

2-2020

## **Mission-Oriented Multirobot Adaptive Navigation of Scalar Fields**

Robert McDonald

Follow this and additional works at: [https://scholarcommons.scu.edu/eng\\_phd\\_theses](https://scholarcommons.scu.edu/eng_phd_theses)

 Part of the [Mechanical Engineering Commons](#)

---

# Santa Clara University

Department of Mechanical Engineering

Date: February 24, 2020

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY  
SUPERVISION BY

**Robert McDonald**

ENTITLED


**Mission-Oriented Multirobot Adaptive Navigation of Scalar  
Fields**

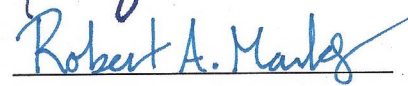
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR  
THE DEGREE OF


**DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING**


  
\_\_\_\_\_  
Thesis Advisor  
Dr. Christopher Kitts

  
\_\_\_\_\_  
Chairman of Department  
Dr. Drazen Fabris

  
\_\_\_\_\_  
Dr. Timothy Hight

  
\_\_\_\_\_  
Dr. Robert Marks

  
\_\_\_\_\_  
Dr. Behnam Dezfouli

  
\_\_\_\_\_  
Dr. Michael Neumann

# Mission-Oriented Multirobot Adaptive Navigation of Scalar Fields

By

Robert McDonald

Dissertation

Submitted in Partial Fulfillment of the Requirements  
for the Degree of Doctor of Philosophy  
in Mechanical Engineering  
in the School of Engineering at  
Santa Clara University, 2020

Santa Clara, California

# Mission-Oriented Multirobot Adaptive Navigation of Scalar Fields

Robert McDonald

Department of Mechanical Engineering  
Santa Clara University  
Santa Clara, California  
2020

## ABSTRACT

Scalar fields are spatial regions where each point has an associated physical value. These fields often contain features of interest, such as local extrema and contours with a value of significance. Traditional navigation techniques require robots to exhaustively search these regions to find the areas of significance, while adaptive navigation allows them to move directly to the points of interest based on measurements of the field taken during the navigation process. This work expands existing adaptive navigation techniques by adding a finite state machine layer to the control architecture, and using it as a discrete mode controller; the state machine allows for the sequencing of individual adaptive navigation control primitives for the purpose of enhancing performance and achieving new mission-level capabilities. For example, it has enabled improvements to existing ridge, trench, and saddle point navigators and the creation of a novel technique for navigating along scalar fronts. In both cases, experimental results demonstrated excellent tracking of the features of interest. Furthermore, mission-level capabilities were developed for low-exposure waypoint navigation and mapping contours around an extremum. These missions were evaluated through the use of 10,000 simulations with success rates of 96.95% for low exposure waypoint navigation and 87.36% for contour mapping.

---

# Acknowledgments

First and foremost I would like to thank my advisor, Dr. Chris Kitts. He has been an excellent guide and source of inspiration throughout my research. I also want to thank my doctoral committee for their feedback and insight, particularly Dr. Michael Neumann, the best coauthor anyone could ask for.

As always I owe a lot to my predecessors in adaptive navigation research, as they have paved the way for my work. My colleagues in the robotics systems lab have lent their support as well. They are an excellent community of students and scholars, and a small sample of the contributors include: Matt Condino, Scot Tomer, Danop Rajabhandharaks, Mike Rasay, and Anne Mahacek.

Last but certainly not least, I would like to thank my family and friends, and anyone else who has had to listen to me babble about robotics over the years. My wife Jill has been particularly patient and supportive while I've been chasing dreams.

A portion of this work has been used for academic publication [1][2][3], and may be used for additional publications in the future.

---

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Adaptive Navigation Literature	2
1.2	Project Statement	5
1.3	Dissertation Organization	6
<b>2</b>	<b>Systems Overview</b>	<b>7</b>
2.1	Control Architecture	7
2.1.1	Robot Control	8
2.1.2	Cluster Control	8
2.1.3	Adaptive Navigator	9
2.1.4	Finite State Machine	10
2.2	Simulation Architecture	11
2.3	Experimental Testbed	11
<b>3</b>	<b>Experimental Verification of Ridge, Trench, and Saddle Navigation</b>	<b>14</b>
3.1	Review of Ridge/Trench/Saddle Controller	14
3.1.1	Cluster Definition	15
3.1.2	Differential Control Strategy	17
3.2	Experimental Results	19
3.2.1	Linear Trench	20
3.2.2	Parabolic Trench	23
3.2.3	Wide Cubic Ridge	27
3.2.4	Narrow Cubic Ridge with Saddle Point	29
3.2.5	Summary	29

<b>4</b>	<b>Navigating Scalar Fronts . . . . .</b>	<b>33</b>
4.1	Control Strategy . . . . .	34
4.1.1	Cluster Formation . . . . .	36
4.1.2	Front Seeking . . . . .	38
4.1.3	Front Following . . . . .	40
4.1.4	State Controller . . . . .	41
4.2	Simulation Results . . . . .	42
4.3	Experimental Results . . . . .	44
4.3.1	Straight Front . . . . .	44
4.3.2	Sinusoidal Front . . . . .	48
4.3.3	Front with Changing Width . . . . .	48
4.4	Summary . . . . .	52
<b>5</b>	<b>Low-Exposure Waypoint Navigation . . . . .</b>	<b>53</b>
5.1	Cluster Formation . . . . .	53
5.2	Control Strategy . . . . .	54
5.3	Simulation . . . . .	60
5.3.1	Simulation Environment . . . . .	60
5.3.2	Results . . . . .	60
5.4	Summary . . . . .	66
<b>6</b>	<b>Mapping Contours Around an Extrema . . . . .</b>	<b>67</b>
6.1	Control Strategy . . . . .	68
6.2	Simulation Results . . . . .	74
6.2.1	Testing on a Simulated Field . . . . .	74
6.2.2	Simulating on Real-World Radiation Data . . . . .	75
6.3	Summary . . . . .	81
<b>7</b>	<b>Conclusion . . . . .</b>	<b>82</b>
7.1	Summary . . . . .	82
7.2	Future Work . . . . .	83

Bibliography . . . . . 84

Appendices . . . . . A1



---

# List of Figures

1.1	Waypoint navigation (top) is conducted by moving the vehicle to a series of points, while path planning and trajectory control prescribe the specific, continuous route to be taken (bottom). . . . .	2
1.2	The Autonomous Benthic Explorer. . . . .	3
1.3	Three automated kayaks traversing a scalar field at Stevens Creek Reservoir in Cupertino, CA . . . . .	4
1.4	The fundamental primitive features of a scalar field. . . . .	5
2.1	The multirobot adaptive navigation control architecture with state-based behavior. From right to left, we have the robot level controller, the cluster controller, the adaptive navigator, and the state controller. . . . .	7
2.2	A single omnidrive robot on a printed scalar field. . . . .	12
3.1	Generic five robot cluster pose definition for the ridge/trench/saddle navigation technique. There are two serial-chain structures defined from robot one. . . . .	17
3.2	The cluster of robots is displayed on a ridge, where it is laterally and rotationally offset from the feature. The lateral differentials center the formation on the ridge and provide rotational commands, while the longitudinal ones set the travel direction. . . . .	18
3.3	State Diagram for Ridge/Trench Following. State zero is used when off the feature, state 1 is used when on the feature . . . . .	19
3.4	The position of all five robots as they follow a linear trench, superimposed upon a rendering of the grayscale scalar field. . . . .	21
3.5	Time history for the absolute tracking error value for the trial depicted in Figure 3.4. . . . .	21
3.6	Controller state ( $s$ ) as the robots travel the path in Figure 3.4. There is only a single transition from state 0 to state 1. . . . .	22
3.7	Time history of the $d_i$ values for the test presented in Fig. 3.4. . . . .	22

3.8	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 grayscale scalar field. The blue path is the same run depicted in Fig. 3.4. . . . . .	24
3.9	Experimental data for five robots navigating up a trench with a parabolic path, superimposed upon a rendering of the grayscale scalar field. . . . .	25
3.10	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 grayscale scalar field. . . . .	26
3.11	Experimental data for five robots navigating down a ridge with a cubic path, superimposed upon a rendering of the grayscale scalar field. . . . .	27
3.12	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 grayscale scalar field. . . . .	28
3.13	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 grayscale scalar field [2]. . . . .	30
3.14	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 grayscale scalar field. In each case, the robot cluster oscillates around the location of the saddle point. . . . .	31
3.15	Time histories of the scalar differentials between robots 2 and 4 (top), and robots 3 and 5 (bottom), for the run presented in Fig. 3.13. Includes both the unfiltered values used in the control computation, and the result after filtering with the response with a 50 sample moving average. . . . .	32
4.1	(a), an idealized front, and (b), a generalized frontal region. The lines represent equally spaced contours of a scalar field, descending in value from left to right, and the highlighted region indicates where the front is located. . . . .	34
4.2	Two pairs of robots follow the edges of a scalar front. . . . .	35
4.3	An example of the robot formation on a scalar front, with example velocity commands. The contour lines are in equal scalar increments, descending from top to bottom. . . . .	36
4.4	Six Robot Cluster Pose Definition: Cluster space pose variables for a six robot cluster with intermediate frame centered between two pairs of robots, and the aggregate cluster frame centered between the two intermediate frames. . . . .	37

4.5	Each controller state is listed with the appropriate value for $s$ , and the slope criteria for each state transition. $m$ is the measured slope of the feature, $m_{min}$ is the minimum slope required for the feature to be considered a front, $m_{hyst}$ is a higher slope requirement for exiting hysteresis states, and $s$ is a variable used in the front following controller. . . . .	41
4.6	A three dimensional rendering of the scalar front used for simulated front following. . . . .	43
4.7	An overhead view of six robots navigating the scalar front; they start off the feature, climb up to it, travel in the positive $y$ until the front ends, then travel back in the $-y$ direction. . . . .	43
4.8	The time history of the cluster and subcluster slopes, as compared to the desired values. The vertical lines indicate the points at which the robot cluster is commanded to reverse direction. . . . .	44
4.9	Experimental data for six robots navigating along a front with a linear path, superimposed upon a rendering of the grayscale scalar field. . . . .	45
4.10	Performance plot for $d_1, d_2, d_5, d_6$ for the experiment displayed in Figure 4.9. . . . .	46
4.11	Time history for the measured slope of the front, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.9. Vertical lines indicate points where the cluster begins the process of reversing direction. . . . .	46
4.12	Controller state as the robots travel the path in Fig. 4.9. . . . .	47
4.13	The centroids of each subcluster and the center of the aggregate cluster as it travels along a front, where the run from Fig. 4.9 is in blue. . . . .	47
4.14	Experimental data for six robots navigating along a front with a sinusoidal path and a ridge feature, superimposed upon a rendering of the grayscale scalar field. . . . .	48
4.15	Time history for the measured slope of the feature, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.14. Vertical lines indicate points where the cluster begins the process of reversing direction. . . . .	49
4.16	Experimental data for five separate runs, where the centroids of each subcluster and of the aggregate cluster are plotted for each, superimposed upon a rendering of the grayscale scalar field. . . . .	49
4.17	Experimental data for the robots as they navigate along a front with changing width, superimposed upon a rendering of the scalar field. . . . .	50

4.18	Time history for the measured slope of the feature, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.17. . . . .	51
4.19	Experimental data for five separate runs on the front with changing width, where the centroids of each subcluster and of the aggregate cluster are plotted for each, superimposed upon a rendering of the grayscale field. . .	52
5.1	A four robot formation used for low exposure waypoint navigation, contour mapping, and seeking multiple extrema. It is an equilateral triangle with a fourth robot at the center to detect curvature. . . . .	54
5.2	Overview of states and transitions for low exposure waypoint navigation.	55
5.3	A simplified example of a low exposure waypoint navigation scenario, where the arrows represent the direction of travel as the cluster follows the displayed contour. Each sub-figure displays a portion of the travel, where the new movement is highlighted in blue. The green diamond is the starting point, while the red one is the end point. The red dashed line is a boundary for the navigation area. . . . .	56
5.4	The contour plot for the interpolated Cs-137 in Chernobyl Exclusion Zone soil samples. The red rectangle represents the boundaries set for the simulations conducted in this chapter. . . . .	61
5.5	The cluster center positions corresponding to a single test of the low exposure waypoint navigation technique superimposed on the scalar field of interest. The red line indicates the right boundary of the operating area.	62
5.6	A plot of the time history of the navigator states for the run presented in Figure 5.5. . . . .	63
5.7	The time history for the scalar value sensed by robot 4 throughout a low exposure waypoint navigation test. . . . .	64
5.8	100 simulated runs displayed on the interpolated radiation data, all of which have the same destination. Green circles represent start points, red indicates the finish, red lines are boundaries. . . . .	65
6.1	An example contour mapping scenario, starting at the green diamond. . .	68
6.2	The state diagram for the contour mapping technique. It includes four states and five transition criteria. . . . .	69

6.3	The contour mapping navigation strategy operating on a simulated scalar field. (a) shows the cluster navigating a contour that does not encircle the peak, and in (b) the navigator recovers from this scenario and begins navigating another invalid contour. In (c) the navigator recovers again to find a viable route, and finally in (d) the cluster recovers once again and continues onward. . . . .	76
6.4	A single contour mapping test conducted on a scalar field interpolated from soil samples taken in the Chernobyl Exclusion Zone. . . . .	77
6.5	A subsection of Figure 6.4 in order to provide a more detailed view of the beginning of the motion. A small portion of each of the three individual contours followed can be seen in this view. . . . .	78
6.6	Time history of the controller state for the test displayed in Figure 6.4. . . . .	79
6.7	Time history of the average scalar value for the test displayed in Figure 6.4. . . . .	79

---

# List of Tables

3.1	Cluster parameter error for the straight trench test presented in Fig. 3.4.	20
3.2	The RMS trenchline tracking error for all five trials plotted in Fig. 3.8.	23
3.3	Cluster parameter error for the parabolic trench test presented in Fig. 3.9.	24
3.4	The RMS trenchline tracking error for all five trials plotted in Fig. 3.10.	25
3.5	Cluster parameter error for the wide ridge test presented in Fig. 3.11.	28
3.6	The RMS ridgeline tracking error for all five trials plotted in Fig. 3.12.	29
3.7	Cluster parameter error for the test presented in Fig. 3.13 with a cubic ridge terminating in a saddle point.	30
3.8	The RMS ridgeline tracking error for all five trials plotted in Fig. 3.14.	31
4.1	Slope errors for the subclusters for the straight front tests presented in Fig. 4.13.	47
4.2	Slope errors for the subclusters for the tests presented in Fig. 4.16.	50
4.3	Slope errors for the subclusters for the tests presented in Fig. 4.19.	51
6.1	Simulation End Conditions	81
1	The parameters for the scalar field equations for each experiment.	A1
2	Greyscale ranges for the scalar fields.	A2
3	Cluster parameter RMS errors for the front tests presented in Figs. 4.9, 4.14, and 4.17.	B1

---

# Nomenclature

## Frequently Used Notation

Robot $i$ position	$x_i, y_i$
Robot $i$ angular rotation	$\theta_i$
Robot space pose vector	$\vec{R}$
Robot $i$ scalar value	$z_i$
Cluster position	$x_c, y_c$
Cluster angular rotation	$\theta_c$
Cluster space pose vector	$\vec{C}$
Set of kinematic transforms	$KIN()$
Jacobian Transform	$J$
Rotation matrix from $b$ to $a$	$R_b^a$
Local gradient estimate	$\vec{g}$
Bearing of the local gradient	$\vec{b}_{grad}$

## List of Acronyms

RMS	Root Mean Square
AUV	Autonomous Underwater Vehicle
ASV	Autonomous Surface Vessel/Vehicle
ABE	Autonomous Benthic Explorer
RSL	Robotic Systems Laboratory
RGB	Red Green Blue
PID	Proportional Integral Derivative
GPS	Global Positioning System

---

# CHAPTER 1

## Introduction

In a traditional robotic navigation scenario, the destination and route are known at the outset, and control strategies are used to implement the desired motion. Commonly used techniques include waypoint navigation, path following, and trajectory control. Examples of these methods are depicted in Figure 1.1. Waypoint navigation can be seen as moving to a series of points; path following controls the vehicle's route to a specific path; and trajectory control is similar to path following with time constraints where the vehicle is controlled to reach each point on the path at a particular time. All of these methods require significant *a priori* knowledge and route planning. By contrast, adaptive navigation techniques leverage environmental information sensed by the vehicle to make course adjustments throughout the navigation process. There are many examples of adaptive navigation techniques including: obstacle avoidance [4], rerouting paths based on air traffic conditions [5], and seeking particular environmental phenomena [6].

A specific class of adaptive navigation is a set of techniques used to navigate scalar fields. A scalar field is a spatial region in which every point has an associated quantity. The scalar values can be artificially created or naturally occurring, and example values of interest are depth, radiation counts, gas concentration, temperature, or any scalar measurement. Intelligently navigating these scalar fields provides the ability to navigate directly to or along points of interest, as opposed to the time and resource intensive alternative of exhaustively mapping a region. The numerous potential applications of these techniques include seeking sources of gas leaks, establishing safe perimeters around radiation environments, following the extent of ocean thermal fronts, *etc.* This work introduces a novel state-based method for sequencing adaptive navigation primitives in order to enhance performance and to accomplish more complex, mission-oriented objectives. This introductory chapter summarizes the literature concerning adaptive navigation of scalar fields (Section 1.1), and presents the project statement outlining the objectives and contributions of this work (Section 1.2).



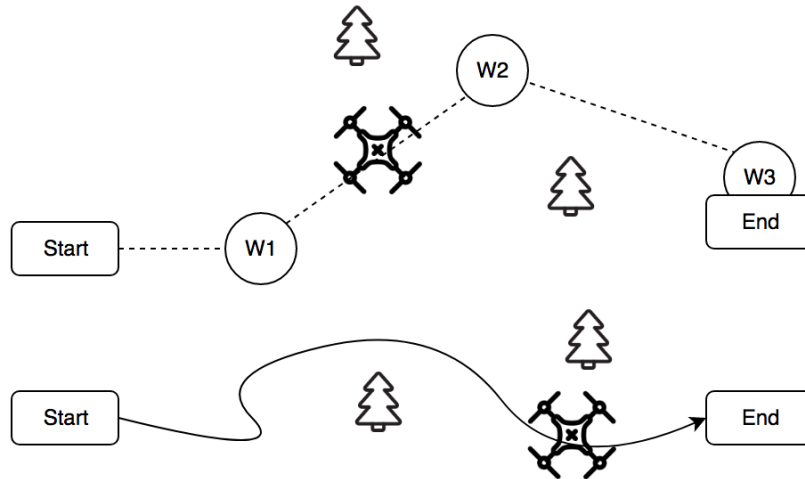


Fig. 1.1: Waypoint navigation (top) is conducted by moving the vehicle to a series of points, while path planning and trajectory control prescribe the specific, continuous route to be taken (bottom).

## 1.1 Adaptive Navigation Literature

Adaptive navigation of scalar fields has been the subject of a large amount of research, most of which has been focused on seeking extrema and following contours. [7] and [8] discuss a variety of general robotic environmental monitoring methods. An extremely thorough literature review has been presented in [1], which can be consulted for additional information about scalar field specific techniques. Adaptive navigation strategies have been implemented with a variety of methods, to include both single and multirobot approaches. Knowledge of the structure of the local scalar field is required to make navigation decisions, typically in the form of gradients or differentials. Several distributed measurements are needed to compute these quantities.

Single robot approaches require additional motion to acquire the needed spatial data. For example, [6] used an autonomous underwater vehicle (AUV) called the Autonomous Benthic Explorer (ABE), depicted in Figure 1.2, to follow bathymetric gradients by adding additional motion to the vehicle path inspired by *Escherichia coli* (*E. Coli.*) bacteria. Other approaches include [9], which used a bio-inspired technique based on the way a moth moves in and out of pheromone plumes, and a simulated method that used a sliding mode controller to move toward extreme points [10]. The same research group also demonstrated sliding mode methods for contour following in simulation and used a real robot to follow a simulated scalar field [11][12].



Fig. 1.2: The Autonomous Benthic Explorer [13]. Image reproduced with permission.

While there is some additional operational overhead involved with deploying multiple robots, many research groups have found that multirobot methods are a more effective way to perform scalar field navigation, as a formation of robots does not require spurious motion to determine the local trend of the scalar field. Instantaneous spatial characterization of the local field is both more efficient and improves the tracking of transient features [14]. The size and shape of the formation can also be tailored to suit the desired application. The majority of the literature focuses on extrema seeking, followed by contour following, and simulated technique verification is far more common than experimental methods; while field demonstrations are extremely rare.

The most common multirobot adaptive navigation application in the literature is source-seeking. [15] uses a multirobot approach with a spatial dither to follow the gradient toward an extrema. Other examples include a probabilistic technique robust to noise which was used to follow gradients in [16] and [17] which seeks sources using gradient and Hessian information. An example of a non-gradient based approach is [18], which used a swarm based approach with simulated attraction forces. Gradient climbing with experimental verification is presented in [19], which used several light sensing robots on a tabletop.



Fig. 1.3: Three automated kayakers traversing a scalar field at Stevens Creek Reservoir in Cupertino, CA [25]. Image reproduced with permission.

Contour focused examples include: [20], in which robots individually tracked a contour while sharing information, [21] which follows contours using a provably stable and converging technique, and a technique presented in [22] that uses a simulated two robot formation with *a priori* knowledge of a flow field to track the edge of a plume. Notable examples with experimental data include [23], which used a multirobot approach to follow a tape boundary, and [24] which used several robots to circumnavigate colored mats using simple controllers similar to tape following techniques.

The Robotic Systems Laboratory (RSL) at Santa Clara University has conducted significant work in the area of multirobot adaptive navigation, with a particular focus on experimental verification, and ultimately full field tests. The gradient based techniques for seeking extrema and following contours have been verified in the field, at Lake Tahoe and Stevens Creek Reservoir in California [25]. Figure 1.3 depicts the autonomous surface vessels used for these deployments as they navigate the bathymetry of the reservoir. The effort was extended to less commonly researched features, including ridges, trenches, and saddle points. These methods were demonstrated in simulation in [1].

Altogether, extrema, contours, ridges, trenches, and saddle points comprise a set of primitive features of a scalar field with varied applications in the field. Figure 1.4 displays these features as they are labeled on a simulated scalar field. As examples, locating maxima is useful for finding sources of dangerous leaks, minima can represent areas without service, contours can represent safe boundaries around hazardous regions, ridges can lead to the impact zone of plumes, trenches are low-exposure approaches,

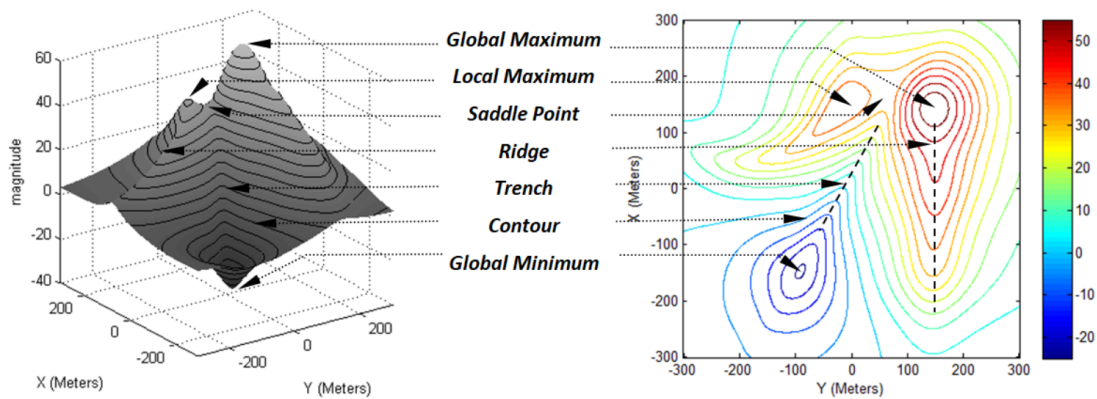


Fig. 1.4: The fundamental primitive features of a scalar field [1].

and saddle points can be low energy gateways between regions.

## 1.2 Project Statement

The objective of this research is to explore how a state-based architecture can improve the effectiveness of adaptive navigation strategies. This is the first time state-based behavior has been applied to the adaptive navigation primitives, and it is used to improve performance of adaptive navigation controllers and to explore three complex mission-level capabilities.

The first contribution is the design and implementation of the state machine architecture itself. Previous work has been focused on individual navigation primitives tailored for specific features, and using a state machine allows these to be sequenced in order to improve performance.

The ridge, trench, and saddle point navigator is the first technique to leverage this architecture. It provides the ability to initially seek the ridge/trench, and to recover in the case that the cluster is no longer spanning the feature. It was also the first full experimental verification of these adaptive navigation primitives.

The first mission-level navigator presented in this work is the first comprehensive method for navigating scalar fronts. It is verified with a full experimental implementation. Two other examples presented include a method for navigating to a waypoint without crossing above a preset scalar threshold and a method for mapping the contours around a local

extremum. A set of transition criteria and failure prevention methods are developed to support these efforts. The techniques are verified via large-batch simulations.

Additionally, the front navigator was the first adaptive navigation technique to adaptively size the cluster formation throughout the navigation process. This showcases the potential for the formation to adapt based upon the characteristics of the local scalar field. Finally, two cluster formations were designed to support the techniques that had experimental verification. Each cluster design required the development of a set of forward and inverse kinematics, and the associated inverse Jacobian transform.

### **1.3 Dissertation Organization**

Chapter 2 describes the architecture of the enhanced adaptive navigation control approach, and shows how the state machine is incorporated. Chapter 3 shows how state-based behavior is used to improve primitive level performance for the ridge/trench following and saddle point station keeping techniques and results of the first experimental implementation of these primitives. Chapters 4, 5, and 6 show how state machine based integration of adaptive navigation primitives can be used to implement more complex, application or mission-level tasks. The first of these, front following, is verified in simulation, and through simple experiments. Other mission-oriented tasks are verified with statistics provided by large-batch simulations.

---

# CHAPTER 2

## Systems Overview

### 2.1 Control Architecture

A layered control architecture, as seen in Figure 2.1, is used to implement the multirobot adaptive navigation techniques, providing convenient abstraction between the navigation commands and the lower level controllers. The robot layer is used to control the vehicles, and their actuators. It receives commands from the cluster control layer, which is responsible for formation keeping and cluster level movements. The cluster level commands are sent by the adaptive navigator, which uses scalar field readings to estimate local field characteristics and to compute appropriate commands given the selected adaptive navigation control primitive. Finally, as part of the research conducted by the author, a state machine layer selects the adaptive navigation primitive to be executed at any particular time using a sequencing policy developed to execute a specific task.

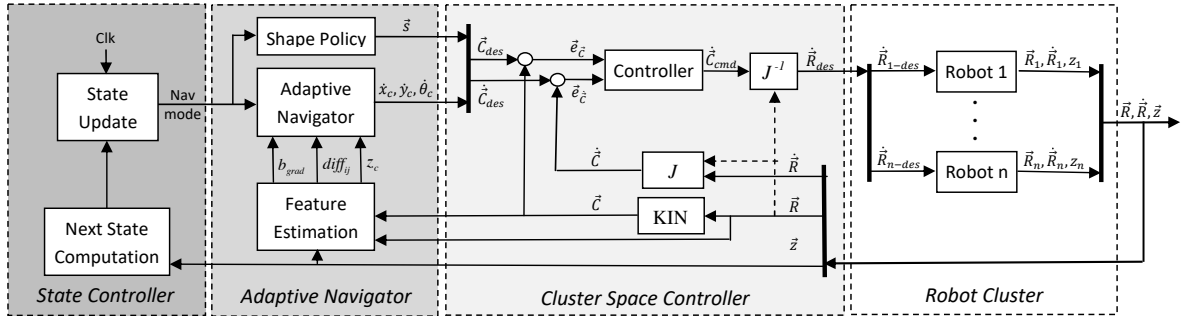


Fig. 2.1: The multirobot adaptive navigation control architecture with state-based behavior. From right to left, we have the robot level controller, the cluster controller, the adaptive navigator, and the state controller [2].

### 2.1.1 Robot Control

The robot control layer is responsible for executing the control of the individual vehicles in a multirobot formation. Depending on the needs of these vehicles, input control specifications can be provided either in the form of vehicle velocity commands (for resolved rate cluster spaces controllers [26]) or force/torque commands (for dynamic cluster space control [27]). Each vehicle implements these commands, executing vehicle control as required. This approach has been implemented for vehicles with varying degrees of freedom, and for robots that are both holonomic and non-holonomic. This variety includes vehicles such as rovers [28], flying drones [29], and autonomous surface vessels [29][30].

### 2.1.2 Cluster Control

The cluster control layer of the architecture is responsible for maintaining the robot formation and translating formation information between robot space and cluster space. This provides a convenient layer of abstraction while performing complex multirobot tasks, which allows the geometry of the formation to be controlled directly as opposed to commanding each individual vehicle. This allows for a high level of customization, as the geometry of the cluster is selected (and manipulated) based upon the requirements of a mission.

This is accomplished by treating the formation as a fully articulated virtual mechanism [31]. The pose information for each vehicle in the planar case,  $(x_i, y_i, \theta_i)$  for  $i = 1-n$ , where  $n$  is the number of vehicles, is represented by the vector  $\vec{R}$ . A more convenient set of variables,  $\vec{C}$ , is selected based upon the desired cluster geometry for the application or altered during the mission based on operational requirements. The cluster vector is computed from the robot space information using a set of kinematic transforms, as seen in Equation 2.1 assuming  $n$  robots with  $m$  degrees of freedom.  $r_{mn}$  corresponds to a robot space variable,  $g_{mn}$  is a kinematic equation for computing a cluster space variable, and  $KIN()$  is the full set of kinematic transformations. The same transforms can be inverted to accomplish the reverse. Mobile robots are frequently designed to accept velocity commands, so the cluster control technique makes use of a Jacobian transform to map the robot velocities to cluster space, as in Equation 2.2, and the reverse can be applied via the inverse Jacobian transform, as in Equation 2.3. The formation control loop itself is typically executed in cluster space, which provides more intuitive behavior.

$$\vec{C} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{mn} \end{pmatrix} = \text{KIN}(\vec{R}) = \begin{pmatrix} g_1(r_1, r_2, \dots, r_{mn}) \\ g_2(r_1, r_2, \dots, r_{mn}) \\ \vdots \\ g_{mn}(r_1, r_2, \dots, r_{mn}) \end{pmatrix} \quad (2.1)$$

$$\dot{\vec{C}} = J(\vec{R})\dot{\vec{R}} \quad (2.2)$$

$$\dot{\vec{R}} = J^{-1}(\vec{R})\dot{\vec{C}} \quad (2.3)$$

This technique has been used for a variety of applications and environments to include guarding an objective [32] and aligning a string of robots to relay communications [33]. It has also been demonstrated that the technique is capable of both holonomic and non-holonomic cluster movement, obstacle avoidance, and has been proved to be Lyapunov-stable [34]. The particular formations used for each technique presented in this work are detailed in the appropriate chapters.

### 2.1.3 Adaptive Navigator

The adaptive navigator is responsible for executing the currently selected adaptive navigation control primitive, which is typically accomplished using information about the local scalar field. These metrics are acquired by processing the scalar readings from each individual robot in the formation. The techniques discussed in this work typically use gradient based methods, scalar differentials, or some combination thereof. These values are used by the adaptive navigation primitive controllers, which ultimately generate the commanded cluster-level velocities. In the simplest case, the commanded formation is static, however the control architecture provides the navigator with the ability to re-shape the cluster throughout the mission as necessary. The cluster controller itself is responsible for maintaining this prescribed formation. The control equations themselves and other details for the primitive controllers for each application are described in the relevant chapters.



## 2.1.4 Finite State Machine

The state machine control layer, which is new to this work and developed by the author of this dissertation, is dedicated to selecting the relevant control primitive based upon mission parameters and sensed scalar information. Every control technique presented in this work leverages this layer of the control architecture for a variety of reasons. Depending on the application, it is used for tracking recovery/robustness, a patrolling behavior, splitting a mission into distinct tasks, and cycling through primitive navigators in order to achieve a more complex net effect. While most of the controllers within the individual states are already well defined, there is significant nuance in selecting and implementing the appropriate state transition criteria in a way that both meets the objectives of the given mission, and provides robustness to: failure, cyclic behaviors, false positives, *etc.*

A transition between controller states typically involves multiple criteria, and each individual criterion can be categorized by both its purpose and method. The purpose of a given criterion either directly flows down from the mission objective, in which case we call it a primary criterion, or it is in place to prevent a failure mode, in which case we call it a secondary criterion. The method of a criterion can be categorized based upon the information used to make the logical decision, which for our purposes can be spatial, temporal, or environmental. For example, a primary spatial criterion would be whether the cluster of robots reached a target destination, whereas a primary environmental criterion would be whether the estimated local gradient of the field was above a certain threshold, and furthermore a secondary temporal criterion could be whether enough time has passed since the last state transition.

As stated previously, the cluster control technique itself has been shown to be Lyapunov stable, so to confirm stability for a state based method as a whole, the stability of the state switching itself needs to be considered. There are several methods to ensure stability when switching between states. A simple solution is to wait for any transient behaviors to die out before switching states, as rapid state changes can cause instability [35]. The work presented in [35] further describes methods for conducting a full Lyapunov stability analysis on a system. This technique is used in [36] to evaluate the consequences of changing cluster states while allocating robots to different tasks; it was determined that if the commanded rate remained less than or equal to the original rate, the switches would be stable. This could be directly applied to the state-based controllers used for navigation in this work. A more formal set of constraints could be

imposed, however the techniques presented here largely use constant-magnitude velocity commands, which by definition meets the required criteria. Even if this were not the case instability would be unlikely as rapid state transitions are not common with this technique, as often times there are minimum time requirements for state transitions, and/or significant spatial movement required before the next transition occurs.

## 2.2 Simulation Architecture

Many of the navigation strategies presented in this work make heavy use of simulation results. In all cases, the simulation software was developed using a combination of MATLAB and Simulink. The structure of the Simulink diagrams is much like that of the control architecture that has already been presented. There is a primary MATLAB function that manages the adaptive navigation and state changing logic, and the Simulink portion manages the appropriate dynamics, tracks systems states, *etc.*, all of which is logged for later analysis.

While additional details of the simulations are described further in the relevant chapters, it is worth noting that the depth of the simulation is different depending on the application. For example, the simulation for verifying the front controller used in Chapter 4 has full vehicle dynamics, sensor noise, *etc.*, while the simulations used in Chapters 5 and 6 have simplified dynamics so that the performance of the state-based controllers can be verified via large batch simulations without taking a prohibitive amount of time and computational resources.

## 2.3 Experimental Testbed

The testbed used in the experimental portions of this work was previously designed specifically for testing adaptive navigation techniques, which can be difficult to test given that a suitable scalar phenomena must be located or created. The testbed consists of up to 12 small omnidrive vehicles (one of which is shown in Figure 2.2) with embedded processing and the capability to send and receive messages over a wireless network. The size of the workspace varies based upon the experiment being conducted, but within this work, it is on the order of several meters in both width and length. The robots themselves have a footprint of approximately 20 *cm* by 20 *cm*. Robot position tracking

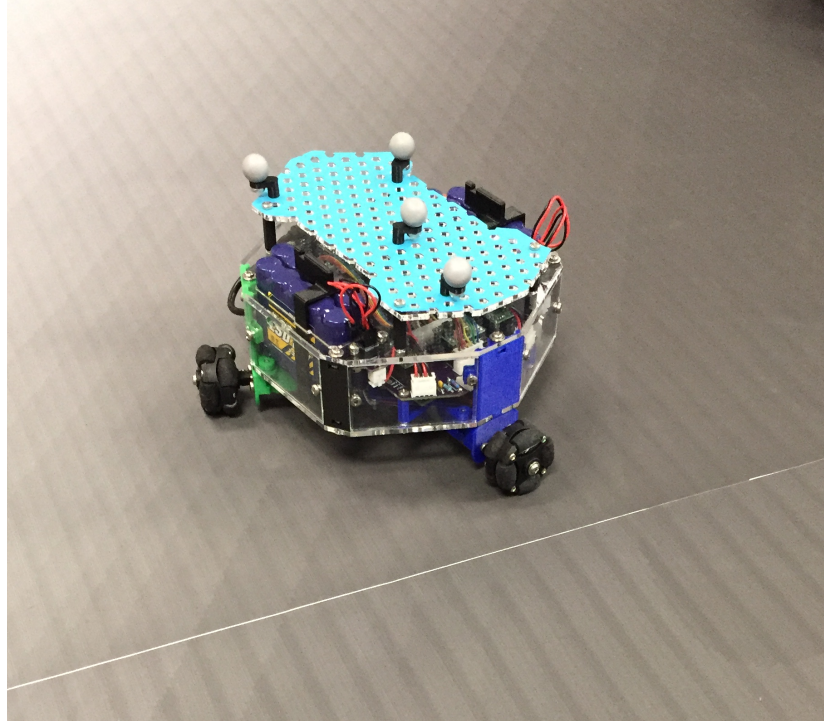


Fig. 2.2: A single omnidrive robot on a printed scalar field.

is performed with a network of 16 infrared cameras that track reflective markers on each robot; tracking accuracy is achieved with a standard deviation of less than  $7.5\text{ mm}$  across the workspace [2], making it significantly smaller than both the workspace and individual robot dimensions.

The key feature of the testbed is the ability to navigate custom scalar fields represented by printed grayscale sheets. The vehicles sense the reflectance of any given point on the sheet using RGB reflectance sensors mounted to the underside of the robots. These sensors have four channels: one with a red filter, one with a green filter, one with a blue filter, and finally a clear channel. For this work only the clear channel is used, meaning we are effectively measuring the reflectivity of the surface. The sensed values are sent to a base station running MATLAB/Simulink, which returns the appropriate robot velocities at a  $10\text{ Hz}$  servo rate based on the results of adaptive navigation and cluster control computations. The computation stack on the robots translates these velocities into wheel speed setpoints and implements them via PID speed control on each of the three motors.

Notable sources of error in the testing process include packet loss over the network, noise from the sensors, and the quality of the printouts. Despite these drawbacks, the testbed

has performed well, and has validated multiple novel adaptive navigation techniques. Full details about this system are provided in [37], to include characterization of the sensors, detailed design information, *etc.*

---

## CHAPTER 3

# Experimental Verification of Ridge, Trench, and Saddle Navigation

While the viability of ridge/trench following and saddle point station keeping was demonstrated in [1], this was done only in simulation, with simplified vehicle dynamics and without any real-world effects. The author of this dissertation played a significant role in the development of work contained in [1], and is the primary contributor for all the further content in this chapter. The experimental implementation and verification of the navigators is the next step for this work. This required the first full implementation of the 5-robot formation. Finally, this is the first adaptive navigator to incorporate state-based primitive switching. This is used to significantly increase the robustness and versatility of the controllers. Without the state-based behavior the ridge/trench/saddle navigator must start on the feature, and remain on it at all times. With the addition of the state machine, the cluster can start off the feature, and while it was a non-issue in the experimental results in this chapter, it also has the potential to recover should it fail to track the feature. In such situations, the state machine would select an alternate adaptive navigation primitive until the cluster could (re)acquire the feature of interest.

### 3.1 Review of Ridge/Trench/Saddle Controller

The majority of previous adaptive navigation research has been in the area of source seeking. [38] navigated up a plume to a source by changing which robot was the leader based upon which vehicles were in the plume. A technique that mimicked the behavior of a moth was used to dither in and out of a spatially planar cross-section of a plume until reaching a source [9]. Navigating down a ridge has the opposite objective, as it can be used to follow a feature outward from a source, and determine the areas impacted by it. This is something that has not been done previously aside from [1] where the algorithm was first presented.

The adaptive technique for ridge/trench/saddle navigation is different from our established extrema and contour following adaptive navigation techniques in that it does not compute gradient vectors. Instead it calculates the scalar differentials between robots in the cluster, and uses those to make control decisions. Accomplishing this requires certain features of the cluster formations, and a new set of control equations.

For this work a ridge is defined as a series of points that are a maximum in a given axis on a scalar surface with monotonic scalar values along the principle line of the feature. A trench is an inverse of a ridge, with a series of points located at directional minima, and monotonic scalar values. These criteria can be met in a number of circumstances, including some that may not be considered a ridge given the current mission objective. To accommodate this, the required pronouncement of feature can be added as a requirement to transitions within the state controller.

### 3.1.1 Cluster Definition

The five robot cluster is shown in Figure 3.1. As it consists of five spatially planar robots, the robot space pose vector  $\vec{R}$  consists of two translation and one rotational degree of freedom for each vehicle, for a total of 15 degrees of freedom. The kinematic transforms transform the robot space variables into the cluster space variables, which include the position and orientation of the cluster frame placed on robot 1 ( $x_c, y_c, \theta_c$ ), the distances between robots in a serial chain ( $d_2, d_3, d_4, d_5$ ), the angles between the serial links ( $\beta_3, \beta_4, \beta_5$ ), and the orientation of each robot relative to the cluster frame ( $\phi_i$ , for  $i \in 1, \dots, 5$ ). For the experiments presented in this work, all desired  $\beta_i$  and  $\phi_i$  values were set to zero, and the desired  $d_i$  values were held constant. The formation size was set manually, and varied from experiment to experiment to accommodate the varying spatial frequencies of each scalar field used in testing.

The kinematic equations used to transform the robot space variables to cluster space are given in Equations 3.1 through 3.15.  $x_i, y_i$ , and  $\theta_i$  are the robot space pose variables for the individual vehicles <sup>1</sup>

$$x_c = x_1 \tag{3.1}$$

---

<sup>1</sup>All inverse tangent computation in this work is ultimately implemented using MATLAB's atan2() function, which is a two input function that computes the result such that the angle is in the proper quadrant.

$$y_c = y_1 \quad (3.2)$$

$$\theta_c = \tan^{-1} \left( \frac{x_1 - x_2}{y_2 - y_1} \right) \quad (3.3)$$

$$\phi_1 = \theta_1 - \theta_c \quad (3.4)$$

$$d_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (3.5)$$

$$\phi_2 = \theta_2 - \theta_c \quad (3.6)$$

$$d_3 = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \quad (3.7)$$

$$\phi_3 = \theta_3 - \theta_c \quad (3.8)$$

$$\beta_3 = \tan^{-1} \left( \frac{x_3 - x_1}{y_1 - y_3} \right) - \theta_c \quad (3.9)$$

$$d_4 = \sqrt{(x_4 - x_2)^2 + (y_4 - y_2)^2} \quad (3.10)$$

$$\phi_4 = \theta_4 - \theta_c \quad (3.11)$$

$$\beta_4 = \tan^{-1} \left( \frac{y_4 - y_2}{x_4 - x_2} \right) - \theta_c \quad (3.12)$$

$$d_5 = \sqrt{(x_5 - x_3)^2 + (y_5 - y_3)^2} \quad (3.13)$$

$$\phi_5 = \theta_5 - \theta_c \quad (3.14)$$

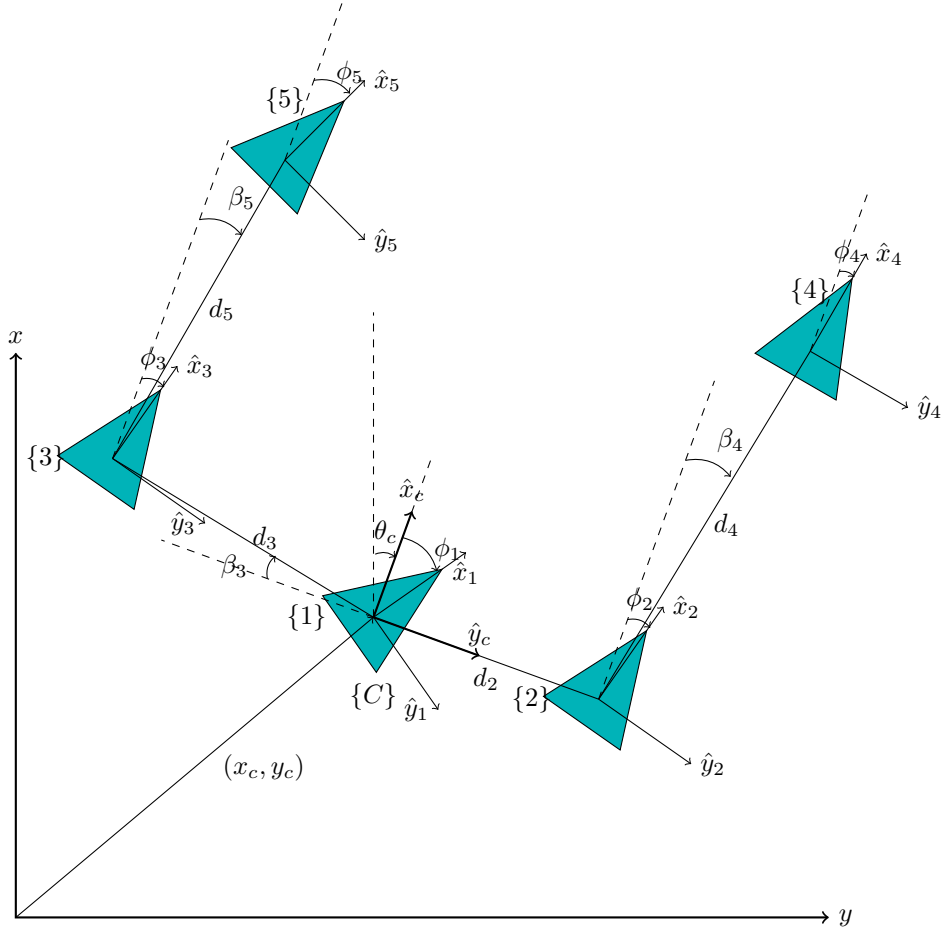


Fig. 3.1: Generic five robot cluster pose definition for the ridge/trench/saddle navigation technique. There are two serial-chain structures defined from robot one [2].

$$\beta_5 = \tan^{-1} \left( \frac{y_5 - y_3}{x_5 - x_3} \right) - \theta_c \quad (3.15)$$

### 3.1.2 Differential Control Strategy

Tracking of the ridge/trench is accomplished by sending cluster level translational and rotational commands based on longitudinal and lateral differentials. Robots 2 through 5 are used to compute these differentials, while robot 1 is used to confirm that the formation is still on the feature. The controller is in state 0 when it is unable to confirm whether it is on the feature, and in state 1 when it can.

The longitudinal differentials are  $z_2 - z_4$  and  $z_3 - z_5$ , which are used for determining the



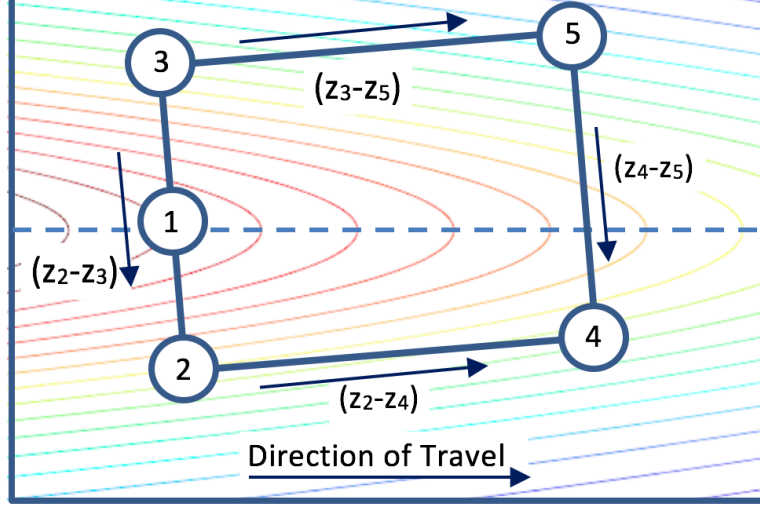


Fig. 3.2: The cluster of robots is displayed on a ridge, where it is laterally and rotationally offset from the feature. The lateral differentials center the formation on the ridge and provide rotational commands, while the longitudinal ones set the travel direction [2].

direction of travel along the feature. The lateral differentials are  $z_2 - z_3$  and  $z_4 - z_5$ , and are used for both centering on the ridge/trench and aligning with it rotationally. The relationship between these differentials and the local field is depicted in Figure 3.2. The differences between these differential pairs are used to make control decisions.

Equations 3.16 through 3.18 translate these scalar differentials into the cluster level velocity commands;  $\dot{x}_c$ ,  $\dot{y}_c$ , and  $\dot{\theta}_c$ . Computed velocity setpoints are discrete, with the cluster differentials used to determine the sign of the values, which for these experiments were  $v_x = 0.1 \text{ m/s}$ ,  $v_y = 0.1 \text{ m/s}$ , and  $\omega_z = 0.3 \text{ rad/s}$ . These values were set based upon the capabilities of the testbed, as there were restrictions on space, update rate, *etc.* When navigating down a ridge  $d = 1$ , and  $d = -1$  when following a trench upward.  $s = 1$  when the cluster is properly situated on the feature (state 1).

$$(\dot{x}_c) = s \times d \times v_x \{ \text{sgn} [(z_2 - z_4) + (z_3 - z_5)] \} \quad (3.16)$$

$$(\dot{y}_c) = d \times v_y \{ \text{sgn} [(z_2 - z_3) + (z_4 - z_5)] \} \quad (3.17)$$

$$(\dot{\theta}_c) = d \times \omega_z \{ \text{sgn} [(z_4 - z_5) - (z_2 - z_3)] \} \quad (3.18)$$

The adaptive navigator determines whether the cluster is aligned with the feature by checking if robot 1 is at a higher scalar level than robots 2 and 3 in the ridge case

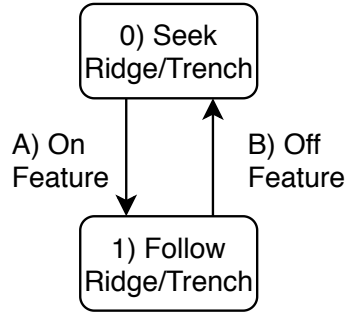


Fig. 3.3: State Diagram for Ridge/Trench Following [2]. State zero is used when off the feature, state 1 is used when on the feature.

(Equation 3.19), and lower in the trench case (Equation 3.20). This relationship is what gives the controller state-based behavior.  $z_{mar}$  is a value that can be adjusted to change the requirements for how pronounced the feature must be. For the experiments presented in this chapter, this value is set to zero. When on the feature and  $s = 1$  the cluster follows the feature as usual, and when off the feature (state 0,  $s = -1$ ) the cluster instead seeks higher scalar values in both  $\hat{x}_c$  and  $\hat{y}_c$  directions while maintaining the same alignment control, until it is spanning the crest of the ridge/trench. This state-based controller is summarized in Figure 3.3.

$$\text{Ridge Criteria: } (z_1 > z_2 + z_{mar}) \text{ AND } (z_1 > z_3 + z_{mar}) \quad (3.19)$$

$$\text{Trench Criteria: } (z_1 < z_2 - z_{mar}) \text{ AND } (z_1 < z_3 - z_{mar}) \quad (3.20)$$

The state-based behavior aids in the tracking of the ridge/trench as the robots can recover if they lose the feature, and also aids in the initial seeking of the feature. For the experiments presented here,  $s$  is only equal to  $-1$  in state 0 before the cluster locates the feature initially, indicating that the formation never lost the ridge/trench after locking onto it initially.

## 3.2 Experimental Results

The experiments to validate these control laws are performed on the testbed summarized in Section 2.3. Tests are performed on several printed scalar features in order to see

Table 3.1: Cluster parameter error for the straight trench test presented in Fig. 3.4 [2].

	RMS Error
$\beta_3$ (rad)	0.0715
$\beta_4$ (rad)	0.0691
$\beta_5$ (rad)	0.0446
$d_2$ (m)	0.0327
$d_3$ (m)	0.0199
$d_4$ (m)	0.0299
$d_5$ (m)	0.0339

how the navigator performed under a variety of conditions. These include a trench with a linear spatial path, a trench with a parabolic path, and two ridges with cubic paths, one that has a wider cross section and one with a narrow cross section terminating in a saddle point. For each set of tests we examine the performance of the formation, the tracking error relative to the center line of the ridge trench, and compare them to the ability of the cluster to continue spanning the center line as it travels.

### 3.2.1 Linear Trench

The linear trench is presented first, as it is the most straightforward scenario. As the scalar measurements are based upon reflectance, the lightest regions of the printout represent the highest values, while the dark areas are associated with low values. The path of each robot for a single run is plotted in Figure 3.4. Figure 3.5 depicts the magnitude of the tracking error over time, which had an RMS error of 82.14 *mm* if we exclude the initial transient behaviors associated with locating the trench while  $s = -1$ . Figure 3.6 provides of the value of  $s$  over time. The tracking error was computed by measuring the perpendicular distance between robot 1 and the center of the trench.

Previous work ignored all cluster and robot dynamics [1], therefore this is the first time this particular cluster formation has been implemented, so special attention was given to the quality of the formation-keeping. Figure 3.7 displays the time history for the  $d_i$  values (set to 0.7 *m*), and Table 3.1 includes performance information for both these and the  $\beta_i$  pose variables. The RMS error values were significantly smaller than the cluster sizing, indicating good performance.

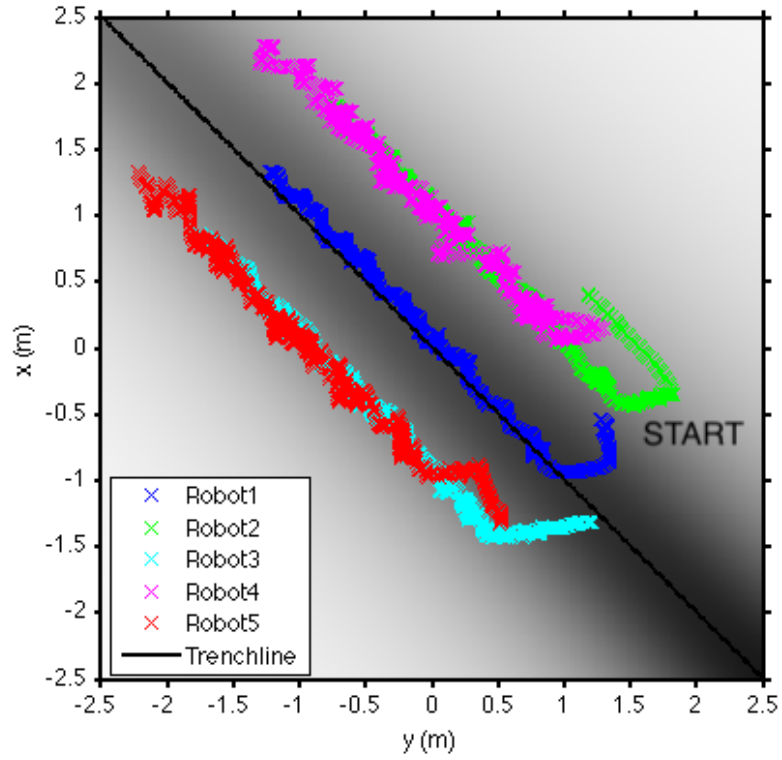


Fig. 3.4: The position of all five robots as they follow a linear trench, superimposed upon a rendering of the grayscale scalar field. [2]

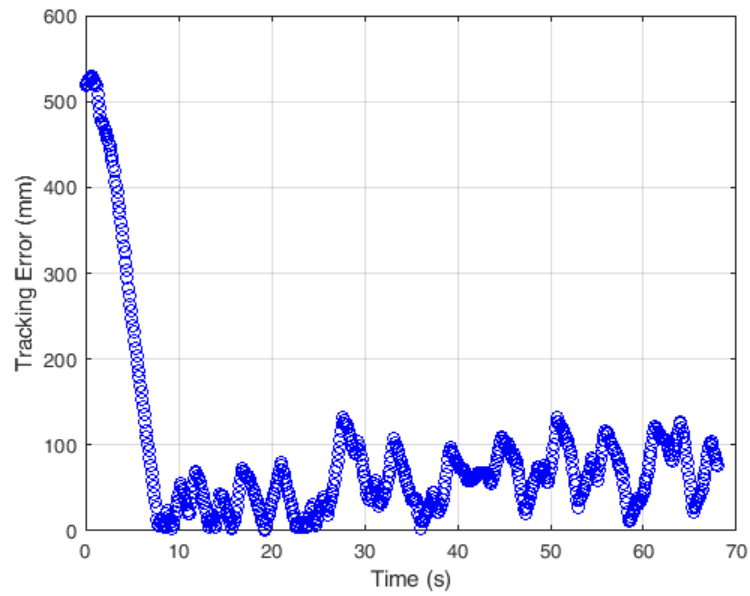


Fig. 3.5: Time history for the absolute value of the tracking error for the trial depicted in Fig. 3.4 [2].

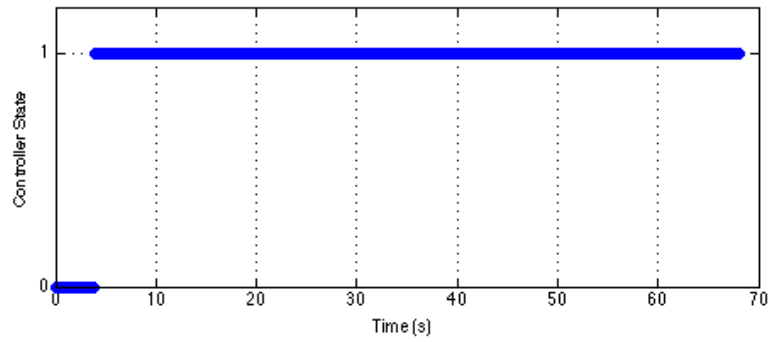


Fig. 3.6: Controller state ( $s$ ) as the robots travel the path in Figure 3.4 [2]. There is only a single transition from state 0 to state 1.

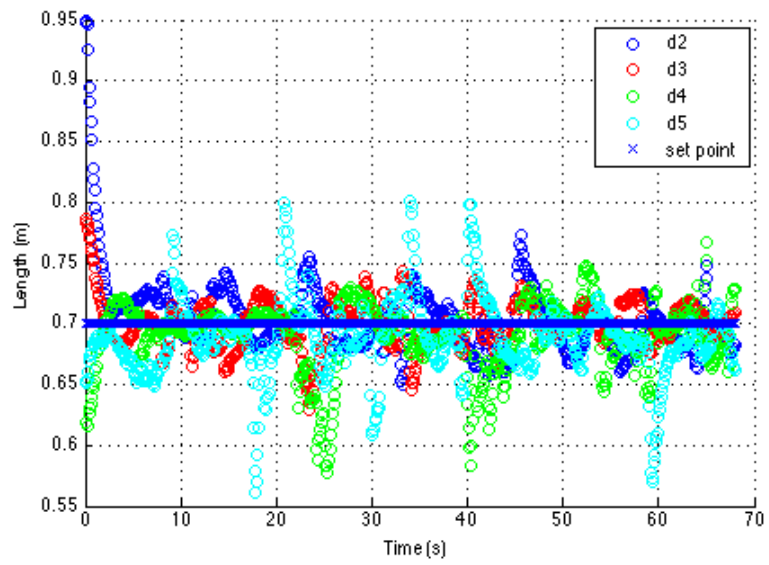


Fig. 3.7: Time history of the  $d_i$  values for the test presented in Fig. 3.4 [2].

Table 3.2: The RMS trenchline tracking error for all five trials plotted in Fig. 3.8.

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

In order to demonstrate the repeatability of this technique on this particular feature, Figure 3.8 superimposes the results of five different runs on top of the field. The robots are consistently successful in navigating the trench, and the RMS tracking error for each run is tabulated in Table 3.2. The variation between separate runs can be due to a number of different factors: the ambient lighting in the room affects both the sensor readings and the accuracy of the vision system calibration, the starting position of the robots changes which portion of the trench is navigated, packet loss reduces the formation performance, *etc.* Despite this, all of the RMS error values are more than an order of magnitude smaller than the total cluster width ( $d_1 + d_2 = 1.4\text{ m}$ ). This is a key comparison because the goal while navigating the feature is to keep the principle line of the ridge/trench within the baseline of the cluster. If the tracking error were to approach the cluster width, it would frequently not be spanning the highest point of the feature, indicating a failure to track it.

### 3.2.2 Parabolic Trench

The next scalar field was selected in order to see how the adaptive navigator handles an input feature with a principle line that consistently changes over time. A trench with a parabolic principal line was used to accomplish this. The paths for all five robots as they follow the feature are plotted in Figure 3.9. The RMS errors for cluster parameters throughout this test are collected in Table 3.3.

Repeated tests with different initial conditions are plotted in Figure 3.10. The RMS tracking errors for these runs are collected in Table 3.4. The errors are larger compared to the width ( $2d_i = 1.4\text{ m}$ ) than they were for the straight trench, indicating that the changing spatial input does in fact impact tracking performance. The tracking is better on the straighter sections of the parabola, further reinforcing this conclusion.

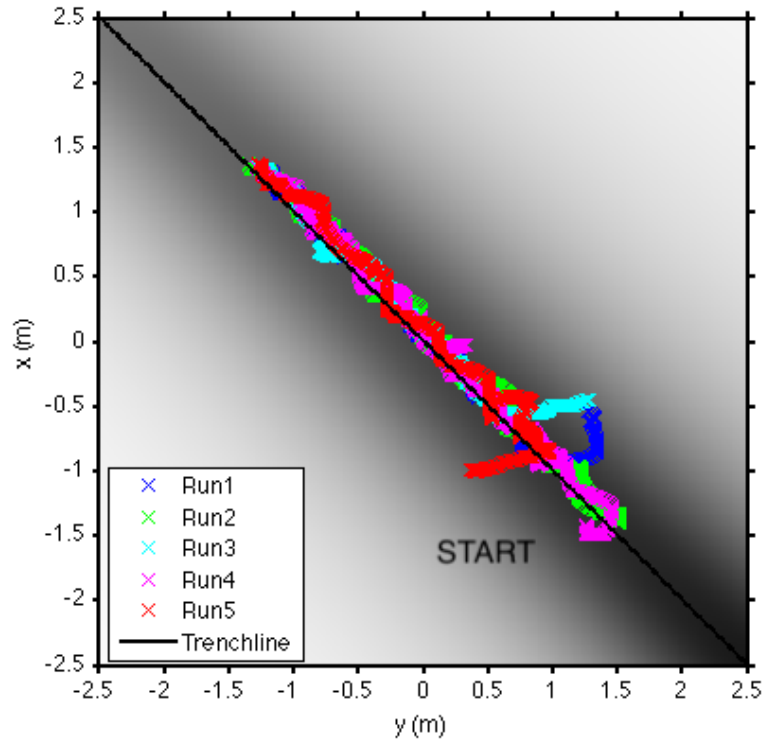


Fig. 3.8: Experimental data for the cluster center from multiple runs as robots navigate up a trench with a linear path, superimposed upon a rendering of the grayscale scalar field. The blue path is the same run depicted in Fig. 3.4.

Table 3.3: Cluster parameter error for the parabolic trench test presented in Fig. 3.9 [2].

	RMS Error
$\beta_3$ (rad)	0.1803
$\beta_4$ (rad)	0.0864
$\beta_5$ (rad)	0.1385
$d_2$ (m)	0.0304
$d_3$ (m)	0.0617
$d_4$ (m)	0.0357
$d_5$ (m)	0.0585

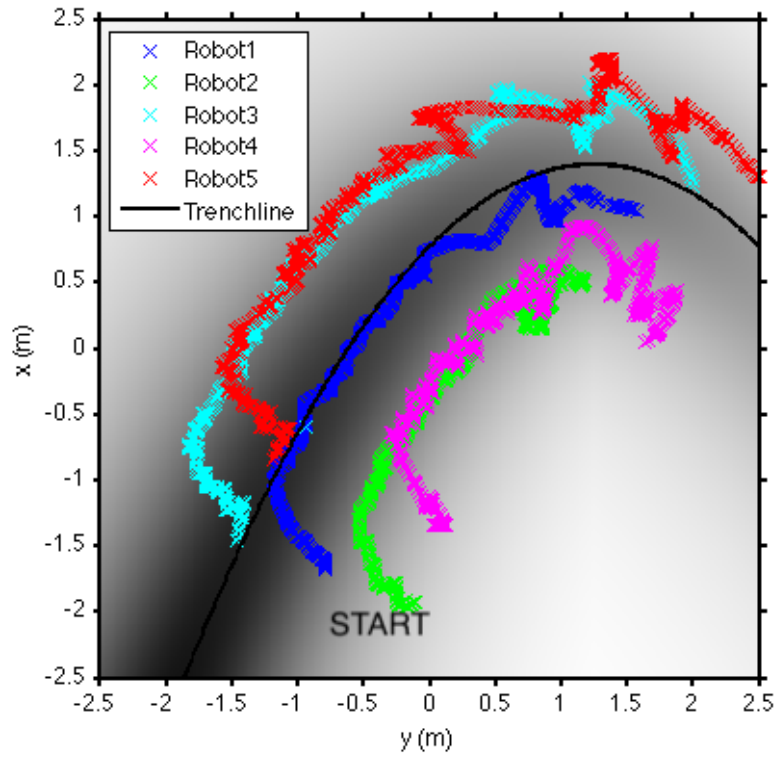


Fig. 3.9: Experimental data for five robots navigating up a trench with a parabolic path, superimposed upon a rendering of the grayscale scalar field [2].

Table 3.4: The RMS trenchline tracking error for all five trials plotted in Fig. 3.10 [2].

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



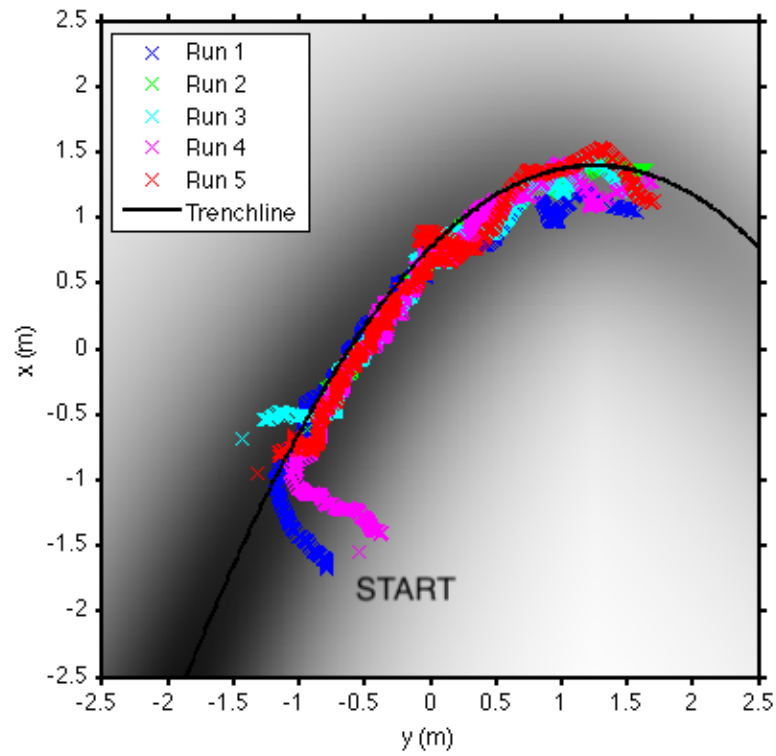


Fig. 3.10: Experimental data for the cluster center from multiple runs as robots navigate up a trench with a parabolic path, superimposed upon a rendering of the grayscale scalar field [2].

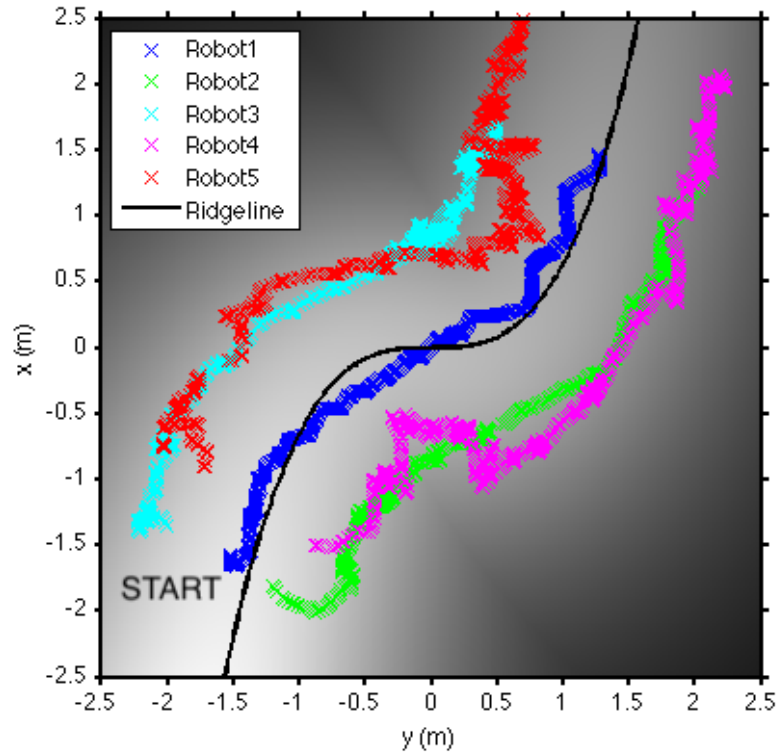


Fig. 3.11: Experimental data for five robots navigating down a ridge with a cubic path, superimposed upon a rendering of the grayscale scalar field [2].

### 3.2.3 Wide Cubic Ridge

The third feature is a ridge with a cubic principal line, and a larger lateral width than the other features tested (the scalar value falls off more slowly as distance from the center line is increased), which required a larger cluster sizing of  $d_i = 0.8 \text{ m}$  to reduce the sensitivity to sensor and field noise. The positions of the robots are displayed by Figure 3.11, with corresponding formation-keeping errors in Table 3.5.

Repeated trials are given in 3.12, with performance data in Table 3.6. Unsurprisingly, the tracking performance is worst near the inflection points, due both to the shape of the feature and the size of the cluster. That said, the cluster width also allows for more error during tracking, which was still far below the half-width of the cluster.

Table 3.5: Cluster parameter error for the wide ridge test presented in Fig. 3.11 [2].

	RMS Error
$\beta_3$ (rad)	0.0572
$\beta_4$ (rad)	0.0515
$\beta_5$ (rad)	0.3675
$d_2$ (m)	0.0478
$d_3$ (m)	0.0319
$d_4$ (m)	0.0517
$d_5$ (m)	0.0596

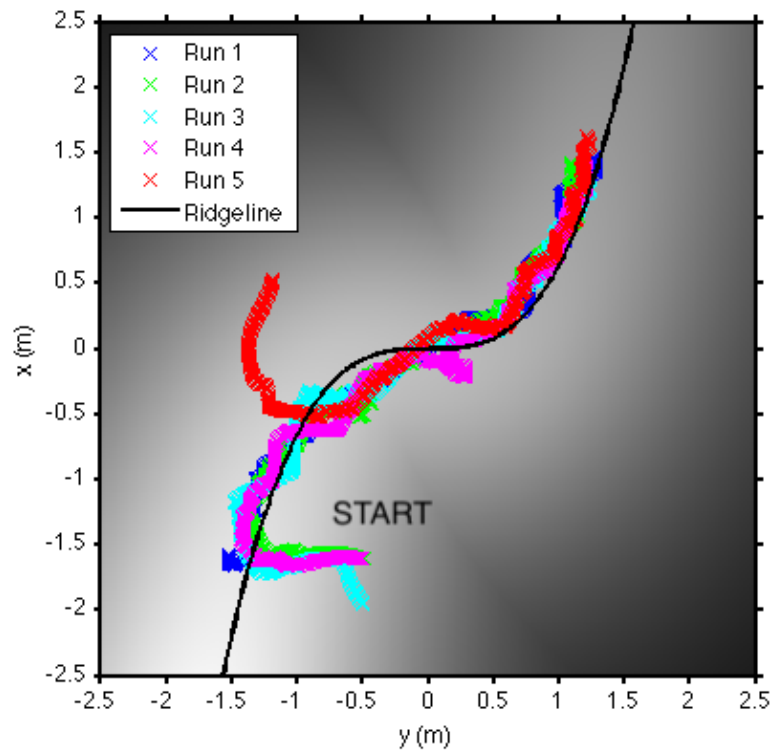


Fig. 3.12: Experimental data for the cluster center from multiple runs as the robots navigate down a ridge with a cubic path, superimposed upon a rendering of the grayscale scalar field [2].

Table 3.6: The RMS ridgeline tracking error for all five trials plotted in Fig. 3.12 [2].

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

### 3.2.4 Narrow Cubic Ridge with Saddle Point

The final scalar field was selected for two reasons; it provides a contrasting feature width to compare to the previous field, and it terminates in a saddle point. Unlike in the previous cases, the cluster is expected to settle at the saddle point at the end of the run. The behavior is demonstrated in Figures 3.13, with formation performance in Table 3.7. The formation size was smaller ( $d_i = 0.5 m$ ) for this run, as the feature was far narrower.

Repeated tests (Figure 3.14) demonstrated the reliability of the technique with low tracking error, as seen in Table 3.8. The tracking error was improved over the wide ridge scenario due to the reduced width of both the feature and the cluster, while still remaining smaller than the now reduced cluster width. The narrower cluster provides finer sensing resolution, resulting in less attenuation of higher spatial frequencies, resulting in better tracking of the turns. Additionally, the cluster settles on the saddle point in all cases. This result is reinforced by examining Figure 3.15 which provides filtered information about the longitudinal differentials over time. These values ultimately reach a steady state, indicating that the cluster settled on the feature.

### 3.2.5 Summary

These experiments verify the previously published adaptive navigation technique for following ridges/trenches and station keeping on saddle points. This was achieved through multiple demonstrations of feature following for a variety of configurations. In all cases the cluster was able to track the feature of interest with a low amount of error compared to the width of the cluster, which indicates that the robots were effectively straddling the feature the entire time. This set of experiments was the first time these navigation

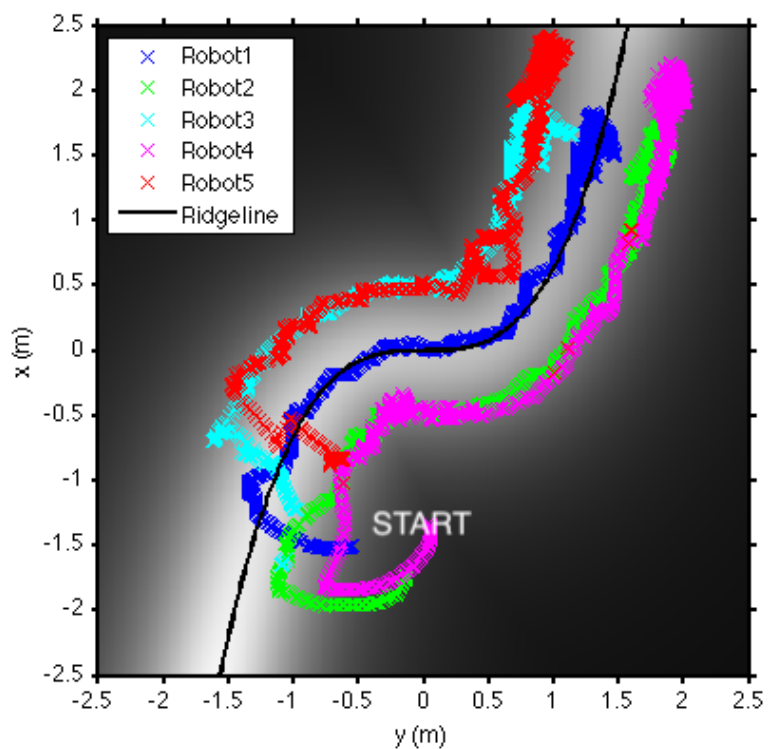


Fig. 3.13: Experimental data for five robots navigating down a ridge with a cubic path until they come to rest at a saddle point, superimposed upon a rendering of the grayscale scalar field.

Table 3.7: Cluster parameter error for the test presented in Fig. 3.13 with a cubic ridge terminating in a saddle point [2].

	RMS Error
$\beta_3$ (rad)	0.2298
$\beta_4$ (rad)	0.0981
$\beta_5$ (rad)	0.1689
$d_2$ (m)	0.0217
$d_3$ (m)	0.0394
$d_4$ (m)	0.0311
$d_5$ (m)	0.0696

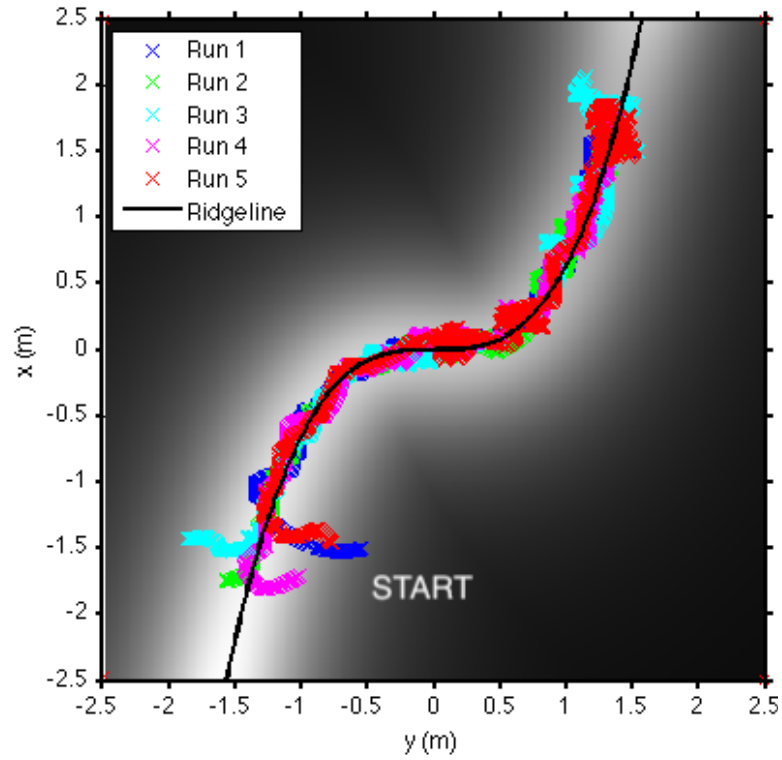


Fig. 3.14: Experimental data for the cluster center from multiple runs as the robots navigate down a ridge with a cubic path until they come to rest at a saddle point, superimposed upon a rendering of the grayscale scalar field. In each case, the robot cluster oscillates around the location of the saddle point [2].

Table 3.8: The RMS ridgeline tracking error for all five trials plotted in Fig. 3.14 [2].

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

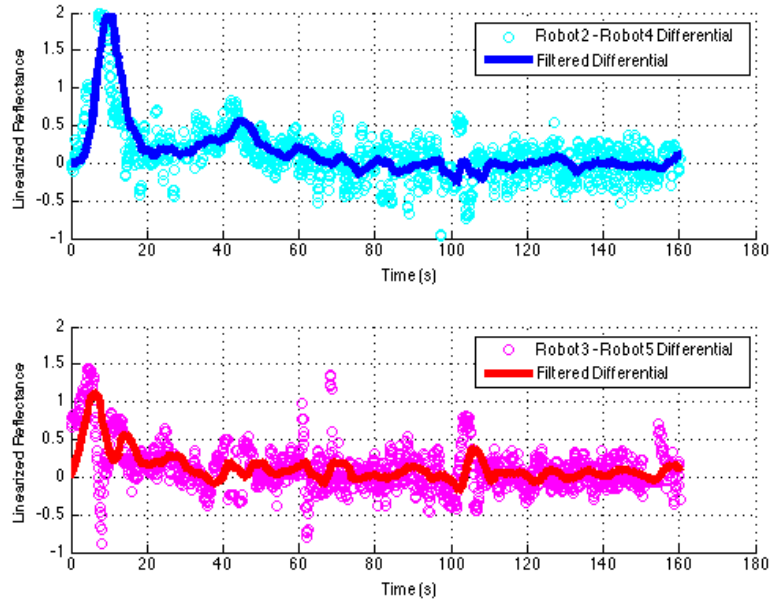


Fig. 3.15: Time histories of the scalar differentials between robots 2 and 4 (top), and robots 3 and 5 (bottom), for the run presented in Fig. 3.13. Includes both the unfiltered values used in the control computation, and the result after filtering with the response with a 50 sample moving average [2].

methods were implemented outside of simulation, with full dynamics, vehicles, and real-world effects like sensor noise, field noise, tracking error, communication latency, *etc.* This means the full set of adaptive navigation primitives described in [1] has now been demonstrated experimentally.

---

## CHAPTER 4

# Navigating Scalar Fronts

For the purposes of this work, we define fronts as a high gradient region at the boundary of two regimes with a comparatively slowly varying scalar value. Figure 4.1 depicts a generic front, with a width in the high gradient direction, and the length perpendicular to it. Part (a) of the figure demonstrates an idealized front, while part (b) is used to illustrate the different possible behaviors, like the wandering of the center line, changing width, changing scalar value, *etc.* Because of these variations, the key to tracking fronts is the local slope measurement, rather than by following a particular scalar value or the spatial center of the feature. The goal of this work is to develop a technique to navigate along the length direction of these fronts while spanning them in the width direction by using several slope measurements. The author of this dissertation is the primary contributor for this work.

Fronts are a common and important phenomena that occur in nature, and the scalar value of interest can be pressure, altitude, salinity, *etc.* Fronts are often scientifically significant in the marine environment, and have been the subject of a large amount of study. For example, [39] uses data sampled in 1999 from the turbidity front of the Río de la Plata Estuary in South America to analyze the structure of the front, as well as various velocity and temperature profiles. Field data from an estuarine salinity front in San Francisco is studied in [40] to determine how marine life interact with it, and in [41] the link between temperature fronts and animal life is studied via the Antarctic Circumpolar Current, the Subtropical Front, and the Subantartic Front. Yet another example from the field is [42], in which the energy dissipation at the 1 *km* wide temperature front in the Kuroshio Current is studied via a significant scientific expedition. In many of these works, a costly, time intensive, and manual field expedition was required, which indicates that automated methods would be valuable.

While the adaptive navigation techniques presented in Chapter 1 can effectively follow contours, which might apply for very well behaved fronts, a more robust approach is required to navigate a generalized front as defined above. There has been some limited



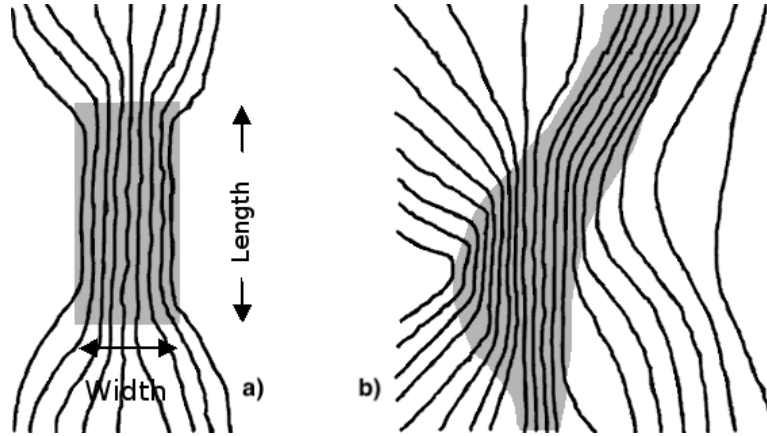


Fig. 4.1: (a), an idealized front, and (b), a generalized frontal region. The lines represent equally spaced contours of a scalar field, descending in value from left to right, and the highlighted region indicates where the front is located [3].

work in this area, such as [43], which conducted field deployments in the Monterey Bay using multiple AUVs to estimate and align with the local temperature gradient. [44] uses methods closer to the applications we are looking for in this work; a single ASV (Autonomous Surface Vessel) was used to determine the extent of the same marine temperature front by traveling along the length while zig-zagging across the width using a  $1\text{ km}$  spatial low pass filter while taking temperature measurements. The local gradient was estimated by taking the difference between sequential measurements. If this value remained above a threshold of  $0.15\text{ }^{\circ}\text{C}/\text{km}$  for  $1\text{ km}$ , it was considered to be on a front; after this condition was met, the vehicle was commanded to continue on for another 30 minutes before changing directions to continue the process. A similar process was used by the same group with the addition of vertical ‘yo-yo’ behavior to track three dimensional features [45].

The work presented here includes a more feature-complete planar multirobot technique, capable of adaptively sizing a formation to span a front while moving along its length, allowing it to find the full extent of the feature without extraneous movement.

## 4.1 Control Strategy

This technique uses a combination of differential and gradient based techniques with a size-adaptive cluster, and state-based behavior. The formation first seeks the front, then follows it, and then uses a state-based set of techniques to patrol back and forth

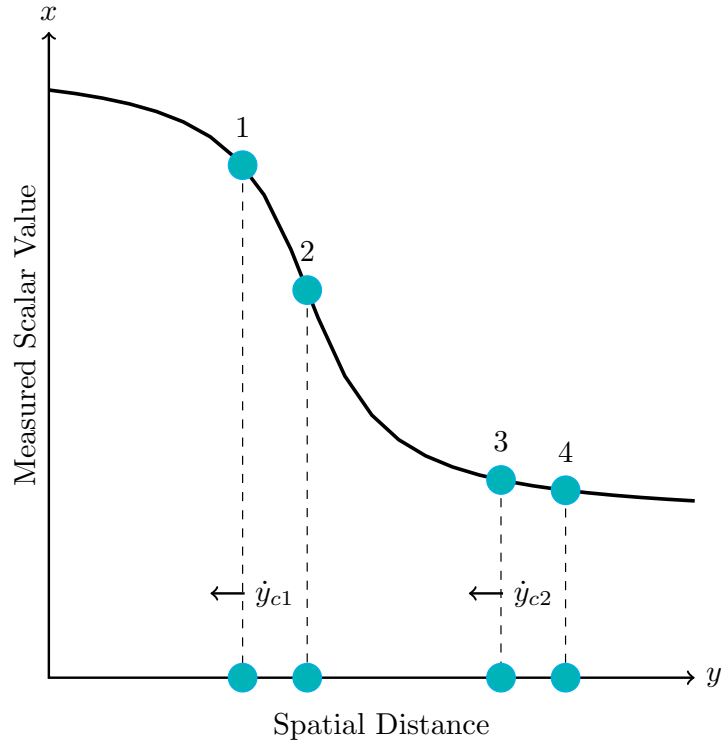


Fig. 4.2: Two pairs of robots follow the edges of a scalar front.

along the feature.

Front-seeking can be accomplished a number of ways, however the examples presented here use the contour following controller. The formation begins off the front, and is set to a desired contour value that is presumed to be present on the opposite side of the feature. Once on the front, the controller switches to following the front.

As previously mentioned, we define a front as a region of high slope, rather than associating it with a particular set of scalar values. The goal is also to span the feature, rather than to simply stay within it. This is accomplished with a six robot cluster. Two pairs of robots are used to track the edges of the front—one on the high side, and one on the low side. An example of this behavior is in Figure 4.2.

While the four co-linear robots can accomplish the spanning requirement, they are unable to ensure proper rotational alignment. This is maintained with the addition of a fifth vehicle, which allows for a more accurate gradient estimate, allowing the cluster to align with the contour direction. Finally, a sixth robot is added to provide symmetry in the gradient computation. Figure 4.3 provides an example of all the front following behaviors, with example velocity commands.

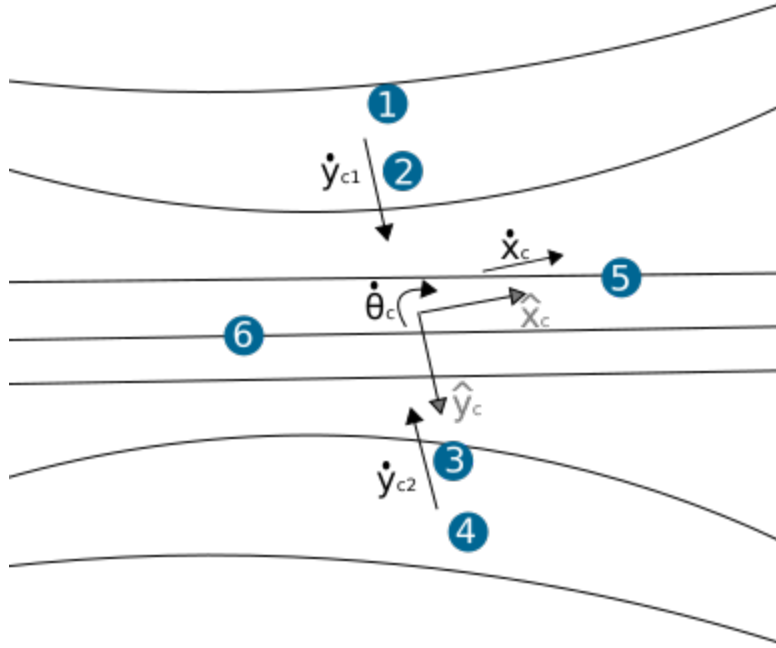


Fig. 4.3: An example of the robot formation on a scalar front, with example velocity commands. The contour lines are in equal scalar increments, descending from top to bottom [3].

#### 4.1.1 Cluster Formation

Figure 4.4 presents the cluster used for front following. It contains 6 planar robots, with a total of 18 degrees of freedom. The formation has two functions: to estimate the local gradient, and to use two subclusters to track the edges of the front. The frames for the subclusters  $\{c_1\}$  and  $\{c_2\}$  are located at the center of robot pairs 1-2 and 3-4, while  $d_1$  and  $d_2$  are the distances to the robots from the frame centers.  $d$  is the distance from the cluster frame  $C$ , located at the midpoint of the two subclusters. The angles of the subcluster frames relative to  $C$  are denoted by  $\beta_1$  and  $\beta_2$  and the positions of robots 5 and 6 relative to  $C$  are represented using the distances  $d_5$  and  $d_6$ , and angles  $\beta_5$  and  $\beta_6$ , respectively. Finally, the rotation angles of individual robots are represented by  $\phi_i$ , where for robots 1 through 4 they are based on  $c_1$  and  $c_2$ , and 5 and 6 are relative to  $\{C\}$ . For this application all  $\beta_i$  values are set to zero in order to align robots 1 through 4. The  $d_i$  values are set based upon the scalar field, based upon the spatial frequencies of interest.

The following kinematic equations (4.1 through 4.13) are used to transform the variables from robot space to cluster space. Recall that  $x_i$ ,  $y_i$ , and  $\theta_i$  are the robot space pose

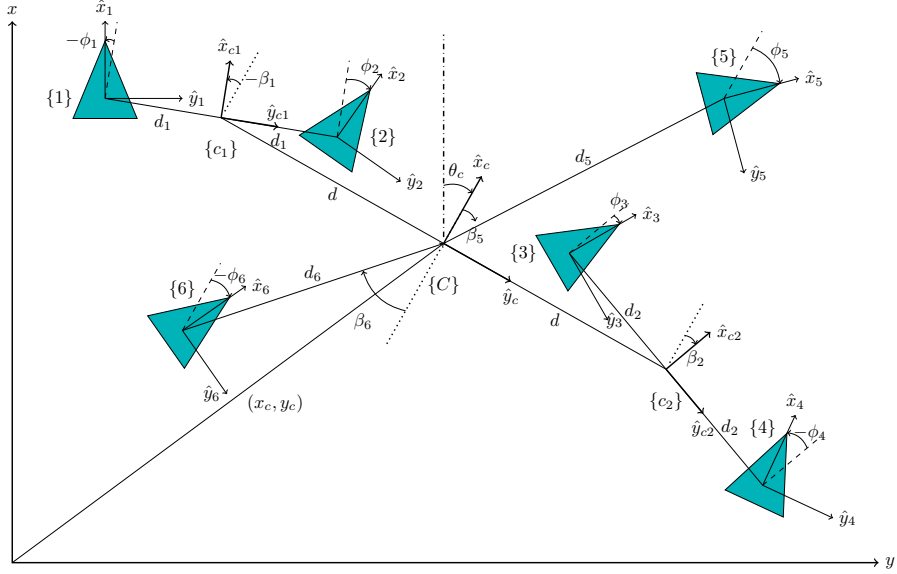


Fig. 4.4: Six Robot Cluster Pose Definition: Cluster space pose variables for a six robot cluster with intermediate frame centered between two pairs of robots, and the aggregate cluster frame centered between the two intermediate frames [3].

variables.

$$x_{ci} = \frac{1}{2}(x_{2(i-1)+1} + x_{2(i-1)+2}); \quad i = 1, 2 \quad (4.1)$$

$$y_{ci} = \frac{1}{2}(y_{2(i-1)+1} + y_{2(i-1)+2}); \quad i = 1, 2 \quad (4.2)$$

$$d_i = \frac{1}{2}((x_{2(i-1)+1} - x_{2(i-1)+2})^2 + (y_{2(i-1)+1} - y_{2(i-1)+2})^2)^{1/2}; \quad i = 1, 2 \quad (4.3)$$

$$x_c = \frac{1}{2}(x_{c1} + x_{c2}) \quad (4.4)$$

$$y_c = \frac{1}{2}(y_{c1} + y_{c2}) \quad (4.5)$$

$$d = \frac{1}{2}((x_{c1} - x_{c1})^2 + (y_{c1} - y_{c2})^2)^{1/2} \quad (4.6)$$

$$d_i = ((x_c - x_{i+2})^2 + (y_c - y_{i+2})^2)^{1/2}; \quad i = 3, 4 \quad (4.7)$$

$$\theta_c = \frac{\pi}{2} + \tan^{-1} \left( \frac{y_{c1} - y_{c2}}{x_{c1} - x_{c2}} \right) \quad (4.8)$$

$$\beta_i = \frac{\pi}{2} + \tan^{-1} \left( \frac{y_{2(i-1)+1} - y_{2(i-1)+2}}{x_{2(i-1)+1} - x_{2(i-1)+2}} \right) - \theta_c; \quad i = 1, 2 \quad (4.9)$$

$$\beta_i = \tan^{-1} \left( \frac{y_{i+2} - y_c}{x_{i+2} - x_c} \right) - \theta_c; \quad i = 3, 4 \quad (4.10)$$

$$\phi_i = \theta_i - \beta_1 - \theta_c; \quad i = 1, 2 \quad (4.11)$$

$$\phi_i = \theta_i - \beta_2 - \theta_c; \quad i = 3, 4 \quad (4.12)$$

$$\phi_i = \theta_i - \theta_c; \quad i = 5, 6 \quad (4.13)$$

### 4.1.2 Front Seeking

Any number of strategies can be used to locate a front, from exhaustive mapping, to random search, to gradient based techniques. For the purpose of this work, we use a gradient-based contour following approach to initially locate the front and align with it. It is assumed that the formation is on one side of the front, and that a desired scalar value is selected such that the robots would have to pass through the front in order to reach it. This does require some *a priori* knowledge of the field to do this without leveraging time history information, but far less than the specific front location.

The first requirement is to compute the local gradient, which for this application is done using information from all six robots to make a planar, least squares approximation of the local scalar field. The normal vector of this plane is projected onto the  $x$ - $y$  plane to

estimate the gradient. Equation 4.14 describes the relevant matrices, where  $A$  contains the robot positions, and  $B$  contains the scalar measurements for each robot. The normal vector of the plane is computed using Equation 4.15, and the gradient is calculated using Equation 4.16.

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \end{bmatrix}, B = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_i \end{bmatrix} \quad (4.14)$$

$$\vec{N} = (A^T A)^{-1} A^T B \quad (4.15)$$

$$b_{grad} = \tan^{-1} \left( \frac{N_y}{N_x} \right) + \pi \quad (4.16)$$

The computed gradient bearing is used to align the the cluster  $\hat{y}_c$  direction with the gradient via Equation 4.17. The scalar level of the cluster is controlled using Equations 4.18 and 4.19, where  $K_{ct}$  is a cross-track gain,  $z_{des}$  is the desired contour level,  $z_c$  is the average scalar value of the subclusters, and  $b_{cf}$  is the desired cluster bearing. By doing so, the cluster seeks a scalar level assumed to be on the far side of the front. Throughout contour navigation, the local slope  $m$  is computed by estimating a one dimensional gradient along the major axis of the robot cluster. This value is continuously checked in order to determine whether a front has been located ( $m > m_{min}$ ).

$$\dot{\theta}_c = \left( b_{grad} - \frac{\pi}{2} \right) - \theta_c \quad (4.17)$$

$$b_{cf} = b_{grad} + d \{ \text{sgn}(z_{des} - z_c) \} \\ \times \min [K_{ct} \times \|z_{des} - z_c\|, \pi/2] - (\pi/2) \quad (4.18)$$

$$\dot{x}_c = \cos(b_{cf}), \dot{y}_c = \sin(b_{cf}) \quad (4.19)$$

### 4.1.3 Front Following

The same gradient computation technique is used when navigating the front, and alignment is achieved with Equation 4.20. This ensures the cluster orientation is such that the two subclusters are aligned with the front in the width direction, while robots 5 and 6 are aligned with the length direction, as pictured in Figure 4.3. This keeps robots 1 through 4 in position to measure the local slope of the front while resizing in order to span it.

$$\dot{\theta}_c = K_\theta(b_{grad} - \frac{\pi}{2} - \theta_c) \quad (4.20)$$

Moving in the length direction of the front is accomplished with a simple velocity set-point (Equation 4.21), where  $s = \pm 1$  based on the controller state, and  $v_x$  is the constant velocity command value.

$$\dot{x}_c = sv_x \quad (4.21)$$

Each subcluster aims to track one edge of the front. The motion commands for each subcluster are in Equations 4.22 and 4.23. The  $z_i$  values are the measured scalar values from each vehicle,  $m_{min}$  is the minimum slope requirement for the front, and  $P$  is a value between 0 and 1.  $P$  is changed in order to set the subclusters to track a slope value appropriate for the relevant feature.

$$\dot{y}_{c1} = \text{sign} \left( Pm_{min} - \frac{|z_1 - z_2|}{2d_1} \right) \quad (4.22)$$

$$\dot{y}_{c2} = \text{sign} \left( Pm_{min} - \frac{|z_3 - z_4|}{2d_2} \right) \quad (4.23)$$

These values are then translated into cluster pose variables such that they can be executed within the control architecture. Equations 4.24 and 4.25 explicitly show how these behaviors are implemented by manipulating the  $y_c$  and  $d$  pose variables.

$$\dot{y}_c = \frac{1}{2}\dot{y}_{c1} + \frac{1}{2}\dot{y}_{c2} \quad (4.24)$$

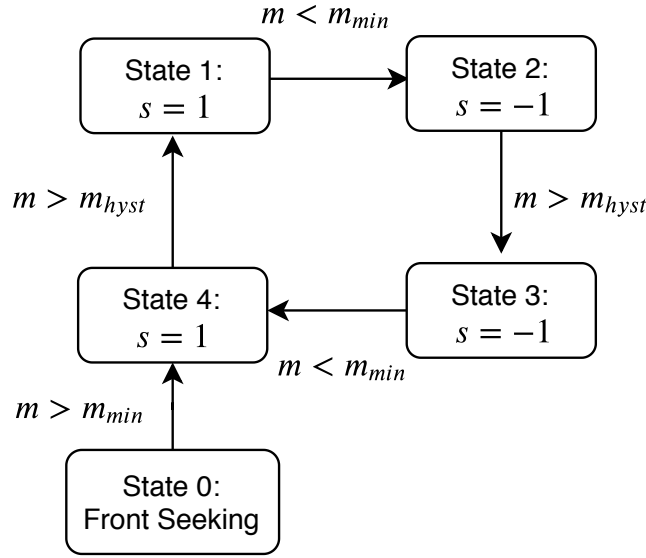


Fig. 4.5: Each controller state is listed with the appropriate value for  $s$ , and the slope criteria for each state transition.  $m$  is the measured slope of the feature,  $m_{min}$  is the minimum slope required for the feature to be considered a front,  $m_{hyst}$  is a higher slope requirement for exiting hysteresis states, and  $s$  is a variable used in the front following controller [3].

$$\dot{d} = \frac{1}{2}\dot{y}_{c2} - \frac{1}{2}\dot{y}_{c1} \quad (4.25)$$

#### 4.1.4 State Controller

The state machine portion of the controller allows for the sequencing of behaviors, and is summarized in Figure 4.5. It begins in state 0, which is the front-seeking behavior described in Section 4.1.2. Once magnitude of the local slope  $m$  exceeds the minimum slope  $m_{min}$ , the controller enters state 4.

The cluster travels in the positive  $\hat{x}_c$  direction in states 1 and 4, and the reverse in 2 and 3. Whenever the cluster reaches one of the ends of the front (detected by a drop in slope), it reverses direction and enters one of the hysteresis states, wherein the exit criteria is a higher slope to avoid oscillation at the edges ( $m_{hyst}$ ). The selection of  $m_{hyst}$  is just as important as selecting  $m_{min}$ , as it needs to be a value commonly found along the length of the front in order to reliably trigger the state transitions. It is assumed that there is enough prior knowledge of the front in order to make effective choices for



these values. In addition to the slope requirements for each state change, there is also a  $\pm\pi/18$  radian alignment requirement for each state transition, as the measured slopes are only accurate if the cluster is properly aligned with the gradient.

## 4.2 Simulation Results

A thorough simulation was developed in MATLAB/Simulink and tested to validate the front navigation technique in circumstances that emulate a probable field scenario. It aims to emulate similar circumstances to those found in [44], by using a temperature front 1600  $m$  long, 400  $m$  wide, and a temperature range of about 2.5  $^{\circ}C$ . In order to exercise the full features of the controllers, the test front, seen in Figure 4.6, incorporates a changing path, changing width, and changing scalar level along the length of the front by using a piecewise-defined function.

The simulation models six non-holonomic robots with first order dynamics, with time constants of 1  $sec$  for translational degrees of freedom, and 5  $sec$  for rotational degrees of freedom. There is added positioning error based upon the specifications of the Garmin GPS unit used in [25], with a standard deviation of 7.5  $m$  [46]. Temperature sensing errors are based upon a Vernier temperature probe [47], adding noise with a standard deviation of 0.5 degrees Celsius and a first order time constant of 4.35  $sec$  to the sensed temperature information.

Figure 4.7 shows the cluster beginning off the front, then moving up to the feature, and patrolling it. For this particular run the fixed parameters were,  $d_1 = d_2 = 12 m$ ,  $d_5 = d_6 = 48 m$ ,  $m_{min} = 0.014 ^{\circ}C/m$ ,  $m_{hyst} = 0.018 ^{\circ}C/m$ , and  $P = 0.5$ . While it can clearly be seen from the overhead view that the cluster successfully tracks the front, we can further examine the performance by looking at time history information. The measured slopes for both the full cluster and the subclusters are plotted in Figure 4.8. This clearly confirms that the cluster properly reverses direction when the  $m_{min}$  requirement is violated, and the subclusters track their slope target with low RMS errors of 0.0021 and 0.00019  $^{\circ}C/m$ .

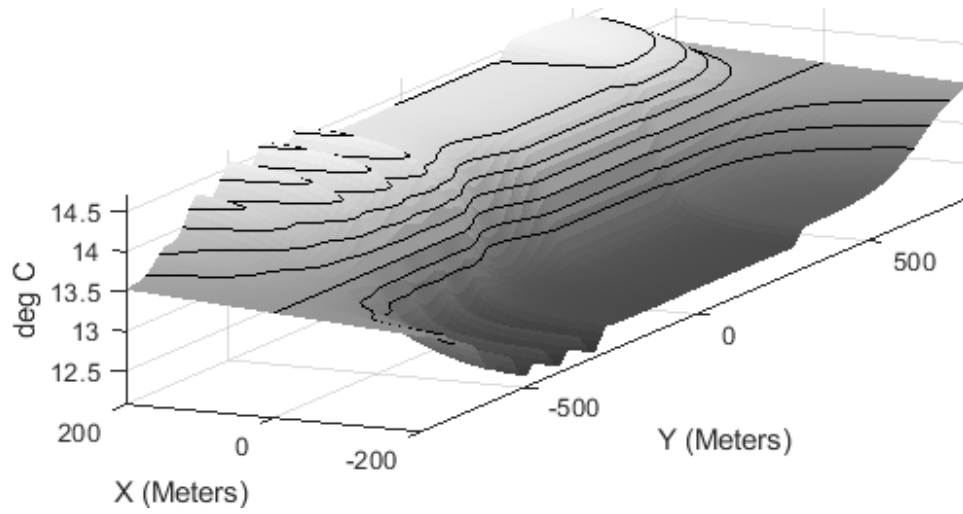


Fig. 4.6: A three dimensional rendering of the scalar front used for simulated front following [3].

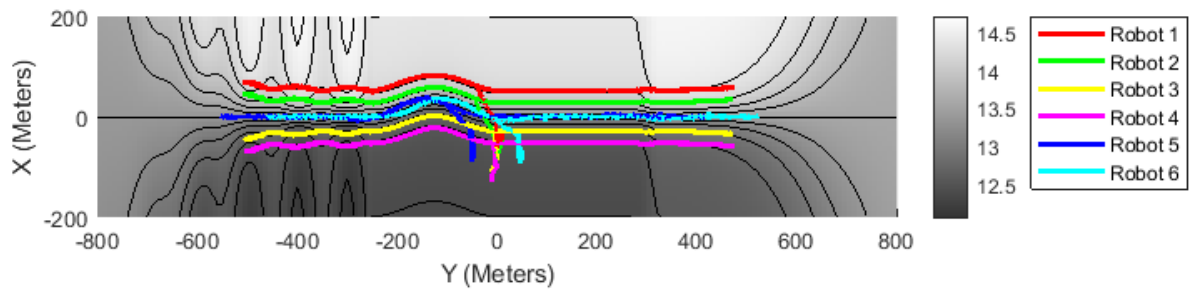


Fig. 4.7: An overhead view of six robots navigating the scalar front; they start off the feature, climb up to it, travel in the positive  $y$  until the front ends, then travel back in the  $-y$  direction [3].

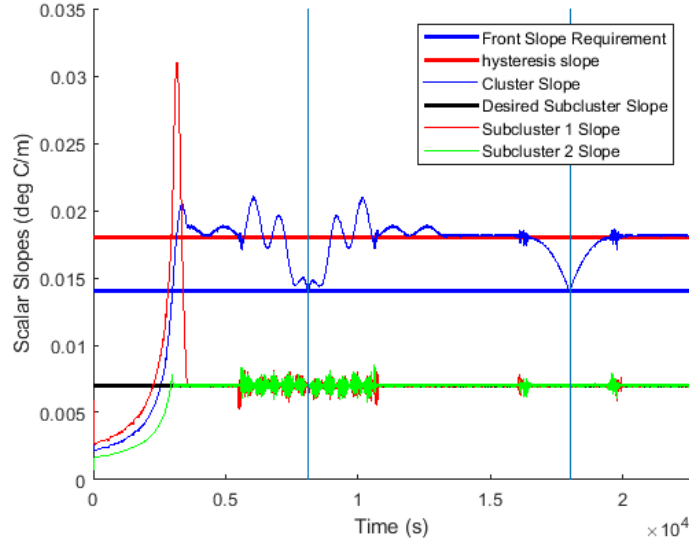


Fig. 4.8: The time history of the cluster and subcluster slopes, as compared to the desired values. The vertical lines indicate the points at which the robot cluster is commanded to reverse direction [3].

## 4.3 Experimental Results

Several sets of experiments were conducted on the adaptive navigation testbed (section 2.3) in order to validate the performance of the front following technique with real-world effects. As with the experimental verification of the ridge/trench/saddle navigation, the front following technique was tested on several different scalar field printouts in order to ensure it could successfully follow a variety of different features.

### 4.3.1 Straight Front

The first scalar field tested is the simplest option—a front with a linear path. An overhead view of all six robot paths for a single test is provided in Figure 4.9. As expected, they move to reach the feature, follow it, and reverse direction up on reaching the end.

The formation performance is particularly relevant in this case, as the cluster was being actively resized, and the position and orientation relative to the front could significantly alter the local slope measurements. The  $d_i$  values with constant set points are plotted in Figure 4.10. A table with the RMS error for the cluster parameters is in Appendix B.

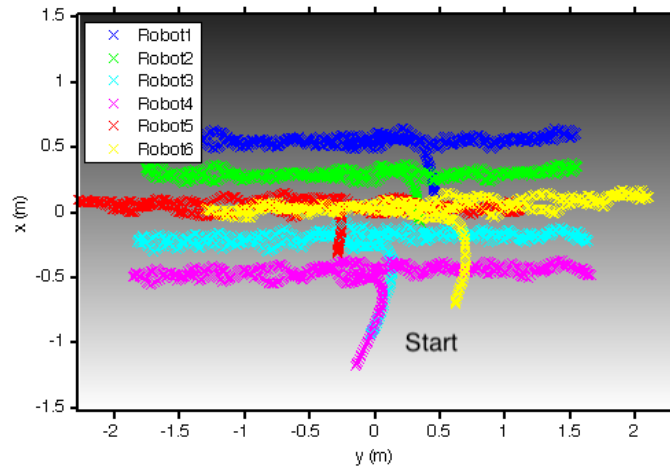


Fig. 4.9: Experimental data for six robots navigating along a front with a linear path, superimposed upon a rendering of the grayscale scalar field [3].

In all cases the tracking errors are on the order of the positioning error of the testbed.

Once again, performance can be analyzed by looking at the measured and desired slopes over time, as in Figure 4.11. The states are plotted over time in Figure 4.12, confirming that the controller is changing modes properly. Once the controller exits state 0, the measured slope  $m$  remains above the threshold  $m_{min} = 2.4 u/m$  (scalar units per meter) until it is commanded to reverse direction. This indicates both that the front is being detected and that the cluster is reversing the direction of travel at the appropriate values. Similarly, the slopes of the subclusters oscillate about the desired setpoint, indicating that they are tracking the feature effectively.

Figure 4.13 is provided to show the repeatability of the technique, and depicts the locations of the cluster and subcluster centroids for five runs on the same scalar field, each of which had different initial conditions. After initial transients, the cluster paths are very consistent. The average and RMS slope errors for the subclusters are listed in Table 4.1. As the distance between robots in a given subcluster is very small, small robot movements can result in large changes in measured error. Despite this, the errors are not too large, and tracking performance is adequate, as confirmed by the overhead plots.

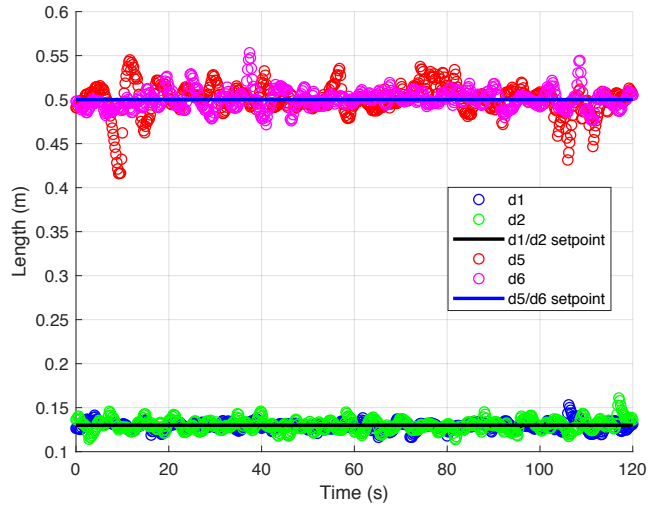


Fig. 4.10: Performance plot for  $d_1$ ,  $d_2$ ,  $d_5$ ,  $d_6$  for the experiment displayed in Fig. 4.9 [3].

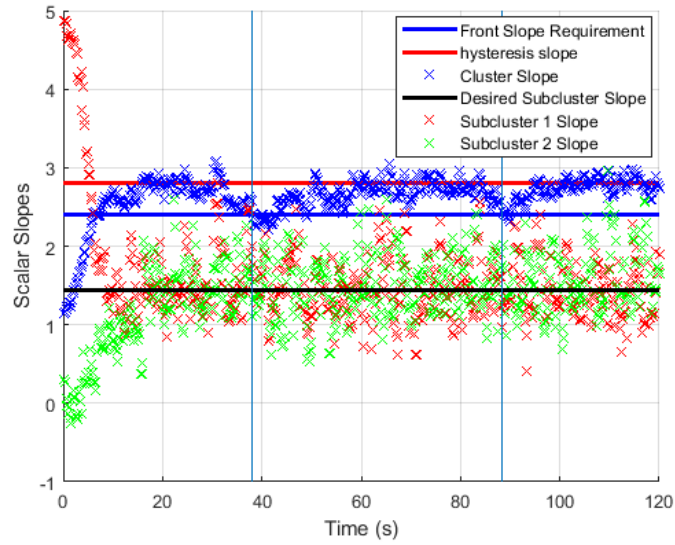


Fig. 4.11: Time history for the measured slope of the front, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.9. Vertical lines indicate points where the cluster begins the process of reversing direction [3].

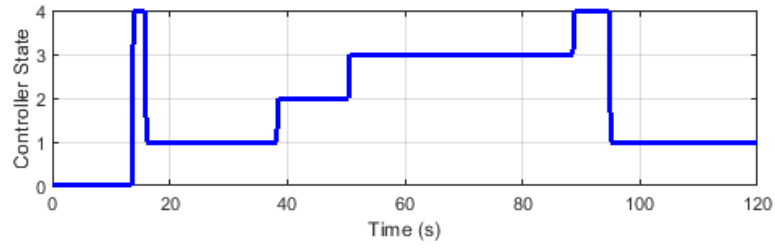


Fig. 4.12: Controller state as the robots travel the path in Fig. 4.9 [3].

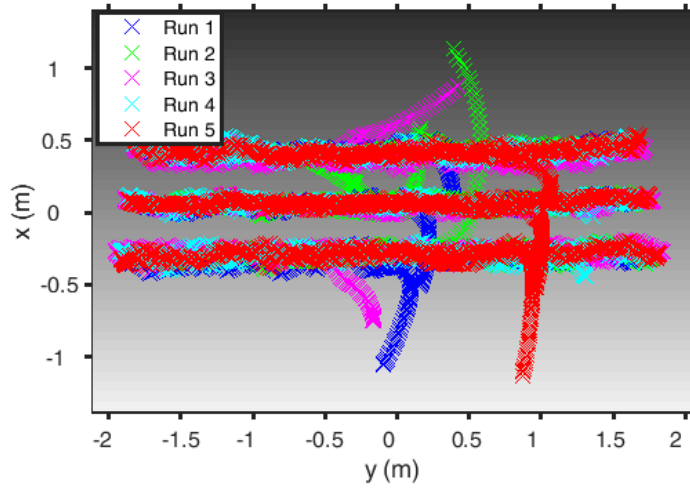


Fig. 4.13: The centroids of each subcluster and the center of the aggregate cluster as it travels along a front, where the run from Fig. 4.9 is in blue [3].

Table 4.1: Slope errors for the subclusters for the straight front tests presented in Fig. 4.13 [3].

	Mean Err. 1 (u/m)	RMS Err. 1 (u/m)	Mean Err. 2 (u/m)	RMS Err. 2 (u/m)
Run 1	0.0345	0.3769	0.0598	0.4024
Run 2	0.0105	0.3767	0.0470	0.4361
Run 3	0.0364	0.3881	0.0107	0.4501
Run 4	0.0045	0.3545	0.0388	0.4613
Run 5	0.0099	0.3753	0.0576	0.4365

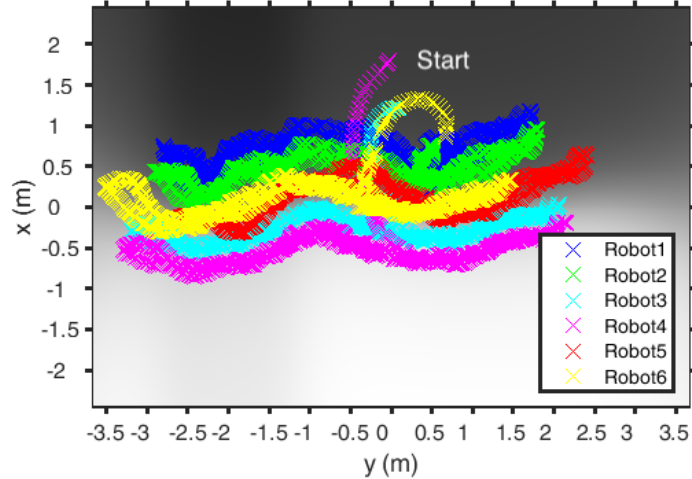


Fig. 4.14: Experimental data for six robots navigating along a front with a sinusoidal path and a ridge feature, superimposed upon a rendering of the grayscale scalar field [3].

### 4.3.2 Sinusoidal Front

The next front has a sinusoidal path, and a localized increase in scalar value to a ridge around  $y = -2$  m. These additions provide additional challenge for the navigator, and showcase the slope-based capabilities of the controller. An example of the cluster navigating the feature is provided in Figure 4.14. Detailed formation data is once again in the appendix.

The tracking performance is analyzed by looking at the time history of the slope values, as in Figure 4.15. This time the slope criteria was  $m_{min} = 1.15$  u/m, and the scaling factor for the subcluster slopes was set to  $P = 0.8$ . Five different runs are summarized with Figure 4.16, all of which have different initial conditions, and are successful. The slope tracking errors associated with these runs are in Table 4.2, and are not significant enough to impact performance.

### 4.3.3 Front with Changing Width

The third and final set of tests were conducted on a scalar front with changing width and slope. Figure 4.17 shows the route followed by the robots, while Figure 4.18 depicts the slope time histories. The slope parameters for this set of experiments were  $m_{min} = 1.1$  u/m and  $P = 0.9$ . Cluster performance information can be found in Appendix B.

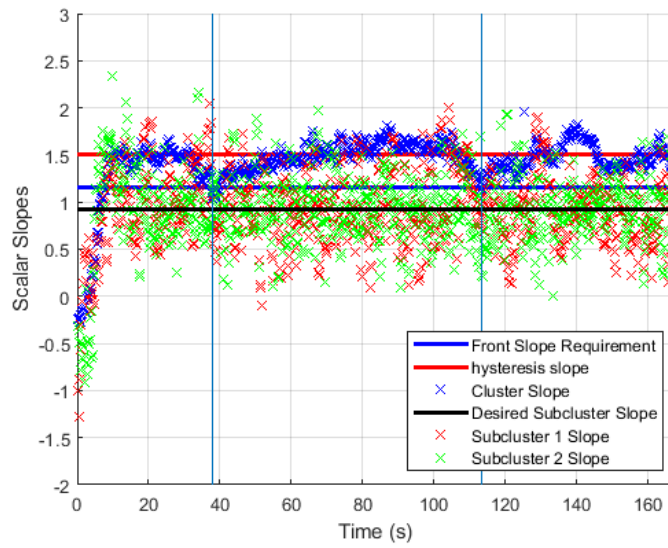


Fig. 4.15: Time history for the measured slope of the feature, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.14. Vertical lines indicate points where the cluster begins the process of reversing direction [3].

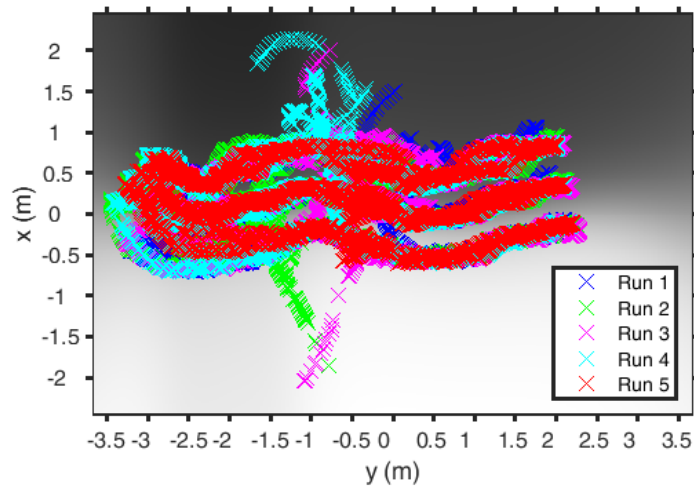


Fig. 4.16: Experimental data for five separate runs, where the centroids of each subcluster and of the aggregate cluster are plotted for each, superimposed upon a rendering of the grayscale scalar field [3].



Table 4.2: Slope errors for the subclusters for the tests presented in Fig. 4.16 [3].

	Mean Err. 1 (u/m)	RMS Err. 1 (u/m)	Mean Err. 2 (u/m)	RMS Err. 2 (u/m)
Run 1	0.0296	0.3644	0.0088	0.3445
Run 2	0.0330	0.3706	0.0153	0.3175
Run 3	0.0241	0.3460	0.0037	0.3236
Run 4	0.0108	0.3216	0.0088	0.3166
Run 5	0.0297	0.3293	0.0052	0.3437

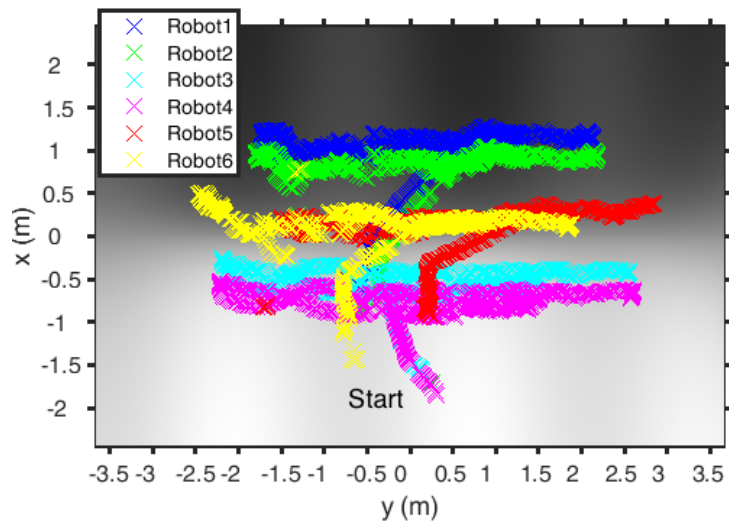


Fig. 4.17: Experimental data for the robots as they navigate along a front with changing width, superimposed upon a rendering of the scalar field [3].

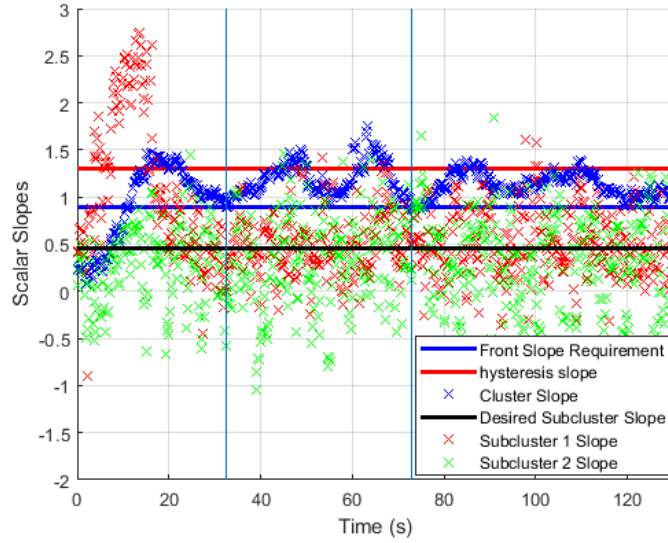


Fig. 4.18: Time history for the measured slope of the feature, and the slopes detected by each subcluster of the formation, for the experiment depicted in Fig. 4.17 [3].

Table 4.3: Slope errors for the subclusters for the tests presented in Fig. 4.19 [3].

	Mean Err. 1 (u/m)	RMS Err. 1 (u/m)	Mean Err. 2 (u/m)	RMS Err. 2 (u/m)
Run 1	0.1264	0.4795	0.1344	0.3157
Run 2	0.0395	0.3294	0.0393	0.4822
Run 3	0.0324	0.2892	0.0937	0.3254
Run 4	0.0462	0.3492	0.1036	0.3276
Run 5	0.1312	0.5866	0.0861	0.4273

Figure 4.19 provides information for repeated runs, all of which are successful after starting with different conditions. While the RMS errors presented in Table 4.3 let to acceptable performance, they are noticeably higher than the tests conducted on other fields. This not surprising due to the increased complexity of this particular scalar field, and the rate at which the slope changes.

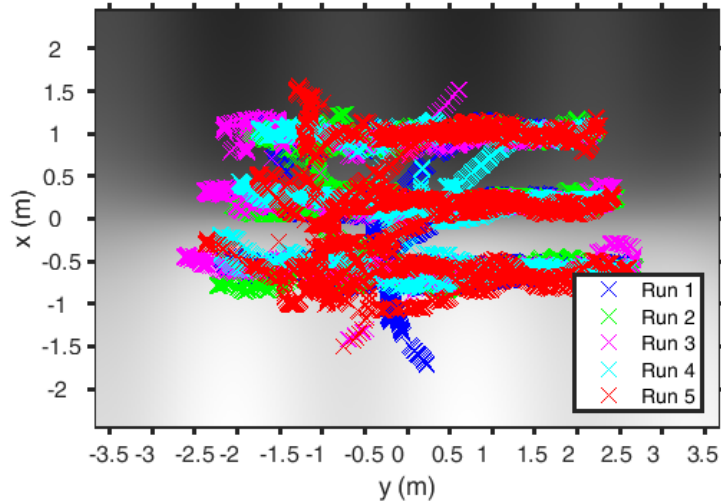


Fig. 4.19: Experimental data for five separate runs on the front with changing width, where the centroids of each subcluster and of the aggregate cluster are plotted for each, superimposed upon a rendering of the grayscale field [3].

## 4.4 Summary

Scalar fronts are a common occurrence in nature, and are seen in a variety of environments, however marine fronts in particular have been the subject of a significant amount of scientific study. While there has been some previous work in adaptively navigating along marine fronts, it has been limited in scope. This work presents the first fully featured, multirobot approach for front navigation, which can navigate a wide variety of fronts, with a variety of sizes and shapes. This technique has now been validated in both a realistic simulation, and in several sets of experiments. In both cases, fronts with a variety of behaviors were used in order to fully exercise the technique. The six robot cluster performed admirably under these varying conditions despite the hardships of real-world effects like latency, packet-loss, field noise, *etc.*

Improvements could be made in order to reduce the amount of *a priori* knowledge about the feature of interest. As discussed previously, setting  $m_{min}$  and  $m_{hyst}$  appropriately is required for good performance, and currently these values are set before navigation. These values could be set adaptively by leveraging time history information while navigating the front. Additional states could also be incorporated into the controller in order to prevent the cluster from traveling indefinitely by using timeouts or setting a spatial boundary for the mission area. The techniques presented in Chapters 5 and 6 demonstrate how the state architecture can handle an out-of-bounds scenario.

---

## CHAPTER 5

# Low-Exposure Waypoint Navigation

This chapter presents a technique for navigating to a predetermined destination without exposing the centroid of the cluster to a scalar value above or below a predetermined limit, or going out of a designated spatial region. The presumption is that there is an asset that needs to be kept safe from a hazardous environment. Another scenario might be supporting travel while maintaining a minimum service level. This is accomplished by using a combination of traditional waypoint navigation and adaptive contour following techniques. While, individually, these navigation schemes are not novel, sequencing them properly and designing the state transition criteria effectively is a significant contribution. The author of this dissertation is the primary contributor for this work.

Avoiding hazards while navigating is a common area of research in mobile robotics. One example [48], which determines whether terrain is a ‘hard’ or ‘soft’ hazard, determines which route to take accordingly. Another is [49] which uses knowledge of regional weather patterns to avoid hazards in flight, while minimizing harmful emissions of the vehicle. While the low-exposure waypoint navigator presented here has a similar objective compared to these examples, it is avoiding a scalar feature rather than physical obstacles or weather patterns. That said, obstacles and weather features could be modeled as scalar fields in order to apply our techniques.

## 5.1 Cluster Formation

In order to focus on the appropriate state-based switching of the adaptive navigation primitives, robot and cluster dynamics were not simulated; the clusters were assumed to be virtual rigid bodies. This also reduced simulation time, an advantageous feature since the navigation capability was evaluated by running 10,000 simulations. In this case the formation was assumed to be an equilateral triangle, as implemented in [25], with a fourth vehicle in the center, as in Figure 5.1. It is noted that a formation with a

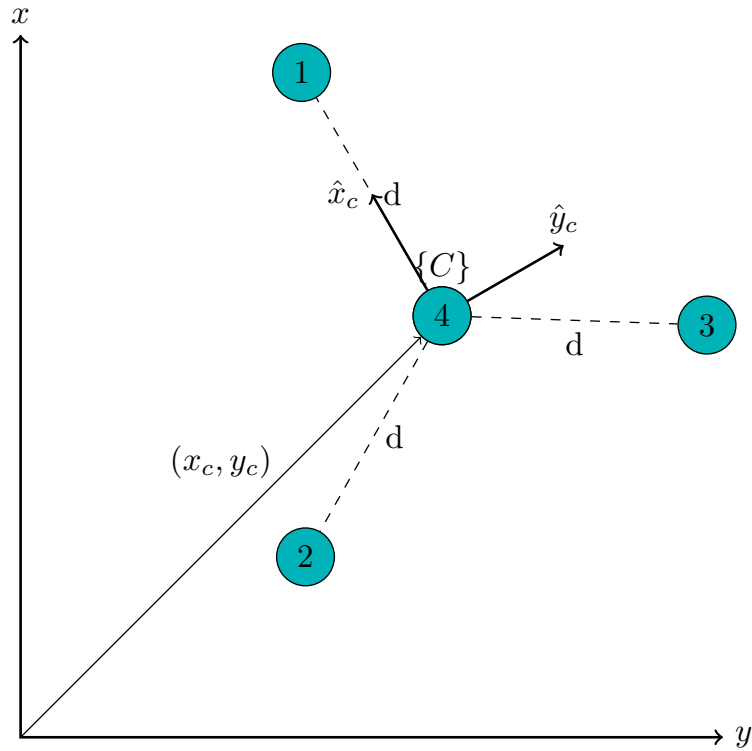


Fig. 5.1: A four robot formation used for low exposure waypoint navigation, contour mapping, and seeking multiple extrema. It is an equilateral triangle with a fourth robot at the center to detect curvature.

minimum of three vehicles is required for the contour following control primitive. The fourth vehicle represents the asset being escorted.

For the stated formation, the cluster frame was placed on the middle robot with  $\hat{x}_c$  pointing to robot 1 and  $\hat{z}_c$  pointed down. The fixed cluster size was denoted by  $d$ . The cluster was assumed to be capable of aggregate holonomic motion with first order dynamics.

## 5.2 Control Strategy

A state-based control technique is used here. An overview of these states and transition criteria are given in Figure 5.2. Three states are used, one of which is a simple go-to, another is the same gradient-based contour following technique used in [25], and the final one is a hysteresis state used when the cluster goes out of bounds and reverses

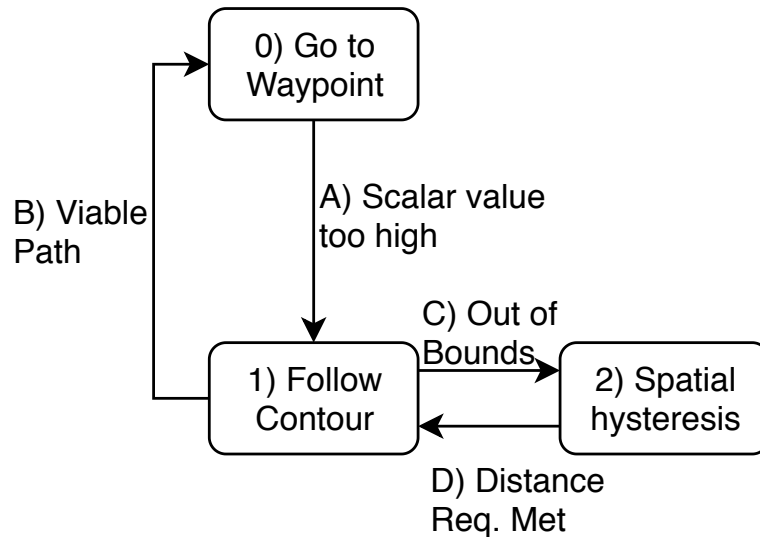


Fig. 5.2: Overview of states and transitions for low exposure waypoint navigation.

its contour travel direction. The formation of robots begins by heading directly toward the target destination until it reaches an unacceptable scalar level, at which point it will switch to contour following until an appropriate route is identified used gradient information.

Figure 5.3 provides a high level view of an example navigation scenario. Figure 5.3A covers the first two states used, the cluster travels directly toward the target destination, reaches the contour, selects a counter-clockwise travel direction based upon local information, and follows the contour line until reaching the boundary. Next, in part B, the cluster reverses direction and tracks the contour in a clockwise direction, until there is a viable route toward the destination, and it leaves the contour line. In C, another local decision is made to travel counter-clockwise, which results in repetitive behavior, and reaching the boundary once again. Finally, in D, the cluster reverses direction, and follows the contour until another clear route to the destination is reached. It does not leave the contour where it did before, as we require it to be significantly closer to the destination to transition again. While this cyclic behavior is not desirable, the state-based behavior successfully limits it in such a way that the cluster will not change states in the same place repeatedly and end up getting stuck in looped behavior.

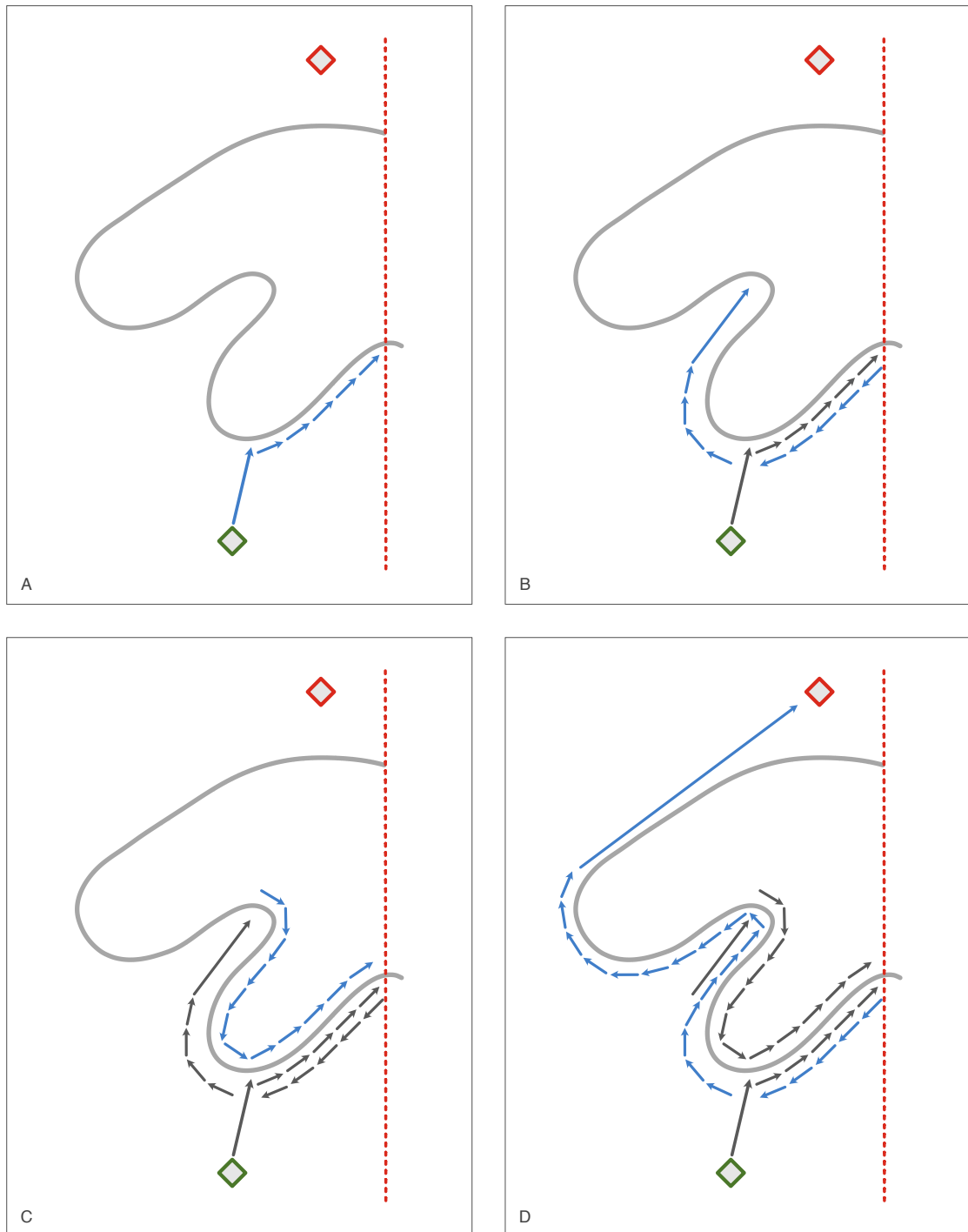


Fig. 5.3: A simplified example of a low exposure waypoint navigation scenario, where the arrows represent the direction of travel as the cluster follows the displayed contour. Each sub-figure displays a portion of the travel, where the new movement is highlighted in blue. The green diamond is the starting point, while the red one is the end point. The red dashed line is a boundary for the navigation area.

## State 0: Go To Waypoint

The first state attempts to go directly to the target destination. Equation 5.1 computes the desired heading based upon the destination and the center of the cluster frame in global coordinates, where  $x_d$  and  $y_d$  are the  $x$  and  $y$  coordinates for the destination.

$$\vec{h} = \begin{pmatrix} x_d - x_c \\ y_d - y_c \end{pmatrix} \quad (5.1)$$

Equation 5.2 is used to compute the  $x$  and  $y$  command velocities for the cluster, where  $R_g^c$  is the rotation matrix from the global frame to the cluster frame. The motion of the overall cluster is holonomic to avoid excess vehicle movement, so the cluster rotation angle  $\theta_c$  is not specified.

$$\begin{pmatrix} \dot{x}_c \\ \dot{y}_c \end{pmatrix} = R_g^c \hat{h} \quad (5.2)$$

The exit criteria for this state is triggered when the scalar value of robot 4 exceeds a predetermined limit, which is set such that there is a safety factor of 10%. The state transition occurs when equation 5.3 evaluates as true. The inequality can be reversed for the case in which the formation needs to remain above a given set point.

$$z_4 > z_{lim} \quad (5.3)$$

This state transition (A) also determines the contour travel direction by comparing the angle of contour vector to that of the destination vector. The gradient vector is computed via Equations 5.4 through 5.7, which is the same method used in [25].

$$\vec{R}_{12} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix} \quad (5.4)$$

$$\vec{R}_{13} = \begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{bmatrix} \quad (5.5)$$



$$\vec{N} = -\vec{R}_{12} \times \vec{R}_{13} \quad (5.6)$$

$$\vec{g} = \begin{pmatrix} N_x \\ N_y \end{pmatrix} \quad (5.7)$$

Using this information, the absolute angle of the local contour is computed using equation 5.8, the angle of the destination vector is computed using equation 5.9<sup>1</sup>.

$$\theta_{cont} = \tan^{-1} \left( \frac{g_y}{g_x} \right) - \frac{\pi}{2} \quad (5.8)$$

$$\theta_{dest} = \tan^{-1} \left( \frac{h_y}{h_x} \right) \quad (5.9)$$

Finally, the contour travel direction,  $d_c$ , is selected by comparing these two angles. If the inequality 5.10 evaluates as true, then  $d_c = 1$ , else  $d_c = -1$ .

$$|\theta_{cont} - \theta_{dest}| < \frac{\pi}{2} \quad (5.10)$$

### State 1: Follow Contour

The second state of this controller is meant to direct the robots around the contour line of maximum allowed scalar value, until a descending path toward the destination is available. The desired bearing of the contour follower is determined using equation 5.11, where  $z_{des}$  is the desired scalar level of the contour,  $z_c$  is the average scalar value of the formation and  $K_{ct}$  is the cross-track gain.

$$b_{cf} = b_{grad} + d_c \{ \text{sgn}(z_{des} - z_c) \\ \times \min [K_{ct} \times \|z_{des} - z_c\|, \pi/2] - (\pi/2) \} \quad (5.11)$$

The  $x$  and  $y$  components are computed using equation 5.12, by simply computing the desired global components from the bearing, then rotating the vector into the cluster

---

<sup>1</sup>Both of these equations make use of the MATLAB ATAN2 function.

frame.

$$\begin{pmatrix} \dot{x}_c \\ \dot{y}_c \end{pmatrix} = R_g^c \begin{pmatrix} \cos(b_{cf}) \\ \sin(b_{cf}) \end{pmatrix} \quad (5.12)$$

The transition (B) to the next state occurs when the destination vector and the negative direction of the gradient are within  $\pi/2$  radians of each other. The angle between the two vectors is computed using equation 5.13.

$$\phi = \cos^{-1}(\vec{h} \cdot \vec{g}) \quad (5.13)$$

While much of the time the controller will perform without issue with this sole criterion, there are some situations in which this will not be effective, as there may be some repetitive motion if the state transitions continue to occur in the same location. In order to prevent this behavior, a secondary criterion was added to ensure that each time the controller exits State 1, it is leaving the contour path at a location that is closer than the previous time the state exited. This criterion requires the inequality 5.14 to be true, where  $\vec{h}_{prev}$  is the destination vector as recorded the last time this state transition occurred, and  $d_{mar}$  is the required additional linear distance toward the destination covered since the last time State 1 was exited.

$$\|\vec{h}\| > \|\vec{h}_{prev}\| + d_{mar} \quad (5.14)$$

Alternatively, if the cluster center passes out of bounds while navigating a contour, the navigator transitions (C) to state 2. The borders of the operational area could be defined any number of ways, however for the simulations presented here, the valid area was rectangular. Given this, the state changes if any of the criteria in Equations 5.15 or 5.16 are met, where the comparison limits are the corners of the rectangle.

$$x_4 > x_{ulim} \text{ OR } x_4 < x_{llim} \quad (5.15)$$

$$y_4 > y_{ulim} \text{ OR } y_4 < y_{llim} \quad (5.16)$$

## State 2: Spatial Hysteresis

Upon entering State 2, the cluster immediately reverses direction ( $d_c = -d_c$ ) in order to move back into a valid operating area. All navigation commands are otherwise the same as State 1. State 2 exists to provide hysteresis when reversing direction; the navigator reverts back to State 1 (transition D) as soon as the centroid of the formation is more than a perpendicular distance  $D_{hyst}$  from the out-of-bounds region. This value was set to 500 meters for the simulations presented here.

## 5.3 Simulation

As there are so many factors in play, these behaviors are difficult to analyze with closed-form mathematical methods, or with limited-run experiments. The technique is validated via 10,000 simulations conducted in parallel.

### 5.3.1 Simulation Environment

The parallel simulations were run using MATLAB/Simulink. Because the goal was to run large numbers of simulations, the dynamics of the cluster were greatly simplified, and the formation was assumed to be rigid. The scalar field of interest was interpolated from a real set of Cs-137 soil samples taken from the Chernobyl Exclusion Zone [50]. Figure 5.4 displays a contour plot of this interpolated radiation data.

### 5.3.2 Results

The sizing and boundary parameters for the batch simulations were  $d = 100\text{ m}$ ,  $y_{lim} = 1.3E5\text{ m}$ ,  $y_{ulim} = 1.62E5\text{ m}$ ,  $x_{lim} = 2.5E4\text{ m}$ , and  $x_{ulim} = -2E4\text{ m}$ . The scalar exposure limit for the center vehicle was set to  $7000\text{ kBq/m}^2$ . Three criteria were set to determine whether a given simulated run was successful: whether or not the cluster made it to the destination, whether the center robot experienced scalar levels in excess of  $1.1z_{lim}$ , and whether the cluster moved more than  $200\text{ m}$  past the boundary of the mission area. Limiting the success criteria allows a large number of simulations to be analyzed without it taking a prohibitive amount of time.

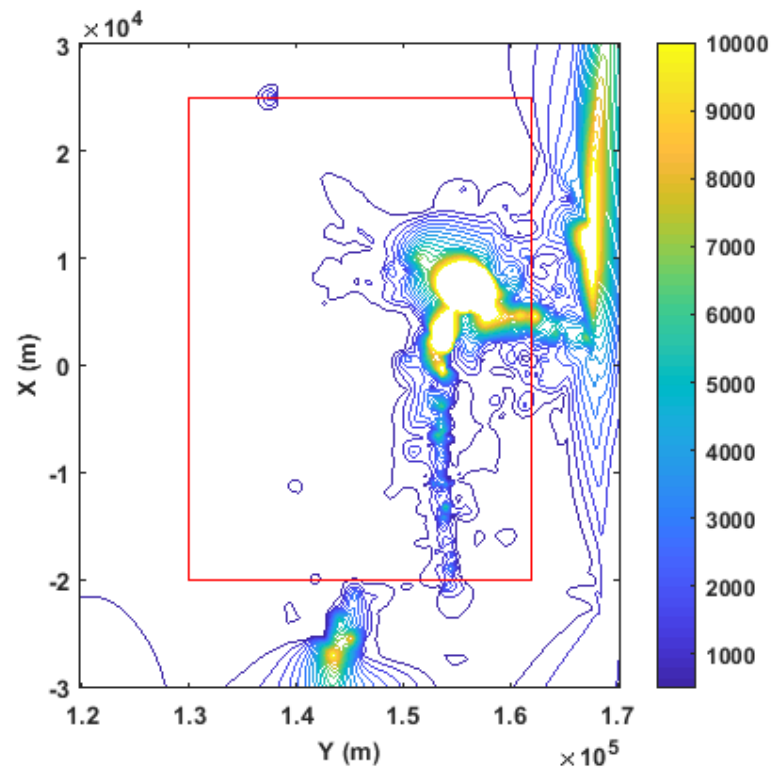


Fig. 5.4: The contour plot for the interpolated Cs-137 in Chernobyl Exclusion Zone soil samples. The red rectangle represents the boundaries set for the simulations conducted in this chapter.

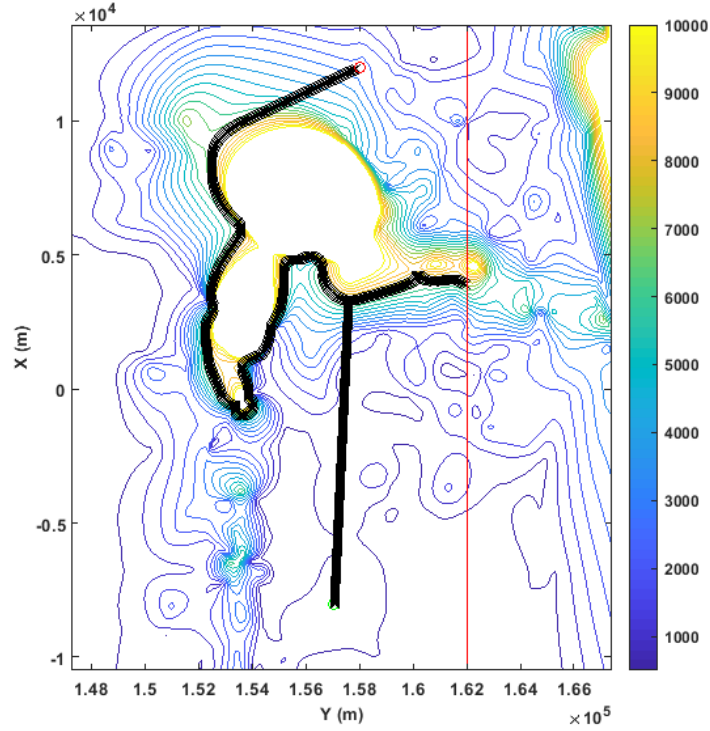


Fig. 5.5: The cluster center positions corresponding to a single test of the low exposure waypoint navigation technique superimposed on the scalar field of interest. The red line indicates the right boundary of the operating area.

Figure 5.5 displays a single simulated run, with a black ‘x’ marking the location of the center robot at each time interval. Green circles denote the starting location for each run, and a red circle marks the finish. The red lines mark the boundaries of the work area; note that the robots reverse the contour following direction upon reaching the edge. In this case, the cluster clearly reaches the final destination successfully.

Figure 5.6 plots the time history of the controller state throughout this test. It begins in state 0, moving directly toward the target destination. When it reaches the contour, it transitions to state 1 and selects a contour following direction based upon the local gradient. There is some repetitive behavior similar to the scenario discussed in the controller section. While backtracking along the contour is not ideal, the state plot clearly shows that it does not get stuck in this repetitive cycle.

The success of this test can be further enforced by examining the scalar value of the center vehicle over time, as in Figure 5.7. In this case, the scalar limit was set to 7,700  $kBq/m^2$ , so we would expect the value to never exceed this value. The plot confirms

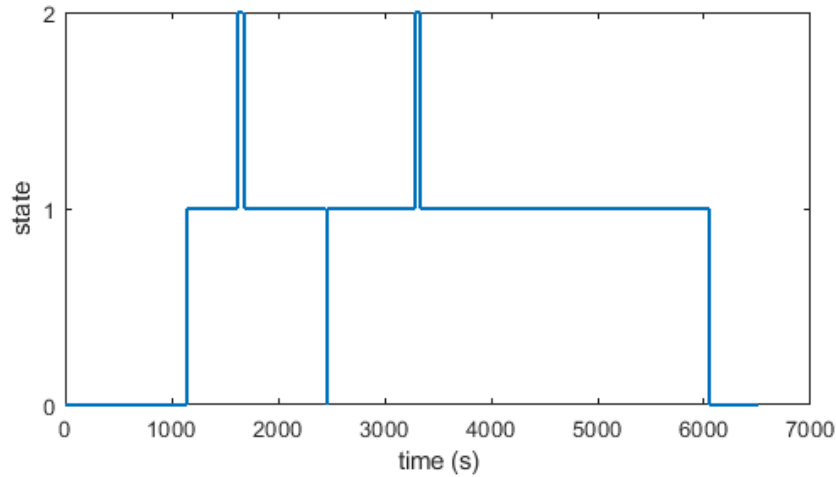


Fig. 5.6: A plot of the time history of the navigator states for the run presented in Figure 5.5.

this is the case, and also contains a long period of time during which the scalar value is in the vicinity of  $7,000 \text{ kBq/m}^2$ . This makes sense, particularly when compared to the state plot, as we know that the cluster was set to follow the 7,000 unit contour for a significant portion of the test.

Figure 5.8 overlays 100 tests with randomized initial conditions. Of the 100 tests, only one passes above the prescribed scalar limit, likely due to tracking error while following the contour, as this was the most commonly observed failure mode. Further tuning of the control gains might alleviate some of these issues; note that only proportional control is used for contour tracking.

In order to further evaluate the reliability of this navigation technique, a batch of 10,000 simulations was run. This time, both the start and end locations were generated from bounded random numbers. All of these successfully made it to the destination, none went out of bounds, and 305 strayed above a scalar value of  $7,700 \text{ kBq/m}^2$ , meaning that 96.95% of the runs were successful. This indicates a high level of repeatability. A future improvement could be to change the factor of safety when selecting the exposure level based upon the sensed gradient of the field. This would allow for more conservative behavior when navigating particularly steep scalar fields.

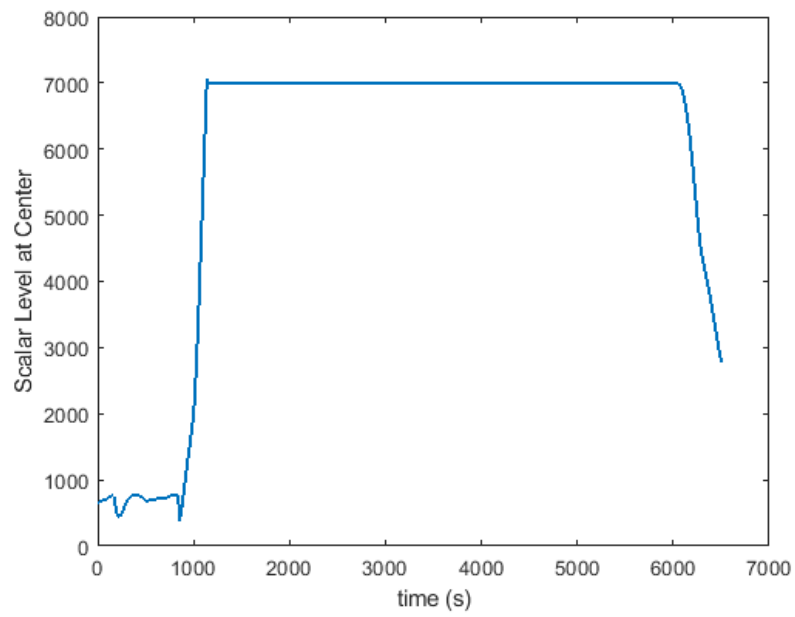


Fig. 5.7: The time history for the scalar value sensed by robot 4 throughout a low exposure waypoint navigation test.

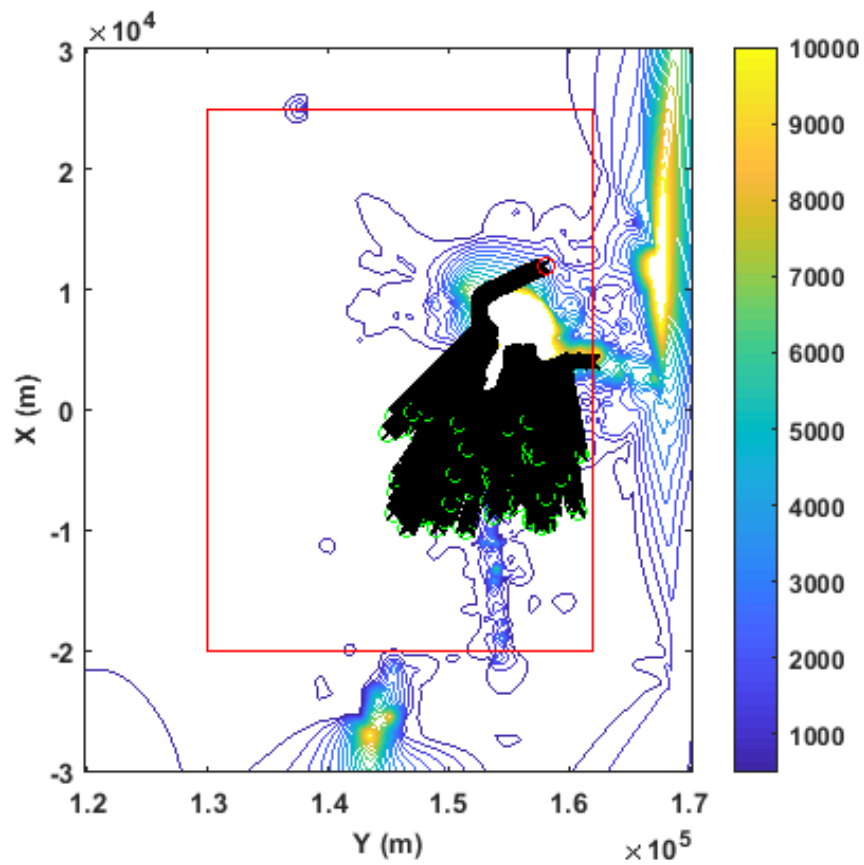


Fig. 5.8: 100 simulated runs displayed on the interpolated radiation data, all of which have the same destination. Green circles represent start points, red indicates the finish, red lines are boundaries.



## 5.4 Summary

The objective of the low exposure waypoint navigation technique is to reach the final destination without the cluster centroid exceeding a predefined scalar level. While some of the vehicle dynamics were simplified in simulation, the technique was successfully vetted using 10,000 simulations on a realistic scalar field based upon real field samples. All recorded failed runs were due to moving above the set scalar limit, as opposed to getting stuck on features of the field, or moving too far out of bounds. The number of failures could likely be reduced by improving the contour tracking beyond a simple proportional controller, or adapting the safety factor of the scalar set point.

---

## CHAPTER 6

# Mapping Contours Around an Extrema

This chapter presents a state based technique for mapping the contours around a local extrema. Contour mapping has a variety of potential applications for environmental characterization; for example, it could generate a focused topographic altitude/-bathymetry map around a peak, or establish exposure perimeters around a hazardous source. This is accomplished by following several contours of decreasing/increasing scalar value which encircle a maximum/minimum. Upon fully circumnavigating one contour, the navigator selects the next contour scalar value. The example scenario examined here is the evaluation of a radiation hazard, using the same data set as in Chapter 5. The author of this dissertation is the primary contributor for this work.

While there has been significant interest in using mobile robots to gain information about a radiation field, this work has largely been constrained to source-seeking and mapping an environment using prior knowledge. Examples include [51] which uses a single mobile robot equipped with a Geiger counter to search for radioactive sources, [52] which maps a known environment to locate radiation generated by a cyclotron, and [53] which takes images at multiple locations in order to estimate the radiation field. The contour mapping technique presented here, rather than simply locating extrema, aims to find them, then map the region around them in order to determine the extent of the hazard. While the technique described in [53] can generate similar information, it is restricted by the need for a sophisticated imaging device, and is less suitable for large scalar fields, as it would require a large number of images.

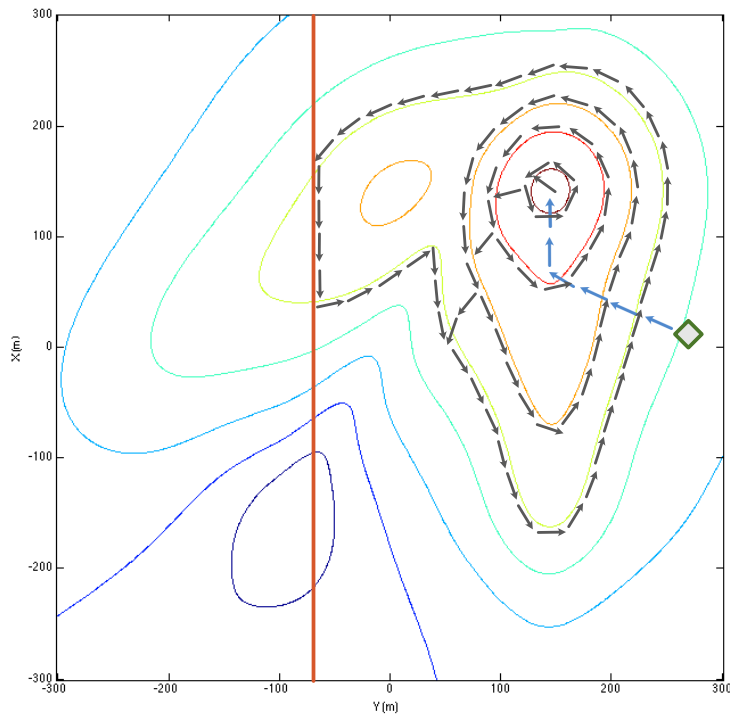


Fig. 6.1: An example contour mapping scenario, starting at the green diamond.

## 6.1 Control Strategy

This technique requires the formation to provide contour following, gradient ascent/descent capabilities, and the ability to sense local curvature. To accomplish this, we use the same formation as in Chapter 5. Figure 6.1 provides an example of the contour mapping navigator in use on a scalar field, where the direction of travel for the cluster is represented by arrows. The controller begins in extrema-seeking mode (marked in blue) until it reaches a peak, then transitions to a state designed to descend incrementally to a target contour level, then to another state for following said contour. The controller continues to descend to and map contours at constant intervals as appropriate, terminating when a predetermined number of contours have been mapped. There is an additional state that is triggered if the cluster passes out of the pre-designated work area, the boundary for which is marked with a red line in the example figure.

Ideally each contour mapped will circumnavigate the peak of interest, however there are scenarios in which the cluster of robots follows a closed contour that does not encircle

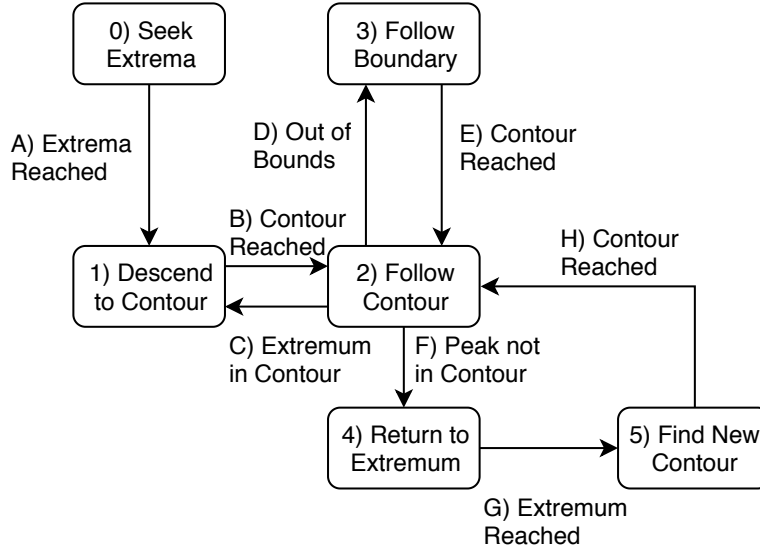


Fig. 6.2: The state diagram for the contour mapping technique. It includes four states and five transition criteria.

the peak. For example, the cluster may descend to a contour which encircles a local minimum. We wish to mitigate this circumstance in two ways. First of all, we need to have a method of checking to see whether the contour encloses the peak. Second, we need a way to recover from this scenario, and preferably, locate a viable contour. The state controller accomplishes this by detecting the behavior, returning to the extrema, and seeking a contour in a new direction. A four robot cluster shown in Figure 5.1 is used to implement this functionality. An overview of the states and transitions used by the navigator is provided by Figure 6.2.

### State 0: Extrema Seeking

The extrema seeking state uses a gradient based approach that we have used previously in other work. The controller commands the cluster to move forward at a constant velocity headed in the direction of ascent. The  $\dot{\theta}_c$  command is generated using Equation 6.1 where  $\omega_c$  is a constant velocity set point,  $\dot{x}_c$  is set to a constant value, and  $\dot{y}_c$  is set to zero.

$$\dot{\theta}_c = \omega_c \text{sign}(\tan^{-1} \left( \frac{g_y}{g_x} \right) - \theta_c) \quad (6.1)$$

The primary and only exit criterion for this state occurs when the extrema is detected,

as determined when the inequality in 6.2 evaluates as true for all  $i = [1 : 3]$ .  $z_{sens}$  is a preset sensitivity parameter which is included to prevent sensor noise from triggering a premature transition (A).

$$z_4 > z_i + z_{sens} \quad (6.2)$$

After this criteria is met, the desired level of the contour to follow  $z_{des}$  is set to  $z_4 - z_{drop}$ , where  $z_{drop}$  is the desired spacing between contours to be mapped. The coordinates of the extrema,  $(x_{peak}, y_{peak})$ , are also recorded.

### State 1: Descend to Contour

This state is responsible for descending to the correct contour from either the previous one, or the extrema. The cluster orients to the gradient, while descending the gradient to reach the correct scalar value, and moving perpendicular to the gradient in the contour direction. The orientation controller for this state is the same as that of state 0. The  $\dot{x}_c$  command is set using equation 6.3 where  $v_c$  is a constant speed set point, and the sign of the  $\dot{y}_c$  command is set to either 1 or  $-1$  depending on whether we wish to circle the contour clockwise or counter-clockwise.

$$\dot{x}_c = v_c \text{sign}(z_{des} - z_4) \quad (6.3)$$

As the primary goal of this state is to ensure we are properly on, and aligned with the desired contour, the primary exit criteria (to use transition B) are twofold; the criterion in Expression 6.4 ensures the cluster is at the right scalar value, and the criterion in Expression 6.5 ensures it is aligned properly.

$$|z_{des} - z_4| < z_{sens} \quad (6.4)$$

$$|\theta_{des} - \theta_c| < \theta_{sens} \quad (6.5)$$

Finally, there is a secondary temporal criterion, for the purpose of mitigating any transient behaviors. This simply requires that the difference in time between the entry and exit of this state is above a given threshold. Once all the criteria have been met, several

values are computed and recorded for later use, including the vector from the peak to the robot position as in Equation 6.6.

$$\vec{P}_c^p = \begin{pmatrix} x_c - x_{peak} \\ y_c - y_{peak} \end{pmatrix} \quad (6.6)$$

## State 2: Follow Contour

This state simply follows a given contour line at the scalar level  $z_{des}$  until it has been circumnavigated. This is done by moving in the contour direction until returning to the starting position. The contour following controller itself is the same as the one presented in State 1. Two quantities are computed and then used to determine whether the circumnavigation has been completed: the radius ( $\|\vec{P}\|$ ) and angle (Equation 6.7) relative to the peak, where  $P_x$  and  $P_y$  are the  $x$  and  $y$  components of  $\vec{P}$ , respectively.

$$\theta_v = \tan^{-1} \left( \frac{P_y}{P_x} \right) \quad (6.7)$$

In order to ensure that the contour has been fully navigated (transition C), the cluster is required to reach the starting point on the contour. A secondary criterion is required in order to prevent the navigator from falsely transitioning when passing this location the first time around. This is achieved through the use of a boolean flag that indicates whether the cluster has reached a point sufficiently far away from the starting point on the contour, using the radius and angle requirements in Equations 6.8 and 6.9, where the minimum values are set based on how far the cluster needs to be from the starting point.

$$|(\|\vec{P}\| - \|\vec{P}_{start}\|)| > R_{min} \quad (6.8)$$

$$\theta_v - \theta_{vstart} > \theta_{min} \quad (6.9)$$

Once these criteria have been met, the requirement to exit this state with transition C is that the cluster reaches the starting point on the contour. This condition is represented by Equations 6.10 and 6.11, where  $R_{mar}$  and  $\theta_{mar}$  are the radius and angle margins of

error. These values are set based on the field characteristics, expected noise, *etc.* They are necessarily smaller than  $R_{min}$  and  $\theta_{min}$ .

$$\|\vec{P}\| = \|\vec{P}_{start}\| \pm R_{mar} \quad (6.10)$$

$$\theta_v = \theta_{vstart} \pm \theta_{mar} \quad (6.11)$$

After determining that the full contour has been navigated, we need to determine whether it encircles the extremum of interest. To do this we use the time history of the spatial position of the cluster center, and determine whether the peak is within the polygon generated by these points. This is accomplished using the `inpolygon()`<sup>1</sup> MATLAB function. If the peak is within the contour the navigator transitions to state 1 and descends to the next contour of interest, if it is outside, it enters state 4 in order to find a better route.

Should the cluster reach the boundary of the region of interest and triggers transition D, the controller enters state 3.

### State 3: Move Along Boundary

This state is entered from states 0 through 2 upon crossing the boundary of the work area. Assuming the work area is rectangular and aligned with the global frame (as it is for the simulations presented in this work), equations 5.15 and 5.16 from Chapter 5 are used to check this condition. The goal of controller state 3 is to direct the cluster to navigate along a boundary line until it reaches conditions appropriate for returning to the previous state, or another boundary line. Throughout navigation the cluster is aligned with the boundary line with a constant forward command velocity and uses a lateral velocity command to track the line.

The direction vector of the boundary line in question is  $\vec{v}_{bound}$ . In the simulations conducted here, the boundary lines are always aligned with either the  $x$  or  $y$  axes. The cluster aligns with this vector using Equation 6.12.

---

<sup>1</sup>This function is based on the methods used in [54], which takes elements from the standard ray casting and winding number algorithms and produces a more computationally efficient result.

$$\dot{\theta}_c = \omega_c \tan^{-1} \left( \frac{v_{bound-y}}{v_{bound-x}} \right) \quad (6.12)$$

The cluster  $\dot{x}_c$  is simply set to a constant value, and  $\dot{y}_c$  is set to a constant value with a sign that changes based on whether the cluster is inside or outside the boundary line. Upon exiting this state the controller returns to the previous state, and the criteria is different based on which state it is returning to. If returning to state 0, the controller requires the gradient vector to be pointing away from the edge of the work area. If returning to state 1 or 2, the transition occurs once the scalar value of the center robot is close to the value of the desired contour, as in Equation 6.4. Alternatively, if the cluster reaches another boundary line before these conditions are met, it will change to a different sub-state in order to continue along the next boundary line.

#### State 4: Return to Extremum

The objective of states 4 and 5 is to recover from a scenario in which the cluster is circling a contour that does not encircle the extremum of interest. State 4 uses simple waypoint navigation to return to the extremum, computing the desired velocities via Equations 6.13 and 6.14.  $R_g^c$  is the rotation matrix from the global frame to the cluster frame as before, and  $k$  is a proportional gain.

$$\vec{h} = \begin{pmatrix} x_{peak} - x_c \\ y_{peak} - y_c \end{pmatrix} \quad (6.13)$$

$$\begin{pmatrix} \dot{x}_c \\ \dot{y}_c \end{pmatrix} = k R_g^c \hat{h} \quad (6.14)$$

The exit criterion for this state is given by Equation 6.15, which requires the magnitude of the peak vector to be below a maximum radius  $R_{max}$ .  $R_{max}$  is typically a very small value, effectively indicating that the formation has returned to the initial location of the extremum.

$$\|\vec{h}\| < R_{max} \quad (6.15)$$

Upon reaching this location the controller enters State 5.



## State 5: Find New Contour

State 5 is responsible for locating a contour in a new direction. The navigator determines which direction to travel by checking an array of available travel bearings. This array represents a predetermined number of angular intervals relative to the peak of interest. The time history from the previous contour following step is used to determine which intervals have already been navigated; these intervals are removed from further consideration. The next viable entry in the array is then selected as the travel direction.

Given this, the command velocities for the cluster are set to match the bearing of the center of the angular interval, as in Equations 6.16 and 6.17, where  $\theta_u$  and  $\theta_l$  are the upper and lower bounds of the interval, respectively.  $k$  is a proportional gain.

$$\dot{x}_c = k \cos \left( \frac{1}{2}(\theta_u + \theta_l) \right) \quad (6.16)$$

$$\dot{y}_c = k \sin \left( \frac{1}{2}(\theta_u + \theta_l) \right) \quad (6.17)$$

The state is exited with transition H, when the scalar value of the cluster is acceptably close to the desired contour value, using the same criterion as 6.4. After a contour of the desired scalar level that encircles the peak is navigated, the array of viable travel directions is reset.

## 6.2 Simulation Results

Two scenarios are evaluated to test the performance of the contour mapping navigator. The first uses a simulated scalar field, designed to trap the cluster on non-viable contours. The second is the same radiation field from the Chernobyl exclusion zone used in the previous chapter. The simulations were conducted using the same MATLAB/Simulink software architecture as in Chapter 5.

### 6.2.1 Testing on a Simulated Field

The scalar field used for these simulations was an artificially generated set of features, with a dominate peak in the center, surrounded by three minima. Figure 6.3 demon-

strates the scenario in which the navigator is initially stuck on a contour, but successfully recovers several times to ultimately complete the mapping task. Part (a) of the figure shows the initial scenario where the navigator would get stuck if not for states 4 and 5. (b), (c), and (d) show the rest of the progression, as the navigator recovers successfully from multiple failure conditions.

1,000 simulations were run on this field without the recovery states, all of which failed. Additionally, all 1,000 of the simulations run with the recovery states successfully mapped three contours. This demonstrates that the recovery technique is a viable method of avoiding this particular failure scenario, particularly when navigating simple scalar fields.

## 6.2.2 Simulating on Real-World Radiation Data

This set of simulations were run using interpolated real-world soil radiation data. Figure 6.4 plots the position for the cluster center over time for a single simulated run. Note that the cluster clearly follows the boundary line upon reaching it, and continues until reaching the contour line on the other side. Figure 6.5 provides a more detailed view of a subsection of the run in order to better display the initial motion. Here the cluster appears to follow the first contour for an additional half loop; however, the controller was not in state 2 for this entire duration. After the initial descent from the peak, approximately one-half loop of pseudo-contour following (i.e., in state 1, not state 2) occurred until the scalar and alignment criteria of Equations 6.4 and 6.5 were satisfied. Hence, the official navigation of the contour (i.e., in state 2) did not commence until the cluster was near the point where it subsequently descended to the second contour.

Figure 6.6 plots the time history of the navigator states throughout the same test. This information can be used to verify that the state level of the navigator is working properly. As we would expect, the cluster begins in state 0 in order to follow the gradient toward a local extrema. Once the local topography indicates the maxima has been reached, the navigator enters state 1 in which it descends toward a contour  $200 \text{ kBq/m}^2$  below the peak. As mentioned previously, the controller remains in this state for a significant amount of time as it settles to the correct scalar value. Upon reaching it, the controller uses state 2 to follow the contour. This process is repeated two more times as contours are mapped. As seen from the time history, the time spent in state 1 is lower in each of these cases, indicating a dependence on the local scalar field, and also

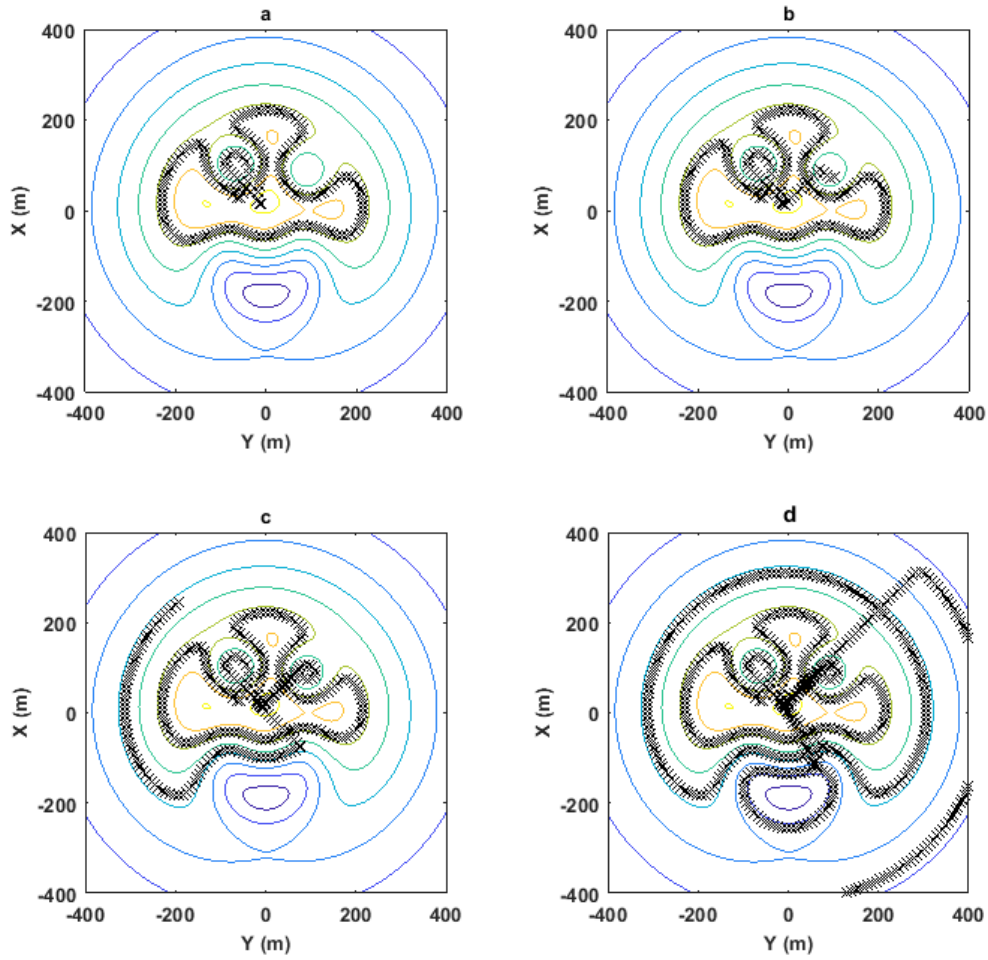


Fig. 6.3: The contour mapping navigation strategy operating on a simulated scalar field. (a) shows the cluster navigating a contour that does not encircle the peak, and in (b) the navigator recovers from this scenario and begins navigating another invalid contour. In (c) the navigator recovers again to find a viable route, and finally in (d) the cluster recovers once again and continues onward.

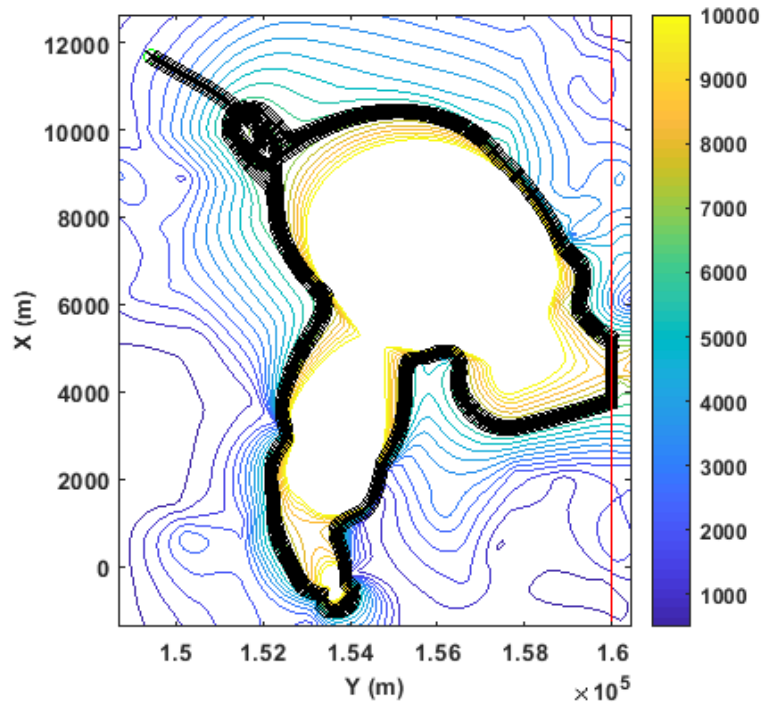


Fig. 6.4: A single contour mapping test conducted on a scalar field interpolated from soil samples taken in the Chernobyl Exclusion Zone.

that the navigation algorithms are robust to these variations. State 1 served its purpose by ensuring good contour alignment for the mapping task in state 2. The cluster of robots eventually began to follow a contour with a much longer perimeter, which lead it to reach the edge of the mission area, triggering a transition to state 3. It remains in this state until the cluster returns to the appropriate scalar level. In this particular test, this sequence occurs twice. The test ultimately terminates upon mapping three contours successfully.

Figure 6.7 can be used as another way to examine the performance of the contour mapping controller. In the beginning of the time history the scalar level climbs consistently until reaching the local peak, after which it steps down as it maps each contour level. There is a large, temporary spike while in state 3, as the scalar level is not being controlled while traveling along the boundary, and high exposure regions are passed before reaching the contour once again.

Ultimately the goal of the contour mapping technique is to approximate the scalar field, which it accomplishes by tracing contour lines. A well behaved field with a limited number of features would be easily reproduced with simple contour tracing. That said,

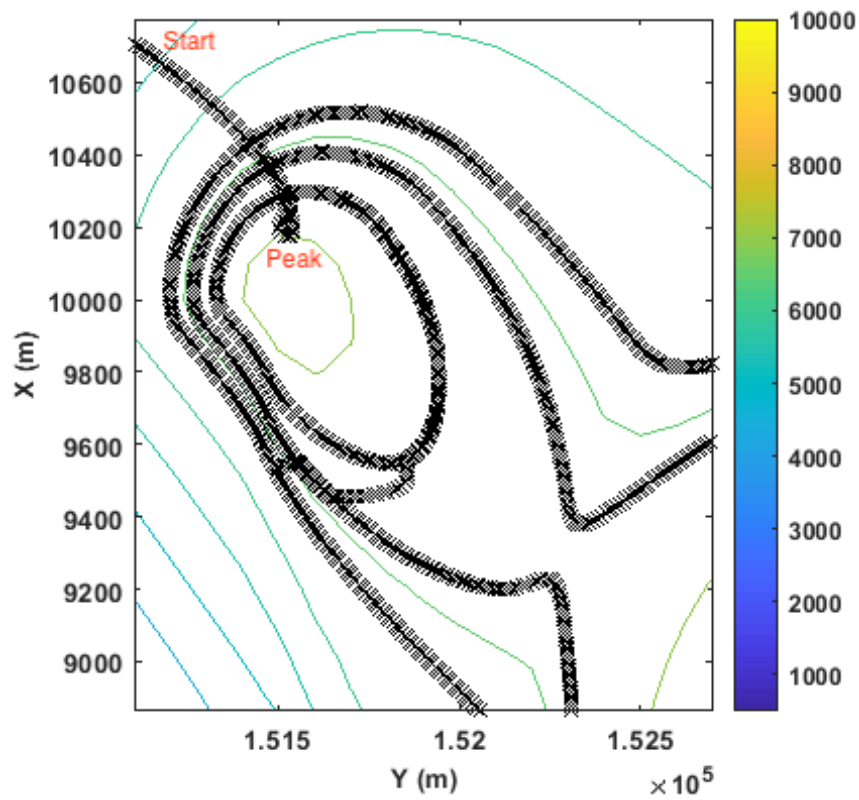


Fig. 6.5: A subsection of Figure 6.4 in order to provide a more detailed view of the beginning of the motion. A small portion of each of the three individual contours followed can be seen in this view.

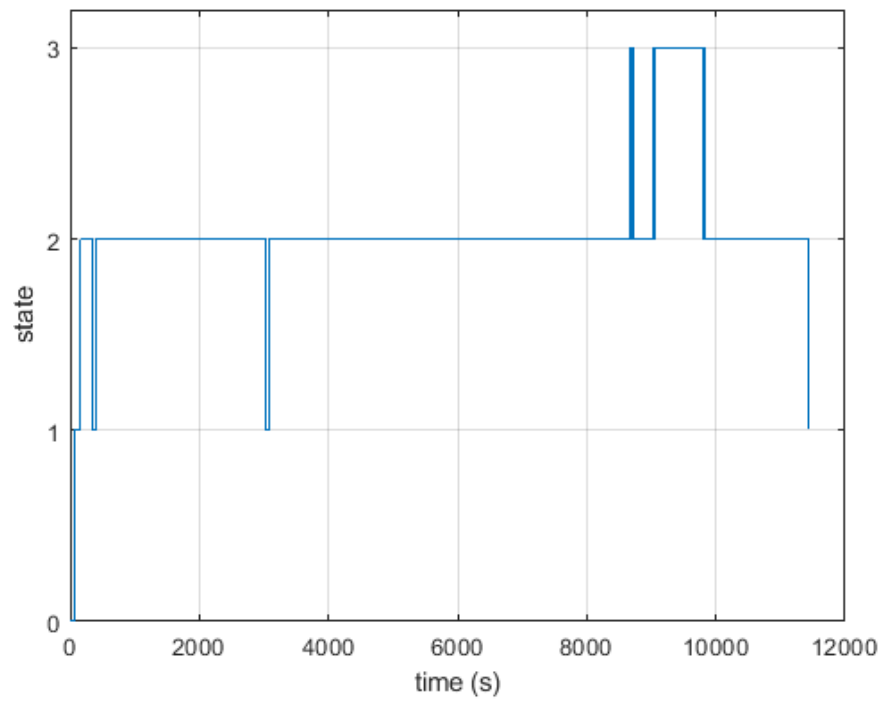


Fig. 6.6: Time history of the controller state for the test displayed in Figure 6.4.

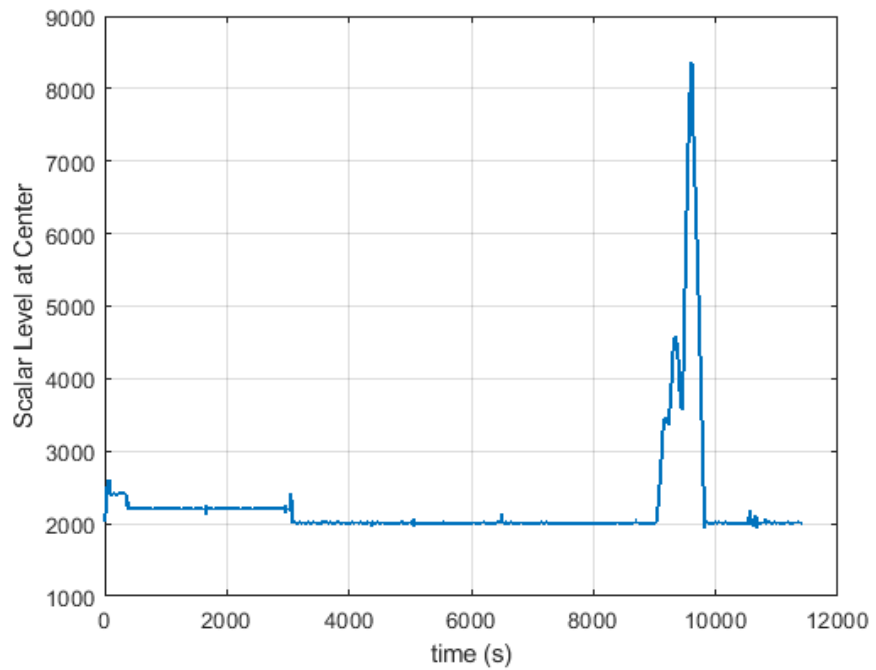


Fig. 6.7: Time history of the average scalar value for the test displayed in Figure 6.4.

fields in nature typically have many different features, including multiple local extrema. Strictly following contours will often lead to bypassing some of these secondary features. For example, the test in Figure 6.4 locates a local extrema, and bypasses several other features as it continues to map contour lines. While the navigation algorithm is still performing as designed in these cases, it does reduce the accuracy of the reconstruction of the field.

10,000 simulations were conducted in order to demonstrate the repeatability of the technique. The initial cluster positions were determined by generating uniformly distributed pseudo-random integers such that the  $x$  position was  $154,000 \pm 5,000$  m, and the  $y$  position was  $4,000 \pm 9,000$  m. Once again the contour spacing was set to  $200$  kBq/m<sup>2</sup>. Three fully circumnavigated contours were required in order to declare success. 87.37% of these tests were successful and stayed within the boundaries. Of the remaining unsuccessful runs, 219 detected that non-viable contours were circumnavigated, but were unable to subsequently find a suitable direction of travel. For these simulations there were eight angular intervals in the array, so some of these errors could be reduced by increasing the granularity of the intervals.

The remainder of the failed runs were due to reaching the time limit of the simulation, and are summarized in Table 6.2.2. It is difficult to diagnose the exact issue for each run, but many of these may have simply run out of simulation time. The recovery and boundary following states in particular can add a significant amount of travel time for the cluster, as it effectively increases the number of contours to be navigated. Alternatively, some of these errors may have been due to the tuning of the transition criteria relative to a particular region of the scalar field. Most of these runs ended in state 2, which indicated either the contours were too long to follow in a timely manner, or the local characteristics of the field were such that a transition was prevented. State 1 likely had similar issues of time and transition criteria, as the control equations are the same, however as it is simply descending to a contour, the time spent in this state is much lower, corresponding to a lower number of timeouts. The runs ending in state 3 were navigating the boundary of the work area which requires a substantial amount of movement, and may have simply taken too much time, or may have been seeking a scalar value that was not present in the work area. Similarly, the runs ending in state 0 may have traveled along the boundary for a significant amount of time before reentering the state. Finally, those ending in 4 or 5 either spend too much time checking available directions, or were unable to find a viable contour while moving outward from the peak in state 5.

Table 6.1: Simulation End Conditions

Outcome	Number of Runs
Success	8,737
Surrounded by non-viable contours	219
State 0	73
State 1	20
State 2	440
State 3	203
States 4 and 5	308

### 6.3 Summary

The goal of the contour mapping technique is to estimate the scalar field in a region around a local extrema. The effectiveness of the recovery states was verified with a 100% successrate on the artificial scalar field. The simulations conducted on the real-world data also had a very high rate of success. 87.37% of them successfully mapped three field contours, indicating that the contour mapping strategy presented in this chapter is both viable and reliable. Failures typically occurred when there was no way to both continue the contour navigation and remain in the designated area. A potential future improvement could be adaptively selecting the contour spacing based upon the gradient of the local field; closer spacing could be used in steeper areas, and wider spacing in flatter regions. Similarly, adaptively selecting some of the contour tracking scalar requirements could account for a wider variety of scalar fields.



---

# CHAPTER 7

## Conclusion

This chapter discusses the current achievements of state-based adaptive navigation, and some potential future work that could expand its capabilities.

### 7.1 Summary

Adaptive navigation of scalar fields encompasses a powerful set of methods for locating points of interest in a region, and executing tasks based upon environmental data. The core objective of this work is to use state-based capabilities to improve the performance of adaptive navigation techniques and to enable a wider variety of applications.

The state-based architecture has been created and integrated with the existing adaptive navigation architecture. It has the capability to sequence adaptive navigation primitives as necessary to accomplish the given task, and can accept a wide variety of transition criteria to govern the switching behaviors. This has been demonstrated to work in a variety of circumstances.

This work has added state-based behavior to improve the performance of ridge, trench, and saddle point navigation. This enhancement allows the navigator to locate a nearby feature, and to recover should it lose the ridge/trench while moving. Performance was verified with a full set of experiments using scalar fields represented by a variety of grayscale printouts.

The front following technique was fully developed within this work, and required not only the development of the state behaviors, but also new adaptive navigators. This front following technique was implemented in both a thorough simulation and laboratory experiments. Two additional mission-oriented techniques were created in this work, a method for navigating to a waypoint without passing above a present scalar level, and a method for mapping contours around an extremum of interest. Both of these techniques

have been demonstrated for the first time, simulated using a real-world radiation data set.

This front navigator includes adaptive sizing of the robot cluster based on the local scalar field. Not only was this critical for this work, but it is the first step toward adding resizing to other navigation primitives. Finally, the front and ridge/trench navigators required the development of two novel cluster formations, the associated kinematic transformations, and the velocity relationships in terms of the Jacobian transforms.

All of these improvements constitute significant advances in the area of multirobot adaptive navigation. Additionally, they provide the framework for future techniques and improvements.

## 7.2 Future Work

The research to date has opened many avenues for future work. While the cluster size is already varied depending on the environment, the sizing choices are typically made with the use of *a priori* knowledge of the field of interest, and is fixed for the duration of the navigation. The front navigation technique leverages adaptive sizing, but strictly for spanning fronts. Future research could provide the capability to generically adaptively size the cluster throughout the mission in order to optimize for the spatial frequencies of interest. This would improve the effectiveness of the individual primitive navigators.

In addition to the sizing of the cluster itself, there is also the notion of adapting the state transition and navigation thresholds based on the characteristics of the local scalar field. For example, it may be beneficial to increase the contour following safety factor for the low exposure waypoint navigation technique based on the magnitude of the local gradient of the field, adjust the front slope thresholds based on the observed slopes throughout navigation, *etc.*

Multirobot adaptive navigation can also be extended in a number of more general ways. For example there is research being conducted into additional highly decentralized navigation methods, and methods for extending some of the navigation strategies into three dimensional applications. There is also interest in experimental verification and field testing of navigators that have not yet been used in full field applications. For example, there is a ridge of geological significance in the Emerald Bay region of Lake Tahoe California that could be navigated with ASVs.

---

# Bibliography

- [1] 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.
- [2] R. T. McDonald, C. A. Kitts, and M. A. Neumann, “Experimental implementation and verification of scalar field ridge, trench, and saddle point maneuvers using multirobot adaptive navigation,” *IEEE Access*, vol. 7, pp. 62950–62961, 2019.
- [3] R. T. McDonald, M. Condino, M. A. Neumann, and C. A. Kitts, “Navigation of Scalar Fronts with Multirobot Clusters in Simulation and Experiment,” *Accepted by IEEE Systems Journal*, 2020.
- [4] T. Xinchu, Z. Huajun, C. Wenwen, Z. Peimin, L. Zhiwen, and C. Kai, “A research on intelligent obstacle avoidance for unmanned surface vehicles,” in *2018 Chinese Automation Congress (CAC)*, pp. 1431–1435, Nov 2018.
- [5] Xiaohao Xu, Chenggong Li, and Yifei Zhao, “Air traffic rerouting planning based on the improved artificial potential field model,” in *2010 Chinese Control and Decision Conference*, pp. 1444–1449, May 2010.
- [6] E. Burian, D. Yoerger, A. Bradley, and H. Singh, “Gradient search with autonomous underwater vehicle using scalar measurements,” in *IEEE OES AUV Conference*, pp. 86–98, 1996.
- [7] M. Dunbabin and L. Marques, “Robots for environmental monitoring: Significant advancements and applications,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.
- [8] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, “Environmental monitoring using autonomous vehicles: a survey of recent searching techniques,” *Current opinion in biotechnology*, vol. 45, pp. 76–84, 2017.

- [9] H. Ishida, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata, “Plume-Tracking Robots: A New Application of Chemical Sensors.,” *Biological Bulletin*, no. 2, p. 222, 2001.
- [10] 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.
- [11] 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.
- [12] 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 *10th IEEE International Conference on Control and Automation (ICCA)*, pp. 1754–1759, Jun 2013.
- [13] Woods Hole Oceanographic Institution, Woods Hole, MA, *Autonomous Benthic Explorer configuration*, 1996.
- [14] P. Ogren, E. Fiorelli, and N. E. Leonard, “Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment.,” *IEEE Transactions On Automatic Control*, vol. 49, no. 8, pp. 1292–1302, 2004.
- [15] E. Biyik and M. Arcak, “Gradient climbing in formation via extremum seeking and passivity-based coordination rules.,” *ASIAN JOURNAL OF CONTROL*, vol. 10, no. 2, pp. 201–211.
- [16] N. A. Atanasov, J. Le Ny, and G. J. Pappas, “Distributed algorithms for stochastic source seeking with mobile robot networks,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, 2015.
- [17] L. Brinon-Arranz, A. Renzaglia, and L. Schenato, “Multirobot Symmetric Formations for Gradient and Hessian Estimation With Application to Source Seeking,” *IEEE Transactions on Robotics*, pp. 1–8, 2019.
- [18] V. Gazi, F. Fidan, L. Marques, and R. Ordonez, “Robot swarms: dynamics and control,” in *Mobile Robots for Dynamic Environments* (E. Faruk Kececi and M. Caccarelli, eds.), ch. 4, pp. 79–107, New York: ASME, 2015.

- [19] S. Li, R. Kong, and Y. Guo, “Cooperative distributed source seeking by multiple robots: Algorithms and experiments,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1810–1820, 2014.
- [20] S. Srinivasan, K. Ramamritham, and P. Kulkarni, “ACE in the Hole: Adaptive Contour Estimation Using Collaborating Mobile Sensors,” in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 147–158, Apr 2008.
- [21] S. Al-abri and F. Zhang, “A Distributed Level Curve Tracking Control Law for Multi-Agent Systems,” *2018 IEEE Conference on Decision and Control (CDC)*, no. Cdc, pp. 2575–2580, 2018.
- [22] J.-w. Wang, Y. Guo, M. Fahad, and B. Bingham, “Dynamic Plume Tracking by Cooperative Robots,” *IEEE/ASME Transactions on Mechatronics*, vol. PP, p. 1, 2019.
- [23] A. Joshi, T. Ashley, Y. R. Huang, and A. L. Bertozzi, “Experimental validation of cooperative environmental boundary tracking with on-board sensors,” in *American Control Conference*, pp. 2630–2635, Jun 2009.
- [24] J. Clark and R. Fierro, “Cooperative hybrid control of robotic sensors for perimeter detection and tracking,” in *American Control Conference*, pp. 3500–3505 vol. 5, Jun 2005.
- [25] T. Adamek, C. Kitts, and I. Mas, “Gradient-Based Cluster Space Navigation for Autonomous Surface Vessels,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 1, pp. 506–518, 2015.
- [26] I. Mas and C. A. Kitts, “Dynamic Control of Mobile Multirobot Systems: The Cluster Space Formulation,” *IEEE Access*, vol. 2, pp. 558–570, 2014.
- [27] M. A. Neumann and C. A. Kitts, “A hybrid multirobot control architecture for object transport,” *IEEE/ASME Transactions on Mechatronics*, vol. 21, pp. 2983–2988, Dec 2016.
- [28] J. Acain, C. Kitts, T. Adamek, K. Ebadi, and M. Rasay, “A Multi-Robot Testbed for Adaptive Sampling Experimentation via Radio Frequency Fields,” *ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications*, vol. 9, pp. 1–7, Aug 2015.

- [29] P. Mahacek, “Design and Cluster Space Control of Two Autonomous Surface Vessels,” Master’s thesis, Santa Clara University, dec 2009.
- [30] P. Mahacek, I. Mas, O. Petrovic, J. Acain, and C. Kitts, “Cluster Space Control of Autonomous Surface Vessels,” *Marine Technology Society Journal*, vol. 43, no. 1, pp. 13–20, 2009.
- [31] C. A. Kitts and I. Mas, “Cluster Space Specification and Control of Mobile Multi-robot Systems,” *IEEE/ASME Transactions on Mechatronics*, vol. 14, pp. 207–218, Apr 2009.
- [32] I. Mas, S. Li, J. Acain, and C. Kitts, “Entrapment/Escorting and Patrolling Missions in Multi-Robot Cluster Space Control,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5855–5861, Oct 2009.
- [33] J. Shepherd, *A Framework for Collaborative Multi-Task, Multi-Robot Missions*. Doctoral thesis, Santa Clara University, Santa Clara, California, 2016.
- [34] I. Mas and C. Kitts, “Obstacle Avoidance Policies for Cluster Space Control of Nonholonomic Multirobot Systems,” *IEEE/ASME Transactions on Mechatronics*, vol. 17, pp. 1068–1079, Dec 2012.
- [35] D. Liberzon, *Switching in systems and control*. Springer Science & Business Media, 2003.
- [36] J. T. Shepard and C. A. Kitts, “A multirobot control architecture for collaborative missions comprised of tightly coupled, interconnected tasks,” *IEEE Systems Journal*, vol. 12, pp. 1435–1446, June 2018.
- [37] S. Tomer *et al.*, “A low-cost indoor testbed for multirobot adaptive navigation research,” in *IEEE Aerospace Conference*, pp. 1–12, March 2018.
- [38] X. Kang and W. Li, “Moth-inspired plume tracing via multiple autonomous vehicles under formation control,” *Adaptive Behavior*, vol. 20, pp. 131–142, Apr 2012.
- [39] M. B. Framiñan, A. Valle-Levinson, H. H. Sepúlveda, and O. B. Brown, “Tidal variations of flow convergence, shear, and stratification at the Rio de la Plata estuary turbidity front,” *Journal of Geophysical Research: Oceans*, vol. 113, no. 8, pp. 1–17, 2008.

- [40] J. E. Cloern, A. D. Jassby, T. S. Schraga, E. Nejad, and C. Martin, “Ecosystem variability along the estuarine salinity gradient: Examples from long-term study of San Francisco Bay,” *Limnology and Oceanography*, vol. 62, pp. S272–S291, 2017.
- [41] C. A. Bost, C. Cotté, F. Bailleul, Y. Cherel, J. B. Charrassin, C. Guinet, D. G. Ainley, and H. Weimerskirch, “The importance of oceanographic fronts to marine birds and mammals of the southern oceans,” *Journal of Marine Systems*, vol. 78, no. 3, pp. 363–376, 2009.
- [42] E. D’asaro, C. Lee, L. Rainville, L. Thomas, and R. Harcourt, “Enhanced Turbulence and Energy Dissipation at Ocean Fronts,” *Science*, vol. 332, 2011.
- [43] A. Branch, M. M. Flexas, B. Claus, A. F. Thompson, Y. Zhang, E. B. Clark, S. Chien, D. M. Fratantoni, J. C. Kinsey, B. Hobson, B. Kieft, and F. P. Chavez, “Front delineation and tracking with multiple underwater vehicles,” *Journal of Field Robotics*, no. January, 2019.
- [44] Y. Zhang, C. Rueda, B. Kieft, J. Ryan, C. Wahl, T. C. O’Reilly, T. Maughan, and F. P. Chavez, “Autonomous tracking of an oceanic thermal front by a Wave Glider,” *Journal of Field Robotics*, 2019.
- [45] Y. Zhang, J. G. Bellingham, J. P. Ryan, B. Kieft, and M. J. Stanway, “Autonomous Four-Dimensional Mapping and Tracking of a Coastal Upwelling Front by an Autonomous Underwater Vehicle,” *Journal of Field Robotics*, vol. 33, no. 1, pp. 67–81, 2016.
- [46] Garmin International, Inc, Olathe, Kansas, *GPS 18x Technical Specifications*, 10 2011.
- [47] P. Gainsford, *Stainless Steel Temperature Probe*. Vernier Software Technology, Beaverton, Oregon, 2 2016.
- [48] S. Karumanchi and K. Iagnemma, “Reactive control in environments with hard and soft hazards,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2863–2868, Oct 2012.
- [49] M. P. Fanti, S. Mininel, M. Nolich, G. Stecco, W. Ukovich, M. Bernabo, and G. Serafino, “Flight path optimization for minimizing emissions and avoiding weather hazard,” in *2014 American Control Conference*, pp. 4567–4572, June 2014.

- [50] V. Kashparov, S. Levchuk, M. Zhurba, V. Protsak, Y. Khomutinin, N. Beresford, and J. Chaplow, “Spatial datasets of radionuclide contamination in the ukrainian chernobyl exclusion zone,” 2017.
- [51] H. Lin and H. J. Tzeng, “Search strategy of a mobile robot for radiation sources in an unknown environment,” in *2014 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, pp. 56–60, June 2014.
- [52] M. Purkait, S. C. Jena, T. K. Bhaumik, K. Datta, B. Sarkar, C. Datta, D. Sarkar, R. Ravishankar, S. K. Mishra, T. Bandyopadhyay, S. Sharma, V. Agashe, and P. K. Pal, “Online radiation mapping of k-130 room temperature cyclotron using mobile robot,” in *2011 2nd International Conference on Computer and Communication Technology (ICCT-2011)*, pp. 104–107, Sep. 2011.
- [53] D. Kim, H. Woo, Y. Ji, Y. Tamura, A. Yamashita, and H. Asama, “3d radiation imaging using mobile robot equipped with radiation detector,” in *2017 IEEE/SICE International Symposium on System Integration (SII)*, pp. 444–449, Dec 2017.
- [54] K. Hormann and A. Agathos, “The point in polygon problem for arbitrary polygons,” *Comput. Geom. Theory Appl.*, vol. 20, no. 3, pp. 131–144, 2001.



---

# Appendix A: Supporting Information for Ridge/Trench/Saddle Navigation

## Scalar Field Equations for Ridge/Trench/Saddle Testing

The general equation for the scalar fields used is:

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

where  $h$  is the height of the scalar field at the origin,  $y_s$  is the slope in the  $y$  direction,  $y_{nl}$  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  $\pm 2$ . The  $z$  values were used to create greyscale ranging from 0 (black) to 1 (white). In order to reduce

Table 1: 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 2: 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

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 greyscale colormap.

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

## Inverse Jacobian

The cluster pose vector,  $\vec{C}$ , is given by Equation 1. The inverse Jacobian transform matrix for the six robot cluster used in front navigation is given below in Equation 2, which when multiplied with  $\dot{\vec{C}}$  provides the robot space velocities. Note that  $s_{ic}$  represents  $\sin(\beta_i + \theta_c)$  and  $c_{ic}$  represents  $\cos(\beta_i + \theta_c)$ . Similarly  $s_c$  and  $c_c$  are equivalent to  $\sin(\theta_c)$  and  $\cos(\theta_c)$  respectively.

$$\vec{C} = \begin{bmatrix} x_c \\ y_c \\ \theta_c \\ \phi_1 \\ d_2 \\ \phi_2 \\ d_3 \\ \phi_3 \\ \beta_3 \\ d_4 \\ \phi_4 \\ \beta_4 \\ d_5 \\ \phi_5 \\ \beta_5 \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -d_2c_c & 0 & -s_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -d_2s_c & 0 & c_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & d_3c_{3c} & 0 & 0 & 0 & s_{3c} & 0 & d_3c_{3c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & d_3s_{3c} & 0 & 0 & 0 & -c_{3c} & 0 & d_3s_{3c} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -d_4s_{4c} - d_2c_c & 0 & -s_c & 0 & 0 & 0 & 0 & c_{4c} & 0 & -d_4 * s_{4c} & 0 & 0 \\ 0 & 1 & d_4c_{4c} - d_2s_c & 0 & c_c & 0 & 0 & 0 & 0 & s_{4c} & 0 & d_4c_{4c} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & d_3c_{3c} - d_5s_{5c} & 0 & 0 & 0 & s_{3c} & 0 & d_3c_{3c} & 0 & 0 & 0 & c_{5c} & 0 & -d_5s_{5c} \\ 0 & 1 & d_5c_{5c} + d_3s_{3c} & 0 & 0 & 0 & -c_{3c} & 0 & d_3s_{3c} & 0 & 0 & 0 & s_{5c} & 0 & d_5c_{5c} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (2)$$

---

# Appendix B: Supporting Information for Front Navigation

## Formation Performance

Table 3: Cluster parameter RMS errors for the front tests presented in Figs. 4.9, 4.14, and 4.17.

	Straight	Sinusoidal	Changing Width
$d_1$ (m)	0.0044	0.0060	0.0073
$d_2$ (m)	0.0062	0.0075	0.0063
$d_5$ (m)	0.0142	0.0099	0.0855
$d_6$ (m)	0.0108	0.0078	0.0356
$\beta_1$ (rad)	0.0269	0.0298	0.1039
$\beta_2$ (rad)	0.0431	0.0541	0.0731
$\beta_5$ (rad)	0.0280	0.0258	0.1484
$\beta_6$ (rad)	0.0233	0.0204	0.2435

## Inverse Jacobian

The cluster pose vector,  $\vec{C}$ , is given by Equation 3. The inverse Jacobian transform matrix for the six robot cluster used in front navigation is given below in Equation 4, which when multiplied with  $\dot{\vec{C}}$  provides the robot space velocities. Note that  $s_{ic}$  represents  $\sin(\theta_i + \theta_c)$  and  $c_{ic}$  represents  $\cos(\theta_i + \theta_c)$ . Similarly  $s_c$  and  $c_c$  are equivalent to  $\sin(\theta_c)$  and  $\cos(\theta_c)$  respectively.

$$\vec{C} = \begin{bmatrix} x_c \\ y_c \\ \theta_c \\ d \\ d_1 \\ \theta_1 \\ d_2 \\ \theta_2 \\ d_3 \\ \theta_3 \\ d_4 \\ \theta_4 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \end{bmatrix} \quad (3)$$

