IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 25, 2019, accepted April 26, 2019, date of publication May 15, 2019, date of current version May 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917120

Experimental Implementation and Verification of Scalar Field Ridge, Trench, and Saddle Point Maneuvers Using Multirobot Adaptive Navigation

ROBERT T. MCDONALD, (Student Member, IEEE), CHRISTOPHER A. KITTS, (Senior Member, IEEE), AND MICHAEL A. NEUMANN, (Member, IEEE)
Department of Mechanical Engineering, Santa Clara University, Santa Clara, CA 95053, USA
Corresponding author: Robert T. McDonald (rmcdonald@scu.edu)

ABSTRACT Adaptive navigation of scalar fields is a compelling capability in which mobile robotic systems make real-time navigation decisions based on sensed measurements of the environment. This capability can enable efficient identification and location of specific features of interest within the field of interest, potentially saving time and energy while also being responsive to changing conditions. Applications can include finding the sources and impact zones of a pollutant, establishing hazard perimeters, finding safe zones, and safe paths of travel. This paper presents new work that experimentally verifies several adaptive navigation control policies for moving to/along critical scalar field features with a group of mobile robots. Specifically, we demonstrate the use of a five robot cluster of sensor-equipped mobile robots to descend ridges within a scalar field, to ascend trenches, and to move to and hold a position at saddle points. This is done through the use of differential measurements across the cluster's formation baselines and control laws that have been previously demonstrated in simulation. This paper also incorporates a new state machine within the adaptive navigation control architecture in order to monitor the performance of the individual control primitives and to respond to conditions such as losing track of the feature of interest. Finally, this paper is the first in which we have experimentally demonstrated control of a five robot group of robots using our cluster space control methodology. The experiments were conducted using a novel indoor multi-robot testbed with the ability to establish customizable scalar fields printed in greyscale on large sheets that are actively sensed by the robots to enable controlled experimental evaluation. Four different fields are used in this study in order to demonstrate the new capabilities of interest.

INDEX TERMS Adaptive navigation, adaptive sampling, differential control, multi-robot formations, formation control, cluster space control.

I. INTRODUCTION

In a traditional navigation scenario, the desired waypoints or trajectory for the vehicle is known and prescribed. Alternatively, Adaptive Navigation (AN) provides the capability to automatically adjust the route and/or destination based on realtime information typically collected by the vehicle itself. Obstacle avoidance is perhaps the simplest AN approach and has been successfully demonstrated through the use of artificial potential fields [1], fuzzy logic [2], tree-based approaches [3], and integrated path planning [4]. Objective-based approaches include techniques for evading pursuers [5] and managing building evacuation routes [6].

Another class of AN focuses on navigating to/along specific conditions of interest without prior knowledge of their position. To date, most work of this nature has focused on scalar fields, which are regions in which a single scalar value of some characteristic of interest, such as temperature or concentration level, is associated with every point in the field. Features of interest within such fields include minima/maxima, contours, ridges, trenches, and saddle points. Such features are critical for a variety of applications ranging

The associate editor coordinating the review of this manuscript and approving it for publication was Luigi Biagiotti.

from environmental monitoring to scientific exploration. For example, for a disaster response application in which the concentration of a toxic gas is being measured, moving to the maximum point in the field is equivalent to finding a source of the gas leak. Contour following can be used to define the extent of the hazardous region in order to establish a safety perimeter. Moving down a ridge allows one to follow the plume to its impact region, while moving up a trench allows one to move towards the source with the minimum possible exposure. Finally, saddle points define safe passage gateways between multiple leak sources.

Nearly all prior work on scalar field AN has focused on finding extreme points or moving to/along contours or level sets. Single vehicle approaches have been explored in simulation and include using a sliding mode controller [7] as well as a bio-inspired plume casting strategy [8] for moving to local maxima. Gradient-based extrema-finding approaches have been demonstrated experimentally, but the need for multidimensional scalar data typically requires the addition of random lateral motions or sinusoidal perturbations to a vehicle's path [9], [10]. Single vehicle contour/level set navigation has been explored for a time-varying field using a sliding mode controller; this has been demonstrated with simulated fields in order to actuate both simulated and real land rovers [11], [12].

Multirobot scalar field AN offers performance advantages compared to single vehicle approaches. These include the ability to instantaneously gather distributed information, to avoid inefficient dithering, to tune aperture size, and to conveniently exploit spatial characterization techniques beyond gradient sensing. Multirobot AN for extrema-finding and contour/level set following has been explored by a wide range of groups over the past 15 years. Simulation has been used to demonstrate bio-inspired and swarm methods [13], [14], gradient-based approaches [15], [16], Hessian-based filtering [17], and even the use of an image processing technique known as the ‘‘snake algorithm’’ [18].

Lab-based experimental verification of multirobot AN techniques can be difficult to accomplish, and field demonstrations of such capabilities are rare. Two significant challenges relating to this limited experimental work include the need to have a capable multirobot system and the desire to have access to an arbitrary scalar field for which truth data can be determined, which isn't changing too quickly, and which has the array of appropriately sized features of interest. Lab-based experimental work generally consists of tabletop installations or a floor-based range, with workspaces typically on the order of 4 m^2 to 50 m^2, respectively. These testbeds are typically specific to a single scalar quantity, are often instrumented with robot position tracking systems, and use projected light for continuous fields or tape/floor mats to create binary level-set regions. Examples of such work include gradient-based source seeking systems [19], [20] and robot groups using a tape-following algorithm to track a level set boundary [21], [22].

For demonstrations of multirobot AN outside of a lab environment, we are only aware of our own work. One of these consisted of three land rovers operating within a $1,000\ m^2$ outdoor test range, with a scalar field consisting of received signal strength of a radio signal. The field was created and sensed by low-cost wireless radio transceivers, with the robots capable of demonstrating gradient-based extrema-finding [23]. The other demonstration was a collection of field deployments in which a cluster of robotic kayaks was used in Stevens Creek Reservoir in Cupertino, CA and in Lake Tahoe, California. These kayaks used sonar sensors to ascend/descend bathymetric gradients and to follow bathymetric contours [24]. These activities covered a workspace of more than $75,000\ m^2$ and used a naturally occurring scalar field.

In addition to the limited amount of experimental work in this field, the scope of work has been limited. While there are a variety of critical features in a scalar field, essentially all work performed to date has focused on extreme points and contours. Our prior work in this regard has identified the application-oriented importance of navigating to/along other features, including down ridges, up trenches, and to saddle points; furthermore, we have developed primitive control laws for all of these capabilities, have developed a unified control framework for implementing these laws, and have demonstrated the control architecture in simulation [25]. While this work highlighted the potential of these strategies, it relied on a number of simplifications, to include the assumption of a perfect formation, no individual robot dynamics, smooth scalar fields without noise, no sensor noise, no communication loss, etc. As the end goal is to use these capabilities in the field, the next logical step was to test ridge, trench, and saddle point navigation in a lab setting such that all these real world effects could be included.

In this paper, we report on what we believe is the first experimental demonstration of these multirobot AN capabilities. We also describe an extension to the control architecture presented in [25] consisting of the addition of a state machine to switch between control primitives; the intent of this was to handle situations when the system lost track of the feature of interest (e.g., moved off a ridge such that it was no longer straddling the feature), allowing the system to backtrack in order to reacquire the feature and then continue. Section II discusses the extended control architecture used to perform these experiments. Section III discusses the experimental testbed, a small indoor system that establishes a scalar field in the form of large mats printed with a greyscale representation of the field. Experimental results are discussed in Section IV along with performance evaluation. Finally, Section V presents conclusions and proposes future work.

II. CONTROL ARCHITECTURE

In [25] we presented a multilayered control system that unified the execution of our AN controllers, each of which was used to navigate to/along a critical feature of interest within

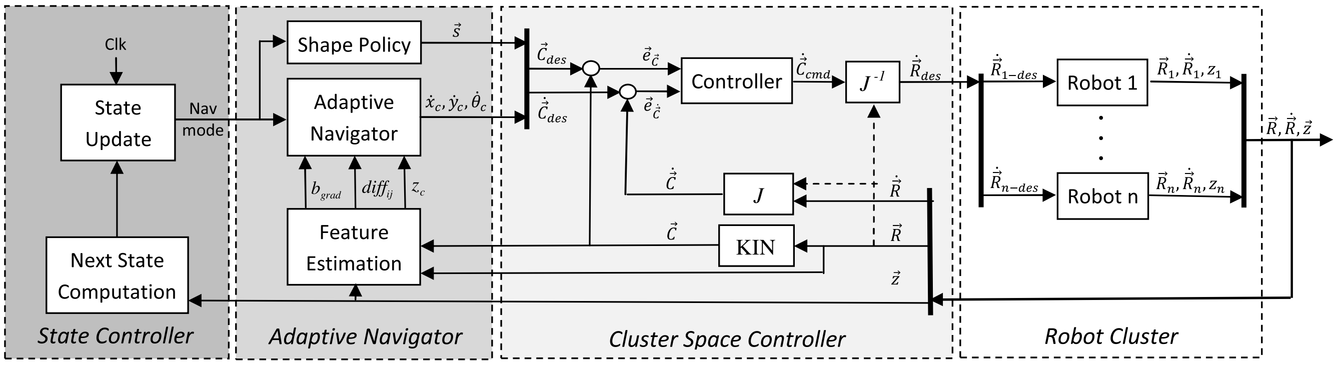

FIGURE 1. The Adaptive Navigation Layered Control Architecture: Moving from right to left, in the lowest layer (right), individual robots execute closed loop velocity commands. The next is a formation control layer that issues robot-level velocity setpoints in order to maintain a specified cluster geometry and to drive the cluster as commanded. An adaptive navigation layer provides cluster geometric setpoints and drive commands based on scalar field measurements and the selected navigation mode. Finally, there is a state controller that manages the transitions between different navigation modes based upon the parameters of the current mission.

a scalar field. This allowed a multirobot cluster to move down/up a ridge/trench assuming that the cluster was already positioned on such a feature; it also allowed a cluster to hold position at a saddle point if such a feature was found at the end of a ridge or trench. This multilayered control architecture is shown in Fig. 1 with an additional top-level state machine that allows switching among control primitives. This allows more general navigation through a field without the constraint of our prior work requiring the cluster to be already positioned on the feature of interest. Given this, the extended control architecture enables the ability to sense, compute, and move within a scalar field, switching primitives via a state-based strategy to achieve the desired goal.

To date, the control primitives have been designed using a minimal number of robots, instantaneous information, and reactive bang-bang control laws. Without question, performance and robustness can be improved with additional robots, temporal filtering, and more sophisticated controllers. The simple approach used here, however, serves as an implemented baseline which to our knowledge is the first experimental demonstration of ridge/trench descent/ascent and saddle point positioning.

The control architecture is implemented in four layers, as seen in Fig. 1. The robots themselves implement closed loop velocity control based on velocity commands issued by the cluster control layer. The cluster control layer issues those commands in order to maintain a prescribed cluster formation (shape and size, using robot position feedback) and to move the aggregate formation as specified by the AN layer. The AN layer implements the control strategy appropriate to the selected AN maneuver, using environmental sensor and robot position data to compute the local scalar field characteristics required for the control law. Finally, the state machine selects the appropriate maneuver to be executed based on the overall navigation objective.

A. ROBOT CONTROL LAYER

In general, the robot controller first converts specified robot-level translational and rotational velocity commands to wheel-specific rotational speed commands via the robot's vehicle-to-wheel inverse kinematics function. Wheel speed is implemented via a closed loop PID function that uses wheel encoder information to estimate actual speed. Specific to the experiments performed for this paper, robot layer control is computed on-board the robot, and a three wheeled omnidrive robot is used.

B. CLUSTER SPACE CONTROL LAYER

Formation control is implemented via the cluster space control methodology, an operational space control approach in which the multirobot formation is represented as a virtualized full degree-of-freedom articulating mechanism [26]. The controller accepts position or velocity specifications for cluster level parameters, which are formally defined as the cluster position, the cluster shape, and the relative orientation of robots with respect to the cluster. These cluster level parameters, defined by $\vec{C}$, and their derivatives constitute the cluster state space for the system. Kinematic transforms for n robots with m degrees of freedom, provided in Equations (1) and (2), relate these variables to conventional robot-specific position variables, defined by $\vec{R}$, and their derivatives. For a planar cluster, $\vec{R}$ contains x_i, y_i, and θ_i for $i = 1 : n$ robots.

$$\vec{C} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{mn} \end{pmatrix} = KIN(\vec{R}) = \begin{pmatrix} g_1(r_1, r_2, \cdots, r_{mn}) \\ g_2(r_1, r_2, \cdots, r_{mn}) \\ \vdots \\ g_{mn}(r_1, r_2, \ldots, r_{mn}) \end{pmatrix} \quad (1)$$

$$\dot{\vec{C}} = J(\vec{R})\dot{\vec{R}} \quad (2)$$

Robot position variables are transformed to cluster variables via forward kinematic relationships in order to generate cluster space errors. Compensation commands are computed in the cluster space, which leads to smooth, well-behaved motions of the formation. For the resolved rate controller used in these experiments, the cluster velocity commands are transformed to robot-specific velocity set-points via the inverse Jacobian function, $J^{-1}(\vec{R})$;[1] this function depends on the pose of the system and must therefore be updated continuously.

Lyapunov stability criteria for this control technique given arbitrary goal trajectories, such as those issued by the adaptive navigation layer, are derived in [28]. Because a state machine switches between these individually stable controllers, overall stability is still a challenge since poor switching choices can drive a system unstable even if individual controllers are stable. This can be addressed via a "slow switching" strategy in which the system is allowed to settle prior to switching to a different control primitive. Less conservative approaches are often possible; for example, we have proven switched stability in cases where we have moved robots between multiple sub-clusters of a system under the condition that the cluster space commanded rate is less than or equal to the current rate.

Furthermore, obstacle avoidance at both the robot and cluster level can be implemented, although such capability was not used for the experiments presented here. Inter-robot collisions are prevented by the formation controller, barring initial conditions with extremely poor formations. The formation would be singular if any two robots were collocated, however in practice this cannot physically occur, and the tracking system used for the presented experiments generates positioning errors smaller than the size of the robot chassis, so it is unlikely to occur as a result of sensor error as well.

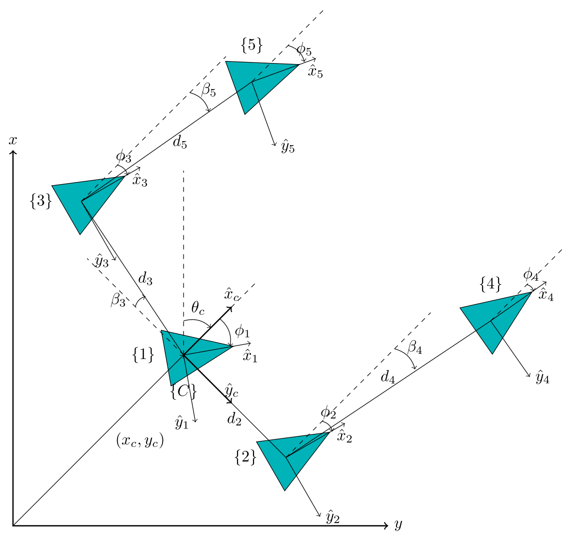

FIGURE 2. **Five Robot Cluster Pose Definition: Cluster space pose variables for a five robot cluster with the cluster frame assigned to the rear-center robot and the position of the other robots defined serially within two different chains of the virtual mechanism.**

The ridge/trench descent/ascent and saddle point positioning capabilities demonstrated in this article use a multidirectional differential control strategy requiring a five robot cluster, as shown in Fig. 2. As a planar, five robot system, it has 15 degrees of freedom, all of which are controlled during AN operations. The cluster space pose vector, $\vec{C}$, includes the position and orientation of the cluster frame (x_c, y_c, θ_c), seven distance and angle shape variables, $(d_2, d_3, d_4, d_5, \beta_3, \beta_4, \beta_5)$, and five relative robot orientation variables, ϕ_i, for $i = 1\text{-}5$. For the AN experiments performed, desired cluster parameters were held constant, with $\phi_i = 0$, $\beta_i = 0$, and constant values for the d_i parameters were selected based upon the size of the feature being navigated.

[1]Full dynamic control is possible through the use of a controller that computes force/torque compensations, in which case a Jacobian transpose transform is used to convert these compensations to robot-specific force and torque inputs. For details, see [27]

Our previous work defined the cluster variables, however the full mathematical framework was not required as the simulation assumed a perfect formation at all times. This is the first time this particular cluster formation has been implemented in full, so this work included the derivation of the applicable forward kinematics and inverse Jacobian. The forward position kinematic functions for this cluster are provided in Appendix A.

C. DIFFERENTIAL-BASED CONTROLLERS

The Adaptive Navigation layer issues aggregate cluster translational and velocity commands in accordance with the specific control primitive associated with the AN function of interest. In general, these functions include extrema finding, contour following, ridge descent / trench ascent, and saddle point positioning. The control primitives themselves compute aggregate cluster motion commands based on estimates of local field characteristics such as the gradient or differentials across the span of the cluster. These estimates are computed in realtime based on measurements of the scalar field made by each robot.

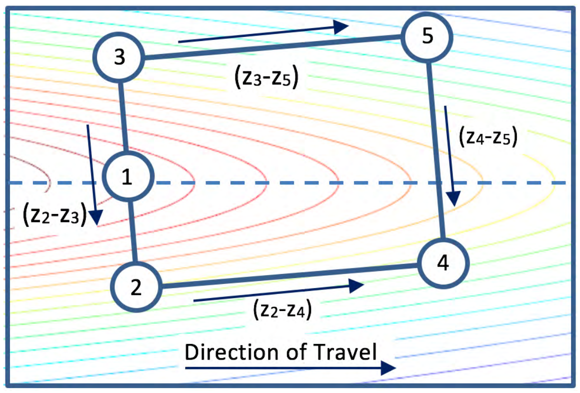

FIGURE 3. **Differential Drive Compensation Signals for Ridge Following: The cluster is straddling the ridge but has both a lateral and rotational offset. Longitudinal scalar differences indicate the desired direction of travel along the ridge. Additionally, lateral scalar differentials can be used in a differential drive strategy to align the cluster laterally and rotationally [25].**

As detailed in [25], the ridge/trench descent/ascent and saddle point MAN control primitives use differential measurements in order for the cluster to properly straddle, align with, and move with respect to the desired feature. Four robots are required for this strategy, meaning that the sensor data from these robots are used to generate the necessary differential measurements; in particular, these are robots 2 through 5 in Fig. 3.

The differential sensor measurements computed by the sensors on robot pairs 2-4 and 3-5 are $(z_2 - z_4)$ and $(z_3 - z_5)$, respectively, and are termed the "longitudinal" differentials. Conversely, sensors on robot pairs 2-3 and 4-5 are used to compute $(z_2 - z_3)$ and $(z_4 - z_5)$, termed the "lateral" differentials. Qualitatively, non-zero longitudinal differentials designate the desired direction of travel (e.g., down a ridge indicated by positive longitudinal differentials, and up a trench indicated by negative longitudinal differentials). Lateral differentials are used to adjust lateral displacement with respect to the ridge (the sum of the differentials indicates a net bias to one side or the other of the feature) as well as to rotationally align the cluster with the local direction of the feature (the difference of the differentials indicates angular misalignment).

Given this qualitative strategy, there is ambiguity that must be resolved. For ridge/trench following, instantaneous analysis of the sensor data from the four robots alone does not guarantee that the robots are on the feature; an additional robot, robot 1 in Fig. 3, is used to ensure that the ridge/trench is being straddled. The condition for this is that the sensor reading for robot 1 should be greater than both robots 2 and 3 for ridge, and vice versa for a trench.[2]

Given these strategic considerations, the AN control laws are given by Equations (3-5), where v_x, v_y, and ω_z are constant velocity setpoints for each velocity component, and s sets the direction of travel based upon the state of the system. In all experiments presented in this work, $v_x = 0.1\ m/s$, $v_y = 0.1\ m/s$, and $\omega_z = 0.3\ rad/s$. $d = 1$ for ridge descent, $d = -1$ for trench ascent, and $s = 1$ when the cluster is properly straddling the feature. As per the qualitative discussion of the control strategy, Equation (3) moves the cluster down the ridge or up the trench, and Equations (4) and (5) are used to laterally place and rotationally align the cluster with the feature. By changing the value of s to 0 or -1, longitudinal travel of the cluster can be halted or reversed; the rationale for this is discussed in the next subsection.

$$(\dot{x}_c)_{des} = s \times d \times v_x \{\text{sgn}[(z_2 - z_4) + (z_3 - z_5)]\} \quad (3)$$

$$(\dot{y}_c)_{des} = d \times v_y \{\text{sgn}[(z_2 - z_3) + (z_4 - z_5)]\} \quad (4)$$

$$\left(\dot{\theta}_c\right)_{des} = d \times \omega_z \{\text{sgn}[(z_4 - z_5) - (z_2 - z_3)]\} \quad (5)$$

D. STATE MACHINE-BASED CONTROLLER SWITCHING

The fourth layer of our unified control architecture is a new element of our work; a finite state machine switches between feature-specific control primitives based on task performance and mission objectives. In [29], it was used to implement a sequence of primitive maneuvers in order to achieve objectives such as methodically exploring local extrema in a region of interest and to navigate between points while maintaining a minimum service level (such as wireless communication link signal strength).

[2]Satisfying this criteria guarantees proper straddling of the feature for idealized features. Margins can be incorporated to account for noise. Interestingly, we note that failure to satisfy this criteria does not guarantee that the cluster is not straddling the feature.

For the experiments presented here, we use this control law/mode switching capability to address feature ambiguity in ridge/trench following, thereby improving robustness of this objective. The case that the cluster may no longer be straddling the ridge is indicated when either of the criterion below is violated:

$$\text{Ridge Criteria: } (z_1 > z_2) \text{ AND } (z_1 > z_3) \quad (6)$$

$$\text{Trench Criteria: } (z_1 < z_2) \text{ AND } (z_1 < z_3) \quad (7)$$

If violated, travel down/up the ridge/trench should be ceased. Strategically, longitudinal travel can be halted or even reversed by changing the controller state, and therefore the value of the parameter s to 0 or -1, respectively. By halting or reversing longitudinal travel, the lateral and alignment controllers can continue to move the cluster in an attempt to unambiguously re-acquire the ridge/trench. For our experiments, we used $s = -1$ in order for the cluster to travel up the scalar surface until proper feature acquisition is achieved.

Given this strategic approach, the Next State Computation block in Fig. 1 assigns the value of 1 or -1 to the s variable in Equation (3) based on realtime sensor data and the appropriate Criteria from Equations (6) or (7); if the criteria is true, $s = 1$, and if false, $s = -1$. This value is propagated almost immediately given that the State Update process occurs at the servo rate common to the State Control, Adaptive Navigation, and Cluster Control layers; for the experiments described in Section IV, this rate was approximately 10 Hz.

III. DESCRIPTION OF TESTBED

Prior work implemented the new MAN controllers via simulation, however these simulations were limited in scope, lacking modeling of individual robot dynamics, and missing real world effects like sensor noise, communication drop outs, and noise in the scalar field itself. As our ultimate goal is to incorporate them into our field-grade multirobot systems in order to perform real applications, the next logical step was to implement and verify the controllers in an experimental setting. There are a variety of groups conducting multirobot research in various forms, many of which have developed their own testbeds. Some examples include the Kilobot testbed, designed to allow for a very large swarms of affordable robots [30]; the Robotarium, a swarm testbed designed to allow researchers to operate it remotely [31]; and IRIS, a multirobot testbed designed to facilitate internet of things research, complete with sophisticated communications equipment [32].

For this step in our multirobot adaptive navigation research, we used a simplified, small-scale indoor multirobot testbed to evaluate and verify the functionality of the control techniques. This testbed uses small, custom-built omnidrive robots, one of which is shown in Fig. 4; twelve of these robots are available, although only five were required to demonstrate the capabilities presented in this article. Each robot has two levels of computation, a linux based processor for managing high level tasks like communication with a

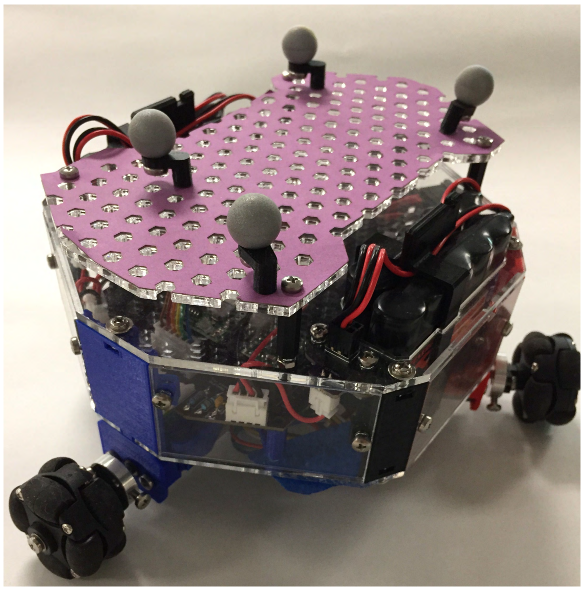

FIGURE 4. One of the five robots used to conduct the experiments.

stationary workstation, and a lower level microcontroller for handling sensing and encoder feedback.

A distinguishing element of this testbed is the manner in which it provides measurable scalar fields that can be tailored in size and shape in order to demonstrate capabilities of interest. This is done by having the robots drive over a surface of printed sheets with greyscale patterns representing the scalar value of interest. This enables the creation of fields with all of the critical types of scalar field features and provides knowledge of the "truth" for the field, which is critical for experimental verification. Each robot has a reflectivity sensor in its bottom and measures the greyscale scalar field value at its location. There is some noise introduced by the quality of the print, however, it does not significantly impact the experimental results as long as the robot cluster is sized appropriately, and the features incorporate a large range of greyscale values.

The robots operate in a field approximately 5 m by 5 m in size, and are tracked by an Optitrack, camera-based, six degree of freedom, rigid body localization system. The tracking performance of the system was characterized by distributing five robots throughout the workspace and recording position data over time. The region of worst performance had a standard deviation of 7.26 mm and 3.66 mm in the x and y directions respectively. The regions with the best tracking quality produced values of 0.058 mm and 0.069 mm.

The robots accept robot-level translation and rotation velocity commands and implement these via on-board closed loop velocity control. The robots communicate via WiFi with a workstation that accepts the sensed scalar field values from each robot as well as position data for each robot from the Optitrack system, and acts as a centralized controller. The Matlab/Simulink-based controller computes compensation commands at a 10 Hz servo rate and issues velocity commands to each robot. Poor network performance was another source of error throughout these tests, as packet loss was frequently high enough to impact the quality of the formation control, however it was still good enough to accomplish the navigation objectives.

The design of this testbed is detailed in [33]; the description in that article reviews the mechanical and embedded system design of the robot, characterizes the reflectivity sensor as a mechanism for measuring arbitrarily printed greyscale scalar fields, describes the implementation of the control software and the internet-based data distribution system, and presents an integrated virtual reality display to visualize scalar fields as they are explored.

IV. EXPERIMENTAL RESULTS

Multiple experiments were conducted for several different ridge/trench configurations. Shapes used included a linear trench, a parabolic trench, a wide cubic ridge, and a significantly narrower cubic ridge terminating in a saddle point. Several trials with different initial conditions were tested for each experiment, including configurations where the robots began on the feature, off the feature, aligned and misaligned with the ultimate direction of travel, as well as with different initial cluster poses.

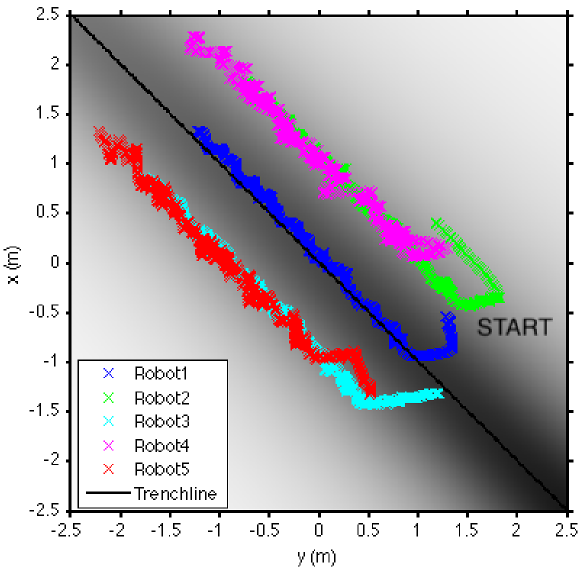

FIGURE 5. Experimental data for five robots navigating up a trench with a linear path, superimposed upon a rendering of the grey scale scalar field.

A. LINEAR TRENCH

The first scalar field is a trench that follows a linear route upward. According to our sensor convention, darker portions of the printout constitute a lower reading, and lighter portions represent higher values; therefore, the robot cluster starts near the darkest end of the trench, and travels up to the lighter end. This behavior is shown in Fig. 5, which plots the location of all five robots in the cluster as it ascends the trench. The absolute value of the tracking error for this test is presented in Fig. 6. Excluding the time spent aligning with the feature,

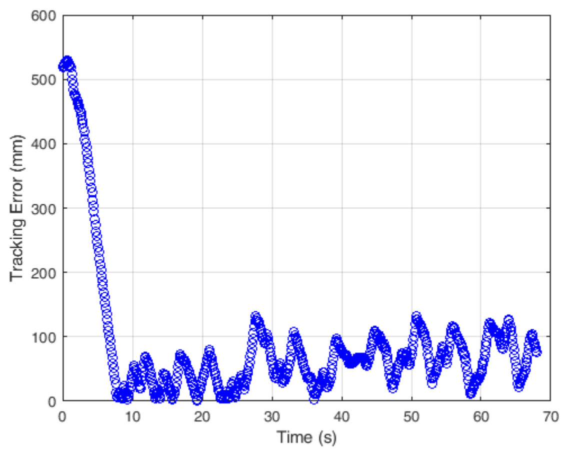

FIGURE 6. Time history for the absolute value of the tracking error for the trial depicted in Fig. 5.

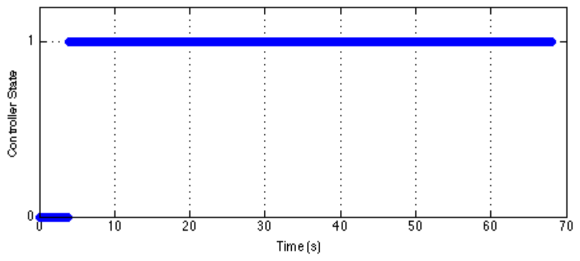

FIGURE 7. Controller state as the robots travel the path in Fig. 5.

the RMS tracking error of this trial was 82.14 mm. The initial position of the cluster was offset enough to violate the criteria in Equation 7, so rather than immediately ascending the trench, the cluster first aligns with the feature, then continues upward. This is indicated by Fig. 7 which contains the value of s over time, and shows that this initial correction lasts for about four seconds.

As an indication of formation control quality during this maneuver, Fig. 8 shows the time history for each of the intra-robot separation distances, d_i, which were all commanded to a setpoint of 0.7 m. Table 1 summarizes the performance of these, and the formation angle parameters, β_i, by providing the RMS error for each parameter. Note that all of these errors are small, indicating the formation performance was more than adequate for these experiments.

Fig. 9 displays the paths of the clusters[3] for five trials on the linear trench, each with different initial conditions, with the cluster starting off the feature for several of these. As seen from the figures the clusters of robots consistently move to (if necessary) and then ascend the trench. The RMS error was calculated for each run using the tracking error time history once the controller first enters the on-ridge condition ($s = 1$). This tracking error was computed by determining the distance between robot 1 and the principle line of the trench. The RMS error values for each trial are listed in Table 2. Each of these

[3]As specified in Fig. 2, the cluster's position is defined to be at origin of the cluster frame, which for this definition is at the location of Robot 1.

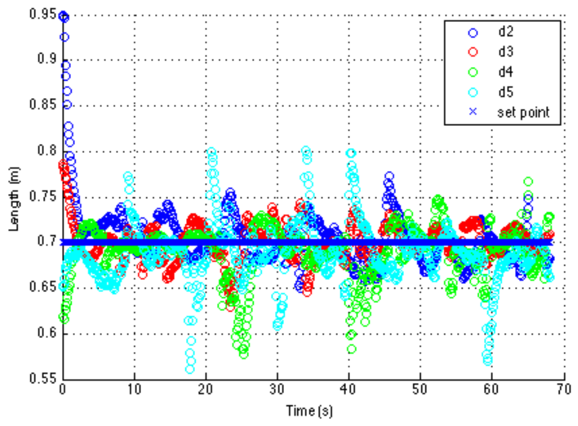

FIGURE 8. Time history of the d_i values for the test presented in Fig. 5.

TABLE 1. Cluster parameter error for the straight trench test presented in Fig. 5.

	RMS Error
β_3 (rad)	0.0715
β_4 (rad)	0.0691
β_5 (rad)	0.0446
d_2 (m)	0.0327
d_3 (m)	0.0199
d_4 (m)	0.0299
d_5 (m)	0.0339

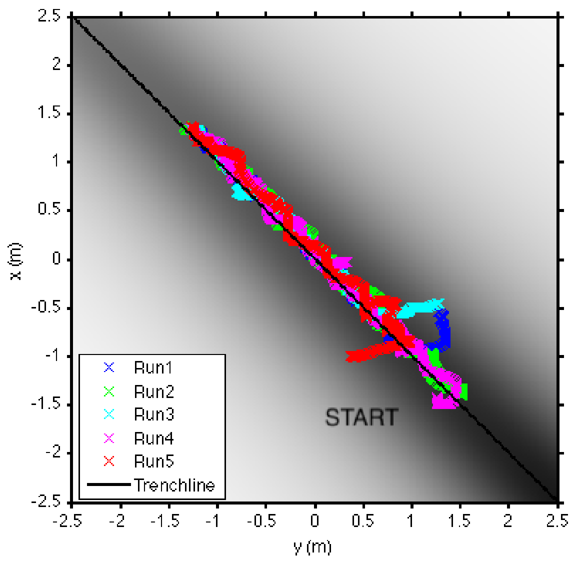

FIGURE 9. Experimental data for the cluster center from multiple runs as robots navigate up a trench with a linear path, superimposed upon a rendering of the grey scale scalar field. The blue path is the same run depicted in Fig. 5.

is less than an order of magnitude lower than the total width of the cluster ($d_1 + d_2 = 1.4$ m) as it straddles the ridge.

B. PARABOLIC TRENCH

The next set of tests required the robot cluster to track a trench with a parabolic shape. This demonstrates an additional challenge for the controller, as it must follow a changing spatial

TABLE 2. **The RMS trenchline tracking error for all five trials plotted in Fig. 9.**

	RMS Error (mm)
Trial 1	82.14
Trial 2	84.82
Trial 3	89.80
Trial 4	83.18
Trial 5	108.48

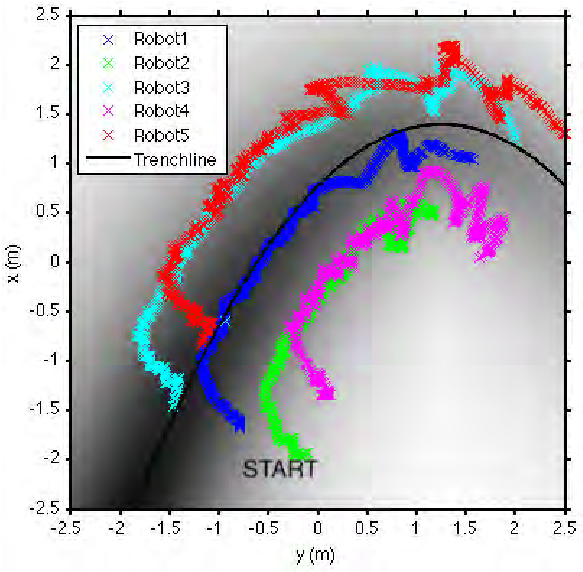

FIGURE 10. **Experimental data for five robots navigating up a trench with a parabolic path, superimposed upon a rendering of the grey scale scalar field.**

TABLE 3. **Cluster parameter error for the parabolic trench test presented in Fig. 10.**

	RMS Error
β_3 (rad)	0.1803
β_4 (rad)	0.0864
β_5 (rad)	0.1385
d_2 (m)	0.0304
d_3 (m)	0.0617
d_4 (m)	0.0357
d_5 (m)	0.0585

input. Fig. 10 depicts a single test run including the positions of each individual robot in the cluster as they travel, and Table 3 displays the RMS error of the cluster parameter set points throughout this test. Fig. 11 shows the paths of the cluster for several tests with varying initial conditions. In all cases the performance is better when the trenchline is straighter, which is expected since this condition is easier to track.

The RMS tracking error values for this set of experiments are displayed in Table 4, which are again small when compared to the cluster width of $2d_i = 1.4\ m$. This indicates that the tracking was not quite as good as the linear trench,

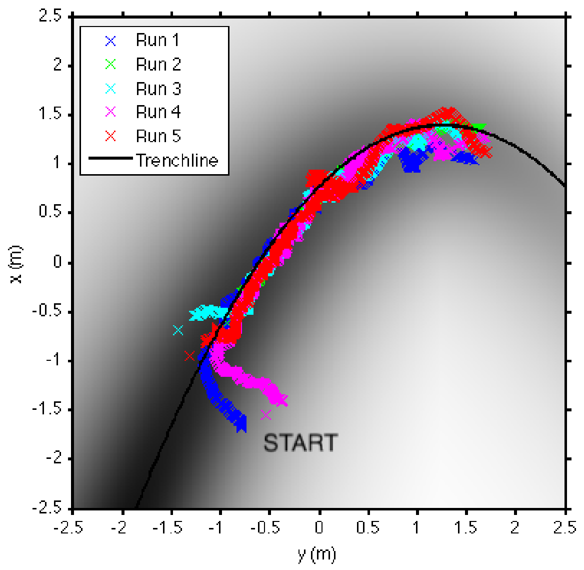

FIGURE 11. **Experimental data for the cluster center from multiple runs as robots navigate up a trench with a parabolic path, superimposed upon a rendering of the grey scale scalar field.**

TABLE 4. **The RMS trenchline tracking error for all five trials plotted in Fig. 11.**

	RMS Error (mm)
Trial 1	163.95
Trial 2	80.93
Trial 3	119.27
Trial 4	128.12
Trial 5	103.95

likely due to the ramp input to the controller resulting from the curved path. Nevertheless, this error is still much smaller than the width of the cluster, indicating that the robots successfully followed the feature.

C. WIDE CUBIC RIDGE

The third set of tests were run on a ridge with a wide profile and a principle path with a cubic shape. This required a larger cluster size ($d_i = 0.8\ m$) to ensure it could track the feature despite noise in the sensor data. Fig. 12 depicts the paths of all five robots for a single trial. The formation keeping performance for this trial can be found in Table 5. The paths of the cluster for five different trials are presented in Fig. 13. It can be discerned from Figs. 12 and 13 that the highest tracking error occurs near the inflection point at the midpoint of the feature caused by the cubic shape.

The RMS tracking error values after reaching the ridgeline for these runs are listed in Table 6. Because both this feature and the associated cluster of robots were wider than the others, higher tracking errors can be tolerated without losing the crest of the ridge. Once again, the error was much lower than the width of the cluster, indicating that the ridgeline was followed successfully.

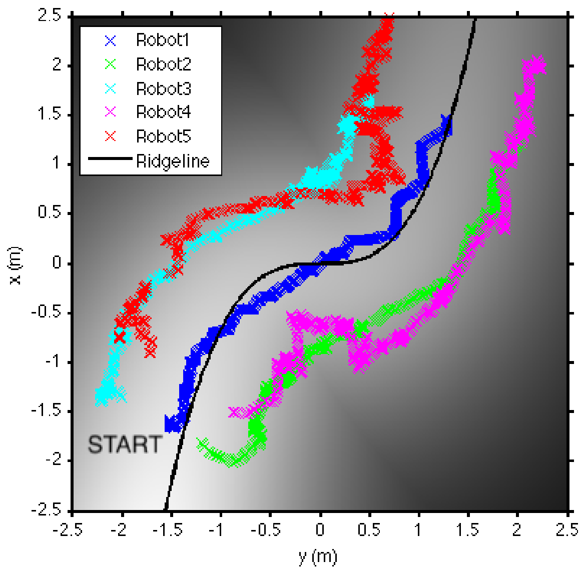

FIGURE 12. Experimental data for five robots navigating down a ridge with a cubic path, superimposed upon a rendering of the grey scale scalar field.

TABLE 5. Cluster parameter error for the wide ridge test presented in Fig. 12.

	RMS Error
β_3 (rad)	0.0572
β_4 (rad)	0.0515
β_5 (rad)	0.3675
d_2 (m)	0.0478
d_3 (m)	0.0319
d_4 (m)	0.0517
d_5 (m)	0.0596

D. NARROW CUBIC RIDGE WITH SADDLE POINT

The final set of tests was conducted on another ridge with a cubic shape, however this one is much narrower, and terminates in a saddle point. The cluster size used for these tests was smaller, with $d_i = 0.5\ m$, so that it could track the smaller feature more closely; see Table 7 for formation performance. The paths of the robots are presented in Figs. 14 and 15. The average ridge tracking RMS error was 70.85 mm, and this smaller value is expected given the size of the cluster.

In all cases the robot cluster settles on the saddle point of the scalar field. There is some oscillation about the point due to the discrete nature of the velocity controller. The time responses in Fig. 16 provide the scalar differentials in the cluster $\hat{x}$ direction. As we would expect, these values are mostly positive until we reach the saddle point, which occurs at around 70 s into the trial. While there is substantial noise from the sensors, the results do show that after reaching the saddle region, the differentials oscillate about zero, as we would expect given the robot behavior displayed in Figs. 14 and 15.

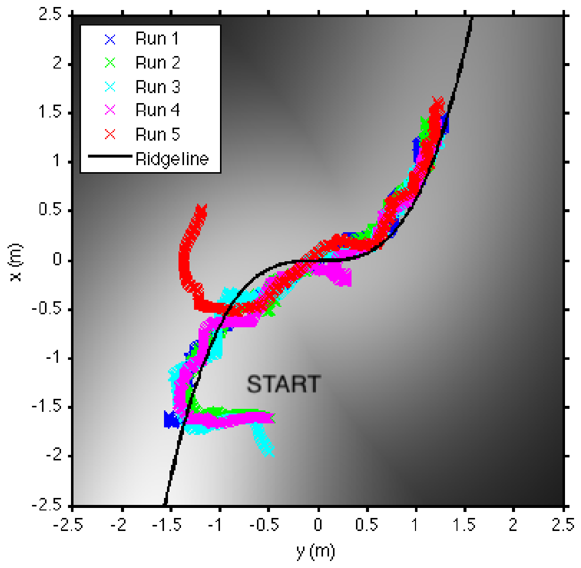

FIGURE 13. Experimental data for the cluster center from multiple runs as the robots navigate down a ridge with a cubic path, superimposed upon a rendering of the grey scale scalar field.

TABLE 6. The RMS ridgeline tracking error for all five trials plotted in Fig. 13.

	RMS Error (mm)
Trial 1	120.78
Trial 2	225.58
Trial 3	256.40
Trial 4	121.80
Trial 5	325.13

TABLE 7. Cluster parameter error for the test presented in Fig. 14 with a cubic ridge terminating in a saddle point.

	RMS Error
β_3 (rad)	0.2298
β_4 (rad)	0.0981
β_5 (rad)	0.1689
d_2 (m)	0.0217
d_3 (m)	0.0394
d_4 (m)	0.0311
d_5 (m)	0.0696

TABLE 8. The RMS ridgeline tracking error for all five trials plotted in Fig. 15.

	RMS Error (mm)
Trial 1	62.75
Trial 2	56.87
Trial 3	99.19
Trial 4	57.82
Trial 5	77.64

E. SUMMARY OF PERFORMANCE AND ERROR SOURCES

In general, tracking errors arose from several sources. To begin, the trajectories being followed varied, requiring

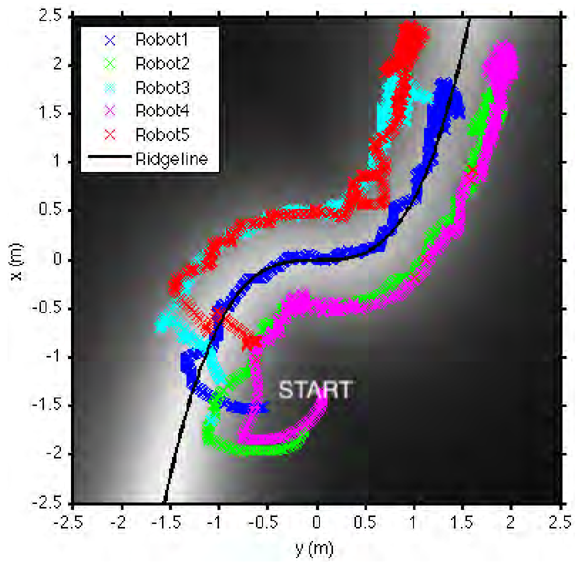

FIGURE 14. Experimental data for five robots navigating down a ridge with a cubic path until they come to rest at a saddle point, superimposed upon a rendering of the grey scale scalar field.

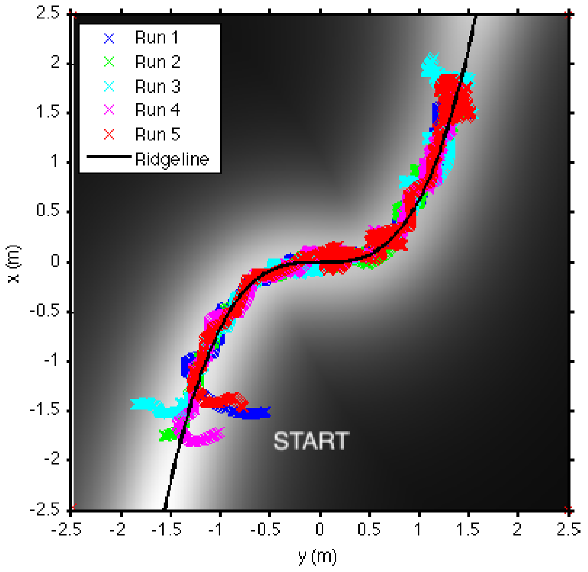

FIGURE 15. Experimental data for the cluster center from multiple runs as the robots navigate down a ridge with a cubic path until they come to rest at a saddle point, superimposed upon a rendering of the grey scale scalar field. In each case, the robot cluster oscillates around the location of the saddle point.

the controllers to track the equivalent of a ramp input. In addition, the tight spacing of the cluster led to amplified cluster space errors for small errors in robot position. Limitations on print quality led to banding on the scalar field printout that injected spatial noise into the scalar field on the order of tenths of a scalar unit. Furthermore, sensor noise included scalar sensor errors on the order of thousandths of a scalar unit and robot tracking errors with standard deviation of errors in range that were no more than 7.26 mm. Finally, although the control loop executed at 10 Hz, occasional

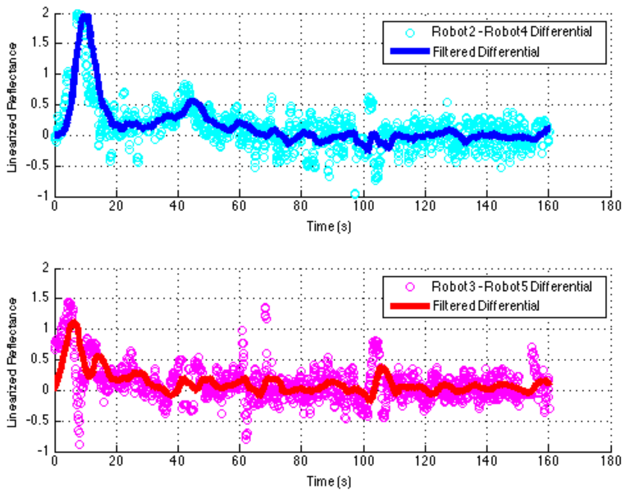

FIGURE 16. Time histories of the scalar differentials between robots 2 and 4 (top), and robots 3 and 5 (bottom), for the run presented in Fig. 14. Includes both the unfiltered values used in the control computation, and the result after filtering with the response with a 50 sample moving average.

delays and packet dropouts resulted in servo loop latencies of up to 1 sec due to traffic on the local network. Ongoing work with the testbed includes further characterization of the sensors, better printouts with less banding, and improved communication. These improvements would allow us to set faster robot velocities, tighter formations, and track more complicated features.

V. CONCLUSION

Multirobot Adaptive Navigation is a powerful technique for navigating and exploring environments, and has many potential applications. To our knowledge, we have presented here the first experimental verification of adaptively descending scalar ridges, climbing trenches, and holding on saddle points using a multirobot formation; this was a large step forward from our past work, as it required full development of the five robot formation, and introduced many real-world effects. These strategies are powerful capabilities for a number of applications, including navigating to areas impacted by pollutants, making minimum exposure approaches, et cetera. We have also extended our unified multirobot adaptive navigation control architecture with a state machine technique for switching between different adaptive navigation controllers based upon sensed local field characteristics and the desired navigation tasks. Finally, we have described our experimental testbed, analyzed the results of our experiments, and characterized the resulting performance.

In ongoing work we are using adaptive sizing of the cluster based on the nature of the field, state-based switching between navigation strategies to support more complex missions, and the ability to navigate three dimensional scalar fields. We will also be pursuing field demonstrations of these techniques in order to validate their capability in real world applications.

APPENDIX A
FORWARD CLUSTER KINEMATICS

The forward kinematics for the cluster formation presented in this work are listed in this section. Note that while the fractions inside the inverse tangent functions could mathematically go to infinity, this does not occur in practice because it would require multiple robots to be in exactly the same location. Additionally, the noise in the tracking system is smaller than the size of the robot chassis, so this condition did not, and could not occur during the experiments. Throughout the experiments, these values were computed using the MATLAB *atan*2() function to provide additional robustness, and account for Cartesian quadrants.

$$x_c = x_1 \tag{8}$$
$$y_c = y_1 \tag{9}$$
$$\theta_c = \tan^{-1}\left(\frac{x_1 - x_2}{y_2 - y_1}\right) \tag{10}$$
$$\phi_1 = \theta_1 - \theta_c \tag{11}$$
$$d_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{12}$$
$$\phi_2 = \theta_2 - \theta_c \tag{13}$$
$$d_3 = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \tag{14}$$
$$\phi_3 = \theta_3 - \theta_c \tag{15}$$
$$\beta_3 = \tan^{-1}\left(\frac{x_3 - x_1}{y_1 - y_3}\right) - \theta_c \tag{16}$$
$$d_4 = \sqrt{(x_4 - x_2)^2 + (y_4 - y_2)^2} \tag{17}$$
$$\phi_4 = \theta_4 - \theta_c \tag{18}$$
$$\beta_4 = \tan^{-1}\left(\frac{y_4 - y_2}{x_4 - x_2}\right) - \theta_c \tag{19}$$
$$d_5 = \sqrt{(x_5 - x_3)^2 + (y_5 - y_3)^2} \tag{20}$$
$$\phi_5 = \theta_5 - \theta_c \tag{21}$$
$$\beta_5 = \tan^{-1}\left(\frac{y_5 - y_3}{x_5 - x_3}\right) - \theta_c \tag{22}$$

APPENDIX B
SCALAR FIELD EQUATIONS

The general equation for the scalar fields used is:

$$z = \frac{h(x_s y + 1 + y_{n1})(x_s x + 1)}{(d/d_{hv})^2) + 1}$$

where h is the height of the scalar field at the origin, y_s is the slope in the y direction, y_{n1} is the nonlinearity in the y direction, x_s is the slope in the x direction, d is the minimum distance from a point (x, y) to the centerline of the feature, and d_{hv} is half the value distance for the feature (the distance from the feature's centerline which cuts the feature's height in half).

All scalar fields were created in Matlab with a domain and range of ± 2. The z values were used to create greyscale ranging from 0 (black) to 1 (white). In order to reduce sensor saturation, the minimum and maximum grey values were sometimes limited, as seen in the table below. Also in the table is the resolution used for the grascale colormap.

TABLE 9. The parameters for the scalar field equations for each experiment.

Parameter	Exp 1	Exp 2	Exp 3	Exp 4
h	-8	-5	1	1
x_s	0	0	-0.075	-0.075
x_{nl}	0	0	0	$(1-u(x-1))(0.25)(x+1)^2$
y_s	-0.125	0.2	-0.075	-0.075
d_{hv}	0.7	0.7	1.5	0.5
Line Eqn.	$x=-y$	$x=-0.5y^2-y+0.5$	$x=y^3$	$x=y^3$

TABLE 10. Greyscale ranges for the scalar fields.

Scalar Field	Grey min	Grey max	Grey resolution
Exp 1	0.1	1	0.0001
Exp 2	0.2	0.95	0.001
Exp 3	0.05	1	0.0001
Exp 4	0.05	1	0.001

During the printing process the size of the fields were increased to approximately $\pm 2.5\ m$.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Automatic Robot Vehicles* New York, NY, USA: Springer, 1986, pp. 396–404.

[2] A. Melingui, T. Chettibi, R. Merzouki, and J. B. Mbede, "Adaptive navigation of an omni-drive autonomous mobile robot in unstructured dynamic environments," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2013, pp. 1924–1929.

[3] S. Sharma and R. Tiwari, "A survey on multi robots area exploration techniques and algorithms," in *Proc. Int. Conf. Comput. Techn. Inf. Commun. Technol. (ICCTICT)*, Mar. 2016, pp. 151–158.

[4] D. M. Helmick, A. Angelova, M. Livianu, and L. H. Matthies, "Terrain adaptive navigation for mars rovers," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2007, pp. 1–11.

[5] S. M. Amin, E. Y. Rodin, M. K. Meusey, T. W. Cusick, and A. Garcia-Ortiz, "Evasive adaptive navigation and control against multiple pursuers," in *Proc. Amer. Control Conf.*, vol. 3, Jun. 1997, pp. 1453–1457.

[6] A. Filippoupolitis, G. Gorbil, and E. Gelenbe, "Autonomous navigation systems for emergency management in buildings," in *Proc. IEEE GLOBECOM Workshops (GC Wkshps)*, Houston, TX, USA, Dec. 2011, pp. 1056–1061.

[7] A. S. Matveev, H. Teimoori, and A. V. Savkin, "Navigation of a unicycle-like mobile robot for environmental extremum seeking," *Automatica*, vol. 47, no. 1, pp. 85–91, 2011.

[8] W. Li, J. A. Farrell, S. Pang, and R. M. Arrieta, "Moth-inspired chemical plume tracing on an autonomous underwater vehicle," *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 292–307, Apr. 2006.

[9] E. Burian, D. Yoerger, A. Bradley, and H. Singh, "Gradient search with autonomous underwater vehicles using scalar measurements," in *Proc. Symp. Auton. Underwater Vehicle Technol.*, Jun. 1996, pp. 86–98.

[10] C. Zhang, A. Siranosian, and M. Krstic, "Extremum seeking for moderately unstable systems and for autonomous vehicle target tracking without position measurements," *Automatica*, vol. 43, no. 10, pp. 1832–1839, Oct. 2007.

[11] A. S. Matveev, M. C. Hoy, A. M. Anisimov, and A. V. Savkin, "Tracking of deforming environmental level sets of dynamic fields by a nonholonomic robot without gradient estimation," in *Proc. 10th IEEE Int. Conf. Control Automat. (ICCA)*, Jun. 2013, pp. 1754–1759.

[12] A. S. Matveev, M. C. Hoy, K. Ovchinnikov, A. Anisimov, and A. V. Savkin, "Robot navigation for monitoring unsteady environmental boundaries without field gradient estimation," *Automatica*, vol. 62, pp. 227–235, Dec. 2015.

[13] X. Kang and W. Li, "Moth-inspired plume tracing via multiple autonomous vehicles under formation control," *Adapt. Behav.*, vol. 20, no. 2, pp. 131–142, Apr. 2012.

[14] V. Gazi, F. Fidan, L. Marques, and R. Ordonez, "Robot swarms: Dynamics and control," in *Mobile Robots for Dynamic Environments*, E. F. Kececi and M. Ceccarelli, Eds. New York, NY, USA: ASME, 2015, ch. 4, pp. 79–107.

[15] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[16] N. A. Atanasov, J. L. Ny, and G. J. Pappas, "Distributed algorithms for stochastic source seeking with mobile robot networks," *J. Dyn. Syst., Meas., Control*, vol. 137, no. 3, 2015, Art. no. 031004.

[17] W. Wu and F. Zhang, "Cooperative exploration of level surfaces of three dimensional scalar fields," *Automatica*, vol. 47, no. 9, pp. 2044–2051, 2011.

[18] I. Triandaf and I. B. Schwartz, "A collective motion algorithm for tracking time-dependent boundaries," *Math. Comput. Simul.*, vol. 70, no. 4, pp. 187–202, 2005.

[19] W. Wu, D. Chang, and F. Zhang, "A bio-inspired robust 3D plume tracking strategy using mobile sensor networks," in *Proc. 52nd IEEE Conf. Decis. Control*, Dec. 2013, pp. 4571–4578.

[20] S. Li, R. Kong, and Y. Guo, "Cooperative distributed source seeking by multiple robots: Algorithms and experiments," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 6, pp. 1810–1820, Dec. 2014.

[21] A. Joshi, T. Ashley, Y. R. Huang, and A. L. Bertozzi, "Experimental validation of cooperative environmental boundary tracking with on-board sensors," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2009, pp. 2630–2635.

[22] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *Proc. Amer. Control Conf.*, vol. 5, Jun. 2005, pp. 3500–3505.

[23] J. Acain, C. Kitts, T. Adamek, K. Ebadi, and M. Rasay, "A multi-robot testbed for adaptive sampling experimentation via radio frequency fields," in *Proc. ASME/IEEE Int. Conf. Mech. Embedded Syst. Appl.*, vol. 9, Aug. 2015, pp. 1–7.

[24] T. Adamek, C. A. Kitts, and I. Mas, "Gradient-based cluster space navigation for autonomous surface vessels," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 1, pp. 506–518, Apr. 2015.

[25] C. A. Kitts, R. T. McDonald, and M. A. Neumann, "Adaptive navigation control primitives for multirobot clusters: Extrema finding, contour following, ridge/trench following, and saddle point station keeping," *IEEE Access*, vol. 6, pp. 17625–17642, 2018.

[26] 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.

[27] I. Mas and C. A. Kitts, "Dynamic control of mobile multirobot systems: The cluster space formulation," *IEEE Access*, vol. 2, pp. 558–570, 2014.

[28] I. Mas and C. Kitts, "Obstacle avoidance policies for cluster space control of nonholonomic multirobot systems," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 6, pp. 1068–1079, Dec. 2012.

[29] R. McDonald, "Techniques for adaptive navigation of multi-robot clusters," M.S. thesis, Dept. Mech. Eng., Santa Clara Univ., Santa Clara, CA, USA, 2017.

[30] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2012, pp. 3293–3298.

[31] D. Pickem *et al.*, "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1699–1706.

[32] J. A. Tran *et al.*, "Intelligent robotic iot system (iris)testbed," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–9.

[33] S. Tomer *et al.*, "A low-cost indoor testbed for multirobot adaptive navigation research," in *Proc. IEEE Aerosp. Conf.*, Mar. 2018, pp. 1–12.

ROBERT T. MCDONALD received the B.S. degree and the M.S. degree in mechanical engineering from Santa Clara University, Santa Clara, CA, USA, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in mechanical engineering. He was a Teaching Assistant with the Department of Mechanical Engineering, and a Research Assistant with the Santa Clara's Robotic Systems Laboratory.

CHRISTOPHER A. KITTS received the B.S.E. degree in mechanical and aerospace engineering from Princeton University, the M.P.A. degree in international and defense policy from the University of Colorado, and the M.S. degree in aeronautics and astronautics and the Ph.D. degree in mechanical engineering from Stanford University. His industrial background includes service as a Satellite Constellation Mission Controller with the U.S. Air Force and work as a Computer Scientist while he was a Contractor with the NASA Ames Research Center's Computational Sciences Division. He is currently an Associate Professor of mechanical engineering, the Director of the Robotic Systems Laboratory, and the Associate Dean of the Research and Faculty Development for the School of Engineering, Santa Clara University. He is a Fellow of the American Society of Mechanical Engineers.

MICHAEL A. NEUMANN received the B.S. degree in mechanical engineering, the M.S. degree in mechanical engineering and applied mathematics, and the Ph.D. degree in mechanical engineering from Santa Clara University. He was a Teaching Assistant and a Research Assistant with the Robotic Systems Laboratory. He has also served with the Peace Corps for three years.