12-25-2011

Cluster Control of Automated Surface Vessels

Paul David Mahacek
Santa Clara University

SANTA CLARA UNIVERSITY

Department of Mechanical Engineering

Date: December 25, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Paul David Mahacek

ENTITLED

Cluster Control of Automated Surface Vessels

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

ENGINEER'S DEGREE IN MECHANICAL ENGINEERING

_____

Christopher Kitts, Thesis Advisor

_____

Drazen Fabris, Chairman of Department

Cluster Control of Automated Surface Vessels

By

Paul David Mahacek

ENGINEER'S DEGREE THESIS

Submitted in Partial Fulfillment of the Requirements
For the degree of Engineer's Degree
in Mechanical Engineering
in the School of Engineering
Santa Clara University, 2011

Santa Clara, California

December 25, 2011

Cluster Control of Automated Surface Vessels

Paul Mahacek

Department of Mechanical Engineering
Santa Clara University
2011

ABSTRACT

This research focuses on the design and control of a fleet of robotic kayaks, and presents experimental data regarding the functionality and performance of the system. One of the key technical challenges in fielding multi-robot systems for real-world applications is the coordination and relative motion control of the individual units. Coordinated formation control of the fleet is implemented through the use of the cluster space control architecture, which is a full-order controller that treats the fleet as a virtual, articulating, kinematic mechanism. The resulting system is capable of autonomous navigation utilizing a centralized controller, currently implemented via a shore-based computer that wirelessly receives ASV data and relays control commands. Using the cluster space control approach, these control commands allow a cluster supervisor to oversee a flexible and mobile formation formed by the ASV cluster. This paper includes an extended appendix which includes MatLab and Simulink code as well as two publications completed in the process of this research.

Keywords: Cluster Space Control, Autonomous / Unmanned Surface Vessel, Obstacle Avoidance, Shielding

CONTENTS

SECTION I: MULTI-ROBOT RESEARCH PROGRAM

Robots are useful. Really useful. You just won't believe how vastly, hugely, mindbogglingly useful they are. You might think a screwdriver is useful, but that's just peanuts to robots. Listen, they offer a wide variety of functions over a large range of applications [1] Due to their endurance, speed, precision, versatility and their ability to withstand conditions far beyond what a human would be able to, they are the perfect tool for research in extreme environments. These environments range from the depths of the ocean to the far reaches of space. In the marine environment specifically, robots like autonomous underwater vessels (AUVs), remotely operated vehicles (ROVs), or unmanned surface vessels (USVs) handle long term exposure and the pressures of the ocean's depth with ease. But these extreme environments can also create many challenges for the robotic systems and their designers. Tasks for mobile robots, like path planning or obstacle avoidance can often become difficult in these remote operations (due to a lack of telepropreoception).

There is increasing interest in applications where not just one but multiple robots are the ideal solution. Spatially diverse operation, flexible arrangement and rearrangement, increased coverage area, increased data, and agent redundancy are just some of the features of multi-robot systems. These features enable operations like simultaneously sampling multiple locations in a dynamic environment or optimizing sensor location and geometry to minimize errors in remote sensing application. Numerous other in-situ, remote sensing and even physical manipulation tasks are enabled by utilizing multi-robot systems.

However, these benefits of multi-robot systems are not without their challenges. Hurdles like communication, sensing, and actuation are difficult with one robot. But the difficulty is magnified exponentially by the inclusion of additional robots into the system. Another of these obstacles in fielding a mobile mult-robot system is the navigation strategy used to guide the group of robots. Assuming that control of one robot is easy (which is often not the case), the task of a mission planner, supervisor, or real-time operator can become very difficult when increasing to just two or three robots under his or her control, and the task becomes nearly impossible as the cluster increases to greater numbers.

The goal of the Santa Clara University Robotic Systems Laboratory's (SCU RSL) work in the field of multi-robot systems is to facilitate the operation of multi-robot systems. A variety of techniques have been suggested and explored for these systems by others working in this field. Decentralized techniques excel when data exchange is limited [2]-[3] and centralized approaches can exploit global information if it is available [4]-[5]. Several different behavioral, nature inspired, and potential field techniques have been demonstrated [6]-[8]. Other less sophisticated systems use techniques like blind leader/follower, and spatially or temporally offset trajectories. But the RSL has focused its efforts on one controller that provides a straightforward method of specifying and monitoring the formation as well as the ability to achieve highly connected and full order control [9].

SECTION II: CLUSTER SPACE CONTROL

The cluster space control approach uses the idea that the entire group is a single unit, "the cluster", and motion commands are given as functions of cluster attributes. The attributes can vary, but are commonly things like position, angle, distance and orientation. The motion of the cluster of mobile robots is similar to that of a virtual kinematic mechanism, and as such all the attributes are easily monitored and varied through a set of independent system state variables. These state variables make up the systems cluster space and are correlated to the robot level variables through a set of kinematic transforms. The kinematic transforms provide several functions including translation of cluster level commands into actuation of individual robots, and the ability to convert data from an assortment of different sensors into the cluster space [9].

The exact control type can vary from a basic PID controller to more sophisticated nonlinear dynamic controllers. The basic PID controller determines the error from the desired cluster velocity or position for each cluster space variable and then uses an inverse Jacobean transform to convert to robot level velocity or position commands. The versions employed in the works presented below are kinematic, resolved rate controllers with the robots handling velocity control on board.

The RSL has demonstrated clusters of up to six vehicles with obstacle avoidance, and has simulations of higher numbers. We have implemented designs using both holonomic and nonholonomic robots, piloted and supervised modes, and a variety of relative and absolute positioning systems and sensing methods. The application reported on here, guarding and shielding, is a follow-on to earlier work in escorting and patrolling [10]-

[13], and has also been applied to other tasks including gradient tracking [14]-[15] and reconfigurable sparse array communication. [16]

Another new technique developed in this work is the application space. Application space is a second layer of abstraction that transforms user-specified application variables into desired cluster space variables. Application variables are typically more detailed and specific than cluster space variables, and often they are used to consolidate several degrees of freedom in the cluster space. These variables can be controlled, prescribed, or tied to environmental interactions.

SECTION III: RESEARCH OBJECTIVES

Having established that robots are useful and that more robots are better, and provided a technique for controlling these groups of robots, we can now explore some of the applications of mobile muli-robot clusters. Looking specifically at the marine environment, such systems include remote sensor nodes, energy harvesting systems, manned ships and their support equipment, and unmanned vehicles operating both under and on the water's surface.

In this work we use a cluster of five Unmanned Surface Vessels (USV) to establish a shield around a vessel. This guarding technique was motivated by work done by the RSL in Lake Tahoe, where high boater traffic often makes it dangerous to operate. In ROV operations when the tether is at the surface it can be very hard to see. Recreational boaters can be unaware, inexperienced or distracted creating a hazard to both the boater

and the crew running the ROV. The desire was for a group of mobile buoys that would create a visual barrier around the operation, a perfect application for cluster space control.

Previous research has shown the robustness of the cluster space in controlling the dynamic motions of a small cluster of unmanned surface vessels [10]-[13] as well as proving obstacle avoidance for land based robots [17]. The objective of this research project was to demonstrate autonomous cluster space control on a larger group of unmanned surface vessels, explore the potential of the application space, to develop a new cluster formation and to implement an obstacle avoidance technique. The resulting cluster space controller was verified in simulation, tested on land-based robots, and finally verified in field testing with a cluster of five USVs. Furthermore the design of the USVs was advanced in several areas including increasing the overall robustness of the onboard electronics and mounting hardware, improvements to the transport mechanism and formalization of the deployment strategy.

SECTION IV: USV TEST BED

The development of the marine test bed has evolved over several iterations. It has been a key design requirement from the beginning to maintain as much similarity with the land based robot test bed in terms of hardware, software, interface, and coordinate system. For example, the use of common bus architecture across all RSL robotic vehicles enables a rapidly reproducible control system capable of transparently controlling multiple platforms, including several different types of land rovers, aerial vehicles, and marine vessels. The current system hardware can be seen in Fig. 1 and Fig. 2 on the next page.

Fig. 1 – Five of the RSL's unmanned surface vessels ready for a test at Lake Del Valle near Livermore, CA

The design of the vessels themselves has been primarily focused on ease of operation and low cost. By mainly using off the shelf parts, like a readily available kayak requiring no permanent modifications, new vessels and replacement components are easily integrated if the need arises. Utilizing plastics, fiberglass, and other corrosion and UV resistant materials ensure a long life in the marine environment.

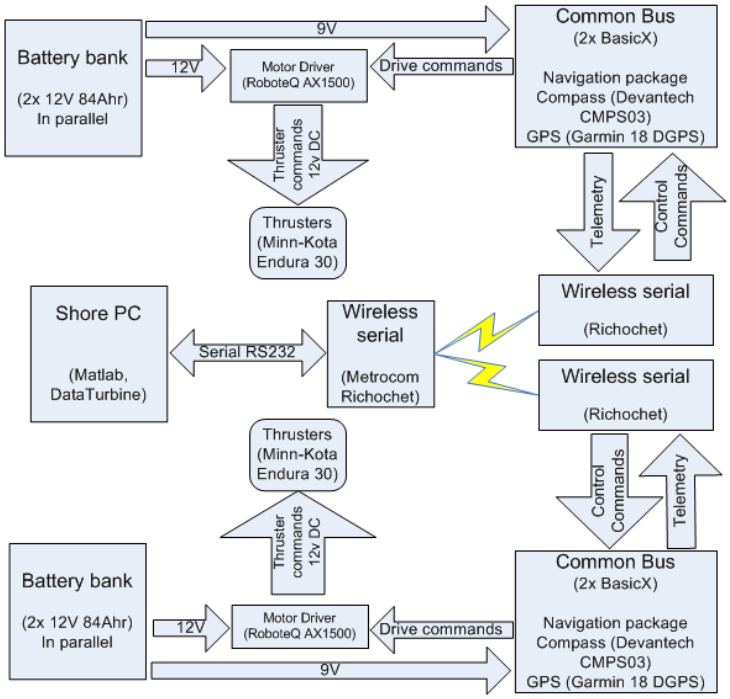


Fig. 2 – Hardware component block diagram example for two kayaks [11]

Upgrades and modifications have been made to the USVs as stated earlier. The onboard electronics have been fully vetted and several components that had previously caused intermittent issues in some of the units have been replaced or eliminated. The mounting structure has been modified to reduce the overall size, weight and required assembly time which has eased both the deployment and transportation of the vessels. Further details of the previous iteration as well as the current control hardware, protocol, propulsion, and power subsystems have been previously described [11]-[13].

SECTION V: PUBLICATIONS

This section is primarily composed of two articles. The first is a journal article describing the work done on this research project. The paper describes the control architecture used to establish the guarding behavior. It reviews the design of the robotic kayaks, and briefly discusses some of the hardware development. Finally it presents simulated and experimental data of the system performance and functionality. It has been accepted for publication in the February 2012 focus issue of IEEE/ASME Transactions on Mechatronics which has a Journal Citation Reports ranking of #1 in Manufacturing Engineering, #4 in Mechanical Engineering, and #4 in Automation & Control.

The second paper was presented at OES/IEEE - AUV2010 in Monterey California on September 3, 2010. It is a review of the initial work done on the most recent phase of the research project. It similarly discusses the cluster space control architecture and briefly notes the hardware, but mainly focuses on the simulation of a smaller cluster of vessels which were then expanded and tested in the field as shown in the journal article.

In addition to these two publications several other articles, talks and posters have been generated from this research. [12] is a journal publication and [20] is a conference publication and talk, both discussing cluster space control of surface vessels. [18] and [19] focus on applications of cluster space control in both marine and non-marine applications. [21] and [22] are respectively a conference paper and a poster, both focusing on using the cluster in a gradient tracking application.

The articles included in appendices A and B have been formatted to fit your viewing device but retain all the original content and are subject to the following disclaimer:

This work was supported in part by the National Science Foundation under Grant CNS0619940, and by financial support from NASA, and Santa Clara University; any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation, NASA, or Santa Clara University.

SECTION VI: SUMMARY AND CONCLUSIONS

In this research we focused on the design and control of a fleet of robotic kayaks, and presented experimental data regarding the functionality and performance of the system. We described the use of a fleet of robotic marine vessels capable of guarding critical assets from threats. Coordinated formation control of the fleet was implemented through the use of the cluster space controller. An application-specific layer was integrated with

the cluster space controller, allowing an operator to directly specify and monitor guarding-related parameters.

This system has been experimentally verified in the field with a fleet of robotic kayaks in this research. The control architecture used to establish the guarding behavior and the design of the robotic kayaks were reviewed, and experimental data regarding the functionality and performance of the system was presented. As a result, the five-robot cluster space definition and control architecture was validated and functionality was proven for this application. This paper includes an extended appendix which includes MatLab and Simulink code as well as two publications completed in the process of this research.

REFERENCES

[1] D. Adams, Adapted from "The Hitchhiker's Guide to the Galaxy" BBC 1978

[2] E. Fiorelli, N. Leonard, P. Bhatta, D. Paley, R. Bachmayer, D. Fratantoni. "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE Journal of Oceanic Engineering*, v 31, n 4, October 2006, pp. 935-948.

[3] Y. Tan and B. Bishop, "Evaluation of robot swarm control methods for underwater mine countermeasures," *Proceedings of the 2004 Annual Southeastern Symposium on System Theory*, v 36, pp. 294-298

[4] K. Tan, and M. Lewis, "Virtual Structures for High-Precision Cooperative Mobile Robotic Control," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996, pp. 132-139.

[5] C.P. Tang, R.M. Bhatt, M.Abou-Samah, and V. Krovi, "Screw-theoretic analysis framework for cooperative payload transport by mobile manipulator collectives," *IEEE/ASME Transactions on Mechatronics*, vol 11, no 2, April 2006, pp. 169-178.

[6] T. Balch and R. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol14, no 6, December 1998, pp. 926-939.

[7] E. Flinn, "Testing for the 'boids'," *Aerospace America*, v 43, n 6, June, 2005, pp. 28-29.

[8] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," *Proceedings IEEE Conf. Decision and Control*, 2001, pp. 2968–2973.

[9] C. A. Kitts and I. Mas, "Cluster space specification and control of mobile multirobot systems," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, April 2009, pp. 207–218.

[10] I. Mas, S. Li, J. Acain, and C. Kitts. "Entrapment/escorting and patrolling missions in multi-robot cluster space control." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO*, Oct 2009, pp. 5855-5861.

[11] P. Mahacek, Design and Cluster Space Control of Two Autonomous Surface Vessels. Santa Clara University Dept. of Mechanical Engineering Masters Thesis, December 2009.

[12] P. Mahacek, I. Mas, O. Petrovic, J. Acain, and C. Kitts. "Cluster space control of autonomous surface vessels." *Marine Technology Society Journal*, v 43 n 1, 2009, pp. 13-20

[13] P. Mahacek, I. Mas, C. Kitts, *"Cluster space control of autonomous surface vessels utilizing obstacle avoidance and shielding techniques,"* Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES , vol., no., pp.1-5, 1-3 Sept. 2010

[14] T. Adamek, Cluster Space Gradient Contour Tracking for Mobile Multi-Robot Systems. Santa Clara University Dept. of Mechanical Engineering Masters Thesis, January 2011.

[15] V. Howard, A Study of Gradient Climbing Techniques Using Cluster Space Control of Multi-Robot Systems, Santa Clara University Dept. of Mechanical Engineering Masters Thesis, March 2011.

[16] G. Okamoto, C. Chen, and C. Kitts. "Beamforming Performance for a Reconfigurable Sparse Array Smart Antenna System via Multi-Robot Control." *Proceedings of the SPIE Defense, Security, and Sensing Conference, Orlando FL*, April 2010, pp. 1-11.
[17] I. Mas, Obstacle Avoidance Policies for Cluster Space Control of Non-Holonomic Multi-Robot Systems. *IEEE/ASME Transactions on Mechatronics,* In Press.
[18] C. Kitts, P. Mahacek, I. Mas, and T. Adamek. "Initial Applications in Cluster Space Control of Multi-Robot Systems." Proceedings IEEE Space Mission Challenges in Information Technology, Palo Alto CA, August 2011. In press.
[19] C. Kitts, P. Mahacek, T. Adamek, and I. Mas. "Experiments in the Control and Application of Automated Surface Vessel Fleets." Proceedings IEEE/MTS Oceans Conference, Waikaloa HI, Sep 2011. In press.
[20] P. Mahacek, I. Mas, O. Petrovic, J. Acain and C. Kitts. "Cluster Space Control of a 2-Robot System as applied to Autonomous Surface Vessels." Proceedings IEEE/MTS Oceans Conference, Quebec City, Canada, pp. 1-5, Sep 2008.
[21] P. Mahacek, T. Adamek, V. Howard, K. Rasal, C. Kitts, B. Kirkwood, and G. Wheat. "Cluster Space Gradient Tracking – Control of Multi-Robot Systems." Proceedings ASME Information Storage and Processing Systems Conference, Santa Clara CA, June 2011. In press.
[22] T. Adamek, S. Li, V. Howard, P. Mahacek, K. Rasal, C. Kitts. "Cluster Space Gradient Tracking – Control of Multi-Robot Systems." Poster in Proceedings IEEE/OES Autonomous Underwater Vehicles Conference, Monterey CA, Sep 2010.

APPENDIX A. JOURNAL ARTICLE

Dynamic Guarding of Marine Assets through Cluster Control of Automated Surface Vessel Fleets

Paul Mahacek, *Student Member, IEEE*, Christopher A. Kitts, *Senior Member, IEEE,*
Ignacio Mas, *Student Member, IEEE*

Abstract—**There is often a need to mark or patrol marine areas in order to prevent boat traffic from approaching critical regions, such as the location of a high-value vessel, a dive site, or a fragile marine ecosystem. In this paper we describe the use of a fleet of robotic kayaks that provides such a function: the fleet circumnavigates the critical area until a threatening boat approaches, at which point the fleet establishes a barrier between the ship and the protected area. Coordinated formation control of the fleet is implemented through the use of the cluster space control architecture, which is a full-order controller that treats the fleet as a virtual, articulating, kinematic mechanism. An application-specific layer interacts with the cluster space controller in order for an operator to directly specify and monitor guarding-related parameters such as the spacing between boats. This system has been experimentally verified in the field with a fleet of robotic kayaks. This paper describes the control architecture used to establish the guarding behavior, reviews the design of the robotic kayaks, and presents experimental data regarding the functionality and performance of the system.**

***Index Terms*—Multi-robot systems, formation control, collaborative control, robot teams, cluster space.**

I. INTRODUCTION

Mechatronic systems provide benefits in a wide range of applications given their strength, speed, precision, and ability to withstand extreme environments. In the marine environment, such systems include remote sensor nodes, energy harvesting systems, manned ships and their support equipment, and unmanned vehicles operating under water and on the surface of the sea.

Unmanned Surface Vessels (USVs) have been used for nearly 70 years in order to reduce the risks and costs associated with activities ranging from military operations to scientific characterization [1]. Early USV systems were remotely piloted and used for applications such as serving as gunnery targets or mine countermeasure drones [2]. Over the past two decades, advances in GPS-based position sensing, wireless communication, navigation, and automation technologies have enabled a variety of new USV applications such as towing objects, mine-sweeping, exploration, and serving as communication relays between underwater assets and remote control nodes. Excellent reviews of the many USV systems that have been developed for such applications are provided in [3-5].

Recent advances in multi-robot control techniques have led to the development of several multi-USV systems. Potential advantages of multi-USV systems include redundancy, increased coverage and throughput, flexible reconfigurability, spatially diverse functionality, and the fusing of physically distributed sensors and actuators; applications capable of exploiting such features range from remote and *in situ* sensing to the physical manipulation of objects [6].

One of the first implemented multi-USV systems was the Massachusetts Institute of Technology's SCOUT system, comprised of several robotic kayaks [7]. In addition to serving as a multi-USV navigation testbed, fleets of 2-4 SCOUT vehicles have been used to explore support applications for autonomous underwater vehicles, such as serving as a communications relay and providing long-baseline navigation services [8]. Researchers at Carnegie-Mellon University have networked two of their OASIS USVs to explore telesupervised aquatic sensing; this system has been demonstrated experimentally with

field studies detecting and characterizing simulated harmful algae blooms [9]. Work at the U.S. Naval Academy (USNA) has focused on using multiple tugboats to cooperatively manipulate and propel other ocean vehicles through the use of swarm navigation techniques [10]. In a 2009 demonstration during the Navy's Trident Warrior exercise, the CARACaS (Control Architecture for Robotic agent Command and Sensing) autonomy architecture was used on several USVs in order to verify the use of this behavior-based control system for asset protection and riverine survey applications [11]. Other concepts include the fleet of small-scale Drosobots developed for sampling applications [12], and the open source Protei development effort to field a fleet of sailing drones for oil and pollution clean-up services [13].

The work presented in this paper aligns with many of the themes presented in [14], which discussed the use of USVs as automated buoys, such as those used by the Naval Undersea Warfare Center [15]. In particular, this work envisioned multi-USV buoy systems for a variety of marine applications ranging from marine traffic management to distributed sensing. Potential benefits identified for such a system included the ability to rapidly deploy a buoy line, the ability to dynamically reposition the buoys, and reduced deployment and maintenance costs.

There are many challenges to fielding multi-USV systems, to include providing robust communications, the incorporation and fusion of distributed sensing and actuation capabilities, the human-machine interfaces to enable efficient monitoring and specification of tasks, and achieving cost-effective production and operation. One particularly challenging issue is the navigation strategy used to guide the absolute and relative motions of the fleet. A wide variety of techniques have been and continue to be

explored for this capability for multi-robot systems in general. When limited information exchange is a primary constraint (due to physical distribution or constrained bandwidth), decentralized control approaches are often pursued [16]-[17]. Behavioral, biologically inspired, and potential field techniques have been successfully demonstrated [18]-[20], although they often lack mathematical formality. Centralized approaches exploiting global information exist, but they are often not preferred due to limited scalability; however, they may be ideal when tight robot interaction is required by applications such as the realtime fusing of sensors or actuators [21]-[22].

Specific to the multi-USV systems previously cited, several systems use a very loose form of coordinated navigation in which each USV blindly follows its own trajectory, but the trajectories are spatially (as with the Drosobots) or temporally (as with OASIS) offset in order to divide and conquer the task at hand. The USNA tugboat fleet, however, employs a much tighter coordination strategy in order to achieve manipulation tasks.

The work presented in this article employs a specific coordinated navigation control approach known as *Cluster Space* control [23], which we have previously demonstrated experimentally on land rover, aerial robot and surface ship systems. We have developed this controller in order to enable benefits such as natural specification and monitoring of formation performance and the ability to achieve highly connected and full-order control. Our current work introduces an application-layer above the centralized formation controller, transforming application-specific specifications into cluster space control specifications; these are used to implement the realtime cluster controller, which in turn determines the drive commands for each individual robot in the fleet. Section II

of this paper reviews the cluster space control approach and its integration with a specific application, that of dynamically establishing a barrier between threatening marine traffic and an asset that must be protected. Section III reviews the design of the multi-USV system. Section IV presents experimental field data, and Section V discusses future work and draws conclusions about the significance of this work.

II. The Cluster Space Controller

Our research in Cluster Space Control is motivated by our vision of a specific class of multi-robot applications that require complete degree-of freedom control of the spatial and motion characteristics of a locally distributed mobile multi-robot system that tightly interacts in realtime. At the same time, we desire transparency for the formation's degrees of freedom in order for a realtime human pilot or supervisory controller to specify, control and/or monitor performance.

Because the cluster space technique allows direct specification of any spatial state variables of interest, it avoids potential drawbacks of other well-known multi-robot control strategies. For example, compared to virtual bodies and artificial potentials approach [20], there is no need to iteratively tune potential fields or to select artificial leader positions in order to achieve the motion characteristics of interest. Compared to leader-follower techniques [24], specification is not limited to the distance and/or angle between leader-follower pairs within the formation. In contrast to virtual body techniques [25], all pose degrees of freedom may be continuously articulated. Some of these advantages come at the cost of increased computation within the realtime control loop; however, the cluster space approach can be implemented with varying levels of (de)centralization [25], and we have had success exploring strategies such as multi-rate

control [26]; these strategies are both suitable for dramatically reducing computational load and the need for information sharing throughout the cluster.

A. The Cluster Space Control Approach

Central to the cluster space strategy are the concepts of considering the n-robot system as a single entity, a "cluster," and of specifying motions with respect to cluster attributes, such as position, orientation, and geometry; we note that all of these attributes may be easily varied such that a reasonable analogy is that a cluster of mobile robots moves like a virtual kinematic mechanism. Our approach is to use the cluster attributes to guide the selection of a set of independent system state variables suitable for specification, control, and monitoring. This collection of state variables constitutes the system's cluster space and can be related to robot-specific state variables through a formal set of kinematic transforms. A supervisory operator or realtime pilot specifies and monitors cluster motion, and control computations are executed with respect to the cluster space variables, (which leads to well-behaved motions in the cluster space). Kinematic transforms allow compensation commands to be derived for each individual robot, and they also allow data from a variety of sensor packages to be converted to cluster space state estimates.

As an example of this, consider the case of a simple, planar two-robot cluster, which is detailed in [23] and shown in Fig. 1. A conventional robot space definition of the pose of this system would include the position and orientation of each robot as measured in the global frame: ${}^{G}\vec{R} = (x_1, y_1, \theta_1, x_2, y_2, \theta_2)^T$. To consider the cluster perspective, assume that a cluster frame is placed at the midpoint between the two robots and oriented towards Robot 1. A reasonable cluster space description of the cluster's

pose would include the location and orientation of the cluster frame, a single variable representing the cluster geometry (in this case, we use the distance to each robot from the cluster origin), and the relative orientations of each robot with respect to the cluster frame; this results in a cluster pose vector of $\vec{C} = (x_C, y_C, \theta_C, d, \phi_1, \phi_2)^T$.

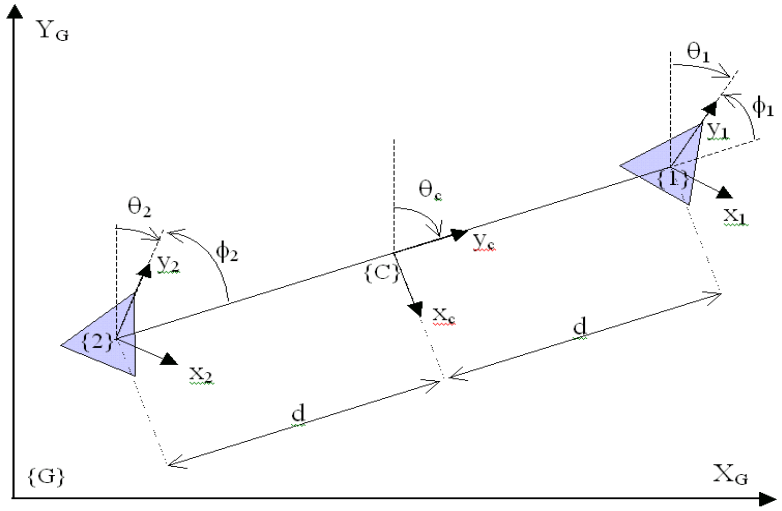


Fig. 1 – Representing the pose of a two-robot system using a cluster space description.

Mathematical relationships that relate these robot and cluster space variables constitute the position kinematic functions; for example, the cluster's *x* and *y* location is the average of the *x* and *y* locations of the two robots. Furthermore, the robot and cluster space velocities, ${}^G\dot{\vec{R}}$ and $\dot{\vec{C}}$, can also be formally related to each other. For example, computing the partial derivatives of the cluster space pose variables allows the development of a Jacobian matrix, , that maps robot velocities to cluster velocities in the form of a time-varying linear function:

$$\dot{\vec{C}} = \begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \\ \vdots \\ \dot{c}_{mn} \end{pmatrix} = {}^G J({}^G\vec{R}) {}^G\dot{\vec{R}} = \begin{pmatrix} \frac{\partial g_1}{\partial r_1} & \frac{\partial g_1}{\partial r_2} & \cdots & \frac{\partial g_1}{\partial r_{mn}} \\ \frac{\partial g_2}{\partial r_1} & \frac{\partial g_2}{\partial r_2} & \cdots & \frac{\partial g_2}{\partial r_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{mn}}{\partial r_1} & \frac{\partial g_{mn}}{\partial r_2} & \cdots & \frac{\partial g_{mn}}{\partial r_{mn}} \end{pmatrix} \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_{mn} \end{pmatrix} \qquad (1)$$

The controller itself can take on several forms given the needs of the system and application. For example, a simple form would consist of a linear PID controller that computes compensations in the form of instantaneous cluster velocity set-points, which are then transformed to individual instantaneous robot velocity set-points through the use of an inverse Jacobian transform; this is a kinematic, resolved-rate controller appropriate for robots with their own velocity-control capabilities. This style of controller is depicted in Fig. 2, and it is the architecture employed for the work reported on in this article. We have also developed and implemented more sophisticated nonlinear dynamic controllers. Such a controller uses a partitioned model-based strategy and computes compensations in the form of the abstracted cluster space forces and torques necessary to manipulate the virtual kinematic mechanism; these compensations are converted by a Jacobian transpose transform to individual robot-level control forces/torques for dynamic control of the individual vehicles [27].

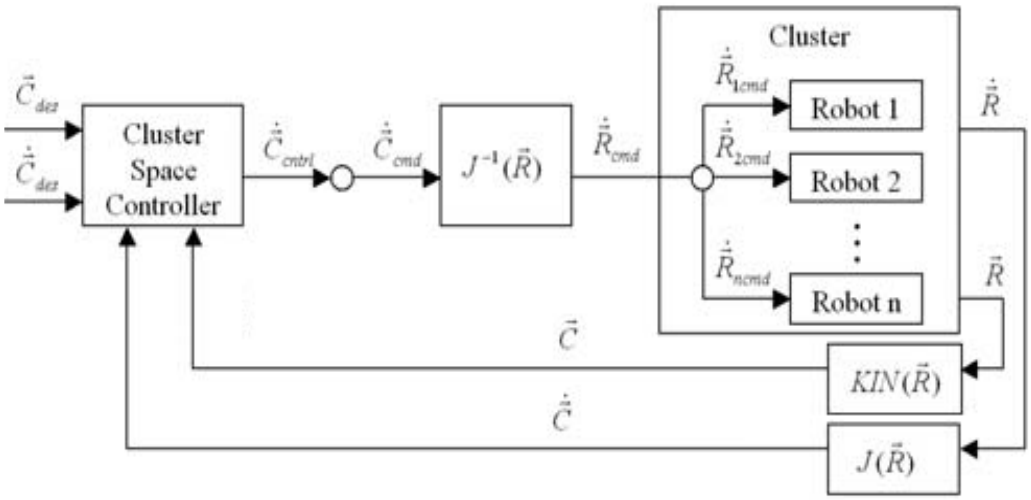


Fig. 2 – Inverse Jacobian Cluster Space Control Architecture for a Mobile Multi-Robot System.

To date, we have successfully implemented cluster space control in experiments with clusters of up to 6 vehicles, for both holonomic and nonholonomic robots, for robots negotiating obstacle fields, for piloted and supervisory control modes, and for a variety of relative/absolute positioning and tracking pose sensing systems. The guarding/shielding application reported here is an extension of our previous work in escorting/patrolling [28]-[30]. In addition, we are applying the control strategy to other applications such as gradient-based environmental sensing [31]-[32] and reconfigurable sparse array communication systems [33].

B. Cluster Space Kinematic Transforms for the Dynamic Guarding Application

In exploring the dynamic guarding application, we have applied the cluster space control framework in numerous ways, each varying the selection of pose variables. For the experiments presented in Section IV, the selection of these variables was driven by the guarding application. This application involves the creation of a "fence" that becomes denser as a threat approaches and which is positioned between the threat and the asset being guarded. From this perspective, the position of the asset being protected and the location of the threat (its bearing from the asset and its proximity) dictate the deployment of the robots in the creation of a fence that is properly positioned with an appropriately dense "fence spacing."

Fig. 3 depicts the relevant reference frames and geometric layout for a planar 5-USV cluster. To complement the sensor data used in experimentation, the global frame was defined with X_G pointing East and Y_G pointing North. The robot space pose vector is

$$ {}^{G}\vec{R} = (X_0, Y_0, \varphi_0, \ldots, X_5, Y_5, \varphi_5)^T $$

where (X_0, Y_0, φ_0) is the pose of the protected object and (X_i, Y_i, φ_i) is the pose of each of the robots where i=(1,2,3,4,5). We note that we are treating the protected asset as an element of the cluster, although it is not directly controlled by the cluster space dynamic guarding policy. We also note that for this application, robot orientation is not critical in establishing a fence; in fact, given the non-holonomic constraints of the boats used in the application, they are not independently specified. For this reason, and to simplify the presentation of the underlying mathematics, we drop them from consideration in independently specifying the pose of the fleet. This leaves us with two degrees of freedom for each of the six fleet entities (five boats and the protected asset), yielding a total of 12 linear degrees of freedom for the robot group.

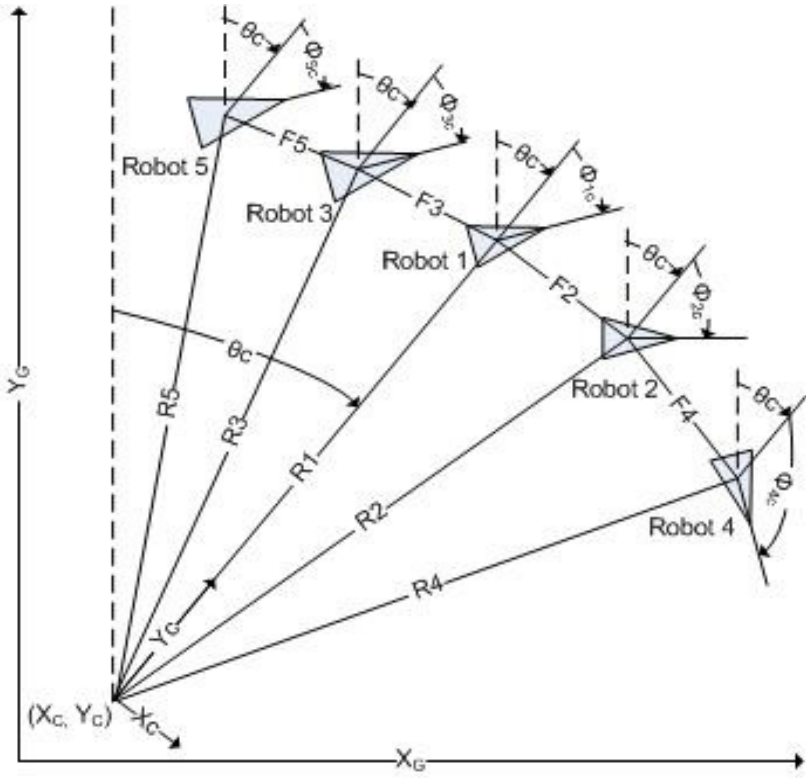


Fig. 3 – 5-USV Cluster Geometry.

From the cluster perspective, we place the cluster frame origin, denoted by (X_c, Y_c), at the location of the protected object, and we orient the frame such that the cluster heading, θ_c, points the frame towards the location of robot 1. The locations of robots 1 - 5 are specified in part by the radial distances, R_1 - R_5, from the asset being protected to each individual robot. In addition, the positions of robots 2 and 3 are defined by a spacing from robot 1 given as F_2 and F_3. Similarly, robots 4 and 5 are each positioned by a spacing F_4 and F_5 from robots 2 and 3, respectively. Further expansion of the cluster can be achieved by adding robots to either end of the cluster using this even-odd convention.

The cluster space pose vector is therefore

$$\vec{C} = (X_c, Y_c, \theta_c, R_1, R_2, R_3, R_4, R_5, F_2, F_3, F_4, F_5, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5)^T.$$

where each φ_i is the relative rotation of each robot with respect to the cluster frame, for i=1-5. As previously stated, given that the kayaks are non-holonomic vehicles, robot orientations are removed as freely specified variables, and the mathematical development that follows is independent of these variables. The controller used for this study uses an inner loop heading controller to orient each vehicle in the direction of desired motions. We have also developed a formally constructed non-holonomic controller for use in systems of this type [34]. Removing the φ_i angles from consideration leaves us with 12 cluster degrees- of-freedom, matching the 12 linear degrees of freedom in robot space.

Given ${}^G\vec{R}$ and $\vec{C}$, the set of forward position kinematic equations, $\vec{C} = KIN({}^G\vec{R})$, is given by Eqs (2)-(7):

$$X_c = X_0 \quad (2)$$

$$Y_c = Y_0 \quad (3)$$

$$R_n=((X_n-X_0)^2+(Y_n-Y_0)^2)^{1/2} \qquad for\ n=1,2,3,4,5 \qquad (4)$$

$$\theta_1=Atan2((X_1-X_0),(Y_1-Y_0)) \qquad (5)$$

$$F_2=((X_2-X_1)^2+(Y_2-Y_1)^2)^{1/2} \qquad (6)$$

$$F_m=((X_m-X_{m-2})^2+(Y_m-Y_{m-2})^2)^{1/2} \qquad for\ m=3,4,5 \qquad (7)$$

Inversely, the set of inverse position kinematic equations, ${}^G\vec{R} = INVKIN(\vec{C})$, is given by Eqs (8)-(15):

$$X_0=X_c \qquad (8)$$

$$Y_0=Y_c \qquad (9)$$

$$X_1=X_c+R_1*Sin(\theta_1) \qquad (10)$$

$$Y_1=Y_c+R_1*Cos(\theta_1) \qquad (11)$$

$$X_i=X_c+R_i*Sin(\theta_1+acos((R_1^2+R_i^2-F_i^2)/(2*R_1*R_i))) \qquad for\ i=2,3 \qquad (12)$$

$$Y_i=Y_c+R_i*Cos(\theta_1+acos((R_1^2+R_i^2-F_i^2)/2*R_1*R_i))) \qquad for\ i=2,3 \qquad (13)$$

$$X_j=X_c+R_j*Sin(\theta_1+acos((R_1^2+R_{j2}^2-F_j^2)/2*R_1*R_{j2}))+acos((R_j^2+R_{j2}^2-F_j^2)/2*R_j*R_{j2}))) \qquad for\ j=4,5 \qquad (14)$$

$$Y_j=Y_c+R_j*Cos(\theta_1+acos((R_1^2+R_{j2}^2-F_{j2}^2)/2*R_1*R_{j2}))+acos((R_j^2+R_{j2}^2-F_j^2)/2*R_j*R_{j2}))) \qquad for\ j=4,5 \qquad (15)$$

The forward and inverse velocity kinematics provide the formal relationship between the robot and cluster space velocities, ${}^G\dot{\vec{R}}$ and $\dot{\vec{C}}$. From (2)-(3), we may compute the partial derivatives of the cluster space pose variables, c_i, and develop a Jacobian matrix, J, that maps robot velocities to cluster velocities in the form of a time-varying linear function:

$$\dot{\vec{C}} = \begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \\ \vdots \\ \dot{c}_{mn} \end{pmatrix} = {}^G J({}^G\vec{R})\,{}^G\dot{\vec{R}} = \begin{pmatrix} \frac{\partial g_1}{\partial r_1} & \frac{\partial g_1}{\partial r_2} & \cdots & \frac{\partial g_1}{\partial r_{mn}} \\ \frac{\partial g_2}{\partial r_1} & \frac{\partial g_2}{\partial r_2} & \cdots & \frac{\partial g_2}{\partial r_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{mn}}{\partial r_1} & \frac{\partial g_{mn}}{\partial r_2} & \cdots & \frac{\partial g_{mn}}{\partial r_{mn}} \end{pmatrix} \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_{mn} \end{pmatrix} \qquad (16)$$

In a similar manner, we may develop the inverse Jacobian, ${}^G J^{-1}({}^G\vec{R})$,, which maps cluster velocities to robot velocities. Space prohibits the complete listing of J and J^{-1}.

C. Control Framework for the Dynamic Guarding Application

The general control architecture depicted in Fig. 2 is used for this application, with two modifications as shown in Fig. 4. First, a basic robot-level obstacle avoidance function is added to protect the individual robots from colliding with each other, the object being protected, and the threatening object. When this occurs, the threatened USV negotiates the obstacle in an independent fashion, momentarily breaking away from the formation. Once the obstacle has been avoided, the USV returns to the cluster. As is common for collision avoidance, the avoidance force is a repulsive function that is summed with other control forces. For each USV, the detection radius and the avoidance potential can be independently specified; circular fields are typically used, but an elongated oval can be defined to better model the outer edge of the vessel. It is interesting to note that we have developed a cluster-level obstacle avoidance algorithm, which allows for the entire cluster to move in unison, maintaining the cluster shape while avoiding a collision [34]; this approach, however, was deemed inappropriate for the guarding application since it would too easily allow the threat to simply "push" the entire barrier out of the way.

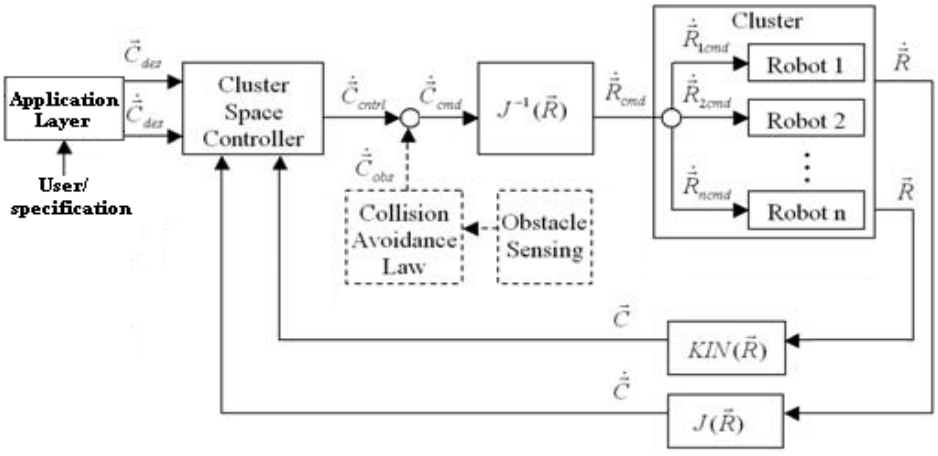


Fig. 4. – Cluster Space Control Architecture for an *n*-Robot system.

The second modification is the augmentation of the input to the controller with an application-space-to-cluster-space function that transforms user-specified application space variables to desired cluster variables. For the implemented guarding application, Fig. 5 indicates the spatial quantities of interest. The overall concept of operation is as follows. The object being protected is at a location (X_{obj}, Y_{obj}). With no threat, the USVs patrol about the object being protected at a minimum specified radius, R_{min}, and evenly spaced in a circle. As a threat approaches from an observed bearing, θ_T, and with a distance, D_T, the USV formation shifts in three ways. First, the USVs rotate about the circumference of the protected region in order to align themselves between the threat and the protected object. Second, the USVs move closer together to form a denser barrier, with some minimum specified spacing, F_{min}. Third, they may also move out towards the threat in order to meet it at a maximum radius of R_{max}.

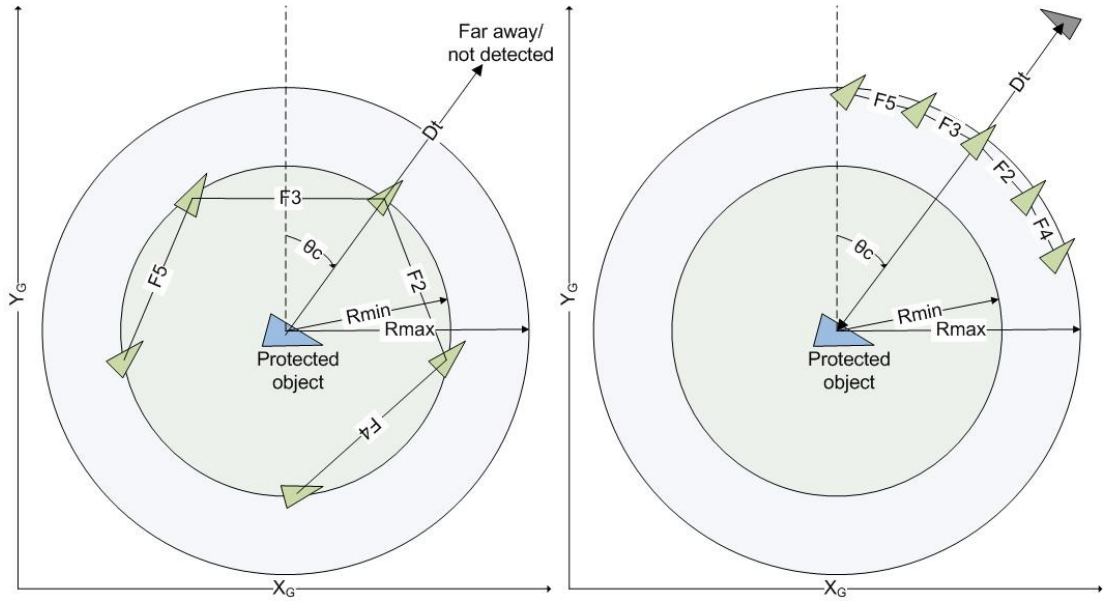


Fig. 5 – Application layer variables showing two cases where the threat is either far away or not detected at all (left), or where the threat is close and the USVs have shifted to guard the protected object or area (right).

Given these specifications, the instantaneous specification for the cluster space controller can be derived from an appropriate set of application-space-to-cluster-space transforms. These transforms convert the application-relevant information, (X_{obj},Y_{obj},R_{min},R_{max},D_T,□$_T$,F_{min}), to cluster space variable inputs. For the guarding application, these transforms are of the form represented in Eqs (17)-(21) assuming $D_T > R_{max}$:

$$X_c = X_{obj} \qquad (17)$$

$$Y_c = Y_{obj} \qquad (18)$$

$$\theta_C = \theta_T \qquad (19)$$

$$R_n = R_{min} + (R_{max} - R_{min})/(D_T - R_{max} + 1) \qquad \text{for } n=1,2,3,4,5 \qquad (20)$$

$$F_n = F_{max} - (F_{max} - F_{min})/(D_T - R_{max} + 1) \qquad \text{for } n=2,3,4,5 \qquad (21)$$

where $F_{max} = 2 * R_n * \sin(\pi/m)$ for m=5 (the number of robots) ; F_{max} is the distance between robots when evenly spaced in a pentagon around the protected asset.

In Fig. 4, this set of application layer transforms operate on the specifications provided by the supervisory operator and provide the resulting cluster space desired values to the cluster control loop. The application transforms essentially act as a set of inverse position kinematics between these two spaces.

There are two critical observations to be made about this architecture. First, realtime control computations are still being performed in the cluster space (e.g., realtime errors and controller compensation commands are cluster space variables). Second, the application space specification of the task is independent of the number of robots. This means that the multi-USV cluster will behave as desired no matter how many USVs are

in the fleet. This is particularly important in order to ensure graceful constitution and degradation of the cluster as the fleet is incrementally fielded and when anomalies occur.

III. HARDWARE

Several iterations of design have occurred to bring the USV system to its present design. The design of the vessels emphasizes versatility, ease of operation, and low cost in all design segments. The use of a common bus architecture across all Robotic Systems Lab cluster vehicles enables a rapidly reproducible control system capable of transparently controlling multiple platforms, including several different types of land rovers, an aerial vehicle, and two different types of USVs. Off-the-shelf components and an adjustable structure facilitate both ease of integration and quick replacement in the case of a malfunctioning component.

Fig. 6 – One of the robotic kayaks.

A. Electronics Hardware and Protocol

The common bus architecture includes all communication and navigation components for each robot in the cluster. The computing stack is made up of two BasicX microcontroller boards. One board accepts drive commands and controls the motor driver boards accordingly in order to run the boat's thrusters. The other board collects position data and interfaces with the wireless communication system. A digital Devantech compass provides heading data, and a Garmin 18 differential GPS unit determines the position and translational velocities; these are low-cost sensors with accuracies on the order of 3° and 3 meters. The modem is a Metricom Ricochet 128Kbits/s unit, which is capable of relatively long range (2+ miles) communication, handles multiple users well, and has frequency hopping for security, noise rejection and utilization of unlicensed frequencies.

B. Propulsion, Power, and Structure

Propulsion is achieved through the use of two Minn Kota Endura 30 thrusters, configured on each side for differential drive. The motor controller is a Roboteq AX1500 interfaced via an RS 232 connection. A standard marine deep-cycle battery is centrally mounted as shown in Fig. 6. This gives the USV more than a three-hour run time at normal operations with a top speed of five knots. The mounting structure is made from 6061 aluminum tubing with a UHMW polyethylene motor mounting plate. Several different sit-on-top style kayaks are currently in use and were selected for their short, wide hulls, which provide greater stability and more agile turning over longer, narrower models. The wiring harness utilizes an automotive-type connector designed for high-

current low DC voltage. Though it is not rated to be submersible, it is waterproof and has been proven to handle brief submersions at shallow depths.

C. Base station

The key element of the base station hardware is the workstation. Several computers have been used over the course of the research and it has been proven on desktops, laptops and even netbooks. Two Metricom Ricochet modems facilitate radio communications. Several pieces of software including DataTurbine (a ring buffered network bus), Matlab, Simulink, and a VRML simulator (shown in Fig. 7 below), work together to retrieve, process, display and redistribute sensor data, system information and robot commands.

Threat detection is handled as a function of the base station, where the threats are manually tracked from shore or onboard the protected vessel. The threats can easily be specified in the observer's local reference frame and appropriate frame transformations are handled in the application layer.

Fig. 7 – The VRML model is capable of replaying simulated cluster formations and trajectories as well as visualizing real-time robot positioning.

IV. TESTING AND RESULTS

The main objective of this research was to apply the cluster space control architecture to a larger multi-USV system with obstacle avoidance while determining the viability of a new shielding technique applied in application space. Four main test cases were run over the course of a multi-day deployment at Lake Del Valle near Livermore, CA (Fig. 8). The first three cases (basic shielding, varying shield size, and threat detection) were run with five robotic kayaks and a simulated boat being protected. The fourth case is of threat detection using four kayaks to protect a SWATH mapping vessel.

Fig. 8 – Testing in Lake Del Valle near Livermore, CA provided variable winds up to 20 knots, low currents, and boat wakes for an excellent dynamic environment. Kayaks showing standard shielding.

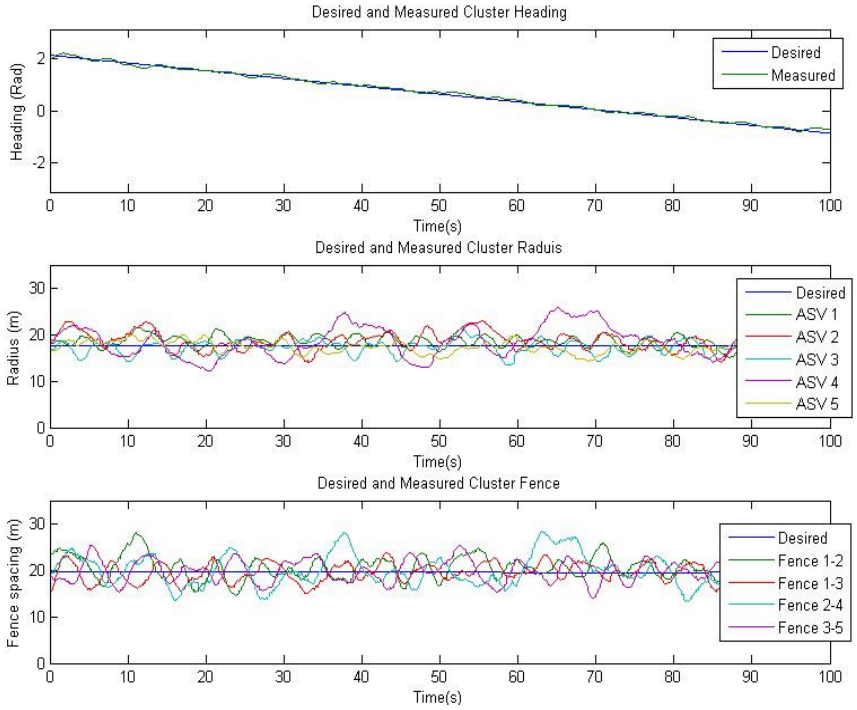


Fig. 9 – Standard shielding, constant radius, no threat. RMSE in table.

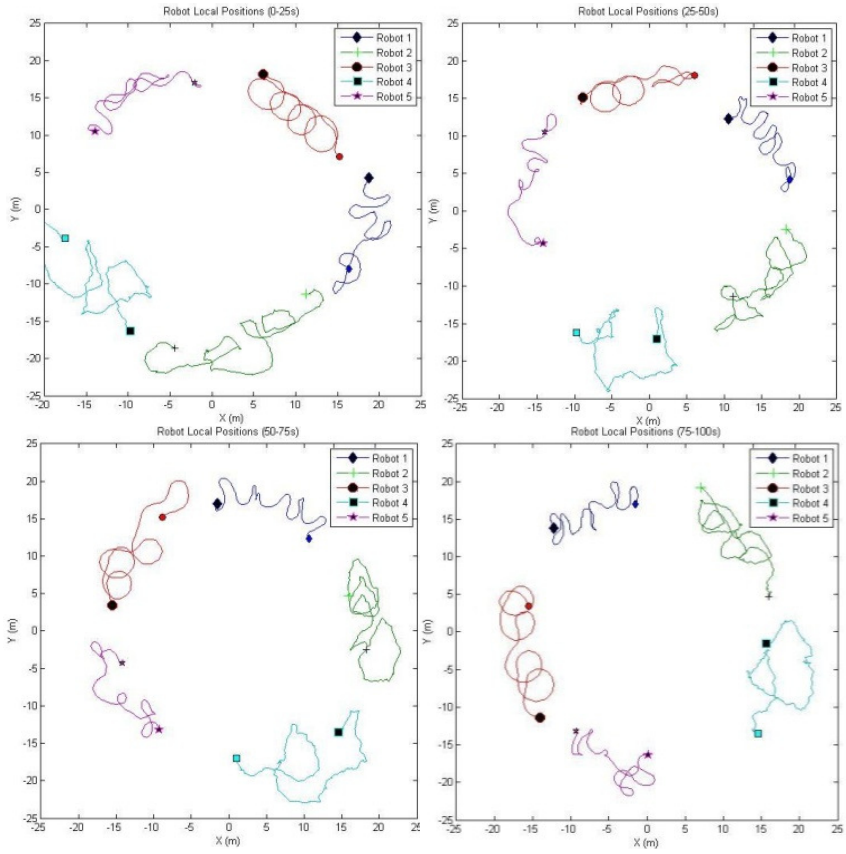


Fig. 10 – Standard shielding, constant radius, no threat (overhead view of same run as Fig. 9.) Looping trail patterns are a function of non-optimized velocity gains, as well as lacking a dead-band around the desired position. Differences in trail patterns can be attributed to various kayak hulls used and the number of service hours on individual thrusters. Further optimization will be attempted in future work.

TABLE A – BASIC SHIELDING: RMS ERROR VALUES FOR THE CLUSTER RADIUS AND FENCE SPACING VARIABLES

Cluster Radii	RMS Error (m)		Cluster Fence Spacings	RMS Errors (m)
R1	1.57		--	--
R2	2.15		F2	2.62
R3	1.57		F3	2.22
R4	3.44		F4	3.46
R5	1.48		F5	2.44

A. Basic Shielding

In the case of the basic shielding technique we are applying it to a simulated boat requiring protection. Using the application space, the operator can set the standard shield radius, the maximum approach and the minimum fence spacing. In this first instance the

standard shield radius is set to 17m, and the minimum fence spacing and approach are disregarded, as there is no threat. When there is no threat, the application space automatically sets the USV fleet into an evenly spaced circular formation and rotates them about the centroid at a constant rate.

The response of each parameter in the cluster space for the run is shown in Fig. 9 and an overhead view is shown in Fig. 10, with initial positions marked by small shapes and the final positions marked by larger shapes. In all overhead view figures shown in this work, the positions of the kayaks are displayed relative to the cluster centroid. This removes any confusion caused in history trails by the cluster translating in the global frame. It can be seen from the graphs that the controller is capable of compensating for dynamics added by the environment including wind, currents, and boat wakes. Table A summarizes the rms errors for the controlled radial and inter-robot spacing parameters; all rms errors are under 4 meters, which we consider to be outstanding given the limited sensor performance and disturbance environment.

B. Shielding while changing size

Similar to the first case, in this scenario the fleet of USVs is rotating at a constant rate around a simulated protected object. Due to changing conditions or in the case of protecting multiple objects, it may be desirable to modify the size of the cluster. In Fig. 11, the cluster variables show the constant rotation and varying radius. Note that the fence spacing is automatically controlled by the application layer to maintain a uniform distribution around the protected object when no threat is present, as this case specifies. Fig. 12 shows an overhead view of the outward spiral maneuver, which is a portion of the test run shown in the preceding figure. Table B summarizes the rms errors of the

controlled radius and spacing parameters; again, excellent results are shown, with all rms errors under 3 meters.

C. Threat detection

The third experimental run demonstrates a case of shielding upon detection of a threat. In this instance the standard radius is set to 17m, the maximum approach is 25m, and the minimum fence spacing is set to 10m.

The overhead view in Fig. 13 shows the threat approaching the protected vessel. As the threat is identified, the cluster begins to rotate between the threat and the vessel. As the threat nears the fence spacing closes further. At this point the threat has been deterred and decided to turn around. In Fig. 14, the individual cluster space variables are shown for a longer portion of this scenario. The later part of the experiment shows the kayaks returning to an evenly spaced rotation about the protected asset as the threat disappears. Table C shows the rms errors to be less than 4 meters.

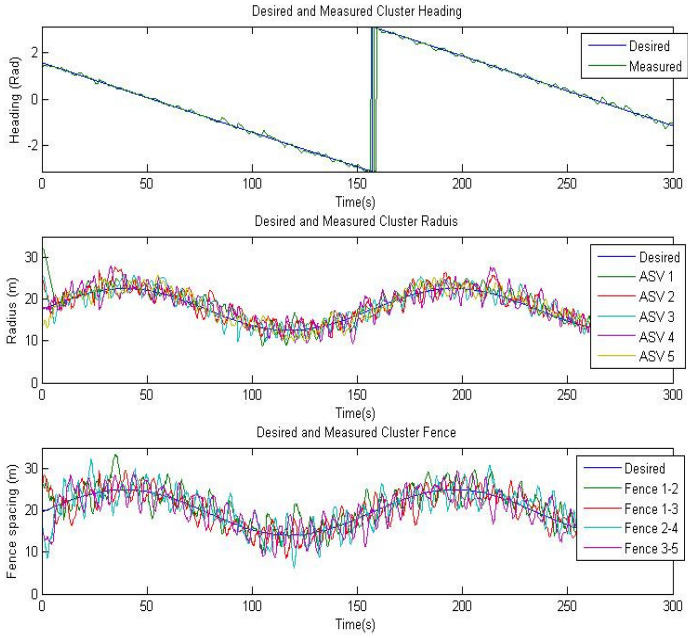


Fig. 11 – Basic shielding cluster variables. Note that the jump in the top plot is caused as the heading wraps from –pi to pi at 180 degrees, and is not an actual discontinuity. RMSE shown in table

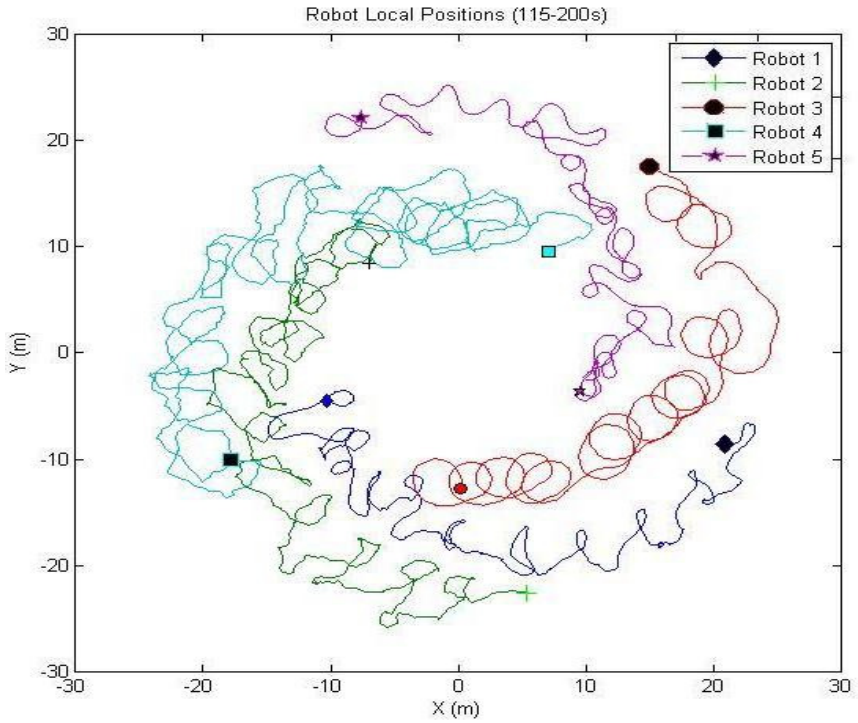


Fig. 12 – Overhead view of a change in shielding radius.

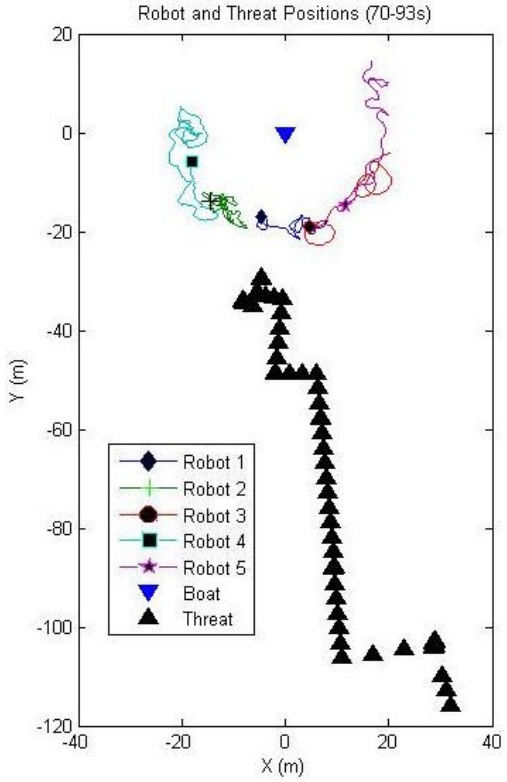


Fig. 13 – Overhead view of shielding technique with threat detection.

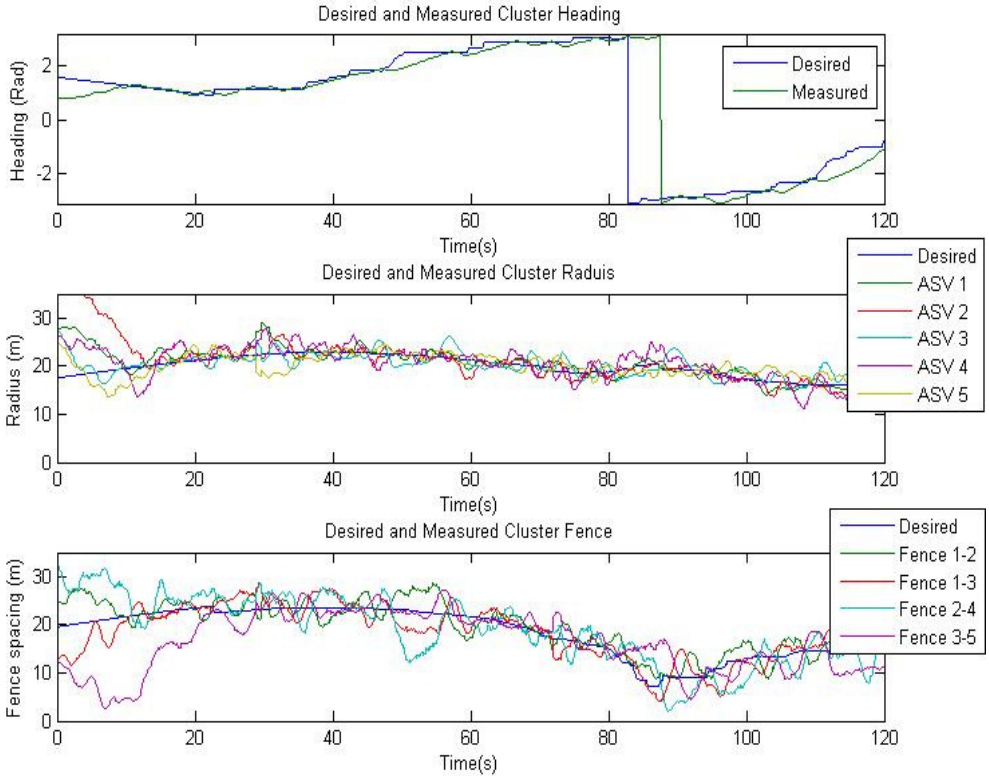


Fig. 14 – Shielding with threat detection cluster space variable. Table showing RMSE does not include the initialization time from 0-20

TABLE B – SHIELDING WHILE CHANGING SIZE: RMS ERROR VALUES FOR THE CLUSTER RADIUS AND FENCE SPACING VARIABLES

Cluster Radii	RMS Error (m)		Cluster Fence Spacing	RMS Errors (m)
R1	2.02		--	--
R2	1.84		F2	2.60
R3	1.79		F3	2.25
R4	2.07		F4	2.88
R5	1.55		F5	2.82

TABLE C–THREAT DETECTION: RMS ERROR VALUES FOR THE CLUSTER RADIUS AND FENCE SPACING VARIABLES

Cluster Radii	RMS Error (m)		Cluster Fence Spacings	RMS Errors (m)
R1	1.32		--	--
R2	1.32		F2	2.64
R3	1.79		F3	2.48
R4	1.81		F4	3.70
R5	1.64		F5	2.97

D. Shielding a mapping vessel

While the previous cases have relied on a simulated cluster centroid, the fourth case uses an actual vessel to demonstrate shielding with threat detection (Fig. 15). The protected vessel is another autonomous surface vessel, a SWATH (small waterplane area twin hull) boat, equipped with a multibeam sonar, AHRS, GPS, and heave sensors designed for shallow water bathymetry. Standard operation typically involves following a preset path (mowing the lawn) to map the desired area. More information can be found in [35]. This case uses four robots for the shielding fleet, using an appropriately modified set of kinematic transforms. We note that the application specifications remain the same, independent of the fact that only four robots are now being used.

Fig. 15 – Shielding with threat detection of a mapping vessel

The application variables for this case are set with the standard radius at 12 m, the maximum approach at 20 m, and the minimum fence spacing at 10 m.

The overhead view, shown in Fig. 16, is broken down into four time steps. In the first step the fleet of four USVs have identified a threat (out of frame to the northeast) and the cluster has rotated to face it. For this four USV case, the cluster heading is

aligned between robots 1 and 2. The fleet has not yet adjusted fence spacing or radius since the threat is still far away.

In step 2, the threat approaches the protected vessel. The kayaks begin to noticeably decrease the fence spacing. At step 3 the threat has continued to approach. The USVs are still tracking along the heading, have come further out and are narrowing the fence spacing.

At step 4 the threat has almost reached the max approach and the USVs have set the fence spacing near the minimum value as set in the application space. The kayaks loiter in these locations, tracking the heading and distance of the threat until it vacates the area.

The individual measured cluster variables are shown in Fig. 17. Table D shows the rms errors for the controlled parameters; as before, all errors are under 4 meters.

V. Ongoing and Future Work

Ongoing work on this project includes a significant level of Matlab/Simulink-based simulation in order to explore alternate implementations of the cluster space controller, using different shape variables. It is worth noting that the version reported on here fits within the leader-follower paradigm; other versions being explored clearly do not, such as defining a fleet centroid and using this as a reference for the center of the barrier. We are also preparing to use a version of this controller during a real-world Summer 2011 mission involving protection of an underwater robot dive area in Lake Tahoe; recreational boaters pose an extreme hazard to these operations given the ability of a boat to catch the high-voltage tether running from the tender boat to the robot.

In general, we continue to apply the cluster space control approach to systems with more robots and additional degrees of freedom in order to explore scalability issues. We are also working to generalize the application-space-to-cluster-space transform architecture by using this specification approach with other applications. Related to this, we plan to integrate our anomaly management algorithms [36] in to the overall multi-robot control system so that the system seamlessly adapts itself in the event of robot faults. Finally, we continue to apply the cluster space control framework to real-world applications, such as our previously mentioned work in gradient-based environmental sensing and reconfigurable sparse communication antenna arrays.

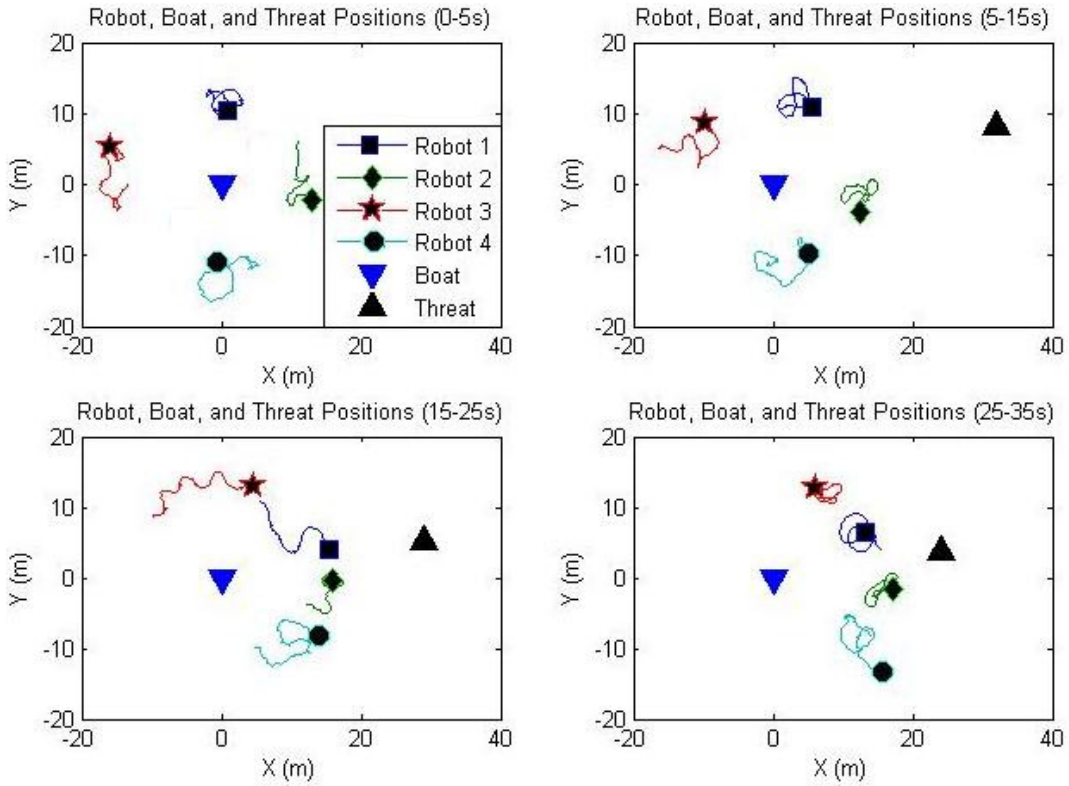


Fig. 16 – Overhead view of shielding technique with threat detection around mapping vessel.

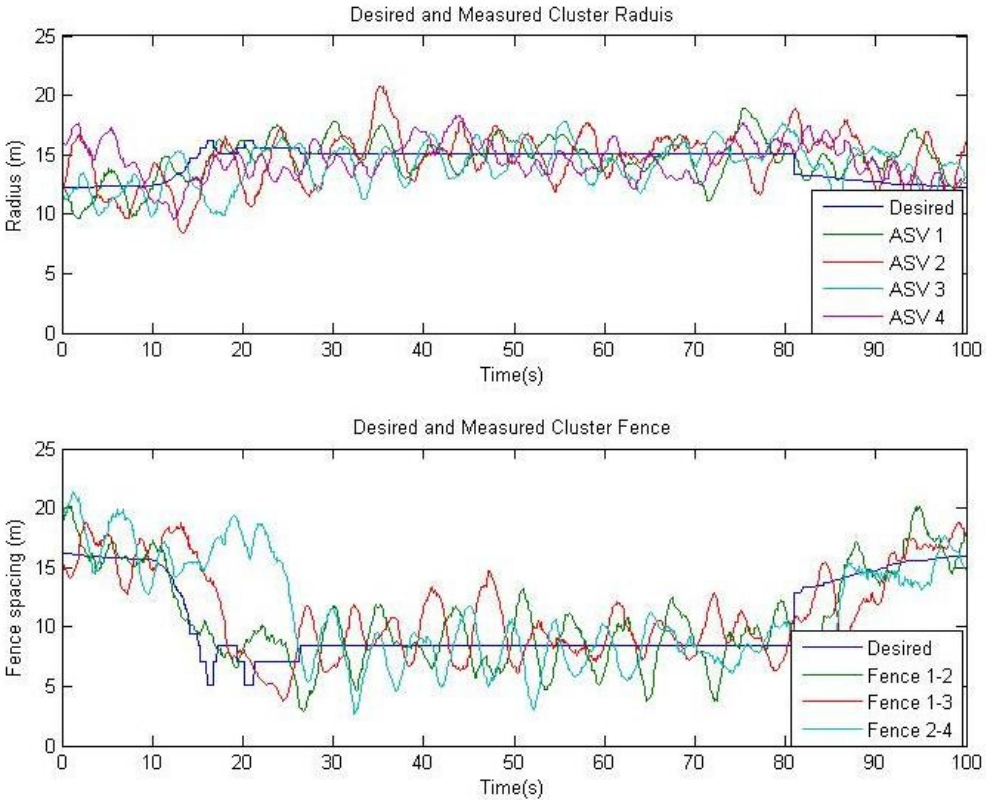


Fig. 17 – Cluster variables shielding with threat detection of a mapping vessel

TABLE D – SHIELDING A MAPPING VESSEL: RMS ERROR VALUES FOR THE CLUSTER RADIUS AND FENCE SPACING VARIABLES

Cluster Radii	RMS Error (m)		Cluster Fence Spacings	RMS Errors (m)
R1	1.58		--	--
R2	2.21		F2	2.33
R3	1.80		F3	2.56
R4	1.90		F4	3.99

VI. SUMMARY AND CONCLUSIONS

In this paper we described the use of a fleet of robotic marine vessels capable of guarding critical assets from threats. Coordinated formation control of the fleet was implemented through the use of the cluster space control architecture. An application-

specific layer was integrated with the cluster space controller, allowing an operator to directly specify and monitor guarding-related parameters.

This system has been experimentally verified in the field with a fleet of robotic kayaks. The control architecture used to establish the guarding behavior and the design of the robotic kayaks were reviewed, and experimental data regarding the functionality and performance of the system was presented. As a result, the five-robot cluster space definition and control architecture was validated and functionality was proven for this application.

Acknowledgment

The authors thank student researchers who assisted with performing experiments and who have contributed to the development of the cluster space approach, to include authors listed in [29]-[36]. This work has benefited greatly through the feedback of numerous colleagues; the authors are particularly indebted to Thomas Ademek, Ketan Rasal and the RSL graduate research team for their time and effort which were critical to this work.

REFERENCES

1. S.J. Corfield and J.M.Young. Unmanned surface vehicles – game changing technology for naval operations. In Advances in unmanned marine systems. Ed. G.N. Roberts and R. Sutton. IEEE, Hertfordshire, 2006, pp. 311-328.

2. S. Saunders, Ed. Mine warfare forces. In Janes Fighting Ships, IHS, London, 2004, pp. 177-18.

3. V. Bertram, Unmanned surface vehicles - A survey. In Skibsteknisk Selskab, Copenhagen, Denmark, 2008.

4. J. Manley, "Unmanned surface vehicles, 15 years of development," *Proceedings of MTS/IEEE OCEANS, Kobe, Japan*, September 2008, pp.1-4.

5. M. Caccia, “Autonomous Surface Craft: prototypes and basic research issues.” *14th Mediterranean Conf on Control and Automation, Ancona*, June 2006, pp. 1-6.

6. C. Kitts and M. Egerstedt, “Design, Control and Applications of Real-World Multirobot Systems.” *IEEE Robotics and Automation Magazine*, v 15, n 1, March 2008, p. 8.

7. J. Curcio, J. Leonard, and A. Patrikalakis. "SCOUT - a low cost autonomous surface platform for research in cooperative autonomy," *Proceedings of MTS/IEEE OCEANS*, vol 1, 2005, pp. 725- 729.

8. J. Curcio, J. Leonard, J. Vaganay, A. Patrikalakis, A. Bahr, D. Battle, H. Schmidt, M. Grund, “Experiments in Moving Baseline Navigation using Autonomous Surface Craft”, *Proceedings of MTS/IEEE Oceans*, vol 1, 2005, pp. 730-735.

9. J. Dolan, G. Podnar, S. Stancliff, K. Low, A. Elfes, J. Higinbotham, J. Hosler, T. Moisan, J. Moisan. “Cooperative aquatic sensing using the telesupervised ocean sensor fleet.” *Proceedings of Remote Sensing of the Ocean, Sea Ice, and Large Water Regions*, vol 7473, 2009, pp. 1-12.

10. J. Esposito, M. Feemster, and E. Smith, "Cooperative manipulation on the water using a swarm of autonomous tugboats," *IEEE International Conference on Robotics and Automation*, May 2008, pp.1501-1506.

11. L. Elkins, D. Sellers, and W.R. Monach. “The Autonomous Maritime Navigation (AMN) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles.” *Journal of Field Robotics*, vol 27, 2010, pp. 790–818.

12. Z.Z. Abidin, M. Arshad, U. Ngah, O. Ping; "Control of mini autonomous surface vessel," *Proceedings of MTS/IEEE OCEANS, Sydney*, May 2010, pp.1-4.

13. Protei, Online at https://sites.google.com/a/opensailing.net/protei/

14. J.A. Curcio, P. McGillivary, K. Fall, A. Maffei, K. Schwehr, B. Twiggs, C. Kitts, P. Ballou. "Self-Positioning Smart Buoys, The "Un-Buoy" Solution: Logistic Considerations using Autonomous Surface Craft Technology and Improved Communications Infrastructure," *Proceedings of MTS/IEEE OCEANS,* September 2006, pp.1-5.

15. Survey of Portable Range Technologies, U.S. Army White Sands Missile Range Special Report, 2005. Available at http://www.jcte.jcs.mil/RCC

16. E. Fiorelli, N. Leonard, P. Bhatta, D. Paley, R. Bachmayer, D. Fratantoni. “Multi-AUV control and adaptive sampling in Monterey Bay,” *IEEE Journal of Oceanic Engineering*, v 31, n 4, October 2006, pp. 935-948.

17. Y. Tan and B. Bishop, “Evaluation of robot swarm control methods for underwater mine countermeasures,” *Proceedings of the 2004 Annual Southeastern Symposium on System Theory*, v 36, pp. 294-298

18. T. Balch and R. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, vol14, no 6, December 1998, pp. 926-939.

19. E. Flinn, “Testing for the ‘boids’,” *Aerospace America*, v 43, n 6, June, 2005, pp. 28-29.

20. N. E. Leonard and E. Fiorelli, “Virtual leaders, artificial potentials and coordinated control of groups,” *Proceedings IEEE Conf. Decision and Control*, 2001, pp. 2968–2973.

21. K. Tan, and M. Lewis, “Virtual Structures for High-Precision Cooperative Mobile Robotic Control,” *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996, pp. 132-139.

22. C.P. Tang, R.M. Bhatt, M.Abou-Samah, and V. Krovi, “Screw-theoretic analysis framework for cooperative payload transport by mobile manipulator collectives,” *IEEE/ASME Transactions on Mechatronics*, vol 11, no 2, April 2006, pp. 169-178.

23. C. A. Kitts and I. Mas, “Cluster space specification and control of mobile multirobot systems,” *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, April 2009, pp. 207–218.

24. G.L. Mariottini, F. Morbidi, D. Prattichizzo, N. Vander Valk, N. Michael, G. Pappas, and K. Daniilidis. Vision-based localization for leader-follower formation control. *IEEE Transactions on Robotics*, vol 25, no 6, December 2009, pp. 1431 -1438.

25. I. Mas and C. Kitts. “Centralized and decentralized multi-robot control methods using the cluster space control framework.” *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Montreal, Canada*, July 2010, pp. 1-8.

26. M. Meserve. Multirate Control Applied to a Cluster Space Robot Formation, Santa Clara University Dept. of Electrical Engineering Masters Thesis, March 2011.

27. I. Mas and C. Kitts. “Model-Based Nonlinear Cluster Space Control of Mobile Robot Formations.” Multi-Robot Systems, Trends and Development, T. Yasuda (Ed), INTECH, Ch 4.

28. I. Mas, S. Li, J. Acain, and C. Kitts. “Entrapment/escorting and patrolling missions in multi-robot cluster space control.” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO*, Oct 2009, pp. 5855-5861.

29. P. Mahacek, I. Mas, O. Petrovic, J. Acain, and C. Kitts. “Cluster space control of autonomous surface vessels.” *Marine Technology Society Journal*, v 43 n 1, 2009, pp. 13-20

30. P. Mahacek, I. Mas, C. Kitts, *"Cluster space control of autonomous surface vessels utilizing obstacle avoidance and shielding techniques,"* Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES , vol., no., pp.1-5, 1-3 Sept. 2010

31. T. Adamek, Cluster Space Gradient Contour Tracking for Mobile Multi-Robot Systems. Santa Clara University Dept. of Mechanical Engineering Masters Thesis, January 2011.

32. V. Howard, A Study of Gradient Climbing Techniques Using Cluster Space Control of Multi-Robot Systems, Santa Clara University Dept. of Mechanical Engineering Masters Thesis, March 2011.

33. G. Okamoto, C. Chen, and C. Kitts. “Beamforming Performance for a Reconfigurable Sparse Array Smart Antenna System via Multi-Robot Control.” *Proceedings of the SPIE Defense, Security, and Sensing Conference, Orlando FL*, April 2010, pp. 1-11.

34. I. Mas, Obstacle Avoidance Policies for Cluster Space Control of Non-Holonomic Multi-Robot Systems. *IEEE/ASME Transactions on Mechatronics,* In Press.

35. E. Beck, W. Kirkwood, D. Caress, T. Berk, P. Mahacek, K. Brashem, J. Acain, V. Reddy, C. Kitts, J. Skutnik, G. Wheat. "SeaWASP: A Small Waterplane Area Twin Hull Autonomous Platform for Shallow Water Mapping." *Marine Technology Society Journal*, vol 43 no 1, 2009, pp. 6-12.

36. C. Kitts. "Managing space system anomalies using first principles reasoning." *IEEE Robotics Automation Magazine*, vol 13 no 4, December 2006, pp. 39 –50.

Cluster Space Control of Autonomous Surface Vessels Utilizing Obstacle Avoidance and Shielding Techniques

Paul Mahacek Ignacio Mas Dr. Christopher Kitts
Santa Clara University Department of Mechanical Engineering
500 El Camino Real
Santa Clara, Ca. 95053

Abstract- **Multi-robot systems offer many advantages over a single robot system including redundancy, coverage and flexibility. One of the key technical challenges in fielding multi-robot systems for real-world applications is the coordination and relative motion control of the individual units. The cluster space control technique addresses the motion control challenge by providing formation control and promoting the simplified specification and monitoring of the motion of mobile multi-robot systems. Previous work has established this approach and has experimentally verified its use for dynamic marine surface vessels consisting of 2 or 3 robots and with varying implementations ranging from automated cluster trajectory control to human-in-the-loop piloting. In this research program, we apply the cluster space control technique to a larger group of marine vessels and include both obstacle avoidance and threat detection with shielding formations. The resulting system is capable of autonomous navigation utilizing a centralized controller, currently implemented via a shore-based computer, that wirelessly receives ASV data and relays control commands. Using the cluster space control approach, these control commands allow a cluster supervisor to oversee a flexible and mobile perimeter formed by the ASV cluster or to detect a threat and establish a shield between the operation and the threat. Theoretical formulation and simulation results demonstrating these capabilities are provided, and plans for future work are discussed.**

I. INTRODUCTION

Robotic systems offer many advantages to accomplishing a wide variety of tasks given their strength, speed, precision, repeatability, and ability to withstand extreme environments. While most robots perform these tasks in an isolated manner, interest is growing in the use of tightly interacting multi-robot systems to improve performance in

current applications and to enable new capabilities. In this application the robots are Autonomous Surface Vessels (ASVs). Creating a multi-ASV or any multi-boat cluster has many potential advantages including redundancy, increased coverage and throughput, flexible reconfigurability and spatially diverse functionality.

For mobile systems, one of the key technical considerations is the coordination of the motions of the individual vehicles. Many techniques have been and continue to be explored. Because of the physical distribution of components and the potential for limited information exchange, decentralized control approaches hold great promise [1], and these techniques have been explored for a variety of systems. Our work, explores a specific centralized approach for potential application to robot clusters of limited size and scope with the understanding that other control modes may be required for expansion to achieve higher performance (vehicles on the order of 1-10 units and several miles range) [2].

A. Cluster Space Approach

The motivation of the cluster space [3] approach is to promote the simple specification and monitoring of the motion of a mobile multi-robot system. This strategy conceptualizes the n-robot system as a single entity, a cluster, and desired motions are specified as a function of cluster attributes, such as position, orientation, and geometry. These attributes guide the selection of a set of independent system state variables suitable for specification, control, and monitoring. These state variables form the system's cluster space. Cluster space state variables may be related to robot-specific state variables, actuator state variables, etc. through a formal set of kinematic transforms. These transforms allow cluster commands to be converted to robot specific commands, and for sensed robot-specific state data to be converted to cluster space state data. As a result, a

supervisory operator or real-time pilot can specify and monitor system motion from the cluster perspective. Our hypothesis is that such interaction enhances usability by offering a level of control abstraction above the robot and actuator-specific implementation details [4-8].

B. Multi- robot Obstacle Avoidance

For any mulit-robot formation control strategy, avoiding collisions with obstacles and with other members of the formation is critical. For cases where the environment is well known and predictable, a preset path can be used. But most environments are dynamic and unknown. Here we utilize a continuous collision algorithm as presented in references [9], [10] and [11]. While this technique can be applied at both the cluster level and the individual robot level, in this work it is only applied at the robot level.

C. The ASVs

The design of the vessels emphasizes versatility, ease of operation, and low cost in all design segments. The use of common bus architecture across all Robotic Systems Lab robotic vehicles enables a rapidly reproducible control system capable of transparently controlling multiple platforms, including several different types of land rovers, an aerial vehicle, and two different types of ASVs. Off the shelf parts, like a readily available kayak requiring no permanent modifications, facilitate both ease of integration and quick replacement if the need arises.

Minor upgrades and modifications have been made to the ASVs including a new deployment and transportation system as well as structural upgrades decreasing the required time for setup and deployment. The details of the control hardware, protocol, propulsion, and power subsystems can be found in previous publications such as [2].

II. THE CONTROLLER

The motivation of this research is to promote the simple specification and monitoring of the motion of a mobile, multi-robot system. Our vision is to enable automated, formalized execution of operator directives based on information such as "Drive North at 5 m/sec in a side-by-side line with a 25 m separation," or "Translate and rotate based on joystick inputs while decreasing the lateral size of the formation." In general, enabling features of a system providing such conceptual level specification include flexibility in the choice of specifications made to define the desired motion and the judicious selection of default values appropriate to the system design and application.

A. Cluster Space State Variables

Fig. 1 depicts the reference frames for the planar 3-ASV problem. To complement the sensor data used in experimentation, the global frame conventions were selected as follows: Y_G points north, X_G points east. The cluster frame is located at (X_0, Y_0) and its orientation is given by θ_1 which is the angle about robot 0 from Y_G to the location of robot 1 as shown below. Robots 1, 2 and 3 are defined by a radial distances, R_1, R_2 and R_3 from robot 0. Robots 2 and 3 are each also defined by a fence spacing from robot 1 given as F_2 and F_3. Further expansion would be added in an even-odd pattern with robot 4 and 5 referenced radially from robot 0. Robot 4 would be fence spaced from robot 2, while robot 5 would be fence spaced from robot 3.

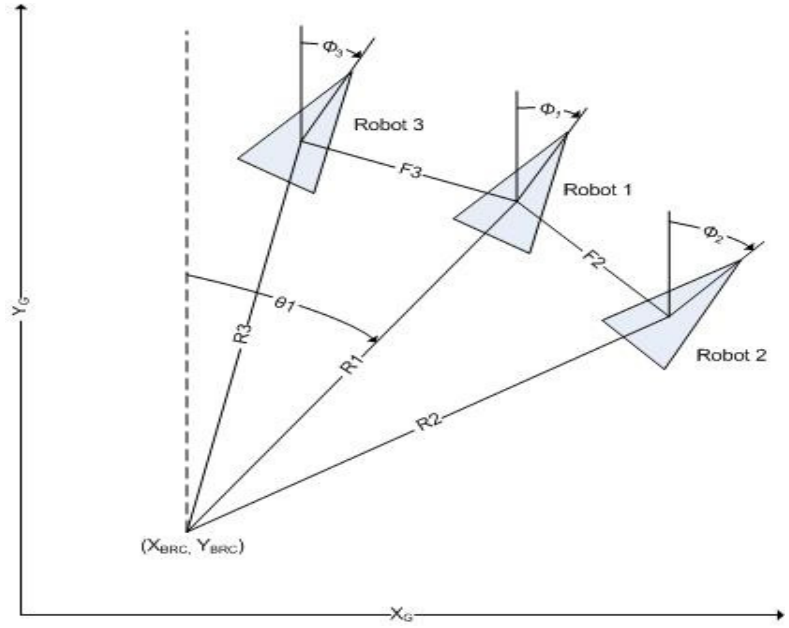


Fig. 1. Cluster and robot

B. Kinematic Equations

The forward position kinematics are a set of equations the allow the transformation from robot variables, X_0, Y_0, φ_0, X_1, Y_1, φ_1, X_2, Y_2, φ_2, X_3, Y_3, and φ_3, to cluster variables, X_{brc}, Y_{brc}, φ_{brc}, R_1, θ_1, φ_{1c}, R_2, F_2, φ_{2c}, R_3, F_3, and φ_{3c}. The inverse kinematics allow the cluster variables to be changed back into the robot variables. The equations for the forward kinematics can be seen in (1-6). And the inverse are shown in (7-12). Note that the equations for X_0, Y_0, φ_0, φ_1, φ_2, and φ_3 trivially correspond directly to their respective cluster variables for the selected cluster definition and are not included.

$$R_1=((X_1-X_0)^2+(Y_1-Y_0)^2)^{1/2} \quad (1)$$

$$\theta_1=Atan2((X_1-X_0),(Y_1-Y_0)) \quad (2)$$

$$R_2=((X_2-X_0)^2+(Y_2-Y_0)^2)^{1/2} \quad (3)$$

$$F_2=((X_2-X_1)^2+(Y_2-Y_1)^2)^{1/2} \quad (4)$$

$$R_3=((X_3-X_0)^2+(Y_3-Y_0)^2)^{1/2} \quad (5)$$

$$F_3=((X_3-X_1)^2+(Y_3-Y_1)^2)^{1/2} \quad (6)$$

$$X_1=X_{brc}+R_1 * Sin(\theta_1) \quad (7)$$

$$Y_1=Y_{brc}+R_1 * Cos(\theta_1) \quad (8)$$

$$X_2=X_{brc}+R_2 * Sin(\theta_1+acos((R_1^2+ R_2^2- F_2^2)/2 * R_1 * R_2)) \quad (9)$$

$$Y_2=Y_{brc}+R_2 * Cos(\theta_1+acos((R_1^2+ R_2^2- F_2^2)/2 * R_1 * R_2)) \quad (10)$$

$$X_3=X_{brc}+R_3 * Sin(\theta_1+acos((R_1^2+ R_3^2- F_3^2)/2 * R_1 * R_3)) \quad (11)$$

$$Y_3=Y_{brc}+R_3 * Cos(\theta_1+acos((R_1^2+ R_3^2- F_3^2)/2 * R_1 * R_3)) \quad (12)$$

C. Control Framework

Fig. 2 presents the control architecture for trajectory based cluster space control of an *n*-robot system. A cluster level PID controller compares cluster position and velocity with desired trajectory values and outputs cluster commanded velocities, which are translated into individual ASV velocities through the inverse Jacobian. Data from the ASVs are converted to cluster space information through the forward kinematics and Jacobian and fed back into the controller. The non-holonomic constraint given by the differential drive motion of the ASVs effectively reduces each ASV from three degrees of freedom down to two. For the cluster of ASVs it becomes a six DOF system. As a consequence, an inner-loop ASV level heading control is needed on each ASV and the cluster space controller does not regulate the cluster parameters corresponding to the yaw orientation of the ASV relative to the cluster.

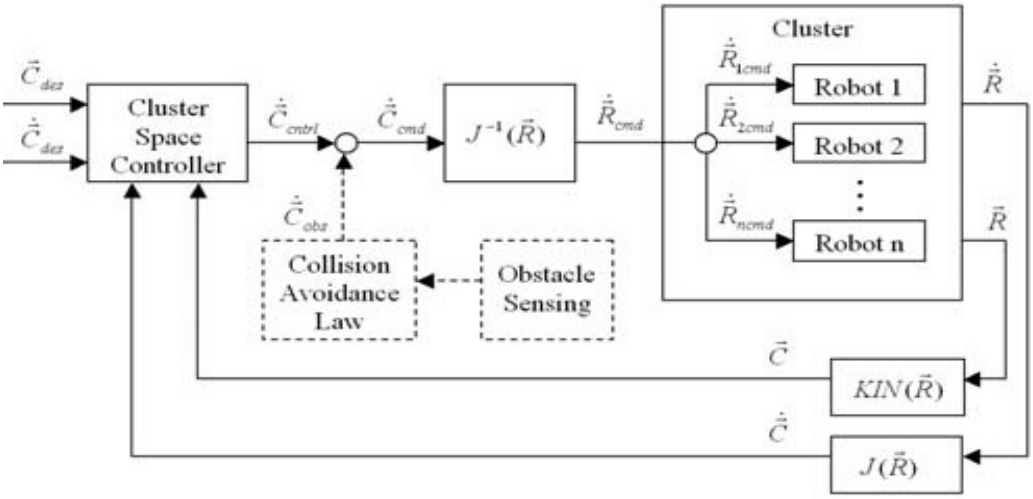


Fig. 2. Cluster Space Control Architecture for an *n*-Robot system

D. Obstacle Avoidance

Different approaches can be used to avoid collisions with obstacles or other robots in the formation depending on the nature of the task. When keeping the formation at all times is not a priority a robot level obstacle avoidance algorithm can be utilized. In this case, the ASVs negotiate obstacles in an independent fashion, momentary breaking away from the formation. Once the obstacle has been avoided the ASV will return to the cluster. This algorithm is input the position of the obstacle and feeds an offset after the cluster space controller as shown in Fig. 2.

For each ASV, the detection radius and the avoidance potential can be set. The function as shown in [8] has a zero value outside the detection radius, and an infinite at the minimum circle enveloping the ASV. Typically circles are used, but for a non-circular shape, like a ship, an elongated oval can be defined to better model the outer edge of the vessel.

E. Application Space and Shielding

Application Space is an additional layer between the cluster space controller and the user interface. It is a more versatile way to modify the interaction of the user with the cluster space variables.

III. SIMULATION AND RESULTS

The main objective of this research was to validate the cluster space control architecture for a multi-ASV System while utilizing obstacle avoidance and to determine the viability of a new shielding technique applied in application space. Two main cases were run, basic shielding, and shielding with threat detection.

A. Basic Shielding

In the case of the basic shielding technique we are applying it to a main vessel deploying an ROV. Using the application space, the operator can set the standard shield radius, the maximum approach and the minimum fence spacing. In this first instance the standard shield radius is set to 17m, the minimum fence spacing and approach are disregarded as there is no threat. When there is no threat, the application space automatically sets the ASVs into an evenly spaced formation and rotates them about the boat and ROV centroid as shown in Fig. 3.

With the random selection of the ASV starting points the initialization might cause them to be on a path that would make them collide with the boat or cross over the ROV. The obstacle avoidance algorithm successfully pushes each of the ASVs away from the boat and rov, avoiding any potential collisions.

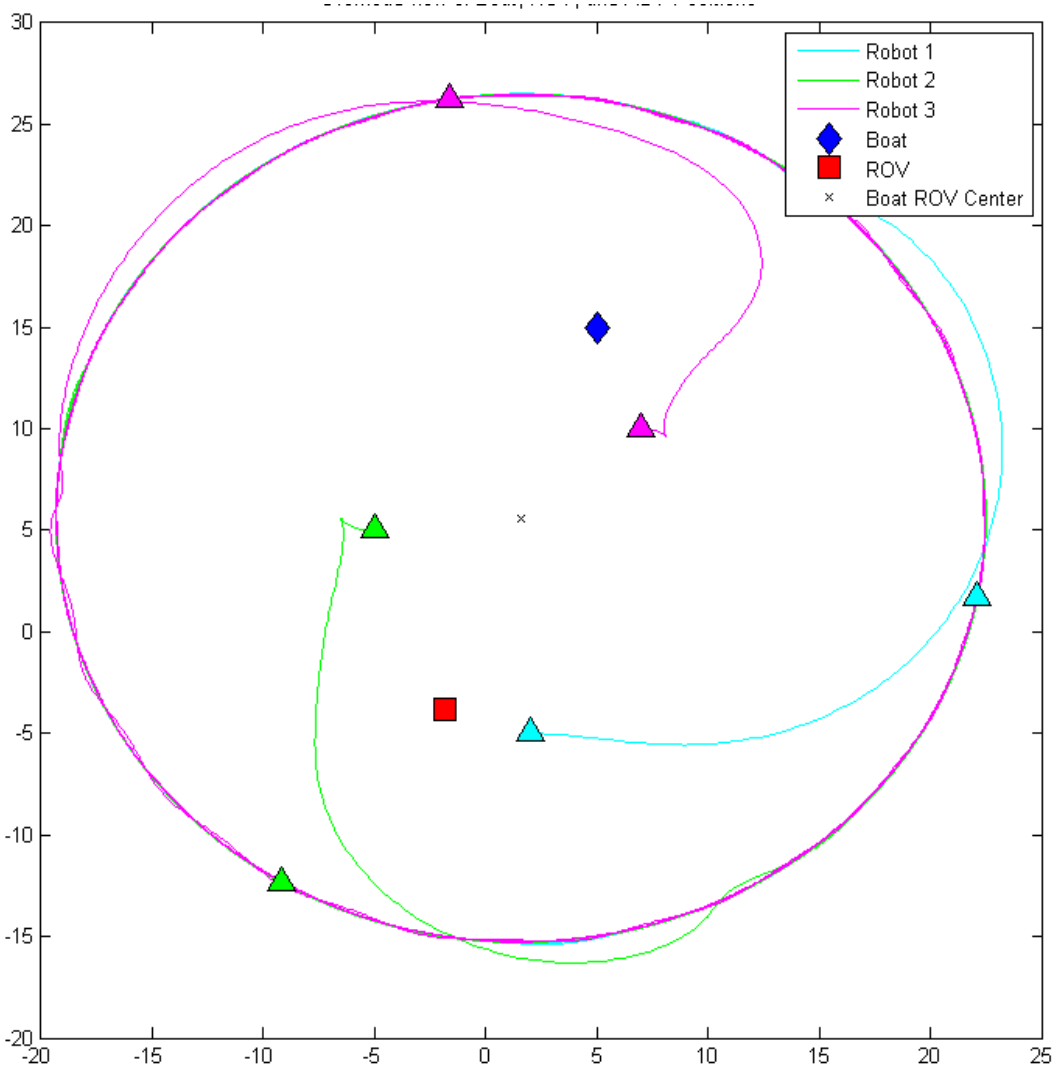


Fig. 3. Standard shielding, no threat.

The response of each parameter in the cluster space for the run shown in Fig. 3. is shown in Fig. 4. It can be seen from the graphs that the controller is capable of compensating for dynamics added by the ASVs even when starting from random locations.

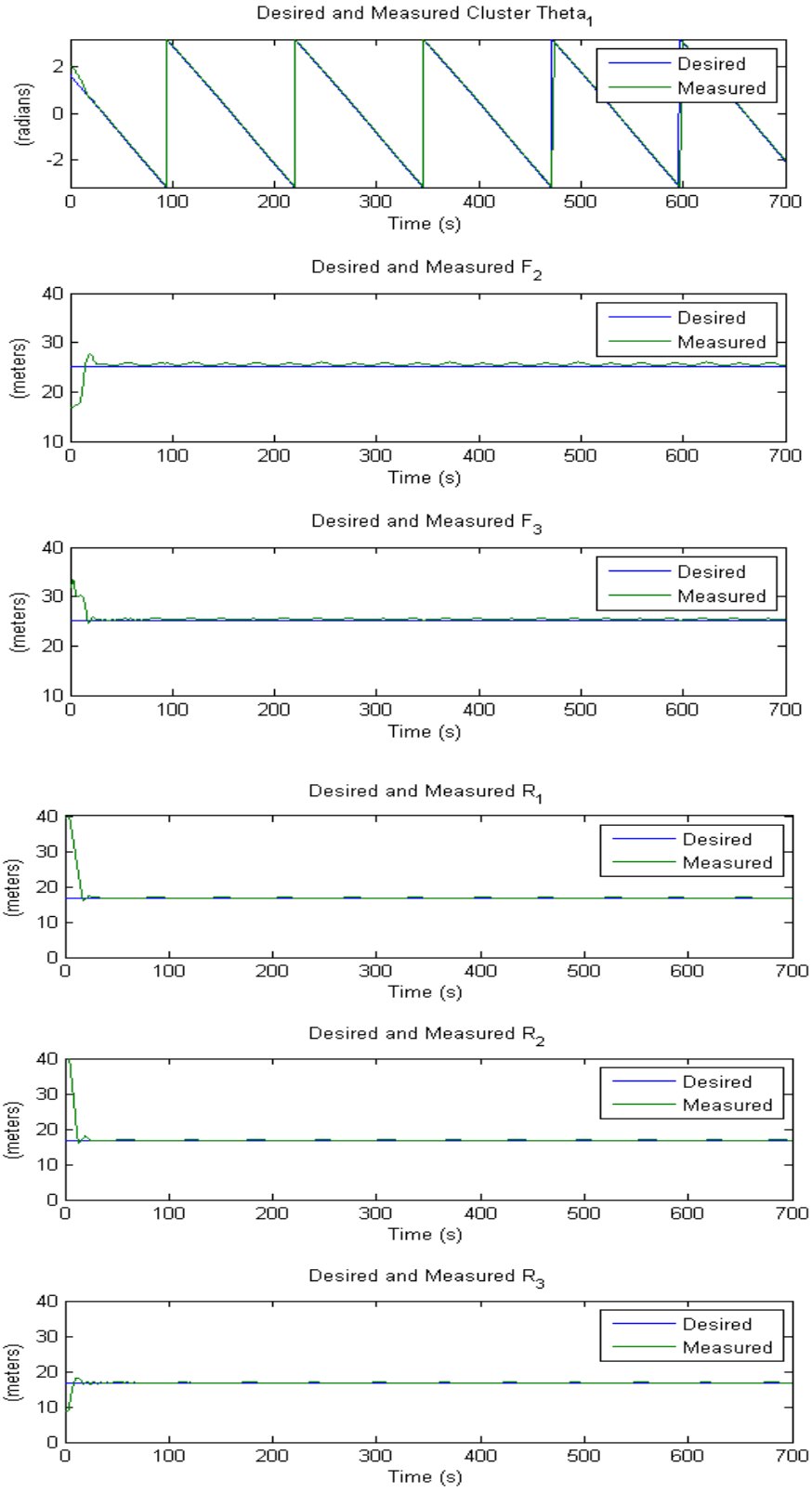


Fig. 4. Basic shielding cluster variables

A. Shielding with Threat Detection

An overhead view of the second case, shielding with threat detection, is shown in Fig. 5, and the individual components of the cluster space variables are shown in Fig. 6. In this instance the standard radius is set once again to 17m, the maximum approach is 50m, and the minimum fence spacing is set to 5m. The overhead view is broken down into five time segments. At step 1 the ASVs are in a standard formation around the boat and ROV centroid. The threat is identified and they rotate to position in between the threat and the boat and ROV centroid.

At step 2 the threat has continued to approach. The ASVs are still tracking along the heading, have come further out and are narrowing the fence spacing. At step 3 the threat has reached the max approach and the ASV have set the fence spacing to the minimum value as set in the application space. It can also be seen that the obstacle avoidance is playing a part, with ASV 1 backing away to avoid a collision with the threat. At step 4 the threat has begun to stand down and the ASVs reduce the radius and increase the fence spacing while still staying between the threat and the boat and the ROV. At step 5 the threat is outside the visible range. The ASVs have fully disengaged and resumed the basic shield around the boat and ROV.

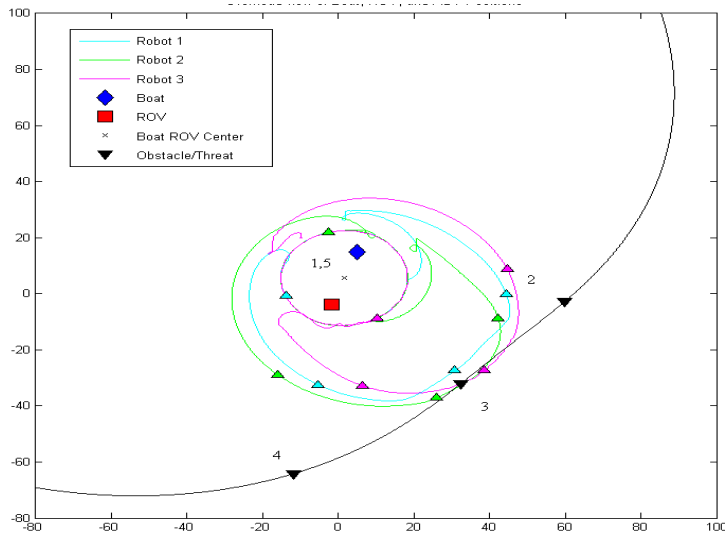


Fig. 5. Overhead view of shielding technique with threat detection.

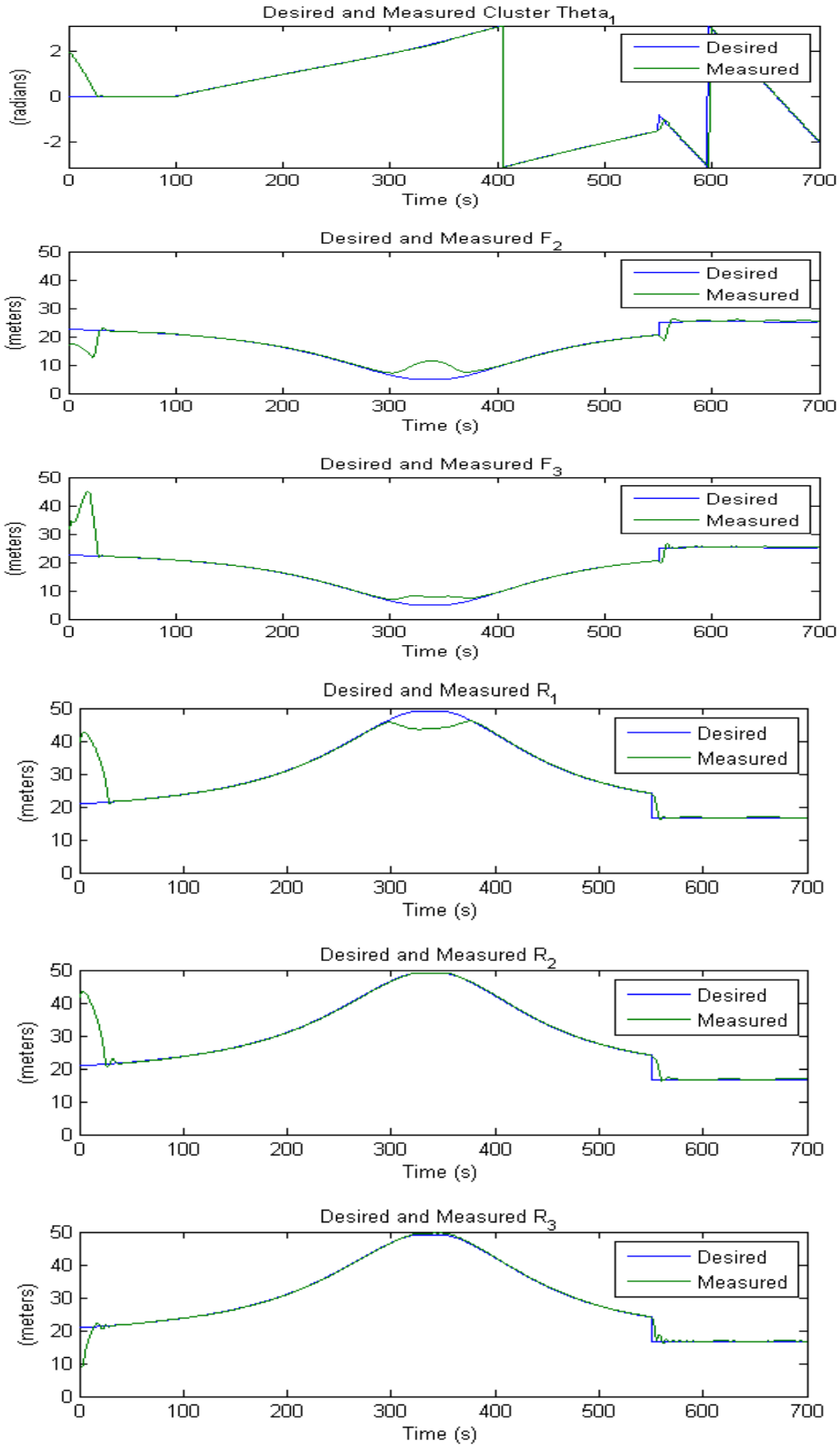


Fig. 6. Shielding with threat detection cluster space variable

IV.CONCLUSION

The cluster space state representation of mobile multi-robot systems was applied and evaluated for a three-ASV system as a means of specifying and controlling the desired mobility characteristics for surface vessels. Both obstacle avoidance algorithms and shielding techniques were successfully implemented and displayed. As a result, the three-robot cluster space definition and control architecture was validated and basic functionality was proven for this application.

Our ongoing and future work in this field is focused on enhancing motion control performance, increasing the number of vehicles, and integrating the motion-control oriented cluster space controller with application layer controllers. We believe that this will lead to enhanced performance for real-world marine applications as well as cost-effective improvements in operating such systems through the reduction of the operator/robot ratio required to control such systems. Field experimental tests of the simulations presented here are scheduled for the next few months.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Ryan Ishizu, Ray Chindaphorn, Tran To, Brian Tully and Mike Rasay, Matthew Kalkbrenner

REFERENCES

[1] D. Siljak, Decentralized Control of Complex Systems, *Academic* 1991.

[2] P. Mahacek, et. al. Cluster Space Control of Autonomous Surface Vessels. Marine Technology Society Journal, Volume 43, Number 1, pp 13-20.

[3] C. Kitts, and I. Mas, "Cluster Space Specification and Control of Mobile Multi-Robot Systems," Mechatronics, *IEEE/ASME Transactions on,* vol. 14, no.2, pp. 207-218, April 2009.

[4] R. Ishizu, The Design, Simulation and Implementation of Multi-Robot Collaborative Control from Cluster Perspective. Advisor: C. Kitts, *Santa Clara Univ. Master of Science in Elec. Eng. Thesis*, Dec 2005.

[5] P. Connolley, Design and Implementation of a Cluster Space Trajectory Controller for Multiple Holonomic Robots. Advisor: C. Kitts, *Santa Clara Univ. Master of Science in Mech. Eng. Thesis*, June 2006.

[6] T. To, Automated Cluster Space Trajectory Control of Two Non-Holonomic Robots. Advisor: C. Kitts, *Santa Clara Univ. Master of Science in Computer Eng. Thesis*, June 2006.

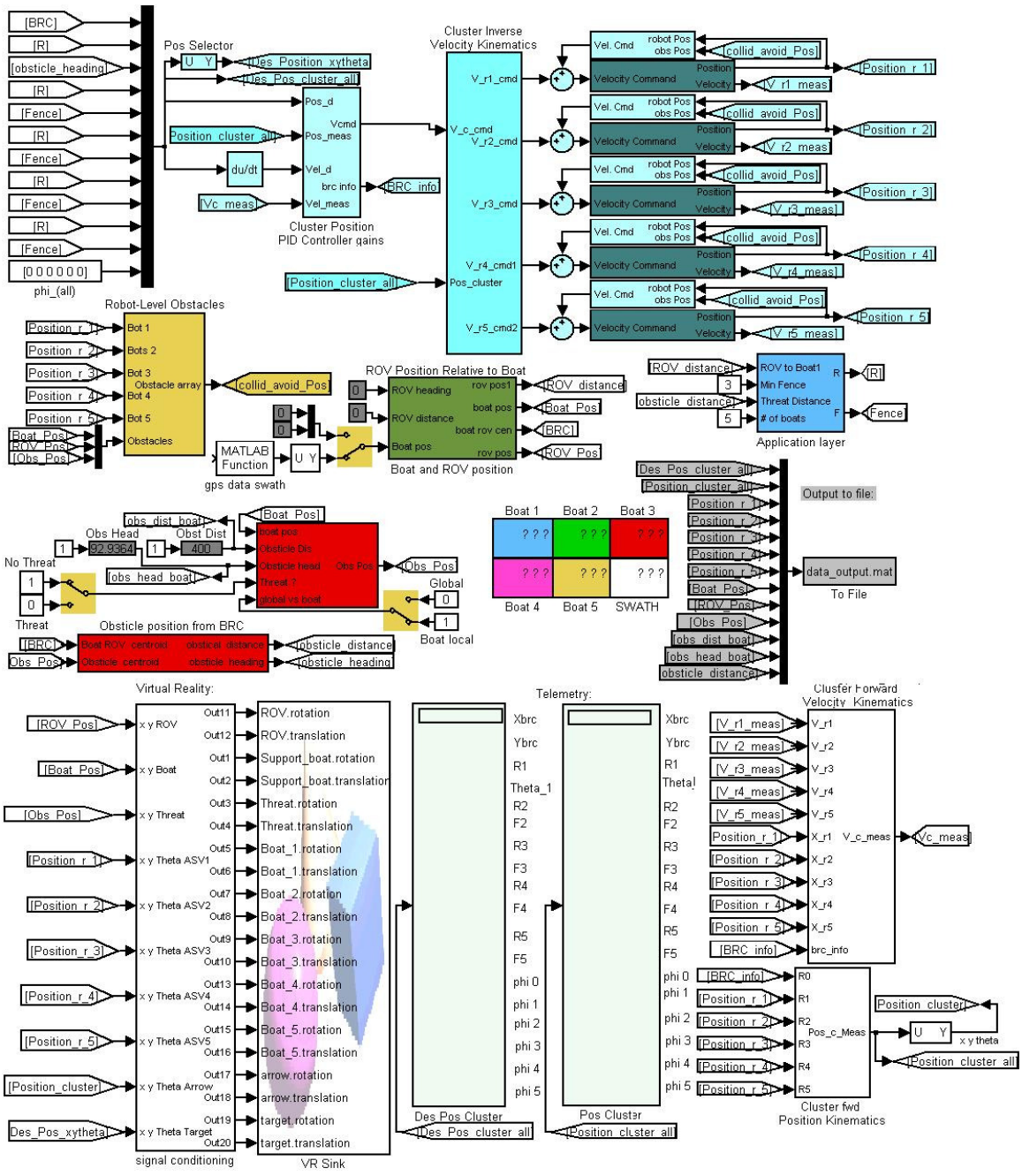
[7] M. Kalkbrenner, Design and Implementation of a Cluster Space Human Interface Controller,. . . Advisor: C. Kitts. *Santa Clara Univ. Master of Science in Mech. Eng. Thesis*, June 2006.

[8] P. Chindaphorn, Cluster Space Obstacle Avoidance for Two Non-Holonomic Robots. Advisor: C. Kitts. *Santa Clara University Master of Science in Computer Engineering Thesis*, June 2006.

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

[10] I. Mas, Obstacle Avoidance Policies for Cluster Space Control of Non-Holonomic Multi-Robot Systems. IEEE

[11] C. Kitts, K. Stanhouse, and P. Chindaphorn, "Cluster space collision avoidance for mobile two-robot systems," *Intelligent Robots and Systems. IEEE/RSJ International Conference on*, Oct 2009.


[BRC]
[R]
[obsticle_heading]
[R]
[Fence]
[R]
[Fence]
[R]
[Fence]
[R]
[Fence]
[0 0 0 0 0 0]
phi_(all)
Pos Selector
U Y
[Des_Position_xytheta]
[Des_Pos_cluster_all]
Position_cluster_all
du/dt
[Vc_meas]
Pos_d
Vcmd
Pos_meas
Vel_d
brc info
Vel_meas
[BRC_info]
Cluster Position
PID Controller gains
Cluster Inverse
Velocity Kinematics
V_r1_cmd
V_c_cmd
V_r2_cmd
V_r3_cmd
V_r4_cmd1
Pos_cluster
V_r5_cmd2
[Position_cluster_all]
Vel. Cmd
robot Pos
obs Pos
[collid_avoid_Pos]
Velocity Command
Position
Velocity
[Position_r_1]
[V_r1_meas]
[Position_r_2]
[V_r2_meas]
[Position_r_3]
[V_r3_meas]
[Position_r_4]
[V_r4_meas]
[Position_r_5]
[V_r5_meas]
Robot-Level Obstacles
[Position_r_1]
[Position_r_2]
[Position_r_3]
[Position_r_4]
[Position_r_5]
[Boat_Pos]
[ROV_Pos]
[Obs_Pos]
Bot 1
Bots 2
Bot 3
Obstacle array
Bot 4
Bot 5
Obstacles
[collid_avoid_Pos]
MATLAB Function
U Y
gps data swath
ROV Position Relative to Boat
ROV heading
ROV distance
Boat pos
rov pos1
boat pos
boat rov cen
rov pos
[ROV_distance]
[Boat_Pos]
[BRC]
[ROV_Pos]
Boat and ROV position
[ROV_distance]
obsticle_distance]
3
5
ROV to Boat1
Min Fence
Threat Distance
# of boats
R
F
[R]
Fence
Application layer
[obs_dist_boat]
[Boat_Pos]
Obs Head
Obst Dist
92.9364
400
Obsticle Dis
Obsticle head
Obs Pos
[obs_head_boat]
Threat ?
global vs boat
[Obs_Pos]
No Threat
Threat
Global
Boat local
Obsticle position from BRC
[BRC]
Obs_Pos
Boat ROV centroid
Obsticle centroid
obsticel distance
obsticle heading
[obsticle_distance]
[obsticle_heading]
Boat 1
Boat 2
Boat 3
Boat 4
Boat 5
SWATH
Des_Pos_cluster_all]
[Position_cluster_all]
[Position_r_1]
[Position_r_2]
[Position_r_3]
[Position_r_4]
[Position_r_5]
[Boat_Pos]
[ROV_Pos]
[Obs_Pos]
[obs_dist_boat]
[obs_head_boat]
obsticle_distance]
Output to file:
data_output.mat
To File
Virtual Reality:
[ROV_Pos]
[Boat_Pos]
[Obs_Pos]
[Position_r_1]
[Position_r_2]
[Position_r_3]
[Position_r_4]
[Position_r_5]
[Position_cluster]
[Des_Pos_xytheta]
x y ROV
x y Boat
x y Threat
x y Theta ASV1
x y Theta ASV2
x y Theta ASV3
x y Theta ASV4
x y Theta ASV5
x y Theta Arrow
x y Theta Target
Out11
Out12
Out1
Out2
Out3
Out4
Out5
Out6
Out7
Out8
Out9
Out10
Out13
Out14
Out15
Out16
Out17
Out18
Out19
Out20
ROV.rotation
ROV.translation
Support_boat.rotation
Support_boat.translation
Threat.rotation
Threat.translation
Boat_1.rotation
Boat_1.translation
Boat_2.rotation
Boat_2.translation
Boat_3.rotation
Boat_3.translation
Boat_4.rotation
Boat_4.translation
Boat_5.rotation
Boat_5.translation
arrow.rotation
arrow.translation
target.rotation
target.translation
signal conditioning
VR Sink
Telemetry:
Xbrc
Ybrc
R1
Theta_1
R2
F2
R3
F3
R4
F4
R5
F5
phi 0
phi 1
phi 2
phi 3
phi 4
phi 5
Des Pos Cluster
[Des_Pos_cluster_all]
Pos Cluster
[Position_cluster_all]
Clusfer Forward
Velocity Kinematics
[V_r1_meas]
[V_r2_meas]
[V_r3_meas]
[V_r4_meas]
[V_r5_meas]
[Position_r_1]
[Position_r_2]
[Position_r_3]
[Position_r_4]
[Position_r_5]
[BRC_info]
V_r1
V_r2
V_r3
V_r4
V_r5
X_r1
X_r2
X_r3
X_r4
X_r5
brc_info
V_c_meas
[Vc_meas]
Cluster fwd
Position Kinematics
R0
R1
R2
R3
R4
R5
Pos_c_Meas
Position_cluster]
U Y
x y theta
[Position_cluster_all]

```
function Output = five_robot_cen
troid_forward_kin_v1(u)
%This function computes the cluster position based on the robot
%positions.
%arguments:     u = [X_0 Y_0 X_1 Y_1 X_2 Y_2 X_3 Y_3 X_4 Y_4 X_5 Y_5
phi_0 phi_1 phi_2 phi_3 phi_4 phi_5]

%Initialize variables
X_0 = u(1);
Y_0 = u(2);
X_1 = u(3);
Y_1 = u(4);
X_2 = u(5);
Y_2 = u(6);
X_3 = u(7);
Y_3 = u(8);
X_4 = u(9);
Y_4 = u(10);
X_5 = u(11);
Y_5 = u(12);
phi_0 = u(13);
phi_1 = u(14);
phi_2 = u(15);
phi_3 = u(16);
phi_4 = u(17);
phi_5 = u(18);

%compute forward kinematics
X_brc = X_0;
Y_brc = Y_0;
R_1 = sqrt((X_1-X_0)^2+(Y_1-Y_0)^2);
Theta_1 = atan2(X_1-X_0,Y_1-Y_0);
R_2 = sqrt((X_2-X_0)^2+(Y_2-Y_0)^2);
F_2 = sqrt((X_2-X_1)^2+(Y_2-Y_1)^2);
R_3 = sqrt((X_3-X_0)^2+(Y_3-Y_0)^2);
F_3 = sqrt((X_3-X_1)^2+(Y_3-Y_1)^2);
R_4 = sqrt((X_4-X_0)^2+(Y_4-Y_0)^2);
F_4 = sqrt((X_4-X_2)^2+(Y_4-Y_2)^2);
R_5 = sqrt((X_5-X_0)^2+(Y_5-Y_0)^2);
F_5 = sqrt((X_5-X_3)^2+(Y_5-Y_3)^2);
phi_brc = phi_0;
phi_1c = phi_1;
phi_2c = phi_2;
phi_3c = phi_3;
phi_4c = phi_4;
phi_5c = phi_5;

Output = [X_brc; Y_brc; R_1; Theta_1; R_2; F_2; R_3; F_3; R_4; F_4;
R_5; F_5; phi_brc; phi_1c; phi_2c; phi_3c; phi_4c; phi_5c;];
```

```
function Output = five_robot_centroid_inverse_kin_v1(u)
%This function computes the robot positions based on the cluster
%position.
%arguments:     u = [X_brc Y_brc R_1 Theta_1 R_2 F_2 R_3 F_3 R_4 F_4
R_5 F_5 phi_brc phi_1c phi_1c phi_2c phi_3c phi_4c phi_5c]
%Initialize variables
X_brc = u(1);
Y_brc = u(2);
R_1 = u(3);
Theta_1 = u(4);
R_2 = u(5);
F_2 = u(6);
R_3 = u(7);
F_3 = u(8);
R_4 = u(9);
F_4 = u(10);
R_5 = u(11);
F_5 = u(12);
phi_brc = u(13);
phi_1c = u(14);
phi_2c = u(15);
phi_3c = u(16);
phi_4c = u(17);
phi_5c = u(18);
X_0 = X_brc;
Y_0 = Y_brc;
X_1 = X_brc+R_1*sin(Theta_1);
Y_1 = Y_brc+R_1*cos(Theta_1);
X_2 = X_brc+R_2*sin(Theta_1+acos((R_1^2+R_2^2-F_2^2)/(2*R_1*R_2))) ;
Y_2 = Y_brc+R_2*cos(Theta_1+acos((R_1^2+R_2^2-F_2^2)/(2*R_1*R_2))) ;
X_3 = X_brc+R_3*sin(Theta_1-acos((R_3^2+R_1^2-F_3^2)/(2*R_3*R_1))) ;
Y_3 = Y_brc+R_3*cos(Theta_1-acos((R_3^2+R_1^2-F_3^2)/(2*R_3*R_1))) ;
X_4 = X_brc+R_4*sin(Theta_1+acos((R_1^2+R_2^2-
F_2^2)/(2*R_1*R_2))+acos((R_2^2+R_4^2-F_4^2)/(2*R_2*R_4))) ;
Y_4 = Y_brc+R_4*cos(Theta_1+acos((R_1^2+R_2^2-
F_2^2)/(2*R_1*R_2))+acos((R_2^2+R_4^2-F_4^2)/(2*R_2*R_4))) ;
X_5 = X_brc+R_5*sin(Theta_1-acos((R_3^2+R_1^2-F_3^2)/(2*R_3*R_1))-
acos((R_5^2+R_3^2-F_5^2)/(2*R_5*R_3))) ;
Y_5 = Y_brc+R_5*cos(Theta_1-acos((R_3^2+R_1^2-F_3^2)/(2*R_3*R_1))-
acos((R_5^2+R_3^2-F_5^2)/(2*R_5*R_3))) ;
phi_0 = phi_brc;
phi_1 = phi_1c;
phi_2 = phi_2c;
phi_3 = phi_3c;
phi_4 = phi_4c;
phi_5 = phi_5c;

Output = [
X_0;Y_0;X_1;Y_1;X_2;Y_2;X_3;Y_3;X_4;Y_4;X_5;Y_5;phi_0;phi_1;phi_2;phi_3
;phi_4;phi_5;];
```

```
function Output = five_bots_jacobian(u)
%This computes the cluster velocities based on robots velocities.
%arguments:     u = [x1_dot y1_dot x2_dot y2_dot x3_dot y3_dot
theta_r1_dot theta_r2_dot theta_r3_dot x1 y1 theta_1 x2 y2 theta_2 x3
y3 theta_3]
%output=[xc_dot yc_dot theta_c_dot phi_1_dot phi_2_dot phi_3_dot p_dot
q_dot beta_dot]
X_0_dot  = u(1);
Y_0_dot  = u(2);
X_1_dot  = u(3);
Y_1_dot  = u(4);
X_2_dot  = u(5);
Y_2_dot  = u(6);
X_3_dot  = u(7);
Y_3_dot  = u(8);
X_4_dot  = u(9);
Y_4_dot  = u(10);
X_5_dot  = u(11);
Y_5_dot  = u(12);
phi_0_dot  = u(13);
phi_1_dot  = u(14);
phi_2_dot  = u(15);
phi_3_dot  = u(16);
phi_4_dot  = u(17);
phi_5_dot  = u(18);
X_0  = u(19);
Y_0  = u(20);
X_1  = u(21);
Y_1  = u(22);
X_2  = u(23);
Y_2  = u(24);
X_3  = u(25);
Y_3  = u(26);
X_4  = u(27);
Y_4  = u(28);
X_5  = u(29);
Y_5  = u(30);
phi_0  = u(31);
phi_1  = u(32);
phi_2  = u(33);
phi_3  = u(34);
phi_4  = u(35);
phi_5  = u(36);
c=five_robot_centroid_forward_kin_v1([X_0 Y_0 X_1 Y_1 X_2 Y_2 X_3 Y_3
X_4 Y_4 X_5 Y_5 phi_0 phi_1 phi_2 phi_3 phi_4 phi_5]);
v_robots=[X_0_dot; Y_0_dot; X_1_dot; Y_1_dot; X_2_dot; Y_2_dot;
X_3_dot; Y_3_dot; X_4_dot; Y_4_dot; X_5_dot; Y_5_dot; phi_0_dot;
phi_1_dot; phi_2_dot; phi_3_dot; phi_4_dot; phi_5_dot];
J_inv = five_bots_centroid_inv_jacobian_matrix_exact(c);
J = inv(J_inv);
Output = J * v_robots;
```

```
function Output = five_bots_inv_jacobian(u)
%This computes robot velocities based on the cluster velocities.
%arguments:     u = [X_brc_dot Y_brc_dot R_1_dot theta_1_dot R_2_dot
F_2_dot R_3_dot F_3_dot phi_brc_dot phi_1_dot phi_2_dot; phi_3_dot
X_brc Y_brc R_1 theta_1 R_2 F_2 R_3 F_3 phi_brc phi_1 phi_2 phi_3]
%output:     output = [J_inv]*[V_c]
%Initialize variables
X_brc_dot = u(1);
Y_brc_dot = u(2);
R_1_dot = u(3);
theta_1_dot = u(4);
R_2_dot = u(5);
F_2_dot = u(6);
R_3_dot=u(7);
F_3_dot=u(8);
R_4_dot=u(9);
F_4_dot=u(10);
R_5_dot=u(11);
F_5_dot=u(12);
phi_brc_dot=u(13);
phi_1c_dot=u(14);
phi_2c_dot=u(15);
phi_3c_dot=u(16);
phi_4c_dot=u(17);
phi_5c_dot=u(18);
X_brc = u(19);
Y_brc = u(20);
R_1=u(21);
theta_1=u(22);
R_2=u(23);
F_2=u(24);
R_3=u(25);
F_3=u(26);
R_4=u(27);
F_4=u(28);
R_5=u(29);
F_5=u(30);
phi_brc=u(31);
phi_1c=u(32);
phi_2c=u(33);
phi_3c=u(34);
phi_4c=u(35);
phi_5c=u(36);
v_cluster=[X_brc_dot; Y_brc_dot; R_1_dot; theta_1_dot; R_2_dot;
F_2_dot; R_3_dot; F_3_dot; R_4_dot; F_4_dot; R_5_dot; F_5_dot;
phi_brc_dot; phi_1c_dot; phi_2c_dot; phi_3c_dot; phi_4c_dot;
phi_5c_dot];
J_inv = five_bots_centroid_inv_jacobian_matrix_exact([X_brc Y_brc R_1
theta_1 R_2 F_2 R_3 F_3 R_4 F_4 R_5 F_5 phi_brc phi_1c phi_2c phi_3c
phi_4c phi_5c]);
Output = J_inv * v_cluster;
```

```
function Output = five_bots_centroid_inv_jacobian_matrix_exact(u)
%This function computes the robot velocities based on the cluster
velocities.
%arguments:     u = [X_brc Y_brc R_1 theta_1 R_2 F_2 R_3 F_3 R_4 F_4
R_5 F_5 phi_brc phi_1c phi_2c phi_3c phi_4c phi_5c]
%output:     output = [J_inv]
%Initialize variables
%X_brc = u(1);
%Y_brc = u(2);
R_1 = u(3);
theta_1 = u(4);
R_2 = u(5);
F_2 = u(6);
R_3 = u(7);
F_3 = u(8);
R_4 = u(9);
F_4 = u(10);
R_5 = u(11);
F_5 = u(12);
%phi_brc = u(13);
%phi_1c = u(14);
%phi_2c = u(15);
%phi_3c = u(16);
%phi_4c = u(17);
%phi_5c = u(18);
Y_1_R_1 = cos(theta_1);
Y_1_theta_1 = -R_1*sin(theta_1);
X_1_R_1 = sin(theta_1);
X_1_theta_1 = R_1*cos(theta_1);

Y_2_R_1 = (R_2*(1/R_2-(-F_2^2+R_1^2+R_2^2)/(2*R_1^2*R_2))*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2));
Y_2_theta_1 = -R_2*sin(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1);
Y_2_R_2 = cos(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1)+(R_2*(1/R_1-(-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2^2))*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2));
Y_2_F_2 = -((F_2*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1))/(R_1*sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2))));

X_2_R_1 = -((cos(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1)*R_2*(1/R_2-(-
F_2^2+R_1^2+R_2^2)/(2*R_1^2*R_2)))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)));
X_2_theta_1 = cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1)*R_2;
X_2_R_2 = -((cos(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1)*R_2*(1/R_1-(-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2^2)))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)))+sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1);
```

```
X_2_F_2 = (cos(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+theta_1)*F_2)/(R_1*sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)));

Y_3_R_1 = (R_3*(1/R_3-(-F_3^2+R_1^2+R_3^2)/(2*R_1^2*R_3))*sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-theta_1))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2));
Y_3_theta_1 = R_3*sin(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-theta_1);
Y_3_R_3 = cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-
theta_1)+(R_3*(1/R_1-(-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3^2))*sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-theta_1))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2));
Y_3_F_3 = -((F_3*sin(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-
theta_1))/(R_1*sqrt(1-(-F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2))));

X_3_R_1 = (cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-
theta_1)*R_3*(1/R_3-(-F_3^2+R_1^2+R_3^2)/(2*R_1^2*R_3)))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2));
X_3_theta_1 = cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-theta_1)*R_3;
X_3_R_3 = (cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-
theta_1)*R_3*(1/R_1-(-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3^2)))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2))-sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-theta_1);
X_3_F_3 = -((cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))-
theta_1)*F_3)/(R_1*sqrt(1-(-F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2))));

X_4_R_1 = -((cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*(1/R_2-(-
F_2^2+R_1^2+R_2^2)/(2*R_1^2*R_2))*R_4)/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)));
X_4_theta_1 = cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*R_4;
X_4_R_2 = cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*R_4*(-((1/R_1-(-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2^2))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)))-(1/R_4-(-
F_4^2+R_2^2+R_4^2)/(2*R_2^2*R_4))/sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2)));
X_4_F_2 = (cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*F_2*R_4)/(R_1*R_2*sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)));
X_4_R_4 = -((cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*R_4*(1/R_2-(-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4^2)))/sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2)))+sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1);
X_4_F_4 = (cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)*F_4)/(R_2*sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2)));

Y_4_R_1 = ((1/R_2-(-F_2^2+R_1^2+R_2^2)/(2*R_1^2*R_2))*R_4*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2));
```

```
Y_4_theta_1 = -R_4*sin(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1);
Y_4_R_2 = -R_4*(-((1/R_1-(-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2^2))/sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2)))-(1/R_4-(-
F_4^2+R_2^2+R_4^2)/(2*R_2^2*R_4))/sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2)))*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1);
Y_4_F_2 = -((F_2*R_4*sin(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1))/(R_1*R_2*sqrt(1-(-
F_2^2+R_1^2+R_2^2)^2/(4*R_1^2*R_2^2))));
Y_4_R_4 = cos(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1)+(R_4*(1/R_2-(-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4^2))*sin(acos((-
F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1))/sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2));
Y_4_F_4 = -((F_4*sin(acos((-F_2^2+R_1^2+R_2^2)/(2*R_1*R_2))+acos((-
F_4^2+R_2^2+R_4^2)/(2*R_2*R_4))+theta_1))/(R_2*sqrt(1-(-
F_4^2+R_2^2+R_4^2)^2/(4*R_2^2*R_4^2))));
X_5_R_1 = (cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*(1/R_3-(-
F_3^2+R_1^2+R_3^2)/(2*R_1^2*R_3))*R_5)/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2));
X_5_theta_1 = cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*R_5;
X_5_R_3 = -cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*R_5*(-((1/R_1-(-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3^2))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2)))-(1/R_5-(-
F_5^2+R_3^2+R_5^2)/(2*R_3^2*R_5))/sqrt(1-(-
F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*R_5^2)));
X_5_F_3 = -((cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*F_3*R_5)/(R_1*R_3*sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2))));
X_5_R_5 = (cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*R_5*(1/R_3-(-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5^2)))/sqrt(1-(-
F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*R_5^2))-sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-
theta_1);
X_5_F_5 = -((cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)*F_5)/(R_3*sqrt(1-(-
F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*R_5^2))));

Y_5_R_1 = ((1/R_3-(-F_3^2+R_1^2+R_3^2)/(2*R_1^2*R_3))*R_5*sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-
theta_1))/sqrt(1-(-F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2));
Y_5_theta_1 = R_5*sin(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1);
Y_5_R_3 = -R_5*(-((1/R_1-(-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3^2))/sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2)))-(1/R_5-(-
F_5^2+R_3^2+R_5^2)/(2*R_3^2*R_5))/sqrt(1-(-
F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*R_5^2)))*sin(acos((-
```

```
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-
theta_1);
Y_5_F_3 = -((F_3*R_5*sin(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1))/(R_1*R_3*sqrt(1-(-
F_3^2+R_1^2+R_3^2)^2/(4*R_1^2*R_3^2))));
Y_5_R_5 = cos(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1)+(R_5*(1/R_3-(-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5^2))*sin(acos((-
F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-
theta_1))/sqrt(1-(-F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*R_5^2));
Y_5_F_5 =-((F_5*sin(acos((-F_3^2+R_1^2+R_3^2)/(2*R_1*R_3))+acos((-
F_5^2+R_3^2+R_5^2)/(2*R_3*R_5))-theta_1))/(R_3*sqrt(1-(-
F_5^2+R_3^2+R_5^2)^2/(4*R_3^2*
R_5^2))));

J_inv =
[1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[1,  0,X_1_R_1,X_1_theta_1,0,0,0,0,  0,  0,  0,  0,  0,  0, 0, 0, 0, 0]
[0,  1,Y_1_R_1,Y_1_theta_1,0,0,0,0,  0,  0,  0,  0,  0,  0, 0, 0, 0, 0]
[1,  0,X_2_R_1,X_2_theta_1,X_2_R_2,X_2_F_2,0,0,0,0,0,0,0,0, 0, 0, 0, 0]
[0,  1,Y_2_R_1,Y_2_theta_1,Y_2_R_2,Y_2_F_2,0,0,0,0,0,0,0,0, 0, 0, 0, 0]
[1,  0,X_3_R_1,X_3_theta_1,0,0,X_3_R_3,X_3_F_3,0,0,0,0,0,0, 0, 0, 0, 0]
[0,  1,Y_3_R_1,Y_3_theta_1,0,0,Y_3_R_3,Y_3_F_3,0,0,0,0,0,0, 0, 0, 0, 0]
[1,0,X_4_R_1,X_4_theta_1,X_4_R_2,X_4_F_2,0,0,X_4_R_4,X_4_F_4,0,0,0,0,0,
0,0,0]
[0,1,Y_4_R_1,Y_4_theta_1,Y_4_R_2,Y_4_F_2,0,0,Y_4_R_4,Y_4_F_4,0,0,0,0,0,
0,0,0]
[1,0,X_5_R_1,X_5_theta_1,0,0,X_5_R_3,X_5_F_3,0,0,X_5_R_5,X_5_F_5,0,0,0,
0,0, 0]
[0,1,Y_5_R_1,Y_5_theta_1,0,0,Y_5_R_3,Y_5_F_3,0,0,Y_5_R_5,Y_5_F_5,0,0,0,
0,0, 0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0, 1]];
Output = J_inv;

TL;DR -Robots are cool
```