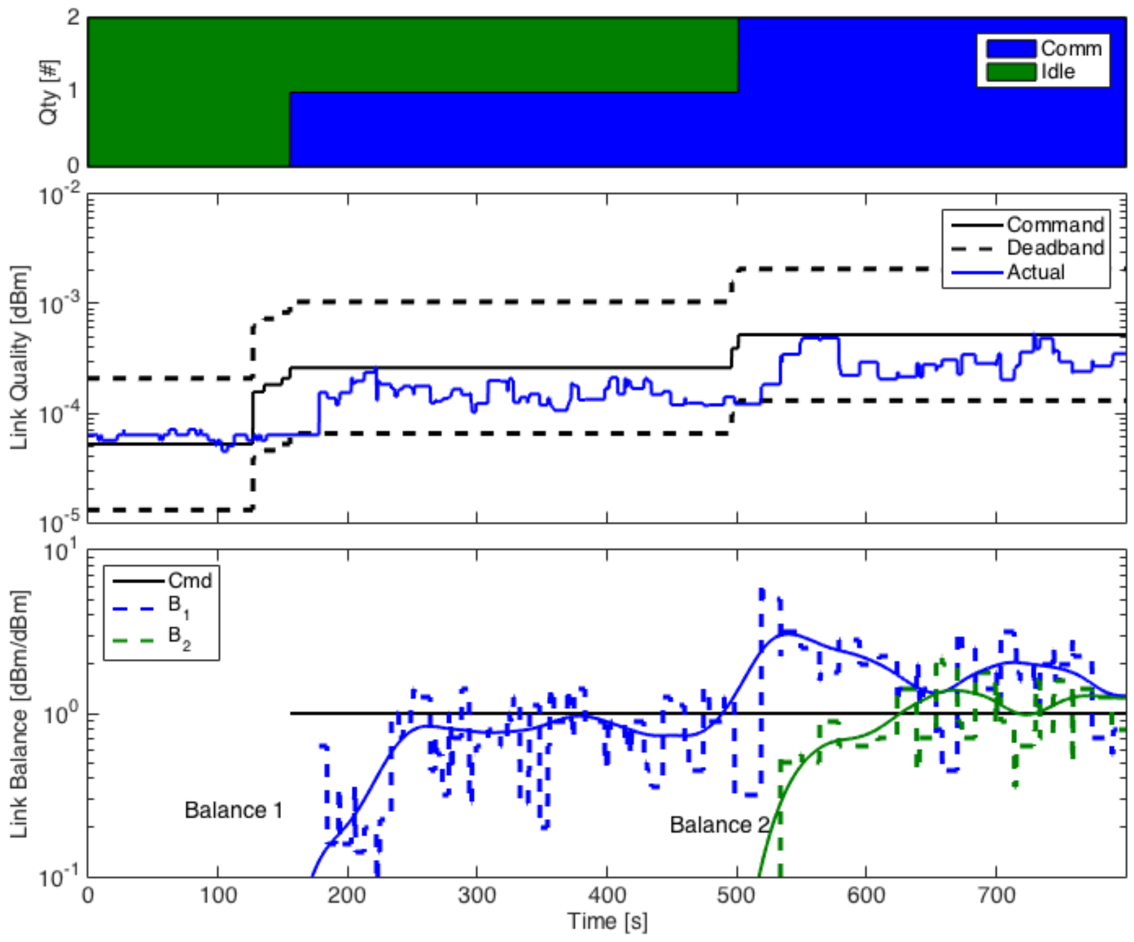


Figure 19: Time history of key states while evaluating the link quality commanded response, forcing configuration change

The top view of Figure 18 for time $t = [0{:}155]$ shows the fixed exogenous end points $E_{1:2}$ and the idle robots $R_{1:2}$ for configuration $N = [0,2]^T$. At this point, the link quality command is increased, as seen in Figure 19, triggering a reallocation, as seen in the middle view of Figure 18 for time $t = [155{:}501]$, and the newly activated robot settles at an equilibrium point near the center of the two exogenous nodes. The command is again increased, triggering another reallocation as seen in the bottom view of Figure 18 for time $t = [501{:}800]$ where both relay robots $R_{1:2}$ move to balance the three links.

The time history plots in Figure 19 show that the sensed RSSI parameters exhibited appreciable quantization and inconsistent sampling. Sensitivity to other parameters, such as robot orientation (due to onboard antenna obstruction), was also noted. It can also be seen that the robots do not move to the geometric center of the end points but instead have a slight bias because of lower transmission fields or steady state offset from the proportional controller. These real-world phenomena are challenging but the control architecture is sufficiently robust to tolerate these unmodeled effects.

3.4.1.2.2 Mobile Endpoints, Simulation

This simulation demonstrates control of link quality and balance with mobile endpoints, gracefully adding and subtracting robots as appropriate for the task.

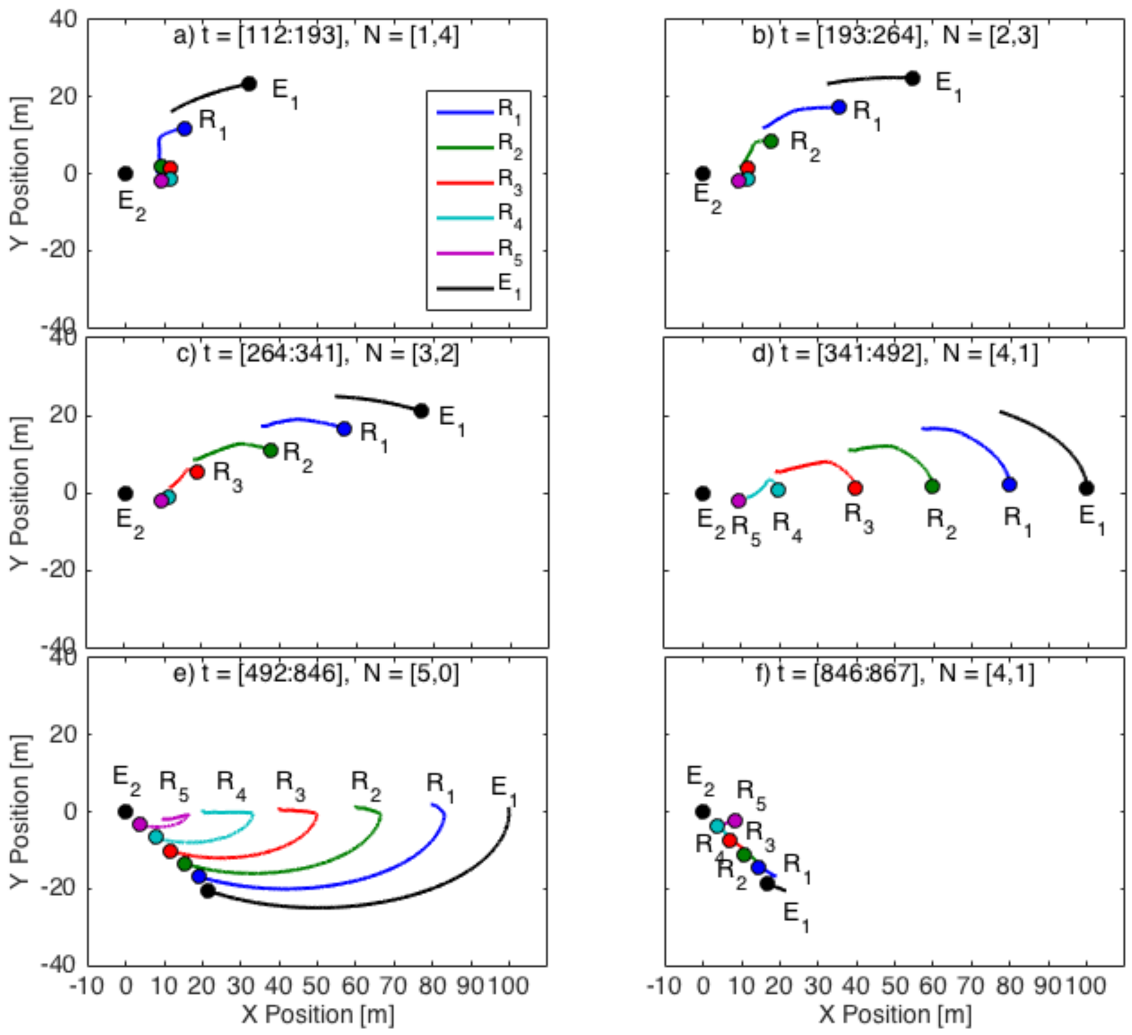


Figure 20: Overhead view of robots *R* and exogenous nodes *X* during specified times for mobile endpoint simulation

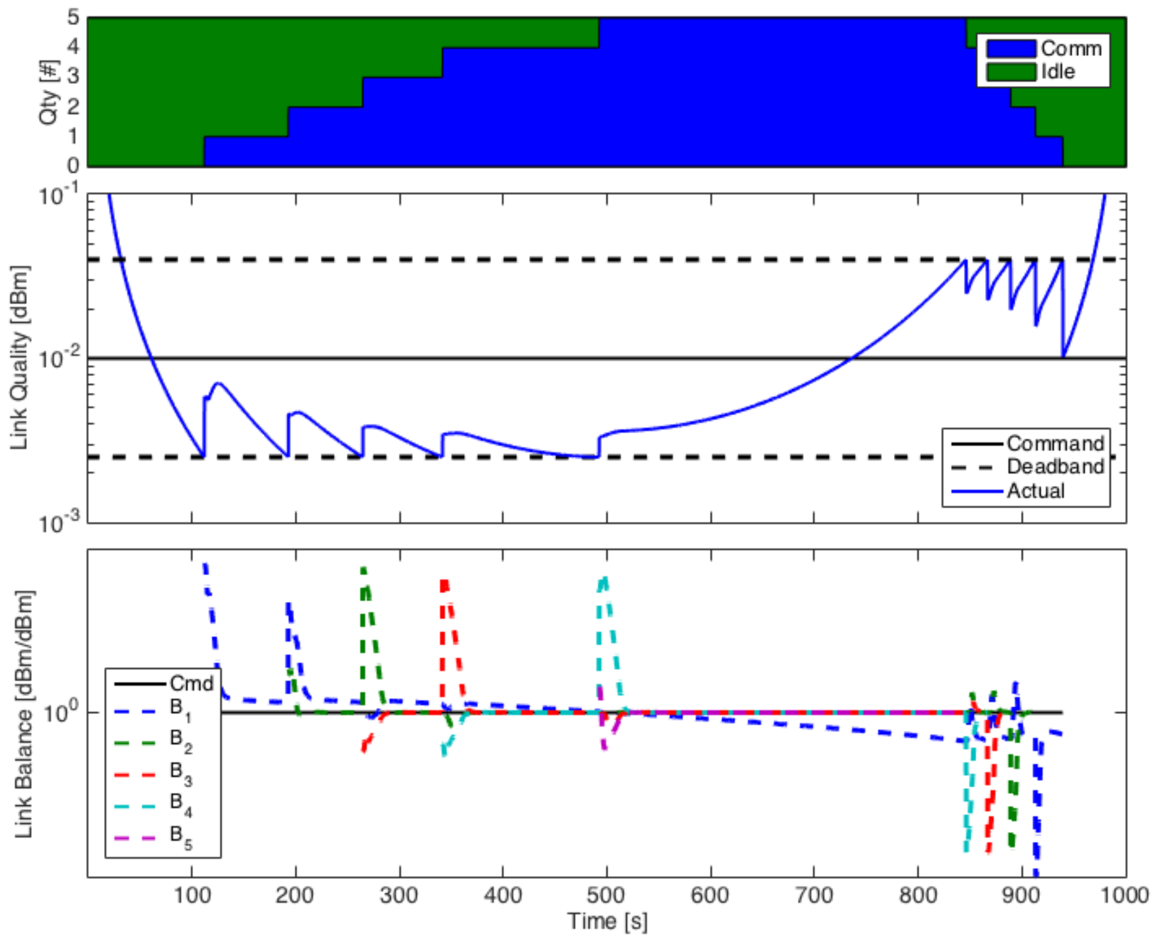


Figure 21: Time history of key system states for mobile endpoint simulation

As the mobile end node progresses through an ellipse, the robots $R_{1:3}$ respond to changing link values by following its motion. Initially, though not shown, all robots are part of the idle cluster and park themselves around $(10,0)$. One by one the robots are moved to the communication task as the link quality drops below the command deadband, seen in Figure 21. The top-left overhead view shows time $t = [193{:}265]$ during which the communications cluster has two robots and the idle cluster has three robots, denoted $N = [2,3]^T$. These robots are commanded by the communication space controller to minimize crosstrack error and balance the measured signal strength, as is plotted in Figure 21, which results in even spacing between the end points. In this first overhead view, robot R_2 can be seen moving from its previously idle position to join the communications cluster with robot R_1 which raises the link quality back within the deadband. During this time, the idle robots $R_{3:5}$ maintain their position at the commanded parking location until they are needed. At time $t = 265$, the link quality state falls outside the control deadband and the allocation policy moves previously idle robot R_3 to the communication task and the robots adjust to maintain balance.

This process is repeated until all robots are part of the communications tasks. As the mobile end node returns toward the base node, the link quality increases until it rises outside the deadband at time $t = 845$ and robot R_5 is reallocated to the idle task which reduces the link quality within the deadband. This process is repeated until all robots have returned to idle. Interestingly, the deadband causes unequal times between transitions as the robots are slower to move into the communication cluster and faster to move out due to the task state definition and allocation policy.

This demonstrates the ability of the control architecture to respond to motion of the exogenous end nodes based on sensed link characteristics and reallocate themselves without any addition command.

3.4.1.2.3 Mobile Endpoints, Experiment

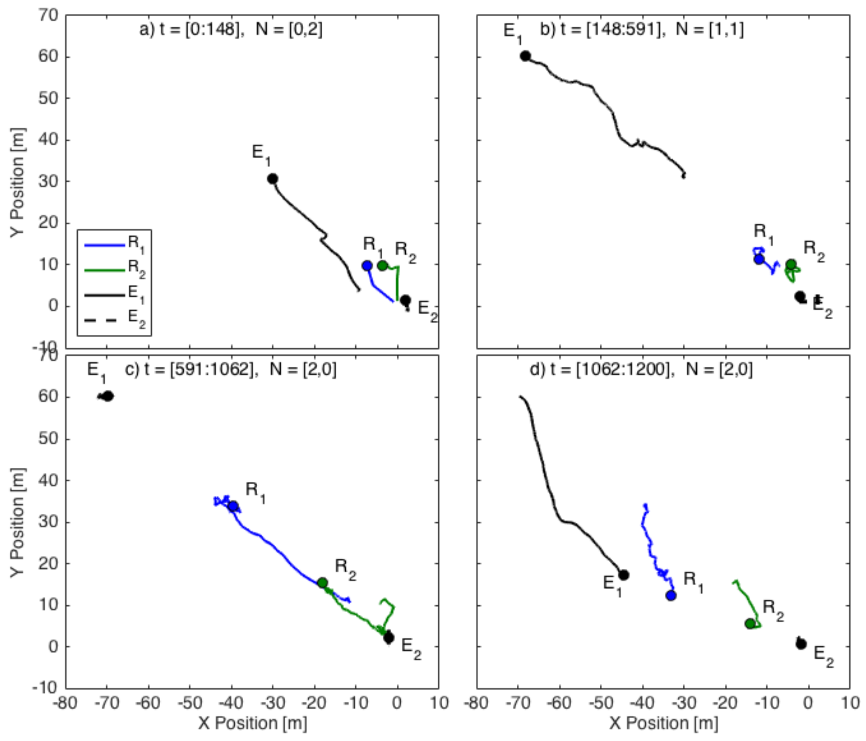


Figure 22: Overhead view of robots *R* and exogenous nodes *E* during specified times for the mobile endpoint experiment

Figure 22 shows the paths taken by the robots and endpoints and Figure 23 shows the corresponding state trajectories. In Figure 22a, for time $t = [0{:}148]$, the mobile endpoint moves away from the stationary endpoint while the link quality remains within the deadband. The robots are allocated to idle, $\vec{n} = [0,2]^T$, and can be seen parking themselves.

At time $t = 148$, the link quality exceeds the lower bounds of the deadband and the allocation policy adds a robot to the communication relay task, changing the configuration vector to $\vec{n} = [1,1]^T$. In Figure 22b, for time $t = [148{:}591]$, the new robot relay moves to balance the communication links while the mobile endpoint continues moving away from the stationary endpoint. Though there is not significant movement of the relay robot, the measured link states, shown in Figure 23, indicate that the balance set point is achieved during this time. This demonstrates the complexity and non-intuitiveness of RF fields and the benefit of communication-space measurement and control; alternatively locating the relay node in the geometric center of the two points would yield worse performance.

At time $t = 591$, the link quality again exceeds the lower bounds of the deadband and the allocation policy adds the second robot to the communication relay task, changing the configuration vector to $\vec{n} = [2,0]^T$. In Figure 22c, for time $t = [591{:}1062]$, both robots move to balance the communication links. The switching transient can be seen in Figure 23, starting at t= ~600sec and settling by t= ~950sec. The final overhead plot, Figure 22d, shows the mobile endpoint arcing back towards the stationary endpoint and the relay robots mimic its motion to maintain link balance.

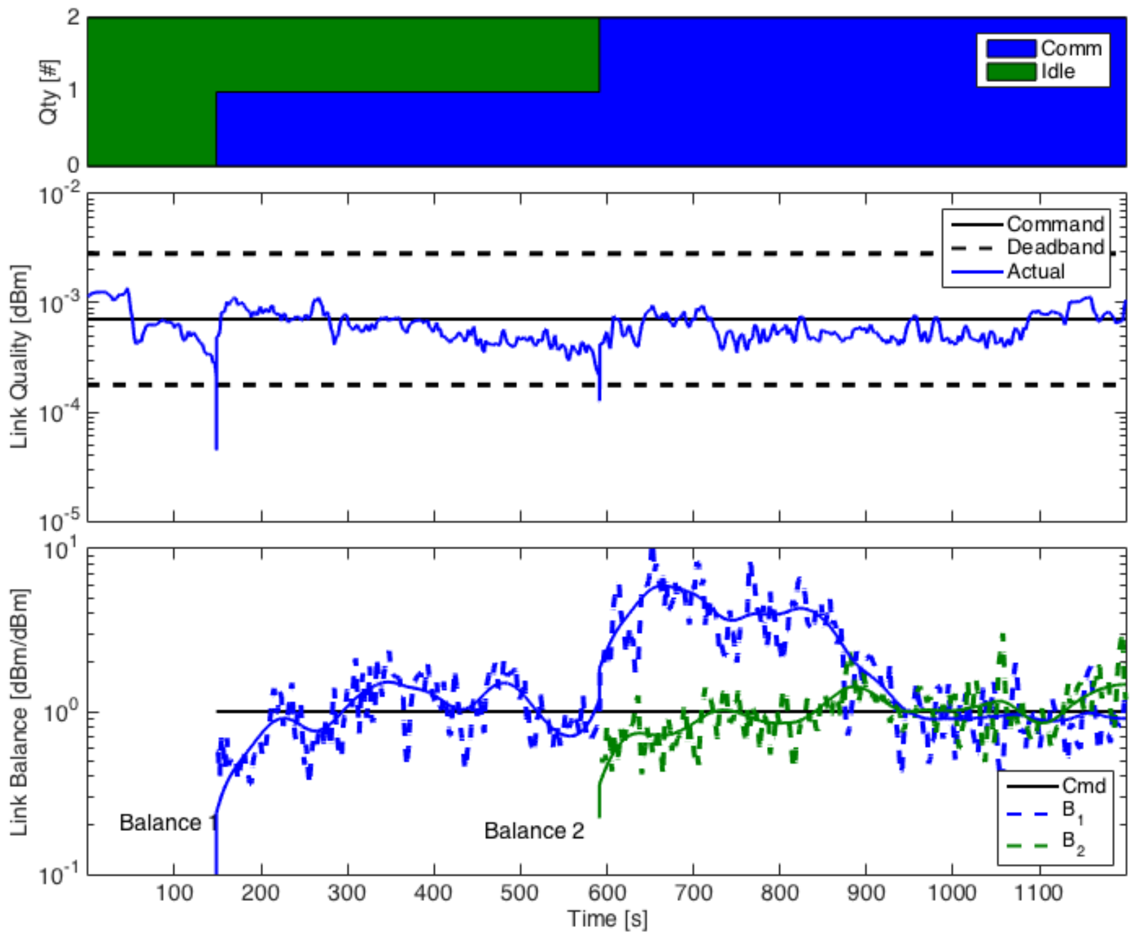


Figure 23: Time history of key system states for the mobile endpoint experiment. This experiment demonstrates task-level control of multi-robot systems in the real world. The system is able to maintain desired link characteristics by sensing the non-intuitive RF environment and adding mobile robotic relays as necessary.

3.4.2 Remote Sampling

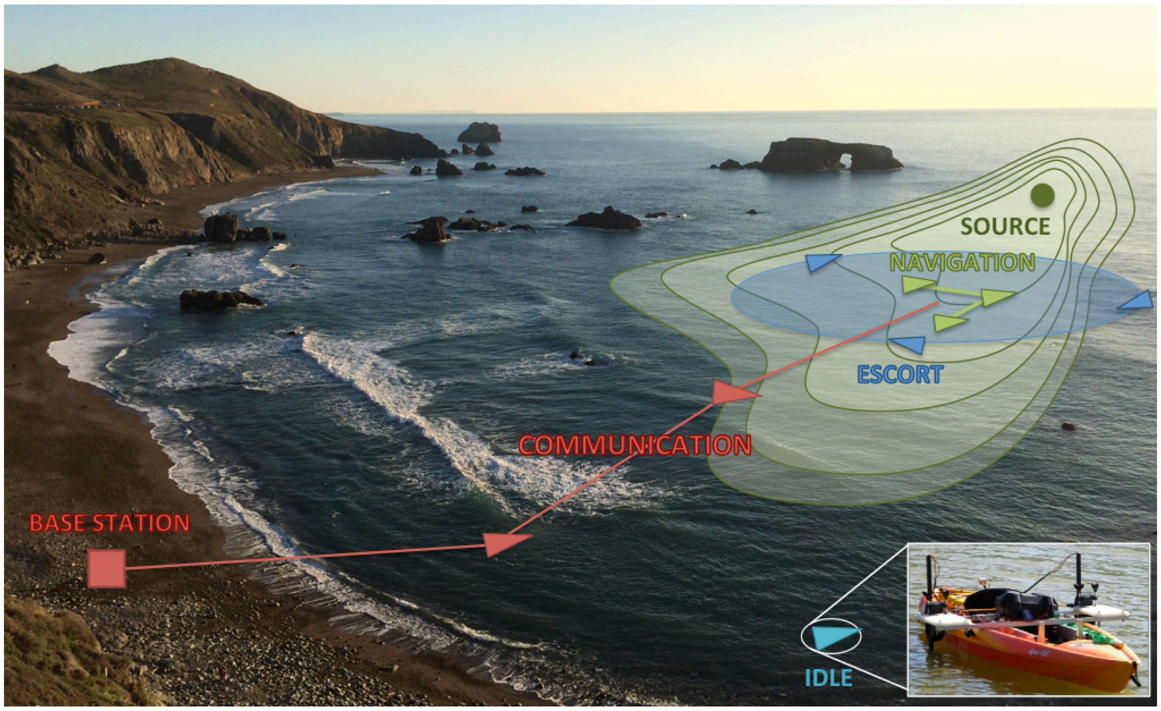
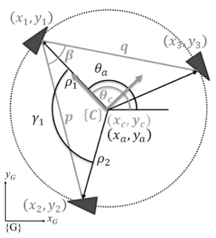
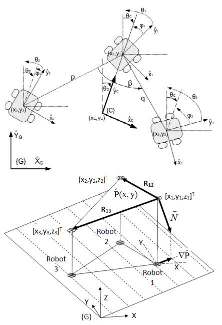
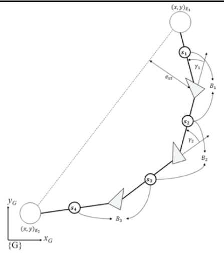


Figure 24: Coordinated tasks performed by multiple coalitions of robots in the collaborative mission example. The scientific sampling task measures a gradient and moves towards the source. The escorting task provides protection to the scientific task. The communication task relays data over long distance to the base station on the shore. Idle robots wait nearby, saving energy until allocated to one of the tasks.

To demonstrate a multi-task collaborative mission, we integrated previously explored tasks of adaptive navigation, escorting, long-range communications and formation control into the following simulated scenario. An initial coalition of three robots uses an adaptive navigation technique to travel along a particular contour line within an environmental scalar field, a function that could be used to determine, for example, the size of a contaminant field. As this coalition navigates in a manner that is unknown *a priori*, another coalition provides a protective escorting function by rotating about the initial coalition. Furthermore, a third coalition of robots establishes a mobile multi-hop communications link in order to maintain a specific quality of service for communications between the initial coalition and a base station. In the context of this federated mission, there is strong coupling between the motions of these three coalitions. The mission is depicted in Figure 24, the motion of the federations is shown in Figure 25, and key state trajectories are shown in Figure 26

Table 2: Example cluster space and task space kinematic transformation equations. These individual tasks are combined for the example collaboration mission.

Task & Cluster Diagram	Cluster Space Kinematics	Task Space Kinematics
 (x_1, y_1) (x_3, y_3) (x_2, y_2) q β θ_a ρ_1 θ_c γ_1 p {C} (x_c, y_c) (x_a, y_a) ρ_2 y_G x_G {G} Target Escorting (n=3) [**62**]	$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{x_1+x_2+x_3}{3} \\ \frac{y_1+y_2+y_3}{3} \\ \text{atan2}\left(\frac{\frac{2}{3}x_1 - \frac{1}{3}(x_2+x_3)}{\frac{2}{3}y_1 - \frac{1}{3}(y_2+y_3)}\right) \\ \theta_1+\theta_c \\ \theta_2+\theta_c \\ \theta_3+\theta_c \\ \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \\ \sqrt{(x_1-x_3)^2+(y_1-y_3)^2} \\ \text{atan2}\left(\frac{-(x_1-x_3)\sin\alpha-(y_1-y_3)\cos\alpha}{-(x_1-x_3)\cos\alpha+(y_1-y_3)\sin\alpha}\right) \end{bmatrix}$	$\begin{bmatrix} x_a \\ y_a \\ \theta_a \\ \rho_1 \\ \rho_2 \\ \gamma_1 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \frac{1}{3}\sqrt{10p^2+2pq\cos\beta+q^2-6pr\sin\alpha} \\ \frac{1}{3}\sqrt{p^2+10q^2+2pq\cos\beta-6qr\sin(\alpha-\beta)} \\ \pi-\beta \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}$ where $r = \sqrt{p^2+2pq\cos\beta+q^2}$ $\alpha = \text{atan}\left(\frac{q\sin\beta}{p + q\cos\beta}\right) - \theta_c - \text{atan}\cot\theta_c$
 Adaptive Navigation [**22**]	$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{x_1+x_2+x_3}{3} \\ \frac{y_1+y_2+y_3}{3} \\ \text{atan2}\left(\frac{\frac{2}{3}x_1 - \frac{1}{3}(x_2+x_3)}{\frac{2}{3}y_1 - \frac{1}{3}(y_2+y_3)}\right) \\ \theta_1+\theta_c \\ \theta_2+\theta_c \\ \theta_3+\theta_c \\ \sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \\ \sqrt{(x_1-x_3)^2+(y_1-y_3)^2} \\ \text{atan2}\left(\frac{-(x_1-x_3)\sin\alpha-(y_1-y_3)\cos\alpha}{-(x_1-x_3)\cos\alpha+(y_1-y_3)\sin\alpha}\right) \end{bmatrix}$	$\begin{bmatrix} \bar{z}_f \\ d_f \\ \theta_f \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ p_f \\ q_f \\ \beta_f \end{bmatrix} = \begin{bmatrix} \frac{z_1+z_2+z_3}{3} \\ \text{N/A} \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix}$ where z_i is measured in the environment. Because of this, the corresponding elements of the task Jacobian are computed as follows: $\vec{R}_{12} = [x_2-x_1 \quad y_2-y_1 \quad z_2-z_1]^T$ $\vec{R}_{13} = [x_3-x_1 \quad y_3-y_1 \quad z_3-z_1]^T$ $\vec{N} = -\vec{R}_{12}\times\vec{R}_{13}$ $J_{(1,1:2)} = [N_x \quad N_y]$
 Long-Range Communications [**65,65**]	$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \rho_i \\ \alpha_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ \sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2} \\ tan2(y_{i+1}-y_i, x_{i+1}-x_i) - \sum_{j=1}^{i-1}\alpha_j \\ \theta_i \end{bmatrix}$	$\begin{bmatrix} e_{xt} \\ B_i \\ \gamma_i \\ \psi_i \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{\left((x_{E_2}-x_{E_1})(y_{E_1}-y_c)-(x_{E_1}-x_c)(y_{E_2}-y_{E_1})\right)^2}{(x_{E_2}-x_{E_1})^2+(y_{E_2}-y_{E_1})^2}} \\ \frac{\rho_i^2}{\rho_{i+1}^2} \\ \alpha_i \\ \phi_i \end{bmatrix}$

3.4.2.1 Unified Motion Control Architecture

The unified motion control architecture consists of robot, cluster, and task space layers as depicted in Figure 13. Table 2 defines the kinematic transformations between the spaces for the selected tasks. The unified control components integrate these individual definitions. The robot allocation vector denotes the quantity of robots assigned to each task, in this case: $\vec{n} = [n_{nav}, n_{escort}, n_{com}, n_{idle}]^T$. The unified forward kinematic transformations are:

$$KIN_{cluster}(r,n) = \begin{bmatrix} KIN_{c_{nav}}(r, n_{nav}) \\ KIN_{c_{escort}}(r, n_{escort}) \\ KIN_{c_{com}}(r, n_{com}) \\ KIN_{c_{idle}}(r, n_{idle}) \end{bmatrix} \tag{93}$$

$$KIN_{task}(c,n) = \begin{bmatrix} KIN_{t_{nav}}(c, n_{sample}) \\ KIN_{t_{escort}}(c, n_{escort}) \\ KIN_{t_{com}}(c, n_{com}) \\ KIN_{t_{idle}}(c, n_{idle}) \end{bmatrix} \tag{94}$$

The unified Jacobian matrices are:

$$J_{cluster}(r,n) = \begin{bmatrix} J_{c_{nav}}(r, n_{nav}) & & & 0 \\ & J_{c_{escort}}(r, n_{escort}) & & \\ & & J_{c_{com}}(r, n_{com}) & \\ 0 & & & J_{c_{idle}}(r, n_{idle}) \end{bmatrix} \tag{95}$$

$$J_{task}(c,n) = \begin{bmatrix} J_{t_{nav}}(c, n_{nav}) & & & 0 \\ & J_{t_{escort}}(c, n_{escort}) & & \\ & & J_{t_{com}}(c, n_{com}) & \\ 0 & & & J_{t_{idle}}(c, n_{idle}) \end{bmatrix} \tag{96}$$

The unified controllers are:

$$K_{cluster}(n) = \begin{bmatrix} K_{c_{nav}}(n_{nav}) & & & 0 \\ & K_{c_{escort}}(n_{escort}) & & \\ & & K_{c_{com}}(n_{com}) & \\ 0 & & & K_{c_{idle}}(n_{idle}) \end{bmatrix} \tag{97}$$

$$K_{task}(n) = \begin{bmatrix} K_{t_{nav}}(n_{nav}) & & & 0 \\ & K_{t_{escort}}(n_{escort}) & & \\ & & K_{t_{com}}(n_{com}) & \\ 0 & & & K_{t_{idle}}(n_{idle}) \end{bmatrix} \tag{98}$$

3.4.2.2 Resource Allocation

For this example scenario, a state machine determines when and how many robots are reallocated and a cost function determines which robots are reassigned.

The sampling task is the highest priority and requires three robots at all times. The communications task has the second priority, and it is provided with the minimum number of robots required to maintain a prescribed level of link quality. The escort task has the third priority, using available robots to maintain a cluster size from 2-4 robots. The idle task has the lowest priority and is used for any robots not required by the other task.

Robots are incrementally transferred to the communication task as necessary, first from the idle task as available, then from the escort task until the minimum is reached. Table 3 below presents the logic for robot reallocation. A reallocation occurs if the link quality Z is exceeds a factor of the desired link quality $\check{Z}$ and the link quality link has stabilized as indicated by a lower threshold of the rate.

Table 3: Resource allocation logic for the example collaborative mission

Link Power Condition	Quantity Condition	Next Allocation
$Z < \frac{1}{4}\check{Z}$ & $\lvert\dot{Z}\rvert < 0.05$	$0 < n_{idle}$	$N = \begin{bmatrix} n_{nav} \\ n_{escort} \\ n_{comm} + 1 \\ n_{idle} - 1 \end{bmatrix}$
	$0 = n_{idle}$ & $2 < n_{escort}$	$N = \begin{bmatrix} n_{nav} \\ n_{escort} - 1 \\ n_{comm} + 1 \\ n_{idle} \end{bmatrix}$
$Z > 2\,\check{Z}$ & $\lvert\dot{Z}\rvert < 0.05$	$0 = n_{idle}$ & $n_{escort} < 4$	$N = \begin{bmatrix} n_{nav} \\ n_{escort} + 1 \\ n_{comm} - 1 \\ n_{idle} \end{bmatrix}$
	$0 < n_{idle}$	$N = \begin{bmatrix} n_{nav} \\ n_{escort} \\ n_{comm} - 1 \\ n_{idle} + 1 \end{bmatrix}$

The selection process chooses the robot assignment resulting in the lowest weighted sum of task space error, shown below:

$$P(r') = -k_1 Z + k_2 \sum_{i=1}^{n_{escort}} (\rho_d - \rho)_i \qquad (99)$$

where k_i are constants weighting the different terms, Z is communications link power, and $(\rho_d - \rho)$ is escort radial distance error. This approach provided acceptable results, comparable to human expectations, where robots were assigned to new roles that were closest in proximity to the equilibrium for the new role.

3.4.2.3 Task coordination

In this example scenario, task coordination guides the escort and communication tasks to supports the navigation task while the navigation and idle positioning tasks are directly specified by the mission goals.

The escort task tracks the navigation task by specifying the desired escort task centroid to the actual centroid of the navigation task. Escort task parameters of heading θ_{escort} and radius ρ_{escort_i} are specified by the operator using the mission state vector. The escort spacing γ_{escort} is specified to be evenly spread around the perimeter. These specifications are expressed by the task coordination function below for a 3-robot escort coalition, where the left hand side is the task commands and the right had side is the function Q of mission commands and actual task states:

$$\begin{bmatrix} \check{x}_{c_{escort}} \\ \check{y}_{c_{escort}} \\ \check{\theta}_{escort} \\ \check{\rho}_{escort_1} \\ \check{\rho}_{escort_2} \\ \check{\gamma}_{escort} \\ \check{\phi}_{escort_{1:3}} \end{bmatrix} = \begin{bmatrix} \hat{x}_{c_{nav}} \\ \hat{y}_{c_{nav}} \\ m_{escort_{orientation}} \\ m_{escort_{radius}} \\ m_{escort_{radius}} \\ \frac{2\pi}{n_{escort}} \\ m_{escort_{heading}} \end{bmatrix} \tag{100}$$

The communication task coordination function includes defining the task kinematics, specifically considering the link strength to the end points being connected. The state of the end points must be included to define fully the task states of link balance B_i and crosstrack error e_{xt}. Those definitions include link signal strength s_i which is a function of many parameters including environmental conditions. In practice, the signal can be measured directly but here we have assumed a line of sight model. Alignment of the communication chain can be coordinated by specifying the command as a function of the end points:

$$e_{xt} = \sqrt{\frac{\left((x_{E_2}-x_{E_1})(y_{E_1}-y_c)-(x_{E_1}-x_c)(y_{E_2}-y_{E_1})\right)^2}{(x_{E_2}-x_{E_1})^2+(y_{E_2}-y_{E_1})^2}} \tag{101}$$

$$B_1 = \frac{(x_{E_1}-x_{c_1})^2+(y_{E_1}-y_{c_1})^2}{\rho_1^2} \tag{102}$$

$$B_3 = \frac{\rho_2^2}{(x_{E_2}-x_c+\rho_2\cos(\alpha_1+\alpha_2)+\rho_1\cos\alpha_1)^2+(y_{E_2}-y_c+\rho_2\sin(\alpha_1+\alpha_2)+\rho_1\sin\alpha_1)^2} \tag{103}$$

$$\check{\gamma}_1 = \operatorname{atan2}\left(\hat{y}_{E_2}-\hat{y}_{E_1}, \hat{x}_{E_2}-\hat{x}_{E_1}\right) \tag{104}$$

3.4.2.4 Simulation Discussion

At the beginning of the simulation, the resource allocation vector is n=[3,4,0,2]. The field value measured by the adaptive sampling task is below the desired value, so it moves up the gradient towards the source and begins moving along the contour line. The escort task tracks the sampling task, matching its own centroid state to the centroid state of the adaptive sampling cluster. Simultaneously, the escort task expands its radius and rotates to patrol at the desired perimeter. Note that the radial escort distance has steady state error due to centripetal acceleration from the state coupling of the cluster control (for more information, specifically on model-based nonlinear compensation schemes, see [**67**]).

At time t=[1] (the initial condition), the communications link quality between the sampling task and the base station is below the desired value which triggers resource reassignment. The allocation vector now changes to n=[3,4,1,1] because the idle task is lowest priority so one robot is moved from idle to communications. The selection algorithm evaluates every resource assignment possibility, selecting the candidate assignment with the lowest weighted error. In this case, the lowest error configuration uses the nearer idle robot (on the right) for communication rather than the further idle robot (on the left).

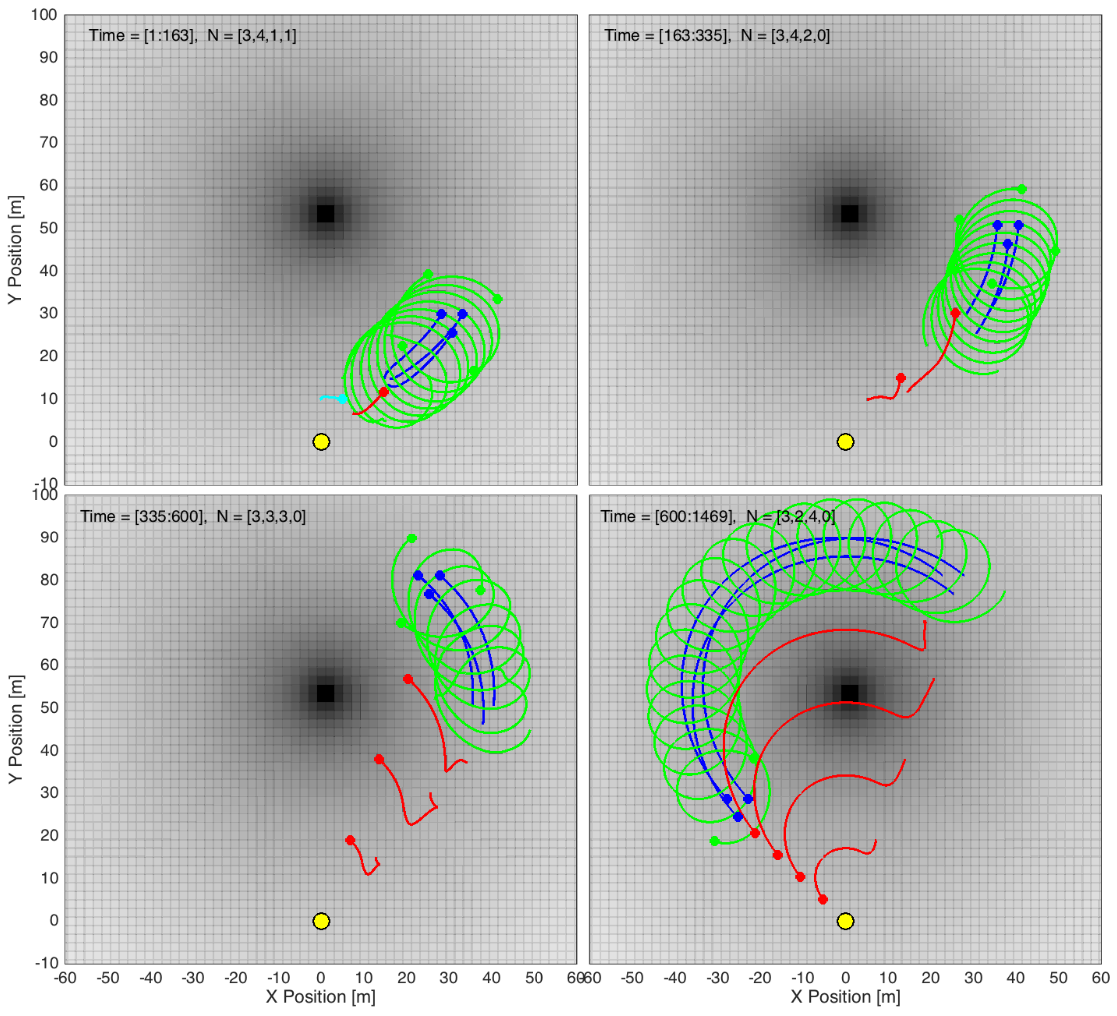


Figure 25: The overhead view of robot positions in specified time windows for the multi-task collaborative mission example. The adaptive sampling task (blue) traverses a contour of a field (gray). The escort task (green) patrols a perimeter around the sampling task. The communication task (red) relays data from the adaptive sampling task to the base station (black). Robots are moved to the communication task to maintain the data link as the sampling task moves away from the base station.

At time t=[163], the link quality falls outside the deadband. The allocation vector now changes to n=[3,4,2,0] because the idle task still held one robot that could be used for communication without impacting the other tasks. The selection algorithm keeps the existing robots assigned to the escort task and deploys the remaining idle robot to the communication task nearest the base station.

At time t=[335], the link quality falls outside the deadband. The allocation vector now changes to n=[3,3,3,0] because the communication task has been prioritized over the escort task. The selection algorithm assigns the robot from the escort task that is nearest the communications relay chain because this yields the lowest error. As a point of comparison, this is a better choice than reassigning a robot far away from the communications chain, which results in higher initial task error and requires the robot to move further to join the task.

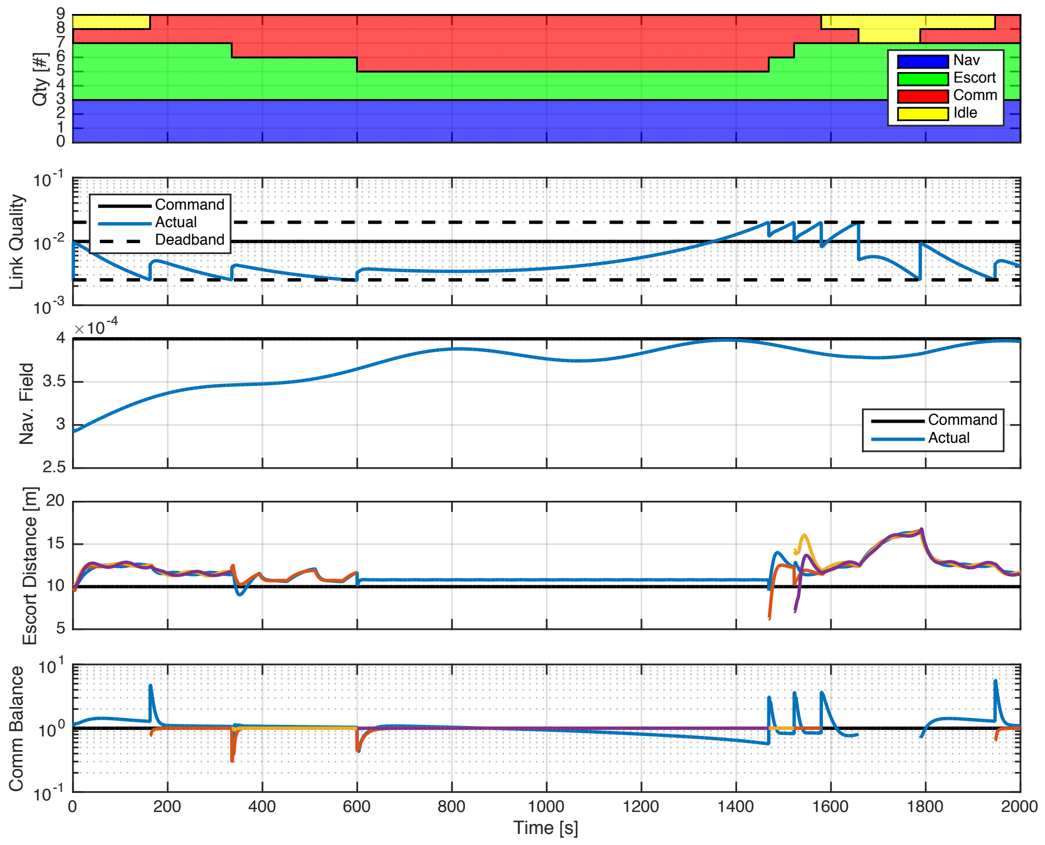


Figure 26: Time history of select states for the multi-task collaborative mission example. The top chart depicts the allocation of robots between the different tasks. The second chart presents communication link quality in comparison to commanded value and the deadband that dictates if the robots are to be reallocated. The third chart presents the measured value of the navigation field, corresponding to the gray shading in Figure 25. The fourth chart presents the radius of the perimeter provided by the escort task, showing transients at reallocation events. The final chart presents the ratio of the communication relay links, also showing the transient at reallocation events.

At time t=[581], the link quality falls outside the deadband. In this case, the additional condition of the link quality rate has not decreased sufficiently indicating the switching transient has not settled. The switching transient finally settles and the robots are reallocated at t=[600]. The allocation vector now changes to n=[3,2,4,0] because the communication task has been prioritized over the escort task. The selection algorithm assigns the robot from the escort task that is nearest the communications relay chain because this yields the lowest error. As a point of comparison, this is a better choice than reassigning a robot far away from the communications chain which results in higher initial task error and requires the robot to move further to join the task.

At time t=[1469], the link quality now rises outside the deadband. The allocation vector now changes back to n=[3,3,3,0] because the communication task no longer needs the additional robot to maintain the desired link quality. The selection algorithm assigns the robot that was at the head of the communication relay to join the escort task, squeezing into position between the nearest escorting robots. This event repeats itself at time t=[1522], where the allocation vector changes to n=[3,4,2,0].

At time t=[1579], the link quality again rises outside the deadband. The allocation vector now changes to n=[3,4,1,1]. The communication task no longer needs the additional robot and the escort cluster has sufficient resources, so the unnecessary robot is shifted into the idle task. This idle robot returns to its starting location near the base station to wait in reserve. This event repeats itself at time t=[1658], where the allocation vector changes to n=[3,4,0,2] and the whole cycle begins again.

3.5 Chapter Summary

In summary, this chapter presents an integrated motion control architecture for collaborative tasks as part of a larger mission. This builds on the architecture from Chapter 2. Multiple tasks are represented with concatenated state vectors and block-diagonal matrices. Resource allocation algorithms assign quantities of robots to specific roles within tasks, redistributing resources as necessary for the mission. Switching robot assignments will be stable so long as the commanded robot velocity is less than or equal to the actual robot velocity. Task coordination functions explicitly define relationships between tasks, specifically task command set points or task states, resulting in coupled task dynamics.

4 Conclusions & Future Work

4.1 Conclusions

Our goal was to develop and verify a unified control architecture for collaborative missions comprised of multiple, multi-robot tasks. Part of this goal included a methodology for designing new tasks, including standard control metrics for performance and stability. The integrated architecture was to be verified in simulation and experiment by integrating a diverse set of tasks into a collaborative mission.

This effort has accomplished the initial research goals, The architecture achieves mission-level control of multiple tasks working in a collaborative manner through resource sharing and coordinated tasks. The approach is formal, with rigorous analysis to provide design guidance and performance predictions. Experiments and simulations demonstrated the architecture for individual tasks and integrated missions.

Individual task-level control provides benefits to the operator and the engineer. Commands are specified naturally and the system responds in an intuitive manner. Issues are managed in the appropriate space, allowing control abstraction at higher levels. Stability and performance are influenced by state definitions (exhibited as Jacobians) as well as control parameters.

Collaborative control provides additional dimensions to the solution space. Coordination at the task level provides mission agility but couples the coalition dynamics which can impact performance. Stable resource allocation is achievable by strategic or gradual transitions between configurations to minimize errors.

For the field of robotics, this research provides a framework for control and analysis of multi-robot system motion for large-scale, highly coupled missions. As systems grow in scale and complexity, the dynamic interaction of subsystems must be considered. Design tools and analysis procedures were created for new tasks and missions and the architecture allows different control and collaboration algorithms. A formal design process provides analytic rigor to truly *engineer* a robotic system instead of ad hoc iteration.

4.2 Future Work

Although it is beyond the scope of this work, this research could continue by evaluating practical strategies for task control and exploring new capabilities for collaboration.

The rigorous nature of the control analysis could be improved for practical purposes. Approximations to the stability conditions could make controller design more tractable, specifically dealing with the pose-dependent Jacobians. For environmentally dependent states (ex: communications signal strength) that are directly measurable, the Jacobians could be estimated in real time to operate in unknown environments without needing to assume a model. Other performance metrics could be explored to determine analytically the benefits and limitations of intermediate space definitions.

Task coordination could consider more complex or dynamic relationships. Tasking a single robot with multiple tasks may over define the system but a best fit may be tolerable for limited resources. Feedback between mutually dependent tasks should be analyzed to identify stability limits. Dynamically retargeting tasks seems advantageous but may have switched stability considerations like with resource allocation. Automatic identification of new coordination schemes could improve resource efficiency and allows the system to define its own needs without designer specification.

Resource allocation should incorporate advanced assignment and switching strategies. This could include any of the suggested methods for increasing stability, like preparing for reconfigurations by gradually transitioning robots between tasks. Perhaps there is intersection with shared resources as suggested for task coordination.

Machine learning techniques should be explored as they relate to multi-robotic control. Certain techniques may be able to optimize control space definitions based on error projections between spaces, such as suggested in [**68**]. Reinforcement learning may find new and better control policies and task coordination functions for improved task and mission performance.

Bibliography

[1] International Federation of Robotics. (2015) Industrial Robot Statistics. [Online]. http://www.ifr.org/industrial-robots/statistics/

[2] Business Insider. (2015, May) The Robotics Market Report. [Online]. http://www.businessinsider.com/growth-statistics-for-robots-market-2015-2

[3] John Greenough. (2015, April) Business Insider. [Online]. http://www.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10

[4] Graham Winfrey. (2014, September) Inc. [Online]. http://www.inc.com/graham-winfrey/how-collaborative-robots-are-changing-small-business-productivity.html

[5] Tanya M. Anandan. (2015, January) Robotics Online. [Online]. http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotics-2015-and-Beyond-Collaboration-Connectivity-Convergence/content_id/5188

[6] Frank Tobe. (2016, March) Collaborative robots are broadening their marketplaces. [Online]. http://www.therobotreport.com/news/collaborative-robots-are-broadening-their-market-spheres

[7] National Science Foundation. (2016) National Science Foundation. [Online]. http://www.nsf.gov/pubs/2016/nsf16517/nsf16517.pdf

[8] CITRIS. (2015) Center for Information Technology Research in the Interest of Society (CITRIS). [Online]. http://citris-uc.org/initiatives/robotics-2/

[9] Gregory Polek. (2014, July) AIN Online. [Online]. http://www.ainonline.com/aviation-news/air-transport/2014-07-11/supplier-choices-and-production-processes-boeing-carefully-manages-risks-777

[10] Adam Mann. (2012, Nov.) Wired. [Online]. http://www.wired.com/2012/11/telerobotic-exploration/

[11] Defense Advanced Research Agency (DARPA). DARPA. [Online]. http://www.darpa.mil/program/collaborative-operations-in-denied-environment

[12] Joseph Jones. (2013, October) Robohub. [Online]. http://robohub.org/harvey-a-working-robot-for-container-crops/

[13] Lillian Sando. (2014, October) Technologist Online. [Online]. http://www.technologist.eu/internet-of-underwater-things-the-next-big-wave/

[14] Dylan A. Shell and Maja J. Mataric, "On foraging strategies for large-scale multi-robot systems," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, Beijing, China, 2006.

[15] Jie Zhao, Xiangguo Su, and Jihong Yan, "A novel strategy for distributed multi-robot coordination in area exploration," in *Measuring Technology and Mechatronics Automation, International Conference on*, Zhangjiajie, China, 2009.

[16] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun, "Collaborative

Multi-Robot Exploration," in *IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, USA, 2000.

[17] Shuai Li, Ruofan Kong, and Yi Guo, "Cooperative Distributed Source Seeking by Multiple Robots: Algorithms and Experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1810-1820, December 2014.

[18] Jack Elston and Eric W. Frew, "Hierarchical Distributed Control for Search and Tracking by Heterogeneous Aerial Robot Networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, USA, 2008.

[19] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas, "Cooperative Air and Ground Surveillance," *IEEE Robotics & Automation Magazine*, pp. 16-26, September 2006.

[20] Ralf Bachmayer and Naomi Ehrich Leonard, "Vehicle Networks for Gradient Descent in a Sampled Environment," in *Decision and Control, IEEE Conference on*, Las Vegas, Nevada, USA, 2002.

[21] Petter Ogren, Edward Fiorelli, and Naomi Ehrich Leonard, "Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292-1302, August 2004.

[22] Thomas Adamek, Christopher A. Kitts, and Ignacio Mas, "Gradient-Based Cluster Space Navigation for Autonomous Surface Vessels," *Transactions on Mechatronics, submitted*, 2014.

[23] Jonathan Fink, Alejandro Ribeiro, and Vijay Kumar, "Robust Control for Mobility and Wireless Communications in Cyber-Physical Systems With Applications to Robot Teams," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 164-178, January 2012.

[24] Dae-Keun Yoon, Jong-Tae Seo, Eui-Jung Jung, and Byung-Ju Yi, "Automatic Lighting Systems Using Multiple Robotic Lamps," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 963-974, June 2014.

[25] Vaibhav Kumar Mehta and Filippo Arrichiello. (2013, December) Connectivity maintenaince by robotic Mobile Ad-hoc NETwork. [Online]. arXiv:1312.2526

[26] J Escareno et al., "Task-based Control of a Multirotor Minature Aerial Vehicle Having an Onboard Manipulator," in *Unmanned Aircraft Systems, International Conference on*, Orlando, FL, USA, 2014.

[27] Sebastian Earhart and Sandra Hirche, "Adaptive Force/Velocity Control for Multi-Robot Cooperative Manipulation under Uncertain Kinematic Parameters," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, Tokyo, Japan, 2013.

[28] Sebastian Earhart, Dominik Sieber, and Sandra Hirche, "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, Tokyo, Japan, 2013.

[29] Dominik Sieber, Frederik Deroo, and Sandra Hirche, "Formation-based approach for multi-robot cooperative manipulation based on optimal control design," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, Tokyo, Japan, 2013.

[30] Ross A. Knepper, Todd Layton, John Romanishin, and Daniel Rus, "IkeaBot: An Autonomous Multi-Robot Coordinated Furniture Assembly System," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[31] Yu Zhang and Lynne E. Parker, "IQ-ASyMTRe - Forming Executable Coalitions for Tightly Coupled Multirobot Tasks," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 400-416, April 2013.

[32] Lynne E Parker and Fang Tang, "Building Multirobot Coalitions Through Automated Task Solution Synthesis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1289-1305, July 2006.

[33] Lovekesh Vig and Julie A. Adams, "Multi-Robot Coalition Formation," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 637-649, August 2006.

[34] Pedro M Shiroma and Mario F. M Campos, "A Task Allocation Protocol Based on Constraint Functions," in *Congresso Brasileiro de Automática* , 2010, pp. 4998-5005.

[35] Brian P Gerkey and Maja J Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939-954, September 2004.

[36] Brian P Gerkey and Maja J Mataric´, "A Framework for Studying Multi-Robot Task Allocation," in *Proceedings from Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II,* Washington DC, USA, 2003, pp. 15-26.

[37] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495-1512, 2013.

[38] Daniel A. DeLaurentis and William A. Crossley, "A Taxonomy-based Perspective for Systems of Systems Design Methods," in *Systems, Man and Cybernetics, IEEE International Conference on*, 2005, pp. 86-91.

[39] Mark W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267-284, Februrary 1999.

[40] A. DeLaurentis Daniel, "Understanding Transportation as System-of-Systems Design Problem," in *AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, 2005.

[41] Ali Mostafavi, Dulcy M. Abraham, Daniel DeLaurentis, and Joseph Sinfield, "Exploring the Dimensions of Systems of Innovation Analysis: A System of Systems Framework," *IEEE Systems Journal*, vol. 5, no. 2, pp. 256-265, June 2011.

[42] Nicos Karcanias and Ali G. Hessami, "System of Systems and Emergence," in *Emerging Trends in Engineering & Technology, International Conference on*, Port Louis, Mauritius, 2011.

[43] Department of Defense, USA, *Systems Engineering Guide for Systems of Systems*. Washington, DC, USA, 2008.

[44] Scott A. Selberg and Mark A. Austin, "Toward an Evolutionary System of Systems Architecture," in *INCOSE International Symposium*, Utrecht, the Netherlands, 2008.

[45] Stavros G. Vougioukas, "A distributed control framework for motion coordination of teams of autonomous agricultural vehicles," *Biosystems Engineering*, vol. 113, no. 3, pp. 284-297, November 2012.

[46] Nathan Schurr et al., "The Future of Disaster Response: Humans Working with Multiagent Teams using DEFACTO," in *AAAI Spring Symposium: AI Technologies for Homeland Security*, 2005, pp. 9-16.

[47] Yohan Lee, Jeremy S. Fried, Heidi J. Albers, and Robert G. Haight, "Deploying initial attack resources for wildfire suppression: spatial coordination, budget constraints, and capacity constraints," *Canadian Journal of Forest Research*, vol. 43, no. 1, pp. 56-65, 2012.

[48] Ali Hessami, "A Framework for Characterising Complex Systems and System of Systems," in *Systems, Man, and Cybernetics, IEEE International Conference on*, Manchester, England, 2013, pp. 1702-1708.

[49] Kemp H. Kernstine, "Inadequacies of Traditional Exploration Methods in Systems-of-Systems Simulations," *IEEE Systems Journal*, vol. 7, no. 4, pp. 528-536, December 2013.

[50] Nicos Karcanias and Ali G. Hessami, "Complexity and the Notion of Systems of Systems: Part (I): General Systems and Complexity," in *World Automation Congress*, Kobe, Japan, 2010.

[51] Jon Holt et al., "Model-based requirements engineering for systems of systems," in *System of Systems Engineering, International Conference on*, Genoa, Italy, 2012.

[52] Denise Jackson, Gregory Sedrick, and Karima Tayeb, "Algorithmic Development of the Effectiveness Prediction for Systems of Systems," in *Southeastern Symposium on System Theory*, Tullahoma, TN, USA, 2009.

[53] Hamid R. Darabi and Mo Mansouri, "The Role of Competition and Collaboration in Influencing the Level of Autonomy and Belonging in System of Systems," *IEEE Systems Journal*, vol. 7, no. 4, pp. 520-527, Dececmber 2013.

[54] Pablo Garcia Ansola, Andres Garcia Higuera, F. Javier Otamendi, and Javier de las Morenas, "Agent-Based Distributed Control for Improving Complex Resource Scheduling: Application to Airport Ground Handling Operations," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1145-1157, December 20014.

[55] George Rzevsky and Petr Skobelev, *Managing Complexity*. Boston, MA, USA: WIT Press, 2014.

[56] W. Ross Ashby, "Principles of the self-organizing system," in *Facets fo Systems Science*, George J. Klir, Ed. New York, NY, USA: Springer US, 1991, pp. 521-536.

[57] E. White Bruce, "A Complex Adaptive Sytems Engineering (CASE)," in *IEEE International Systems Conference*, Vancouver, Canada, 2009.

[58] Rodney A. Brooks, "A Robust Layered Conrol System for Mobile Robots," Massachusetts Institute of Technology, Memo 1985.

[59] O. Khatib et al., "Coordination and Decentralized Cooperation of Multiple Mobile Manipulators," *Journal of Robotic Systems*, vol. 13, no. 11, pp. 755-764, 1996.

[60] Maja J. Matarić, "Reinforcement Learning in the Multi-Robot Domain," *Autonomous Robots*, vol. 4, no. 1, pp. 73-83, March 1997.

[61] Christopher A. Kitts and Ignacio Mas, "Cluster Space Specification and Control of Mobile Multirobot Systems," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 207-218, 2009.

[62] Ignacio Mas, Steven Li, Jose Acain, and Christopher A. Kitts, "Entrapment/Escorting and Patrolling Missions in Multi-Robot Cluster Space Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, 2009.

[63] Paul Mahacek, Ignacio Mas, Ogi Petrovic, Jose Acain, and Christopher Kitts, "Cluster space control of autonomous surface vessels," *Marine Technology Society Journal*, pp. 13-20, March 2009.

[64] Ignacio Mas, Christopher A. Kitts, and Robert Lee, "Model-Based Nonlinear Cluster Space Control of Mobile Robot Formations," in *Multi-Robot Systems, Trends and Development*, Toshiyuki Yasuda, Ed.: InTech, 2011, pp. 53-71.

[65] John T. Shepard and Christopher A. Kitts, "Task Oriented Multi-Robot Cluster Control for Communications Link Management (submitted)," *IEEE Systems Journal*, Submitted 2016.

[66] Daniel Liberzon, *Switching in Systems and Control*. Boston, MA, USA: Birkhauser, 2003.

[67] Ignacio Mas and Christopher A Kitts, "Dynamic Control of Mobile Multirobot Systems: The Cluster Space Formulation," *IEEE Access*, vol. 2, no. 2014, pp. 558-570, May 2014.

[68] J. Kober and J. Peters, "Adaptation, Learning, and Optimization," in *Reinforcement learning in robotics: A survey*, M. Wiering and M. Otterlo, Eds.: Springer Berlin Heidelberg, 2012, vol. 12.

[69] Ignacio Mas and Christopher Kitts, "Obstacle Avoidance Policies for Cluster Space Control of Nonholonomic Multirobot Systems," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 6, pp. 1068-1079, Dec 2012.

[70] Digi International, Inc , "XBee®/XBee-PRO® ZB RF Modules ," Reference Manual 2012.

[71] Robert Zlot and Anthony Stentz, "Multirobot control using task abstraction in a market framework," in *Collaborative Technology Alliances Conference*, 2003.

[72] Dae-Keun Yoon, Jong-Tae Seo, and Byung-Ju Yi, "Automatic Lighting System Using Multiple Robotic Lamps," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 963-974, June 2014.

[73] Michael C Yip and David B Camarillo, "Model-less feedback control of continuum manipulators in constrained environments," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 880-889, August 2014.

[74] Xie Wenlong, Su Jianbo, and Lin Zongli, "New coordination scheme for multi-robot systems based on state space models," *Systems Engineering and Electronics, Journal of*, vol. 19, no. 4, pp. 722-734, August 2008.

[75] Giuseppe Tortora, Paolo Dario, and Arianna Menciassi, "Array of Robots Augmenting the Kinematics of Endocavitary Surgery," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, p. 1821, Dec 2014.

[76] Bryan J. Thibodeau, Andrew H. Fagg, and Brian N. Levine, "Signal Strength Coordination for Cooperative Mapping," Amherst, USA, 2005.

[77] John D. Sweeney, Roderic Grupen, and Prashant Shenoy, "Active QoS Flow Maintenance in Controlled Mobile Networks," in *the Fourth International Symposium on Robotics and Automation (ISRA), Proceedings of*, Queretaro, Mexico, 2005.

[78] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohme, "Coalition

structure generation with worst case guarantees," *Artificial Intelligence*, no. 111, pp. 209-238, 1999.

[79] Narek Pezeshkian, Hoa G. Nguyen, and Aaron Burmeister, "Unmanned Ground Vehicle Radio Relay Deployment System for Non-Line-Of-Sight Operations," San Diego, USA, 2007.

[80] Narek Pezeshkian, Joseph D. Neff, and Abraham Hart, "Link Quality Estimator for a Mobile Robot," in *Int. Conf. on Informatics in Control, Automation and Robotics*, Rome, Italy, 2012.

[81] Garret Okamoto, Chi-Wei Chen, and Christopher Kitts, "Beamforming Performance for a Reconfigurable Sparse Array Smart Antenna System via Multiple Mobile Robotic Systems," in *Proceedings of the SPIE - The International Society for Optical Engineers*, 2010.

[82] Miles O'Brien and Kate Tobin. (2013, December) National Science Foundation. [Online]. http://www.nsf.gov/news/special_reports/science_nation/collaborativerobots.jsp

[83] James P. Minas, John W. Hearne, and John W. Handmer, "A review of operations research methods applicable to wildfire management," *International Journal of Wildland Fire*, vol. 21, no. 3, pp. 189-196, February 2012.

[84] Vaibhav Kumar Mehta and Filippo Arrichiello. (2013, Decmber) arXiv.org. [Online]. http://arxiv.org/abs/1312.2526v1

[85] Ignacio Mas and Christopher Kitts, "Multi-Robot Object Manipulation Using Cluster Space Control," in *ASME Information Storage and Processing Systems Conference*, Santa Clara, CA, 2010.

[86] I. Mas, O. Petrovic, and C. Kitts, "Cluster space specification and control of a 3-robot mobile system," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, Pasadena, CA, USA, 2008, pp. 3763-3768.

[87] Paul Mahacek, Christopher A. Kitts, and Ignacio Mas, "Dynamic Guarding of Marine Assets through Cluster Control of Automated Surface Vessel Fleets," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 65-75, 2012.

[88] Chao-Wei Lin, Mun-Hooi Khong, and Yen-Chen Liu, "Experiments on Human-in-the Loop Coordination for Multirobot Systems with Task Abstraction," *Automation Science and Engineering, IEEE Transactions on*, vol. PP, no. 99, 2015.

[89] Daniel Liberzon and A. Stephen Morse, "Basic Problems in Stability and Design of Switched Systems," *IEEE Control Systems Magazine*, vol. 19, no. 5, pp. 59-70, Oct 1999.

[90] Shuai Li, Ruofan Kong, and Yi Guo, "Cooperative Distributed Source Seeking by Multiple Robots: Algorithms and Experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1810-1820, December 2014.

[91] Ross A. Knepper, Todd Layton, John Romanishin, and Daniela Rus, "IkeaBot: An Autonomous Multi-Robot Coordinated Furniture Assembly System," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013.

[92] Farangis A Khosro, R Rehal, F Wilken, and Stavros Vogioukas, "Sensor-based Stooped Work Monitoring in Robot-aided Strawberry Harvesting," in *ASABE Annual Intl. Meeting*, Montreal, 2014.

[93] Farangis Khosro, R Rehal, and S Vougioukas, "A Low-Cost, Efficient Strawberry Yield Monitoring System," in *ASABE Annual Intl. Meeting*, New Orleans, 2015.

[94] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, February 1987.

[95] Matthew A Joordens and Mo Jamshidi, "Consensus Control for a System of Underwater Swarm Robots," *IEEE Systems Journal*, vol. 4, no. 1, pp. 65-73, March 2010.

[96] Aleksandar Jevtic, Alvaro Gutierrez, Diego Andina, and Mo Jamshidi, "Distributed Bees Algorithm for Task Allocation in Swarm of Robots," *IEEE Systems Journal*, vol. 6, no. 2, pp. 296-304, June 2012.

[97] Hyun-Ja Im, Chang-Eun Lee, Young-Jo Cho, and Kim Sunghoon, "RSSI-Based Control of Mobile Cooperative Robots for Seamless Networking," in *Control, Automation, and Systems, International Conference on*, Jeju Island, Korea, 2012.

[98] Mong-ying A. Hsieh, Vijay Kumar, and Camillo J. Taylor, "Constructing Radio Signal Strength Maps with Multiple Robots," in *Robotics & Automation, Proceedings of the 2004 IEEE International*

Conference on , New Orleans, USA, 2004.

[99] Brian P. Gerkey and Maja J. Mataric, "Sold!: Auction Methods for Multirobot Coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758-768, October 2002.

[100] Jack Elston and Eric W. Frew, "Hierarchical Distributed Control for Search and Tracking by Heterogeneous Aerial Robot Networks," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008.

[101] John J. Craig, *Introduction to Robotics: Mechanics and Control*, Third Edition, Ed.: Pearson Prentice Hall, 2005.

[102] Bernd Brüggemann, Alexander Tiderko, and Markus Stilkerieg, "Adaptive Signal Strength Prediction based on Radio Propagation Models for improving Multi-Robot Navigation Strategies," in *Robot Communication and Coordination, 2009. ROBOCOMM '09. Second International Conference on*, 2009.

[103] Nicola Bezzo, Yuan Yan, Rafael Fierro, and Yasamin Mostofi, "A Decentralized Connectivity Strategy for Mobile Robot Swarms," in *18th World Congress of the International Federation of Automatic Control*, Milan, Italy, 2011, pp. 4501-4506.

[104] Nicola Bezzo et al., "A Cooperative Heterogeneous Mobile Wireless Mechatronic System," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 20-31, February 2014.

[105] Spring Berman and Vijay Kumar, "Abstractions and Algorithms for Assembly Tasks with Large Numbers of Robots and Parts," in *Conference on Automation Science and Engineering*, Bangalore, India, 2009.

[106] K Benkic, M Malajner, P Planinsic, and Z Cucej, "Using RSSI value for distance estimation in wireless sensor networks based on ZigBee," in *Systems, Signals and Image Processing, International Conference on*, Bratislava, 2008, pp. 303-306.

[107] Tucker Balch and Ronald C. Arkin, "Behavior-Based Formation Control for Multi-Robot Teams," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926-939, Decmeber 1998.

[108] Amit Ailon and Ilan Zohar, "Control Strategies for Driving a Group of Nonholonomic Kinematic Mobile Robots in Formation Along a Time-Parameterized Path," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 2, pp. 326-336, April 2012.

[109] Michael Seamus Agnew, Patrick Dal Canto, Christopher A. Kitts, and Steven Q Li, "Cluster Space Control of Aerial Robots," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Montreal, Canada, 2010.

Appendices

A. Multi-Robot Test Bed Description

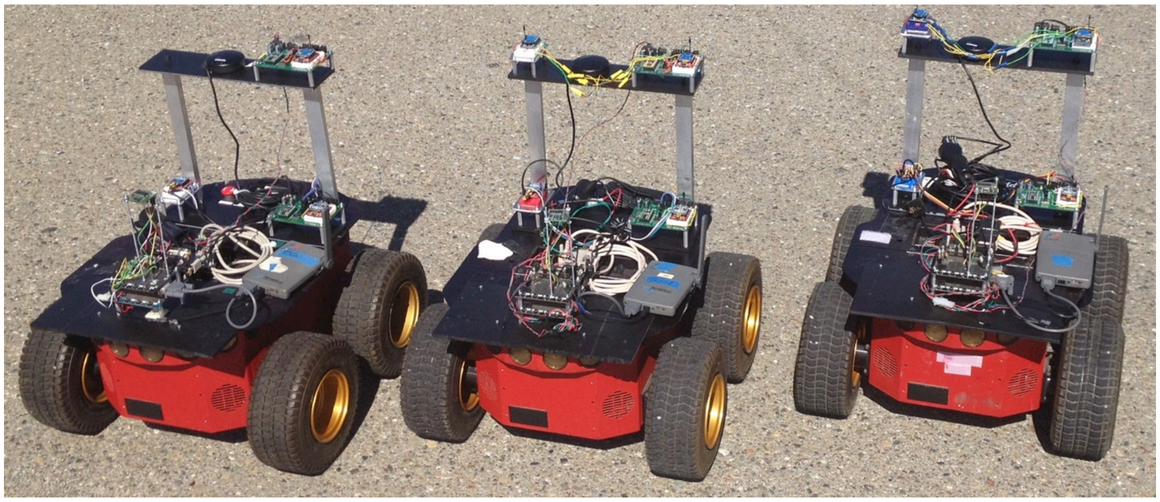


Figure 27: Multi-Robot Testbed with Communications Relay Test Bed

Experimental work used the proven SCU multi-robot infrastructure, with hardware added for this particular application. The SCU multi-robot test bed has been developed over a number of years by various students. Control computations are performed in real-time in the MathWorks Simulink environment. Internally developed software based upon DataTurbine, a real-time data streaming engine, is used to route telemetry and commands between serial COM ports and Simulink, and commands from Simulink back to COM ports. The data on the COM ports is transmitted using wireless Ricochet modems to BasicX microcontrollers onboard Adept Mobile Robot Pioneer robots. These microcontrollers send translation and rotation commands to the Pioneers and acquire GPS position, compass, and wheel speed measurements which are relayed back to the Simulink controller via the (ancient) Ricochet communication link and DataTurbine infrastructure. This test bed is optimized for development speed and as such has recognized inefficiencies. Using a reasonably powerful laptop, the system maintains a 5 Hz update rate, and has been run faster using multiple networked computers for more demanding computations.

The robot motion was characterized using sine sweeps so that most of the development could be performed in simulation, allowing more testing time for experiments rather than debugging. To anyone following (or concurrent with) me in this lab, I highly recommend beginning with simulations of your system using these (or your own) robot models as they provide repeatability and control of all parameters, which significantly aids debugging. The robot forward and rotational velocity response given a commanded velocity may be approximated as a second-order system with two zero order holds, shown here as Pade approximations:

$$G_{translation}(s) = \left(\frac{1}{\left(\frac{s}{2\pi 0.4}\right)^2 + 2(0.7)\left(\frac{s}{2\pi 0.4}\right) + 1}\right)\left(\frac{s^2+30s+300}{s^2-30s+300}\right)^2 \tag{105}$$

$$G_{rotation}(s) = \left(\frac{1}{\left(\frac{s}{2\pi 0.15}\right)^2 + 2(0.7)\left(\frac{s}{2\pi 0.15}\right) + 1}\right)\left(\frac{s^2+30s+300}{s^2-30s+300}\right)^2 \tag{106}$$

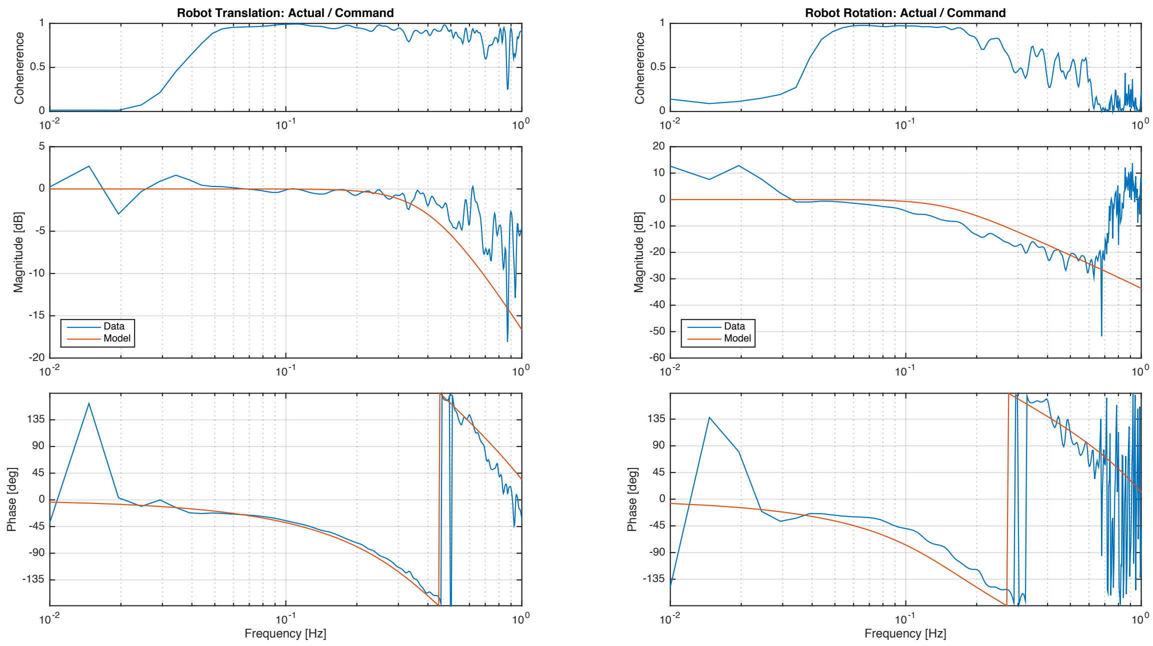


Figure 28: Sine sweep frequency response of Pioneer-AT robot

If this process is unfamiliar, one can loosely follow this code:

Create stepped sine sweep:

```
t_end   = 300; % end time [s]
t_ramp  = 1; % ramp time [s]
a_ramp  = 0; % ramp amplitude
a_sweep = [250]; % sweep amplitude

F = [0.05 2.5]; % frequency sweep start and end [Hz]

dt = ts;
Xi = [0:dt:t_end];
Y = [];
X = 0;
% assemble stepped sine sweep command
for i = 1:length(a_sweep)
    % ramp offset
    Yi = a_ramp*ones(size(Xi));
    Yi(find(Xi<t_ramp))          = Yi(find(Xi<t_ramp))-a_ramp/t_ramp*(t_ramp-Xi(find(Xi<t_ramp)));
    Yi(find(Xi>Xi(end)-t_ramp))    = Yi(find(Xi>Xi(end)-t_ramp))-a_ramp/t_ramp*Xi(find(Xi<t_ramp));

    % chirp
    Yi(find(Xi>t_ramp,1,'first'):find(Xi>t_end-t_ramp,1,'first'))                           =
Yi(find(Xi>t_ramp,1,'first'):find(Xi>t_end-t_ramp,1,'first')) ...
        +a_sweep(i)*chirp([0:dt:t_end-2*t_ramp],F(1),t_end-2*t_ramp,F(2),'logarithmic',-90);

    % concatenate
    Y = [Y,Yi];
    X = [X,Xi+X(end)];
```

```
end
X = X(2:end);

t = X.';
r_cmd       = zeros(length(t),2); % initialize
r_cmd(:,1) = Y; % forward velocity command (enabled)
% r_cmd(:,2) = Y; % rotational velocity command (disabled)

% figure(1);clf
% plot(t,r_cmd);grid on
% r_cmd = [t,r_cmd];
```

Then run the simulation, recording the actual values, and compute the transfer functions and coherence:

```
nfft = 2^10;%2^(nextpow2(length(t(idx)))-1);
window = [];
noverlap = [];

G = [];
COH = [];

[G(:,1),F] = tfestimate(rdot_cmd_L(idx,2*ri-1)*Kt,rdot_act_L(idx,2*ri-1),window,noverlap,nfft,1/dt);
[G(:,2),F] = tfestimate(rdot_cmd_L(idx,2*ri)*Kr,rdot_act_L(idx,2*ri),window,noverlap,nfft,1/dt);
[COH(:,1),F] = mscohere(rdot_cmd_L(idx,2*ri-1)*Kt,rdot_act_L(idx,2*ri-1),window,noverlap,nfft,1/dt);
[COH(:,2),F] = mscohere(rdot_cmd_L(idx,2*ri)*Kt,rdot_act_L(idx,2*ri),window,noverlap,nfft,1/dt);
```

which can be plotted against models:

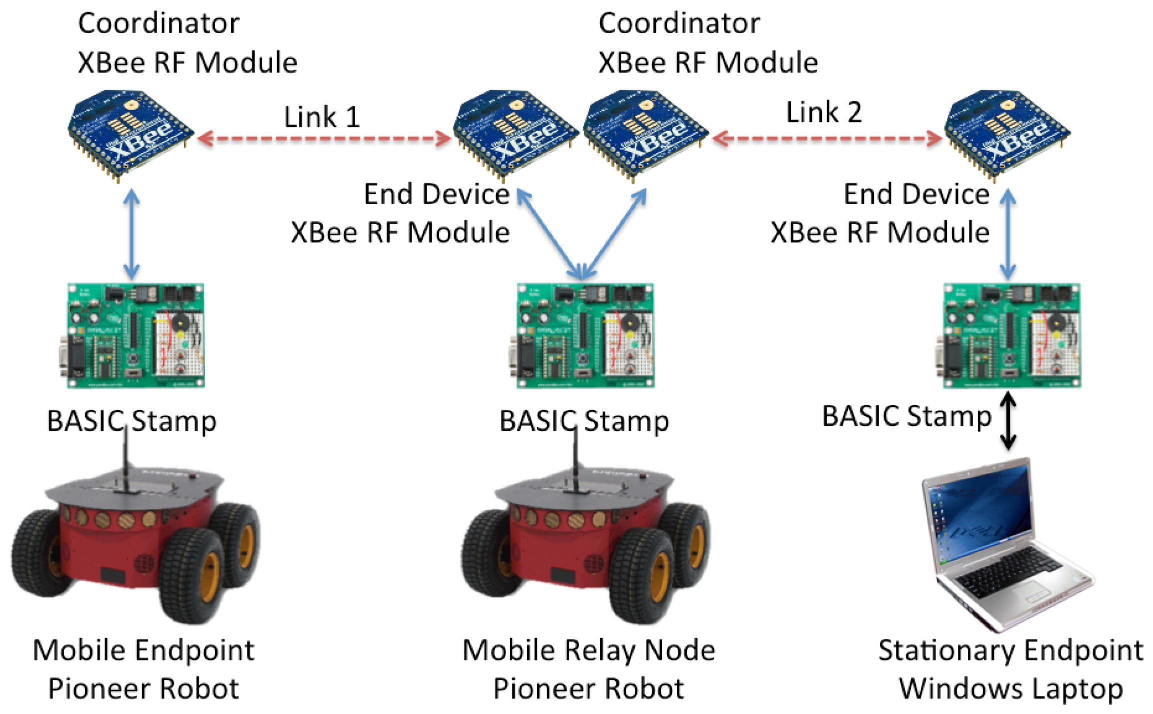
```
% plant estimate
ts = 1/5
s = tf('s');
z = tf('z',ts);
w = 2*pi*0.15;
Z = 0.7;
Gr = tf(B_pade,A_pade)^n_z/((s/w)^2 + 2*Z/w*s+1);
w = 2*pi*0.4;
Z = 0.7;
Gt = tf(B_pade,A_pade)^n_z/((s/w)^2 + 2*Z/w*s+1);

Gt = freqresp(Gt,2*pi*F);Gt = squeeze(Gt);
Gr = freqresp(Gr,2*pi*F);Gr = squeeze(Gr);

figure(16);clf;set(gcf,'WindowStyle','Docked');set(gcf,'Color','White')
subplot(5,2,1)
semilogx(F,COH);
ylabel('Cohenerence')
title('Robot Translation: Actual / Command')

grid on
% axis tight
```

```
ylim([0 1]);
xlim([1e-2 1e0]);

subplot(5,2,[3 5]);
semilogx(F,20*log10(abs([G(:,1),Gt])))
ylabel('Magnitude [dB]')
grid on
% axis tight
xlim([1e-2 1e0]);
legend('location','SW','Data','Model')

subplot(5,2,[7 9]);
semilogx(F,180/pi*(angle([G(:,1),Gt])));
ylabel('Phase [deg]')
grid on
ylim([-180 180])
set(gca,'YTick',[-180:45:180])

xlabel('Frequency [Hz]')
axis tight
xlim([1e-2 1e0]);
```

Regarding implementation, using embedded MATLAB functions within Simulink (for calculating kinematics, Jacobians, etc) is far faster than alternative block types (ex: S-Functions) because they are compiled on runtime. As a further benefit, these same MATLAB functions can be used independently (outside of Simulink) for debugging, performance analysis or even symbolic analysis.

Details on the nonholonomic heading controller may be found in [**69**].

B. Communication Relay Test Bed



The added communications relay test bed is comprised of a chain of Digi International XBee Series 2 wireless modules [**70**] mounted upon each mobile robot. The end node broadcasts a message which is relayed between robots until it reaches the base node. At each node, a BASIC Stamp microcontroller measures the link quality as a received signal strength indicator (RSSI), appends the measurement to the original message, and relays it to the next node. Two RF modules per relay node were necessary because the RSSI measurement only occurs for the last hop in the communication chain. Measurements were attempted at 1 Hz (with significant effort to overcome limitations of the BASIC Stamps), but were often inconsistent, adding a realistic challenge to the control. The data below depicts the signal strength with respect to distance.

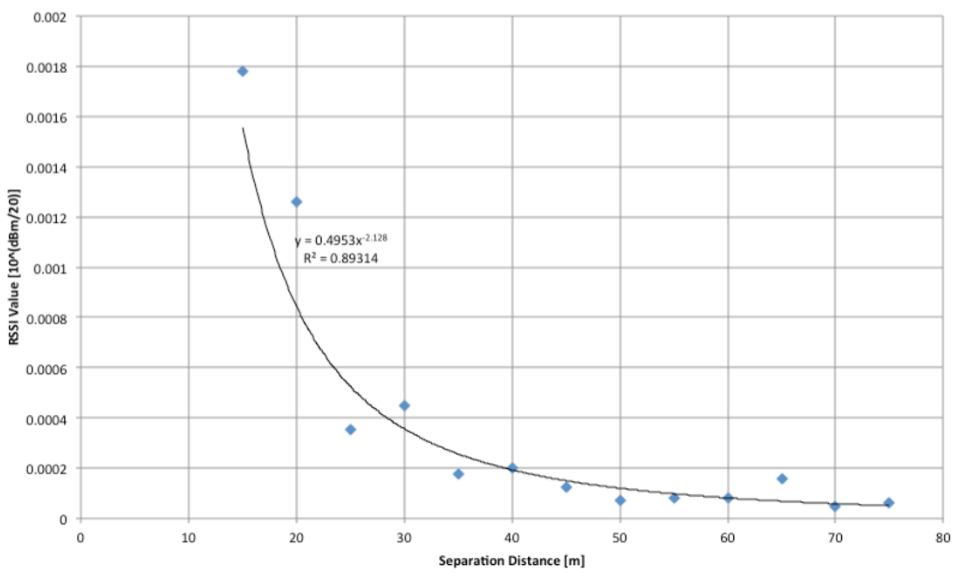


Figure 29: Received signal strength indicator (RSSI) verses separation distance for Xbee Series 2 RF Modules. This data suggests model RSSI = 0.5/distance^2 (indicated by black line). This data was collected by Adwait Bhalerao and Matthew Chin.

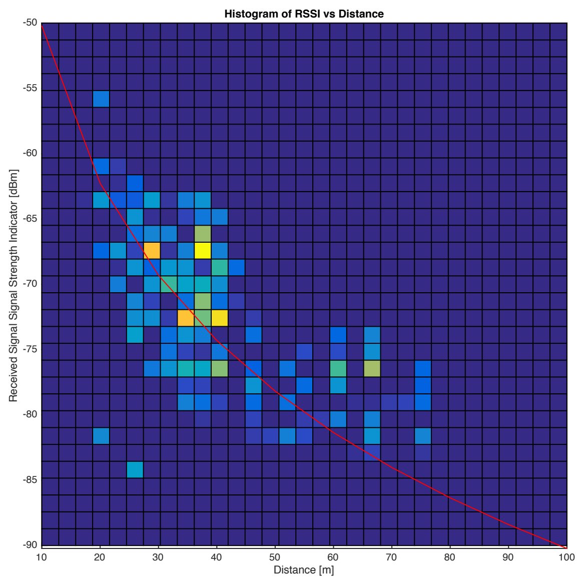


Figure 30: Composite histogram of received signal strength indicator (RSSI) verses separation distance while running experiments. This data suggests the model: RSSI = 0.31/distance^2

C. Task examples

Prior work was repeated to evaluate the proposed task design methodology. While not exhaustive, these particular examples demonstrate some degree of generality and relevance for the design method.

1. Escorting

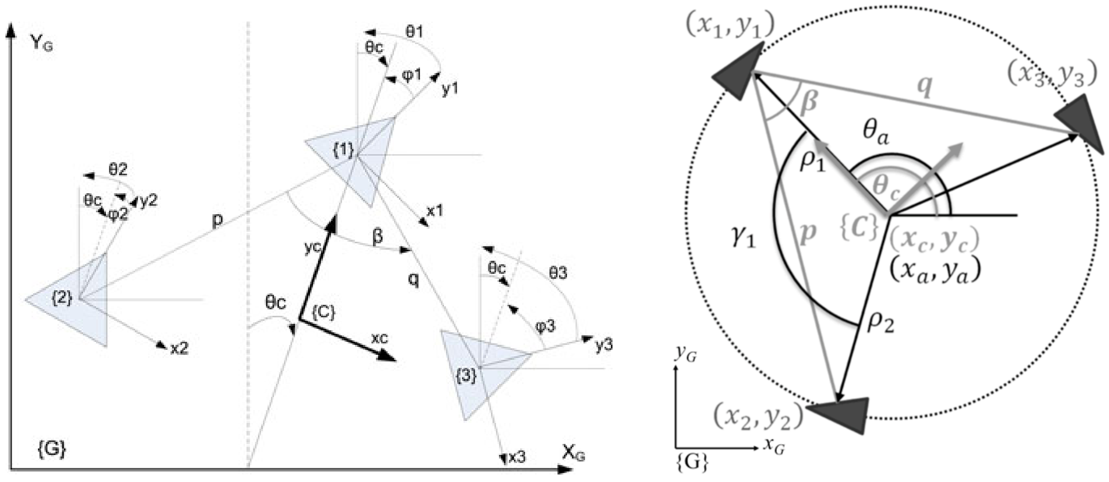


Image from [**69**]

Cluster Space Definition:

$$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix} = \begin{bmatrix} \dfrac{x_1 + x_2 + x_3}{3} \\ \dfrac{y_1 + y_2 + y_3}{3} \\ \operatorname{atan2}\left(\dfrac{\frac{2}{3}x_1 - \frac{1}{3}(x_2 + x_3)}{\frac{2}{3}y_1 - \frac{1}{3}(y_2 + y_3)}\right) \\ \theta_1 + \theta_c \\ \theta_2 + \theta_c \\ \theta_3 + \theta_c \\ \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \\ \operatorname{atan2}\left(\dfrac{-(x_1 - x_3)\sin\alpha - (y_1 - y_3)\cos\alpha}{-(x_1 - x_3)\cos\alpha + (y_1 - y_3)\sin\alpha}\right) \end{bmatrix}$$

Task Space Definition:

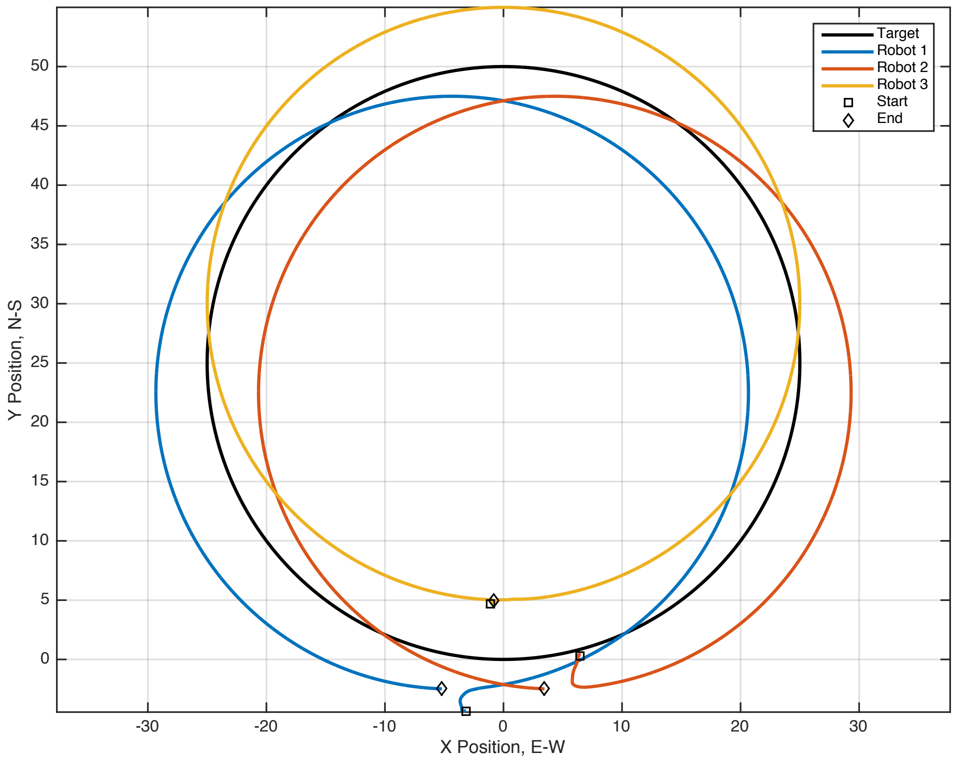
$$\begin{bmatrix} x_a \\ y_a \\ \theta_a \\ \rho_1 \\ \rho_2 \\ \gamma_1 \\ \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \frac{1}{3}\sqrt{10p^2 + 2pq\cos\beta + q^2 - 6pr\sin\alpha} \\ \frac{1}{3}\sqrt{p^2 + 10q^2 + 2pq\cos\beta - 6qr\sin(\alpha - \beta)} \\ \pi - \beta \\ \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}$$

where

$$r = \sqrt{p^2 + 2pq\cos\beta + q^2}$$

$$\alpha = \operatorname{atan}\left(\frac{q\sin\beta}{p\ +\ q\cos\beta}\right) - \theta_c - \operatorname{atan}\cot\theta_c$$

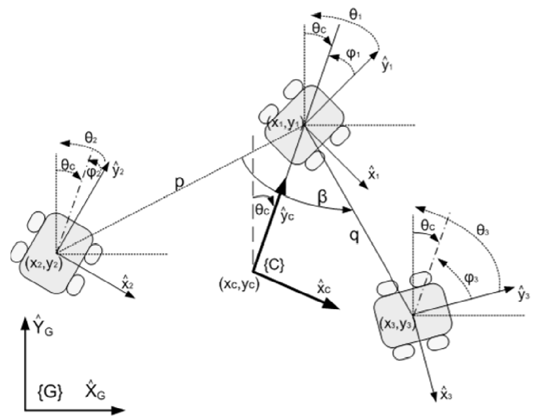
Example Results

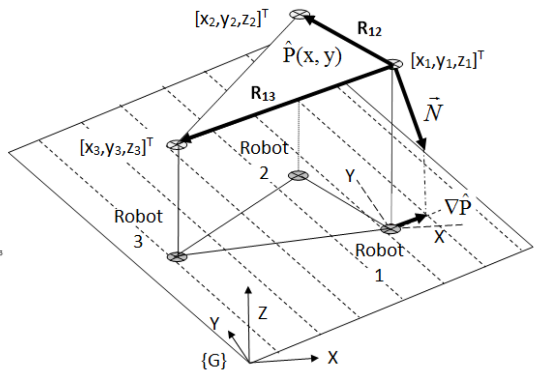


Discussion

Escorting is a simple task but it demonstrates the architecture layers. The task space states are closely related to the geometric states of the cluster. The simulation results show effective tracking of a target while maintaining orientation.

2. Navigation





Images from [**22**]

Cluster Space Definition

$$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix} = \begin{bmatrix} \dfrac{x_1 + x_2 + x_3}{3} \\ \dfrac{y_1 + y_2 + y_3}{3} \\ \operatorname{atan2}\left(\dfrac{\frac{2}{3}x_1 - \frac{1}{3}(x_2 + x_3)}{\frac{2}{3}y_1 - \frac{1}{3}(y_2 + y_3)}\right) \\ \theta_1 + \theta_c \\ \theta_2 + \theta_c \\ \theta_3 + \theta_c \\ \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\ \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \\ \operatorname{atan2}\left(\dfrac{-(x_1 - x_3)\sin\alpha - (y_1 - y_3)\cos\alpha}{-(x_1 - x_3)\cos\alpha + (y_1 - y_3)\sin\alpha}\right) \end{bmatrix}$$

Task Space Definition

$$\begin{bmatrix} \bar{z}_f \\ d_f \\ \theta_f \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ p_f \\ q_f \\ \beta_f \end{bmatrix} = \begin{bmatrix} \frac{z_1 + z_2 + z_3}{3} \\ \mathrm{N/A} \\ \theta_c \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ p \\ q \\ \beta \end{bmatrix}$$

where z_i is measured in the environment. Because of this, the corresponding elements of the task Jacobian are computed as follows:

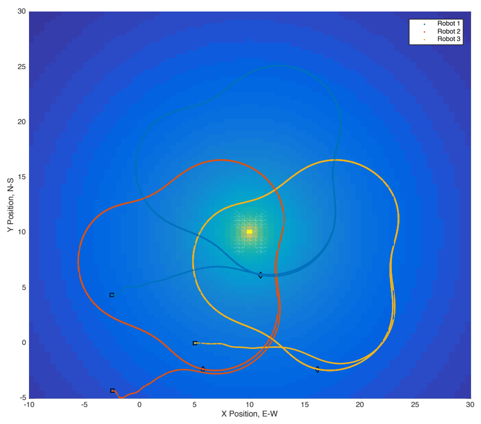
$$\vec{R}_{12} = [x_2 - x_1 \quad y_2 - y_1 \quad z_2 - z_1]^T$$

$$\vec{R}_{13} = [x_3 - x_1 \quad y_3 - y_1 \quad z_3 - z_1]^T$$

$$\vec{N} = -\vec{R}_{12} \times \vec{R}_{13}$$

$$J_{(1,1:2)} = [N_x \quad N_y\,]$$

Example Results



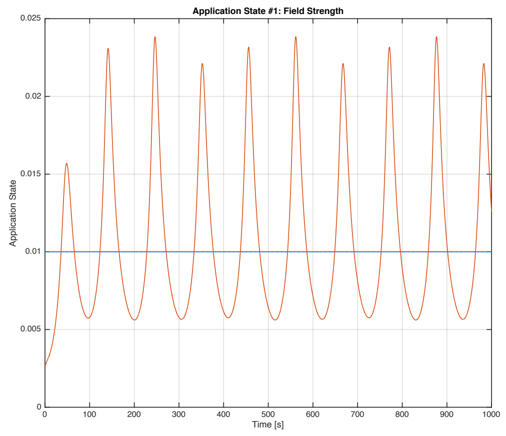


Figure 31: Robots traversing constant field contours around a uniform source.

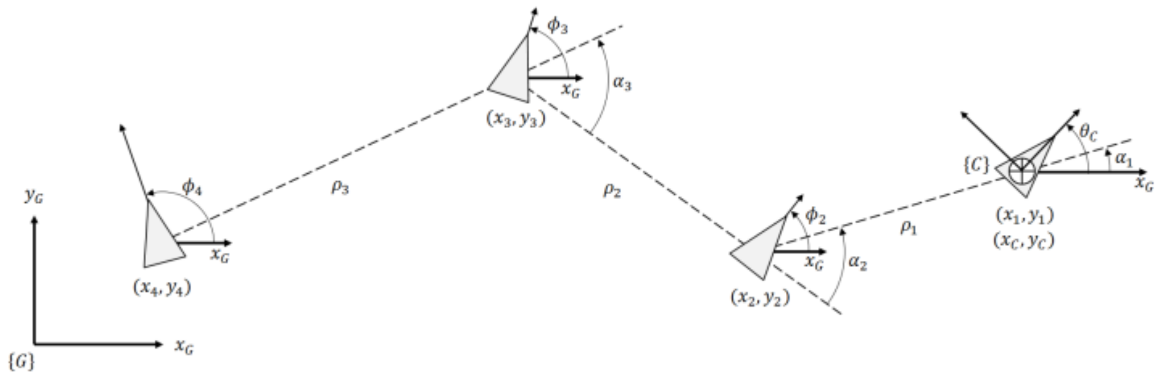
Discussion

A unique aspect of this application is its responsive nature. The system tracks environmental conditions rather than strictly following operator commands. To do so, the system must measure the environmental states and estimate gradients to orient itself within the environmental field.

As can be seen in the results plot, there is a consistent undulation to the robot tracks that was never fully understood.

3. Communications

Cluster Space Definition



Cluster frame:

$$x_c \triangleq x_1 \tag{107}$$

$$y_c \triangleq y_1 \tag{108}$$

$$\theta_c \triangleq \theta_1 \tag{109}$$

Chain length:

$$\rho_i \triangleq \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{110}$$

Chain angle:

$$\alpha_i \triangleq atan2(y_{i+1} - y_i, x_{i+1} - x_i) - \sum_{j=1}^{i-1} \alpha_j \tag{111}$$

Node orientation:

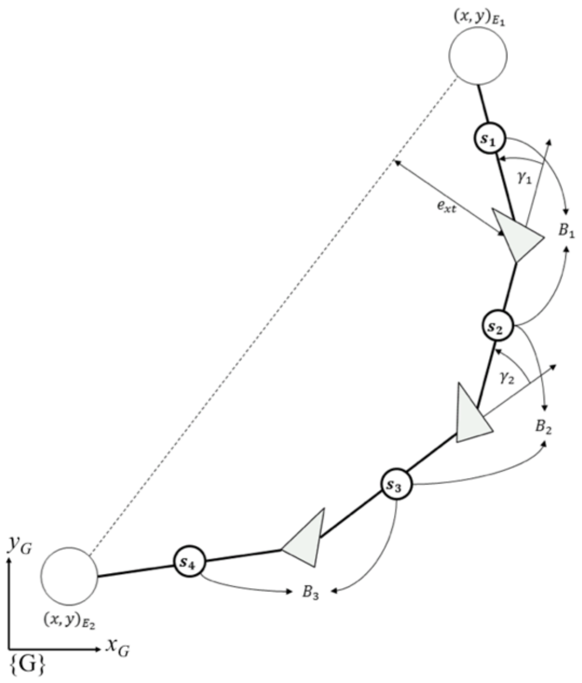
$$\phi_i \triangleq \theta_i \tag{112}$$

The cluster pose vector:

$$\begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \rho_i \\ \alpha_i \\ \phi_i \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ \sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2} \\ tan2(y_{i+1}-y_i, x_{i+1}-x_i) - \sum_{j=1}^{i-1}\alpha_j \\ \theta_i \end{bmatrix} \tag{113}$$

where $atan2(\dots,\dots)$ is the two-argument function that calculates a four-quadrant arc tangent with a range of $[\pi, -\pi]$..

Task Space Definition



Balance:

$$B_i \triangleq \frac{s_{i+1}}{s_i} = \begin{cases} \frac{\left(x_{E_1}-x_{c_1}\right)^2+\left(y_{E_1}-y_{c_1}\right)^2}{\rho_1^2} \text{ for } i=1 \\ \frac{\rho_2^2}{\left(x_{E_2}-x_c+\rho_2\cos(\alpha_1+\alpha_2)+\rho_1\cos\alpha_1\right)^2+\left(y_{E_2}-y_c+\rho_2\sin(\alpha_1+\alpha_2)+\rho_1\sin\alpha_1\right)^2} \text{ for } i=n_p-1 \\ \frac{\rho_i^2}{\rho_{i+1}^2} \text{ otherwise} \end{cases} \tag{114}$$

Crosstrack error:

$$e_{xt} = \sqrt{\frac{\left((x_{E_2}-x_{E_1})(y_{E_1}-y_c)-(x_{E_1}-x_c)(y_{E_2}-y_{E_1})\right)^2}{\left(x_{E_2}-x_{E_1}\right)^2+\left(y_{E_2}-y_{E_1}\right)^2}} \qquad (115)$$

Angle of alignment

$$\gamma_i = \alpha_i \qquad (116)$$

Orientation:

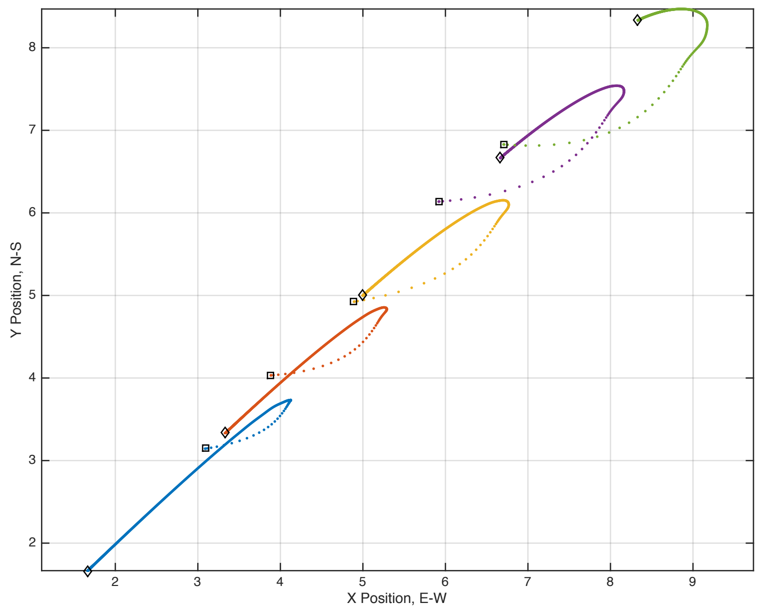
$$\psi_i = \phi_i \qquad (117)$$

Task pose vector:

$$\begin{bmatrix} e_{xt} \\ B_i \\ \gamma_i \\ \psi_i \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{\left((x_{E_2}-x_{E_1})(y_{E_1}-y_c)-(x_{E_1}-x_c)(y_{E_2}-y_{E_1})\right)^2}{\left(x_{E_2}-x_{E_1}\right)^2+\left(y_{E_2}-y_{E_1}\right)^2}} \\ \frac{\rho_i^2}{\rho_{i+1}^2} \\ \alpha_i \\ \phi_i \end{bmatrix}$$

where (x_{E_1}, y_{E_1}) and (x_{E_2}, y_{E_2}) are the positions of the end stations that are being connected by the multi-robot communication system.

Results



Many results are presented in the body of this work. Here we show the end node moving away from the origin, then beginning to return, and a series of robots tracking its motion in the communication task space.

Discussion

A unique aspect of this task is the state dependence on uncontrolled states. Computation of the task states requires knowledge of these external states, much like the adaptive navigation task. In this case, we use a model-based Jacobian to direct the robots to move appropriately. Our chosen model is simple but it is reasonably sufficient, even for experiments where this simplified model is inaccurate. Per a literature review, accurately modeling communications environments is complex due to non-uniform antenna radiation patterns, shadowing of vehicles, interference and multi-path effects. Furthermore, these can influence system stability. If a vehicle overshoots its target position (or communication task command) and must turn around, the measurement in its new orientation may flip the direction of the error and cause it to turn around again. This suggests a need for full characterization of a system prior to evaluating dynamic response.