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

Paul Mahacek, *Student Member, IEEE*, Christopher A. Kitts, *Senior Member, IEEE*, and 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. In this paper, we describe the control architecture used to establish the guarding behavior, review the design of the robotic kayaks, and present experimental data regarding the functionality and performance of the system.**

***Index Terms*—Cluster space, collaborative control, formation control, multirobot systems, robot teams.**

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]. In 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 multirobot 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 test bed, 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, Pittsburgh, PA, 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 demonstration in 2009 during the Navy's Trident Warrior exercise, the Control Architecture for Robotic agent Command and Sensing (CARACaS) 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 study 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 study envisioned multi-USV buoy systems for a variety of marine applications ranging

Manuscript received April 16, 2011; revised August 25, 2011; accepted October 8, 2011. Date of publication December 8, 2011; date of current version January 9, 2012. Recommended by Technical Editor J. Xu. This work was supported in part by the National Science Foundation under Grant CNS0619940, in part by the National Aeronautics and Space Administration (NASA), Washington, DC, and in part by Santa Clara University, Santa Clara, CA.

The authors are with the Robotic Systems Laboratory, Santa Clara University, Santa Clara, CA 95053 USA (e-mail: PMahacek@scu.edu; ckitts@scu.edu; iamas@scu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2011.2174376

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 multirobot 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 real-time 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 study presented in this paper 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 study introduces an application layer above the centralized formation controller, transforming application-specific specifications into cluster-space control specifications; these are used to implement the real-time cluster controller, which in turn determines the drive commands for each individual robot in the fleet. In Section II, we review the cluster-space control approach and its integration with a specific application, i.e., dynamically establishing a barrier between threatening marine traffic and an asset that must be protected. In Section III, we review the design of the multi-USV system. In Section IV, we present experimental field data, and in Section V, we discuss future work and draw conclusions about the significance of this study.

II. Cluster-Space Controller

Our research in the cluster-space control is motivated by our vision of a specific class of multirobot applications that require the complete degree-of-freedom (DOF) control of the spatial and motion characteristics of a locally distributed mobile multirobot system that tightly interacts in real time. At the same time, we desire transparency for the formation's DOFs in order for a real-time 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 multirobot 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 DOFs may be continuously articulated. Some of these advantages come at the cost of increased computation within the real-time 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 multirate control [26]; these strategies are both suitable for dramatically reducing computational load and the need for information sharing throughout the cluster.

A. 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 real-time pilot specifies and monitors cluster motion, and control computations are executed with respect to the cluster-space variables (which lead 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 -pace state estimates.

As an example, 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 toward 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

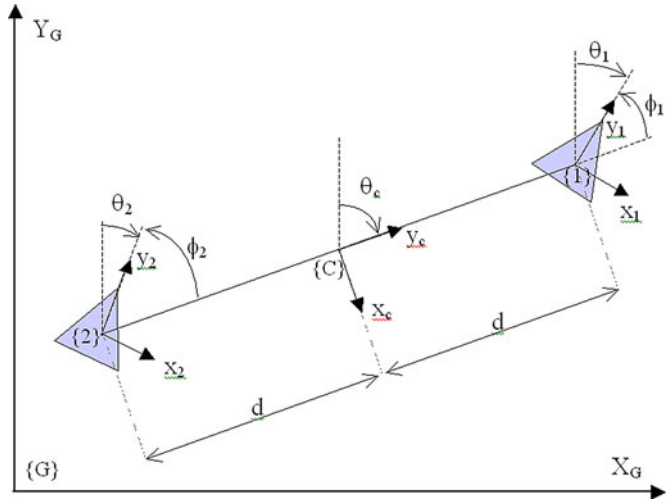

Fig. 1. Representing the pose of a two-robot system using a cluster-space description.

to the cluster frame; this results in a cluster pose vector of $\vec{C} = (x_C, y_C, \theta_C, d, \phi_1, \phi_2)^T$.

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, i.e., ${}^G\dot{\vec{R}}$ 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 J that maps robot velocities to cluster velocities in the form of a time-varying linear function

$$\dot{\vec{C}} = \begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \\ \vdots \\ \dot{c}_{mn} \end{pmatrix} = {}^G J({}^G\vec{R}) {}^G\dot{\vec{R}}$$

$$= \begin{pmatrix} \frac{\partial g_1}{\partial r_1} & \frac{\partial g_1}{\partial r_2} & \cdots & \frac{\partial g_1}{\partial r_{mn}} \\ \frac{\partial g_2}{\partial r_1} & \frac{\partial g_2}{\partial r_2} & \cdots & \frac{\partial g_2}{\partial r_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{mn}}{\partial r_1} & \frac{\partial g_{mn}}{\partial r_2} & \cdots & \frac{\partial g_{mn}}{\partial r_{mn}} \end{pmatrix} \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_{mn} \end{pmatrix}. \quad (1)$$

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 study reported on in this paper. We have also developed and implemented more sophisticated nonlinear dynamic controllers.

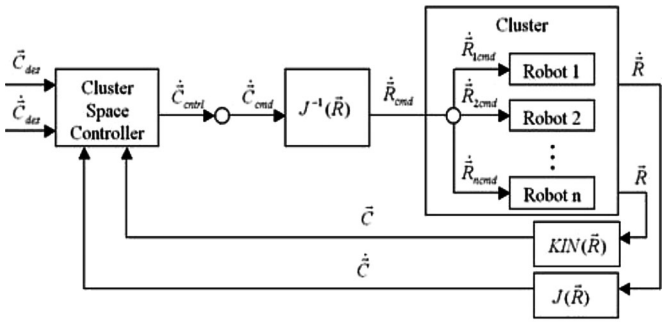

Fig. 2. Inverse Jacobian cluster-space control architecture for a mobile multi-robot system.

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 the dynamic control of the individual vehicles [27].

To date, we have successfully implemented the cluster-space control in experiments with clusters of up to six 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 on 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."

In Fig. 3, the relevant reference frames and geometric layout for a planar 5-USV cluster are depicted. 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

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

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 $i = (1, 2, 3,$

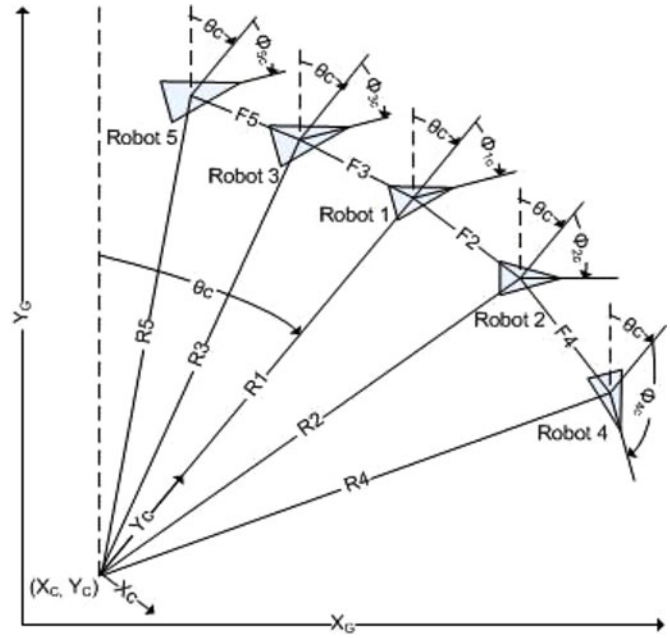

Fig. 3. Five-USV cluster geometry.

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 nonholonomic 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 DOFs for each of the six fleet entities (five boats and the protected asset), yielding a total of 12 linear DOFs for the robot group.

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 toward 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

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

Here, 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 nonholonomic 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 nonholonomic controller for use in systems of this type [34]. Removing φ_i angles from consideration leaves us with 12 cluster DOFs, matching the 12 linear DOFs in robot space.

Given ${}^G\vec{R}$ and $\vec{C}$, the set of forward-position kinematic equations, i.e., $\vec{C} = \text{KIN}({}^G\vec{R})$, is given as follows:

$$X_c = X_0 \tag{2}$$

$$Y_c = Y_0 \tag{3}$$

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

$$\theta_1 = A\tan 2((X_1 - X_0), (Y_1 - Y_0)) \tag{5}$$

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

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

Inversely, the set of inverse-position kinematic equations, i.e., ${}^G\vec{R} = \text{INVKIN}(\vec{C})$, is given as follows:

$$X_0 = X_c \tag{8}$$

$$Y_0 = Y_c \tag{9}$$

$$X_1 = X_c + R_1 * \sin(\theta_1) \tag{10}$$

$$Y_1 = Y_c + R_1 * \cos(\theta_1) \tag{11}$$

$$X_i = X_c + R_i * \sin\left(\theta_1 + a\,\cos\left(\frac{(R_1^2 + R_i^2 - F_i^2)}{2 * R_1 * R_i}\right)\right) \tag{12}$$

for $i = 2, 3$

$$Y_i = Y_c + R_i * \cos\left(\theta_1 + a\cos\left(\frac{(R_1^2 + R_i^2 - F_i^2)}{2 * R_1 * R_i)}\right)\right) \tag{13}$$

for $i = 2, 3$

$$X_j = X_c + R_j * \sin\left(\theta_1 + a\cos\left(\frac{(R_1^2 + R_{j-2}^2 - F_j^2)}{2 * R_1 * R_{j-2}}\right)\right) + a\cos\left(\frac{(R_j^2 + R_{j-2}^2 - F_j^2)}{2 * Rj * R_{j-2}}\right) \quad \text{for } j = 4, 5 \tag{14}$$

$$Y_j = Y_c + R_j * \cos\left(\theta_1 + a\cos\left(\frac{(R_1^2 + R_{j-2}^2 - F_{j-2}^2)}{2 * R_1 * R_{j-2}}\right)\right) + a\cos\left(\frac{(R_j^2 + R_{j-2}^2 - F_j^2)}{2 * Rj * R_{j-2})}\right) \quad \text{for } j = 4, 5. \tag{15}$$

The forward and inverse velocity kinematics provide the formal relationship between the robot and the cluster-space velocities, i.e., ${}^G\dot{\vec{R}}$ and $\dot{\vec{C}}$. From (2) and (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

$$\dot{\vec{C}} = \begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \\ \vdots \\ \dot{c}_{mn} \end{pmatrix} = {}^{G}J({}^{G}\vec{R})\,{}^{G}\dot{\vec{R}}$$

$$= \begin{pmatrix} \frac{\partial g_1}{\partial r_1} & \frac{\partial g_1}{\partial r_2} & \cdots & \frac{\partial g_1}{\partial r_{mn}} \\ \frac{\partial g_2}{\partial r_1} & \frac{\partial g_2}{\partial r_2} & \cdots & \frac{\partial g_2}{\partial r_{mn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{mn}}{\partial r_1} & \frac{\partial g_{mn}}{\partial r_2} & \cdots & \frac{\partial g_{mn}}{\partial r_{mn}} \end{pmatrix} \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \vdots \\ \dot{r}_{mn} \end{pmatrix}. \quad (16)$$

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. 3.

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.

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, in Fig. 5, the spatial quantities of interest are shown. The overall concept of operation is as follows. The object being protected is at a location ($X_{\rm obj}$, $Y_{\rm obj}$). 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 toward the threat in order to meet it at a maximum radius of $R_{\max}$.

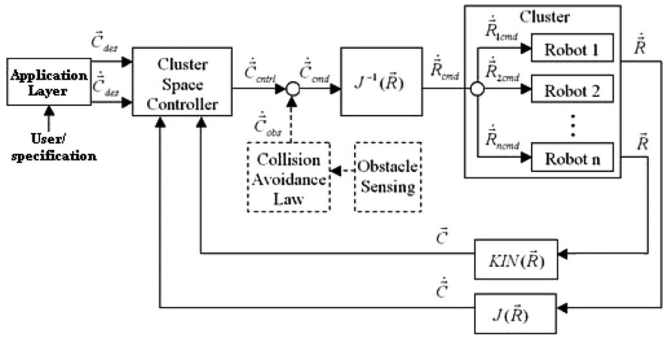

Fig. 4. Cluster-space control architecture for an n-robot system.

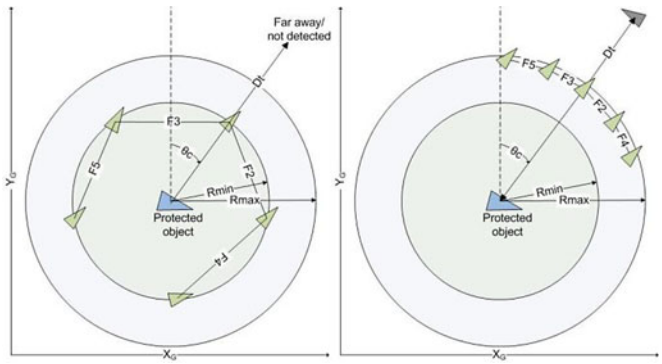

Fig. 5. Application-layer variables showing two cases, where the threat is either far away or not detected at all (left), or where the threat is close and the USVs have shifted to guard the protected object or area (right).

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 ($X_{\rm obj}$, $Y_{\rm obj}$, $R_{\min}$, $R_{\max}$, D_T, θ_T, $F_{\min}$) to cluster-space variable inputs. For the guarding application, these transforms are of the form represented as follows, assuming $D_T > R_{\max}$:

$$X_c = X_{\rm obj} \quad (17)$$

$$Y_c = Y_{\rm obj} \quad (18)$$

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

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

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

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.

This set of application-layer transforms operate under the specifications provided by the supervisory operator and provide the resulting cluster-space desired values to the cluster-control loop (see Fig. 4). 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, real-time control computations are still being performed in the cluster space (e.g., real-time 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 Laboratory 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.

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 m). The modem is a Metricom Ricochet 128 Kbits/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 3-h run time at normal operations with a top speed of 5 kn. The mounting structure is made from 6061 aluminum tubing with an 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.

Fig. 6. One of the robotic kayaks.

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

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 Virtual Reality Modeling Language (VRML) simulator (see Fig. 7), 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.

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 the application space. Four main test cases were run over the course of a multiday deployment at Lake Del Valle near Livermore, CA (see 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 small waterplane area twin-hull (SWATH) mapping vessel.

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

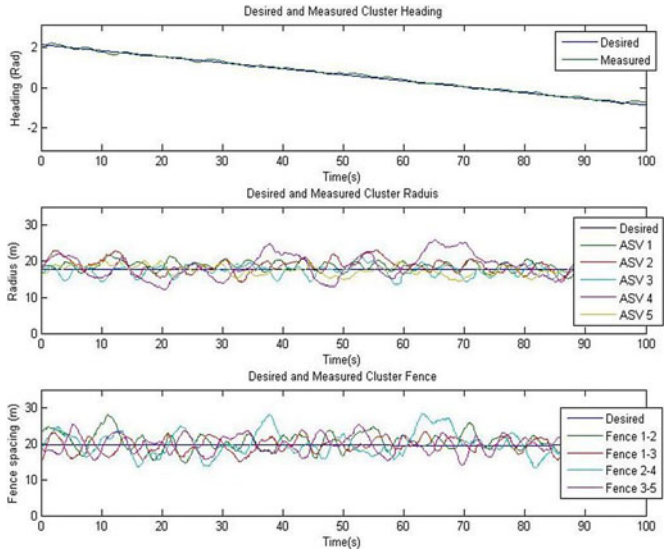

Fig. 9. Standard shielding, constant radius, no threat. (The rms error in Table I.)

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 17 m, 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 study, 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 I summarizes the rms errors for the controlled radial and interrobot spacing parameters; all rms errors are less than 4 m, which we consider to be outstanding given the limited sensor performance and disturbance environment.

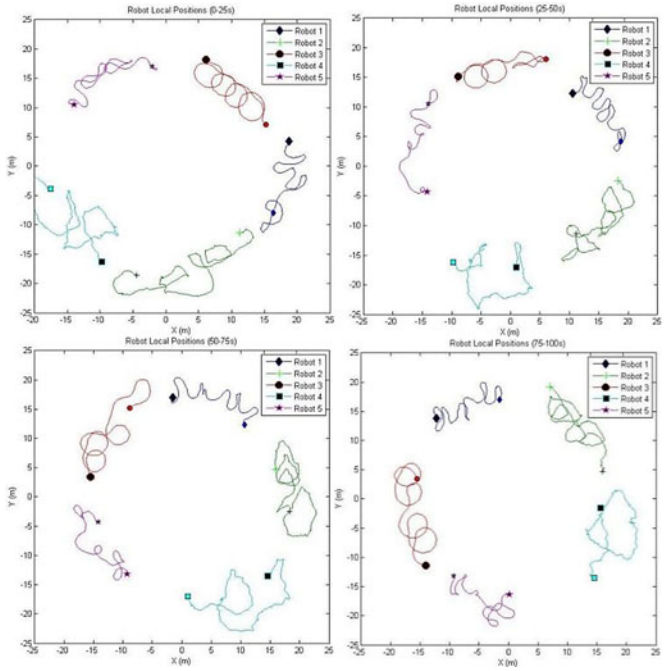

Fig. 10. Standard shielding, constant radius, no threat (overhead view of the same run as in Fig. 9). Looping trail patterns are a function of nonoptimized velocity gains, as well as lacking a dead band around the desired position. Differences in trail patterns can be attributed to various kayak hulls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

TABLE I
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

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 with the constant rotation and varying radius are shown. 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. In Fig. 12, an overhead view of the outward spiral maneuver is shown, which is a portion of the test run shown in Fig. 11. In Table II, we summarize the rms errors of the controlled radius and spacing parameters; again, excellent results are shown, with all rms errors less than 3 m.

C. Threat Detection

The third experimental run demonstrates a case of shielding upon detection of a threat. In this instance, the standard radius is

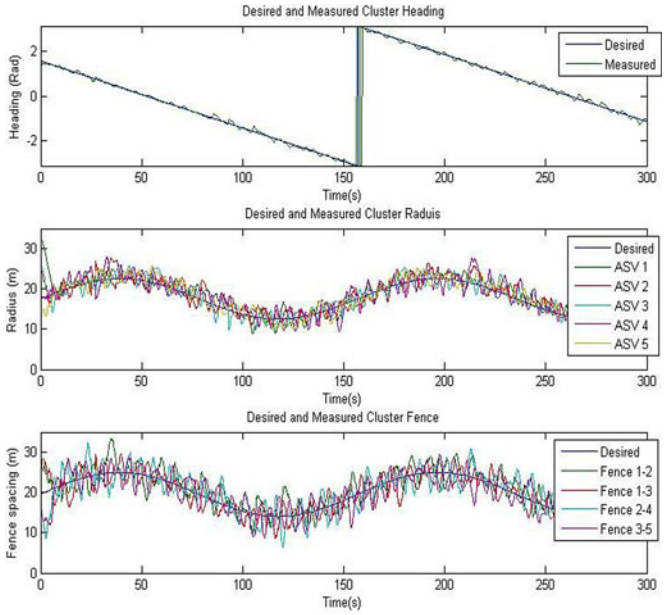

Fig. 11. Basic shielding cluster variables. Note that the jump in the top plot is caused as the heading wraps from $-\pi$ to π at 180°, and is not an actual discontinuity. (The rms error is given in Table II.)

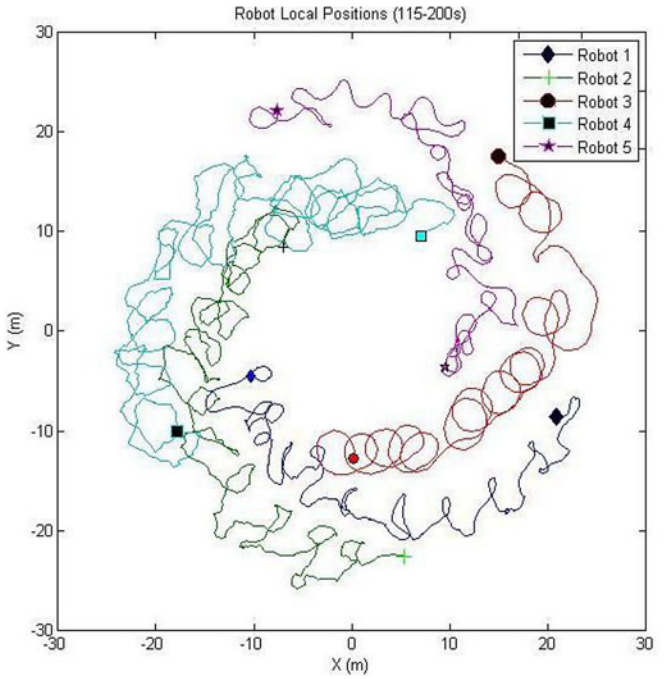

Fig. 12. Overhead view of a change in the shielding radius.

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

Cluster Radii	RMS Error (m)		Cluster Fence Spacings	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

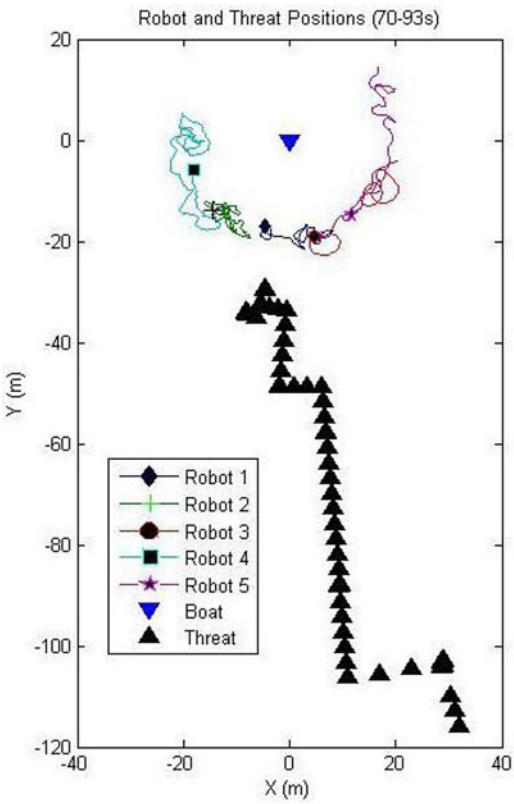

Fig. 13. Overhead view of the shielding technique with threat detection.

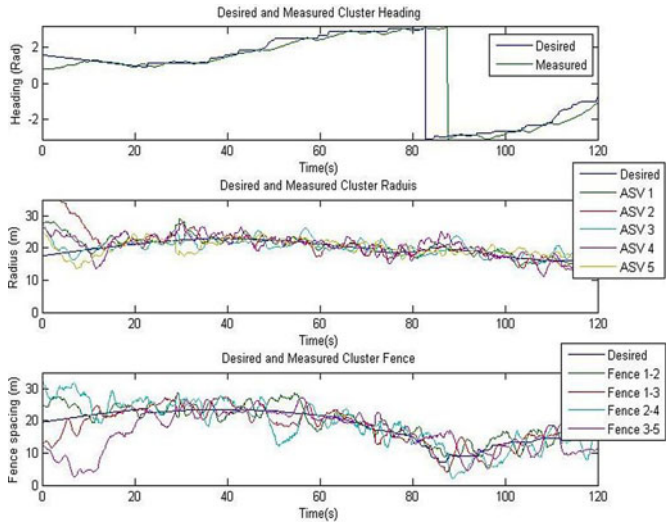

Fig. 14. Shielding with the threat-detection cluster-space variable. (Table III shows rms error not including the initialization time from 0 to 20 s).

set to 17 m, the maximum approach is 25 m, and the minimum fence spacing is set to 10 m.

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

TABLE III
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

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

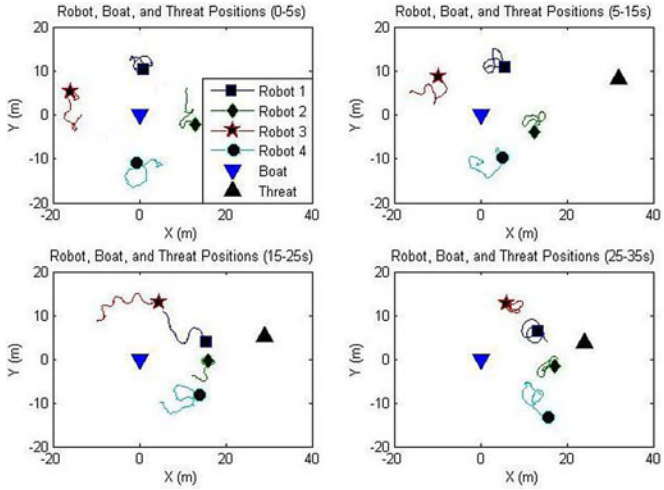

Fig. 16. Overhead view of the shielding technique with threat detection around a mapping vessel.

scenario. The latter part of the experiment shows the kayaks returning to an evenly spaced rotation about the protected asset as the threat disappears. In Table III, the rms errors less than 4 m are given.

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 (see Fig. 15). The protected vessel is another autonomous surface vessel, a SWATH boat, equipped with a multibeam sonar, attitude and heading reference system (AHRS), GPS, and heave sensors designed for shallow-water bathymetry. The 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.

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 divided into four time steps. In step 1, the fleet of four USVs has 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.

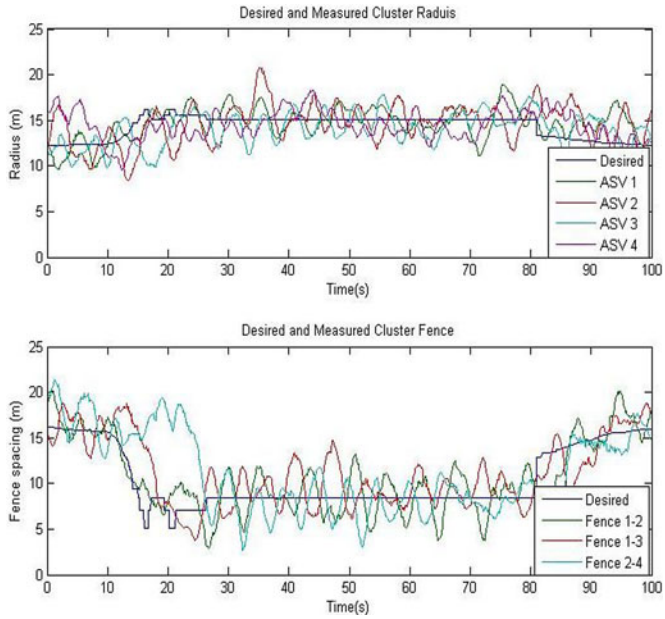

Fig. 17. Cluster variables shielding with threat detection of a mapping vessel.

TABLE IV
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

The individual measured cluster variables are shown in Fig. 17. In Table IV, the rms errors for the controlled parameters are given; as before, all errors are less than 4 m.

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 here fits within the leader–follower paradigm, and 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 DOFs 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] into the overall multirobot 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.

VI. Summary and Conclusion

In this paper, we described the use of a fleet of robotic marine vessels capable of guarding critical assets from threats. The 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 were presented. As a result, the five-robot cluster-space definition and control architecture were validated and the functionality was proven for this application.

Acknowledgment

The authors would like to thank student researchers who assisted with performing experiments and who have contributed to the development of the cluster-space approach and to include authors listed in [29]–[36]. This work was benefited greatly through the feedback of numerous colleagues; the authors are particularly indebted to T. Ademek, K. Rasal, and the RSL graduate research team for their time and effort, which were critical to this study. Any opinions, findings, and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, NASA, or Santa Clara University.

References

[1] S. J. Corfield and J. M. Young, "Unmanned surface vehicles: Game changing technology for naval operations," in *Advances in Unmanned Marine Systems*, G. N. Roberts and R. Sutton, Eds. Hertfordshire, U.K: IEEE Press, 2006, pp. 311–328.

[2] S. Saunders, "Mine warfare forces," in *Janes Fighting Ships*. London, U.K: IHS, 2004, pp. 177–180.

[3] V. Bertram, "Unmanned surface vehicles: A survey," in *Skibsteknisk Selskab*. Copenhagen, Denmark, 2008.

[4] J. Manley, "Unmanned surface vehicles, 15 years of development," in *Proc. MTS/IEEE Oceans*, Kobe, Japan, Sep. 2008, pp. 1–4.

[5] M. Caccia, "Autonomous surface craft: Prototypes and basic research issues," in *Proc. 14th Mediterranean Conf. Control Autom.*, Ancona, Italy, Jun. 2006, pp. 1–6.

[6] C. Kitts and M. Egerstedt, "Design, control and applications of real-world multirobot systems," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, p. 8, Mar. 2008.

[7] J. Curcio, J. Leonard, and A. Patrikalakis, "SCOUT: A low cost autonomous surface platform for research in cooperative autonomy," in *Proc. MTS/IEEE Oceans*, 2005, vol. 1, pp. 725–729.

[8] J. Curcio, J. Leonard, J. Vaganay, A. Patrikalakis, A. Bahr, D. Battle, H. Schmidt, and M. Grund, "Experiments in moving baseline navigation using autonomous surface craft," in *Proc. MTS/IEEE Oceans*, 2005, vol. 1, pp. 730–735.

[9] J. Dolan, G. Podnar, S. Stancliff, K. Low, A. Elfes, J. Higinbotham, J. Hosler, T. Moisan, and J. Moisan, "Cooperative aquatic sensing using the telesupervised ocean sensor fleet," in *Proc. Remote Sensing Ocean, Sea Ice, Large Water Regions*, 2009, vol. 7473, pp. 1–12.

[10] J. Esposito, M. Feemster, and E. Smith, "Cooperative manipulation on the water using a swarm of autonomous tugboats," in *Proc. IEEE Int. Conf. Robot. Autom.*, 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," *J. Field Robot.*, vol. 27, pp. 790–818, 2010.

[12] Z. Z. Abidin, M. Arshad, U. Ngah, and O. Ping, "Control of mini autonomous surface vessel," in *Proc. MTS/IEEE Oceans*, Sydney, Australia, May 2010, pp. 1–4.

[13] Protei. (2011). [Online]. Available: http://sites.google.com/a/opensailing.net/protei/

[14] J. A. Curcio, P. McGillivary, K. Fall, A. Maffei, K. Schwehr, B. Twiggs, C. Kitts, and P. Ballou, "Self-positioning smart buoys, the "Un-Buoy" solution: Logistic considerations using autonomous surface craft technology and improved communications infrastructure," in *Proc. MTS/IEEE Oceans*, Sep. 2006, pp. 1–5.

[15] J. A. Curcio, P. McGillivary, K. Fall, A. Maffei, K. Schwehr, B. Twiggs, C. Kitts, and P. Ballou. (2005). Survey of Portable Range Technologies, U.S. Army White Sands Missile Range Special Report. [Online]. Available: http://www.jcte.jcs.mil/RCC

[16] E. Fiorelli, N. Leonard, P. Bhatta, D. Paley, R. Bachmayer, and D. Fratantoni, "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE J. Oceanic Eng.*, vol. 31, no. 4, pp. 935–948, Oct. 2006.

[17] Y. Tan and B. Bishop, "Evaluation of robot swarm control methods for underwater mine countermeasures," in *Proc. Annu. Southeastern Symp. Syst. Theory*, 2004, vol. 36, pp. 294–298.

[18] T. Balch and R. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[19] E. Flinn, "Testing for the 'boids'," *Aerospace Amer.*, vol. 43, no. 6, pp. 28–29, Jun. 2005.

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

[21] K. Tan and M. Lewis, "Virtual structures for high-precision cooperative mobile robotic control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 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 Trans. Mechatronics*, vol. 11, no. 2, pp. 169–178, Apr. 2006.

[23] C. A. Kitts and I. Mas, "Cluster space specification and control of mobile multirobot systems," *IEEE/ASME Trans. Mechatronics*, vol. 14, no. 2, pp. 207–218, Apr. 2009.
[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 Trans. Robot. Automat.*, vol. 25, no. 6, pp. 1431–1438, Dec. 2009.
[25] I. Mas and C. Kitts, "Centralized and decentralized multi-robot control methods using the cluster space control framework," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, Montreal, Canada, Jul. 2010, pp. 1–8.
[26] M. Meserve, "Multirate control applied to a cluster space robot formation," M.S. thesis, Dept. Electr. Eng., Santa Clara Univ., Santa Clara, CA, Mar. 2011.
[27] I. Mas and C. Kitts, "Model-based nonlinear cluster space control of mobile robot formations," in *Multi-Robot Systems, Trends and Development*, T. Yasuda, Ed., Rijeka, Croatia: In Tech, ch. 4, 2011, pp. 53–70.
[28] I. Mas, S. Li, J. Acain, and C. Kitts, "Entrapment/escorting and patrolling missions in multi-robot cluster space control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 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 Technol. Soc. J.*, vol. 43, no. 1, pp. 13–20, 2009.
[30] P. Mahacek, I. Mas, and C. Kitts, "Cluster space control of autonomous surface vessels utilizing obstacle avoidance and shielding techniques," in *Proc. IEEE/OES Auton. Underwater Vehicles*, Sep. 1–3, 2010, pp. 1–5.
[31] T. Adamek, "Cluster space gradient contour tracking for mobile multi-robot systems," M.S. thesis, Dept. Mech. Eng., Santa Clara University, Santa Clara, CA, Jan. 2011.
[32] V. Howard, "A study of gradient climbing techniques using cluster space control of multi-robot systems," M.S. thesis, Dept. Mech. Eng., Santa Clara University, Santa Clara, CA, Mar. 2011.
[33] G. Okamoto, C. Chen, and C. Kitts, "Beamforming performance for a reconfigurable sparse array smart antenna system via multi-robot control," in *Proc. SPIE Defense, Security, Sensing Conf.*, Orlando, FL, Apr. 2010, pp. 1–11.
[34] I. Mas and C. Kitts, "Obstacle avoidance policies for cluster space control of nonholonomic multirobot systems," *IEEE/ASME Trans. Mechatronics*, to be published. DOI: 10.1109/TMECH.2011.2159988. [Online]. Available: http://ieeexplore.ieee.org.
[35] E. Beck, W. Kirkwood, D. Caress, T. Berk, P. Mahacek, K. Brashem, J. Acain, V. Reddy, C. Kitts, J. Skutnik, and G. Wheat, "SeaWASP: A small waterplane area twin hull autonomous platform for shallow water mapping," *Marine Technol. Soc. J.*, vol. 43, no. 1, pp. 6–12, 2009.
[36] C. Kitts, "Managing space system anomalies using first principles reasoning," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 39–50, Dec. 2006.

Paul Mahacek (S'11) received the B.S., M.S., and Engineer's degrees in mechanical engineering from Santa Clara University, Santa Clara, CA.

He is currently a Research Engineer for l'Université Pierre et Marie Curie and is stationed at the Laboratoire d'Océanographie de Villefranche in Villefranche-sur-Mer, France.

Christopher Kitts (S'98–A'00–M'03–SM'05) received the B.S.E. degree from Princeton University, Princeton, NJ, the M.P.A. degree from the University of Colorado, Boulder, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA.

He was a Research Engineer and an Operational Satellite Constellation Mission Controller. He was an Officer in the U.S. Air Force Space Command, a NASA Contractor with Caelum Research Corporation, Department of Defense (DoD) Research Fellow, and the Graduate Student Director of the Space Systems Development Laboratory, Stanford University. He is currently an Associate Professor at Santa Clara University, Santa Clara, CA, where he is also the Director of the Robotic Systems Laboratory.

Ignacio Mas (S'06–M'12) received the Engineering degree in electrical engineering from the Universidad de Buenos Aires, Buenos Aires, Argentina, and the Ph.D. degree in multirobot navigation and formation control from the Mechanical Engineering Department, Santa Clara University, Santa Clara, CA.

He was a Satellite Systems Engineer and Spacecraft Systems Designer. He is currently a Researcher in the Robotic Systems Laboratory, Santa Clara University.