Feature Relative Navigation for Autonomous Underwater Vehicles

by

Andrew Arnold Bennett

B.S., Mechanical Engineering (1985)
Massachusetts Institute of Technology

M.S., Mechanical Engineering (1986)
Stanford University

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Ocean Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1997

Author .....
Department of Ocean Engineering
July 21, 1997

Certified by .....
John J. Leonard
Assistant Professor of Ocean Engineering
Thesis Supervisor

Accepted by .....
J. Kim Vandiver
Chairman, Committee on Graduate Students

Feature Relative Navigation for Autonomous Underwater Vehicles

by

Andrew Arnold Bennett

Submitted to the Department of Ocean Engineering
on August 12, 1997, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Ocean Engineering

Abstract

This thesis presents a technique for adaptively mapping features in the ocean using an autonomous underwater vehicle (AUV). An adaptive behavior was developed in response to the challenge of locating and mapping an unknown number of target features without the aid of *a priori* maps. The design is an extension of the concept of state configured layered control, a modified form of behavior-based intelligent control of AUVs. The new adaptive feature mapping behavior incorporates planning and mapping capabilities which allow the vehicle to alter its trajectory on-line in response to sensor data. New waypoints are selected by evaluating the expected utility of visiting a given location (is it close to previously mapped portions of a feature while not having been visited before?) balanced against the expected cost (weighted distance and heading change) of visiting a particular cell.

The technique has been developed based on the assumptions of a point sensor attached to a non-holonomic, dynamically controlled survey-class AUV such as the *Odyssey II*. Testing has been conducted in a simulation of the Charles River basin constructed from actual bathymetric data and using trenches as target features. Performance metrics are developed and an analysis of the efficiency and robustness of the technique over a variety of system parameters and environmental conditions is examined. Extensions into the realms of remote sensing systems (e.g. scanning beam sonars) and concurrent mapping and localization are also discussed. The technique provides a foundation to investigate challenging new missions for AUVs, such as tracking dynamic features (e.g. mixing fronts), ground-truthing of satellite data, and autonomous navigation in natural terrain.

Thesis Supervisor: John J. Leonard
Title: Assistant Professor of Ocean Engineering

Acknowledgements

First and foremost I would like to thank Prof. Chryssostomos Chryssostomidis for bringing me into the world of ocean research. Without him to help me I would not be here today.

I would also like to thank Prof. John Leonard for keeping me on track, helping me to pull it all together and especially for encouraging me to continuously reexamine my work. Because he continually questioned, I continually improved.

To my thesis committee members, Prof. Henrik Schmidt and Prof. Michael Triantafyllou, a special thank you for your guidance, advice, suggestions and help.

Jim Bellingham and the MIT Autonomous Underwater Vehicles laboratory deserve an extra special thanks for getting me started and showing me what operating an autonomous vehicle is *really* like. Data for the simulation environment came about thanks to Jax and Molly's data collection, Don Atwood's project management skills and Seamus Tuohy's skills at modeling and simulation. Many thanks to all of you.

There are, of course, may others to whom I owe my thanks. No matter how many I list, there will always be more, but here's a partial one anyway: Thanks to Chris Smith for the discussions, Tim Downes, Rere Quinn and Judi Sheytanian for being there when I got desperate, Tom Consi, Jim Bales, Brad Moran, Tom Vaneck and all the gang at Sea Grant for listening to my ideas, critiquing my papers and presentations and just generally helping me out, Michel Perrier and Jerome Vaganay for showing me that research is not only hard work, but great fun, Claudia Rodriguez for the Cuban coffee and Diane DiMassa for the ice cream.

I especially want to thank my mother and father, who just knew I would go far, thereby giving me the motivation to attempt this, and most especially my wife Joice "The Elf", whose patience exceeds that of mere mortals (we're still married after all of this!).

Finally, support for this work was obtained in part from the NASA Earth System Science Fellowship Program (NASA reference number 3843-GC94-0231), whose assistance is gratefully acknowledged (how often do you get the chance to borrow a

U-2?).

And to anyone who I forgot to name (and you are legion, I know), my sincerest apologies and most heartfelt thanks.

Dedication

To my father, whose advice on life consisted of two rules:

1. Don't do anything stupid.

2. Marry your best friend.

And therefore to my best friend Joice.

Thank you both for believing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

The goal motivating this thesis is to provide tools to enable one or more autonomous underwater vehicles (AUVs) to navigate over missions of long duration and large extent, and to obtain detailed information about specific features of interest in the ocean environment. The three key questions for this investigation are the following:

- Can an AUV efficiently locate and sample a collection of features in an unknown environment?
- Can an AUV interpret the data on-line and determine its trajectory to efficiently obtain data from a specific feature of interest?
- Can an AUV navigate over long distances using natural terrain features as navigation references?

Two of the basic tools of field research are population surveys and statistical sampling. These methods are used to determine the distribution and characteristics of some feature in a region and/or to track the change in a population within that region over time. The accepted approach of data collection involves detailed blanket surveys of the area and post-processing of the data to determine what interesting features were found and what their distribution was at the time of the survey. The problem with this approach is that there is generally as much data recorded about uninteresting areas as there is about the actual items of interest. A better approach

is a vehicle which could locate and sample such a population in detail, spending more of its time focused on interesting areas (i.e. areas containing pertinent data) and less on those areas deemed to be uninteresting.

While blanket surveys are somewhat useful for sampling static or slowly evolving phenomena, they are especially poor when applied to rapid dynamic phenomenona. Thermal and salinity mixing zones, plankton blooms, and post-storm runoff are all examples of rapidly evolving features. Current techniques used to study these types of phenomena involve a mixture of field stations, remote sensing, and complex modeling in an attempt to predict the evolution of such features over time. Manned and unmanned vehicles are then dispatched to the area in an attempt to gather more information before the phenomenon dissipates. In some cases the feature may be so short-lived that the window of opportunity to gather data closes soon after measurements have begun. In others, the phenomenon is of such complex three-dimensional structure that there is no good way to predict where to gather data without a detailed predictive model. Currently, the only way of knowing if useful data was successfully gathered is if the information obtained results in an improved predictive model, which in turn helps direct future attempts. It would be extremely useful if a vehicle were capable of making on-the-spot trajectory changes to maximize the number of samples taken in and around the feature and/or to stay with the feature as it moves and evolves with time.

In natural terrain where artificial navigation aids are unavailable, identifiable environmental features can be used as position references. Replacing artificial navigation beacons with natural features reduces the labor and equipment associated with the deployment and retrieval of navigation markers in field operations and can greatly extend the ease with which missions can be performed. Furthermore, even when an external artificial navigation system is available, locating, identifying, and tracking these natural features may still be necessary to guide the AUV mission and to maximize the time spent gathering valuable data.

All of these situations require the competence of *feature relative navigation* (FRN). This thesis develops techniques by which an AUV can adaptively map natural

terrain features by reacting to the presence and/or absence of such features.

1.1 Motivators for feature relative navigation

Three scenarios are presented here as motivators for the contributions made by this thesis. The first application, adaptive mapping of a region, is based on a present day real world industrial need. The second application, adaptive mapping of a dynamic feature, is illustrated by a well-studied scientific application but is intended to represent a broad class of similar scientific and military applications. The third scenario is applying the concept of concurrent mapping and localization in the field.

1.1.1 Adaptive mapping of a region

Recently oil companies have expressed interest in drilling for oil in the Sea of Okhotsk off of the eastern coast of Russia [95]. Although the sea is prone to thick seasonal ice cover, the technology now exists to install bottom-mounted well heads, which means that expensive ice-proof off-shore production platforms are no longer necessary. However, the Sea of Okhotsk is not only prone to winter ice but also portions of it are relatively shallow (see Fig. 1-1). Consequently, ice keels carve trenches into the seabed each winter as the pack ice moves about [99]. These ice keels could easily cut oil pipelines laid on the seabed, resulting in both a huge financial loss and an ecological disaster.

Since the ocean circulation patterns of the Sea of Okhotsk are not known in detail (among other factors) [57], the distribution of ice keel trenches cannot be predicted. Conventional mapping technology such as towed vehicles, ship-mounted sensors, and ROVs cannot be employed due to the ice cover and the extent of the area to be surveyed. One solution to this problem is an underwater vehicle which is capable of extended operation under the ice.

An AUV equipped with a feature relative navigation capability is ideally suited for such a mission. The vehicle should be capable of locating and mapping the extent, depth, and distribution of ice keel trenches so that researchers can develop an under-

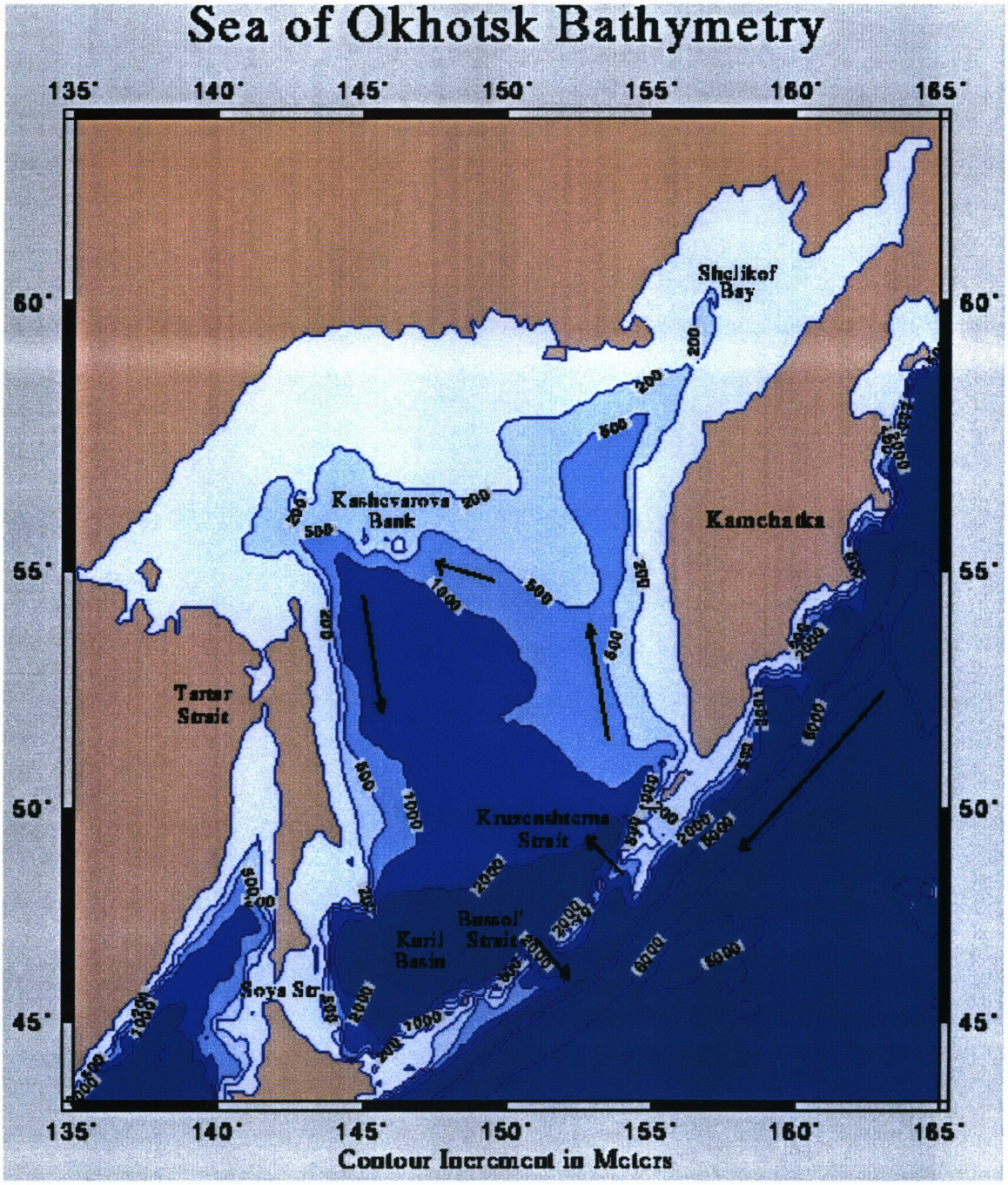


Figure 1-1: Bathymetric map of the Sea of Okhotsk (from TOPEX[57]). In this figure, Japan is to the south, China and Russia to the west and the Kamchatka Penninsula to the east. The sea is prone to unusually large amounts of pack-ice, extending south to the Japanese island of Hokkaido, where it persists for up to three months out of every year.

standing of how the ice keels move and affect the seabed over time – thereby allowing proper placement of oil pipelines. This requires repeated under-ice missions to locate and identify all trenches in a region. Because of the nature of the problem, there is no *a priori* map for the vehicle to work with and only the most basic environmental information (e.g. average depth, overall slope, etc.).

1.1.2 Adaptive mapping of a dynamic feature

In the summer of 1996, the Haro Strait PRIMER experiment was conducted [43] between Seattle and Vancouver just south of Stuart Island. The purpose of this experiment was to construct a three-dimensional "snapshot" of the ocean at the site of a mixing zone between salt and fresh water. To this end, a variety of buoys and vehicles were deployed, including a surface ship and multiple AUVs (Figure 1-2).

The AUVs were deployed in the approximate area where the mixing was taking place. At each deployment the vehicle was given a set of waypoints which were expected to bisect the mixing activity. This was augmented during the mission by a simple supervisory control scheme utilizing one channel of the acoustic navigation system; thus allowing the user to alter the vehicle's trajectory in response to the current observed conditions. The mixing zone was extremely dynamic (currents exceeding 2 m/s were routinely measured) and could easily be seen at the surface. While observation of the underlying structure was the focus of the experiment, the front was recorded in an aerial photograph taken by an ER-2 aircraft at 60,000 ft (Figure 1-3).

This scenario is an example of the potential benefits of adaptive feature mapping. The front was characterized by a change in salinity and temperature as well as by current shear. Both conductivity and temperature sensors are readily available for use on AUVs, and a current shear sensor is currently employed on an unmanned underwater vehicle (UUV) used by the Naval Undersea Warfare Center (NUWC) [68]. Both the Haro Strait experiment and experiments conducted by NUWC have shown that given such sensors an AUV is capable of sensing the front as the vehicle passes through it. The next step is then to have the vehicle autonomously adapt its path in response to the presence of the front, thereby focusing its time and available energy

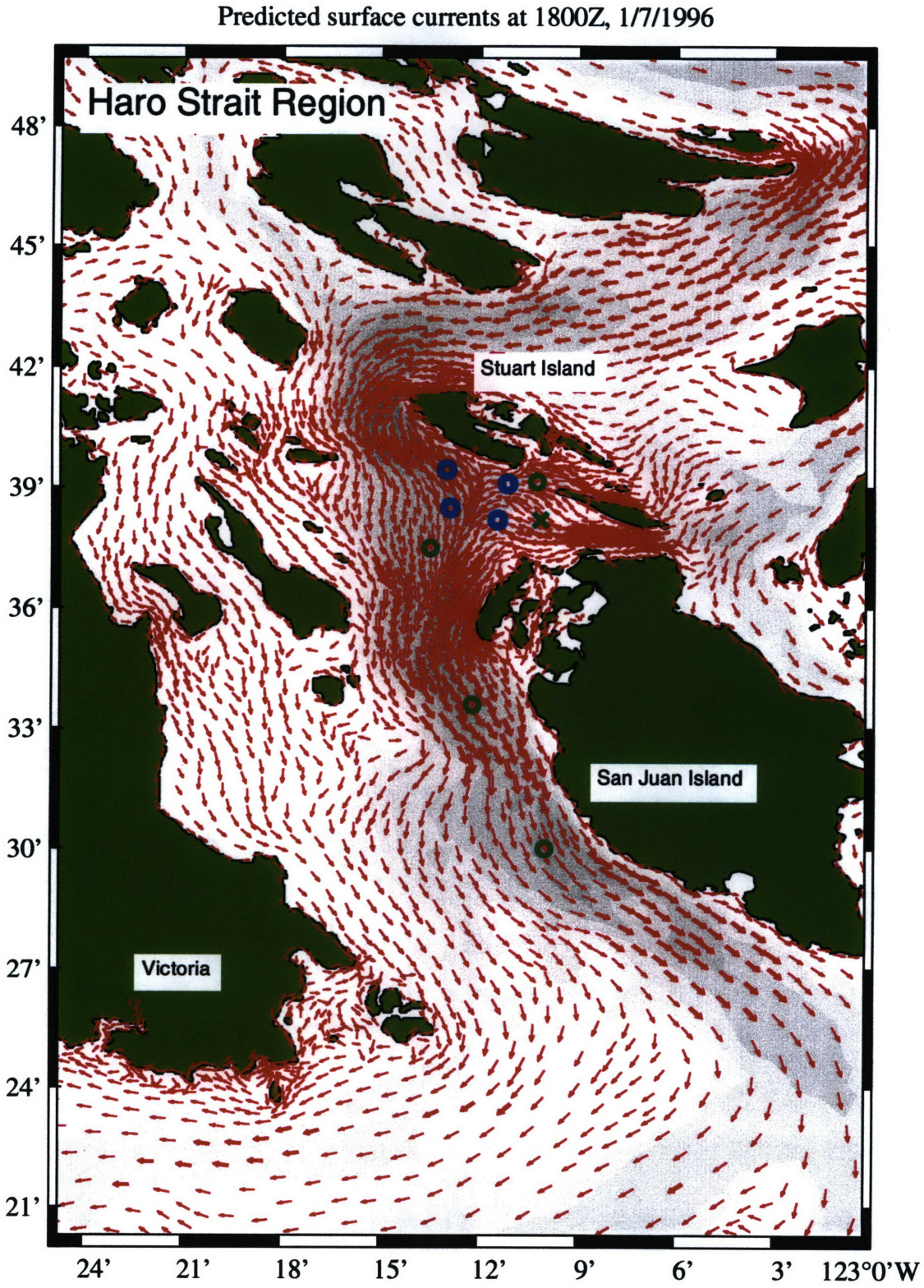


Figure 1-2: Current map from the Haro Strait PRIMER experiment, Summer 1996. From Schmidt [43].

Figure 1-3: Aerial Infrared image of the Haro Strait mixing zone taken from 60,000 ft. From Schmidt [43]. Note the churning in the center of the image, where the currents meet just south of Stuart Island.

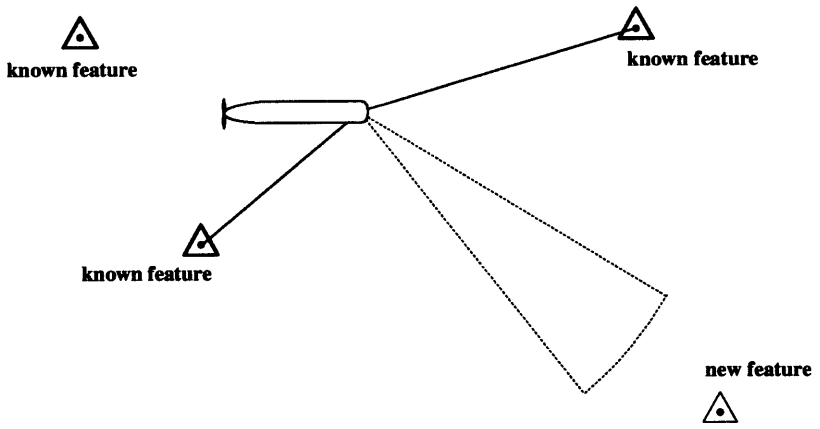


Figure 1-4: Multi-sensor, multi-target CM&L. The vehicle tracks one or more known features while searching for new ones.

on the phenomenon of interest.

1.1.3 Concurrent mapping and localization

The ultimate goal of intelligent navigation is *concurrent mapping and localization* (CM&L). Stated briefly, CM&L is the process of building a map of an environment and simultaneously using that map to estimate the vehicle's location.

CM&L is a natural method of navigation for humans and animals. We use CM&L whenever we go for a walk or drive a car – we look around and remember features (natural or artificial) such as hills, valleys, trees, buildings, etc. and use them as navigational aids to monitor our progress and to determine if or when we pass by again.

Imagine an AUV which senses naturally occurring features in its environment and remembers them, much the same way that a human would. As the vehicle ventures further into an unknown environment, it continuously checks its on-board navigation system (inertial navigation system, dead reckoning, etc.) while at the same time monitoring the location and descriptions of environmental features. With a sufficient number and type of sensors (e.g. an electronically steered sonar), the vehicle can update its map continuously by tracking the location of multiple features and their relative locations (see Figure 1-4). Alternatively, with one sensor the vehicle can venture ahead until it is no longer confident of its position and then return to

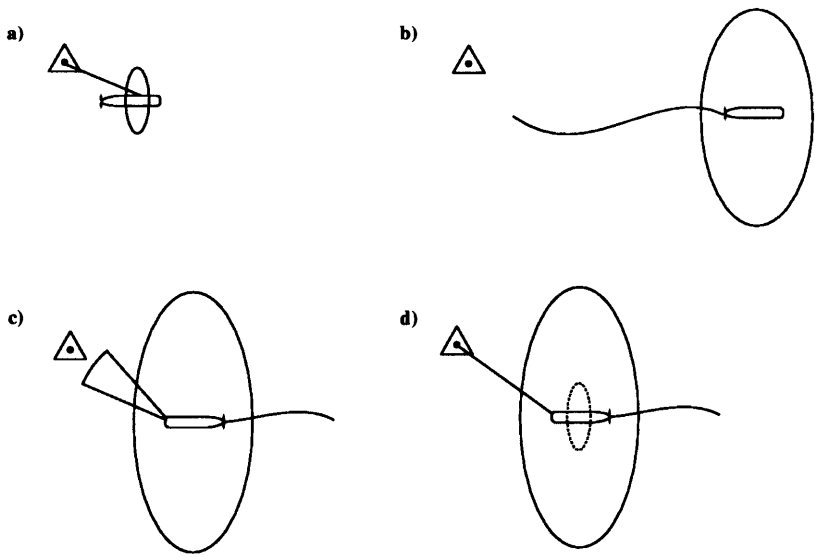


Figure 1-5: Single-sensor CM&L. a) Vehicle locates a feature as a navigational reference. b) AUV moves away until its positional error exceeds threshold. c) Vehicle returns to vicinity of known feature and attempts to relocate it. d) Vehicle reduces position error by relocating itself relative to a known feature.

a previously mapped feature to correct for navigational error (Figure 1-5). This *relocation* ability allows the AUV to place a bound on the traditionally unbounded error growth associated with dead reckoning and INS systems.

1.2 Key issues for AUV research

In recent years, autonomous underwater vehicles have progressed from the status of experimental equipment to that of useful oceanographic instruments. However, the full realization of the capabilities of AUVs will require advances in a number of critical areas:

- Power - AUV missions are extremely power-limited. Improved power storage would greatly increase vehicle capabilities and duration.
- Communications - long range AUV/operator communications would allow the human to assist in object recognition and in solving unanticipated problems.
- Sensors - lower power and more accurate sensors would allow the vehicle to carry a larger and more detailed sensor suite. This would increase the total

amount of data returned per mission.

- Information Processing - the limits of on-board processors and information processing algorithms determine the level of detail of on-board maps, internal models, and the amount of data assimilation that can be accomplished by the vehicle. Improvements in both hardware and software would allow for more thorough and detailed information processing, which in turn would lead to more detailed maps and models and therefore a more capable vehicle.

- Intelligent Control - improved sensor management would help the vehicle to gather increasingly valuable information using the power available. Improved intelligent control would also increase the vehicle's ability to handle unforeseen circumstances, as well as increase survivability.

- Navigation - better navigation capabilities would improve the quality of data returned, and would allow the vehicle to extend mapping/sensing missions into unknown or poorly known regions.

This thesis addresses the last two topics: intelligent control and navigation. The questions we wish to answer are: Given an AUV with certain maneuvering capabilities and a fixed energy supply, how do we insure that it will return with the data it was sent to gather? How do we know where it got the data? What can be done to insure that the AUV gathers the data as efficiently as possible? How can we insure that the vehicle spent as much time as possible gathering useful data? These questions are among those at the heart of autonomous vehicle research.

Current navigation systems depend on the deployment of external navigation arrays. These beacons are costly and difficult to position in deep water. Ideally, an AUV should be able to navigate without such an array, relying instead on recognition of environmental features. Also, once deployed the vehicle should be able to exploit the advantages of such an intelligent controller to quickly locate and identify whatever it was sent to examine, spending as much time as possible gathering useful data. In addition to reducing the cost and difficulty of conventional deployment, the

vehicle could be used for missions not ordinarily within the purview of AUVs such as rapid response to a transient event, tracking thermal plumes and eddies in open water, locating and tracking plankton blooms, and sampling post-storm runoff.

1.3 Intelligent control

The desire to track static or dynamic features introduces the issues of intelligent control into the feature relative navigation problem. This can be viewed as one aspect of the problems of sensor management: Given a vehicle of limited duration and particular capabilities, how can we insure the best use of that vehicle? What can it do to maximize the time and effort spent on the phenomenon that we wish to learn about? If the feature we wish to sense is dynamic in nature, how can we insure that the vehicle stays with the feature as it evolves?

The underwater environment is dangerous and unpredictable. An AUV operating in such an environment is obligated to detect and respond to a variety of conditions, both foreseen and not. A fast and adaptable intelligent control system is crucial to the successful operation of an AUV in this situation, and it will also determine limits of the overall ability of the AUV to function when unsupervised. There are two basic approaches to intelligent control in use today: planning based systems and behavior based systems [18]. Planning systems sense the current state of the environment, examine the goals of the system and then devise a set of actions, or a *plan*, to achieve that goal. The *plan* is reexamined, and modified accordingly, as new data becomes available. Behavior based systems consist of several small special-purpose controllers, each of which are designed to respond to different conditions. Which behavior controls the vehicle depends on its relative importance and the current situation. Planning systems are more flexible and thorough, whereas behavior-based systems are computationally simpler and faster.

1.4 AUV navigation

Current AUVs rely on five primary forms of navigation: the long baseline (LBL) array, the ultra-short baseline array (USBL), dead reckoning, the inertial navigation system (INS), and terrain-based navigation. Each form of navigation has relative advantages and disadvantages:

- LBL arrays, while accurate, are cumbersome to set up – particularly in deep ocean environments; their range is limited to a few kilometers [117].

- USBL consists of a target beacon placed in the water column and multiple receivers mounted on the vehicle. Measuring the phase difference between the receivers gives the vehicle a bearing to the beacon [109]. This makes it ideal for homing functions.

- Dead reckoning relies on accurate estimates of the vehicle's velocity and initial position. Effects such as side slip and external currents, if they exist, must be detected and modeled in order to be accounted for. Because this is usually not the case, it often leads to a steadily growing error in estimated position [46].

- Inertial navigation systems are more sophisticated than dead reckoning systems. They rely on accelerometers and gyros to monitor changes in the speed and attitude of the vehicle [60]. While the technology is mature, they are expensive and prone to drift over time and must get periodic updates from external sources. Currently, the size, power requirements and cost of accurate INS systems restrict their applicability in small AUVs.

- Terrain based navigation uses *a priori* maps of the operational region, which it then compares to the sensed environment. A match between the sensed environment and the stored map locates the vehicle in space. An example of this is the TERCOM navigation system used until recently by cruise missiles [51], and geophysical map navigation as presented in Tuohy [111]. The primary drawback to such systems is the expense and difficulty in generating the *a priori* maps.

The method of navigation put forth in this thesis is feature relative navigation (FRN). The purpose of FRN is not necessarily to determine the position of the vehicle in a global coordinate frame (sometimes referred to as the localization problem), but rather to locate the vehicle relative to some feature of interest – with the intent of maximizing the amount of vehicle time and energy collecting information about that feature.

For our purposes, features can be classified into two main groups: contour-based features and area-based features. The type of feature will dictate the type of maneuvers that will be used to survey and map that feature. In this thesis we will examine both types of features, with particular emphasis on contour-based features.

1.5 Assumptions

This thesis examines the case of a single sonar altimeter mounted on a dynamically controlled AUV tracking an isobath in detail. This particular case was chosen for the reasons of versatility, economy, opportunity, and repeatability. Versatility because the techniques developed for the sonar altimeter case are applicable to other point-sensor situations such as magnetic, thermal, or turbidity sensors. Economy and opportunity because all vehicles operating near the seafloor, regardless of type, have at least one sonar altimeter mounted upon them. The approach presented here may therefore be easily employed on any vehicle, not necessarily one which has been purpose-built or modified for the task. The chosen scenario is repeatable because topographic features are stationary in time and, therefore, any changes in vehicle behavior over multiple tests would be the result of changes introduced into the system and not in the environment.

We also assume a non-holonomic vehicle. This work employs a dynamic simulation based on the *Odyssey*-class of dynamically controlled survey AUVs (Figure 2-3). The AUV *Odyssey* is an untethered, dynamically-controlled vehicle with a non-zero turning radius. Dynamically controlled vehicles have a minimum velocity below which fin authority is lost. This minimum requirement restricts the operational range of

vehicle speeds. In contrast, a holonomic vehicle such as the ROV *JASON* can stop, hover, and turn in place. Such capabilities are outside those of survey class AUVs, and so are not considered here.

Finally, the methodology used is based on a simulated ocean environment which in turn is based on real data wherever possible. The simulator employed has been used by the MIT Autonomous Underwater Vehicles Laboratory, and has been validated by several years of field tests [12]. Sensor models are based on the dynamics of the Tritech Model ST500 conical beam sonar transducer with simulated noise modeled after that found in data obtained in tests with the AUV *Odyssey IIb* in the Charles River, located between Cambridge and Boston, Massachusetts. The bathymetry of the test area is derived from sonar altimeter data collected in the Charles River basin during the summer of 1993.

1.6 Contributions

This thesis makes the following contributions:

- **A technique** for adaptive feature mapping employing behaviors which possess the capabilities of mapping and path planning. Unlike conventional reactive behaviors, adaptive behaviors alter their internal state as new information about the environment is received. This in turn allows the behavior to alter its output in response to the present situation.

- **An analysis** of the performance and robustness of the adaptive feature mapper, examining time-on-target, total path length needed to map a feature, amount of feature mapped per unit time, amount of feature mapped per meter traveled, and performance of the adaptive behavior in comparison to a conventional lawnmower-type survey.

- **Extensions** of adaptive behaviors into the realms of adaptive mapping of dynamic features, multiple map-mode representations, incorporation of error estimation, alternative waypoint determination techniques employing pre-compiled

lookup tables, cost functions & decision-theoretic techniques, and navigation applications, including relocation and concurrent mapping & localization (CM&L).

1.7 Structure of this thesis

In this thesis we present the problem of feature relative navigation, focusing specifically on the problems associated with efficiently locating and mapping an unknown number of features located in a given area with no *a priori* map of that area.

In Chapter 2 we review prior research in the three fields of navigation, mapping, and intelligent control.

Chapter 3 examines the specific challenge of locating and mapping bathymetric features (in this case trenches) in the Charles River basin and the extension of the behavior based intelligent control paradigm to include the development of *adaptive behaviors*, which have mapping and planning capabilities within the overall layered control environment.

Chapter 4 develops metrics of robustness and efficiency. Navigational error, environmental disturbances, and parameter sensitivity are systematically examined. Alternative search strategies, including omnidirectional waypoint lookup tables, directional lookup tables, and cost functions are examined under various conditions and compared.

Chapter 5 discusses applications and extensions to FRN, including tracking and mapping dynamic features, multiple-vehicle applications, navigational resets (re-identification of previously mapped features), and the concept of concurrent mapping & localization (CM&L).

Chapter 6 concludes this thesis by summarizing the contributions and making suggestions for future research.

Details of the software implementation are provided in the Appendix.

Chapter 2

Literature Review and Problem Formulation

Feature relative navigation (FRN) lies at the intersection of three different research topics: AUV navigation, map building, and intelligent control (Fig. 2-1). This chapter surveys the literature in each of these three areas and defines feature relative navigation in relation to these research topics. First, we give some background on the use of AUVs as instruments for oceanographic exploraton. Next we examine autonomous vehicle navigation techniques. Both traditional and non-traditional approaches to AUV navigation are discussed, with an emphasis on the assumptions made and sources of error for various navigation methods. Consideration of techniques for map building introduces the important considerations of choice of representation and management of computational complexity. Intelligent control is an extremely broad field; our treatment here describes the basic distinctions between planning and behavior-based approaches, laying the foundation for development of a hybrid system incorporating aspects of both approaches.

2.1 Exploration with AUVs

The noted marine explorer Robert Ballard once said that an unexplored region of the ocean is an area approximately the size of the United States which has had, at

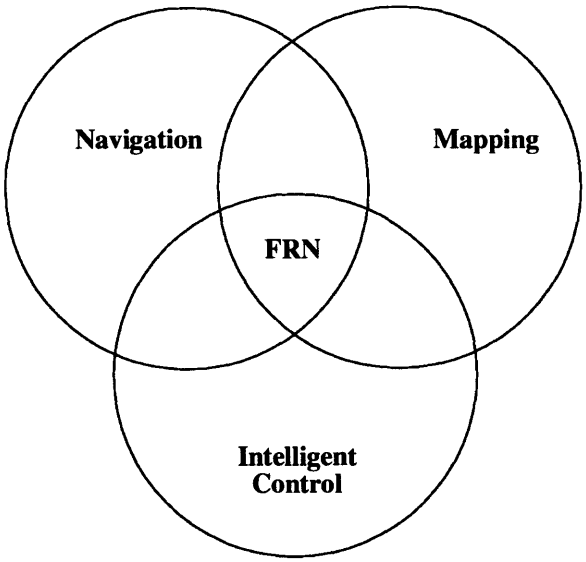


Figure 2-1: Feature relative navigation is the union of navigation, mapping and intelligent control.

most, one sounding taken [2]. An explored area is a region the size of a small state in which a single sounding has been taken. Now imagine trying to describe a whole state based on one altimeter reading taken from an aircraft flying high overhead on a moonless night. Add to that the fact that the resolution will only tell you the altitude to within coarse limits and it quickly becomes apparent that this is not the best way to survey the country in any detail. This is analogous to the task that faces ocean exploration today. Compounding this further is the fact that the ocean is vast and the number of vehicles suitable for deep water exploration is limited.

One solution is to employ remotely operated vehicles (ROVs), which have been used with great success for many years. The cables connecting them to the surface provide a two-way conduit – for sensory information to go up to the researchers and for power to go down to the vehicle, allowing it to stay on station indefinitely. However, the same umbilicus which provides power and communications also limits movement and economy: ROVs are not capable of large horizontal excursions, are slow to deploy and recover, and cannot be operated in rough seas. Each deep water ROV requires a support ship costing upwards of $10,000 per day and several kilometers of expensive cabling.

Another common economical solution is the towed vehicle, or "sled." Towed

vehicles are inexpensive and can explore large swaths of the ocean at a time. Their drawbacks are, as with ROVs, the cost of cabling, deployment/retrieval times, and sensitivity to rough seas. In addition, the drag on the tow body and cable restrict the towing ship to only a few knots. Furthermore, the potential for cable twisting and kinking requires that the tow ship execute large (1-2 NMi or more) turns to prevent this. These restrictions make towed vehicles ill-suited to rapid-response situations and for operation in areas requiring multiple tight passes.

In response to these shortcomings, autonomous underwater vehicles (AUVs) have been developed. An AUV is an unmanned, untethered submersible vehicle capable of operating without human supervision or intervention. Because they are untethered, they are capable of operating at depth and of tight maneuvers regardless of surface conditions.

Historically, AUVs have been large and expensive [91] (see Figure 2-2); an historical review of AUVs can be found in Bellingham [6]. Recently Bellingham has pioneered the application of low cost survey-class vehicles for science applications with emphasis on inexpensive, off-the-shelf components and low-power sensors [12, 13] (Figure 2-3). These vehicles are optimized for long-range mapping and survey operations designed to cover large sections of the ocean floor economically. The *Odyssey*-class vehicles have since been proven in the Arctic [16], deep ocean environments [13], and as part of a real-time tomographic network [100]. Recently, the advent of the Autonomous Oceanographic Sampling Network (AOSN) has increased the deployment possibilities [7, 37].

2.2 AUV navigation

Navigation can be thought of as answering three basic questions: Where am I? Where am I going? How do I get there? Geodetic maps and navigation techniques help to answer the first question while the second and third questions can be thought of as the problems of path planning [4, 63] and obstacle avoidance [20, 21, 108] respectively. Similar to Leonard [66], this thesis is concerned with the first question from a local

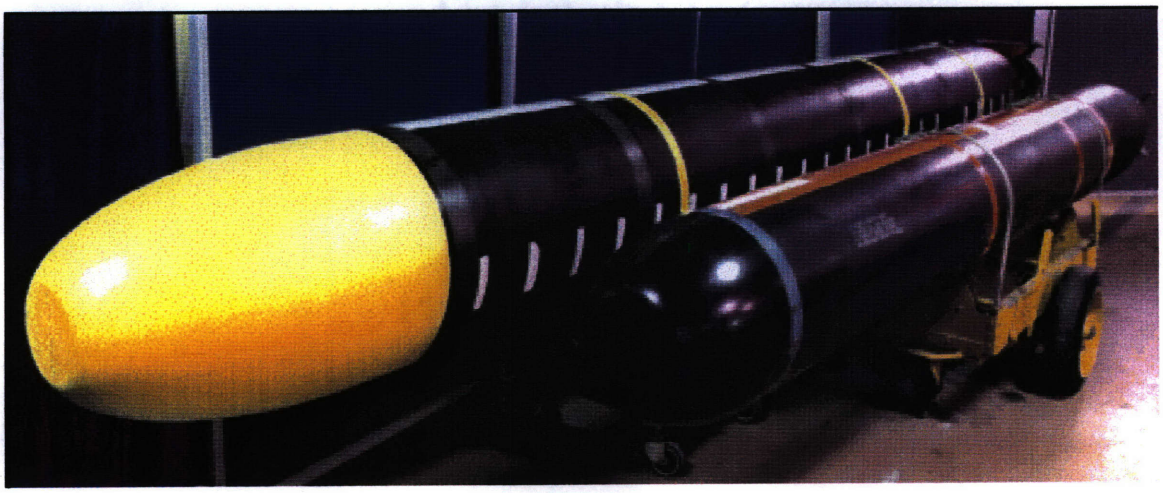


Figure 2-2: The Naval Undersea Warfare Center's unmanned underwater vehicles [67].

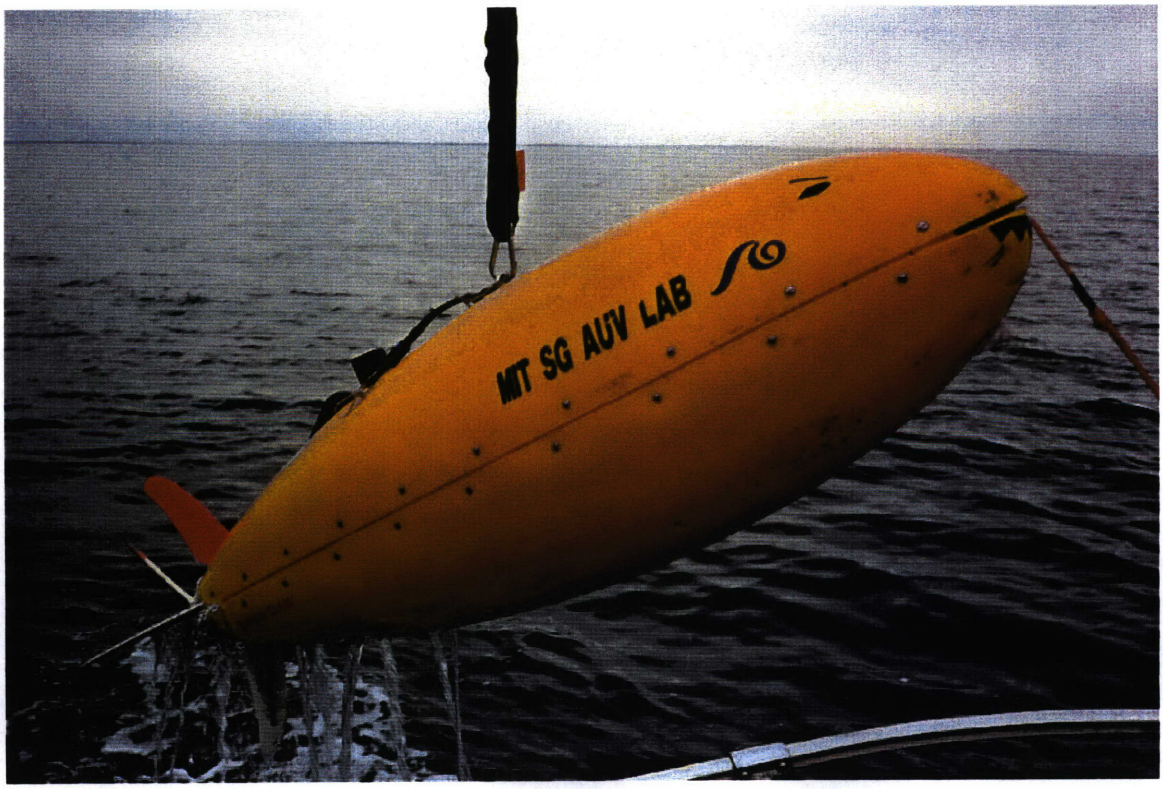


Figure 2-3: MIT Sea Grant's AUV *Odyssey II*.

perspective and with the other two from an information-gathering perspective.

All mobile robots must navigate in order to function successfully. However, the term "navigation" can mean many different things according to the context in which it is used. Three common forms of robot navigation are local navigation relative to an array of known beacons or features, global navigation relative to a geodetic reference frame, and path planning (either to avoid obstacles and/or to reach a goal). We will begin our discussion with the first two: navigation in either a local or a global sense.

The essence of navigation is knowing where you are, either relative to something else or relative to the earth as a whole. Knowing one's position is essential for an AUV to function successfully. This may be for the purposes of map generation [66, 107], navigation in harsh or difficult environments such as thermal vent fields [73] or under ice [8], or in dynamic regions such as frontal mixing zones [100]. To be of value to the researcher, the location of any data gathered must be known either relative to a geodetic (latitude, longitude), to a known feature (the launch point in the ice), or to other aspects of the phenomenon being studied, such as the relative distribution of thermal vents in a vent field.

Geyer *at al* [46] provides us with an overview of AUV navigation options. Six primary navigation methods are: acoustic positioning, dead reckoning, inertial navigation, radio navigation, satellite navigation, and geophysical map matching. Because the ocean is essentially impenetrable to electromagnetic energy (except at very low frequencies), radio-based and satellite navigation systems are only useful for occasional position fixes, and then only if the vehicle is capable of surfacing. For deep water applications, such as automated bathymetric mapping, seafloor photography, or military operations which require stealth, this is not an option [11].

Table 2.2 presents the primary forms of navigation available to AUVs and some associated references. The following sections discuss the state of the art in each of the major areas of navigation, and present some of the primary technologies and important sources of error. The need for external arrays and *a priori* information is discussed where applicable.

Navigation Technique	References	Output(s)	External Reliance	Spatial Availability
Dead Reckoning	[76]			
Magnetic compass		Magnetic heading	Autonomous	Worldwide
Gyrocompass		Geodetic heading	Autonomous	Worldwide
Water log		Water speed wrt/vehicle	Autonomous	Worldwide
Doppler sonar		Water speed wrt/ground or water	Autonomous	Within 100m of bottom
Inertial Navigation	[45], [60]			
Gimballed platform		Lat., long., depth, vertical velocity, pitch, roll heading	Autonomous	Worldwide
Strap down RLG (ring laser gyro)	[69]	Lat., long., depth, vertical velocity, pitch, roll heading	Autonomous	Worldwide
Acoustic Navigation	[82]			
Long baseline	[39], [55], [117]	Position relative to net	Active acoustic	Near transponder net
Short baseline	[109]	Position relative to beacon	Active acoustic	Near beacon
***a priori* Map Matching**	[46], [112]			
Gravity	[80]	Latitude, longitude	Autonomous	Surveyed area(s)
Magnetic	[114]	Latitude, longitude	Autonomous	Surveyed area(s)
Topographic	[49], [51], [53], [54]	Latitude, longitude	Autonomous	Surveyed area(s)

Table 2.1: Comparison of different AUV navigation methods

2.2.1 Dead-reckoning

Dead reckoning is the simplest form of vehicle navigation, ideally requiring very little information from the environment. In dead reckoning, a vehicle simply integrates its sensed or assumed velocity (V_s) along the sensed direction of travel (H_s) to determine how far it has traveled. More practically, the vehicle starts with an assumed (N, E) position, resolves its current velocity into V_N, V_E using the relations

$$V_N = V_s * cos(H_s) \tag{2.1}$$

$$V_E = V_s * sin(H_s), \tag{2.2}$$

integrates these velocities over the time since the last update

$$Pos_N = \int_{t_0}^{t_1} V_N dt + Pos_{Ninit} \tag{2.3}$$

$$Pos_E = \int_{t_0}^{t_1} V_E dt + Pos_{Einit}, \tag{2.4}$$

and then adds the result to the previously determined position. The accuracy of dead reckoning navigation depends upon the accuracy of water speed and heading sensors as well as the magnitude of currents. Any error in these sensors integrates into a position error which will grow in time.

Sources of error

Errors in dead reckoned position are introduced as discrepancies between the sensed heading or velocity and the actual values. The rate of error growth depends on the particular situation and can grow without bound. There are many sources of error which are specific to each sensor, but they all manifest in a few basic forms.

The chief sources of error for heading sensors (e.g. a compass) are mounting error, measurement quantization error, sensor noise errors, and magnetic anomaly errors. Mounting errors are a static bias in the sensed heading brought about by inaccurately attaching the compass to the vehicle frame. These errors can be of any magnitude but can easily be compensated for by pre-mission calibration. Measurement quan-

tization and sensor noise errors (if random noise) tend to average out over the life of a mission and hence do not pose a significant source of vehicle navigation error. Magnetic anomaly errors can come in many forms and magnitudes. The effects of such anomalies depend upon the size and duration of the anomaly. Common examples are nearby concentrations of iron or magnetic materials and on-board/nearby electrical equipment. Disturbances due to on-board magnetic interference or metals can be minimized with adequate shielding and/or careful calibration & modeling of the vehicle before deployment. Gyrocompass errors are typically due to spin axis drift, and can be compensated for with accurate magnetic compass measurements and auto-leveling servos or gimballing.

Water speed sensor errors fall into two categories: those associated with the sensor design itself and those which are inherent to any body-mounted water-relative speed sensor. For mechanical systems, the primary source of error is the friction in the sensor wheel or paddle hub, resulting in a dead-band at low speeds and a slower-than-true measurement of water speed when operating. This results in poor measurements at speeds of less than 0.5 kts. At high speeds the rotational inertia of the wheel can account for errors on the order of 1%-3%, but this is outside of the operating regime of survey-class AUVs. All body-mounted water speed sensors suffer from both current and sideslip insensitivity. Because the sensor is generally mounted parallel to the longitudinal axis of the body, any currents affecting the vehicle as a whole will go unnoticed. Likewise, any sideslip of the vehicle (across body water flow) will also go unnoticed, especially during turning maneuvers.

Acoustic doppler sonars measure vehicle speed by detecting the difference in velocities between the vehicle and some target. While this method works well when a suitable reference is in the operational range of the sonar (generally a few hundred meters at 100-300khz), it can fail in open water situations. If the vehicle is in a body of water subject to a uniform current, and that body of water is larger then the range of the sonar, then the motion of the vehicle relative to the outside world will go unnoticed by the sensor because no outside reference is detected. Also, like all sonar systems which track a target, doppler sonar water speed sensors suffer from dropped

returns. Additional errors can be introduced by misalignment of the sonar relative to the heading sensor, resulting in a miscalculation of along-track and cross-track velocity.

In general, a dead reckoning navigation system can be used to provide relatively accurate navigation in situations with few changes in heading, such as parallel-track (a.k.a. "lawnmower") survey missions and round-trip (out-and-back) missions. In these situations, the cross-track navigational error due to heading error on an outbound leg is offset by the error on the return leg. On the other hand, along-track errors due to water speed measurement biases are generally cumulative over the life of a mission. Note also that errors due to environmental factors such as a current will not be resolved in this fashion and will cause continually increasing errors throughout the mission or until the area with currents has been vacated. This type of error can be avoided by operating near the bottom and with the use of an appropriate acoustic doppler sonar unit.

2.2.2 Inertial navigation

Inertial navigation systems (INSs) provide continuous latitude, longitude, depth, velocity, and orientation information. Their chief advantage is the ability to operate for long periods of time without external navigation or sensor input. They are particularly well suited to applications where external navigational updates are rare or nonexistant. This is a common scenario for an AUV, in particular one which is operating at great depth or in a clandestine operation.

An INS must be aligned before each use. This is done to calibrate the inertial orientation of the INS with respect to the current location and orientation of the vehicle. This must be performed in either a stationary situation or where the velocity is well known since the INS accelerometers will not detect an initial steady velocity. There are a variety of reliable ways to perform this calibration.

INS systems come in two basic forms: gimballed and strap down. In a gimballed system the INS is isolated from the vehicle via a gimballed platform and can be oriented independently from the vehicle. The system is torqued to account for earth

rotation and vehicle motion over the earth in order to keep the INS level with respect to the horizon. Because they maintain a constant horizon, gimballed INSs have a lower computational burden. This made them predominant in early systems with less powerful processors. However, the gimballing system requires a larger mass, has greater power consumption, and a shorter mean time between failures (MTBF) than the more modern strap down INSs [46].

Today the strap down INS is more prevalent, owing to both the power of modern microprocessors as well as the reduced mass and volume of INS systems employing ring laser gyros in place of the older mechanical gyros [69]. A strap down INS is rigidly attached to the vehicle frame. Factors such as earth rate and vehicle rate are modeled into the INS calculations. Current RLG-based INSs installed in AUV systems can achieve accuracies of 0.25 NMi/hr in position and $\pm$ 2 ft/s RMS in velocity [88].

Sources of error

INS error is a function of many things: the type of sensor suite used, the mathematical models employed to account for earth rate, vehicle rate and local gravity field, initial INS alignment, and the trajectory followed by the vehicle. Key sources of error are accelerometer bias, which integrates into both velocity and position error, and gyroscope drift (much more significant in the days before ring laser gyros). Both sources of error can be mitigated with the use of external navigation sensors such as radar altimeters, doppler sonar units or even occasional GPS (Global Positioning System) navigation fixes.

Because errors are so dependent on the design of the individual unit, the preferred method of determining error rates is to consult the manufacturer's specifications. For example, the Litton LN-200 IMU lists heading bias variations of 0.35 degrees per hour (1 σ) with a 100 second correlation time and an accelerometer bias variaton of 50 μg (1 σ) with a 60 second correlation time.

2.2.3 Acoustic navigation

Acoustic navigation systems such as long baseline (LBL) and ultra-short baseline (USBL) [109] navigation systems both employ external transducers or transducer arrays as aids to navigation. In LBL navigation systems, an array of transponders is deployed and surveyed into position. The vehicle sends out an acoustic signal which is then returned by each beacon as it is received. Position is determined by measuring the travel time between the vehicle and each beacon, measuring or assuming the local sound speed profile, and knowing the geometry of the beacon array. With this information the relative distances between the vehicle and each array node can be calculated. It is then a simple task to calculate the position of the vehicle by locating the intersection point of spheres of appropriate radii from the beacons in the array. A variant of this system is *hyperbolic navigation*, in which the vehicle does not actively ping but instead listens to an array of beacons whose geometry is known. Each beacon pings in a specific sequence relative to the others at its specified frequency. By knowing which beacon pings when and the geometry of the array, the vehicle can reconstruct where it must be in space in order to hear the ping sequence as recorded. This system has the advantage of saving the vehicle the power expenditure of active pinging, but is by necessity forced to work at the update rate dictated by the array.

In a two beacon array, the vehicle can determine its location to two possible positions; the solution is obtained by estimating the distance traveled between fixes and eliminating the position which would be impossible to reach in the given time between fixes (when used with a kalman filter, this technique is referred to as "error gating"). Note however that in the event of a two-beacon array, the position error is infinite when the vehicle and beacons are colinearly aligned. To avoid this, arrays typically consist of three or more transponders. For reasons of both geometry and signal strength, the best results are achieved when the vehicle operates within the area bounded by the array.

USBL navigation consists of a single transponder mounted on the seabed (or a ship) and a 2-dimensional receiver array mounted on the vehicle. By measuring the

difference in arrival times of a single sonar ping (i.e. the phase angle) between two hydrophones, the bearing from the vehicle to the beacon can be determined. If the beacon responds to vehicle interrogation, then the time delay (and hence distance, as with an LBL array) can be calculated. Knowing distance and direction to the beacon allows for local navigation [109]. Knowing the latitude/longitude of the beacon allows for geodetic navigation.

Sources of error

Errors in both LBL and USBL arrays come from many sources. The key sources of error can be broken down into two primary categories: timing-based errors and errors in the assumed array geometry. The former errors typically manifest as sound speed assumption (or measurement) errors, measurement noise, and beacon response delay (which is interpreted by the system as being a ping from a more distant beacon). Array geometry errors are due to beacon positioning errors, beacon motion during a mission, and signals from extraneous beacons (see Vaganay *et al.* [117]). The magnitude of position errors resulting from these types of errors vary from meters to hundreds of meters.

Positioning error comes from inadequately or improperly surveying the relative and/or geodetic positions of the array beacons. In the event that only local navigation is desired, then only relative beacon positions are relevant. If the navigation is to be geodetic-referenced, then the beacons must be located globally as well. Self-calibrating beacons simplify the task by reducing the surveying task to only one beacon with the others determining their own positions relative to the first. However, this raises the possibility of relative position errors due to errors in the measurement or estimation of the local sound speed.

Acoustic errors can manifest in several ways. An inaccurate sound speed profile will appear as a distance bias in the calculations. Reflection or multipath errors will appear as additional beacons or greater-than-actual distances. Over longer distances and shallow refraction angles, there is also the risk of shadow zones or "dropped" beacons. If the topography is sufficiently severe, beacons may be occluded by rocks

or other seabed formations.

Measurement noise that is white noise in nature can easily be filtered out. Multipath errors are difficult to detect and filter. Beacon response delay manifests as additional travel time and can cause bias in the estimated distance to the transponder; it too is difficult to filter.

2.2.4 Map-based navigation

The concept behind map-based navigation is to determine the position of the vehicle with respect to an *a priori* map of a spatially varying property using vehicle mounted sensors. Map-based navigation is centuries old and has been successfully applied to land, air, and ocean vehicles. While adaptive contour following does not rely upon *a priori* maps, it is important to understand how such maps are utilized for the purposes of constructing and reading one as the mission proceeds, and so we will use recent developments in map-based navigation as a starting point.

Map-based navigation can be divided into two basic forms: local map navigation, for use in obstacle avoidance or trajectory planning, and geodetic navigation – or the use of globally-referenced maps to locate the vehicle's position in terms of latitude, longitude, and depth. If a local map can be tied back into global coordinates, then it can be thought of as another form of geodetic navigation. We will focus on geodetic navigation here and address the issues of obstacle avoidance and local map navigation in the next section.

All forms of map-based navigation are motivated by the desire to operate at an arbitrary location without the additional expense or problems associated with the installation of artificial beacons. In principle, the process is simple: gather information about the surrounding terrain and match that information to an on-board map or database of terrain information. When the vehicle has a match to the database, then it knows its location on the map. This is analogous to the method which humans use to navigate; we find our way to our destination by locating and identifying landmarks which are familiar to us – either from past experience or via a map which has been constructed for our benefit.

In practice this form of navigation is not so simple. The vehicle is attempting to navigate by matching a set of sensed data $\{\hat{X}\}$ with an *a priori* map or dataset of stored data {X}. Two key problems are the cost and difficulty of generating the *a priori* maps and the computational complexity of searching for a peak in the n-dimensional correlation surface, where n is the number of dimensions in the map or sensor data set. Typically, map making expense is governed by both the type of data being collected and the desired resolution of that data. Determining the map resolution has a direct effect on the size and level of detail of the search needed to locate the vehicle in space. Since the vehicle could be in any of a large number of possible orientations relative to the original dataset, the search must be performed over all possible locations and orientations. This is a potentially large search space, necessitating some simplifications and/or simplifying assumptions in order to make the search more tractable. Typical simplifications are: restricting the types of map data stored (what sensor values, how many different sensors), lowering map resolution, "patchy" maps (maps of key areas only), restricting vehicle orientations (to reduce the correlation problem), and using inertial navigation or dead reckoning systems to limit the valid search area.

Geodetic maps

For the purposes of this thesis, geodetic maps are defined as maps of physical properties of the earth. Geodetic properties which have been used in navigation are: the earth's magnetic field, the gravitational field, and topography or bathymetry. Each is discussed in the following sections.

Magnetic maps

Evidence exists that geomagnetic navigation is employed by birds, fish, and other animals for migration and general navigation [119]. The magnetic flux density of the earth varies according to latitude, the presence of man-made and natural anomalies, and even one's depth in the ocean, increasing from 6 to 30 nanoTeslas per 1 km of depth, depending on location [86]. Additionally, there are small but predictable

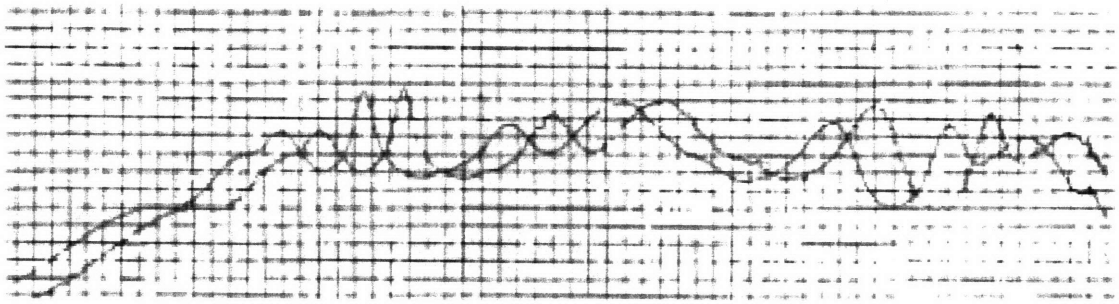


Figure 2-4: Recorded output of two magnetometers with 30 meter along-track separation. In this figure, the horizontal axis is time (which translates to along-track distance) and the vertical axis is magnetic field intensity. Note how the same set of features are visible in both sensor plots. From Tyren [115].

variations in the earth's magnetic flux from day to night, and large arbitrary changes during magnetic storms (which are approaching the height of their 11 year cycle at this time); magnetic maps can be rendered useless for the duration of such storms. Useful magnetic maps, generated by satellites or surface ships, can be employed by underwater vehicles by accounting for the daily field variations and by calculating the effective magnetic field at depth using a Laplace field equation, setting the boundary conditions at the ocean surface [111].

The primary research in magnetic navigation has been in reference to local navigation issues such as using local magnetic flux variations as a ground reference. Tyren [115] experimented with autocorrelation along a ground track by towing two magnetometers with a 30 meter along-track separation (see Figure 2-4). The time displacement between the two curves gives a direct measure of ground speed.

Other researchers have investigated the use of magnetometers for locating magnetic anomalies such as wrecks or mines. These anomalies could potentially be used as "beacons" for magnetic navigation systems [93]. Full magnetic navigation systems similar to the topographic navigation system TERCOM [51] (Terrain Contour Matching) are also currently under development.

Gravometric maps

Research into the nature of the earth's gravitational field has demonstrated that it is far from uniform and indeed possesses a varied topography [38, 120]. These variations are due to a variety of factors, especially the effects of local topography [44] and density inhomogeneities [113]. Variations in the earth's magnetic field on the ocean's surface relative to a regular ellipsoidal model have been measured to be on the order of 30-50 mgal [56][1]. Gravity maps were originally gathered on behalf of the US Navy for the purposes of inertial navigation system calibration [98]. To an INS, the effects of a change in the local gravitational field are indistinguishable from accelerations of the vehicle itself. Gerber [45] proposed the use of a gravity gradiometer as an aid to inertial navigation systems. Jircitano *et al.* extended this idea to the AUV community, performing navigation simulations using a model of the Bell Aerospace Textron Gravity Gradiometer System (GGS) [56] with good preliminary results.

The drawbacks to such a system are the size, expense, and complexity of a gradiometer (see Geyer [46]). Of more practical and immediate concern is the requirement that the gradiometer be mounted on an inertially stabilized and vibrationally isolated platform, making its use on small survey-type AUVs such as the *Odyssey IIb* difficult.

Topographic maps

Topographic maps are one of the most recognizable map forms. They consist of lines of equal altitude (or depth, for bathymetric maps) which divide the terrain into layers, much like a cake. Tightly clustered lines indicate steeper slopes, while looser groupings are flatter regions (see Figure 2-5).

For humans, using topographic maps is one of the most familiar forms of map-based navigation (it has even become a sport: orienteering). When porting to the world of autonomous robots, however, the problem of *representation* becomes key to successful mapmaking and utilization. The issue of representation is critical to any

[1] $1 milligal = 1 mgal = 10^{-5} m/sec^2$

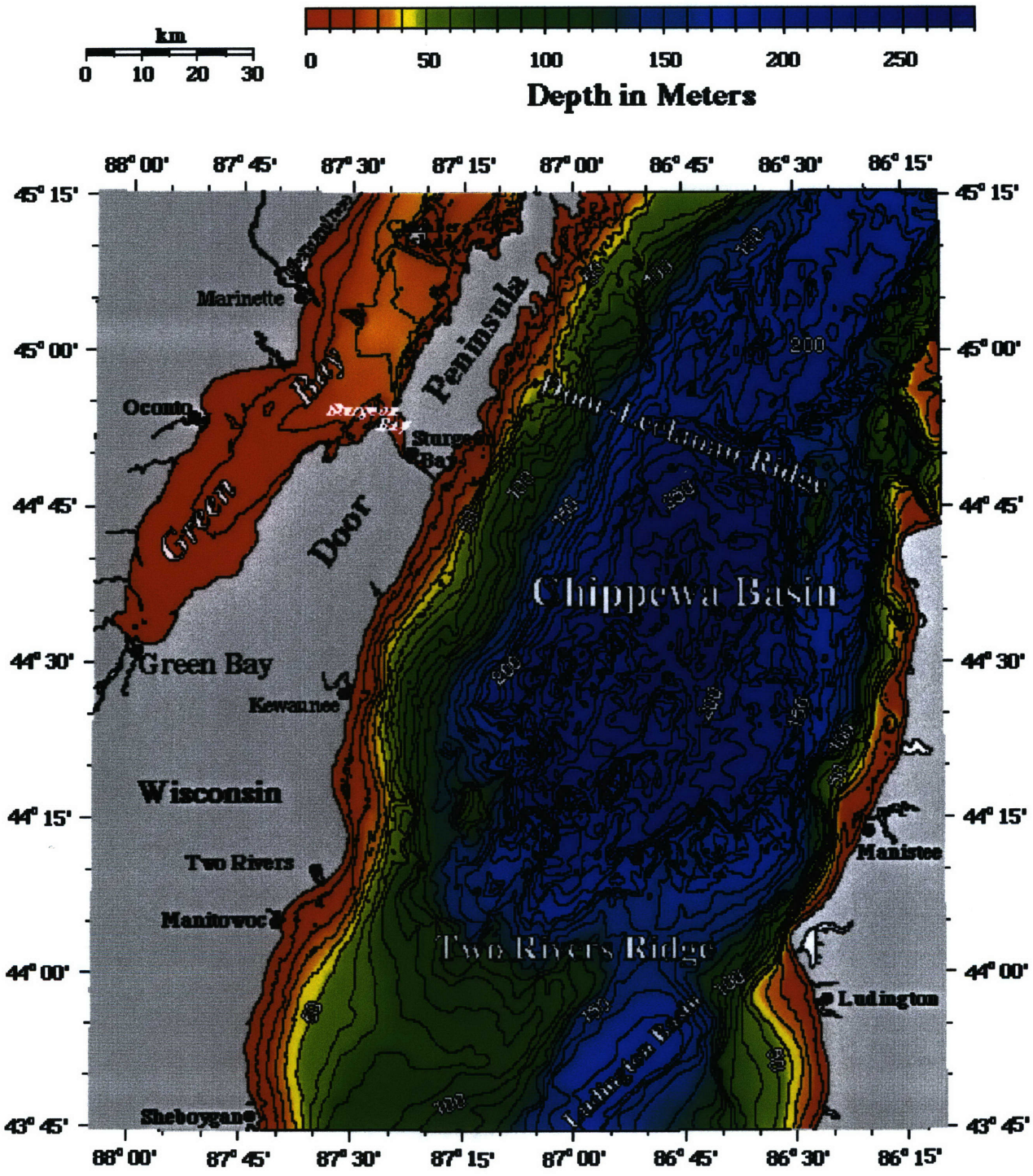


Figure 2-5: Topographic contour map of northern Lake Michigan[85]. In this map, the lines of constant depth (isobaths) are color-highlighted to assist the reader (cooler colors are deeper).

form of geodetic mapping, but since digitized topological maps predate gravimetric and magnetic maps by decades, we will address the issue here as it pertains to AUV navigation.

Topographic representations

When the computer revolution came to cartography, the chosen solution to map representation was the vector format [96]. In vector format, a feature is stored as a series of {X,Y} locations. Each location has an *information tag* associated with it such as population, altitude, type of feature, etc. These {X,Y} coordinate pairs were then stored as a list, which in itself generated numerous tiling and indexing schemes (adaptive and fixed tiling, quadtrees, R-trees, Morton codes, Peano codes, etc.). Accessing and manipulating these indexing schemes has since become an entire branch of cartography. Later, when computer displays became predominantly raster-oriented, maps were stored in either raster or vector format with storage choices made according to projected use. Raster format is generally preferred for simpler, low resolution functions where accessing the database quickly is of key importance, while vector format predominates in applications where speed is not as crucial as precision and representation over a variety of scales.

This dichotomy has since carried over to the world of mobile robots. With robot systems there is also the added problem of restricted data storage. Most detailed cartographic databases consume large amounts of storage space, and robots generally do not possess such reserves. Indeed, they may contain very little space for *a priori* datasets, reserving storage instead for the data they were sent to collect. If the mission is one of map-making, then the vehicle is generally expected to add to its on-board maps in a timely and reliable fashion. They are also required to access and utilize their maps quickly and efficiently if they are to navigate successfully. Survey-class AUVs such as the *Odyssey IIb* have an additional handicap in that they cannot stop and think while determining position owing to the basic nature of a dynamically controlled underwater vehicle.

A topographic map is basically a two-dimensional projection of a three-dimensional

function of position. While the real world can be non single-valued (such as in the case of caves and overhangs), topographic maps – especially ones designed for robot use – generally assume the world is single-valued in the z-direction:

$$z = f(x, y). \tag{2.5}$$

If we view the world as $f(x, y)$, then the topography of the bottom can be represented by the slopes and extrema of $f(x, y)$. This is the approach used by many when formulating a model of the world. How we choose to represent the slopes and extrema varies according to application.

Mathematically we can view slopes as directional derivatives of the function f at the point (x, y) in the direction β. For any topographic description $z = f(x, y)$ the slope in the direction β is

$$f'_\beta(x, y) = \frac{\delta f}{\delta x}(x, y) * sin(\beta) + \frac{\delta f}{\delta y}(x, y) * cos(\beta). \tag{2.6}$$

To represent extrema, we need the second derivative in the β direction

$$f''_\beta(x, y) = \frac{\delta^2 f}{\delta x^2} * sin^2(\beta) + 2 * \frac{\delta^2 f}{\delta x \delta y} * sin(\beta) * cos(\beta) + \frac{\delta^2 f}{\delta y^2} * cos^2(\beta). \tag{2.7}$$

It follows that the *gradient*(∇f) of a vector whose magnitude,

$$||\nabla f|| = \sqrt{\left(\frac{\delta f}{\delta x}\right)^2 + \left(\frac{\delta f}{\delta y}\right)^2}, \tag{2.8}$$

at a given (x, y) is by definition the direction of the maximum rate of change of f at that point, i.e. the steepest slope. The direction of maximum slope at that point is

$$\beta_{max} = tan^{-1}\left(\frac{\frac{\delta f}{\delta x}}{\frac{\delta f}{\delta y}}\right). \tag{2.9}$$

Calling on work going back to Cayley [33] and Maxwell [79], Haralick *et al.* [50] break the world into topological primitives. They base their topological descriptions

on the special cases of these derivatives, calling them $\omega^{(1)}, \omega^{(2)}, \lambda_1$, and λ_2, where $\omega^{(1)}$ is the unit vector in the direction in which the second directional derivative of f has the greatest magnitude (i.e. the direction of greatest change in slope), $\omega^{(2)}$ is orthogonal to $\omega^{(1)}$, λ_1 is the value of the second derivative of f in the direction of $\omega^{(1)}$, and λ_2 is the value of the second derivative in the direction of $\omega^{(2)}$. Harlick also observes that if values of $\omega^{(1)}$ and $\omega^{(2)}$ are calculated first, then the values of the first directional derivatives can be simply calculated as $\nabla f \cdot \omega^{(1)}$ and $\nabla f \cdot \omega^{(2)}$.

Using these definitions, they then define a set of basic topological features: *peak, pit, ridge, ravine, saddle, flat,* and *hillside.* A peak is a local maxima, where all adjacent areas are lower. A pit is also an extrema excepting that it is a local minima, where all adjacent areas are higher. A ridge is a set of points $\{x, y\}$ forming a line such that the points on either side of the ridge are lower than the ridge points. A ravine is of the same construction as a ridge excepting that all points adjacent to the set $\{x, y\}$ are higher than the ravine line. (Note that ridge and ravine lines need not be level. They can curve or slope up or down.) A saddle point is where a local minimum occurs in one direction while a local maximum occurs in a direction perpendicular to that. A flat is simply where the surface is level (i.e. zero gradient). Finally, a hillside is all other points not covered by the previous definitions. It may be a tilted plane, a convex or concave slope, or contain an inflection point between the two slopes.

This *extrema*-based classification scheme is popular because it explicitly retains key topographic features in an easy to manipulate analytic form, and allows the reconstruction of an arbitrary surface with the use of selected primitives. Since Haralick *et al.* were using this approach in a desktop-based still-image vision system, storage, and processing requirements were not a concern.

Other forms of the *extrema*-based map have been employed. Nackman [87] proposed the critical point configuration graph (CPCG). The CPCG uses a subset of the full list of primitives: *peaks, pits,* and *passes.* Peaks and pits are as before, and passes fall into the same topological class as saddle points. Any topographic feature can then be constructed using combinations of these primitives. Orser and Roche brought the concept to the underwater community by focusing on the identification

and extraction of topographic features of bathymetric maps as a navigation aid. The central concept was the extremal point topography network (EPTN), which reduced the topological primitives set to peaks, pits, valleys, ridges, saddle points, and a special class of closed contours around the extrema which consist of inflection points. This final feature is used to aid in the differentiation between "hills" and "dales." Several methods of extracting the EPTN were tested on a sonar data set from Lake Winnipesaukee, NH. The issues of navigation were not, however, addressed.

Haralick [50] describes topography as a function of first and second directional derivatives of the terrain. Using this system, the world can be categorized into the basic types: peak, pit, ridge, ravine, saddle, flat, and hillside, with hillside having the subcategories: inflection point, slope, convex hill, concave hill, and saddle hill.

Kweon [61] classifies topography into four basic classes: peaks, pits, ridges, and ravines. Linking these features is a connectivity tree referred to as the *Topographic Change Tree*, which interpolates between topographic classes.

The primary drawback of *extrema*-based topological representations is the need for the extrema themselves to exist in the area of operation in order for the representation to work. Mountainous regions are easy to describe, while areas such as abyssal plains, which lack any outstanding topological characteristics, are relatively difficult.

Terrain Contour Matching (TERCOM)

Perhaps the most successful topographic navigation system today is the TERCOM system (see Hatch [51] and Hostetler [54]), used by cruise missiles. TERCOM relies on inertial navigation to guide the missile between navigational "waypoints". These waypoints are regions of sharp topographic relief which are well known, and have been imaged and quantized in advance. To reduce the solution space, each waypoint is treated as a separate map. Also, the topography is quantized into large pre-defined regions with an average altitude stored for each region. The TERCOM system senses the average altitude in the regions adjacent to the missile's actual path, and compares these values to the quantized regions adjacent to the intended ground track. The vehicle then makes course adjustments to compensate for discrepancies between the

missile's expected position and its actual location. Because of the enormous storage requirements for the maps, every possible data and computational load reduction technique has been applied. The key here is the techniques of mapping only the specific waypoint regions, relying on inertial navigation between these waypoints, and the decision to gradually increase map resolution (and reduce the corresponding mapped area) as the missile approaches its target, thereby progressively reducing the tolerable navigational error only as necessary (referred to as "accuracy funneling"). Even with these measures, the system is still only accurate to within, at best, 30-100 meters [89]. Final guidance is handled by the Digital Scene Matching Area Correlator (DSMAC) [32], which employs a vision-based template matching system.

The real technical difficulty with TERCOM lies not with the guidance system itself, but in the technical infrastructure needed to create the digitized maps. For any *a priori* map-based navigation system to be effective, databases must be built up of every part of the world in which they might potentially be used. The cost of this exercise to the US for cruise missile systems alone is estimated to approach the total investment made in TERCOM-equipped cruise missile hardware. It is reported that following the Iraqi invasion of Kuwait in 1990, the US military had to embark upon a crash program to prepare data for the TERCOM guidance systems of its Tomahawk missiles for use in the area. The lack of suitably dramatic topography in southern Iraq, coupled with the lack of suitable datasets for the missiles resulted in missile paths which "wandered" far from the straight line path in order to maintain the necessary navigational fixes. The resulting missile effective radius was correspondingly reduced.

In contrast to TERCOM is SITAN, or Sandia Inertial Terrain Aided Navigation system [54]. SITAN uses single terrain clearance measurements and incorporates them into an extended Kalman filter. Unlike TERCOM, SITAN explicitly uses each new sensor reading. A derivative of SITAN was employed by Jircitano [56] for gravity-based AUV navigation. However, SITAN relies on a linearization of the terrain model and is therefore subject to divergence.

Multiple maps

The majority of map-based navigation approaches employ a single map and one type of sensor. In contrast, Tuohy has developed techniques for geophysical navigation using multiple *a priori* maps of different geophysical features [111]. Rather than attempting to locate a specific feature and comparing it to an *a priori* map, he instead uses the concept of *contour intersection* to determine all places where two different geophysical parameters (e.g. magnetic field, gravitational field, bathymetry) would have coincident isocontours. Ambiguity involving multiple points of intersection are handled by a navigation-directed gating function. Like all map-based systems, this method of geophysical map-based navigation is limited by the quality and resolution of its *a priori* maps and the vehicle's on board sensors.

Local maps

Local maps are used by autonomous vehicles for the purposes of path planning [4, 63] and obstacle avoidance [20, 21, 108]. They are either *a priori* maps of the region of operation or they are constructed by the vehicle [66, 105]. Once created or generated, these maps are used by the vehicle for the purposes of determining where to go next to a) gather more information, b) avoid obstacles and/or, c) achieve a goal. If any portion of the map is tied back to geodetic coordinates, then the map may also be used for localization purposes. However, this is usually a secondary function after the more immediate concerns of vehicle maneuvering.

There are two dominant philosophies to vehicle maneuvering in the local environment: trajectory planning and potential field. Trajectory planning approaches have been applied from the earliest days of mobile robots. The concept is simple: given the location of all local obstacles and given the position and trajectory of the vehicle itself, it is a simple matter to calculate a path through the field of obstacles. The basic motion planning problem can be defined as follows [63]:

- Let $\mathcal{R}$ be a rigid object (the robot) moving in a euclidean space, called the *Workspace*, represented as $\Re^N$ with $N = 2$ or 3.

- Let $\mathcal{O}_1, \ldots, \mathcal{O}_n$ be fixed rigid objects distributed in $\Re$ where the $\mathcal{O}$'s are *Obstacles.*

- Assume that both the geometry of $\mathcal{R}$ and $\mathcal{O}_1, \ldots, \mathcal{O}_n$ and the locations of $\mathcal{O}_1, \ldots, \mathcal{O}_n$ in $\Re$ are accurately known. Assume also that no kinematic constraints limit the motion of $\mathcal{R}$.

- The problem can then be stated as follows: Given an initial position & orientation and a goal position & orientation of $\mathcal{R}$ in $\Re$, generate a *path* $\mathcal{T}$ specifying a continuous sequence of positions and orientations of $\mathcal{R}$ which avoid contact with all $\mathcal{O}_i$'s. This *path* $\mathcal{T}$ starts at the initial position & orientation and ends at the final position & orientation. Report if no such path exists.

This basic problem has since increased in complexity and been extended by various means over the years. Udpa [116] introduced the concept of shrinking the robot to a point in an appropriate configuration space. Lozano-Perez *et al.* [71] extended this concept to include polygonal & polyhedral robots and obstacles without rotation. Chatila [34] extended motion planning to include incomplete knowledge of the environment. In 1983 and 1984, Schwartz and Sharir [101, 102, 103, 104] published a series of papers, called the "Piano Mover's Problem" series, which introduced planning of free paths for polygonal objects which were allowed to both rotate and translate in 2-D space.

In 1985, Khatib [58] presented the *potential field* approach as a real-time collision-avoidance system for mobile robots. He then went on to extend this to motion planning. At approximately the same time, Brooks [22] showed that planning was unnecessary in a mobile robot using a potential field approach, insofar as obstacle avoidance was concerned. Barraquand and Latombe [5] later combined the potential field approach with random techniques to enable a robot to escape from the *local minima* problem. In 1991, Koren [59] showed that the potential field approach has inherent limitations brought about by the nature of the dynamics of interaction with groups of potential fields.

Other refinements, such as cell decomposition [70], non-holonomic (i.e. car-like) vehicles [64], moving obstacles, multiple vehicles [63], and efficient terrain-covering algorithms [52] have also been examined. All of these approaches assume some knowledge of the environment, either in a detailed *a priori* map or some map with a measure of uncertainty in location or sensor accuracy.

2.2.5 Summary

The discussion of mapping and navigation presented above is necessary in order to understand the requirements, limitations, and structure of the feature relative navigator presented in the next chapter. This thesis presents a feature relative navigation approach which in turn is capable of utilizing some form of navigation for referencing back to the world. To this end, we have presented the most common and popular forms of navigation and their associated limitations. In future chapters we will discuss the navigation choices made, their impact on the FRN approach, and the effects the associated sources of error have upon the system.

2.3 Map building

In the mobile robotics community, the process of map building is an essential component of navigation and data collection. It is therefore important to understand what is meant by "a map" and why the type and resolution of the the map that is chosen is important. The choice of map is influenced by the type of representation, method of representation, and level of detail for a given vehicle, sensor suite and mission. Rather than viewing the world at one level, Stewart [107] chooses to view the world as a series of multi-layered representations running from "low-level" or physical descriptions to "high-level" or cognitive representations.

Low level representations tend towards physical or sensor-based descriptions (e.g. rough, smooth), while high level descriptions are more abstract and have some implicit model of the world included (e.g. desk, chair). High-level representations consist of *primitives* designed to show the characteristic nature of the feature(s) they rep-

resent. These primitives not only represent the feature but also aid in the vehicle's ability to extrapolate information, either to direct additional information surveys or to reduce/eliminate the need for further sensor use. For example, if a vision system without an effective high-level mapping and modeling system sees an occluded object, it may have to maneuver the vehicle to gather more information about that object – its dimensions, location, etc. If, however, the system was capable of classifying the sensed object, then it does not need additional information to determine the extent of that object as the remaining information might be inherent in the object description.

To illustrate the difference, imagine the task of mapping an office building. Storing a detailed volumetric description of a desk would take a substantial amount of storage space, logging whether a given element of volume, or *voxel*, was occupied or not. If there were 50 desks, it would take 50 times the storage space. On the other hand, an object-level description would have some pre-existing knowledge of a desk (e.g. width, length, height, location of the center). A description of a desk is now just "desk centered at location $\{x, y\}$" and 50 desks is "desks with centers at the locations $\{\vec{x}, \vec{y}\}$" – a small increase of storage space for a large increase in the number of objects.

In contrast, low-level representations are more tightly coupled to the sensors and the physics of the environment. Such maps are cellular in nature, consisting of either pixels (2-D) or voxels (3-D). These representations require more storage then a high-level representation but can retain more detailed information that may be lost in a more abstract representation. This is particularly true for natural features which do not lend themselves to simple descriptions – if, for example, we describe a natural feature such as a seafloor trench as a half of a prolate spheroid, we encapsulate one essential aspect of the trench (e.g. a sharp depression in the seafloor), but lose information about the exact details of the shape of the trench (which may be useful if we wish to differentiate one particular trench from another). On the other hand, if we describe the same trench voxel-by-voxel, we retain the detailed information of the trench (assuming a suitably chosen voxel size) and what makes it unique, but at the cost of increased storage space for the description.

How one chooses to represent the world and what type of map to use depends upon

what uses the map will be put to and, conversely, dictates what uses the map will be good for. There are several key issues that need to be considered when choosing a representation:

- Type of map - Do we need a map that is low level (i.e. cellular), high level (i.e. objects), or something in between?
- Information - What is the map going to represent? How is the information stored? How does this affect ways in which the map can be used?
- Efficiency - How easy is it to access? Can it be updated? Is the information easy to manipulate? Can key information be directly accessed, or must it be reconstructed? If reconstructed, how easily and quickly can it be done?
- Completeness - Does the map represent sufficient information to be usable by the vehicle to perform the desired misson? Is it thorough enough? Is it accurate enough?

The role that the map will play in a vehicle will dictate the necessary answers to these questions. These answers in turn will also prescribe the limits to which these maps may be employed.

While there are many intermediate representations and techniques, such as quadtrees, adaptive tiling, fixed tiling, R-trees, etc. (see Robinson, *et al.* [96]), we will focus on the two extremes of cell-based and object-based descriptions to better emphasize the differences between them. Note, however, that it is common practice to employ some hybrid approach rather than to use one method or the other.

Type of map

As a higher-level example, Leonard [66] constructed rooms out of walls and vertices. Knowing the location of the vertices allows the reconstruction of the room without the penalties of storing a map made of tousands of individual cells, each of which stores the presence or absence of every portion of every wall. Thus, a room can be mapped using a minimum of storage resources. This philosophy can be scaled up

without limit, mapping any number of rooms with only a modest increase in storage requirements.

In contrast, Stewart [107] argued that because vehicle and/or sensor target position may not be generally well known (e.g. a free-floating vehicle in an underwater environment), a low-level or cellular representation may be better suited for mapping purposes. This representation, coupled with the Dempster-Shafer inference rule, would be a method whereby the vehicle can construct and maintain a description of the environment and express its confidence about the representation of that environment. Pagac *et al.* [90] used this approach to construct an office environment map similar to the one used by Leonard [65] and divided it up into 50 mm cells. While the resulting room description consumes much more storage space, it also contains more details in the form of both the geometry of the room and a measure of the level of confidence that the vehicle has in the location of each map element in the room. This increased level of detail may or may not be desirable, depending on the uses to which the map is to be employed.

Because the feature relative navigation system presented in this thesis is desiged to operate in a world of natural features, a high-level representation of areas and contours was deemed inappropriate for the AUV on-board maps. Instead, a cellular approach was chosen to represent features in order to better capture the details of those natural features. However, there are cases where a high-level description may be preferred, such as when the vehicle is determining whether or not all of a specific feature has been mapped yet. Fortunately, such high-level descriptions can be derived from low-level maps at the expense of additional processing, allowing us to take advantage of the best of both worlds when necessary.

Information

Having chosen a map representation, the next question is what information we wish to represent in our map. Information is represented in computer mapping systems in two ways. The simplest level of information a cell can contain is whether the cell is occupied or not. This serves to indicate the presence or absence of a feature (if the

map represents features) or whether a cell has been visited/sensed or not. If the cell has been measured, then some value is stored there. If not, then that space is empty.

A more intricate and interesting level of information representation on a cell-based map is when the cell contains a pointer to a more detailed description elsewhere. Such a map might indicate the presence of information at a particular location (cell), the contents of that cell then being a pointer to a more detailed description in another location. This in effect combines the low-level representation (cells) with a high-level representation (objects). An example might be the distribution of a plankton bloom. A given map cell could act as a pointer to all relevant information about the bloom at that location. Information such as temperature, dissolved oxygen, and turbidity are then accessed by referencing that cell. This allows us to store complex information without sacrificing the simplicity of a cell-based map; hence we choose to employ a map of this type.

Efficiency

In the case of map manipulation, efficiency can be thought of as the ease with which the vehicle can access and manipulate information stored in the map. In general, cellular maps consume more space in memory than high-level descriptions, but the structure makes them well suited to matrix manipulation techniques. A cellular or voxel map can be mapped directly into a two- or three-dimensional array, making access and manipulation simple. An object-oriented map sacrifices speed for storage efficiency. Information about a specific location must be reconstructed from the abstract representation of the region. Another efficiency/size trade off is data compression, the most common example being image compression to reduce storage and transmission requirements. The reduced size is balanced against the time needed to "reinflate" the image and the amount of detail loss that is considered acceptable.

We have chosen a cell-based map representation, thus making the standard search and sort routines for matrices available for our purposes. This representation can still be used for more object-oriented information representations by increasing the dimensionality of our map and associating pointers to other data locations with individual

cells in the original map. For example, an $[m \times n \times D]$ array-format representation may store temperature, conductivity, turbidity, and bottom depth as individual dimensions of D at any given vehicle location (m_i, n_i).

Completeness

Whether a map can be considered "complete" or not depends upon its use and method of construction. An *a priori* topographic map of an area might be presumed to be complete insofar as it contains the necessary information for someone to navigate when using it. If there is no information about a specific region on that map, then that map (or at least that region) is considered incomplete. The map's accuracy generally meets some predefined standard for maps of its type.

In our particular case of an autonomous vehicle trying to make its own maps, the search region is considered unknown until the vehicle itself maps it. It is now up to the vehicle to determine if a given area is completely mapped or not, and if so, how accurate that map is. Singh [105] used an information theoretic approach to gauging whether or not sufficient information has been obtained about any given location. A high entropy rate generated by each sonar ping indicated new information was being received and added to the map. When the rate fell off sufficiently, that area was considered mapped. Aramaki and Ura [1] used an "index of reliability" which is related to the cumulative probability that the depth measured in a given location was accurate. When the index exceeds 0.90, the area is considered mapped.

Because we assume a point sensor mounted on a non-holonomic AUV such as an *Odyssey IIb*-class vehicle, we must physically visit each map cell to sample it. If the cells are sufficiently large (e.g. several meters on a side) and the sensing rate is sufficiently high (the AUV *Odyssey IIb* sonar altimeter samples its environment at approximately 5 Hz), we may state that any cell visited by the AUV where the vehicle passes in or close to the center of that cell has been satisfactorily sensed. Therefore, cells which have been so visited may also be considered completely mapped for the purposes of feature-relative navigation. Hence, our measures of completeness become simply a question of whether or not a given cell has or has not been visited by the

AUV.

2.4 Intelligent control

Intelligent control of autonomous vehicles is a vast and expanding field of research. As such, only portions can be covered in this thesis. Specific highlights will be presented which are relevant to the issues facing autonomous underwater vehicle navigation and control.

Software architectures developed for land robotics do not necessarily transfer directly into the underwater environment. Additional factors must be considered when designing for marine systems, such as:

- Sensors – Sea water blocks most forms of electromagnetics and causes signal distortion, blockage, and attenuation in acoustic systems. This makes it difficult and energy intensive to reliably sense the vehicle's surroundings as well as communicate with support ships.

- Dynamics – Land robots are typically confined to two dimensions, while the ocean environment is inherently 3-D. This increases the number of degrees-of-freedom which must be modeled from three to six, along with the accompanying increase in modeling and mapping complexity. Also, the hydrodynamics of maneuvering and control for a free-floating marine vehicle are much more complex than for a wheeled robot on land.

- Real Time – In the world of land robotics, real time computation and control capabilities are desirable, but not necessary for safe functioning of the vehicle. In the marine environment, the presence of currents and the free-floating nature of AUVs require real time sensing and control just for basic functionality.

Planning-based intelligent control

Current intelligent control methods can be divided into four broad categories: planning, reactive, hybrid, and blended. Planning architectures (sometimes referred to

as "stop-look-think-act" systems) are the original and best studied method of autonomous vehicle control. In a planning system, data is first collected and processed. Processed data is then used by a *planner* to decide what to do next. This method of control allows time for the optimum path, course of action, or sequence of acts to be determined, regardless of the level of complexity of the task. There are, however, penalties to be paid in terms of capabilities and performance in real-world situations. Planning systems generally assume "perfect" sensors, large or infinite computational power and all the time that may be required to devise the best plan. The DEVISOR/HOMER [118] system operates in a marine environment with several buoys, ships, and natural objects in the area. All objects are assumed to be identified at all times. In other words, the red buoy will *never* be mistaken for the green buoy by the vehicle. Also, it is assumed that nothing moves while the vehicle formulates its plans. Finally, the vehicle's position and the positions of all other objects is precisely known at all times. In the real world, sensors are imperfect, computational resources are finite, the time in which to make decisions is short, and *nothing* stops and waits while the vehicle makes up its mind about what to do. In practice, most planning systems can only function in simple, limited, highly structured environments, or with abundant amounts of time [3, 110, 118].

Behavior-based intelligent control

In the 1980's, Prof. Rodney Brooks of the MIT Artificial Intelligence Laboratory proposed a purely reactive form of vehicle control [22, 23]. Called *subsumption* architecture, this method of control is based on the premise that living creatures have the ability to react to certain stimuli without consciously doing so. For example, when someone accidentally touches a hot object, they automatically react by pulling away from it. Only afterwards do they consciously realize what happened. The use of this approach divides vehicle control into multiple *behavior* modules, or *behaviors*, where each behavior reacts automatically to certain stimuli. Modules communicate with each other by reinforcing or inhibiting each other's output with the resulting net outputs controlling the vehicle.

This approach has several key advantages: low processing requirements (many behaviors are simple enough to be embedded in individual motor controllers), fast reaction times (extremely important for real time robotics), and modular architecture (easy to add or delete behaviors). In the process of implementation, however, two serious disadvantages were discovered: the "scaling problem" and "situatedness."

The scaling problem is that as more modules are added to a robot, the the overall complexity of the system as a whole rapidly increases as the factorial of the number of behavior modules involved – due to the behavior-behavior connections. The actions of the vehicle become difficult to predict because of this complexity of interactions between the behaviors [24, 27, 31, 78]. This unpredictability is considered interesting or even desirable in some areas of artificial intelligence (AI) [75], but not when one is trying to create a trustworthy AUV.

Situatedness is a problem inherent in any behavior-based vehicle. A given vehicle using a certain set of behaviors is suited for a specific environment and lacks the ability to alter its behavior if confronted by unforeseen conditions. A typical vehicle mission may have many different phases, or situations, each of which requires its own set of behaviors. For example, a bottom-photography mission has at least three distinct phases: 1) deployment from the ship and transit to the search area, 2) photographing the bottom, and 3) return and retrieval. Each of these phases requires different sets of behaviors, some of which actually conflict with each other (e.g. Obstacle Avoidance vs. Rendezvous and Docking).

In a subsumptive system, the only recourse is to pre-plan for all possible contingencies, incorporating the necessary behavior modules and their interrelationships in advance. However, doing so returns us to the scaling problem as well as the problem of how to insure that only the proper behaviors control the vehicle at the appropriate times.

Hybrid and blended intelligent control

Several people have proposed adapted forms of subsumption architecture in response to these difficulties. These new forms of intelligent control can be divided into two

basic forms: *hybrid* and *blended* systems. Hybrid intelligent control systems use elements of "traditional" AI to control behavior modules (e.g. Connell's Symbolic-Subsumption-Servo architecture [36]), while blended control uses the behavior-based structure of subsumption architecture incorporating individual modules designed along the lines of more traditional AI systems. Both types of intelligent control seek to exploit the fast reaction abilities of a behavior-based vehicle while also exploiting the learning and planning abilities of the traditional planning architecture.

Two key examples of hybrid control are *learning augmented subsumption architecture* [36] and *state configured layered control* (SCLC) [9]. In learning augmented subsumption architecture, the vehicle designer uses various learning techniques such as Q-learning [3] and reinforcement learning [3] to teach the robot how to perform the task. This method has some serious drawbacks in that it takes several time-consuming and potentially disastrous training runs to teach the robot. Attempts at using simulator data to augment the learning process have proven inadequate due to the "sanitized" nature of simulated data – the simulation-based training systems lack the uncertainty and noise found in real-world robots [3].

State configured layered control

In state configured layered control (see Figure 2-6), the behavior modules are all manipulated by a central *state table* that controls which behaviors are running at any moment and what their operational parameters are. In its simplest form, the state table is preconfigured for all foreseeable contingencies. During a mission, the state table automatically switches specific behaviors on or off, and changes operational parameters of running behaviors according to current conditions. This method overcomes the limitations of *situatedness*, but is limited by the programmer's ability to foresee all possible situations that the vehicle will encounter.

Blended control incorporates the abilities of a planner with the structure of a behavior-based system. The result is a set of *smart behaviors* which can forecast and act in much the same fashion as a planning system, but without the same levels of sophistication and detail. Each behavior is aware of both the environment and the

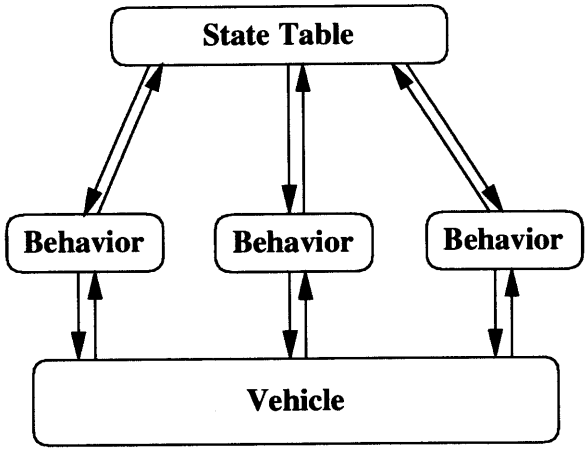


Figure 2-6: State configured layered control. From Bennett [18].

status of other behaviors. This awareness allows behaviors to rethink their plans or even negotiate with other behaviors for control of the vehicle. In a conflict situation, the behaviors can yield to each other according to each one's relative importance and other conditional factors. Thus, for example, if an obstacle avoidance (O/A) module insisted on steering the vehicle in one direction while a survey module insisted on going in the other, the survey module would "know" that the O/A behavior is dominant and therefore would alter its survey path to accommodate the demands of the O/A module. Although this scheme solves the situatedness problem inherent in behavior-based systems and introduces the ability to plan for contingencies on the fly, it introduces the additional problems of determining which behaviors dominate which and when (thereby further complicating the scaling problem) as well as increase the risks that a) the smart behaviors may not always run in real time and b) the net computational load of the combined behaviors will overload the vehicle processor.

Two forms of layered control were proposed as solutions to these problems: *arbitrated layered control* and *supervised state configured layered control.* Arbitrated layered control was described in 1990 by Bellingham *et al.* [10], while supervised state configured layered control was proposed and implemented in simulation in 1993 by Bennett [18].

Arbitrated layered control

In arbitrated layered control, the behavior modules are all competing for control of the vehicle, but they are no longer allowed direct access to the dynamic controller. Instead, they all submit control commands to a central arbitrator which in turn passes the "winning" command set to the dynamic controller (see Figure 2-7). In its simplest form, the arbitrator's task is to determine which behavior has priority over all others and passes those commands along to the controller. More commonly, the arbitrator will attempt to choose a command which will satisfy as many behaviors as possible. It does so by determining which commands are mutually compatible and/or do not directly interfere with each other. It then takes the resulting fused command and passes it along to the controller. For this method to work, the modules must have three new capabilities: some method of handling partial states of completion, some way of interaction with the arbitrator for the purposes of negotiation, and the ability to modify output to accommodate the goals and requests of other behaviors.

The keys to successful implementation of arbitrated layered control are the restriction of communication paths and the ability to communicate only via a central *data structure.* In the traditional form of subsumption architecture, any behavior can communicate with or influence any other behavior. This was what led to the scaling problem. In arbitrated layered control, communication between behaviors is restricted to posting to and reading from a central data structure (sometimes referred to as a "blackboard"). This restriction has two immediate benefits: 1) specific behaviors can be inserted or removed at will without disrupting interconnections between behaviors and 2) behaviors cannot arbitrarily influence the inputs our outputs of other behaviors. Each behavior is responsible for reading and processing information posted to the data structure and then acting upon it if necessary. The resulting commands from each of the behaviors are then fed into the *arbitrator* which attempts to satisfy as many behaviors as possible. For example, if a bottom following behavior requested a certain depth and speed but no heading, and a survey command requested a certain heading and speed but no depth, then the arbitrator can issue a fused command to

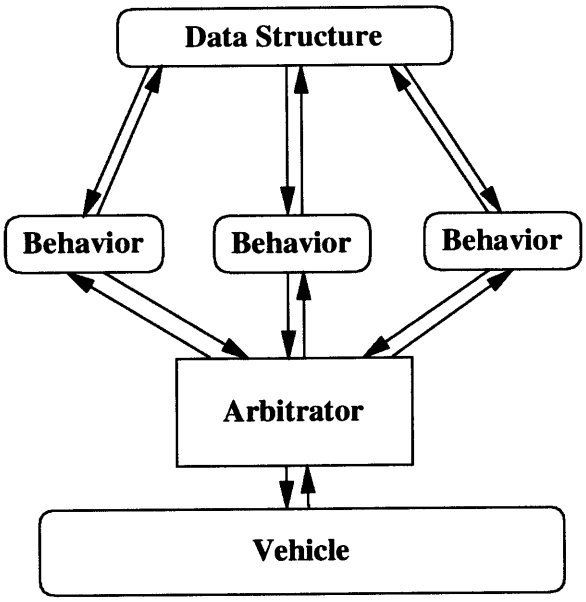


Figure 2-7: Arbitrated layered control. From Bennett [18].

the dynamic controller with a depth set to satisfy the bottom follower and a heading set to satisfy the surveyor, with a speed set to accommodate both or whichever of the two was considered "dominant."

An extension of this approach if the concept of *aspirations* [18]. If, for example, an O/A behavior wants to steer away from a detected target, it would normally issue a heading change to do so. It does not matter to the O/A system which way the vehicle turns, only that it does. It therefore issues the desired heading change. If there were another behavior also controlling heading, such as a survey behavior, then that behavior might also be requesting a heading change. Both requested headings would satisfy the need to avoid collision, but the arbitrator would accept the O/A command over the survey command because survival behaviors must always win to preserve the vehicle. Now replace the rigid command of the O/A system with the *aspiration* of turning away from the obstacle. It now only wants a heading change that steers the vehicle safely away from the danger but does not care what that heading is. Instead of a fixed heading change, it hands the arbitrator a range of valid headings which will satisfy the O/A requirement. This same type of output is also issued by all other behaviors, including the survey behavior. Instead of making a simple dominance-based decision, the arbitrator now determines a heading (depth, speed,

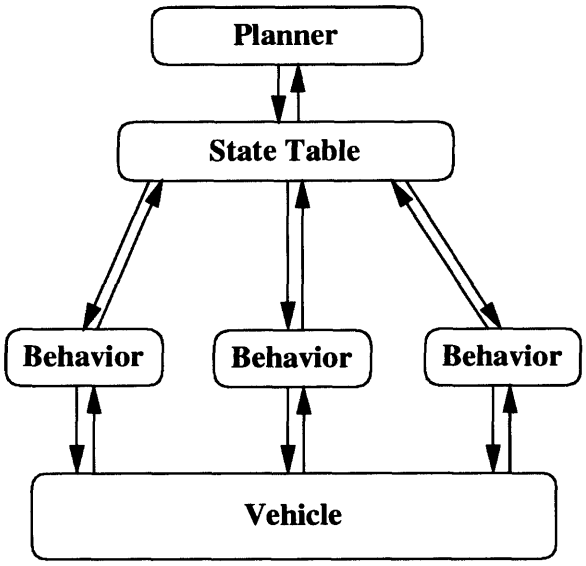


Figure 2-8: Supervised state configured layered control. From Bennett [18].

etc.) which satisfies as many behaviors as possible, starting with the most important (the O/A behavior). Thus it is possible to satisfy multiple behaviors without extensive interconnections.

Supervised state configured layered control

State configured layered control, with or without arbitration, offers a simple and effective method of linking a high-level planning module to a collection of low-level behavior modules in much the same fashion as a craftsman with a box of tools. If the state table is considered an intermediate layer (see Figure 2-8) it can be used as part of a flexible *plan and compile* architecture [83]. In this form the state table contains all of the operational parameters needed for a given phase of the mission or the current state of the vehicle as well as information relating to any contingency plans and other possible states. However, instead of following a pre-determined set of vehicle states, the state table acts in the role of a *transition table*, receiving and relaying information from the behaviors to a master planner. The planner in turn can alter any or all of the behavior sets and their transition rules in the state table as new information becomes available during the mission. Using this structure allows a planner to work out a long time horizon and potentially time consuming calculations

without sacrificing the fast reaction abilities of a behavior-based system. This form of layered control offers the potential to take advantage of the strengths of a planner-based system (flexibility and adaptability) while at the same time retaining the speed and efficiency of subsumption architecture.

Adaptive sampling

Adaptive sampling is the technique of modifying the trajectory of a vehicle in response to sensed data in order to obtain the optimal sampling pattern which will most efficiently characterize a given phenomenon. As such, it can be viewed as a form of intelligent planner if situated within an intelligent control structure. There are two primary forms of adaptive sampling: *field based* and *feature based.*

A field based approach attempts to determine the best sampling interval or locations to adequately characterize a distributed phenomenon. The sampling interval, speed, and locations depend upon the spatial & temporal capabilities (i.e. maneuvering, speed) and the resolution & dynamic characteristics of the sensors employed.

Bellingham *et al.* [17, 121] examined the problem of adaptive sampling of a distributed oceanic phenomenon using the survey-class AUV *Odyssey II.* The best path was determined based upon the presumed spatial frequency of the phenomenon, its rate of evolution and the speed & endurance of the AUV. The resulting path was designed to be the optimum sampling strategy needed to obtain the best possible distribution of statistical information about a region, given the duration and speed restrictions imposed by the vehicle. Cooperative strategies were also examined; i.e. the spatial and temporal information obtained using coordinated fleets of AUVs.

Singh [106] and Burien [29, 30] examined the issue of field based adaptive sampling from the perspective of gradient following in an effort to locate the local maxima or minima in a given search area. The approach proved effective in two different scenarios using a sonar in Herring Pond, near Falmouth, Massachusetts (Singh), and in simulation using bathymetric and thermal data (Burien). Their technique is designed to locate the local maxima or minima and not map a distributed feature.

A feature based approach is designed to locate one or more features in the world

and map their number and/or extent. In this case, the vehicle responds to the presence or absence of a feature. It is designed to maximize the vehicle's "time on target" or ratio of the time spent examining features vs. total mission time.

Information theoretic approaches such as in Stewart [107], Singh [105] and Aramaki [1] used a sonar as the sensor and treated the entire world as one feature. Portions of this "feature" were then imaged until either the change in the rate of new information dropped below a given threshold (Singh), or confidence in the value of a given voxel (Stewart) or pixel (Aramaki) exceeded a given threshold.

2.5 Summary

This chapter presents a review of the state of the art in autonomous underwater vehicle use, navigation, and intelligent control. First some background is presented on the use of AUVs as oceanographic exploration equipment. Having done so, we then discussed the three fields of AUV navigation, map building & maintenance, and intelligent control and how each of these areas of research influence the subject of feature relative navigation.

Both traditional and non-traditional methods of navigation are discussed and presented, along with their associated weaknesses and sources of error. Map building techniques are then presented in the context of use by autonomous vehicles for the purposes of navigation. Issues of representation, storage, management, and computational complexity are discussed and presented. Finally, the broad subject of intelligent control is presented and discussed in the context of planning vs. behavior-based approaches, with emphasis on the creation of hybrid systems incorporating aspects of both approaches.

The following chapter presents the challenge put forward by the thesis committee, the method chosen to respond to that challenge and the results of that approach.

Chapter 3

Adaptive Feature Mapping

The goal of feature relative navigation (FRN) is to locate and map features of the ocean environment without the use of an *a priori* map. The previous chapter has described how this problem lies at the intersection of previous research in navigation, mapping, and intelligent control. This chapter focuses on the intelligent control aspect of FRN, describing a technique for adaptive region mapping based on a new extension of layered control called adaptive layered control. This technique is inspired by state configured layered control (SCLC) as proposed by Bellingham [9].

To develop this technique, the following question was used as a case study: "How can an AUV find and map a trench in the Charles river (as shown in Figure 3-2)?" One of the central criticisms of layered control is that the missions performed to date have been relatively simple [28]. The task of adaptively mapping the Charles River trench provides enough complexity that the current approaches to layered control break down; thus motivating the development of adaptive layered control, which incorporates map representation and planning within the layered control paradigm. These extensions provide the capability to meet the challenge of finding the trench in the Charles River.

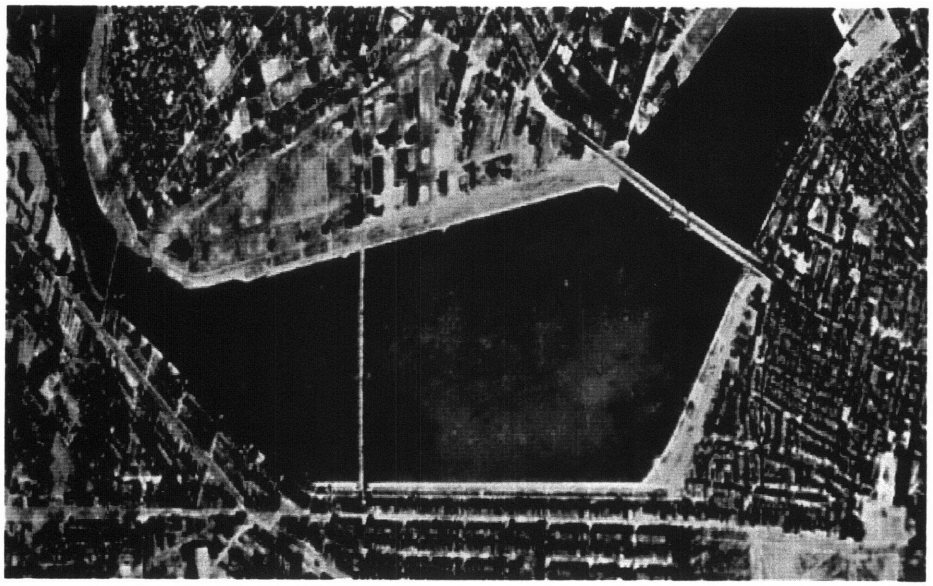


Figure 3-1: 1922 Aerial Photograph of the Charles River basin. The basin is framed by MIT at the top, Massachusetts Avenue to the left, the Longfellow Bridge to the right, and Storrow Drive at the bottom. Note that the esplanade was not constructed until 1930. (Photo courtesy of Massachusetts Historical Society).

3.1 The challenge

To investigate techniques for feature relative navigation, we consider the following challenge: how can an AUV locate and map an unknown number of features in an arbitrary location in the ocean with no *a priori* maps? To simplify this challenge, we make the following assumptions:

- Trenches are chosen as the feature of interest because of the static nature of bathymetric features (thereby facilitating repeatable tests). The trenches in the Charles River Basin are chosen due to the availability of appropriate bathymetric data and easy access for future field tests.

- A survey class vehicle, the AUV *Odyssey II*, is available as the sensing platform. The choice of vehicle class has a direct impact on the problem approach due to the nature of dynamic control of survey-class AUVs (i.e. no capacity for hovering and a finite turning radius).

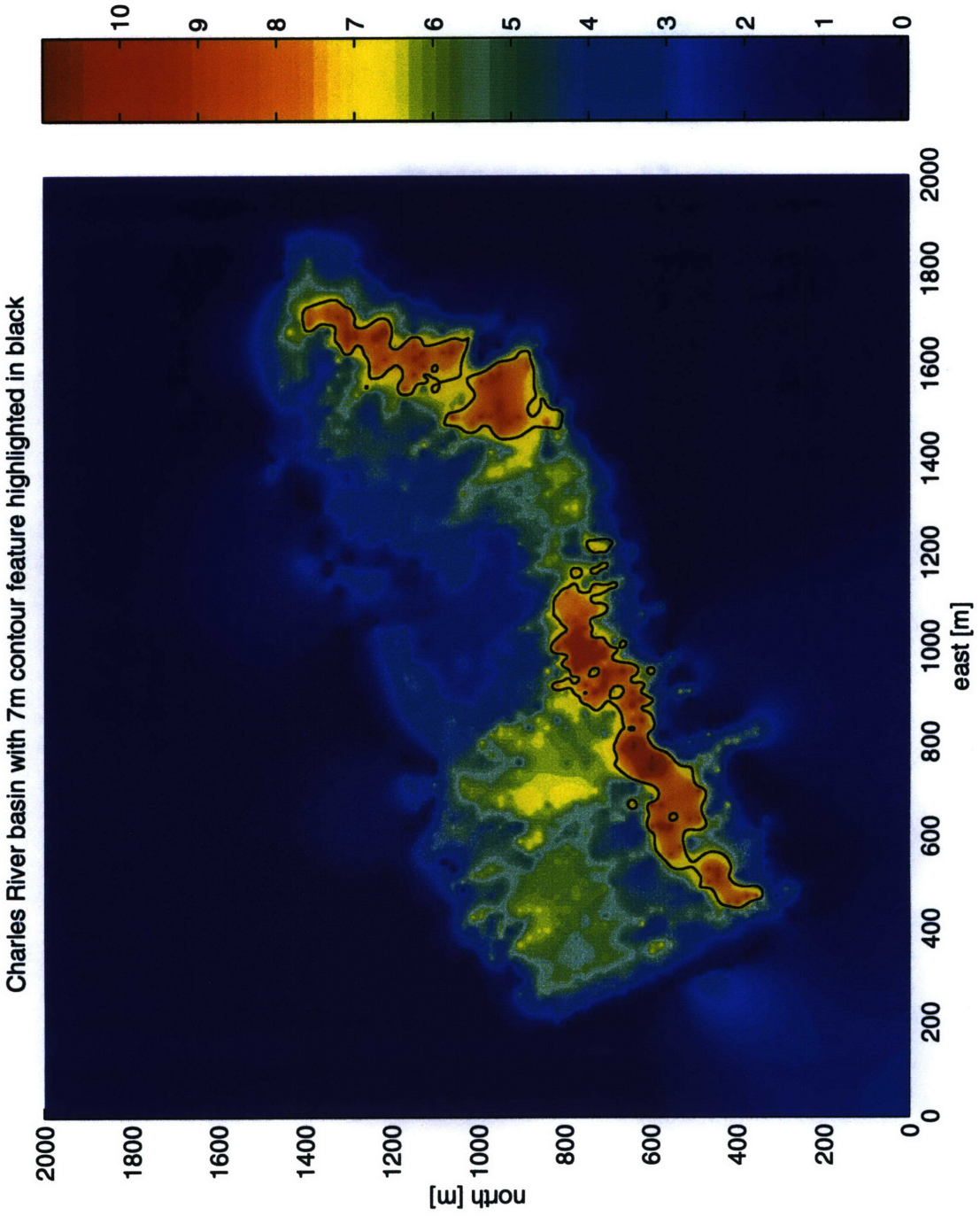


Figure 3-2: Charles River Basin with 7 meter contour emphasized.

- The AUV *Odyssey II* software environment is used [15] in anticipation of field testing at some future date.
- A sonar altimeter (rather than a swath-type sensor) is chosen as the primary sensor because it is both commonly available and because it can be viewed as a point sensor, thereby avoiding a solution which is unique to the peculiarities of the sonar sensing modality.
- Accurate navigation using an acoustic long baseline array and dead reckoning is available. This last assumption will be relaxed in Chapter 4.

The desirable aspects for adaptively mapping a feature in a pre-designated area such as the Charles River are good sampling of the feature, efficiently finding the feature, and quickly determining if the vehicle has located all features in the assigned region.

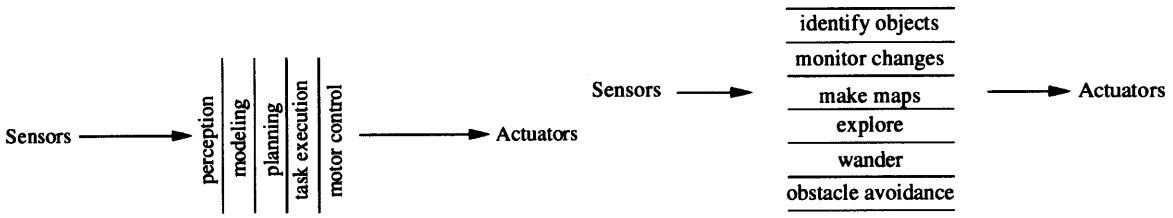
3.2 A behavior based approach

The form of intelligent control chosen was based on a *behavior based* intelligent control system, a form of *subsumption* architecture.

3.2.1 Subsumption architecture

As described in Chapter 2, the subsumption architecture is a form of intelligent control first proposed by Prof. Rodney Brooks of the MIT Artificial Intelligence Laboratory [22]. Like Brooks, we wish certain aspects of vehicle behavior to occur automatically, regardless of the actual situation the vehicle is in or what its current activities are. However, unlike Brooks, we also wish to be able to predict the vehicle's behavior as much as possible in order to prevent unanticipated and/or undesirable vehicle activity while performing its assigned mission.

Traditionally, a task is broken into *functional units*, each executed in turn after the other (Figure 3-3a). This form of decomposition assumes that each task is composed



(a) Traditional intelligent control (b) Behavior based intelligent control

Figure 3-3: Traditional functional decomposition of intelligent control for a mobile robot system vs. behavior based decomposition of a mobile robot [22].

of sub-tasks which in turn may or may not be further decomposed. The execution of these tasks is *sequential*, i.e. the next function using as its input the output of the previous function.

In the subsumption architecture, vehicle functionality is viewed as a series of *concurrent* task-achieving behaviors (see Figure 3-3b). These behaviors are each achieved separately and then tied together to form the robot control system. The advantages of this system are: (1) concurrent execution of multiple behaviors, (2) multiple goals (e.g. trying to achieve a point in space while avoiding obstacles), (3) expandability – new behaviors can be added onto existing layers, and (4) robustness – older, underlying layers maintain core competency beneath overlaying behaviors.

The process states that we first create a complete root control system, referred to as the *zeroth* layer, or level 0 competence. This is thoroughly debugged and tested. Once proven, we add another layer of competence onto this, called the first level competence. Level 1 can monitor the same input data as level 0 and can also monitor its output (see Figure 3-4). These layers are each finite state machines and execute continuously, issuing their own instructions to the actuators while at the same time inhibiting the output of those layers below them if necessary. The complexity of the overall behavior of the robot emerges from the interaction of these layers.

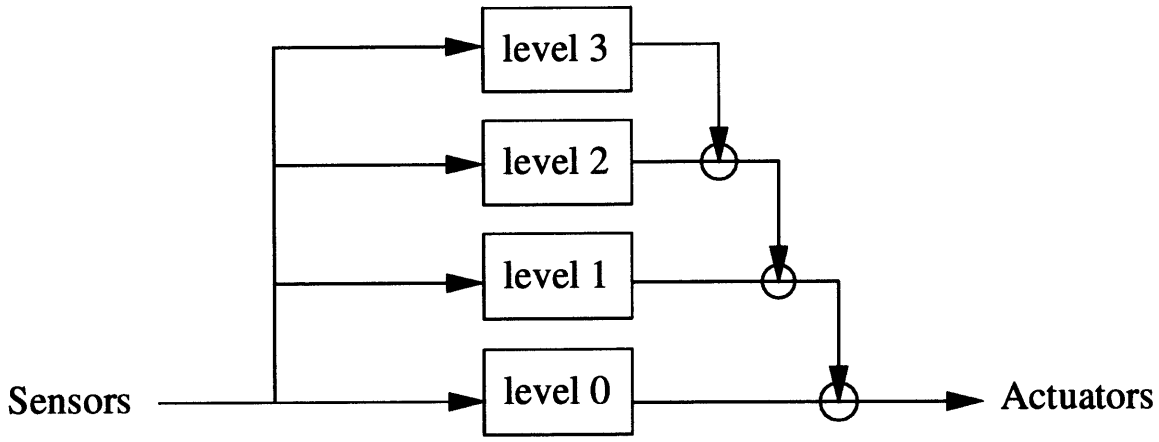


Figure 3-4: Subsumption is a series of competence layers, each of which can suppress the layer below it.

3.2.2 Criticisms of subsumption

In practice, this form of control quickly revealed fundamental problems associated with *scaling* and *situatedness*. The problem of scaling is a result of the number of interconnections, which in the worst case can increase as the factorial of the number of behaviors implemented on the vehicle. The resulting complexity of interactions was cited as a virtue of behavior based systems, since the goal was emergent intelligence and complexity. However, this same complexity was considered a shortcoming by the autonomous vehicle community for essentially the same reason: if all of the interactions could not be adequately modeled and predicted, the vehicle's behavior in a given situation also could not be modeled and predicted. This meant that a vehicle programmed with a purely subsumptive architecture could not be completely trusted to perform as predicted when confronted with a complex task.

The second shortcoming is commonly referred to as *situatedness* [18]. This is also a natural side effect of a subsumptive system. Since each behavior is a finite state machine, running concurrently with all the others, the current behavior suite is the only state the vehicle as a whole will function in. Such systems are also referred to as *non-taskable* systems. That is, the vehicle cannot be assigned a new task without reprogramming the whole system. In spite of these limitations, behavior-based control remains attractive because of the potential benefits of fast reaction

times and computational simplicity [10, 18, 77].

3.3 Layered control

In 1990 Bellingham and Consi proposed the concept of *layered control* as an adaptation of the subsumption architecture suitable for high level control of AUVs. Three primary contributions were made: (1) a methodology for keeping layered control simple [10], (2) an extension to accommodate mission planning [9], and (3) a validation of the approach via implementation on numerous AUV platforms [12].

Bellingham's layered control approach for AUVs grew out of an effort to address the problem of scaling. The principal difference between layered control and subsumptive architecture is the restriction of the interaction between layers. In a layered control system, each layer is assigned a relative priority number and executes without interacting with any other. Conflicting outputs are then resolved using a fixed prioritization scheme, with higher priority layers overriding lower ones. In general, mission safety functions (e.g. obstacle avoidance) have the highest priority. These non-interconnected layers are referred to as *behaviors*; the primary difference is that layers can interconnect, while behaviors cannot (see Figure 2-7).

In subsumption architecture, each layer can look into and influence those beneath it. Thus, if a newly added layer needs processed sensor information it can obtain that information from a lower layer via the interconnection of layers. Because a layered control system expressly prohibits interaction between behaviors, there is now the potential for redundant processing among behaviors when each behavior needs to process the same sensor information in the same way. To prevent this, sensor processing is pushed "outside" of the layered control system and pre-processed for all of the behaviors. Sensor data is then made available to all behaviors via a central data structure. Although this violates the premise of parallel execution and speed, sensor data processing is a critical prerequisite to all behaviors, and therefore is not truly an additional burden. Furthermore, performing the processing function as a separate task makes it possible to implement a distributed architecture. This allows

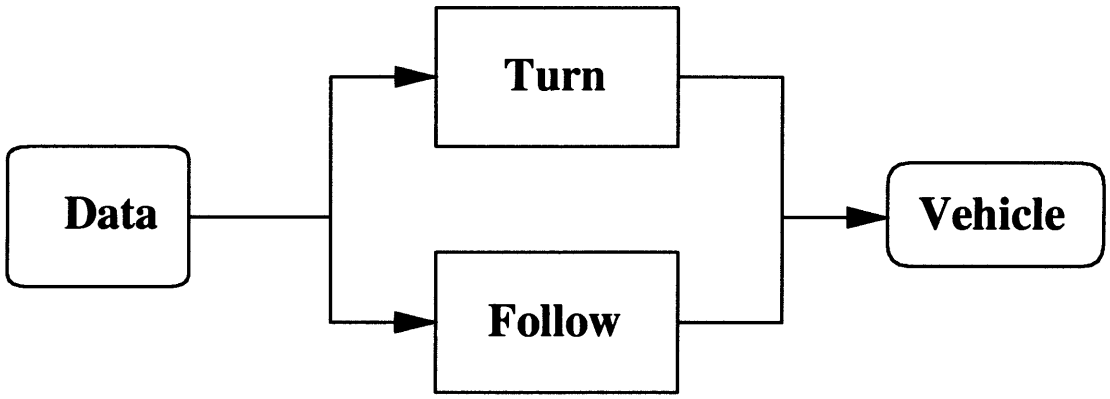


Figure 3-5: Implementation of simple layered control. In this implementation, there were two vehicle behaviors: Follow, which held a steady course over a feature, and Turn, which reversed vehicle heading, alternating left and right, in an attempt to reacquire the feature.

us to move the task of data processing and filtering onto separate processor(s) when available, thus reducing the burden on the main computer.

This approach has been field tested and proven robust [8, 16]. It has the primary advantages of a behavior-based intelligent controller (i.e. computational simplicity, speed, and quick reaction time) without the problems of interconnections and scaling found with a purely subsumptive system.

3.4 Implementation 1: a simple reactive system

Returning to the challenge of mapping the Charles River trench, we now describe our first method of implementation of the adaptive feature mapper, utilizing layered control. Computational efficiency and speed of execution were considered primary design criteria and drove the design towards simplicity. Assuming accurate navigation was available, the approach used two behaviors, follow and turn (see Figure 3-5. The follow behavior held a steady vehicle heading as long as a feature was being detected, while the turn behavior reversed course, alternating to the left and to the right, whenever a feature was passed.

- Deploy - Go to the survey site, consisting of the following behaviors:

  `waypoint` - controls vehicle transit to survey location.

`power_monitor` - monitors vehicle power consumption and aborts mission if power levels too low.

- Search - Search for bathymetric features, with the following active behaviors:

  `trench_finder` - identify trench-like features and reverse course if feature is passed.

  `power_monitor` - monitor vehicle power consumption and aborts mission if power levels too low.

- Recovery - Prepare for recovery.

  `recover` - shuts off thruster and pitches vehicle up for a coast to surface.

The key new behavior created for this configuration was the `trench_finder`, which was designed to monitor the sonar altimeter and depth sensor data, reconstruct the water column at the current location, and identify trench-like features in the bathymetry stream. Two methods of identification were chosen: depth triggered and slope triggered. For the purposes of this test, the 7 meter contour of the Charles River was chosen as the depth triggered target feature, while slopes in excess of 10 degrees were chosen for the slope trigger (see Figure 3-2).

Figure 3-6 was a simple reactive approach triggered by the 7 meter depth contour. The behavior of the system was quite promising, showing that a simple reactive feature finder was sufficient under the correct conditions. However, if the vehicle did not choose the correct direction at the outset, there would never have been an initial contact and therefore no reaction and subsequent sampling in the first place. Figure 3-7, a slope-triggered mission, demonstrates a major pitfall for any reactive behavior-based system. In this mission, the vehicle encountered the feature at $N, E = (600, 800)$ and reacted by turning back into the feature. Soon after it detected the feature again and turned once more. On the second turn, however, it missed the feature entirely and continued along until it reached the edge of the search region.

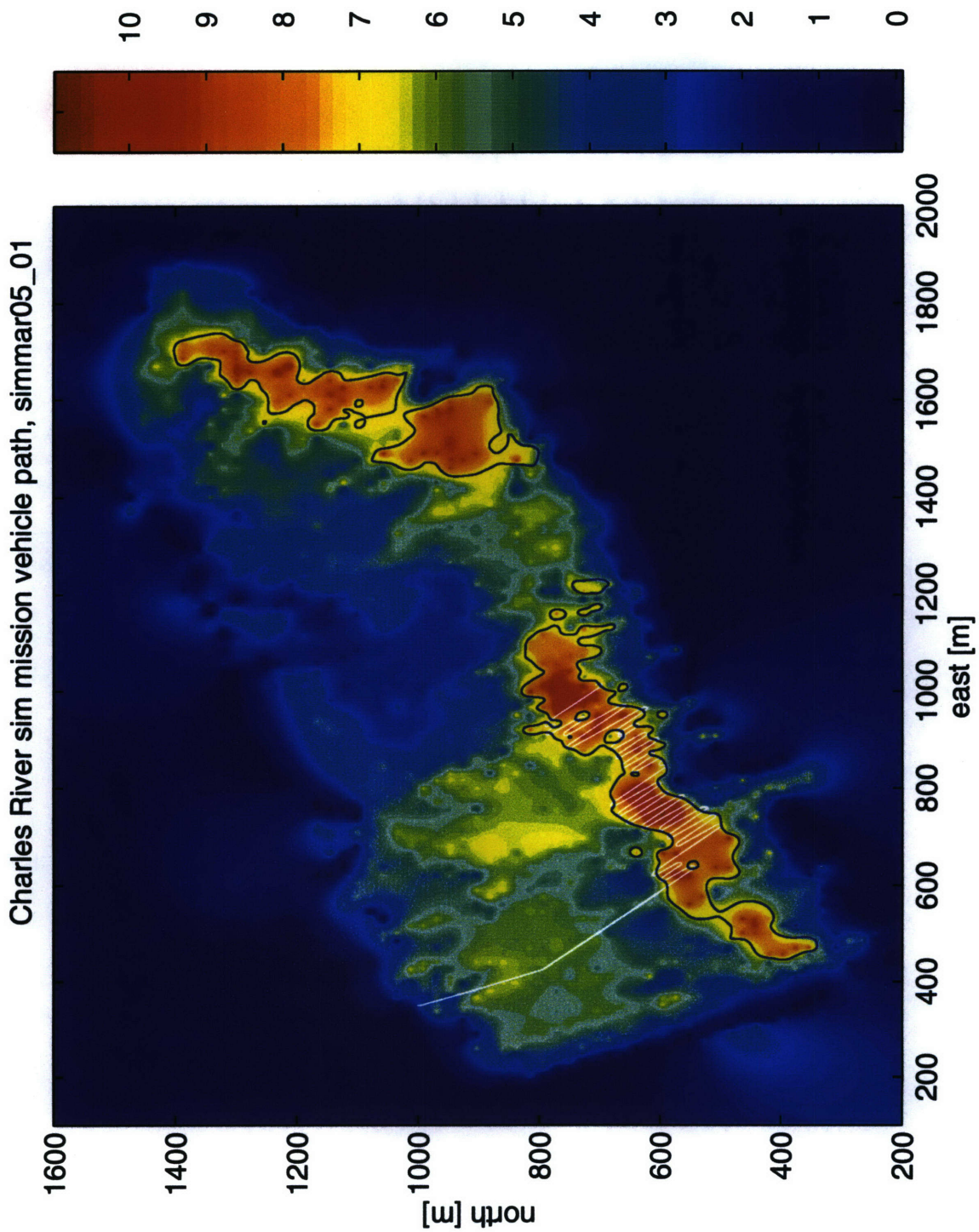


Figure 3-6: March 05 two-phase, three-state mission. Phase 1 is a simple **waypoint** behavior. Phase 2 is the **trench_finder** behavior, which is a three-state behavior: **search**, **follow** and **turn**. The behaviors were simple reactive: reverse course whenever a feature is passed.

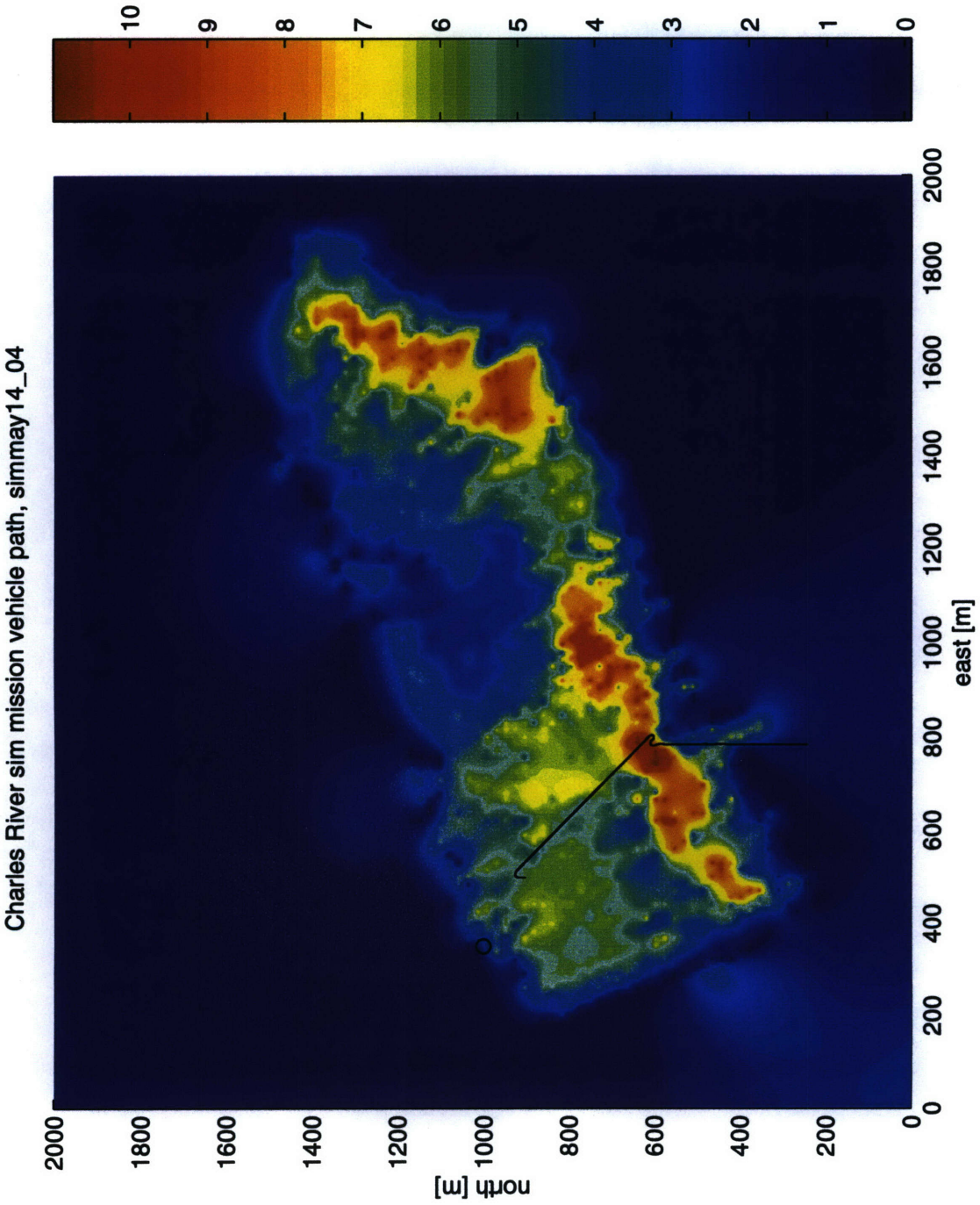


Figure 3-7: May 14 simple reactive mission. In this case, `trench_finder` was set to trigger off of slope. The behavior was fooled because it re-approached the trench edge at a shallow slope angle relative to the vehicle path. The vehicle continued off of the map as it attempted to reacquire the feature.

3.5 State configured layered control

While layered control solved the problems of scaling it did not address situatedness. To make a behavior-based intelligent controller *taskable*, it needs the capability of handling multi-phase missions. In 1991 Bellingham and Consi introduced State Configured Layered Control (SCLC) to address this issue.

> The difficult aspect of implementing the architecture defined above [layered control] comes not from the individual behavior definitions, that are relatively simple, but rather from the coordination of the goal oriented behaviors... It is up to the user to ensure that the behaviors take control at the appropriate phase of the mission, and relinquish control when their function has been served [9].

These observations motivated Bellingham to propose the concept of state-configured layered control (SCLC).

SCLC allows an autonomous vehicle to perform multi-phase missions without human intervention. The key concepts to SCLC are the *state table* and the ability to turn behaviors on or off as required. During the course of a mission, a state table dictates the rules for transition from one state to another (see Figure 3-8). At each state, necessary behaviors are executing while unnecessary ones are idle, thereby both reducing the computational burden and allowing for multiple vehicle activities which would otherwise conflict (e.g. obstacle avoidance and rendezvous & docking). Missions are programmed into the vehicle via a *mission script* which contains all possible states and transition rules between those states.

Figures 3-10 and 3-9 show a five-phase bottom following mission from 1993 using the in the *Odyssey I* NetSim simulation environment, employing SCLC. In this mission, a hypothetical photographic mapping operation has been designed with a launch phase, a pick up phase and three mission phases (two mapping and one transit). In this particular case the recovery operation is presumed to be on a beach. The state table has been preprogrammed with four states and associated behavior groups:

- Deploy - Go to the survey site, consisting of the following behaviors:

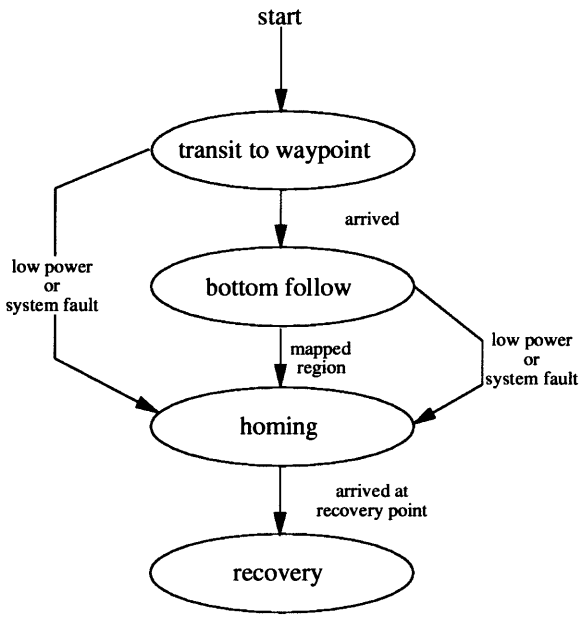


Figure 3-8: Sample states and transitions for SCLC.

`waypoint` - controls vehicle transit to a survey location.

`obstacle_avoidance` - prevents collision.

`power_monitor` - monitors vehicle power consumption and aborts mission if power levels too low.

- Mapping Transit #1 - Move to first waypoint while mapping/photographing on the way.

  `waypoint` - controls vehicle transit to a new location. May be repeated for multiple legs of a multi-phase survey.

  `bottom_follow` - maintain constant altitude over seabed.

  `obstacle_avoidance` - prevents collision.

  `power_monitor` - monitors vehicle power consumption and aborts mission if power levels too low.

- Transit #2 - Move to next mapping area.

  `waypoint` - controls vehicle transit to a new location using new assigned depth.

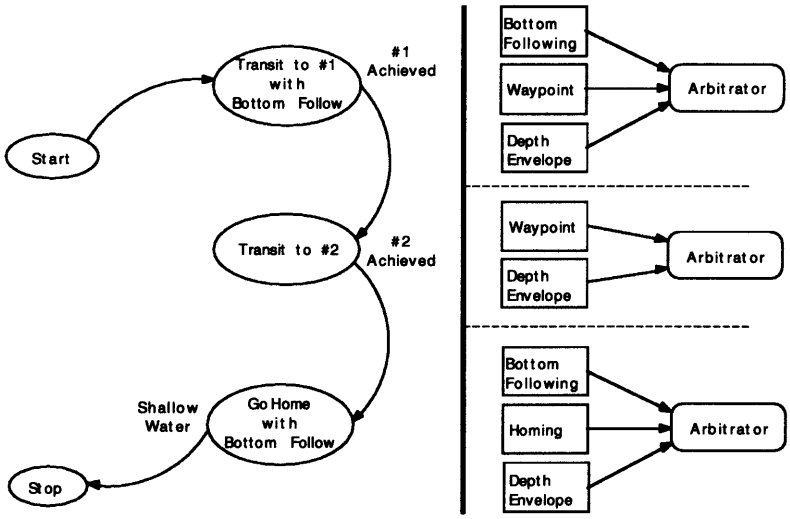


Figure 3-9: Multi-phase bottom following mission states diagram. Here the main mission phases are displayed. The first mapping phase consists of the behaviors `bottom_following`, `waypoint` and `depth_envelope`. The second mapping phase uses `waypoint` and `depth_envelope` and the final mapping phase reactivates `bottom_following` with a new altitude, `homing`, and continues to use `depth_envelope`. In all phases the safety behaviors of `power_monitor` and `obstacle_avoidance` are functional.

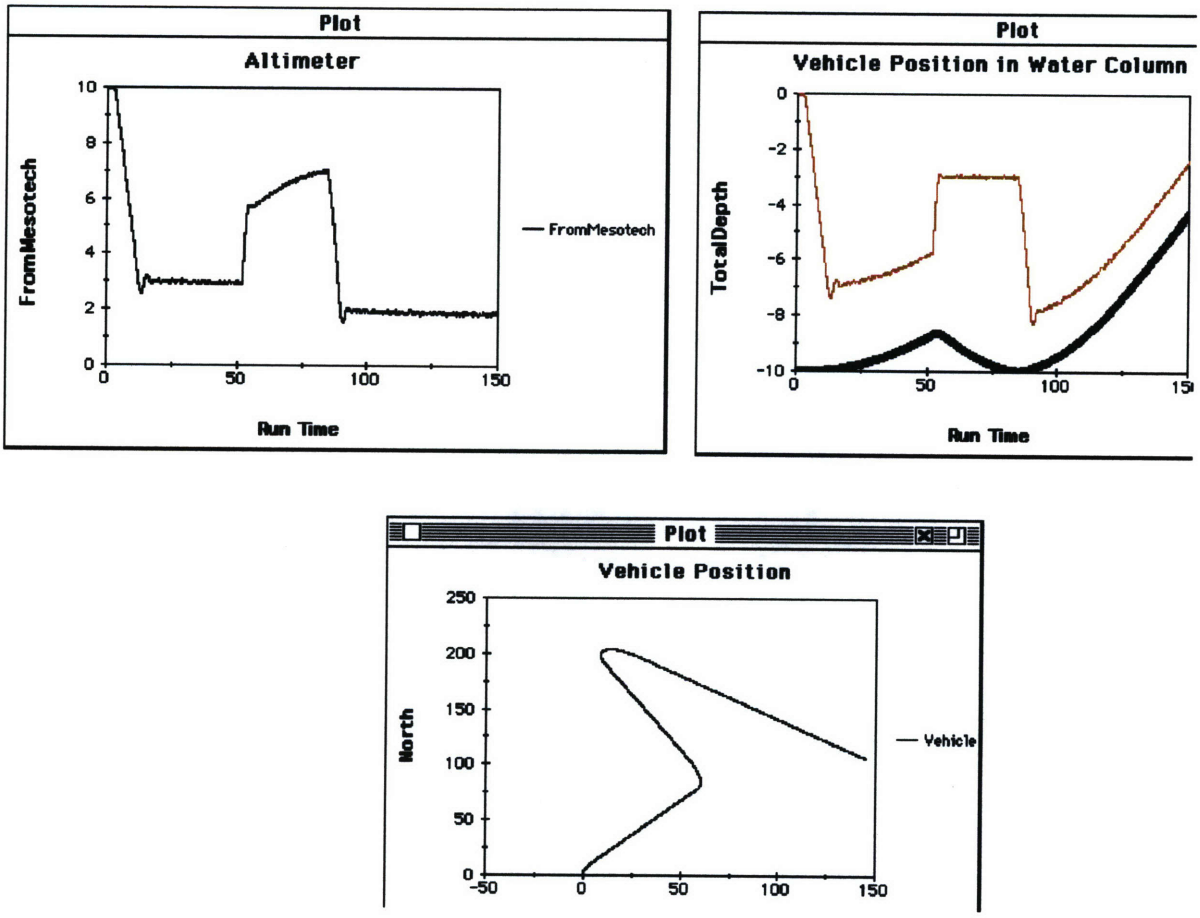


Figure 3-10: Multi-phase bottom following mission vehicle track. In phase 1 the vehicle is launched. In phase 2 it descends to a constant altitude while traversing to a prespecified waypoint. Upon arrival, phase 3 directs the vehicle to a new waypoint while maintaining a constant depth. Phase 4 again specifies a (different) constant altitude while the vehicle is commanded to head towards a homing beacon set in shallow water. Phase 5 is the recovery.

`obstacle_avoidance` - prevents collision.

`power_monitor` - monitors vehicle power consumption and aborts mission if power levels too low.

- Mapping Transit/Return - Transit to pick-up location while mapping bottom.

  `waypoint` - controls vehicle transit to a pickup location.

  `bottom_follow` - maintain constant altitude over seabed.

  `obstacle_avoidance` - prevents collision.

  `power_monitor` - monitors vehicle power consumption and stops vehicle if power levels too low.

- Recovery - Prepare for recovery by operators.

  `rendezvous_&_docking` - maneuver vehicle into capture frame.

  `power_monitor` - monitors vehicle power consumption and stops vehicle if power levels too low.

The key elements to note are that in each phase of the mission, some behaviors are switched on, some switched off, and others (`waypoint` and `bottom_follow`) have new parameters assigned to them appropriate to the next mission phase. Note also that the safety behavior `obstacle_avoidance` must be disabled during rendezvous & docking operations or else the vehicle will never enter the docking frame. Finally, the safety behavior `power_monitor` changes from prematurely aborting the mission to simply shutting the vehicle off when the vehicle is nearing its assigned pickup point.

3.6 Implementation 2: adding search

As shown in Figure 3-7, if the vehicle using a simple reactive approach loses contact with the feature of interest, it is liable to head off in any direction. Therefore, an improved method of locating and tracking features is called for – in particular the ability to search for initial features and to reacquire a feature if that feature is lost.

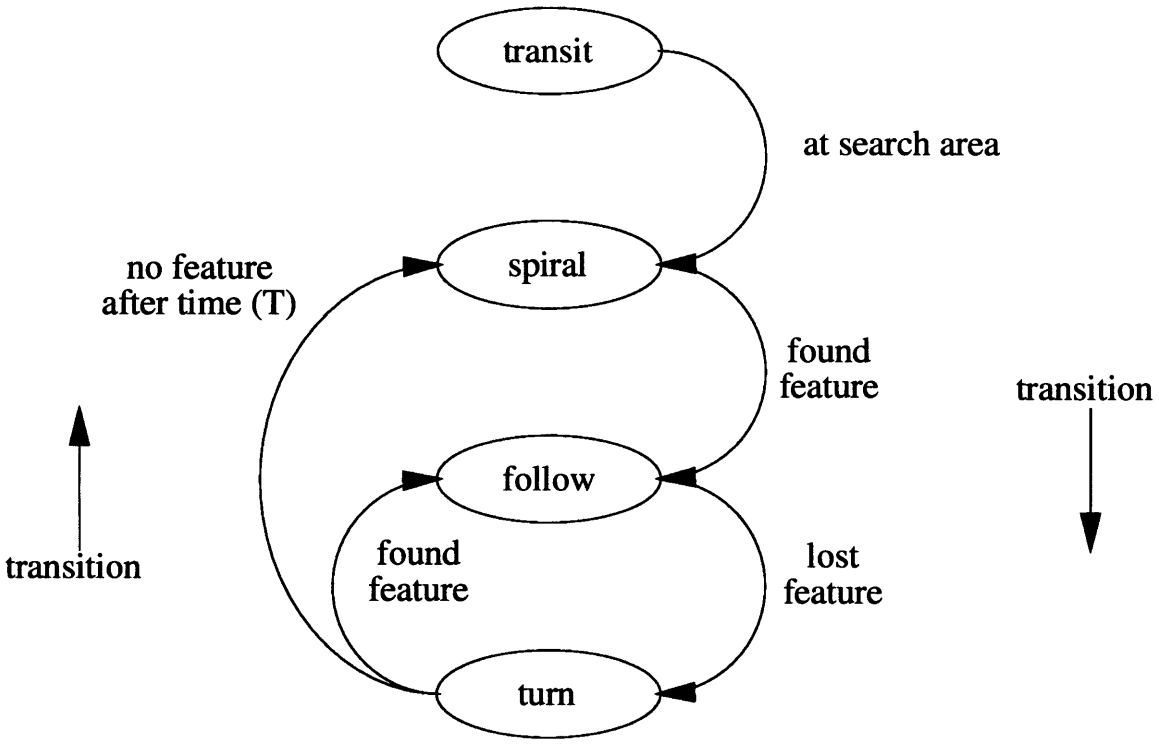


Figure 3-11: Functional states of the improved `trench_finder`.

The shortcomings of the slope-based and depth-based reactive approaches were addressed by combining these two approaches into one *recognizer*. This recognizer-based behavior functions in a similar fashion to the simple reactive system except that now, if the feature is not immediately reacquired, the vehicle begins a search in an attempt to locate more of the feature of interest. The operating states of this new recognizer-based implementation are shown in Figure 3-11.

To improve the initial search strategy and to generate an efficient search pattern, the initial straight-line search behavior was replaced with a spiral search pattern using the Archimedean spiral trajectory

$$r = \alpha\phi \tag{3.1}$$

which, unlike a logarithmic spiral, is a spiral with a constant interval of $2\pi a$ between each lap (see Figure 3-12). There are two principal reasons for choosing such a spiral over the more conventional "lawnmower" approach (see Figure 3-13). The first is because of the linear nature of the lawnmower survey and that of some trenches (especially man-made trenches such as those created in dredging); it is conceivable that

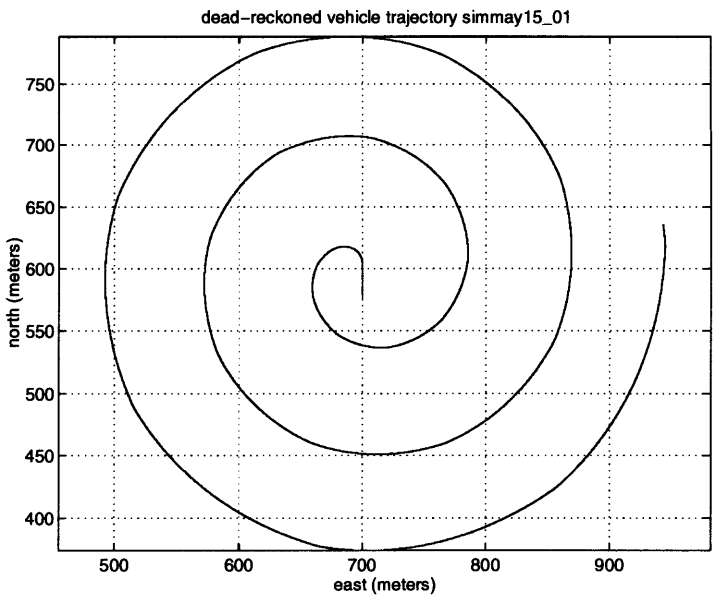


Figure 3-12: Archimedean spiral search pattern.

under the right conditions and given a sufficiently large search interval a lawnmower-type search may straddle a feature without detecting it (see Figure 3-13). In contrast, a spiral approaches the search area at a continually changing path angle, thereby decreasing the likelihood that a distributed feature could go unnoticed.

Second and more importantly, for a given search interval α a spiral trajectory covers a search area using a shorter path, thereby saving energy with a more efficient search. Given that we wish to map an area A, the lawnmower search will cover the area

$$A = XY \tag{3.2}$$

with a path length of

$$S = X + Y\frac{X}{2\pi\alpha}, \tag{3.3}$$

while the Archimedean spiral covers the same area

$$A = \pi r^2 = \pi\alpha^2\phi^2 \tag{3.4}$$

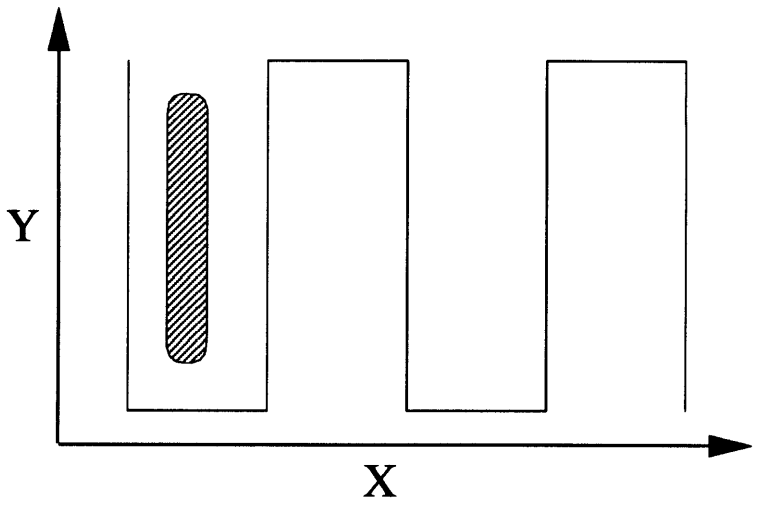


Figure 3-13: A typical "lawnmower" survey pattern. Note that it is possible to miss a linear feature under the right conditions. This can be corrected by either re-surveying at a right angle to the initial survey or by decreasing the interval between survey legs.

with a path length of

$$S = a\int_0^{p1}\sqrt{(1+\phi^2)}d\phi = \frac{a}{2}\phi_{p1}\sqrt{(1+\phi_{p1}^2)} + sinh^{-1}\phi_{p1}, \tag{3.5}$$

(for large values of ϕ,

$$S \approx \frac{\alpha}{2}\phi^2). \tag{3.6}$$

Comparing the **area covered per unit distance traveled** yields, for the lawnmower survey

$$\frac{A}{S} = \frac{XY}{X(1+\frac{Y}{2\pi\alpha})} = 2\pi\alpha\left(\frac{Y}{2\pi\alpha+Y}\right), \tag{3.7}$$

and for the spiral

$$\frac{A}{S} = \frac{\pi\alpha^2\phi^2}{\frac{\alpha}{2}\phi^2} = 2\pi\alpha. \tag{3.8}$$

Given the fact that Y is a search area axis and α is our search interval, we can see that for any combination of α and Y such that α and Y are both nonzero (true for any real world search situation),

$$2\pi\alpha \;>\; 2\pi\alpha\left(\frac{Y}{\alpha+Y}\right). \tag{3.9}$$

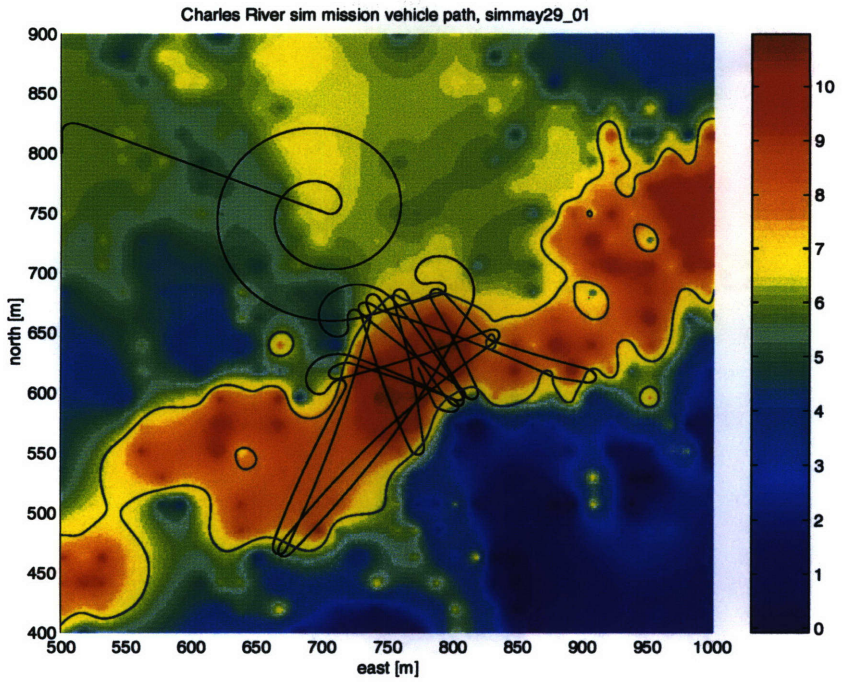


Figure 3-14: Improved `trench_finder` with spiral search directed feature reacquisition.

Hence a spiral search pattern is always more efficient than a lawnmower pattern for a given area. Note that in the limiting case where $Y >> \alpha$ the two search patterns approach the same relative efficiency.

The results of this improved `trench_finder` are shown in Figure 3-14. In this mission, the vehicle succeeds in maintaining contact with the feature throughout the life of the mission.

These improvements resulted in a robust, reactively-driven trench finder with the ability to locate, track, and stay "within" a feature. However, as shown in Figure 3-14, the reactive nature of the system resulted in oversampling of the feature in some areas while undersampling in others. This is because as a purely reactive system the trench finder has no way of differentiating newly discovered portions from those already visited.

3.7 Extending SCLC

In a pure state configured layered control system, the states are pre-defined before the mission. This can lead to two key problems: 1) contingency planning and 2) adaptability. Mataric observes that the solution to the former is to develop a system whereby "reactive, constant-time run-time strategies can be derived from a planner, by computing all possible plans off-line beforehand [77]." If the planner is sufficiently thorough, then all possible states can be anticipated, and therefore the necessary state will be available when required. This, however, leaves the problem of adaptability. The vehicle mission shown in Figure 3-14 has sufficient states to address the basic problem at hand, i.e. mapping the trench, but it is dependent on the random changes in direction brought about by the behavior's reactive nature to move the vehicle into unmapped parts of the feature. This is inefficient and prone to possible failure if conditions are such that portions of the feature are never reached. A superior method would be for the behavior of the vehicle to *adapt* to the feature to insure that all of it is adequately mapped.

A sufficiently complex state table may be capable of performing this mission, but it would have to be tailored to each feature and, given the assumption that there is no *a priori* information available, would not be feasible under the constraints we have placed upon the survey. What is required is a behavior which does not merely change its output according to some pre-determined mapping function, but instead changes itself in response to current and past sensory information, i.e. a behavior which learns and adapts to the environment. This learning may take place within the behavior or as a separate function. The output is then the result of *planning* on the part of the behavior, based upon the current information and any predictions the planner may make.

To accomplish this in the framework of the existing AUV layered control system, the *adaptive behavior* was developed. Adaptive behaviors are behaviors which are capable of adapting their output to the current environment based on information obtained during the course of a mission. Unlike a more traditional behavior, an

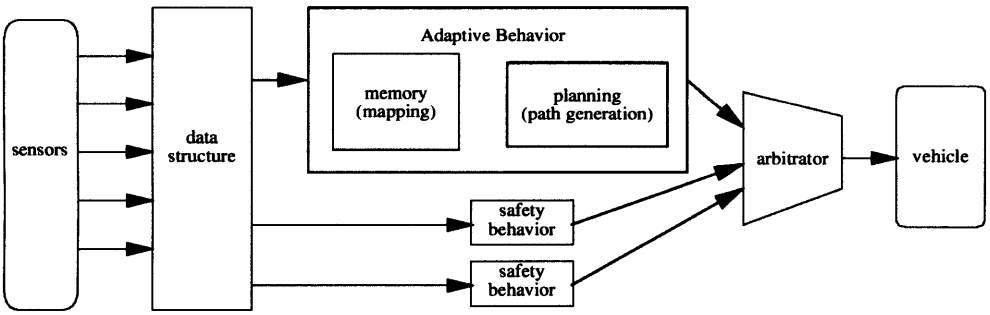


Figure 3-15: Schematic of an adaptive behavior incorporated into a layered control system.

adaptive behavior has (or has access to) memory, a capacity for planning and can contain multiple states internally (see Figure 3-15). This capacity allows the behavior to alter its output in response to the current situation while taking mission history into account.

Adaptive behaviors are also, by design, encapsulated. This means that their internal states and interactions are known in advance of their use, making them predictable in the field. By doing this, a user can employ sophisticated "canned" vehicle competencies – with the knowledge of how they will function – without the need to model multiple behavior interactions in the field. In keeping with the layered control paradigm, the adaptive behavior is assigned a priority lower (less important) than any safety behaviors that may be employed during the mission (e.g. `power_monitor`, `obstacle_avoidance`, etc.). The output is restricted to vehicle heading, depth and speed, while inputs are restricted to a few mission-specific global parameters.

As mentioned above, the memory of the adaptive behavior may or may not be internal to the behavior itself (depending on whether we wish to share the information with other behaviors). In either case, for the purposes of adaptive feature mapping we will employ mission-specific memory in the form of a map of the exploration area which is constructed and maintained during the course of the mission.

3.7.1 Mapping

To properly sample a feature, we need to insure that we explore as much of the feature as possible in the time allotted. As shown in the previous section, while a purely reactive system can be made robust, it lacks the ability to deliberately search out unknown regions. To do this requires some form of memory ("I have been here before").

To this end a *mapper* was designed and implemented. The function of the mapper is to construct and maintain a data structure, or map, which represents the history of vehicle activities related to the feature-relative search aspects of the mission. To be useful to an autonomous vehicle, the map (data structure) must meet certain criteria:

- The map should accommodate different types of features extracted from a broad range of sensor types. The map must be extensible so that new sensors can be easily integrated into the mapping framework.

- The implemented data structure should be easy to search efficiently. This allows for easy modification and updating and allows for the eventual use of prediction based on current data.

- The data structure should be amenable to the use of multiple representations of features. This allows for the possibility of map-based navigation using multi-sensor modality (see Tuohy [111]) as well as the simultaneous use of both area-based representations and feature-based representations.

- The map should include the ability to partition information. In particular we wish to be able to differentiate between *feature* vs. *non-feature* and between *known* vs. *unknown* regions.

These requirements can be satisfied with a form of *grid-based* representation. A grid-based map has the advantages of simplicity and flexibility. A grid-based map is easy to search and manipulate because matrix-based representations are natural choices for implementation on computer systems; optimized search and sort routines for n-dimensional matrices are commonplace. A grid is flexible because the nature

of the individual cells of a matrix-based representation is not fixed and can therefore represent anything required, even including pointers to more extensive datasets.

The particular form of grid-based map we have employed is constructed in the form of a multi-dimensional matrix. Thus, an $[m \times n \times D]$ matrix can represent D different $[m \times n]$ maps of a given region. Our matrix has at least $D = 2 \times s$ dimensions, where s is the number of sensor modalities employed. For each sensor, one layer of D represents the *feature map* while the second represents the *visitaton map.*

The *feature map* is the representation of the feature of interest as perceived by the associated sensor type. It partitions the world into two categories: *feature* or *no feature.* It makes no assumptions about whether the vehicle (or sensor) has visited a given area or not other than maintaining an estimate of the sensor readings in a cell determined to be *feature.* In the case of a bathymetric feature, the *feature map* maintains the average depth in each *feature* cell.

The *visitation map* tracks all locations in the search area in which the associated sensor has been employed. In the case of a point sensor such as temperature, this corresponds to the vehicle location at the time of sensing. When the sensor can be remotely employed (e.g. a scanning sonar), this will correspond to all areas where the sensor has been active, which may or may not include the vehicle's actual track. Thus, a horizontally employed conical sonar beam would (in the case of a two dimensional visitation map) project into a fan-shaped structure overlaid onto the visitation map.

The visitation map partitions the world into *visited* regions and *unvisited* regions. Just as with the feature map, *unvisited* cells are empty while *visited* cells contain relevant information about that region. Note that the set of *feature* cells is a wholly contained subset of the set of all *visited* cells.

If the feature is distributed in a three dimensional region (a thermal plume, for example) then each map becomes an array of voxels rather than pixels. Extending the map is a question of increasing the dimensionality D of the array, adding an extra layer to account for each layer of voxels. Extensions of the same multi-dimensional array search and update methods can still be employed, although they become much more complex and time-consuming [122].

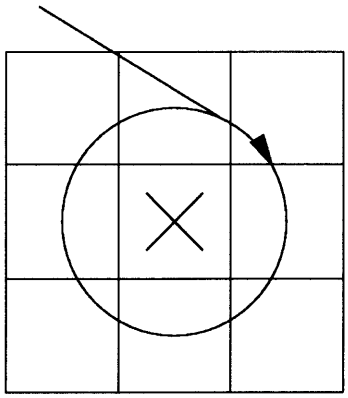


Figure 3-16: Trapping occurs in a simple trajectory generator when the target cell is smaller than the AUV's turning radius. The solution is a more intelligent trajectory generation and/or monitoring of vehicle progress.

The values of m and n are dictated by the dimensions of the search area and the chosen cell size. The cell size is dictated by both the dimensions of the features being sought and the spatial resolution of the sensor modality. In the case of a point sensor mounted on a dynamically-controlled vehicle such as the AUV *Odyssey*, the turning radius of the vehicle will influence effective sensor "spatial resolution." It is desirable that the size of the feature be much larger than the vehicle's turning radius. In most cases, the lower bound of cell size will be dictated by the dynamic capabilities of the sensor platform.

In the case of our bathymetric feature mapper, the matrix is $m \times n \times 2$ where the two maps are the bathymetric feature map and the bathymetric visitation map. Because we are employing a sonar altimeter in search of trench-like features, our visitation map is updated by vehicle location via the navigator. The feature map logs all locations which the vehicle has visited and which meet the feature detector's criteria for candidate trench-like features. The cell counts m and n are dictated by the chosen survey area which have a characteristic cell size of 10×10 meters. This size is generally set to be in excess twice the turning radius of the AUV to prevent the problem of "trapping," which can occur in a simple trajectory generator if the vehicle attempts to sample a small cell via the shortest path (see Figure 3-16). The solution to this potential pitfall is the implementation of a more intelligent trajectory generation scheme and/or monitoring of vehicle progress to catch such situations.

3.7.2 Waypoint generation

With a map available, the AUV now has a sense of history. If it knows where it has been, then it can also *plan* where to go next. Using the map as it is created allows the `trench_finder` to make decisions about where more of a feature is likely to be found, thereby improving the overall efficiency of the mapping operation. If all of a feature has been found, `trench_finder` can then seek out more features in the assigned area. These two functions are referred to as "locating more feature" and "seeking new features."

The `trench_finder` is designed to give preference to locating more of a known feature based on those elements of a feature that have already been located. It does so by creating a set of candidate cells $\{C\}$, which are those cells which are adjacent to feature cells $\{F\}$ and not members of the visitation cell set $\{\bar{V}\}$. This is performed via map superposition (see Figure 3-18). This set of candidate cells $\{C\}$ are then ranked according to distance and orientation relative to the AUV, with preference given to those cells in the immediate path. In its present form, `trench_finder` ranks these according to a previously generated lookup table (see Figure 3-17). Each time the adjacency map is updated the list of candidate cells is also updated and re-ranked according to the current vehicle location and orientation.

If no more of a known feature is available, then `trench_finder` will seek out new features. This occurs when the list of candidate cells is empty ($[C] = \emptyset$). Seeking new features is performed by first setting the `waypoint` sub-behavior to proceed to the center of the largest unmapped region. Upon arrival, the planner disables `waypoint` and instructs the `search` behavior to execute a spiral search pattern until either a) a new feature is found, or b) the spiral covers the area. If the vehicle completes its spiral search without locating any new features, then it will proceed to the largest remaining unmapped area in the search zone and repeat the process.

15	23	10	18	14
19	7	2	6	22
12	4		3	11
24	8	1	5	17
16	20	9	21	13

Figure 3-17: Sample pre-generated waypoint lookup table. In this particular table, no special preference is given to any particular direction. Other lookup tables may be biased towards one particular direction, resulting in the vehicle working its way from waypoint to waypoint along that direction.

3.8 Implementation 3: `trench_finder`

Our third implementation of adaptive region mapping, called the `trench_finder` behavior, adds mapping and planning to SCLC as discussed above. The adaptive behavior `trench_finder` is incorporated into the AUV *Odyssey* layered control structure as shown in Figure 3-15. There are two on-board maps, the visitation map and the feature map. The visitation map is updated by the current vehicle data structure while the feature map is validated by the `trench_finder` behavior.

The internal construction of the adaptive behavior `trench_finder` can be thought of as consisting of a supervisory planning state and eight internal states, many of which alter their internal settings in response to the mission data and to the overall mission goals (see Figure 3-20). Only one internal state is active at any time, with the transition between states triggered according to the current vehicle location and status of the on-board map. The states are:

`initialize` - Execution of `trench_finder` begins by reading in the global mission parameters from the mission script. These parameters include:

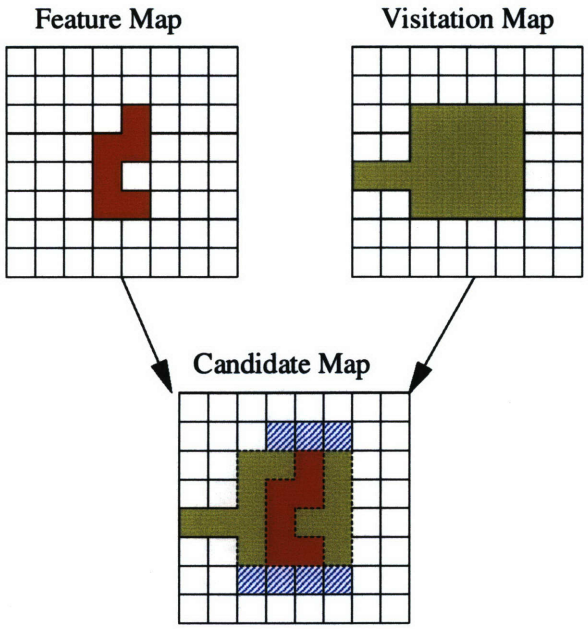


Figure 3-18: Waypoint candidate generation process. The *feature map* on the upper left tracks all known candidate features. The *visitation map* on the upper right tracks all places where the vehicle has been. Candidate waypoints are determined by examining the intersection of the two maps. bBsed on the assumption that natural features are continuous, candidate waypoints are defined as those areas which are adjacent to or near a known feature or features, but which have not yet been visited by the vehicle. This list of candidate waypoints is then ranked, as shown in Figure 3-19.

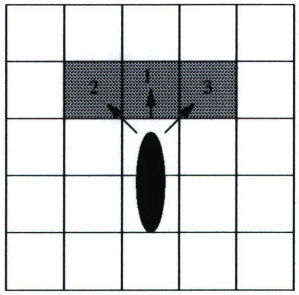


Figure 3-19: Waypoints are visited by the vehicle in a sequence determined by a relative ranking of the candidate waypoints. This ranking can be based on a lookup table (see Figure 3-17) or based on a function of distance and location relative to the current vehicle location and orientation.

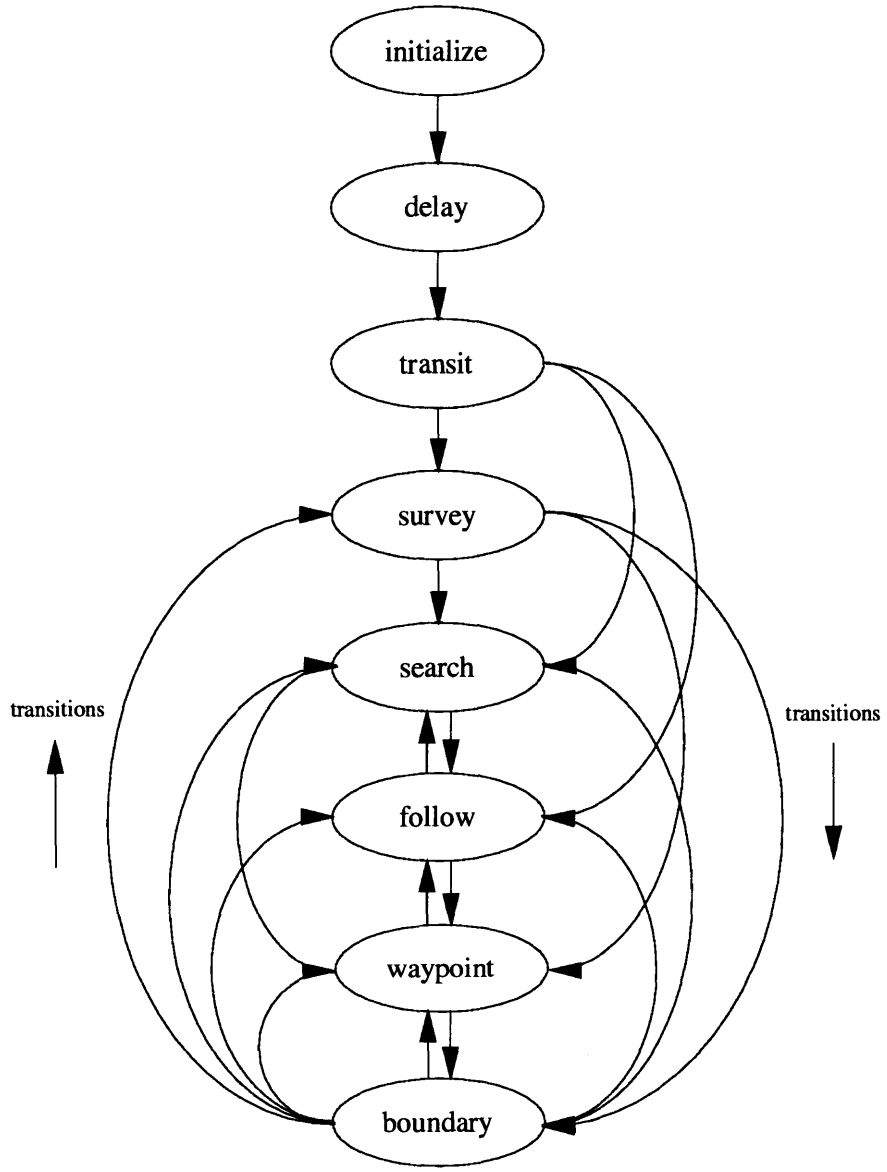


Figure 3-20: Internal states of the `trench_finder` behavior.

north_lim, south_lim, east_lim, west_lim - Boundaries of the search area, defined by the user in meters. The vehicle origin is set by the navigation system (e.g. long baseline, dead reckoning).

search_mode - The type of search mode to be used by the vehicle when attempting to locate features in the search zone. Modes available are spiral, lawnmower, wagon wheel (i.e. radiating out from a point such as a hole in the ice), rectangular, or a simple reactive search (i.e. set out at a random heading until one edge of the search zone is reached).

search_center_north, search_center_east - If the user wishes, a starting point of the search may be explicitly defined.

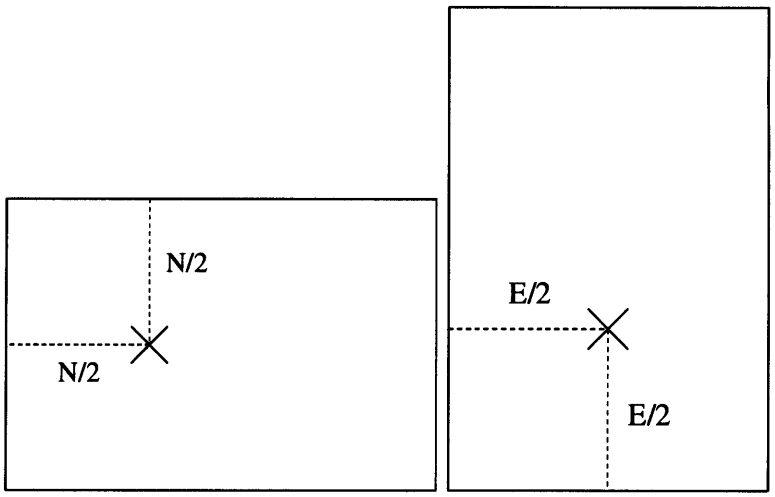
search_interval - The interval between legs of the search. Rectangular searches use this interval directly while a spiral search uses 2π times this interval.

detect_thresh - If the user has any *a priori* information about the feature such as a threshold value, it is entered here.

detect_type - `trench_finder` can trigger off of slope, depth, or a combination of the two. Depth can be defined as a specific threshold or a range of values.

`delay` - The behavior waits for sensor filters and settings to initialize and stabilize before attempting to read and process the data structure.

`transit` - Transit causes `trench_finder` to behave like a simple waypoint routine. Its function is to steer the vehicle into the survey area from the launch point. If search_center coordinates have been specified, transit will take the vehicle to that location, breaking off prematurely if a potential feature is discovered and the vehicle is within the search zone. If no search coordinates have been specified, then transit will take the vehicle to a point which is equidistant from the narrower of the boundaries and that same distance from the western or southern end (see Figure 3-21).



(a) East-West search zone

(b) North-South search zone

Figure 3-21: Initial search starting point determination. The starting point is set to maximize the initial search efficiency in the even that features are few or non-existent.

survey - If desired, a coarse survey of the search zone may be conducted using either the spiral or lawnmower survey patterns. This initial survey is intended to seed the waypoint state with candidate initial search sites. The course survey can be set by the user to be at a search interval of 10 to 100 times the standard search interval.

search - The search state is the default **trench_finder** state whenever there are no current waypoints available. A search is performed at the start of the mission and at any point in the mission when no more candidate feature locations are currently available from the **waypoint** state. The search type may be in the form of a traditional lawnmower search, a simple rectangular box pattern, an Archimedean spiral search (the default search state), or a wagon wheel search which consists of multiple legs radiating out from a central starting point (e.g. a hole in ice cover, a recharging station, or a communications buoy).

follow - When a previously unmapped feature element has been detected **trench_finder** switches to the follow state. This state instructs the vehicle to hold course until the feature is passed. Once the feature has been passed,

the follow state switches to the waypoint state to acquire more of the new feature.

`waypoint` - Waypoint is the primary active `trench_finder` state. This state examines a list of candidate feature locations and chooses where to direct the vehicle to go to next. Waypoint transitions to the follow state when a feature has been detected. If no further candidate waypoints are currently available, execution is passed back to the search state.

`boundary` - Boundary is a `trench_finder`-specific safety state. If the vehicle passes outside of the search zone in the course of tracking a sensed feature or searching for a new feature, waypoint will direct the vehicle back into the search zone by setting a waypoint location centered in the search field. Transition from this state occurs immediately upon the vehicle's re-entry into the search area.

All state transitions are governed by the `planner` supervisory state which monitors the current vehicle state, location, and orientation. This information is combined with the on-board maps to determine which state is appropriate at the moment. The general sequence of state priorities is boundary (if the vehicle is outside the search area), waypoint (if waypoint generation results in a nonzero waypoint list), search (if no waypoints are currently available), and follow (if a feature is detected).

3.9 Results

Samples of the performance of the improved `trench_finder` are shown in Figures 3-22 through 3-31. Figures 3-22 through 3-26 are of a mission conducted on November 20, 1996. Figures 3-27 through 3-31 are from a May 22, 1997 simulation of a different part of the Charles River trench.

Figure 3-22 shows the vehicle track from the perspective of the entire basin. The vehicle launch point was moved near the trench to simplify the examination of the trench finding phase of the mission. In this mission the vehicle can be seen to locate and map the entire main trench. In Figure 3-23 the vehicle path can be seen to make

occasional excursions to what appear to be outlying locations. This is due to the vehicle running out of "nearby" candidate locations and so it returns to less likely candidate waypoints.

Figure 3-24 shows the vehicle's feature map. The "*" symbols denote the cells which contain candidate features. Figure 3-25 is the visitation map of all cells which the vehicle has sensed – in this case (with a downward looking altimeter) the vehicle has also physically visited these cells. Finally, Figure 3-26 shows the union of the two maps. The mission was halted before all candidate cells had been exhausted and a new spiral search was begun. Note the unsearched cells (candidate waypoints) at 650 North by 840 East and all along the feature cells at 1140 East. We can see from Figure 3-23 that there is more of the feature of interest (the 7 m contour) just to the east, which we would expect to be found had the mission continued.

In Figures 3-27 through 3-31 the vehicle was launched south of the northeastern trench. In this case the vehicle started the mission with a preliminary spiral survey until it located a portion of the target feature. At that point, the vehicle began to map the trench as before. In Figure 3-28 the vehicle path can be seen in more detail. Note that the northeast trench is composed of two pits with a small saddle in between. In this mission the vehicle has migrated from the first trench to the second during the survey. Given sufficient time, we expect that it would have completed its survey of the second part of the trench and returned to the first. Figures 3-29, 3-30, and 3-31 are the *feature map*, *visitation map*, and the combination of the two (as used by the waypoint planner), respectively.

3.10 Related research

Two of the most important questions that have been asked in the field of behavior-based AI (BBAI) are: 1) how to design behaviors? and 2) how can a behavior-based approach be extended beyond simple reactive tasks (such as obstacle avoidance) to more complex problems?

Bellingham and Consi [9] espouse the concept of state configured layered control

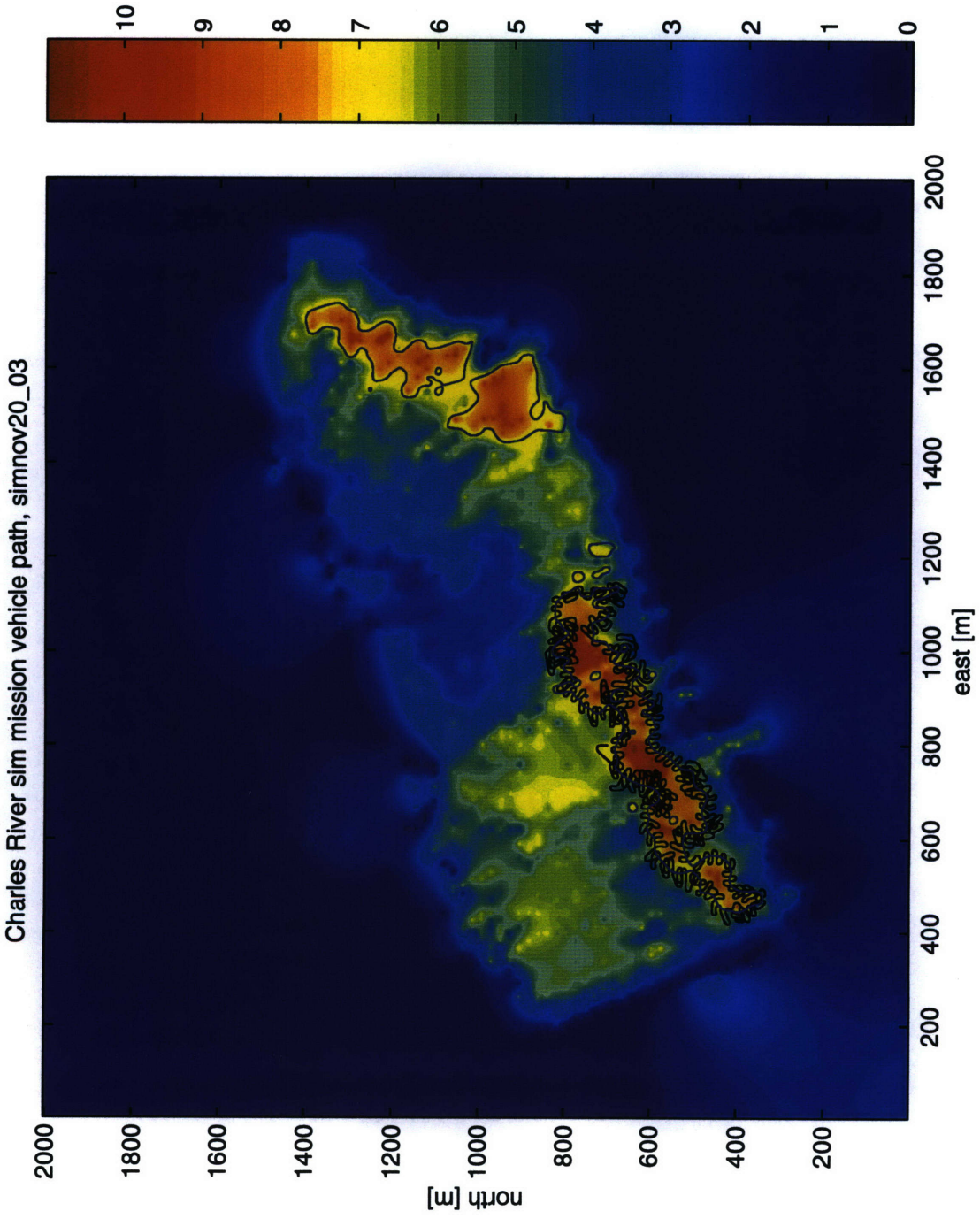


Figure 3-22: Vehicle path from November 20 simulator mission. Vehicle is started near feature to save transit time, but has no *a priori* information about the feature.

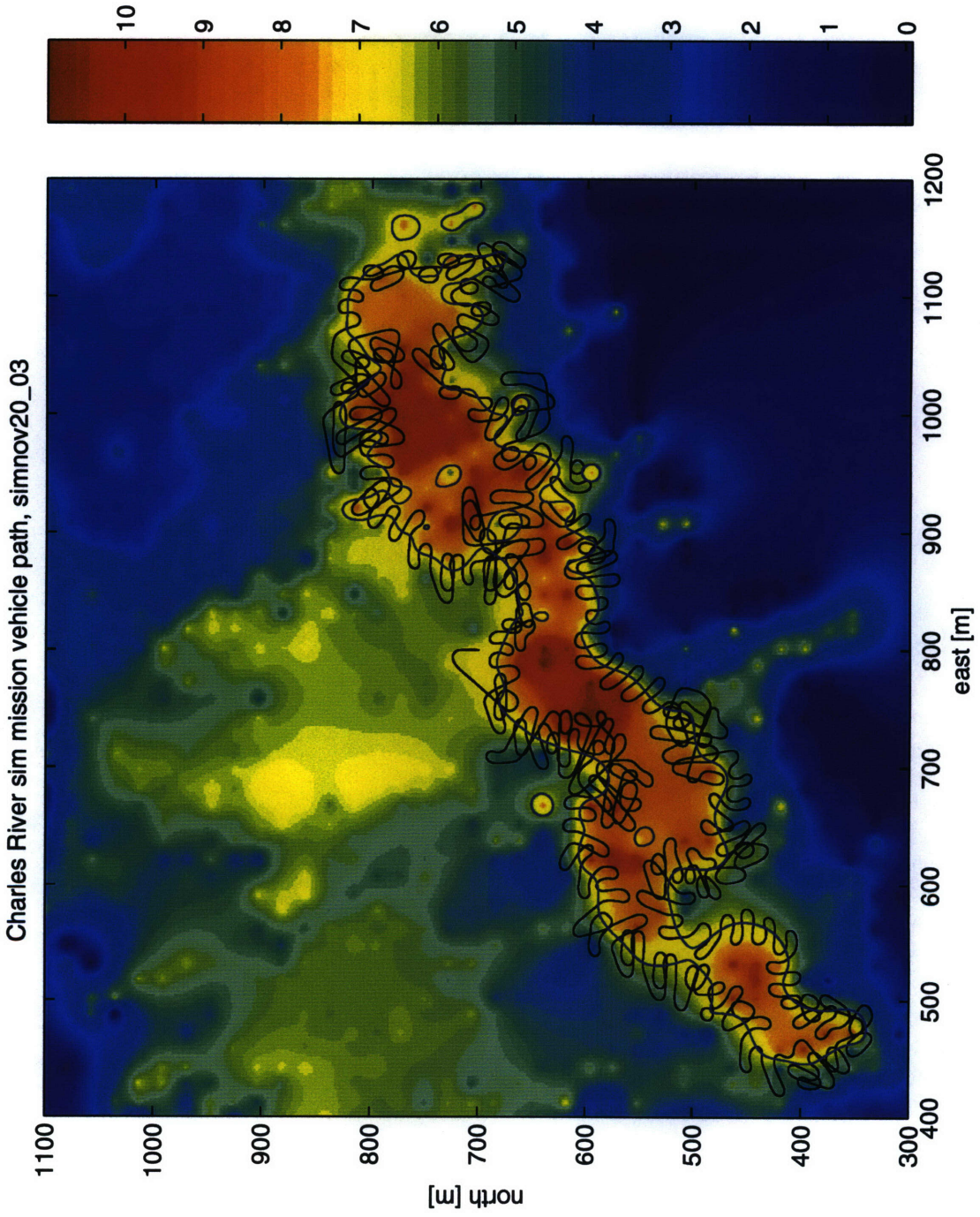


Figure 3-23: Close up of vehicle path from November 20 simulator mission.

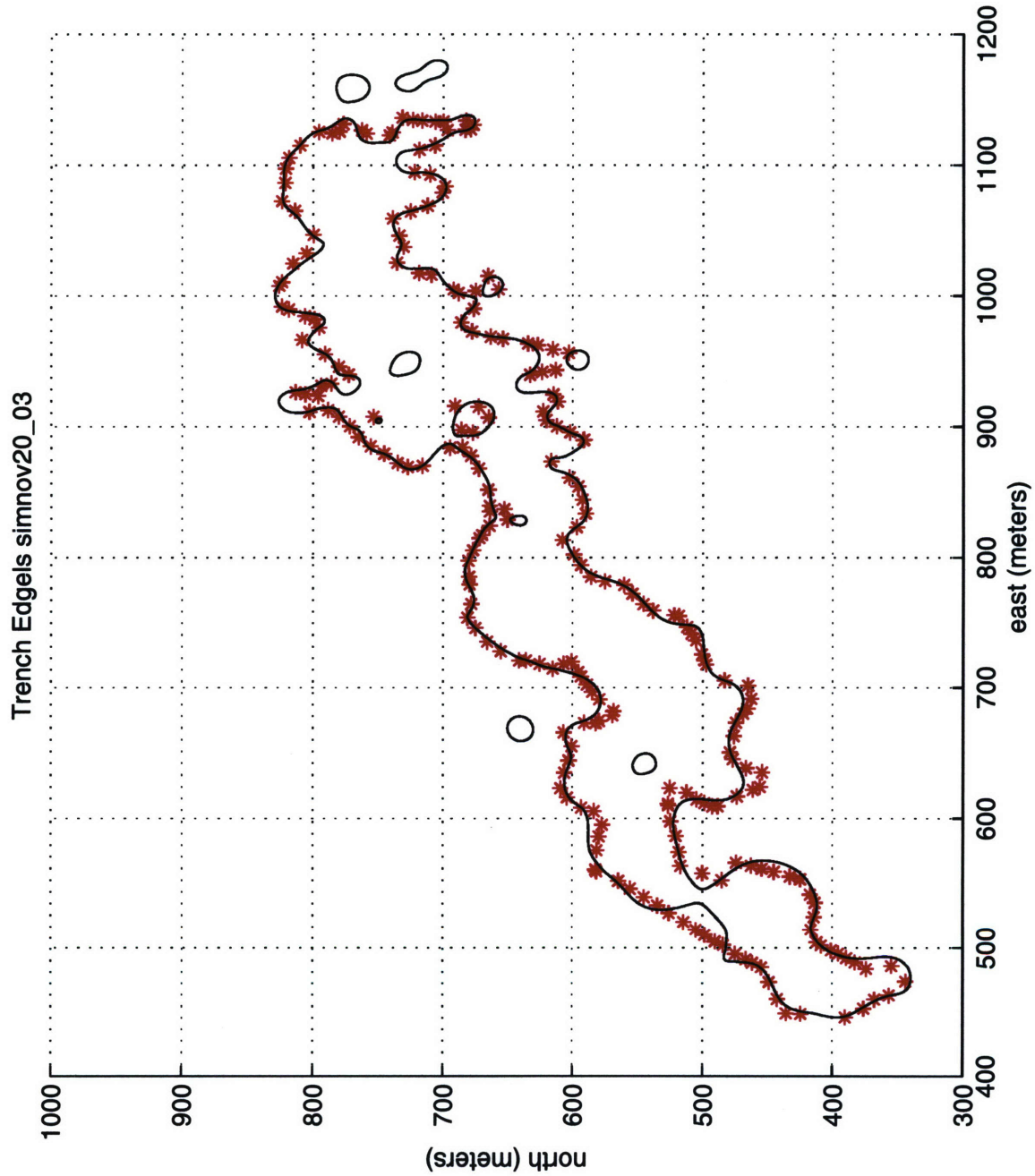


Figure 3-24: Feature map generated by `trench_finder` on November 20 simulator mission.

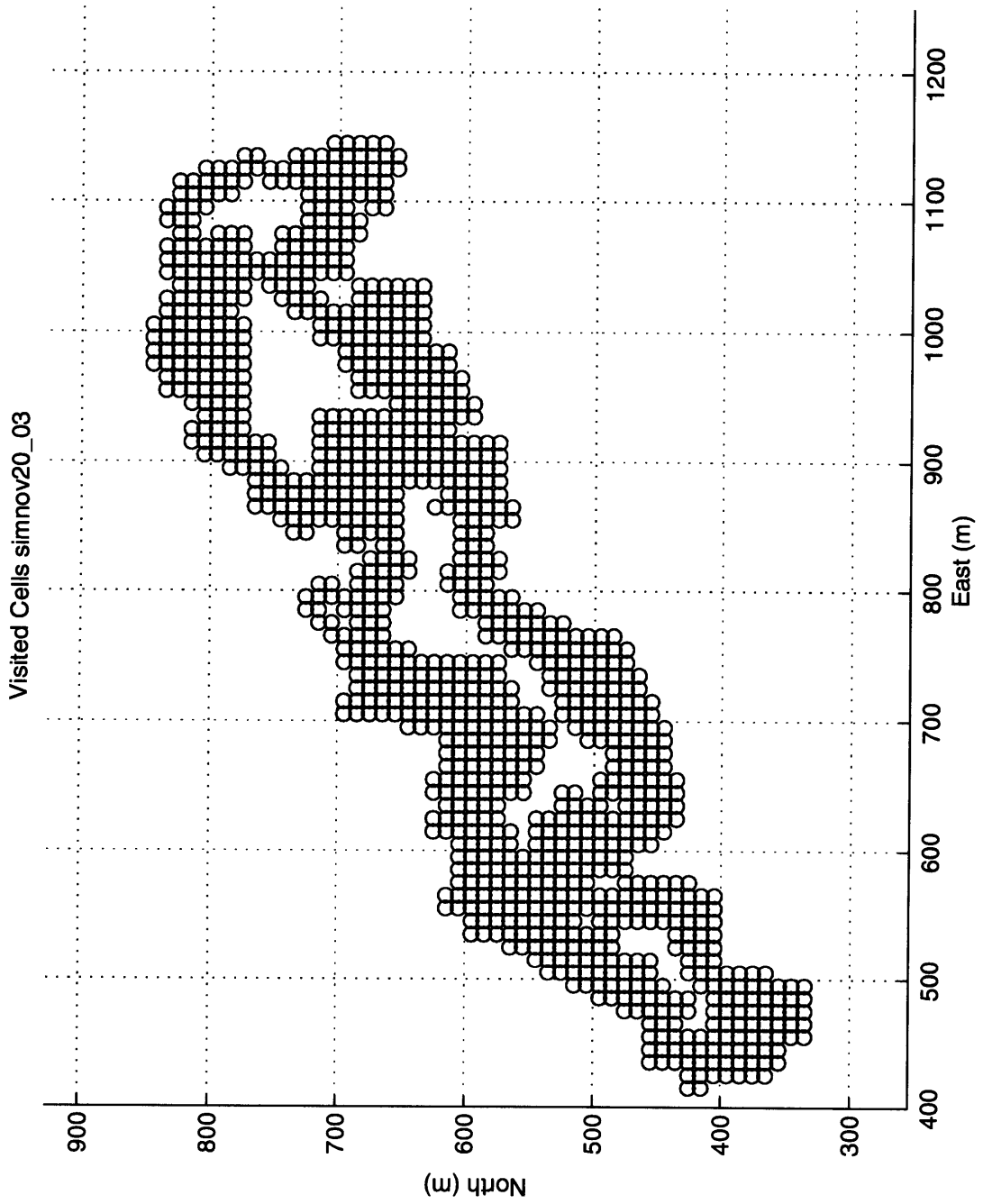


Figure 3-25: Visitation map generated by `trench_finder` on November 20 simulator mission.

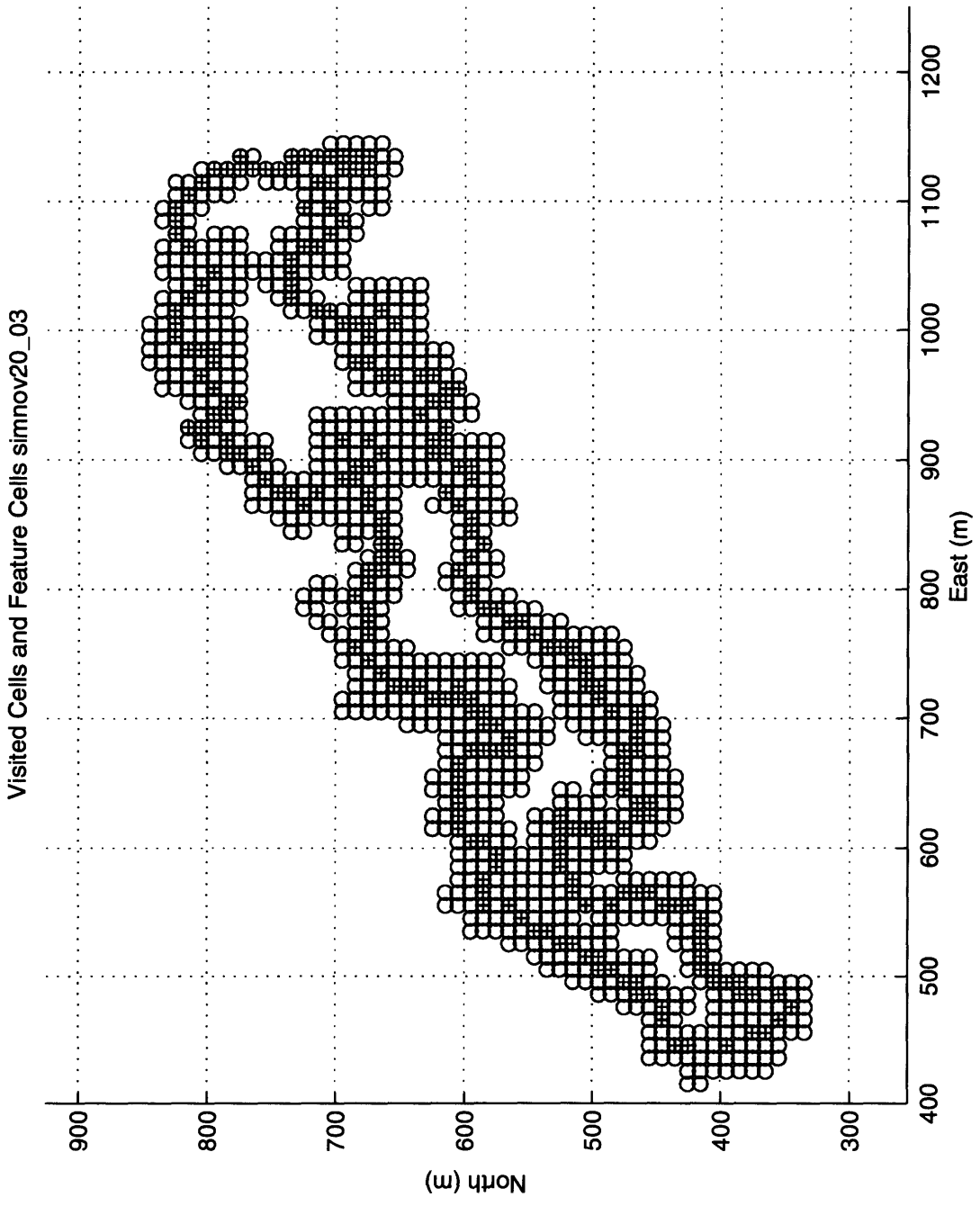


Figure 3-26: Feature map ("+") overlaid onto the visitation map ("o") generated by `trench_finder` on November 20 simulator mission. Note the unvisited candidate adjacency cells along the eastern edge of the mapped feature.

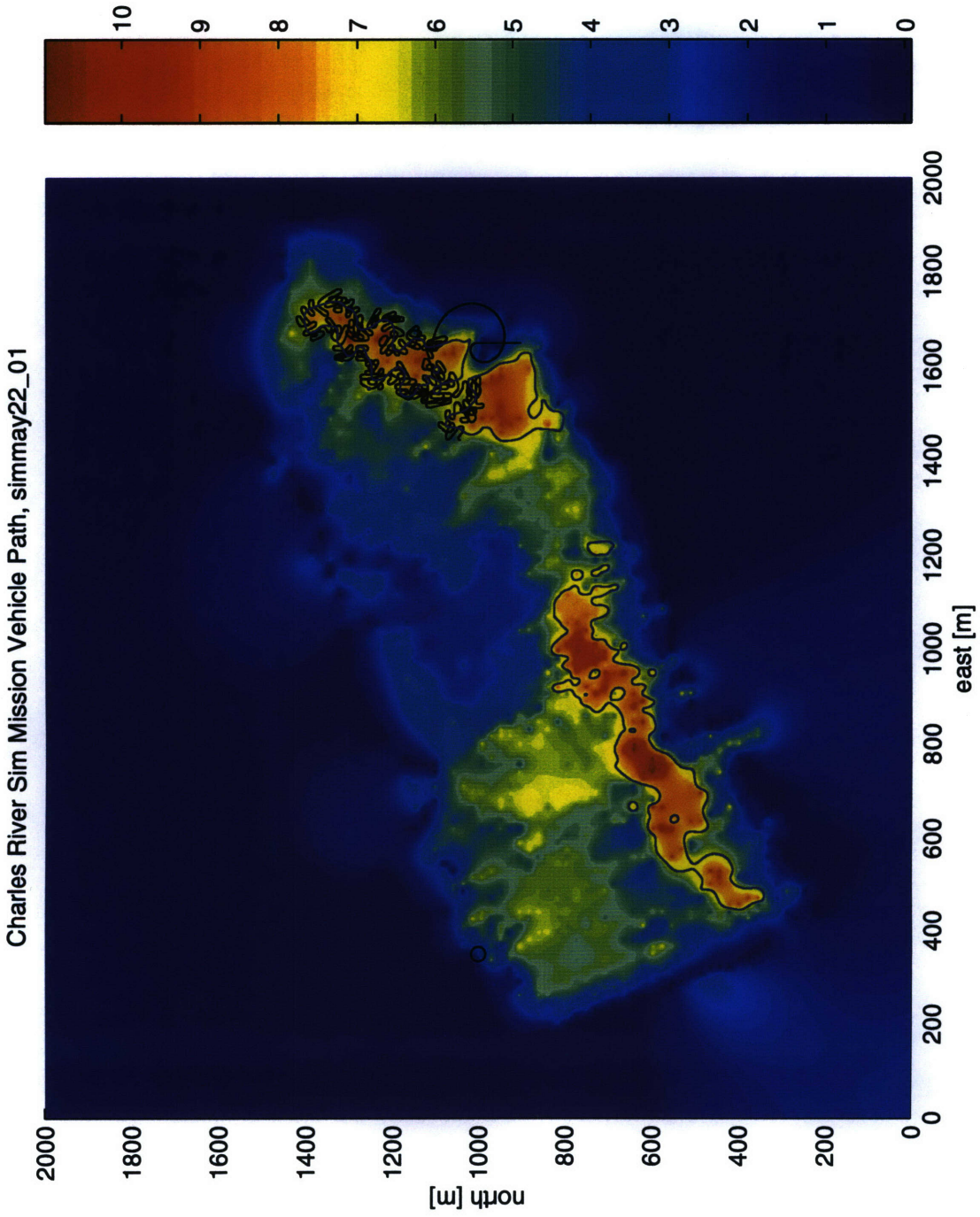


Figure 3-27: Vehicle path from May 22 simulator mission.

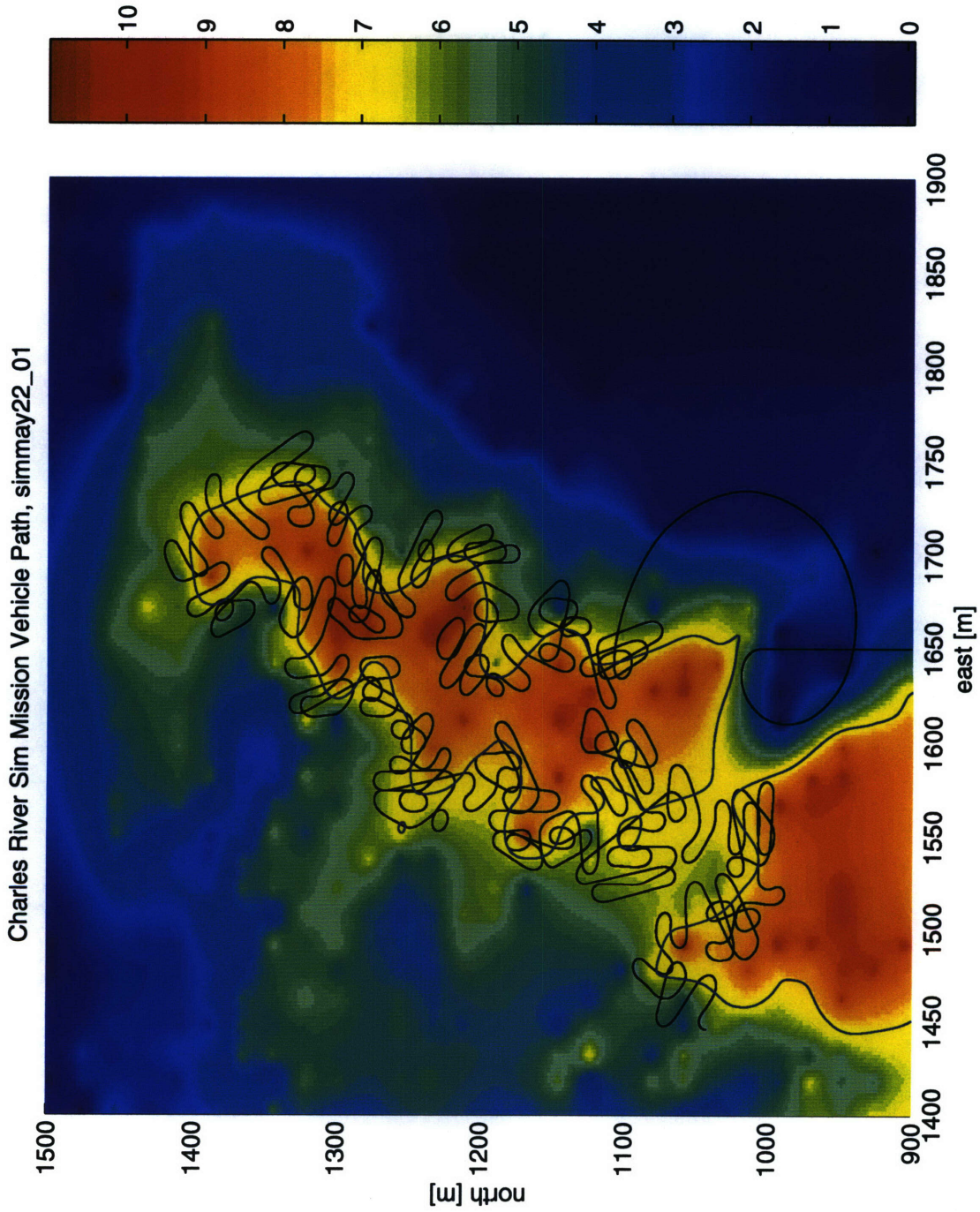


Figure 3-28: Close up of vehicle path from May 22 simulator mission.

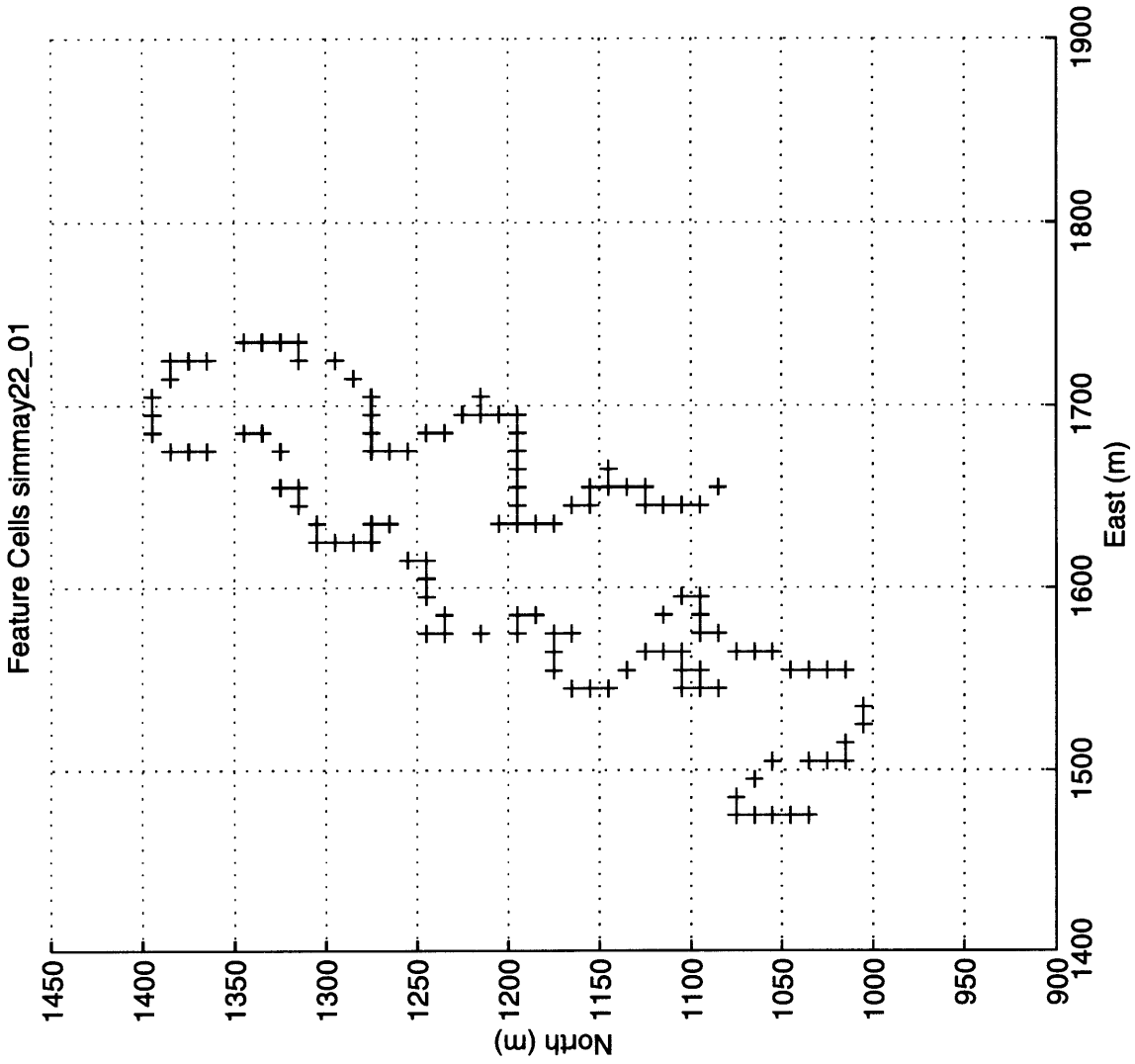


Figure 3-29: Feature map from May 22 mission.

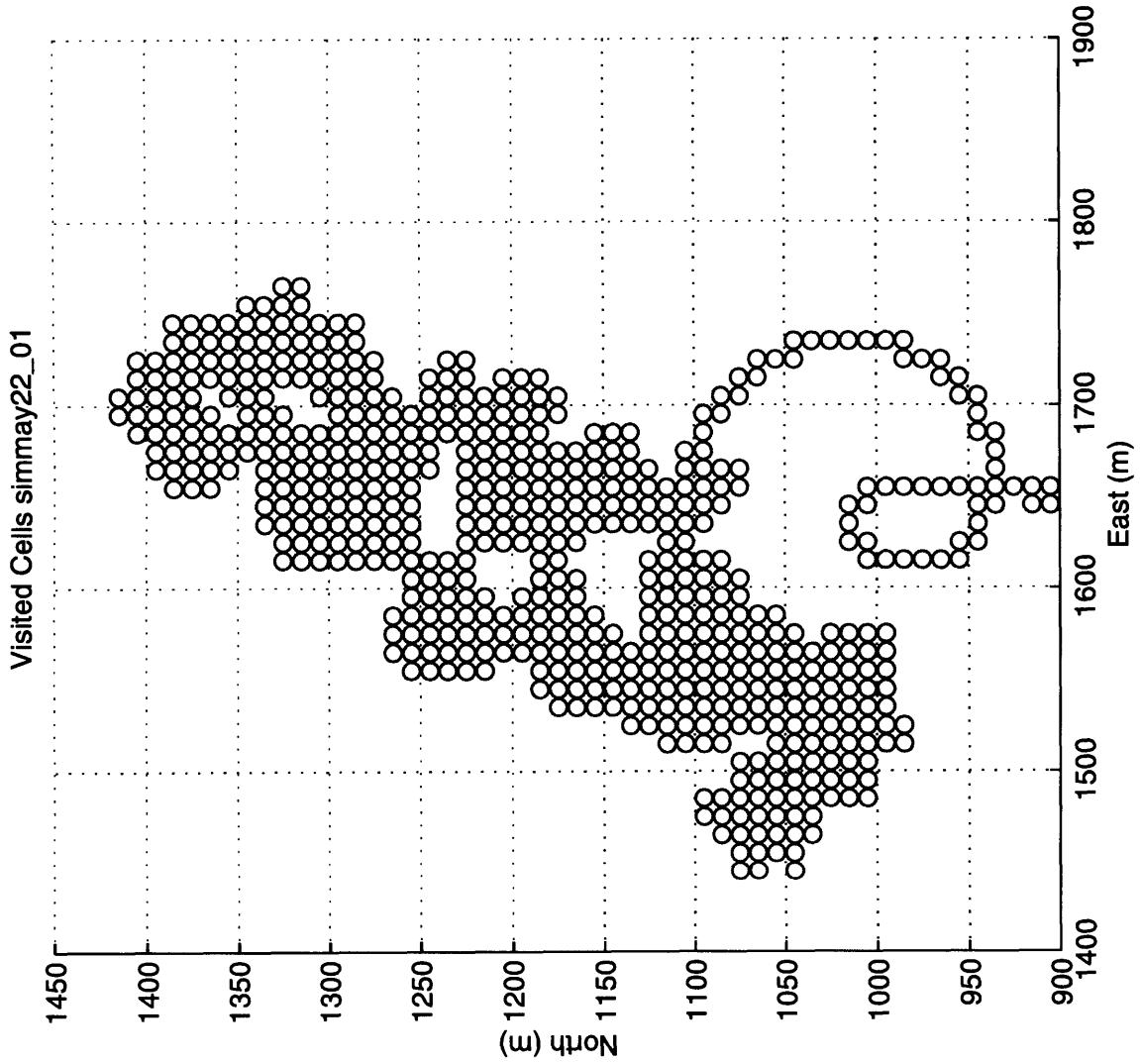


Figure 3-30: Visitation map generated by May 22 simulator mission.

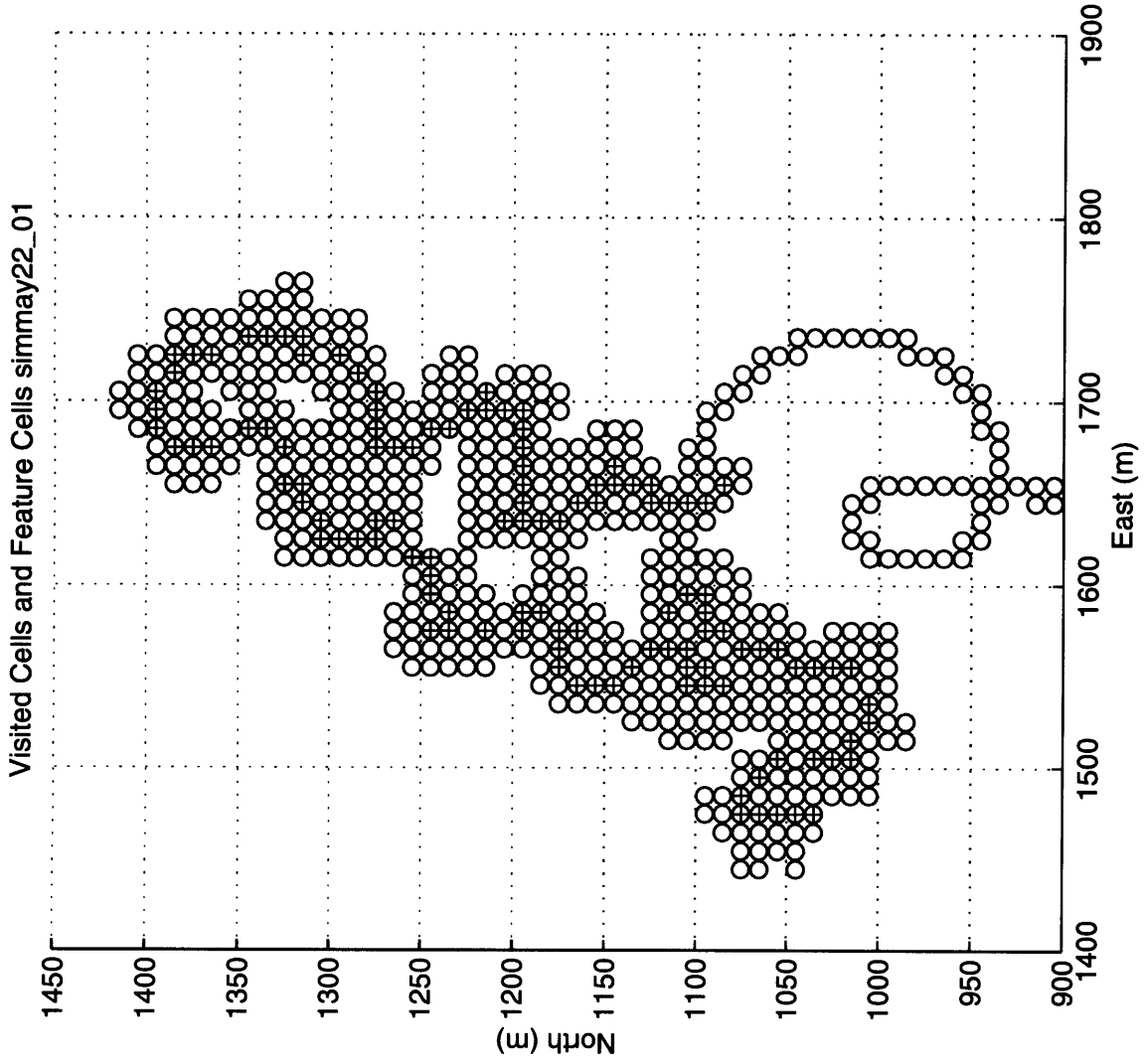


Figure 3-31: Overlay of feature map and visitation maps from May 22 simulator mission.

(SCLC) to handle multi-phase missions and point out that the behaviors employed are the same as those employed in the layered control scheme [14]. Their state configured layered control approach is based on the use of multiple sets of simple behaviors chosen from a central behavior library. Different sets are executed at each phase as the mission unfolds. The advantage of this approach is the reduction in the number of active behaviors at any time. This reduces, but does not eliminate, the interaction issues found in any behavior-based intelligent control system.

Mataric provides a recent summary of the current state of the debate in BBAI [77]. She divides the intelligent control community into three main arenas:

> *Planner-based* or *deliberative* strategies that typically rely on a central world model to verify sensor information and generate actions [35, 47, 62, 84]. The information in the central model is then used by a *planner* to determine a course of action necessary to satisfy the vehicle's goals. Planners can be quite powerful if sufficient time and information is available. Their chief drawback is the inability to handle rapidly changing environments in a timely fashion [25, 26].
>
> *Purely reactive* or bottom-up approaches embed the robot's control strategy in a set of preprogrammed condition-reaction pairs, or "reflexes." Reactive systems are characterized by no internal models, minimal state, and simple programming. They are characteristically present-time only. They are favored for their speed and have proven effective when the number of behaviors is low and the problem can be completely specified at design-time. They are limited during execution by their inability to store information dynamically.
>
> *Hybrid* systems attempt to find a compromise between these two extremes. The most promising approach has been to employ a reactive low-level system with a planner-based high-level decision maker. This has resulted in a large an diverse body of work, ranging from internalized plans [92] to contingency plans [36] and many more. A common division of labor is to utilize low-level competencies to handle immediate vehicle safety while the high-level planner determines the

optimal action sequence to satisfy the mission goals [18].

A specific subsection of hybrid systems is the *behavior-based* approach. This approach is closely tied to reactive systems but extends the function of an individual component by endowing it with more than a simple lookup table or reflexive ability. In contrast to a reactive component, behaviors can have some degree of "state" and can therefore utilize various forms of state representation.

A criticism of behavior-based systems is that because the specific meaning of what a behavior is has not been precisely pinned down, behavior-based systems lack a rigorous set of definitions and an associated systems analysis. Further, this lack of rigorous definitions makes it difficult to perform direct comparisons of the performance of different systems. However, it is this same lack of rigid definitions that has allowed so many ideas in so many forms and implementations to be attempted.

In spite of the criticism, there are elements that all behavior-based systems have in common: 1) There is no centralized representation operated on by one controlling reasoner; representations are instead distributed among many different behaviors or competencies and are maintained and updated in a distributed fashion. 2) All behavior-based systems contain multiple separate behaviors, each of which functions independently of the others. 3) The resulting vehicle actions are determined externally to the behaviors, either through a prioritization scheme [10], a voting scheme [92], or spreading of activation [74].

As Mataric points out

> ...the general constraints on behavior-based systems roughly mandate that behaviors be relatively simple, incrementally added to the system, that their execution not be serialized, that they be more time-extended than simple atomic actions of the particular agent, and that they interact with other behaviors through the world rather than internally through the system [77].

This thesis draws on the current research in the field to propose a new form of hybrid intelligent controller. In our case, the additional intelligence is not at

some "higher level" but is instead embedded in the low-level behaviors in a restricted fashion. Specifically, the issue of dynamic storing of information is directly addressed. The intent is to endow a reactive system with the minimum level of intelligence required to perform tasks which require dynamic learning of the environment. The learning is embodied in the form of a continuously changing map of the environment which is updated as required by the behavior itself. The result is a large increase in vehicle capability and functionality through an incremental increase in the complexity of the intelligent control system.

3.11 Summary

This chapter has described an intelligent control technique to perform adaptive region mapping. The task of finding a trench in the Charles River was used as a case study. Initial attempts to find the trench with a purely reactive system failed because the AUV would easily lose track of the feature. A new implementation using state configured layered control prevented the vehicle from losing the feature, but the AUV could easily become "trapped," inefficiently revisiting the same terrain over and over again, and failing to fully map the feature. To overcome these effects, a new approach called *adaptive layered control* was developed and implemented. In adaptive layered control, a behavior alters its own internal state in response to the sensed environment, taking into account the mission history. It builds up a map of feature locations, uses this map to generate candidate locations for other parts of the feature, and uses these candidate locations as navigation waypoints.

The new approach meets the challenge posed at the outset of the chapter, demonstrating the ability to track the Charles River trench. Given this capability, the next chapter examines the robustness and efficiency of the technique in the presence of navigation error and environmental disturbances.

Chapter 4

Robustness and Efficiency

In the previous chapter we developed a technique for adaptive feature mapping which incorporates adaptive behaviors into the layered control paradigm. This chapter examines the issues of robustness and efficiency. The key questions addressed are: What is the effect of navigation error? How sensitive is the performance of the system to the choice of internal parameters of the behavior? What metrics can be used to analyze the effectiveness of the approach? And, can the efficiency of the search be improved?

Robustness is addressed from both the feature-relative and global contexts. We examine how well the system tracks a feature, as well as whether it can stay with a feature in the presence of navigational uncertainty and environmental disturbances. We also examine how these disturbances and internal parameters affect the overall sampling strategy, and how this is reflected in the vehicle path.

Efficiency metrics are defined in terms of the amount of feature mapped per unit time, total path length traveled, the amount mapped in comparison to a conventional "lawnmower" survey of the same area, and how a given survey compares to surveys with different parameters. The feature is reconstructed from mission data and compared to contour maps generated off-line from the original data set.

Tradeoffs are demonstrated between overall robustness and efficiency, particularly under poor navigation conditions and the influence of external forces. All tests are performed in the Charles River basin simulation environment, using the 6 m and 7

m contours as targets. These features are demonstrated in Figures 4-1 and 4-2. For clarity, we present the contours in isolation when viewing vehicle results.

4.1 Performance metrics

The `trench_finder` adaptive behavior is necessarily of complex construction. In the field of intelligent control, the accepted method of determining the overall performance of a system, the effects of parameter changes, and the sensitivity to system and environmental disturbances is to extensively test the system over a range of parameter settings and environment conditions. This need for extensive testing is due in part to the wide range of intelligent controller designs in existence as well as the inherent complexity found in an individual design. In the case of an encapsulated competency such as `trench_finder`, which attempts to contain the task of sensing and reacting to unknown features in an unknown environment within one element of a behavior-based intelligent control system, this becomes a particularly complex and involved task. In all cases, the goal is to determine the range and optimum levels of the behavior in question, based on the environment, the mission goals, and the system parameters. As Russell and Norvig [97] observe:

> There is no accepted theory of architecture design that can be used to prove that one design is better than another. Many theoreticians deride the entire problem as "just a bunch of boxes and arrows."

We argue here that this lack of rigid metrics can also be viewed as an indication of the relative infancy of the field, and of the breadth of opportunities and approaches that this so-called "bunch of boxes and arrows" encourages.

In this section we will develop performance metrics and show how `trench_finder` performs in comparison to a standard lawnmower-type survey. We then proceed to examine the performance of the system in the presence of environmental disturbances. Finally, we systematically alter system parameters and examine the effects of these changes.

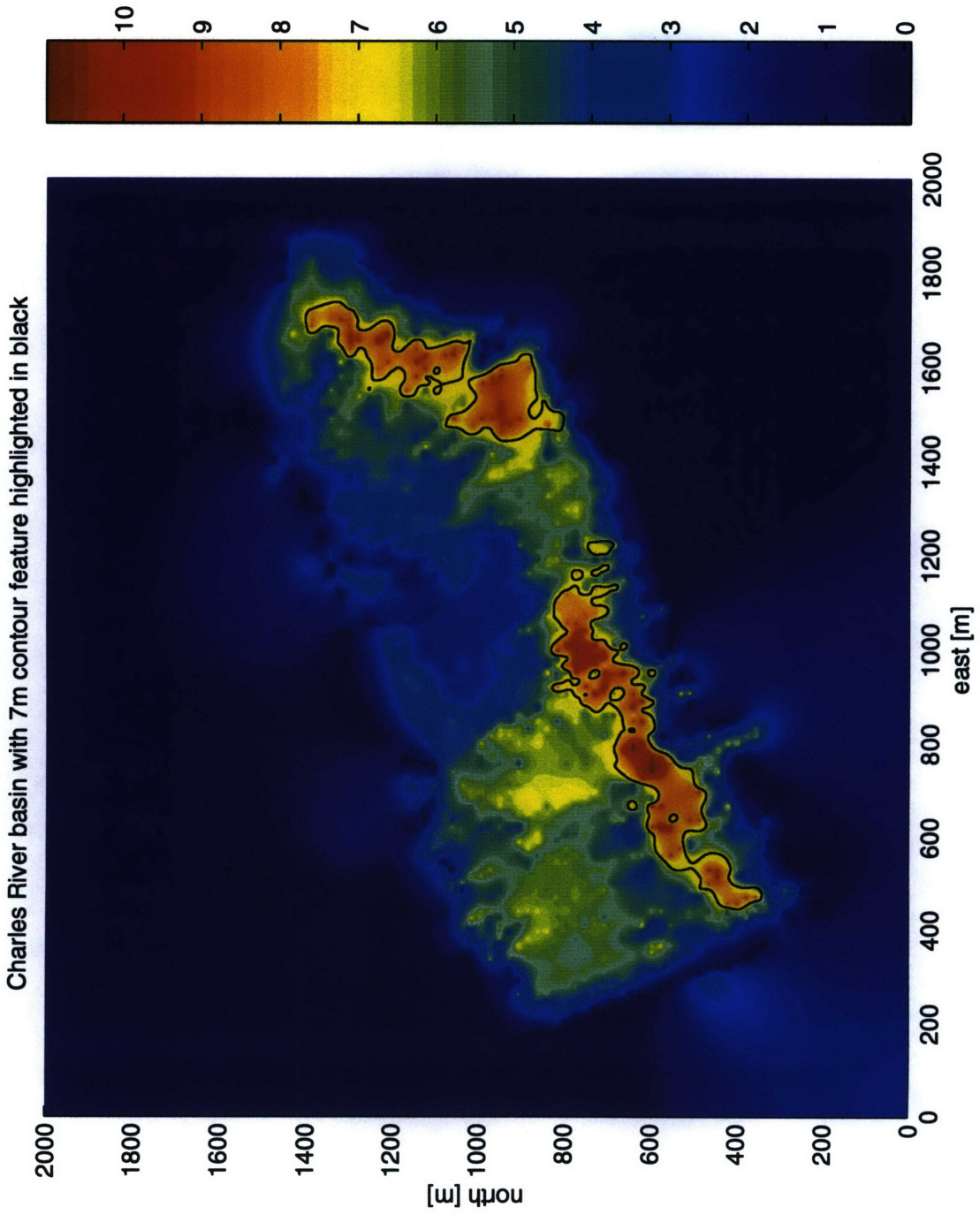


Figure 4-1: Charles River basin with 7 m contour highlighted.

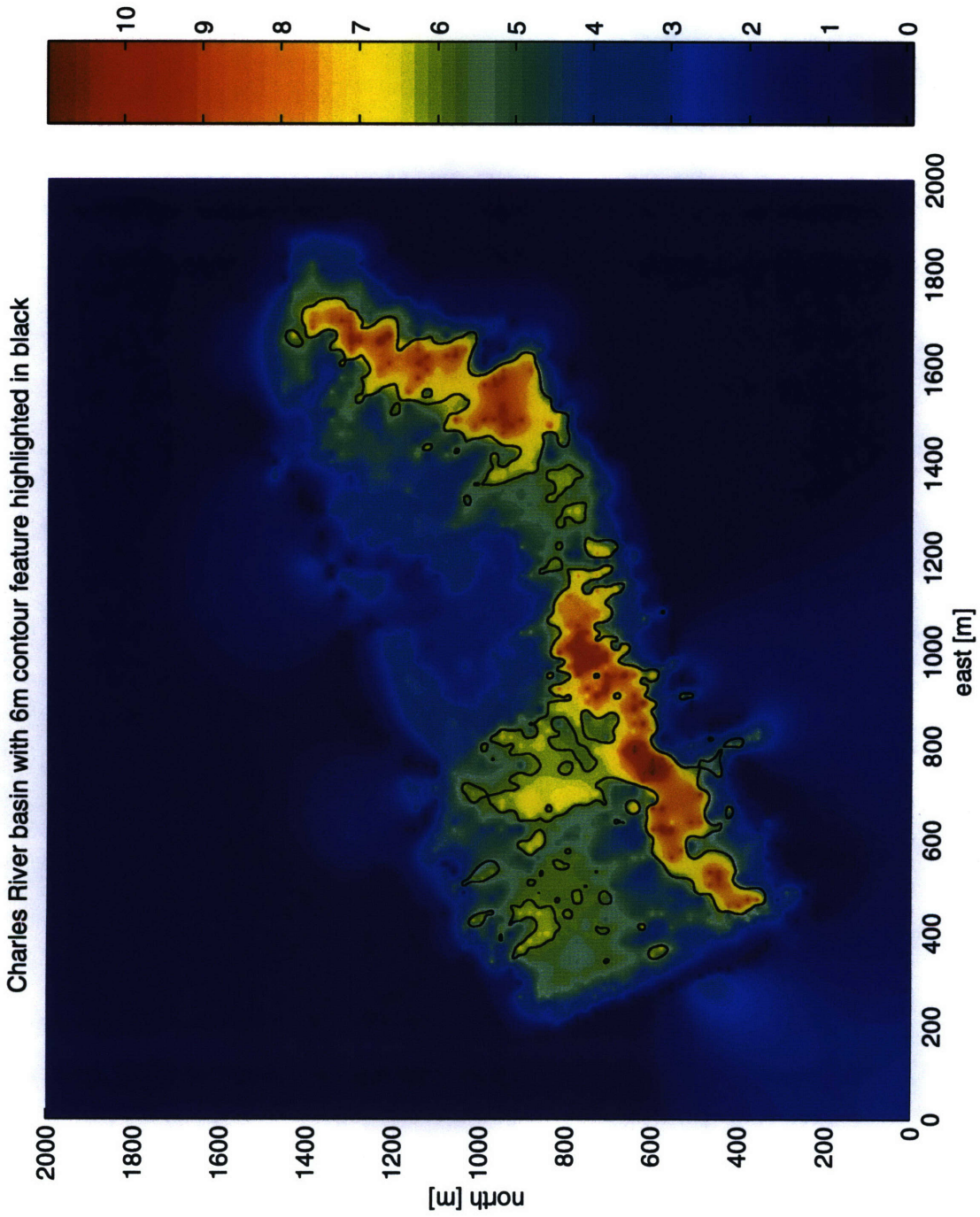


Figure 4-2: Charles River basin with 6 m contour highlighted.

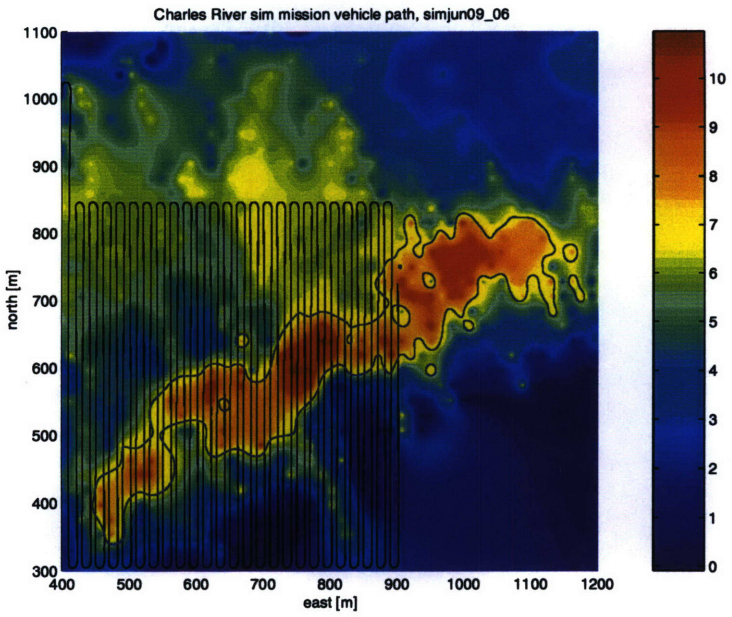


Figure 4-3: Path of a lawnmower-type survey of the Charles River basin 7 m contour. Mission time is 8 hours with an interval between laps of 10 m.

4.1.1 Comparison to a conventional survey

The most common method of surveying an unknown region is the "lawnmower" type survey. This has many advantages when mapping unknown terrain: low maneuvering demands (which is important for towed array systems), covers large swaths of terrain quickly, and it is relatively easy to reassemble the swaths into a complete picture. If, however, the goal is *not* to map a whole region, but to map a specific portion whose extent is unknown beforehand then this may be relatively inefficient when compared to a feature-relative approach. Figures 4-3 through 4-5 are a mapping survey of the 7 m contour, which we will compare to the mission of Figure 3-23.

The lawnmower survey of Figures 4-3 through 4-5 was an eight hour mission designed to map the feature at the same density (every 10 m) as the `trench_finder` mission. In the process, the survey covered a total path length of approximately 27.5 km. The mission successfully mapped approximately two-thirds of the trench before the time ran out. In contrast, the `trench_finder` mission lasted 5 hours and mapped the whole of the feature with a path length of approximately 14.3 km.

Using these surveys as a baseline, the metrics which we can employ are:

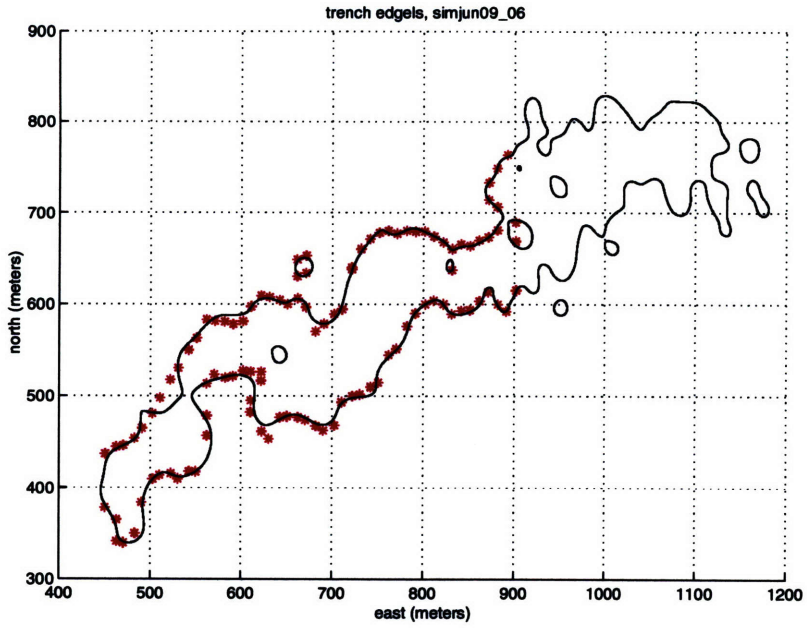


Figure 4-4: Feature map generated by the lawnmower-type survey shown in Figure 4-3.

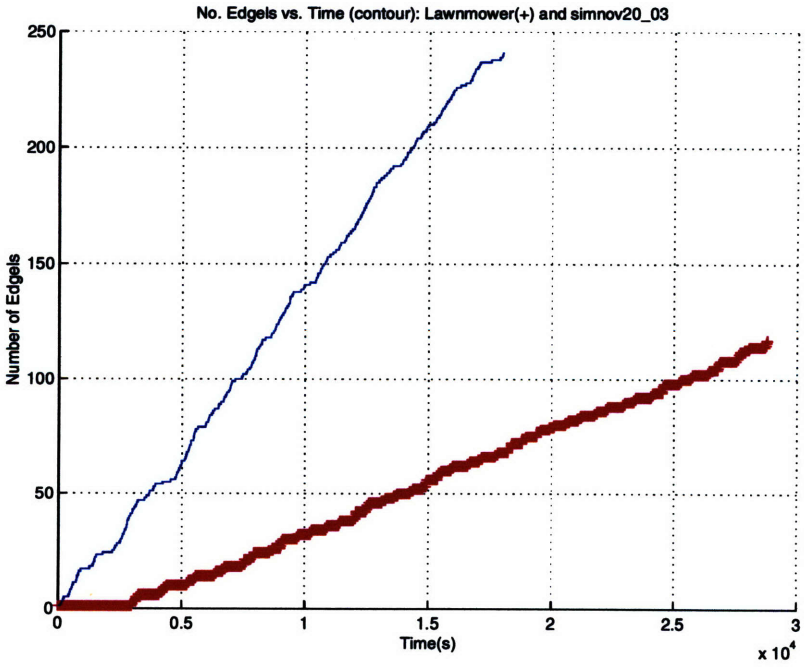


Figure 4-5: Amount of feature mapped (called "feature hits" during lawnmower survey of 7m contour as shown in Figure 4-3 (heavy line), vs. the vehicle mission shown in Figure 3-23 (light line).

- Survey path length relative to a lawnmower-type survey of the same duration
- Survey path length relative to the size/length of the feature
- Amount of feature found per unit time
- Amount of feature found per unit distance

These metrics and particularly direct examination of the vehicle path are used in the following sections to gauge the relative and overall performance of the `trench_finder` under varying conditions and using different operational parameters.

4.2 Effects of navigation error and external forces

The form and effect of navigation error depends upon the type of navigation system employed. For AUVs such as the *Odyssey II*, navigation is either an externally referenced system, such as LBL or the WHOI SHARPS/SNAP system, or wholly internal navigation system such as dead reckoning and/or INS. Externally referenced systems such as LBL are susceptible to error due to missed beacon pings, multipath signals, and even extraneous beacon signals from other arrays (see Vaganay, *et al* [117]). Internally referenced systems are susceptible to unmodeled and/or unsensed external forces on the vehicle and to unmodeled biases brought about by sensor miscalibration or failure. Common sources of such undetected/unmodeled influences are sideslip during turing maneuvers, vehicle fouling (e.g. kelp, rope, old nets, etc.), and external currents. Common sensor errors are compass bias (mismounted compass, unmodeled magnetic influences) and speed sensor bias (friction, limited dynamic range). We examine each source of error separately in the following sections.

4.2.1 Navigation error

LBL navigation

Figures 4-6 and 4-7 show a mission with a 3 msec LBL beacon timing error. This type of timing error is common in LBL systems in actual field operations, and poses

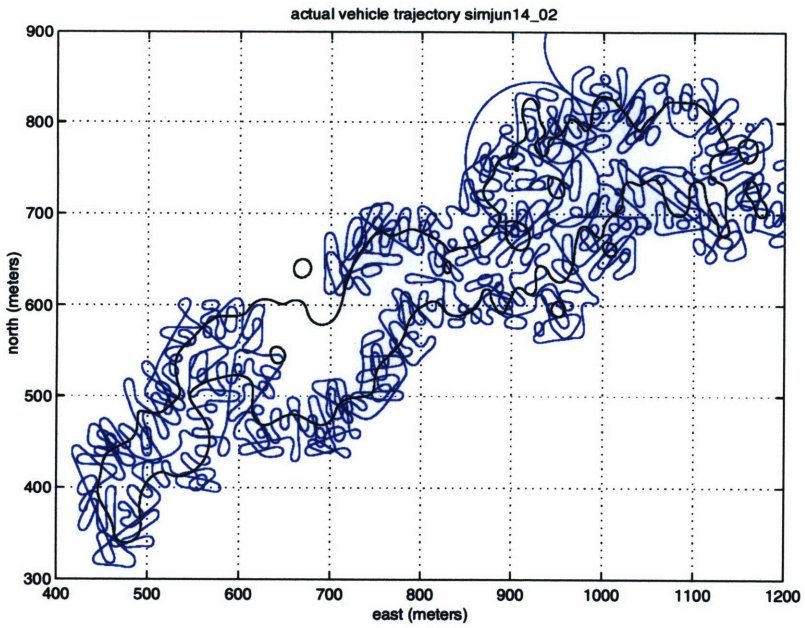


Figure 4-6: Path of the `trench_finder` behavior with a simulated 3 msec LBL timing noise. The path is not significantly affected. This noise level is typical of normal expected field operations.

no difficulty here.

Dead-reckoning navigation

Figures 4-9 through 4-15 show the effects of heading and water speed sensor errors on the `trench_finder` system. Figures 4-9 and 4-11 show the effect of a heading bias of -1.88 degrees. This form of bias can be the result of an unmodeled influence on the heading sensor or due to a mounting error during installation. Figures 4-13 and 4-15 show the effect of a water speed sensor bias of +17 cm/s. This is a very large offset and would likely be the result of fouling of the sensor or due to excessive internal friction (in the case of a mechanical water speed sensor).

Note that in both cases, the system continued to track and map the feature successfully. Since the feature maps generated are based on the internal navigator, the accuracy of the feature maps, especially Figure 4-14, have suffered in proportion to the magnitude of the navigational error. However, if the error can be determined *a posteriori*, this can be corrected in post processing. Alternatively, if the vehicle is being tracked during operations, corrections can be made with the aid of this

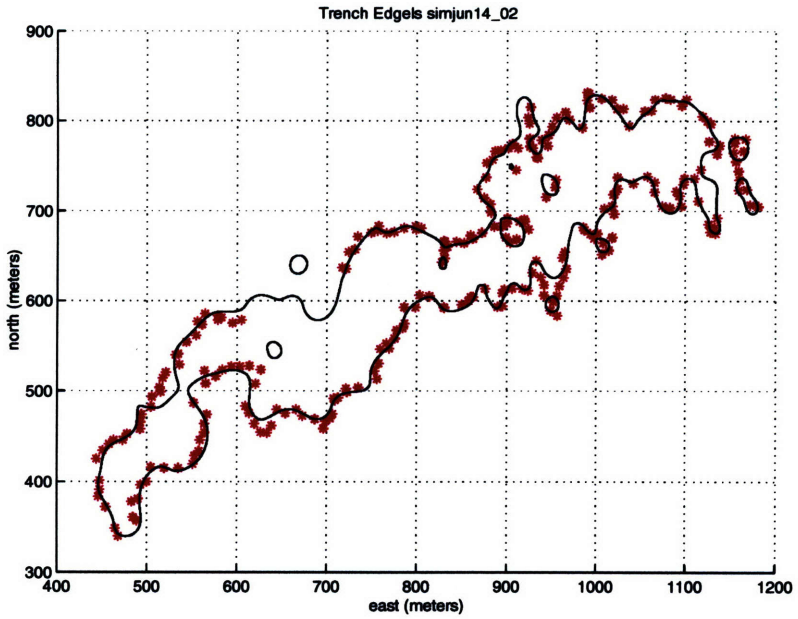


Figure 4-7: Feature map constructed by `trench_finder` based on the on-board LBL navigation system in the presence of timing noise of approximately 3 msec.

additional navigational data.

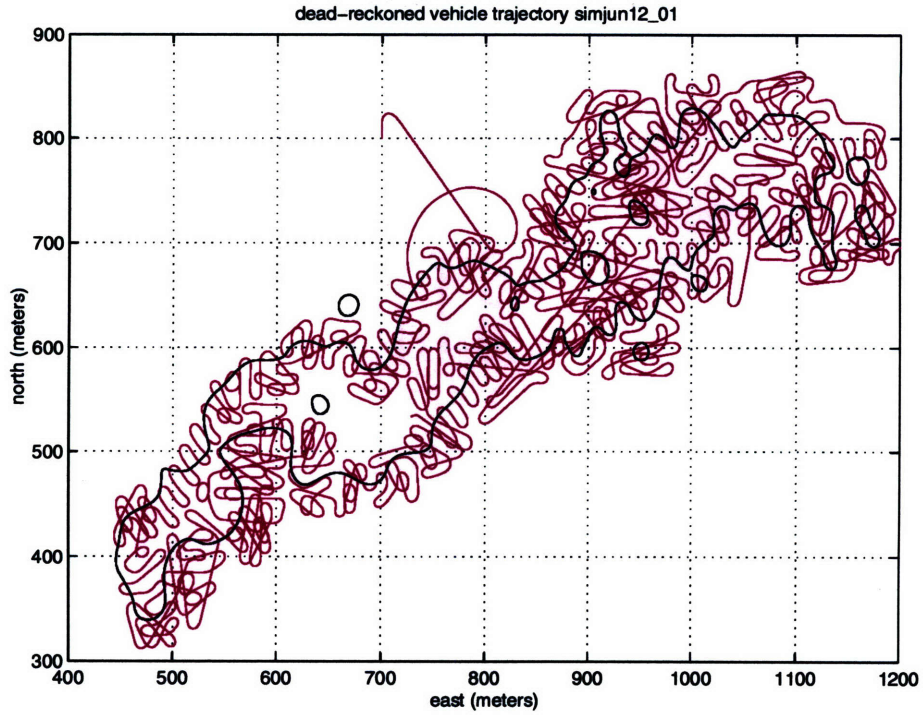


Figure 4-8: Dead-reckoned path of the `trench_finder` behavior with a heading bias of -1.88 degrees.

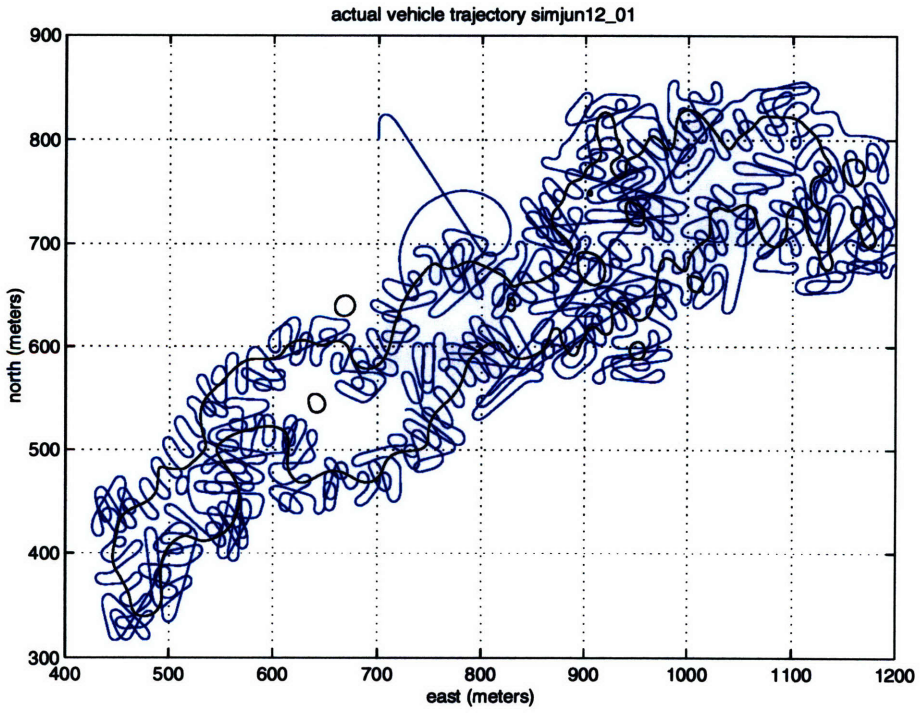


Figure 4-9: Actual path of the `trench_finder` behavior with a heading bias of -1.88 degrees.

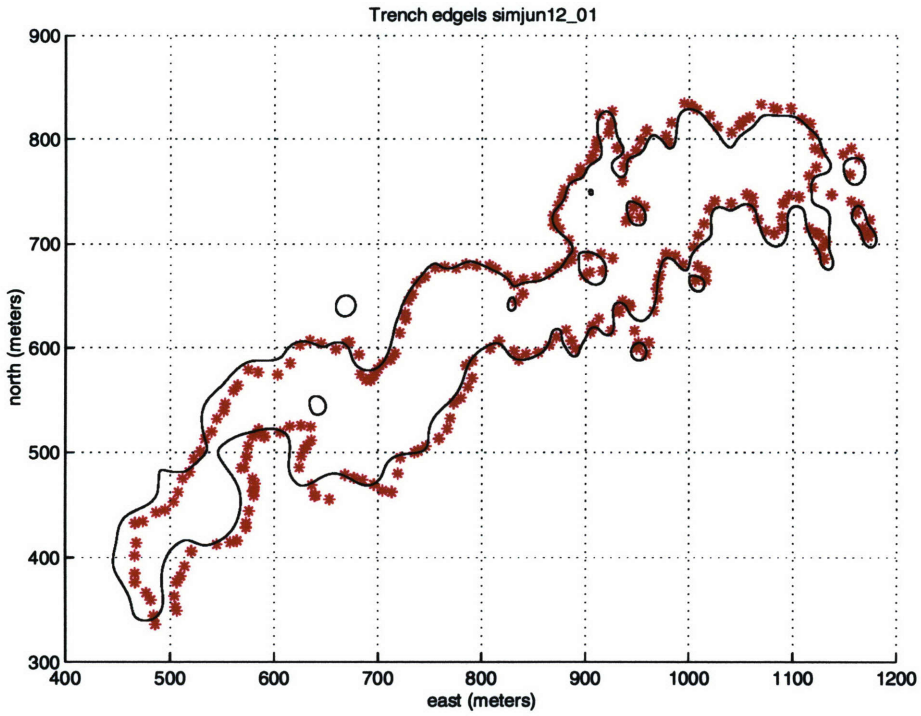


Figure 4-10: Feature map constructed with the navigational data from the mission of Figure 4-9. Note the effect of the heading bias (particularly on the southern portion of the feature) of the resulting estimated feature location.

4.2.2 External forces

Steady external forces which act upon the whole of the vehicle (such as currents) are particularly hard to model and predict in a dead reckoning or INS navigation system. This is due to the fact that the vehicle drifts along within the current itself. The INS system cannot measure such a steady-state drift due to the lack of acceleration, and a dead-reckoner which relies on a speed sensor that measures waterspeed relative to the vehicle will similarly be mislead by the lack of relative motion between the vehicle and the current as it drifts along. Navigation errors build up quickly when the `trench_finder` behavior relies solely upon such a dead-reckoning system. Figures 4-16 through 4-19 show the results of two area mapping missions in the presence of an external current. Figures 4-16 and 4-17 show the effect of an unmodeled 15 cm/s southerly current while Figures 4-18 and 4-19 show the effects of an easterly current. In both occasions, the vehicle maintained contact with the feature despite the current.

In the case of Figures 4-16 and 4-17, the vehicle encountered what the dead-

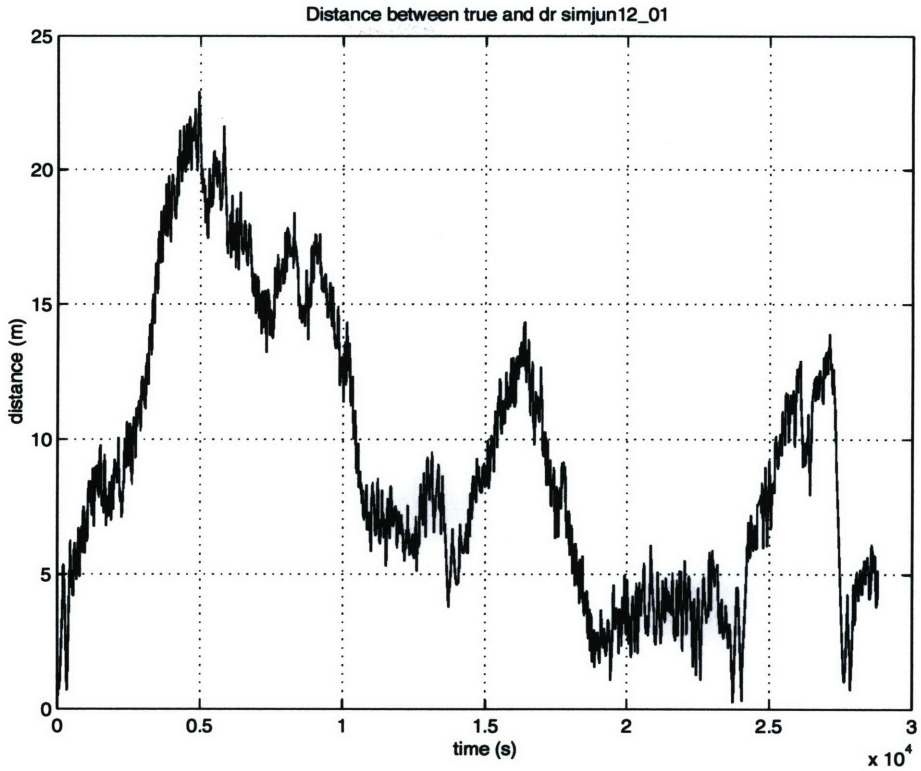


Figure 4-11: The distance shown indicates the difference between actual and dead-reckoner estimated position for mission of Figure 4-9.

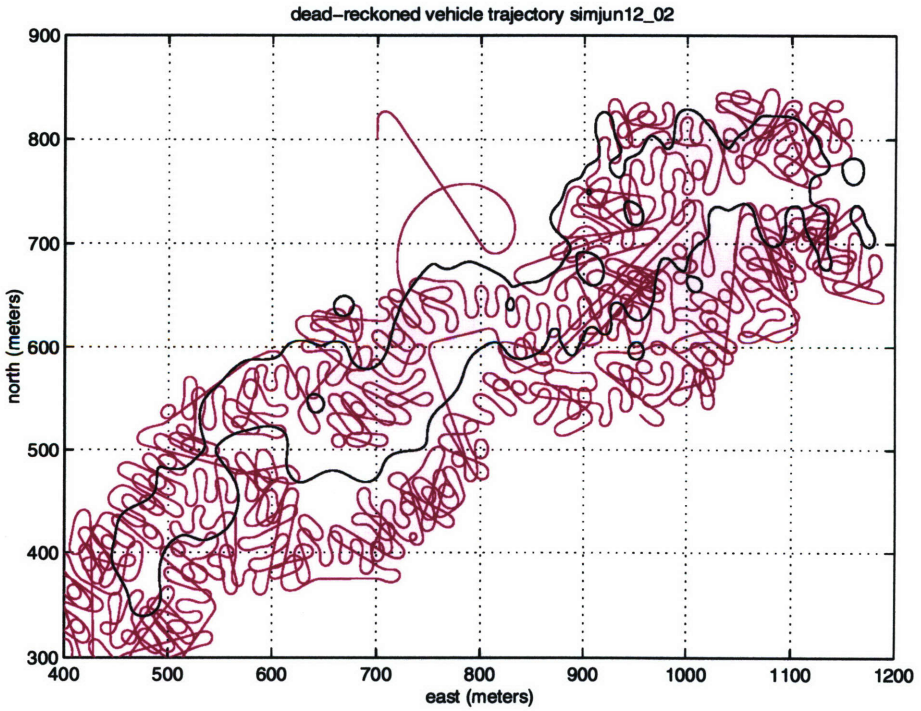


Figure 4-12: Dead reckoned path of the `trench_finder` behavior with a water speed sensor bias of approximately +17 cm/s.

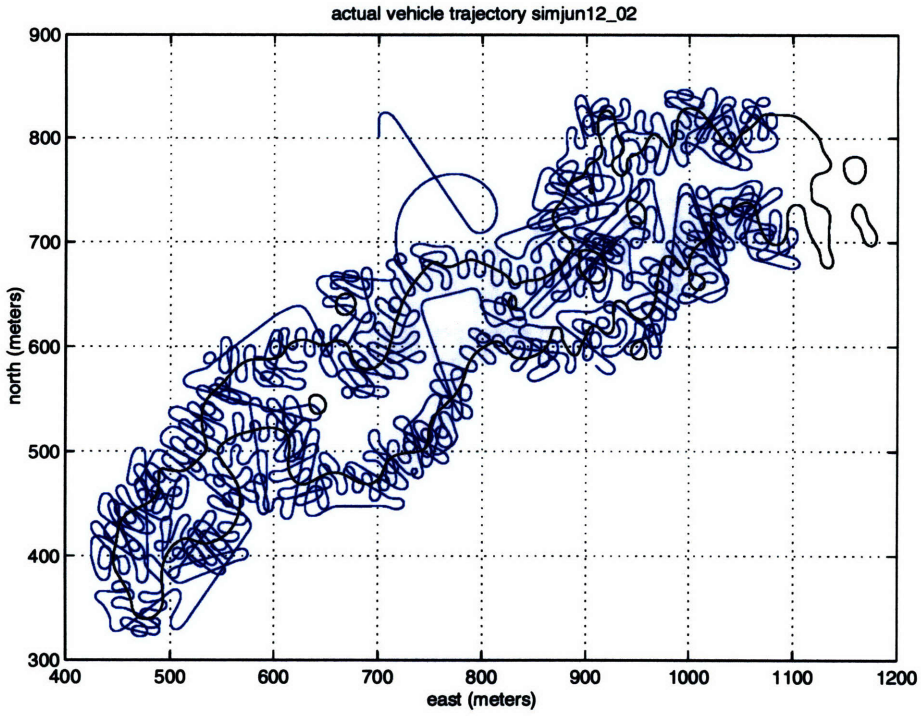


Figure 4-13: Actual path of the `trench_finder` behavior with a water speed sensor bias of approximately +17 cm/s.

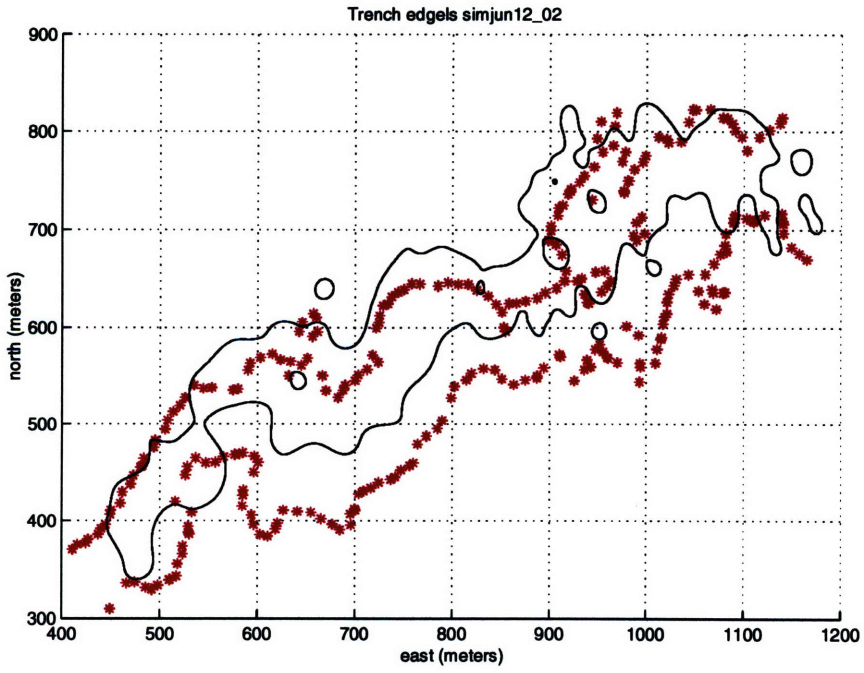


Figure 4-14: Feature map constructed with the navigational data from the mission of Figure 4-13. Note the effect of the speed sensor bias on the resulting estimated feature location.

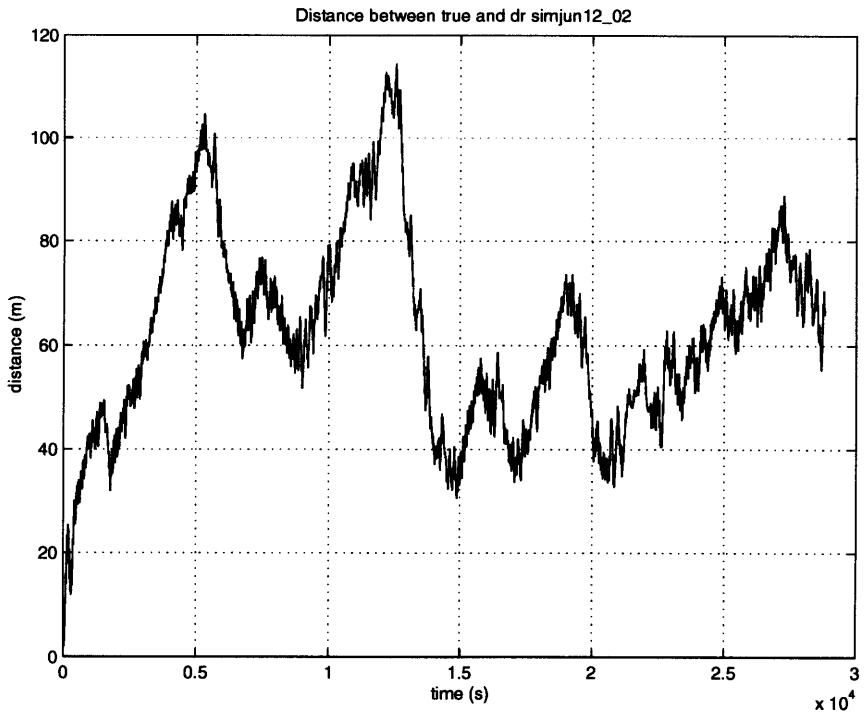


Figure 4-15: This distance measures the difference between actual and dead-reckoner estimated position for mission of Figure 4-13. Note the significant impact the speed sensor has on total error, as compared to Figure 4-11.

reckoner believed to be the northern edge of the search area. At that point the vehicle stopped its northward progress, allowing the current to push it off of the map. Figures 4-18 and 4-19 show the vehicle attempting to map an area while drifting in a 15 cm/s easterly current. The vehicle maintains contact with the feature, but is gradually pushed across the feature to the east each time it travels in a cross-current or along-current direction.

In Figures 4-20 and 4-21 the vehicle is again subjected to an easterly current of 15 cm/s. This time, however, the vehicle is attempting to map the 7m contour only and not the entire feature. The resulting true vehicle path shows the vehicle "dwelling" on one particular contour while the dead reckoned path shows that the vehicle believed itself to be gradually working its way westward. It maintained position until the dead-reckoning (DR) system determined that it had reached the western edge of the search area (200 m E). At this point `trench_finder` began to search adjacent areas, which allowed the current to gradually push the vehicle off of the map to the east.

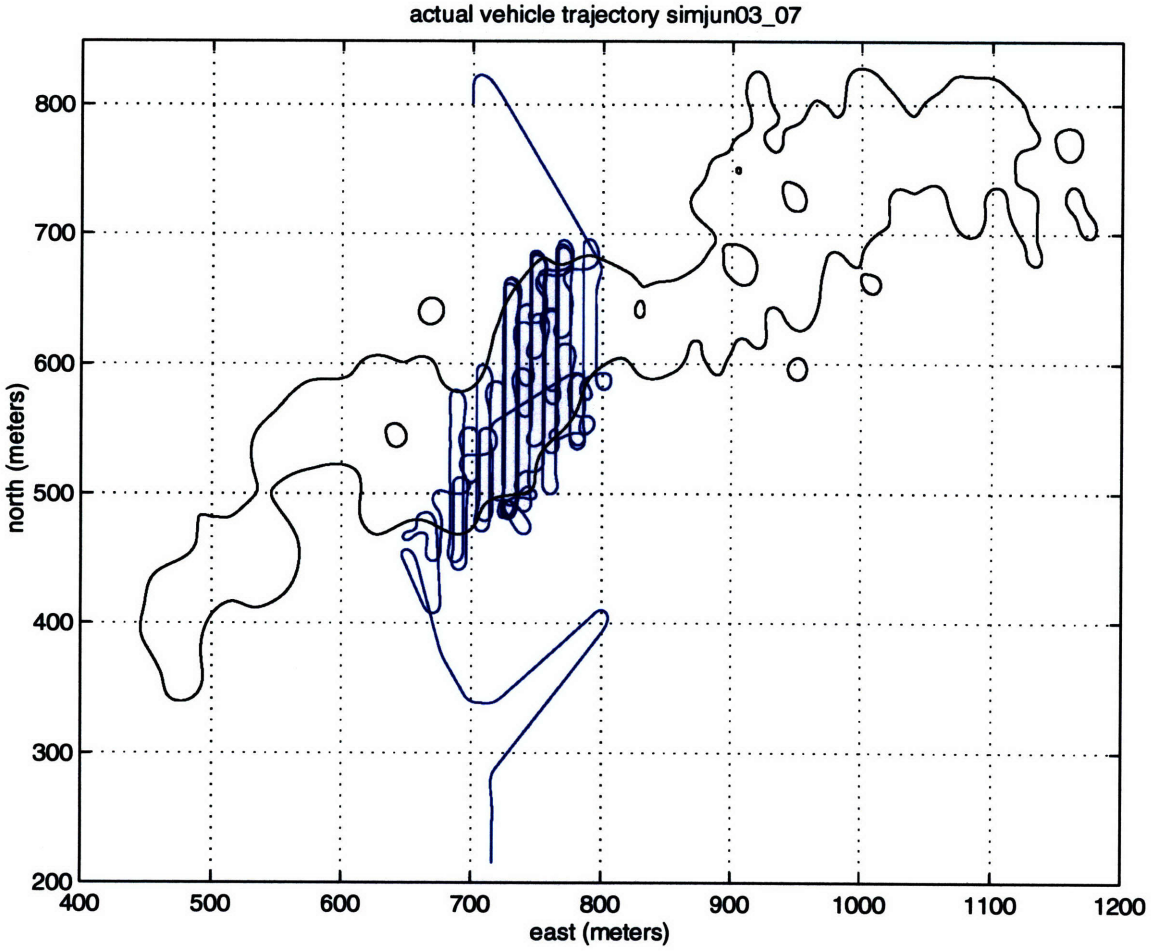


Figure 4-16: Actual path of vehicle attempting to map region contained by the 7 m contour using dead reckoning in an unmodeled 15 cm/s southerly current. Vehicle was moving with a speed of 1.0 m/s.

Figure 4-22 shows the effect of an external easterly current on `trench_finder` when it employs an LBL navigation system. The external navigation array allows the vehicle to track its position relative to the beacons, thereby countering the effect of the current.

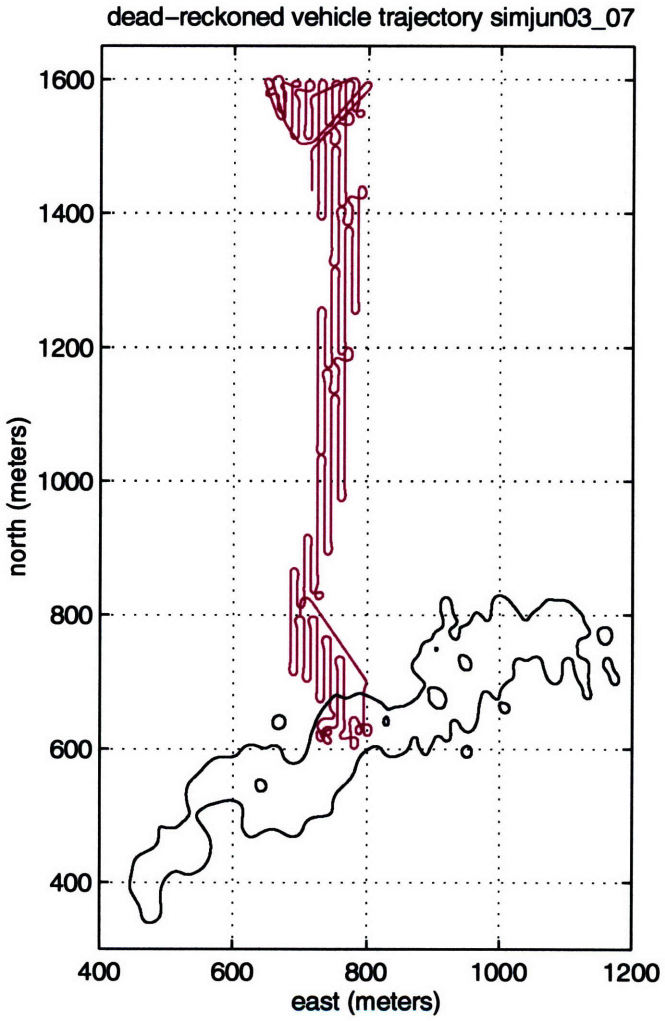


Figure 4-17: Dead reckoned path of vehicle attempting to map a region contained by the 7m contour in an unmodeled 15 cm/s southerly current. Vehicle was moving with a speed of 1.0 m/s. Note that because of the current, the dead reckoning system had calculated that the vehicle had reached the northern limit of the search area (1600m north). This resulted in the vehicle drifting to the south with the current as it mapped what it believed to be valid adjacencies to the east and west of the feature.

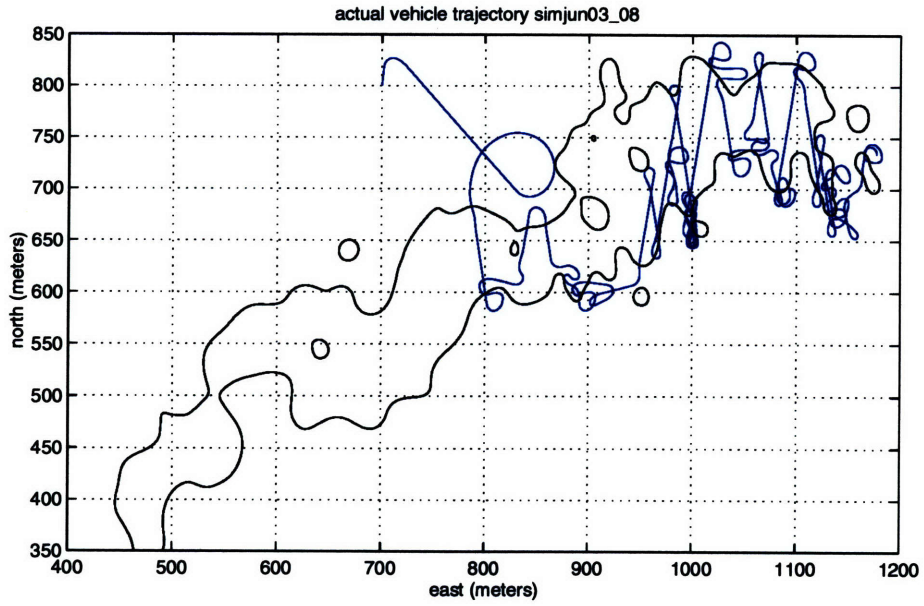


Figure 4-18: Actual path of vehicle attempting to map region contained by the 7m contour using dead-reckoning navigation in an unmodeled 15 cm/s easterly current. Vehicle was moving at a speed of 1.0 m/s.

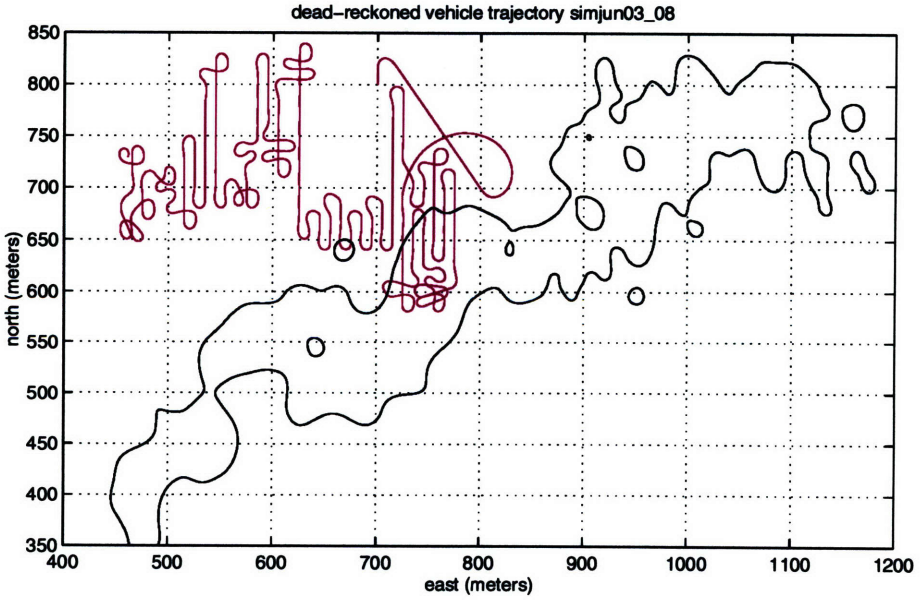


Figure 4-19: Dead reckoned path of vehicle attempting to map region contained by 7 m contour with an unmodeled 15 cm/s easterly current. Note that the vehicle believed it was working its way westward while in fact it was being displaced gradually eastward by the current.

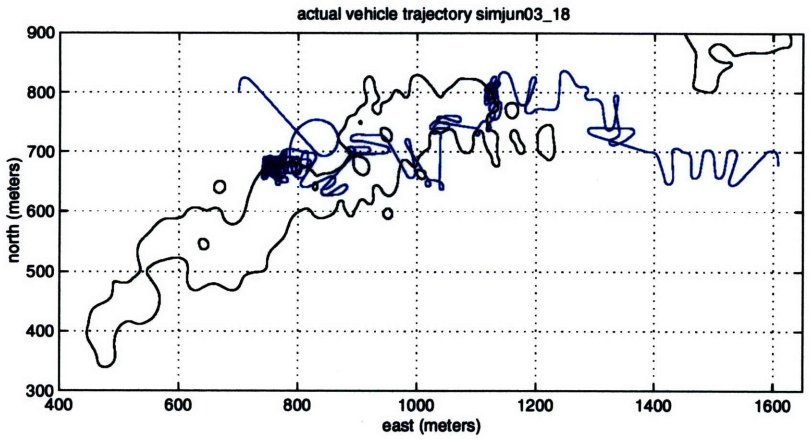


Figure 4-20: Actual path of vehicle attempting to map 7 m contour with an unmodeled 15 cm/s easterly current. Vehicle was moving at a speed of 1.0 m/s. Note that the vehicle "dwelled" on the feature until compelled to turn away due to error buildup in the DR navigation system.

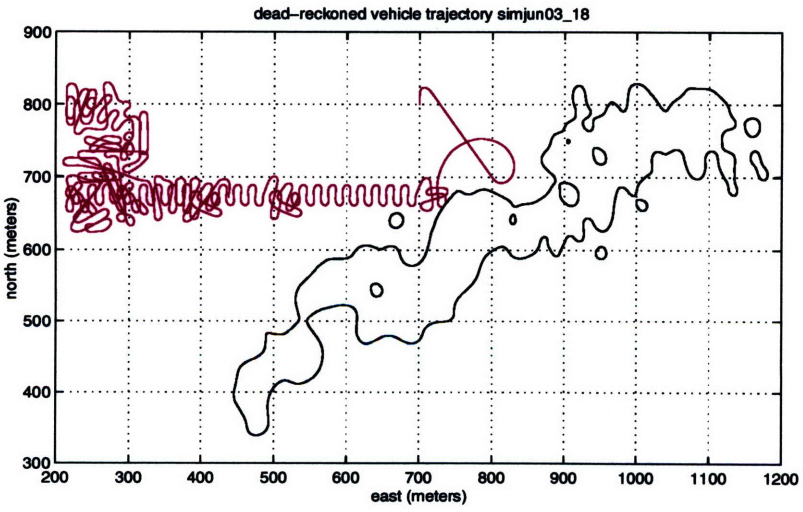


Figure 4-21: Dead reckoned path of vehicle attempting to map 7 m contour with an unmodeled 15 cm/s easterly current. Vehicle was moving at a speed of 1.0 m/s. The unmodeled current caused the DR system to believe it was working its way westward until it reached the western limit of the search area (200 m east). At this point the vehicle began to search around the area, allowing the current to gradually push the vehicle off of the map to the east.

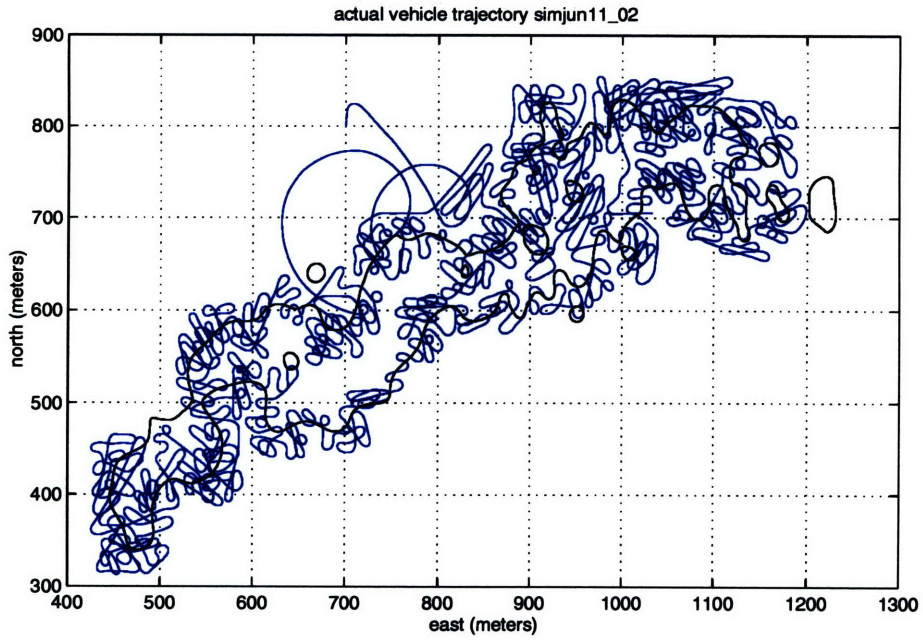


Figure 4-22: Path of the `trench_finder` behavior with a simulated 3 msec LBL timing noise and an easterly current of 10 cm/s. Note that the path is not significantly affected by these disturbances. This noise level is typical of normal expected field operations, while the currents are slightly faster then those found in the Charles River basin.

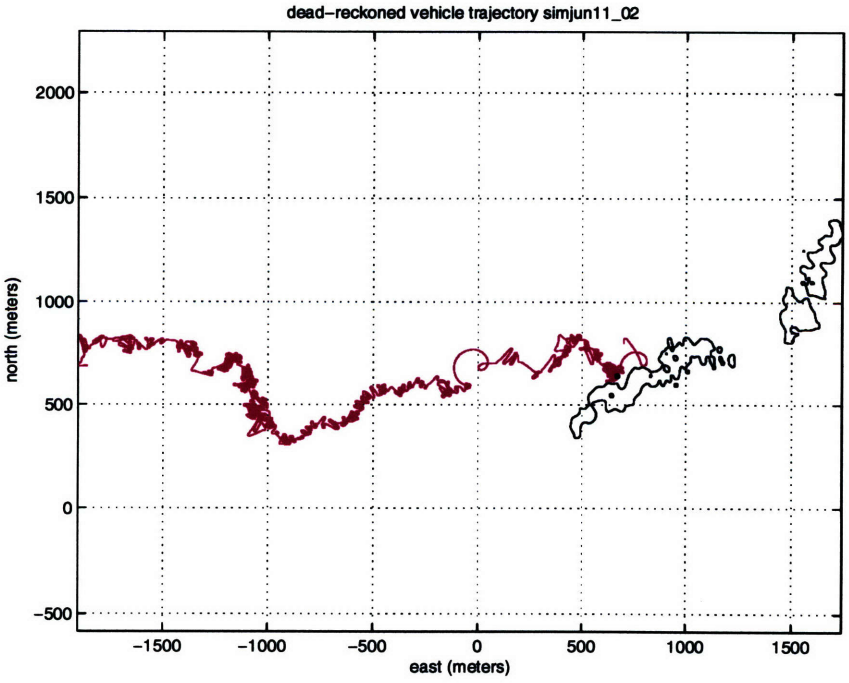


Figure 4-23: Dead reckoned vehicle path of the mission of Figure 4-22.

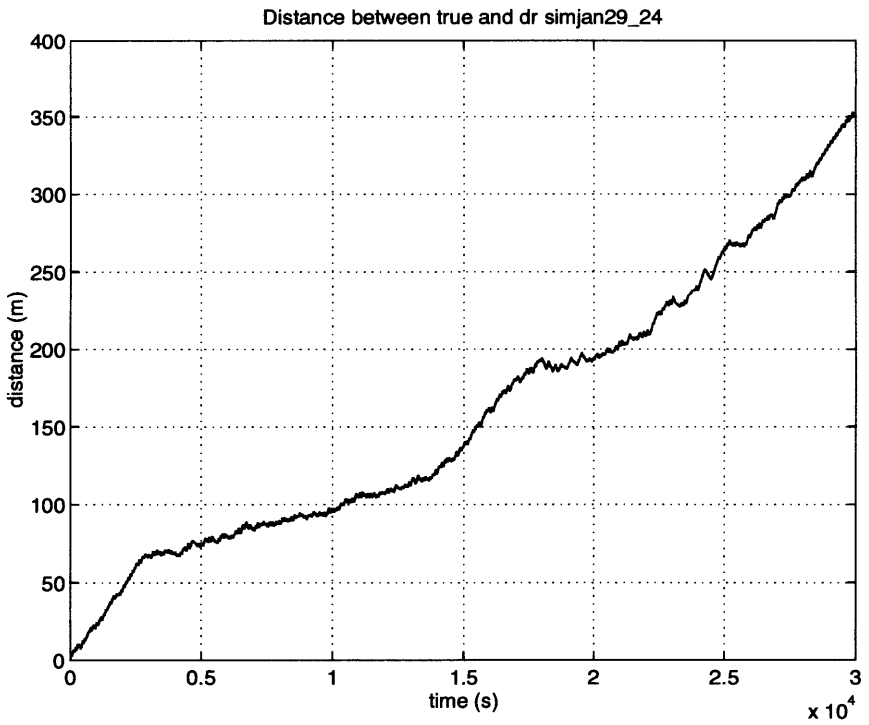


Figure 4-24: Error growth in dead reckoning navigation system for mission shown in Figure 4-25. Note that while the dead reckoner believed the vehicle was approximately 350 meters away from the true position, the vehicle itself had actually lapped the feature and was working its way around the feature for a second time.

4.2.3 Combined navigation and external current error

Figures 4-25 and 4-26 show the combined effects of a -1.1 degree heading bias, a 5 cm/s speed sensor bias, and an external current of 0.4 cm/s N, 11 cm/s E. Mission duration was 8 hours 20 minutes, starting at [800 E, 700 N] and finishing at approximately [450 E, 400 N], traveling a total path length of 24.5 km. While the dead reckoning navigation system showed steadily increasing error, the actual vehicle path overlapped the original path for the last portion of the mission, showing `trench_finder`'s ability to maintain contact with a feature despite disturbances.

4.3 Parameter sensitivity

As discussed previously, there are several operational parameters which can be adjusted to alter the performance characteristics of the `trench_finder` behavior. In this

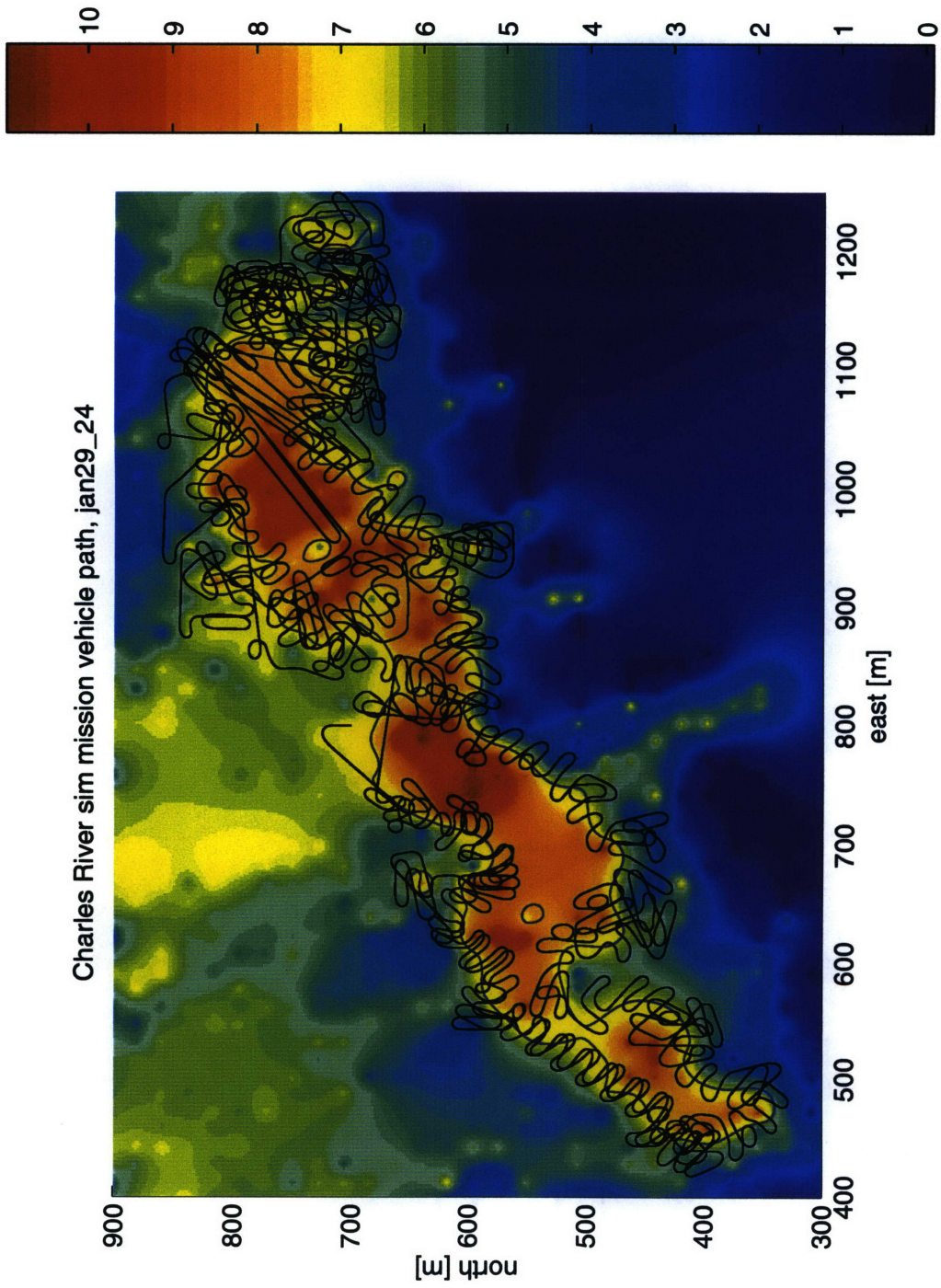


Figure 4-25: Actual vehicle path from mission combining a heading bias of -1.1 degrees, a speed bias of -5 cm/s, and an external current of 0.4 cm/s N, 11 cm/s E. The vehicle started its mission at [800 E, 700 N] and ended at approximately [450 E, 400 N].

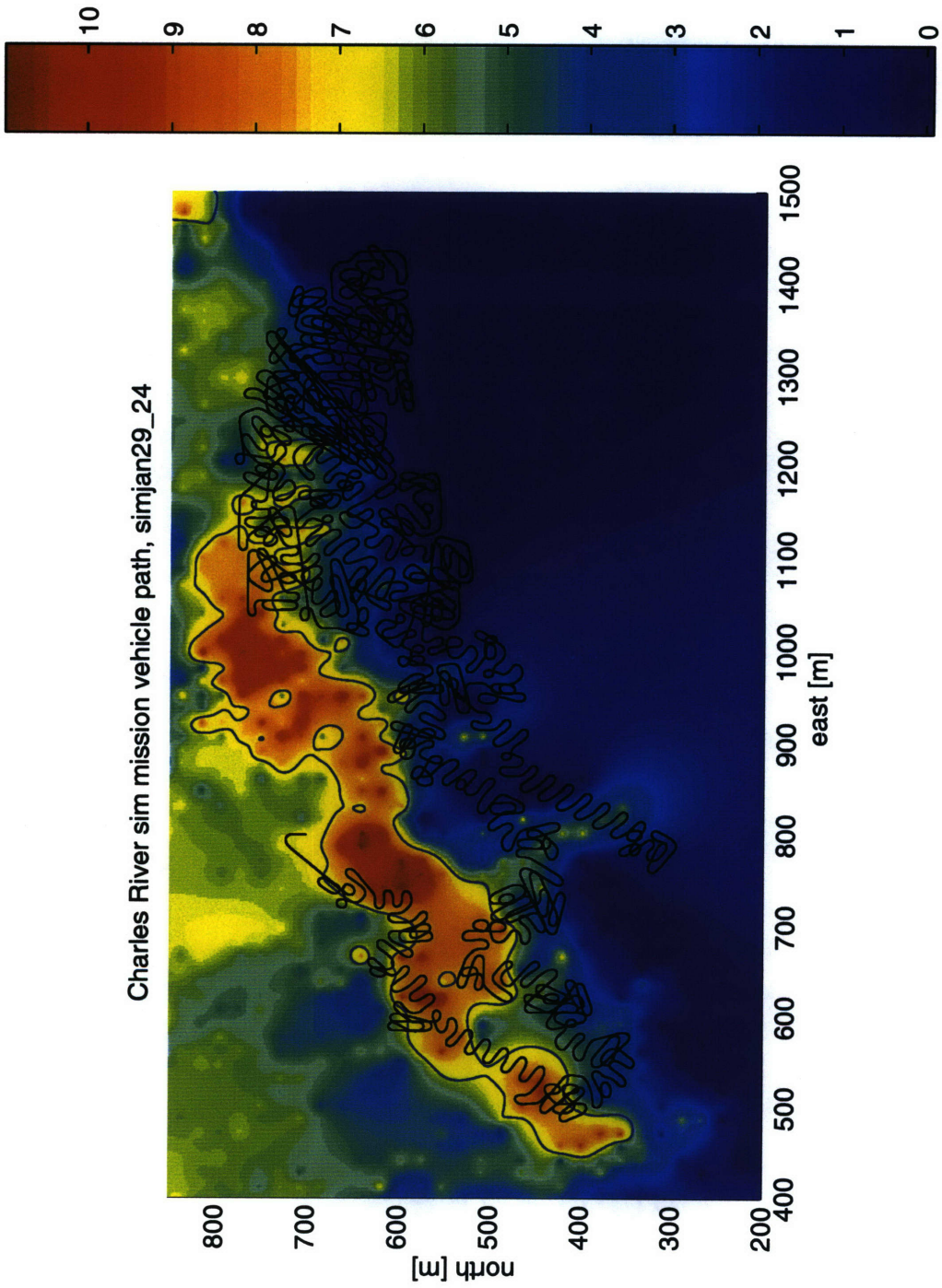


Figure 4-26: Dead reckoned vehicle path from mission shown in Figure 4-25. Note that while the dead reckoner navigation error continued to increase, the `trench_finder` behavior continued to successfully map the feature.

section we proceed to examine the sensitivity to and effects of varying the parameters of the `trench_finder` behavior under various simulated conditions. Throughout all cases, the default settings are the same (as shown in Table A.1) unless specifically set otherwise.

4.3.1 Map cell size

Varying the on-board map cell size has the effect of trading resolution for area covered in a given time. This is not surprising given the fact that a larger map cell size increases the pixelating effect that any cell-based map will have. When choosing map size the user must balance the conflicting concerns of the vehicle dynamics, desired feature resolution, time or energy available for the mission, and the desired area of coverage.

Figures 4-27 through 4-32 illustrate the effect of cell size on vehicle path. Each vehicle mission has the same operational parameters, boundaries, initial conditions, and environmental factors, but uses a different internal map cell size. Figure 4-27 shows the resulting mission when the map cell size is 20×20 meters. The vehicle path covers a large area quickly and relatively coarsely. The corresponding feature map generated is therefore also coarse, but gives some indication of the extent of the feature. Such a cell size (or larger) would be desirable if the desire is to locate several features over a large area within a limited time.

Figures 4-29 and 4-30 shows the standard 10×10 meter cell survey. This "standard" cell size was chosen as a compromise between the level of resolution and the rate of coverage. Figure 4-29 shows the vehicle path, while figure 4-30 shows the resulting feature map.

Finally, Figures 4-31 and 4-32 show a mission using a cell size of 5×5 meters. This results in a slow but detailed map of the feature, as seen in Figure 4-32. For a non-holonomic vehicle such as the *Odyssey II*, smaller cell sizes create a need for a more thoughtful approach to the problem of path planning. This is particularly important when the desired cell size is less than the turning radius of the AUV being employed.

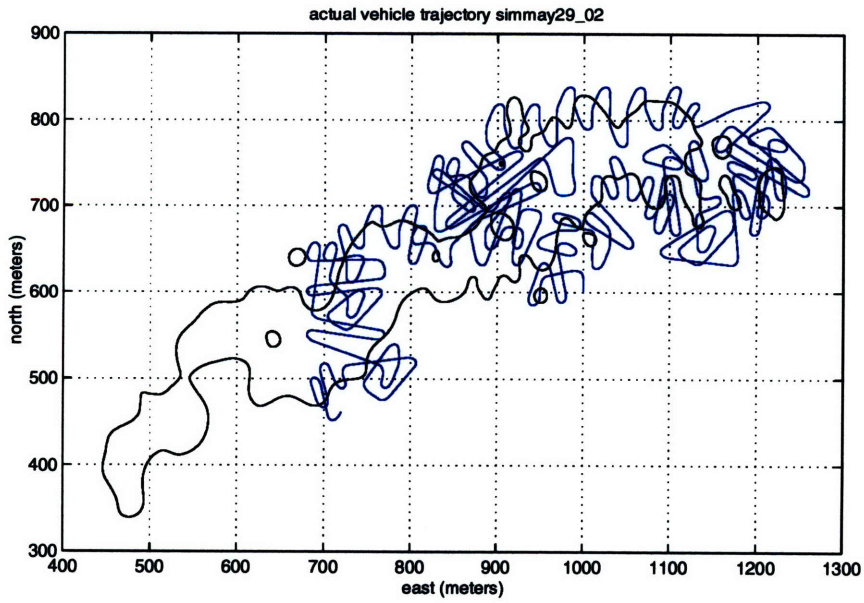


Figure 4-27: Vehicle path resulting from a map cell size set at 20 × 20 meters. In this mission the vehicle is mapping the 7 m contour line in the Charles River Basin. Mission time was 2 hours and 40 minutes.

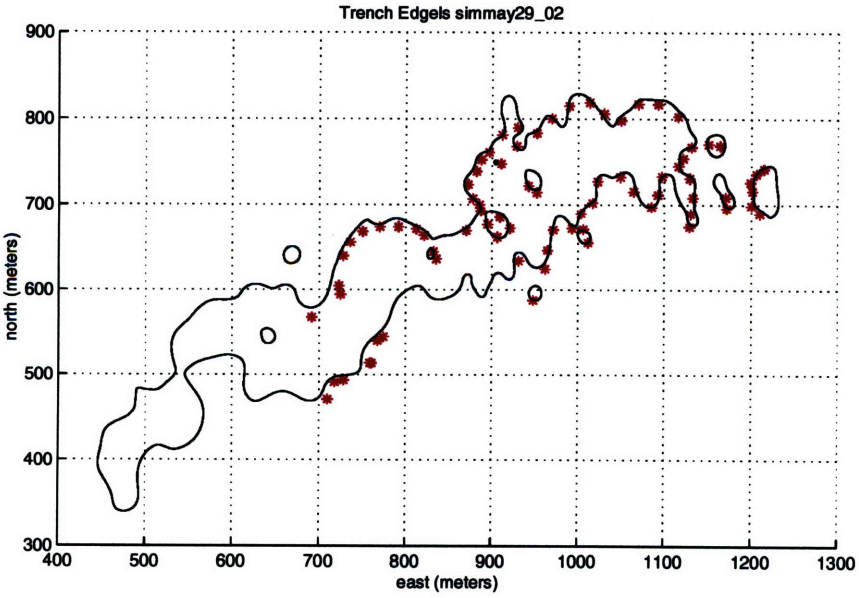


Figure 4-28: Feature map resulting from a map cell size set at 20 × 20 meters.

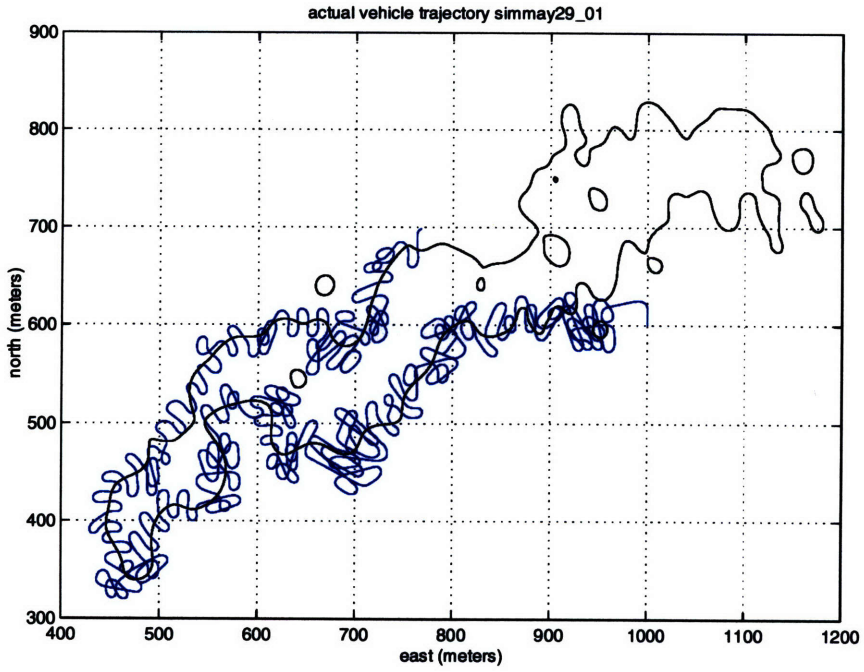


Figure 4-29: Vehicle path resulting from a map cell size of 10 × 10 meters. As with Figure 4-27, the mission was 2 hours and 40 minutes long. The resulting vehicle path covers less of the feature, but at a higher level of detail.

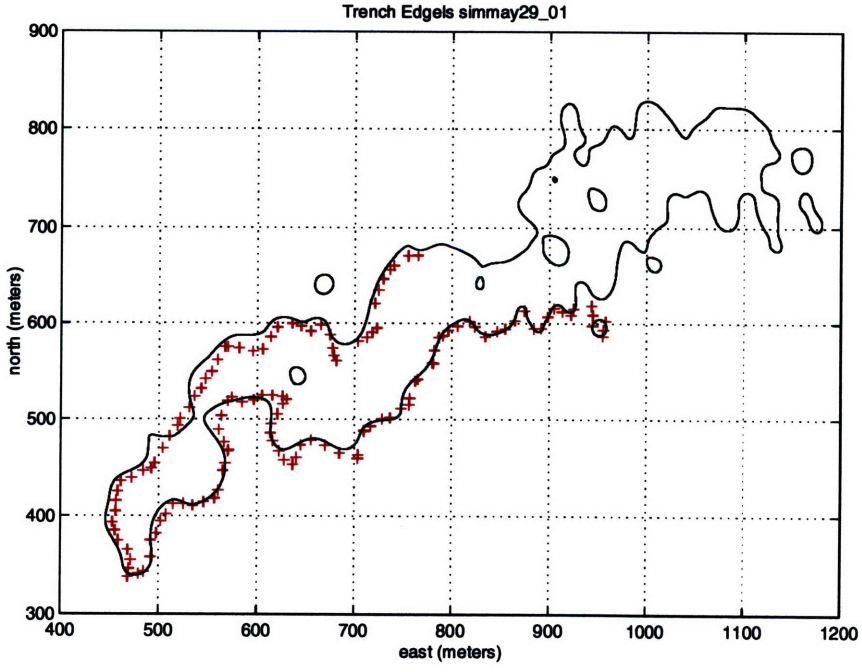


Figure 4-30: Feature map resulting from a map cell size set at 10 × 10 meters.

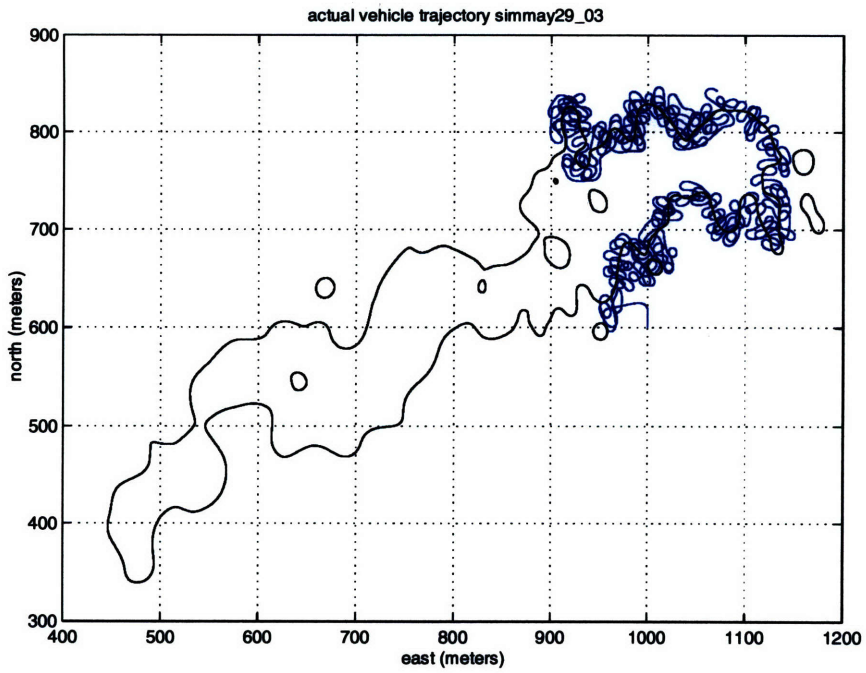


Figure 4-31: Vehicle path resulting from a map cell size of 5 × 5 meters. As with the previous two examples, the mission duration was 2 hours and 40 minutes long. However, in this mission the cell size was set to 5 meters. The resulting vehicle path is a more detailed sampling of a smaller portion of this particular feature.

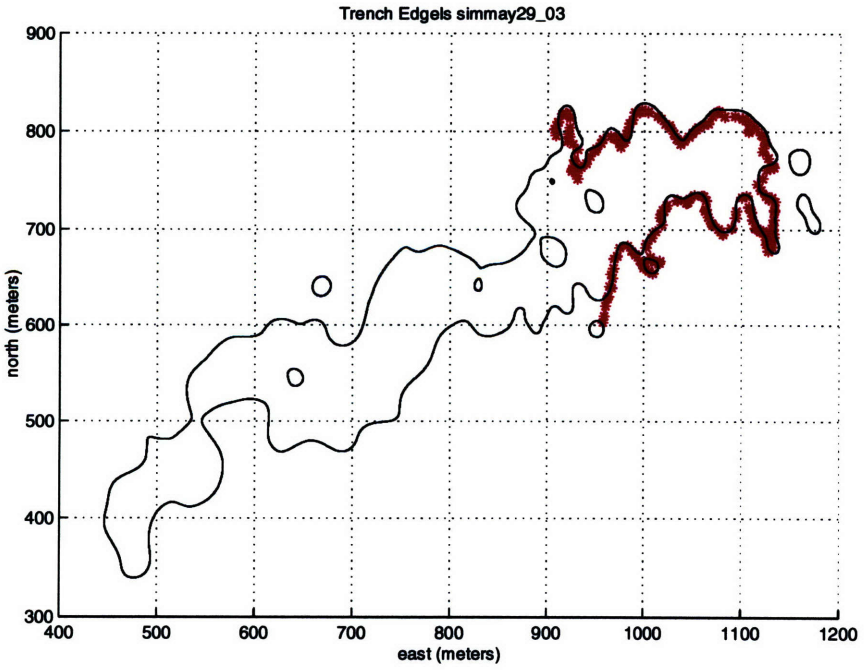


Figure 4-32: Feature map resulting from a map cell size set at 5 × 5 meters.

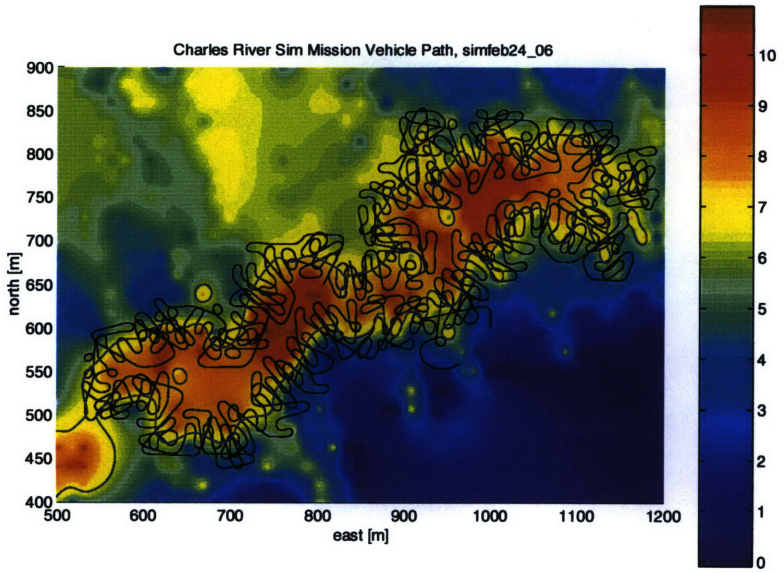


Figure 4-33: Close up of vehicle path from a contour-based search mission.

4.3.2 Region vs. contour search

The `trench_finder` behavior can be set to search for and map either contour-based or region-based features. Region-based mapping is desirable when detailed information about a body, such as the interior structure of a trench or a high-salinity region, is desired. Conversely, `trench_finder` can also be instructed to *avoid* specific regions, such as areas of high temperature gradient (particularly useful for a polymer-hulled vehicle near thermal vent fields). Figure 4-33 shows a contour-based mission, while Figure 4-35 shows a region-based mission. The corresponding feature-maps are shown in Figure 4-34 for the contour mission and Figure 4-36 for the region-based mission.

4.3.3 Search zone

Figure 4-37 shows what happens if a feature is bisected by the limits of the search zone. In this example the white line indicates the limit of the valid search area. The vehicle is launched at [700 N, 500 E] and is instructed to start searching to the east. It quickly locates the feature and begins mapping until it reaches the search zone limit. It then turns back into the search zone and proceeds to locate and map that

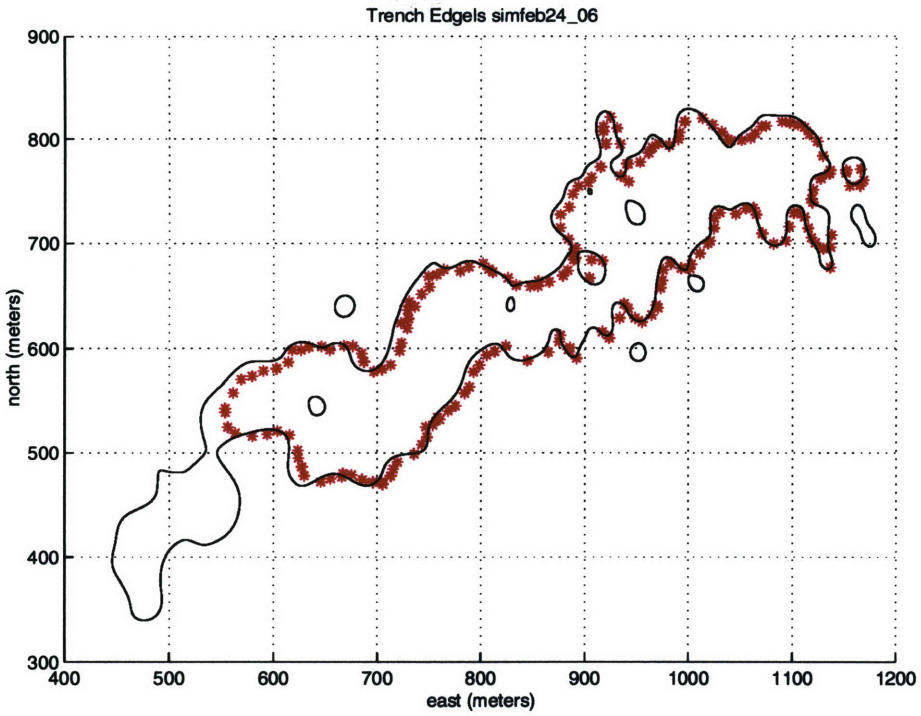


Figure 4-34: Feature map generated by `trench_finder` on the contour-based mission shown in Figure 4-33.

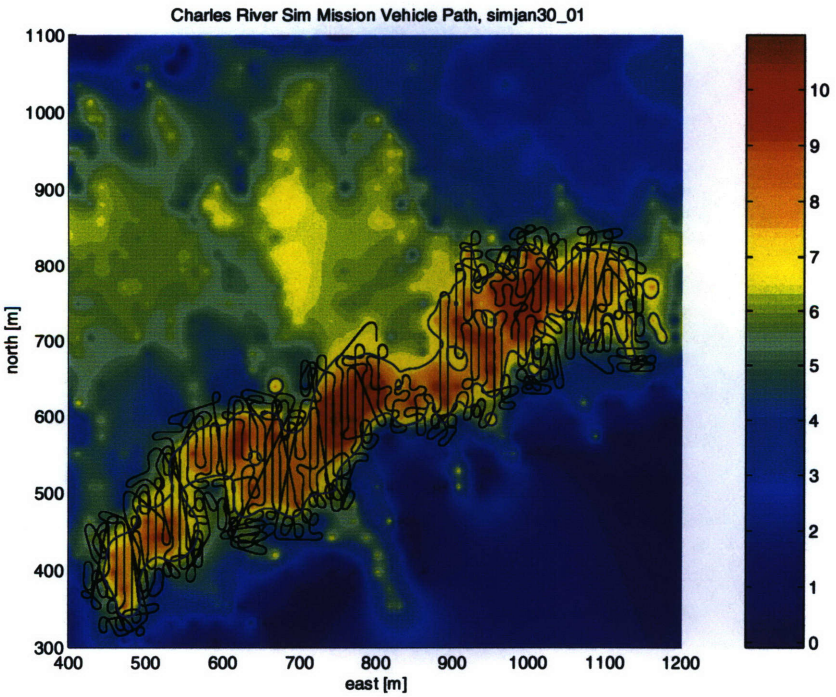


Figure 4-35: Close up of vehicle path from a region-based mission.

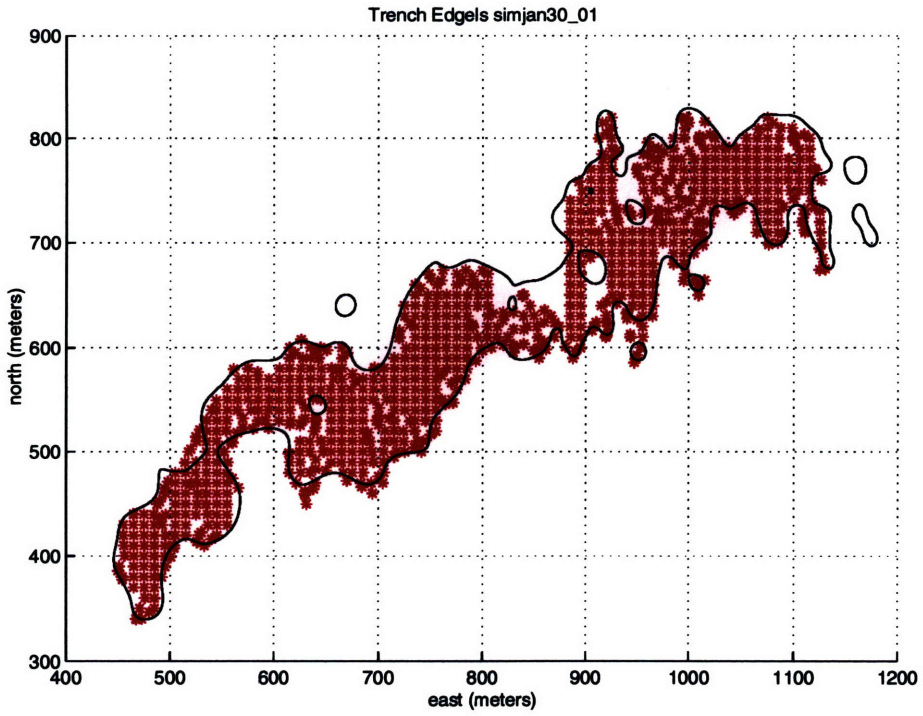


Figure 4-36: Feature map generated by the region-based mission shown in Figure 4-35.

portion of the feature which is accessible without leaving the assigned search area.

4.3.4 Waypoint selection

Lookup tables

The performance of `trench_finder` can be altered by changes in the pre-compiled lookup tables used to rank the sequence of available candidate waypoints. As an example, a waypoint set which favors a consistent directionality is shown in Figure 4-38. The lookup table set is designed to favor consistent progress around a feature along the general direction the vehicle is currently heading. The result is that if the vehicle begins to map a feature in a particular direction (in this case, generally clockwise), the lookup table set will continue to favor this.

In contrast, a lookup pattern which encourages a more arbitrary selection of waypoints (such as the lookup pattern shown in Figure A-1) results in a more "random" search. The difference between the two can be seen by comparing Figures 4-38 and

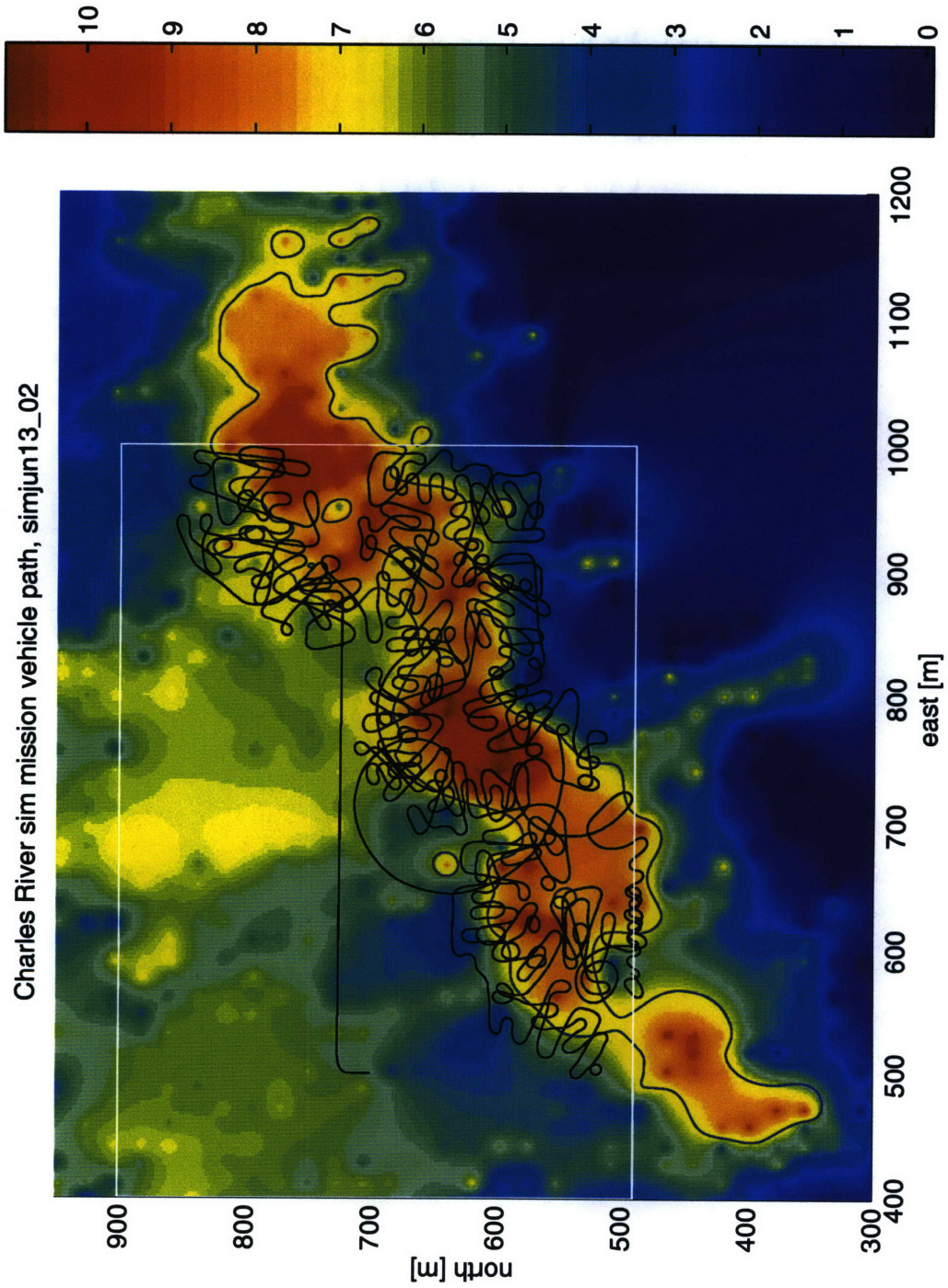


Figure 4-37: Effect of a search zone boundary bisecting a feature. In this mission the white boundary line marks the limits to the search area. `trench_finder` adapts its mapping operation to accommodate this restriction.

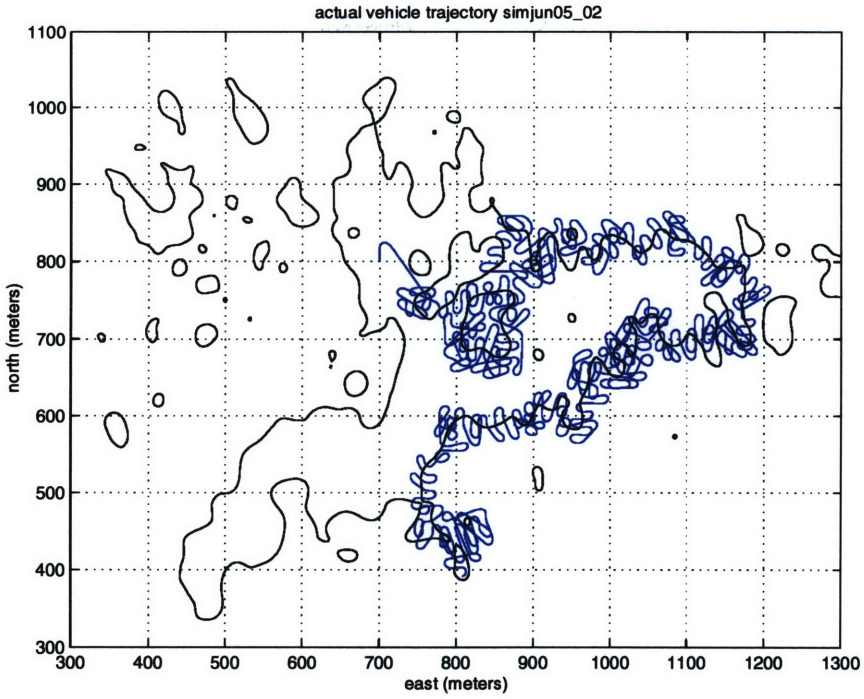


Figure 4-38: Close up of vehicle path from a mission using a lookup table which favors a particular directionality. In this mission, progress which leads the vehicle along its current general heading is favored. This mission mapped the 6 m contour in the Charles River basin.

4-39. In Figure 4-38, the more directional lookup table resulted in a path which gradually worked its way around the main feature, but missed the smaller detached features in the process. In FIgure 4-39, the non-directional lookup table resulted in more exploration in the immediate area, which resulted in less exploration of the main feature but more of the nearby smaller features being discovered.

Cost function

We can replace the pre-compiled competence of the lookup table with a cost function that compares the relative merits of each candidate waypoint. The question then becomes what to use as a cost metric? The metric chosen should reflect some important aspect of the vehicle, its performance or its environment, which we wish to keep to a minimum. This can be any individual characteristic of the vehicle, the sensed environment, or some combination of these.

In the case of the AUV *Odyssey II* exploring an unknown region, our desires are

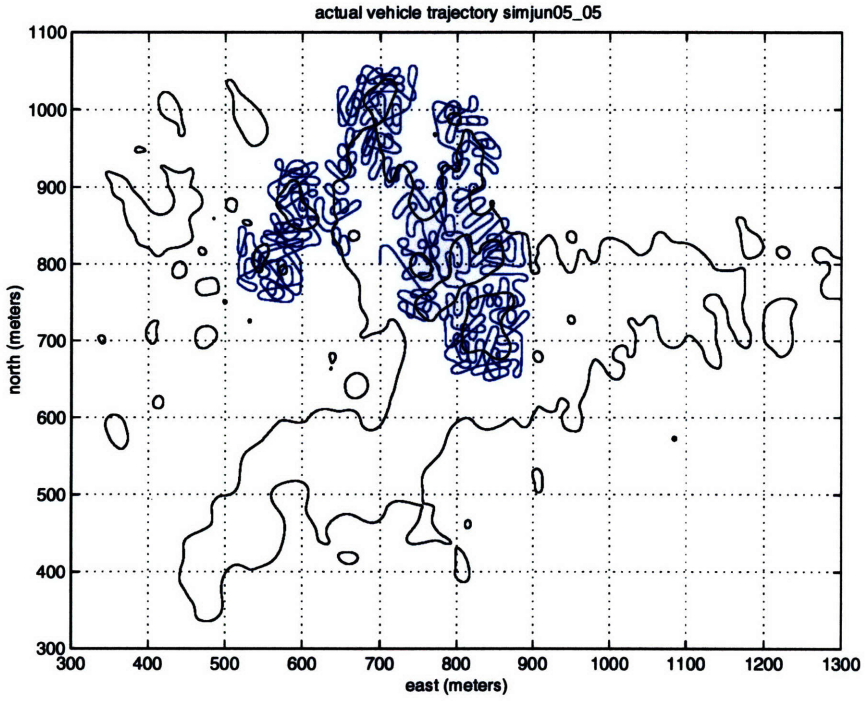


Figure 4-39: Close up of vehicle path from the same mission as in Figure 4-38 but using a lookup table which does not favor any particular direction. The vehicle makes less progress around the feature, but the non-directional design of the lookup table causes it to explore the surrounding area more. This results in more of the smaller, isolated features being discovered early in the mission. This mission mapped the 6 m contour in the Charles River basin.

that the vehicle acquire as much information about the feature as possible within the limitations of the available power supply. Furthermore, in the event we are relying on a dead-reckoning system, we wish to minimize the error buildup in the DR navigator.

It is the nature of non-holonomic, dynamically controlled underwater vehicles to slow down and side-slip when turning. This makes such maneuvers undesirable from the perspective of both dead-reckoning navigation and the desire to cover the maximum path length over the feature in the time allotted. A cost function which penalizes turning would therefore be desirable. However, a function which only penalizes turns would result in a vehicle which always ignores potential features to either side in order to avoid turning to explore them. Clearly there must also be a cost associated with traveling too far in a straight line as well. We have therefore chosen the cost function

$$\begin{aligned} \mathcal{W} &= min\left\{\alpha\sqrt{(X_{wp}-X_{veh})^2+(Y_{wp}-Y_{veh})^2}\right. \\ &\quad \left.+\beta\left|\theta_{veh}-tan^{-1}\left(\frac{Y_{wp}-Y_{veh}}{X_{wp}-X_{veh}}\right)\right|_{0<\Delta\theta<\pi}\right\} \end{aligned} \tag{4.1}$$

where α and β are weights which alter the relative cost of turning vs. distance to candidate waypoints. When using the cost function, the vehicle determines where to go next by compiling a list of candidate waypoints as explained in Section 3.7.2. It then ranks these candidates by cost value $\mathcal{W}$, low to high, and proceeds to visit each waypoint in sequence. Every time the on-board map is updated, the list of candidate waypoints is regenerated. When all candidate waypoints are exhausted, the vehicle begins to search for new features using a standard spiral pattern. Searching is not subject to cost-function calculations.

The results of this cost function are shown in Figures 4-40 through 4-43. Figure 4-40 shows the basic waypoint-directed search. Figure 4-41 shows the same search using a cost function where the ratio of the weights α and β is 1:1. Figures 4-42 and 4-43 use weighted heading to distance ratios of 50:1 and 1:50, respectively. As expected, penalizing heading over distance results in a search with fewer overall heading changes, while penalizing distance over heading changes results in a survey which stays close to the initial point of contact with the feature.

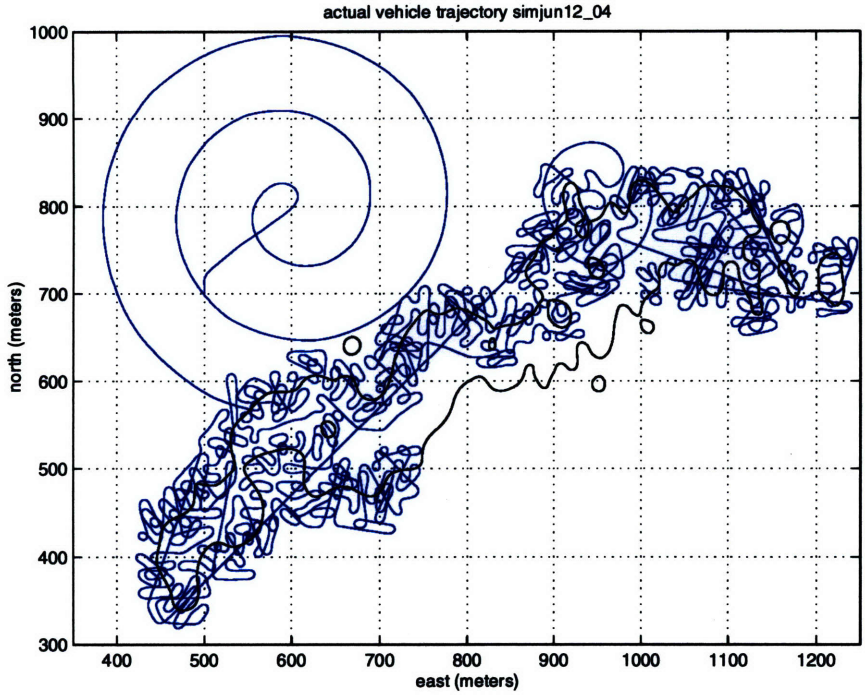


Figure 4-40: A typical waypoint-directed mission. This mission shows the vehicle launched from [700 N, 500 E] and proceeding to search from [800 N, 600 E]. It first locates the trench at approximately [550 N, 550 E] and proceeds to map the feature.

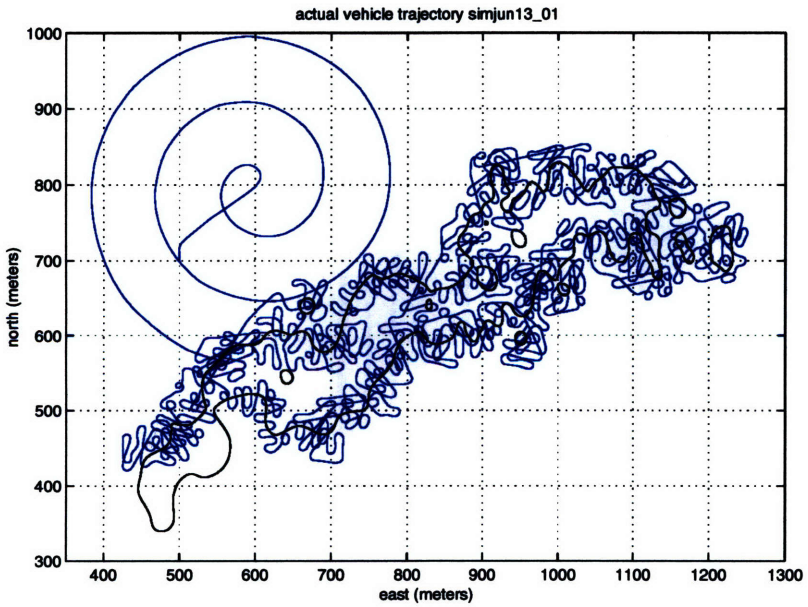


Figure 4-41: This mission is the same as that shown in Figure 4-40, except that waypoints are ranked by cost function rather than by lookup table. Note that the overall mission is more efficient than the waypoint directed mission, with more of the feature located in the same total mission time and fewer long transits to visit missed candidate waypoints.

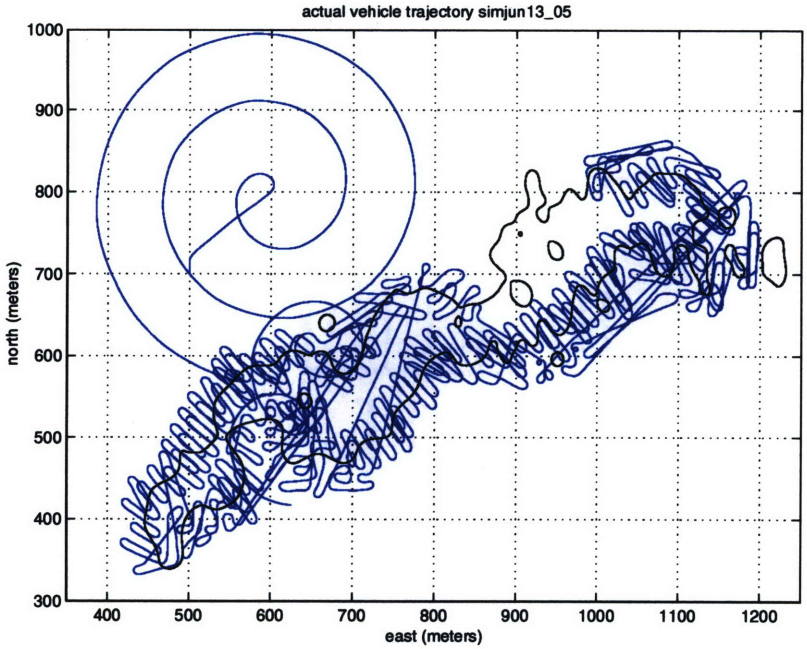


Figure 4-42: In this mission, the ratio of heading to distance is 50:1, making turning very "costly" when ranking waypoints. The resulting vehicle path shows much longer stretches between turns as the vehicle attempts to avoid the more expensive act of turning.

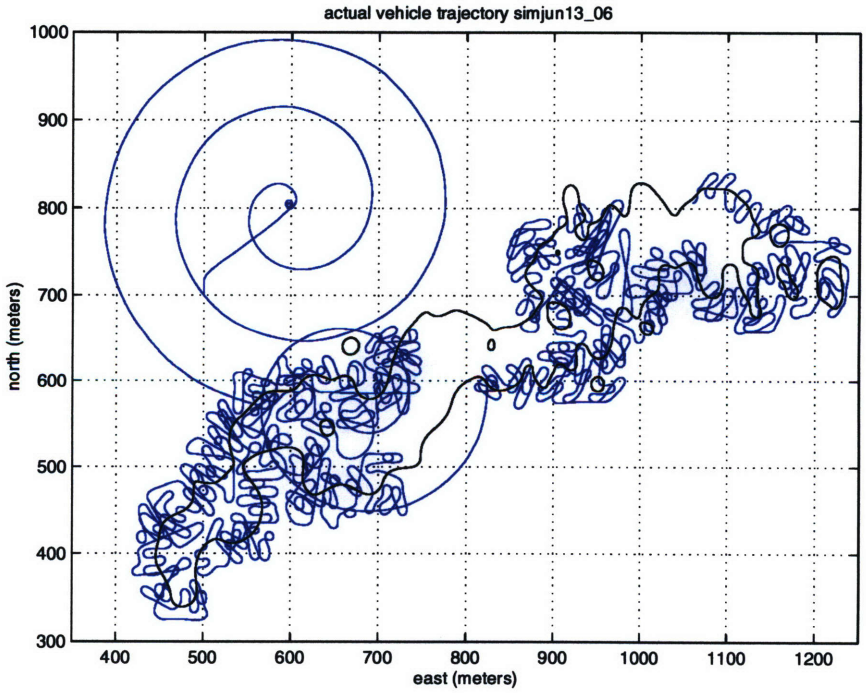


Figure 4-43: Here the ratio of heading to distance is 1:50. Turning is therefore not as costly as in Figure 4-42, resulting in shorter "runs" over the feature and more frequent changes in heading.

Adjacency search radius

Figure 4-29 shows a mission conducted with a search radius of r = 2 cells. Figure 4-44 shows the effect of decreasing the radius of the adjacency search from two cells to one in the same mission (all other parameters are kept constant). The effect of reducing the radius of the adjacency search is to reduce the number of candidate waypoints relative to each known feature. An increased search radius results in more candidate waypoints to be investigated around each known feature element and therefore more mission time being spent on a thorough search around each feature element, resulting in a less general search of the total assigned region. Reducing the search radius (and corresponding number of candidate cells surrounding a known feature cell) results in a less thorough search of any given feature cell's surroundings, and consequently leads to a more dispersed search overall.

Figure 4-44 shows the vehicle path when searching for the 7 m contour in the Charles River basin, using a search radius of r = 1. This means that only waypoints which are immediately adjacent to a known feature cell are considered. Fewer potential waypoints to visit mean that the vehicle executes more frequent spiral searches.

4.4 Summary

In this chapter we have examined the adaptive feature mapper from the perspective of overall efficiency and robustness. A set of metrics were developed which allowed us to compare the effects of environmental, system, and parameter changes. The effects of these changes were presented to demonstrate the overall and relative efficiency of the adaptive behavior `trench_finder`.

In the next chapter we explore the possible extensions to the current approach, how these extensions can be implemented, and what their potential applications may be. In particular we explore the effects of moving and/or time-evolving features, multiple vehicle tactics, using previously explored features as navigation aids, and the extension of this navigation paradigm into the realm of concurrent mapping and localization (CM&L).

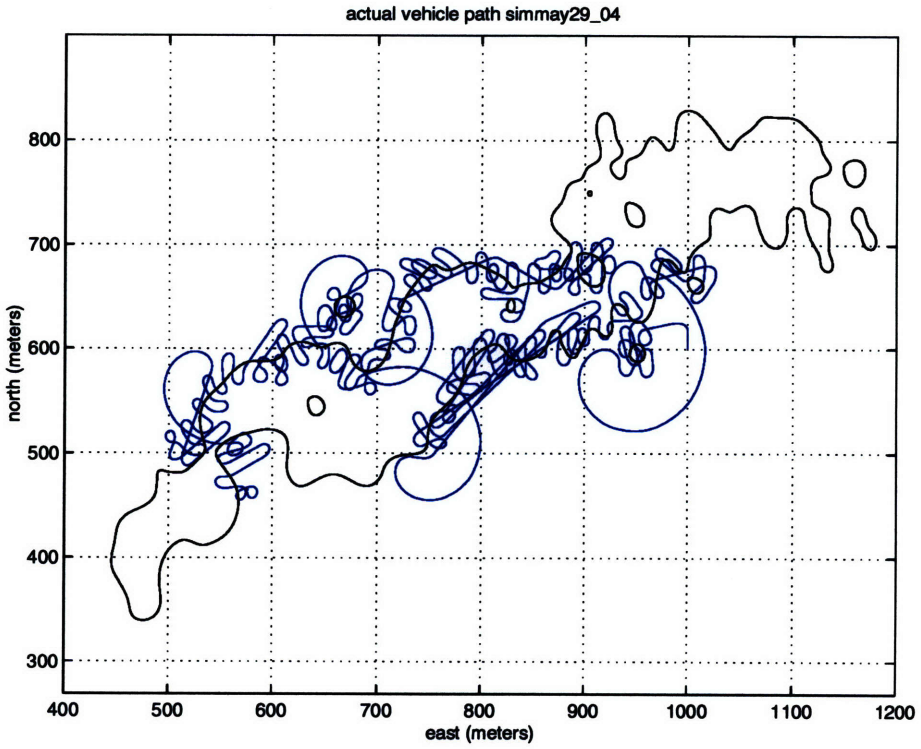


Figure 4-44: Vehicle path resulting from mission with search radius r = 1. In this mission, the vehicle has a search radius of r = 1. This results in fewer potential waypoints and thus the spiral search pattern is invoked several times to acquire more feature elements. Compare to Figure 4-29, with a search radius of r = 2.

Chapter 5

Extensions

This chapter takes the system designed and presented in Chapters 3 and 4 and proposes improvements to the feature relative navigator, the simulation environment, and extensions into the three principal research areas of navigation, mapping and intelligent control. We also examine extensions into the realms of dynamic features, feature identification and concurrent mapping and localization.

5.1 Extensions to the feature relative navigator

In Chapter 1, we motivated our interest in feature relative navigation with three scenarios. The first, mapping multiple bathymetric features in an unknown environment, was the primary focus of this thesis. We now examine the issues involved with extending the results of this thesis into the realm of dynamic features (Scenario 2) and into different sensor modalities (a necessary step towards Scenario 3). We also discuss the benefits of improved map representations and the effects of uncertainty as well as the changes that can be made to accommodate these effects.

5.1.1 Dynamic features

This thesis focused on bathymetric features for reasons of repeatability, testability, and proximity:

Repeatability - bathymetric features are static and therefore repeatable during tests. This insures that any changes to vehicle behavior are due to changes in the intelligent controller and not in the environment.

Testability - the assumed vehicle of opportunity is an *Odyssey IIb* class AUV. Besides mission-specific sensors, this type of AUV is equipped with a standard sensing package of X-, Y-, and Z-axis accelerometers, heading sensor, pitch & roll sensors, a depth sensor, and a sonar altimeter. The presence of this default package is sufficient for bathymetric measurement operations, regardless of any other mission packages the vehicle may be carrying. Since the sensors are present and operating during all missions, the `trench_finder` behavior can be run and tested in the background during any mission of opportunity without being given the priority necessary to control the vehicle. This allows the behavior to be field tested whenever the opportunity presents itself.

Proximity - we have shown that the Charles River has interesting topography and it is also close to MIT, making it a convenient and desirable testing facility. Indeed, Dr. Harold Edgerton regularly used the Charles River basin for acoustic testing [41, 42].

We now examine the issues involved with locating and tracking dynamic features in the ocean environment. To clearly list and examine the issues involved, we will decompose the problem of dynamic features into a series of steps of increasing complexity. These steps are moving rigid features, simple changing dynamic features, and complex changing dynamic features. Of the three, moving rigid features is the most interesting. This is because it is both a necessary and important foundation for the second and third steps, and also because it is of interest and use in itself.

Moving rigid dynamic features has a real-world analogue in the problem of locating and tracking a man-made object in the ocean. A specific example is locating and tracking a submarine or surface vessel with a magnetic sensor. In Section 1.1.2 we discussed a simple dynamic feature tracking scenario and showed the ability of the `trench_finder` adaptive behavior to track a rigid feature while subjected to an ex-

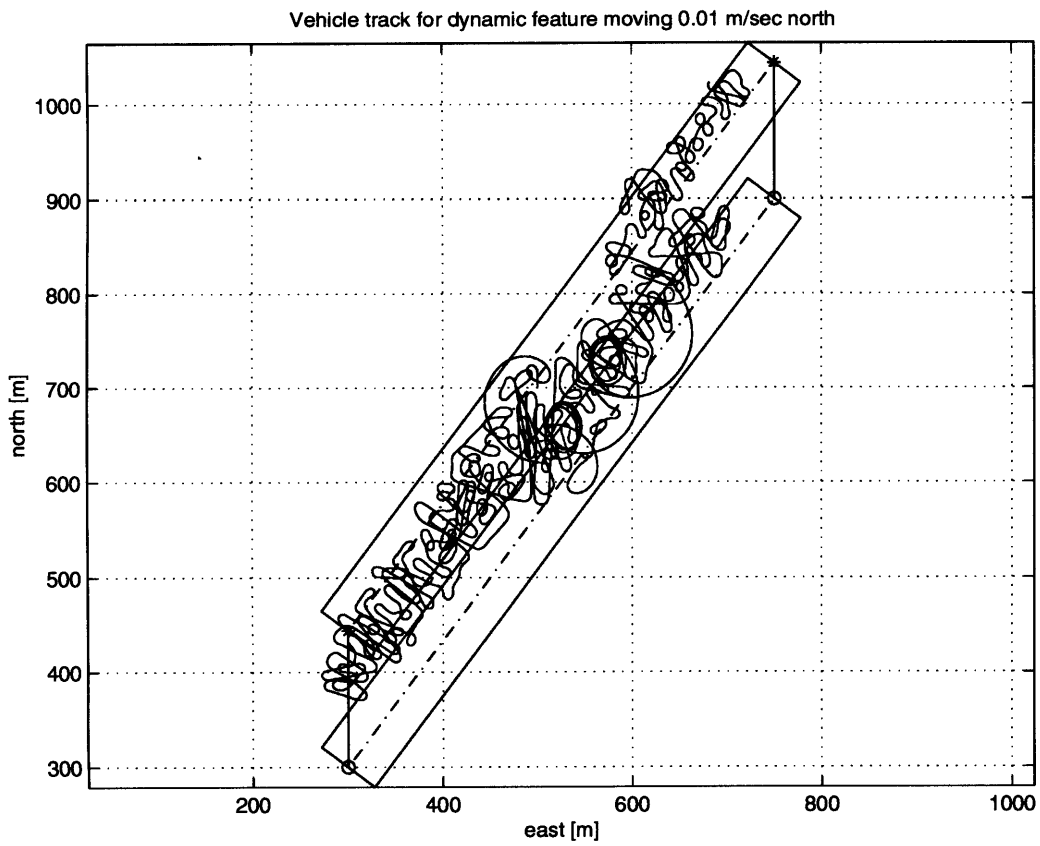


Figure 5-1: Path of `trench_finder` attempting to track a feature as it drifts to the north at 0.01 m/s.

ternal current, causing the vehicle to drift over the feature (see Figure 4-18). We now reverse the situation and attempt to track a "drifting" rigid feature in the absence of a current. Early work with a dynamic feature has shown that the vehicle is capable of tracking a slowly moving feature using the present software configuration, as shown in Figure 5-1. However, the resulting map generated by the vehicle (which assumes a non-moving feature) is meaningless since the vehicle map assumes static features.

What is needed is a method of reliably tracking a moving feature while at the same time maintaining a relatively accurate map of that feature and its location in the world. To perform this task, the vehicle must be capable of not only tracking a feature, but also predicting where that feature will be in the future. This insures that the vehicle will extend its search in the appropriate direction if or when the feature is lost and needs to be reacquired. To predict where a given feature will be, the vehicle

must have more information about the feature. This means that in addition to the characteristics of the feature, the FRN must also have some internal *model* of the feature and its expected behavior.

Model-based adaptive feature mapping

In the moving feature example above, the current FRN was shown to be able to track simple, slow-moving rigid features. This capability will quickly break down in situations where the vehicle turns the opposite way from the direction of travel of the feature or when the moving feature travels into regions already considered mapped by the vehicle. This is due to the fact that the FRN assumes static features and therefore treats those portions of the moving feature, whether visited before or not, as portions of a larger static feature. In addition, the FRN assumes the feature to be static. Therefore, if the feature moves into areas considered already mapped, the vehicle will cease to follow the feature because the map would have indicated the area the feature is entering to be known and hence no longer of interest. To solve this problem we need to take the concept behind the adaptive feature mapper to the next logical step: model-based feature relative navigation.

In Chapter 3 we showed the evolution of the adaptive feature mapper from a simple reactive behavior to a multi-state reactive behavior and finally to a behavior which incorporates the concepts of mapping (memory) and path planning (prediction). The prediction capability of `trench_finder` is optimized for static features. It is designed to predict where more of a given feature may be found based on the current knowledge available. From another perspective, we can observe that the adaptive feature mapper determines where more of a feature may be located by using an *internal model* of the feature. In the case of `trench_finder` this model makes the following assumptions: 1) the feature is static and 2) the feature is continuous. It uses these assumptions to make predictions about where to look for more of a feature.

In the case of moving features, the type of feature can be extended by the addition of motion, at first as a rigid body, but eventually we will wish to alter the overall shape and volume as well. This new feature definition requires a corresponding addition to

the feature relative navigator in the form of an explicit predictive model. The new *model-based* adaptive feature mapper will then use this on-board model of a feature to predict what will be the best path to follow in order to locate and sample more of a moving, changing feature. This model may be as simple or as sophisticated as we wish, incorporating any dynamic information or sensor data prediction capabilities that may be required. For example, in order to track a thermal plume rising from a vent field, we load a model of a thermal plume into the vehicle. This model will then take the sensory information as it becomes available and predict what temperature, gradient, and motion might be expected – based on the data obtained thus far. The vehicle then uses these predictions to locate more of the feature, which in turn improves the accuracy of the predictions.

For a model-based adaptive feature mapper to be successfully employed in the FRN, we first need to extend the cost function currently installed by making utility an explicit part of the system. It is currently implicit in the form of the candidate waypoint cell list. All cells in the list are presumed to have an equal relative utility, thus making the cost of visiting each cell the deciding factor. Combining cost and utility directly allows us to make net-utility calculations based on the feature model and its predictions. The new adaptive feature mapper then chooses where to go next based on the net utility of each cell.

This would be done by first predicting the most likely locations where the moving feature might be found and indicating these locations in the on-board map. The likeliest locations would then have the highest utilities, making them the most attractive locations to visit. This utility would then be balanced against the cost of visiting each cell, yielding a net utility which could then be used to determine which cells should be visited and in what order. As new information about the world is obtained, cell utilities and costs would be re-ranked, thereby keeping the vehicle continually updated about where to go next.

5.1.2 On-board map improvements

The current on-board map uses a cell-based representation of the world because of its ease of use and expandability. However, by pixelating the world, we are also sacrificing detail. We can avoid this loss of detail at the expense of increased processing load by incorporating a feature-based representation (the "high-level" representation discussed in Section 2.3) of the world in addition to our cell-based map. Feature-based maps require more processing capacity since they do not store positional information explicitly, as in a cell-based map, but they are more versatile in that each feature can be assigned point-specific values, such as positional uncertainty or relative utility. If we wish to know whether to visit a given cell, we take the utility of each of the objects in the area and spread their utility over the area defined by their associated positional uncertainties. We then deduct the cost of visiting that cell and thereby obtain the net utility of visiting any given cell. This results in a net utility for each cell which we then treat in the normal fashion. This positional representation also relates back to the concept of model-based feature representation. As our on-board model makes new predictions about the location of target features, those new positions (and any associated changes in uncertainty) can be fed back into the utility calculations by updating the information about those specific features. This new information is reflected in the next round of relative utility calculations.

This concept of spreading the net utility over a set of cells ties back into the concept using a metric of merit (such as a confidence value) which is distributed over the map. This is analogous to the idea of using relative uncertainty, as found in Stewart [107]. The important difference being that since the utility and positional uncertainty is tied to each specific feature, it is not cumulative over the life of a cell and therefore each feature and its associated positional uncertainty can be separately tracked and calculated. The advantage to this is that new information about a specific feature (such as reducing positional uncertainty) will result in a more accurate map every time it is regenerated. The drawback is that by keeping each feature separate, we are forced to recalculate the net certainty and/or utility for each map as required,

which increases the computational load on the vehicle.

5.1.3 Other sensor modalities

The FRN system presented in this thesis was developed with the assumption of a point sensor mounted on a non-holonomic autonomous underwater vehicle. This assumption was made to facilitate the development of an FRN which would be applicable to the broadest class of sensors (such as magnetic, gravometric, thermal, and salinity sensors) and vehicles of opportunity (survey-class AUVs).

If we relax this assumption and assume the use of sonar as the sensing modality of choice, then we can exploit the remote sensing capabilities of a steerable sonar beam to improve the performance of the FRN system. Doing so offers the advantages of improved vehicle trajectories – the vehicle does not need to be at the sensing location in order to obtain information about that point in space. This reduces the maneuvering requirements of the AUV which reduces both power consumption and error growth rate in dead reckoning brought about by unmodeled motion during turning maneuvers.

The cost of such sensing systems are a) the increased power requirements of the more capable sensing systems and b) increased processing requirements brought about by the larger dataflow from the sensors. If the processing and power capacities are available, then the approach presented in this thesis can be extended to accommodate such systems by incorporating the sensed location relative to the vehicle, taking into account any positional inaccuracies or averaging brought about by the type of sensor modality employed.

5.1.4 Relocation

There is a large body of existing work using point-type features for the purposes of relocation. We propose a method of navigation using distributed natural features; both contour-based and area-based. Key parts are locating and identifying these features so that they can be used. Assuming the FRN system is directed to map

static features, it may also be usable as an aid to navigation for systems such as INS and dead-reckoning. In Section 1.1.3 we presented the idea of relocation, using the reacquisition of a previously mapped feature to limit the growth of navigational error in a vehicle relying on an inertial navigation or dead-reckoning navigation system.

The method of relocation depends upon the type of feature being detected. As mentioned in more detail in Section A.1, we can divide our approach to features into two classes: region-based and contour-based features. The type of feature directs not only how it is initially mapped, but also how it can be reacquired at a later time and how to efficiently direct the vehicle once the feature has been located.

In all cases, the initial mapping is conducted as presented in Chapters 3 and 4. Once the feature has been mapped, however, it can be considered *a priori* information from the perspective of reacquisition. This means that a more carefully planned strategy can be employed when reacquiring the feature for the purposes of reidentification and navigational reset.

An example is the mission shown in Figure 5-2. In this mission, dead-reckoning navigational errors resulted in the vehicle remapping a previously mapped portion of the feature. The resulting dead-reckoner-derived contour line is shown in Figure 5-3. Note that the portion of the contour line which has been remapped contains portions which are recognizable as the same feature; specifically the westernmost portion of the feature which is again mapped at the final, southernmost, portion of the mission (for reference purposes the reader is directed to Figure 4-25 for a view of the actual vehicle path). A human can quickly identify this portion of the contour as being the same place. The goal then is for the vehicle to also make such an identification. Having done so, the navigational error can be correspondingly reduced.

Learning directed relocation

When attempting to reacquire a feature, the vehicle can use prior information to direct the search to re-map an area. A given pair of feature (or feature/adjacency) cells will be somewhat unique. A larger set of cells is even more unique (this is the basic idea behind the TERCOM navigation system mentioned in Section 2.2.4). The

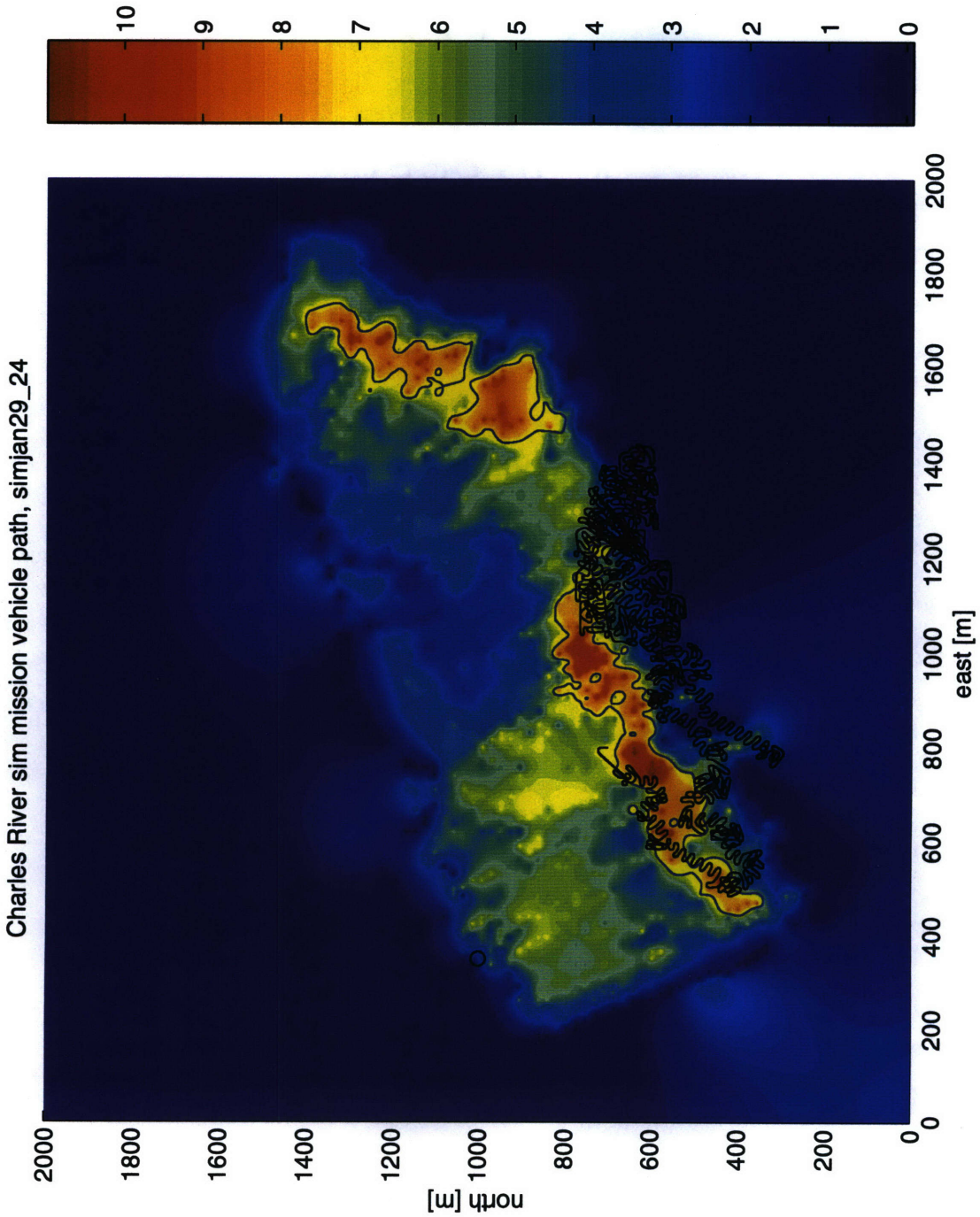


Figure 5-2: Dead Reckoned vehicle path from mission shown in Figure 4-25. Note that the first leg of the mission (the northwest portion of the vehicle path) has been remapped by the vehicle at the end of the mission (the south/central, terminating, portion of the vehicle path).

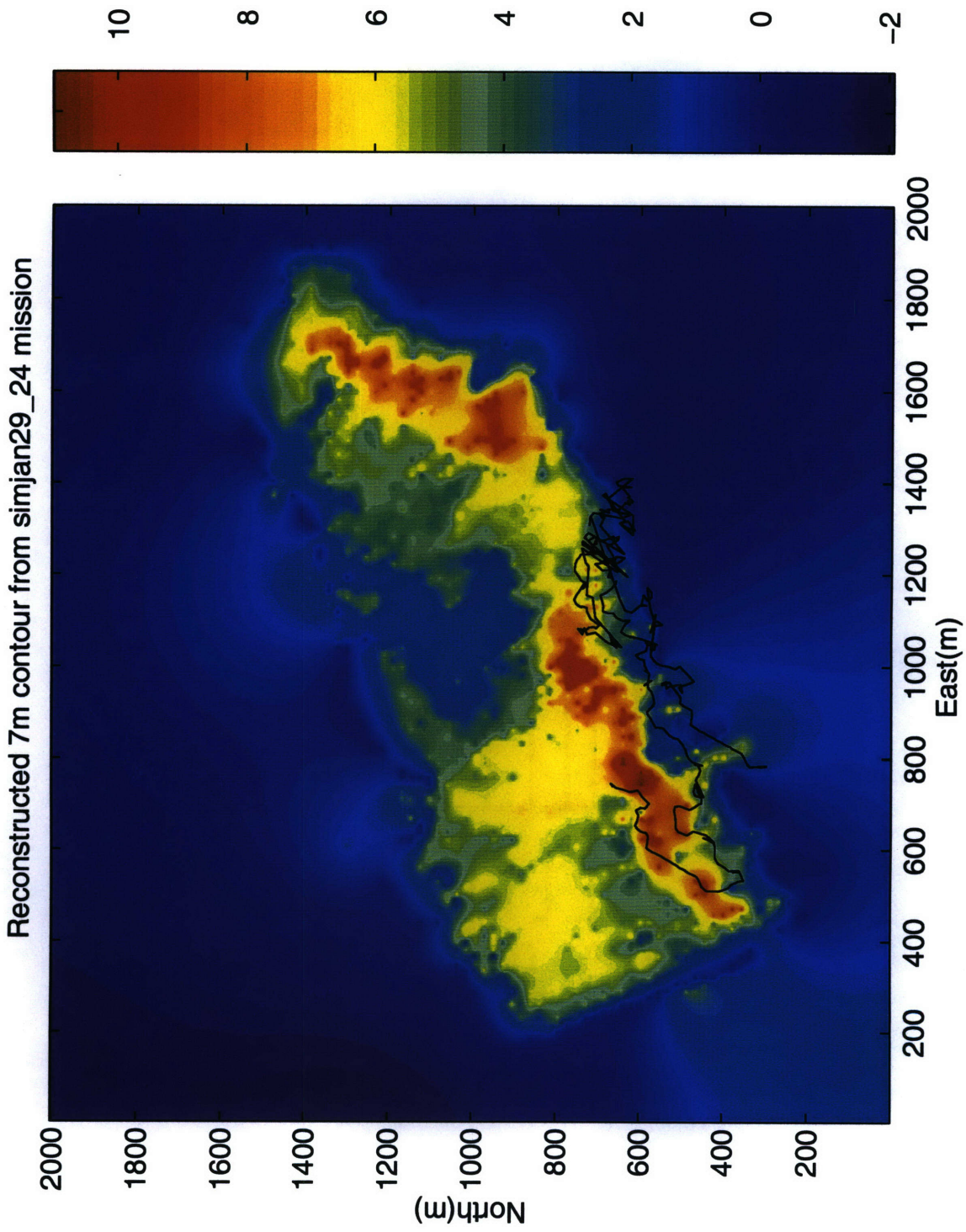
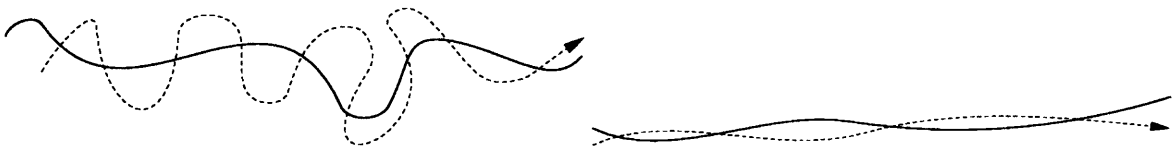


Figure 5-3: Dead Reckoned vehicle path from mission shown in Figure 4-25. Note that the first leg of the mission (the northwest portion of the vehicle path) has been remapped by the vehicle at the end of the mission (the south/central, terminating, portion of the vehicle path).



(a) Tracking a high-curvature contour (b) Tracking a low-curvature contour

Figure 5-4: *A priori* knowledge of the feature allows for prediction of the relative curvature of the contour, thereby enabling the intelligent controller to generate a more efficient contour tracking path. In these figures, the solid line represents the feature and the dashed line represents the vehicle track.

vehicle can then perform a pattern match of the cells to the map and generate a list of candidate sites where such sets of feature cells exist. Using this list enables the controller to predict the likeliest candidate cells to visit next to confirm or eliminate a possible matching region or feature. This progressive template matching continues until the feature is uniquely identified or determined to be a new feature.

Consider a case of a region-based feature: the vehicle is attempting to relocate a trench and approaches the area where a previously mapped trench is believed to exist. Upon sensing the first feature cells, the vehicle confirms that it has indeed reached one side of *some* trench, but with no idea where along the trench yet (or even if it is indeed the correct trench). Using the feature map gathered earlier in the mission, the intelligent controller can then direct the vehicle to cross the trench from one side to another and compare this cell-profile set to all possible profile sets for that particular trench. If a match is found, then the vehicle has been relocated. If not, then the feature may be new or some mapping error may have occurred. In either case, more information about the trench is required and so the controller directs the vehicle into a mapping of the entire feature.

If the feature is a contour, the vehicle can exploit its prior information about the local curvature of the contour in order to facilitate a more efficient feature sampling path (see Figure 5-4). Once the contour has been sampled, local extrema of curvature are extracted and compared to equivalent regions on the previously mapped contour [72, 81].

5.2 Improvements to the simulation environment

The current simulation environment is designed to model bathymetric features as observed by a single downward-looking sonar altimeter. The datafile employed can be either a Delaunay-triangulated file, a regular grid of depth values, or an array of [X,Y,Z] coordinates. Other sensors modeled include temperature and salinity sensors. Both of these sensors are based on a simple stratified ocean model (i.e. a "layer cake" model). Improvements to the FRN system will necessitate improvements to the simulation environment.

5.2.1 Improved sonar model

Earlier in this chapter we discussed the potential advantages of sonar as a remote sensing modality which would enable the FRN to map a feature with reduced maneuvering requirements. Testing this capability in simulation requires additions to the simulated environment and sensors.

The environment simulation, as implemented, assumes a single downward looking sonar altimeter. This is accomplished by simulating the mounting location and orientation of the sonar on the vehicle and taking into account the vehicle orientation in space, the sonar beam pattern, and the geometry of the bottom where the beam intersects the bottom model. We can implement the tools necessary for a sonar-specific FRN by first implementing a multiple fixed-sonar model, such as an obstacle avoidance (O/A) sonar arrangement, followed by a model of one or more steerable beam models.

The O/A sonar model has already been implemented but remains unused. This model exploits the primitives established by the single sonar altimeter model, but changes the mounting location and orientation of the sonar. Using this method allows any number of pencil-beam sonars to be mounted at any position on the vehicle. Extending this model to a scanning-beam sonar is somewhat more complex.

While the mounting of a scanning beam can simply be modeled as a time-varying positional mount of a standard fixed-beam sonar, there is potentially a difference

in the beam pattern and its representation. Adequately modeling the physics of a mechanically scanned pencil-beam sonar can be done with only small extensions to the pencil-beam model currently employed. An electronically-scanned sonar is somewhat more complex in nature and requires a more detailed acoustic model than the simple model currently installed. Fortunately, such sonars are well understood and have been adequately characterized for use in simulation environments. There are therefore no obstacles to the development of a more detailed acoustic model, if desired.

5.2.2 Dynamic features

A simple dynamic feature in the form of a drifting trench has already been incorporated into the bathymetric model. This addition is only useful for representing one or more linear trenches moving in a simple planar field. The next step is to replace this simple rigid feature with a more dynamic one. This can be accomplished in two steps. First, the simple trench must be replaced with a feature which can alter its shape and size over time. As a first approach this can be a simple geometric feature. This allows us to describe the feature as a set of vertices with trajectories associated with each vertex. The moving vertices can then be set to move as a group (representing a rigid object such as a submarine, a ship, or a whale) or each vertex can be set to move independently (representing a simplified form of a more complex spreading phenomenon, such as an oil slick). It is believed that this model will be of use in the development of several FRN improvements, such as model-based feature tracking, feature re-recognition, and uncertainty representation.

Having done this, the next step is to replace the simple geometric model with a more realistic, three-dimensional model. Such representations could be derived from oceanographic models, such as the thermal/saline mixing model used in the HARO Strait/PRIMER experiment [43] or even spreading models currently used to predict the growth and distribution of oil slicks. These models are necessarily much more complex, requiring a more powerful system to calculate and feed the necessary sensory information to the vehicle in a timely fashion. However, it is believed that

such concerns would not be significant in relation to the potential benefits.

5.3 Summary

This chapter examined potential improvements to the FRN system as presented in this thesis. The two principal areas of potential improvement are the feature relative navigator itself and the environmental model used for development.

Extensions in the FRN include dynamic features, model-based representations, multiple map modalities (both cellular-based and feature based), other sensor modalities (such as steerable beam sonar), and the exploitation of previously obtained information via the process of relocation. Improvements to the simulation environment include an improved acoustic model in the form of multiple and/or steerable sonars and an improved dynamic environmental model in the form of deformable geometric or other more complex oceanographic-based models.

The next chapter summarizes the contributions made by this thesis and draws some conclusions based on the work presented.

Chapter 6

Conclusions

The goal of this thesis is to endow autonomous underwater vehicles with the capability to locate and adaptively map an unknown number of features in an unknown environment without the aid of *a priori* maps. To meet this challenge, we have developed an adaptive behavior, `trench_finder`, as an extension to the layered control concept. This new type of behavior brings the capabilities of mapping and planning to the realm of layered control, allowing the vehicle to adapt its trajectory in response to the presence or absence of a feature. In this chapter, we conclude this thesis with a summary of the contributions made and reiterate some of our insights on possible extensions into future work.

6.1 Contributions

Although this thesis utilized a sonar altimeter scanning bathymetric features as our model, the approach formulated here can be directly applied without modification to any vehicle employing a point sensor. Extension into remote sensing modalities such as a scanning sonar can also be accomplished with minor modifications. As presented, this thesis has three principal contributions:

- A technique for adaptive feature mapping which employs behaviors possessing the capabilities of mapping and path planning. Unlike conventional reactive

behaviors, adaptive behaviors alter their internal state as new information about the environment is received. This in turn allows the behavior to alter its output in response to the current situation.

- An analysis of the performance and robustness of the adaptive feature mapper – examining time-on-target, total path length needed to map a feature, amount of feature mapped per unit time, amount of feature mapped per meter traveled, and performance of the adaptive behavior in comparison to a conventional lawnmower-type survey.

- We also present extensions of adaptive behaviors into the realms of adaptive mapping of dynamic features, multi-vehicle adaptive mapping, alternative mapping techniques employing pre-compiled lookup tables, cost functions and decision-theoretic techniques, and navigation applications including relocation and concurrent mapping localization (CM&L).

6.2 Motivation

This work was motivated by three examples: mapping of ice-keel trenches, adaptive mapping of dynamic features, and concurrent mapping and localization. The first two examples are drawn from present-day, real world problems. The third is an area of study whose development promises to increase the range of AUV capabilities.

The problem of ice-keel trench mapping is representative of a large class of problems where one or more features are located in an area where *a priori* maps are unavailable, either because the area has never been mapped or because it changes too frequently for the creation of reliable maps. Missions which fall into this category include mapping thermal vent fields, ocean dumping sites, runoff and outfall areas, and locating man made objects (barrels, mines, etc.).

The Haro Strait experiment was discussed as an example of adaptive mapping of dynamic features. The objective of this type of mission is to locate and examine a feature whose dynamic nature makes its shape, extent, and location impossible to

predict in detail. Adapting the vehicle path to follow such a feature allows us to both map a feature whose *a priori* location is unknown and to follow a feature as it moves through or with the water column. Examples of this type of mission are mapping of mixing zones, rapid-response situations, locating fronts (thermal, salinity, turbidity, etc.) and tracking dynamic features such as upwellings, thermal plumes, or eddies.

The third application, concurrent mapping and localization, is a technique whereby a vehicle continually maps its surroundings and uses that map to limit navigational error growth. Such a competency would allow an AUV to navigate in regions where no *a priori* maps or navigation arrays exist. This would greatly increase the autonomy of AUVs and allow their use in situations where conventional navigation systems cannot or need not be employed.

To achieve this, several advances in intelligent control of AUVs must take place. The contributions of this thesis, feature-relative navigation, is a necessary and important step.

6.3 Implementation

The approach we have chosen is based on state configured layered control, a form of behavior-based intelligent control which has been created in response to the problems of scaling and situatedness. This approach showed promise in the early stages, although it could be easily misled by noisy data and was incapable of multi-phase missions.

The addition of state configured layered control eliminated the problems encountered by the first generation system, and proved robust when tested on a variety of target features. In spite of this, it too proved lacking due to a tendency to oversample some regions while undersampling others.

The final form of the feature-relative navigator overcame this limitation with the addition of mapping and planning capabilities. Cell-based feature and visitation maps are created by the vehicle and updated whenever new information is obtained. Utility and cost-based decisions are made on where to go next to obtain more information

about a feature or to locate new features. The addition of memory and prediction enabled the vehicle to systematically map any assigned feature, as well as to know when to begin a search for more features. This new adaptive behavior met the challenge put forth by the committee: generate a map of the trenches in the Charles River without the aid of *a priori* maps.

6.4 Performance metrics

Having successfully met the challenge of the committee, the next step was to describe and quantify the performance of the system. While universal metrics of performance for intelligent systems do not exist, it is still possible to make qualitative judgments about the overall performance of the system when operating with different parameters and under various environmental conditions.

Metrics were developed in the form of total path length traveled, number of feature elements located overall, number of feature elements located per unit time (or distance), and overall time-on-target. These metrics were compared to a standard lawnmower-type survey and also between missions to compare the relative effectiveness of various parameter settings.

Parameters such as map cell size, adjacency search radius, and cost function weights were altered and their effects examined. Search zone and area- vs. contour-based surveys were examined and compared. Different forms of navigational error and currents of varying strength were also examined and their effects analyzed.

The results of this analysis of robustness and efficiency showed the feature-relative navigator to be capable of locating and tracking a feature in the presence of relatively large disturbances and even in the presence of large navigational error.

6.5 Issues for further research

The feature-relative navigation system presented here is a first step in a promising line of research. This is illustrated by the number of potential improvements and new

capabilities which this new competency enables.

6.5.1 Improvements to `trench_finder`

Improvements to `trench_finder` include probabilistic forms of feature representation, map generation, and map maintenance. The cost analysis function can be expanded with the inclusion of extended decision analysis tools. Extensions to feature-relative navigation include feature-based descriptions, relocation capabilities, and dynamic feature tracking.

Probabilistic forms of representation and information manipulation could include the addition of feature-based probabilistic data association, cell-based stochastic back projection, and extension of these techniques to voxel-based representations.

The addition of feature-based descriptions opens the way for feature recognition and from that comes the potential for relocation. Relocation in turn is a necessary step towards the goal of concurrent mapping and localization.

Finally, while this approach was developed with a non-holonomic, survey class vehicle in mind, it would be educational to examine how the system performs on a holonomic (i.e. zero turning radius, hover-capable) vehicle.

6.5.2 Implementation on board an AUV

The simulation environment used in this thesis was based on real bathymetric data. However, it is still a simulation and subject to limitations. There are always unforeseen problems which must be dealt with when moving from the world of simulation, no matter how realistic in appearance, to the world of actual robotic vehicles.

The `trench_finder` adaptive behavior has been designed in the same dynamic simulation environment used to develop behaviors for the *Odyssey II*-class AUVs. It should therefore be relatively easy to port the behavior into the actual vehicle for field testing in the Charles River basin. The field tests would prove invaluable for revealing any limitations and/or unforeseen complications which a purely simulated environment may have hidden.

6.5.3 Simulation environment

The simulation environment employed for the development work in this thesis is based on real-world bathymetric data and employs an acoustic sensor model based on a standard sonar altimeter. This environment, while adequate for the purpose for which it was designed, could be extended and improved in several ways. Among the potentially most rewarding are a more extensive sonar model, the addition of non-acoustic sensor modalities, an improved bottom model, and the capability for dynamic, evolving features.

The current sensor model employed is based on a conical-beam sonar altimeter. The extension to multiple- and/or a steerable-beam sonar model opens the door to other, more sensor-specific applications. These include the capability to look ahead and to the sides of the vehicle, and to focus attention upon a specific location using a steerable beam. The inclusion of non-acoustic sensor modalities such as magnetic, thermal, salinity, or gravimetric sensors would allow for testing of the `trench_finder` using other environmental models – plus the possible use of multiple sensor navigation techniques (following in the footsteps of Tuohy [111]).

Finally the bathymetry in the simulation environment is a faceted model based on a Delaunay triangulation of irregularly spaced bathymetric data. If additional realism in the bathymetry is desired, then a fractally-enhanced bottom model could be included; such a model draws on the work of Goff & Jordan [48] and Dutton [40].

6.5.4 Feature representation

As mentioned before, the current form of feature representation can be extended by the addition of both a feature-relative representation and by the extension of the current on-board maps into three dimensions. The work presented here employs a two-dimensional representation of the feature of interest. Extension into three dimensions would enable the mapping and possible identification of more complex features, such as local magnetic disturbances, as well as complex dynamic features such as thermal plumes, upwellings, and mixing zones.

6.6 Summary

In this chapter we reviewed the main points presented in this thesis. We re-examined the motivating scenarios, the challenge presented by the committee, the approach taken to meet this challenge, and the results of that approach. We also re-examined the issues of robustness, efficiency, and metrics of performance. Finally, we reviewed the potential for extensions and improvements to the work presented here, including improvements to `trench_finder`, extensions based on `trench_finder`, field testing, the steps needed for feature-relative navigation, as well as improvements and extensions to the simulation and testing environment.

Appendix A

Implementation Details

A.1 Functional description of `trench_finder`

All vehicle missions are controlled via a mission script, as described by Bellingham [15]. In this section we will present a full mission script, but focus on that part of the mission script which pertains to the `trench_finder` behavior.

The `trench_finder` mission script is as follows:

```
P_state: one detect_type 0 (depth) 2 (contour)
  sensor:          speed_bias_noise(m/s) 0
  sensor:        compass_bias_noise(deg) 0
  sensor:        real_current_north(m/s) 0
  sensor:         real_current_east(m/s) 0
  sensor:             current_noise(m/s) 0
  sensor:          real_acc_noise(m/s^2) 0
  sensor:           c_weight1_drop(bool) 0
  sensor:           c_weight2_drop(bool) 0
  sensor:                  real_wdw1(N) 0
  sensor:                  real_wdw2(N) 0
  sensor:                 real_veh_w(N) 1173
  sensor:                 real_veh_b(N) 1173
  sensor:           c_modem_active(bool) 1
  sensor:                uo_depth_map_k 0.5
  sensor:            real_depth_noise(m) 0.0
```

```
  sensor:   real_sonar_frac_dropout(0-1) 0.0
  sensor:                      real_z(m) 1.5
  sensor:                    real_u(m/s) 1.4
  sensor:                uo_control_kpp 1
  sensor:                uo_control_kdp 0.9
  sensor:                uo_control_kip 0.2
  sensor:                uo_control_kph 0.5
  sensor:                uo_control_kdh 1.5
  sensor:                uo_control_kih 0.0
  sensor:             u_bottom_noise(m) 0.0
  sensor:             u_bottom_type(int) 7
  sensor:                      real_x(m) 600.0
  sensor:                      real_y(m) 1000.0
  sensor:            u_veh_init_north(m) 600.0
  sensor:             u_veh_init_east(m) 1000.0
  sensor:                  real_psi(rad) 0.0
  sensor:        u_veh_init_heading(rad) 0.0
  sensor:    real_lbl_timing_noise(mSec) 3
  sensor:         u_lbl_beacon_number(#) 4.0
  sensor:               u_lbl_1north(m) 500.0
  sensor:                u_lbl_1east(m) 500.0
  sensor:               u_lbl_1depth(m) 10.0
  sensor:               u_lbl_2north(m) 350.0
  sensor:                u_lbl_2east(m) 600.0
  sensor:               u_lbl_2depth(m) 10.0
  sensor:               u_lbl_3north(m) 200.0
  sensor:                u_lbl_3east(m) 500.0
  sensor:               u_lbl_3depth(m) 10.0
  sensor:               u_lbl_4north(m) 350.0
  sensor:                u_lbl_4east(m) 400.0
  sensor:               u_lbl_4depth(m) 10.0
behavior: mission_timer 1
  b_arg:                         time(s) 24000.0
behavior: trench_finder 2
  b_arg:                   north_lim(m) 1600.0
  b_arg:                   south_lim(m) 200.0
```

Symbol or variable	Description	Nominal value
$\begin{bmatrix} X_{west} & Y_{south} \\ X_{east} & Y_{north} \end{bmatrix}$	$WESN$ search zone boundaries.	$\begin{bmatrix} 200 & 200 \\ 2000 & 1600 \end{bmatrix}$
η	Search mode (Spiral, Lawnmower, Wagon Wheel, Reactive)	Spiral
α	Search Interval	15.75 m
$\mathcal{D}$	Detection type (scalar, gradient)	Scalar
τ	Detection threshold	7.0 m
δ	Deadband	± 2 cm
ρ	Waypoint adjacency search radius	2 cells
`data_filter`	Data filtering	Moving window
`cell_size`	Map cell size	10×10 meters
`waypoint_method`	Waypoint search method(LOOKUP, DIRECTIONAL, COST)	LOOKUP
`follow_mode`	Contour vs. region search	CONTOUR
$\begin{bmatrix} \mathcal{O}_{east} & \mathcal{O}_{north} \end{bmatrix}$	Search origin	[800, 700]

Table A.1: Inputs for `trench_finder` and nominal values used in the Charles River basin.

```
 b_arg:                    east_lim(m) 2000.0
 b_arg:                    west_lim(m) 200.0
 b_arg:                 search_mode(int) 333
 b_arg:           search_center_north(m) 700.0
 b_arg:            search_center_east(m) 800.0
 b_arg:               search_interval(m) 2.5
 b_arg:                 detect_thresh(m) 7.0
 b_arg:                 detect_type(int) 2
behavior: setpoint 3
 b_arg:                     heading(deg) 0.0
 b_arg:                        depth(m) 1.5
 b_arg:                      speed(m/s) 1.0
 b_arg:                         time(s) 24000.0
```

The `trench_finder` behavior inputs are listed following the `behavior: trench_finder 2` line in the mission script. Since it is part of a layered control system, functions such as vehicle depth, speed, and the mission timer are left to the control of other behaviors, such as `setpoint`. `trench_finder` is primarily a heading-control behavior.

The `trench_finder` behavior inputs are listed in Table A.1.

Search area - The boundaries of the search zone, in meters, from an origin chosen by the user at launch time. Unless otherwise stated, operations are assumed to take place in the first quadrant.

Search mode - When the vehicle is searching in an attempt to locate a feature, it can do so by one of several methods: a spiral pattern, lawnmower search, wagon wheel search (i.e. multiple legs radiating out from a central launch point - particularly useful when the vehicle has been launched from an ice hole), and a simple reactive search (i.e. the vehicle chooses a random heading and maintains it until a feature is found or a search area boundary is reached, then it turns back into the search area on a new arbitrary heading).

Search interval - Used for lawnmower and spiral searches; it is defined as the interval between laps of a lawnmower-type search and $\frac{1}{2\pi}$ times the interval between spiral laps, respectively. This should be set according to the characteristic dimensions of the feature. For most potential targets, it is recommended that the interval be set at no larger than one half the average expected short axis of the feature (if known).

Detection type - `trench_finder` is designed to detect gradient or scalar threshold values. In the case of a point sensor, the gradient is constructed from sensory information obtained along the vehicle's path (the user must therefore be aware that the gradient constructed by the vehicle is the one which is perceived by the vehicle along its track and not necessarily the steepest gradient). Note also that construction of the estimated gradient using a point sensor necessitates a delay due to the need to construct the gradient from present plus recent sensor data. Either gradient or threshold values are compared against filtered vehicle sensory data to determine if the current vehicle location is coincident with a potential feature of interest.

Detection threshold - `trench_finder` is designed to detect a specific value or gradient. This initial value is derived from basic *a priori* information about the environment. In the case of static bathymetric features, a specific contour value, bottom gradient, or mean seafloor depth is entered which acts as a threshold for the detector. In rapid response situations (e.g. satellite directed response), the threshold may be derived from remote sensing data. Significant deviations from local mean

values may also be employed.

Data filtering - Raw sensor data is filtered via a moving average filter, as described in Section A.2. All activities performed by `trench_finder` are then based on this filtered data stream.

Dead band - The purpose of data filtering is to minimize the effects of noise and false information in the system. The dead band operates on the assumption that the sensed, filtered data has temporal variability which exceeds the dynamic characteristics of the vehicle – resulting in unnecessary and undesirable oscillations in the vehicle trajectory as it attempts to follow the feature. To prevent this undesirable vehicle maneuvering over the feature, a dead band is installed. The `trench_finder` uses a dead band of ± 2 cm, based on field experiments conducted in the Charles River.

Cell size - The map cell size is based on the conflicting demands of the desire to maximize the resolution of the feature description being generated, minimize the time spent sampling any one feature, and the dynamic limitations of the AUV. If the intent is to map the distribution of features, then a larger cell size can be employed (limited by the estimated characteristic size of the feature). Larger cells result in a low resolution mapping of the region (see Figure 4-27), while smaller cells yield a more detailed map, but at the cost of more vehicle time necessary to generate the map (see Figure 4-31).

Waypoint search radius - `trench_finder` searches for candidate features by first examining the region immediately adjacent to known feature elements, as shown in Figure 3-18. The vehicle then chooses which of these cells to visit by comparing this list of candidate waypoints to a lookup table of map cells which are immediately adjacent to the vehicle (see Figure 3-19). This search can be expanded to encompass all map cells within a radius of two cells, if desired.

Waypoint search method - As discussed in Section 3.7.2, the waypoint search method determines how the vehicle ranks candidate waypoints. Methods available are pre-compiled lookup tables (either vehicle heading-dependent or heading-independent) or a cost/utility function (see Eqn. 4.1). In the case of a a lookup table, the waypoint

19	20	10	17	16
21	7	2	6	18
12	4	X	3	11
24	8	1	5	15
22	23	9	14	13

Figure A-1: Sample pre-compiled lookup table for `trench_finder`. The available unvisited waypoints are ranked according to the table. Visited cells are ignored. In this drawing, a known feature cell is centered at the "X".

search criteria are pre-compiled for the sake of speed and simplicity. A simple lookup table is shown in Figure A-1. More sophisticated lookup tables (e.g. orientation and position dependent) are also available for use.

Contour vs. region search - `trench_finder` can be set to search for a specific contour or gradient or a range of contours/gradients. It will also search all of an area on one side or another of a defined contour. If that contour is closed, it will search the interior or exterior of that contour, as desired.

Starting point - If a specific starting point in the search area is desired, it is given here. If not, the starting point is defined as shown in Figure 3-21.

A.2 Sonar filter

While most of the vehicle sensors are primarily prone to random noise, the sonar altimeter is also susceptible to multiple-, false- and dropped returns. In the case of the AUV *Odyssey* in the Charles River Basin, both 200 kHz and 500 kHz sonars were used. The 200 kHz unit suffered from absorption into the bottom – the bottom simply "vanished" from the vehicle's perception. To counter this, a 500 kHz unit was installed. While the problem of absorption was overcome, at that frequency the silty riverbed was highly specular in nature, acting like a mirror to the sonar signal. As a result, returns which exceeded the unit's cone angle were lost, causing increased dropouts over severe terrain and at high vehicle pitch and roll angles.

To counter these effects, a moving average filter [94] was employed. We can think

of a moving average process as some $\{X_t\}$ such that

$$X_t = \beta(B)\varepsilon_t \tag{A.1}$$

where

$$\beta(z) \equiv b_0 + b_1 z + \ldots + b_l z^l \tag{A.2}$$

with $\{\varepsilon\}$ a purely random process. Further, we can assume without loss of generality that $b_0 = 1$ or that ε_t has unit variance. This is because we can define a new random process

$$\varepsilon^* = b_0 \varepsilon_t \tag{A.3}$$

or

$$\varepsilon^* = \varepsilon_t / \sigma_\varepsilon \tag{A.4}$$

($\sigma_\varepsilon \neq 1$), but not both, that guarantees our desired parameters.

Note that the moving average, while resembling an autoregressive process, is not the same. The key difference between an autoregressive and a moving average process is that in the autoregressive case, some value X_t is the finite linear combination of the current and all past values of $\{\varepsilon_t\}$. By extension, we can observe that a given ε_t will influence all future $(X_t, X_{t+1}, \ldots)$ to some extent. In contrast, a moving average process expresses X_t as a linear combination of $\{\varepsilon_t\}$, but only to a finite extent into the past, thereby also limiting the influence of a given ε_t to a limited set $\{n\}$ of future values of $\{X\}$, namely $(X_{t+1}, \ldots, X_{t+n})$.

This is an important distinction in the case of our sonar filter. The nature of the sonar data stream is that of a random signal with random noise superimposed and the occasional dropped return. Because of the random nature of the seafloor and the trajectory of the vehicle, there is no reason to expect that the sonar return ε_{t-n} from some arbitrary point in the past should have any influence on the current return ε_t. On the other hand, because of the smooth changes found in nature, it is not unreasonable to expect that adjacent portions of the bottom are quite similar to the point being sampled at time t. The key is to determine the optimum number of

samples to use in the moving average.

For our purposes, a special case of the moving average filter (sometimes referred to as a *moving window* filter) has been chosen. In this case the weights $b_0, b_1, \ldots, b_n$ are all of equal weight such that

$$\sum_{i=0}^{n} b_i = 1 \tag{A.5}$$

or

$$b_0 = b_1 = \ldots = b_n = \frac{1}{n+1} \tag{A.6}$$

and the processed signal has the resulting variance and autocorrelation functions

$$\sigma_X^2 = \frac{\sigma_\varepsilon^2}{n+1} \tag{A.7}$$

$$\rho(r) = \begin{cases} 1 - \frac{|r|}{n+1} & , \quad |r| \leq n, \\ 0 & , \quad |r| > n. \end{cases} \tag{A.8}$$

$$\tag{A.9}$$

Note that this is also a stationary process, i.e. its statistics do not change over time and therefore it does not bias or influence the input data over time. Note also, however, that the input sonar data stream is *not* stationary over long time scales. Correctly employing such a filter (choosing the coefficients b_i requires knowledge of the scale of the feature of interest, the sampling rate of the vehicle as it tracks or searches for the feature, and the dynamic capabilities of the vehicle.

Altimeter - the assumed sonar is a Tritech model ST500. The ST500 is a 500 kHz sonar altimeter with a 20 degree conical beam and a range of 40 m. The bottom of the Charles River basin is composed of silt over clay, and acts as a specular surface at 500 kHz. The result is a poor target surface and a loss of contact with the bottom if the specular reflection angle exceeds the cone angle of the sonar. This was demonstrated in field experiments, as shown in Figure A-2. At approximately t = 500 seconds, the vehicle passes over the lip of the trench while attempting to maintain a constant

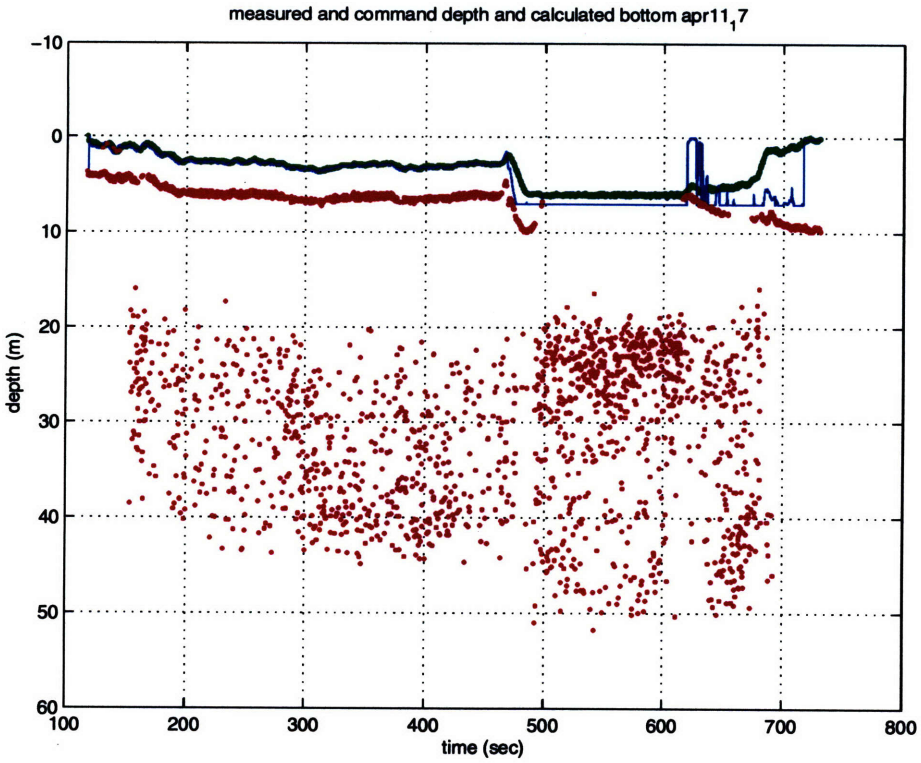


Figure A-2: AUV *Odyssey IIb* during bottom-following mission in the Charles River Basin on 11 April 1995. In this figure, blue is the commanded depth, green is the vehicle depth and red is the reconstructed water column. Note the poor sonar returns during the mission and especially after collision at t = 500 sec.

altitude. The steeply sloped trench wall caused a temporary loss of contact with the sonar altimeter. At the same moment, the vehicle was instructed to return to the launch point. The resulting vehicle path was a "corkscrew" motion into the trench wall as the vehicle attempted to dive and turn at the same time. The vehicle remained pinned against the trench wall until pulled free by the operators at t = 620 seconds.

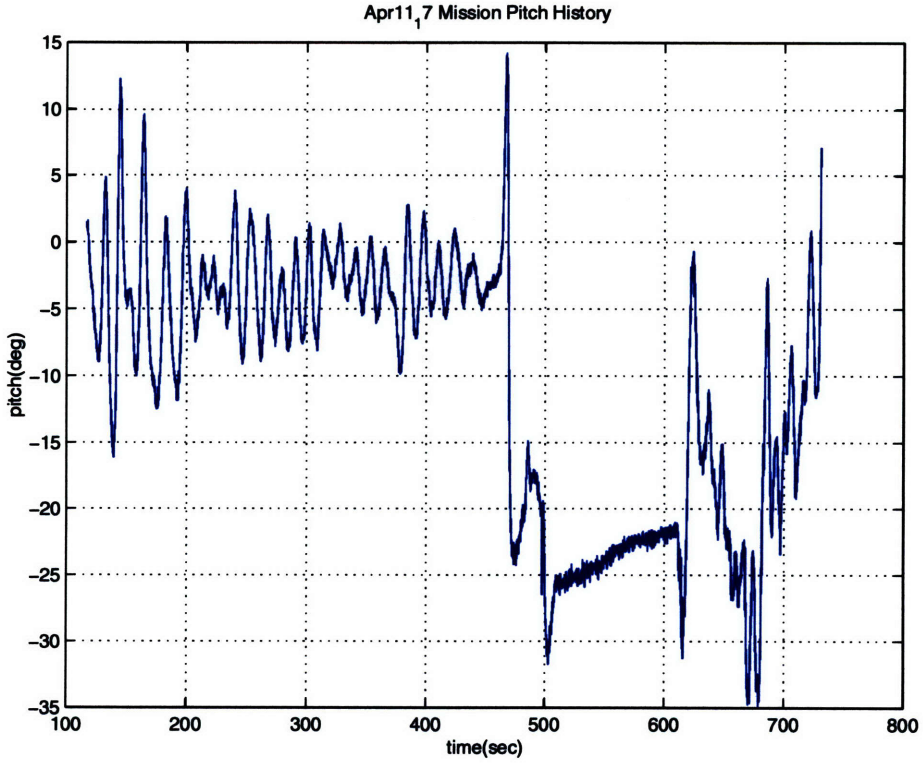


Figure A-3: Vehicle pitch angle during the mission of 11 April 1995. Note the large pitch excursions leading into the bottom collision at t = 500 sec. The vehicle first pitched up as it came over the embankment and then pitched steeply down, just as the bottom dropped away into the trench interior. The combined pitch plus steep slope resulted in a temporary loss of contact. This, combined with the turning maneuver executed at the same (inopportune) moment, resulted in a collision with the bottom and corresponding sonar return failure.

Bibliography

[1] K. Aramaki and T. Ura. Surveying and map-drawing by underwater vehicles based on ultrasonic range sensors. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 416–425, September 1995.

[2] R. Ballard. MIT Annual Robert Bruce Wallace Lecture, November 1993.

[3] J. Bares, M. Hebert, T. Kanade, and E. Krotkov. AMBLER: An autonomous rover for planetary exploration. *Computer*, 2(6):18–24, 1989.

[4] J. Barraquand and J. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2328–2335, Sacramento, CA, USA, April 1991.

[5] J. Barraquand and J. C. Latombe. Numerical potential field techniques for robot path planning. Technical Report Report No. STAN-CS-89-1285, Department of Computer Science, Stanford University, Stanford, CA, USA, 1989.

[6] J. G. Bellingham. Review of autonomous underwater vehicles. Technical report, Sea Grant College Program, Massachusetts Institute of Technology, February 1991.

[7] J. G. Bellingham. The Autonomous Oceanographic Sampling Network (AOSN), 1996. http://www.mit.edu:8001/research/seagrant/aosn.htm.

[8] J. G. Bellingham, C. Chryssostomidis, M. Deffenbaugh, J. J. Leonard, and H. Schmidt. Arctic under-ice survey operations. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 50–59, 1993.

[9] J. G. Bellingham and T. R. Consi. State configured layered control. In *Proceedings of the IARP 1st Workshop on Mobile Robots for Subsea Environments*, pages 75–80, Monterey, CA, USA, October 1990.

[10] J. G. Bellingham, T. R. Consi, R. Beaton, and W. Hall. Keeping layered control simple. In *Proceedings AUV '90*, 1990.

[11] J. G. Bellingham, T. R. Consi, U. Tedrow, and D. Di Massa. Hyperbolic acoustic navigation for underwater vehicles: Implementation and demonstration. In *Proceedings AUV '92*, pages 304–309, 1992.

[12] J. G. Bellingham, C. A. Goudey, T. R. Consi, J. W. Bales, D. K. Atwood, J. J. Leonard, and C. Chryssostomidis. A second generation survey AUV. In *IEEE Conference on Autonomous Underwater Vehicles*, Cambridge, MA, 1994.

[13] J. G. Bellingham, C. A. Goudey, T. R. Consi, and C. Chryssostomidis. A small, long-range autonomous underwater vehicle for deep ocean exploration. In *Proc. International Society of Off-shore and Polar Engineers*, 1992.

[14] J. G. Bellingham and D. Humphrey. Using layered control for supervisory control of underwater vehicles. In *Proceedings ROV '90*, 1990.

[15] J. G. Bellingham and J. J. Leonard. Task configuration with layered control. In *Proceedings of the IARP 2nd Workshop on Mobile Robots for Subsea Environments*, pages 193–302, Monterey, CA, USA, May 1994.

[16] J. G. Bellingham, J. J. Leonard, J. Vaganay, C. Goudey, D. Atwood, T. Consi, J. Bales, H. Schmidt, and C. Chryssostomidis. AUV operations in the arctic. In *Proceedings of the Sea Ice Mechanics and Arctic Modeling Workshop*, Anchorage, Alaska, USA, April 1995.

[17] J. G. Bellingham and J. S. Willcox. Optimizing AUV oceanographic surveys. In *Proc. IEEE Symposium On Autonomous Underwater Vehicle (AUV) Technology*, June 1996.

[18] A. A. Bennett. Combining planning with reactive architectures in an autonomous underwater vehicle. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 437–445, September 1993.

[19] A. A. Bennett, J. J. Leonard, and J. G. Bellingham. Bottom following for survey class autonomous underwater vehicles. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 327–336, September 1995.

[20] J. Bird and M. Goulding. Obstacle avoidance with resource constrained detection. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 441–453, 1991.

[21] J. Borenstein and Y. Koren. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[22] R. A. Brooks. A robust layered control system for a mobile robot. Technical Report AI Memo 864, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, USA, September 1985.

[23] R. A. Brooks. Achieving artificial intelligence through building robots. Technical Report AI Memo 899, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, USA, May 1986.

[24] R. A. Brooks. Challenges for complete creature architectures. In *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, USA, 1990. MIT Press.

[25] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, pages 3–15, June 1990.

[26] R. A. Brooks. Intelligence without reason. Technical Report No. 1293, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, USA, April 1991.

[27] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[28] D. P. Brutzman. Internet protocol over seawater (ip/sw): Towards interoperable underwater networks. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 444–457, 1995.

[29] E. Burien. Search methods for an autonomous underwater vehicle using scalar measurements. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.

[30] E. Burien, D. Yoerger, A. Bradley, and H. Singh. Gradient search with autonomous underwater vehicles using scalar measurements. In *Proceedings AUV '96*, 1996.

[31] J. F. Canny. *The Complexity of Robot Motion Planning.* PhD thesis, MIT Artificial Intelligence Laboratory, 1987.

[32] J. R. Carr and J. S. Sobek. Digital scene matching area correlator (DSMAC). In *SPIE Image Processing for Missile Guidence*, volume 238, pages 36–41, San Diego, July 1980.

[33] A. Cayley. On contour and slope lines. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 18(4):264, December 1859.

[34] R. Chatila. Path planning and environment learning in a mobile robot system. In *Proceedings of the European Conference on Artificial Intelligence*, Orsay, France, 1982.

[35] R. Chatila and J.P. Laumond. Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation.* IEEE, 1985.

[36] J. H. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2719–2724. IEEE, 1992.

[37] T. B. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94, 1993.

[38] Kahn W. D. Accuracy of mapping the earth's gravity field fine structure with a space borne gravity gradiometer mission. Technical Report N.84-30473, NASA Goddard Geodynamic Branch, 1984.

[39] M. Deffenbaugh. A matched field processing approach to long range acoustic navigation. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.

[40] G. H. Dutton. Fractal enhancement of cartographic line detail. *The American Cartographer*, 8(1):23–40, 1981.

[41] H. Edgerton, V. E. MacRoberts, and P. Mui. Charles River basin hydrographic survey. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, March 1985.

[42] H. E. Edgerton. 1967 Charles River basin sonic soundings. Technical report, New England Division, US Army Corps of Engineers, October 1967. Charles River Watershed Study, Hydrolics and Hydrology Interim Memorandum #2.

[43] R. Forsberg. Topographic effects in airborne gravity gradiometry. In *Proceedings, 15th Gravity Gradiometer Conference*, Colorado Springs, February 1987.

[44] M. A. Gerber. Gravity gradiometer: Something new in inertial navigation. *Astronautics and Aeronautics*, pages 18–26, May 1978.

[45] E. Geyer, P. Creamer, J. D'Appolito, and R. Gains. Characteristics and capabilities of navigation systems for unmanned untethered submersibles. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 320–347, 1987.

[46] G. Giralt, R. Chatila, and M. Vaisset. An integrated navigation and motion control system for autonomous multisensory mobile robots. In *First International Symposium on Robotics Research*. MIT Press, 1984.

[47] J. A. Goff and T. H. Jordan. Stochastic modeling of seafloor morphology: Inversion of sea beam data for second-order statistics. *Journal of Geophysical Research*, 93(B11):13,589–13,608, November 1988.

[48] J. Golden. Terrain contour matching (TERCOM): A cruise missile guidence aid. In *SPIE Image Processing for Missile Guidence*, volume 238, pages 10–18, San Diego, July 1980.

[49] editor H. Schmidt. Haro Strait '96: Frontal Dynamics PRIMER Experiment summary and cruise reports. Technical report, Massachusetts Institute of Technology Department of Ocean Engineering, Cambridge, MA, USA, 1997.

[50] R. Haralick, L. Watson, and T. Laffey. The topographic primal sketch. *Int. J. Robotics Research*, 2(1):50–72, 1983.

[51] R. R. Hatch, J. L. Luber, and J. H. Walker. Fifty years of strike warfare at the applied physics laboratory. *Johns Hopkins APL: Technical Digest*, 13(1):113–124, 1992.

[52] S. Hert, S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an AUV. In J. Yuh, T. Ura, and G. A. Bekey, editors, *Underwater Robots*, pages 17–45. Kluwer Academic Publishers, 1996.

[53] P. R. Hinrichs. Advanced terrain correlation techniques. In *Record of the 1976 Position Location and Navigation Symposium*, pages 89–96, 1976.

[54] L. Hostetler and R. Andreas. Nonlinear Kalman filtering techniques for terrain-aided navigation. *IEEE Trans. Automatic Control*, 28:315–323, March 1983.

[55] M. Hunt, W. Marquet, D. Moller, K. Peal, W. Smith, and R. Spindel. An acoustic navigation system. Technical Report WHOI-74-6, Woods Hole Oceanographic Institution, 1974.

[56] A. Jircitano, J. While, and D. Dosch. Gravity based navigation of AUV's. In *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, pages 177–180, Washington, DC, USA, June 1990.

[57] L. Kantha, D. M. Beitsell, S. L. Harper, and R. R. Leben. Altimetry in marginal, semi-enclosed and coastal seas - part I, 1996. http://www.cast.msstate.edu/Altimetry.

[58] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 500–505, St. Louis, MO, USA, March 1985.

[59] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398–1404, Sacramento, CA, USA, April 1991.

[60] M. Kuristsky and M. Goldstein. Inertial navigation. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles.* Springer-Verlag, 1990.

[61] I. S. Kweon and T. Kanade. Extracting topographic features for outdoor mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1992–1997, Sacramento, CA, USA, April 1991.

[62] J. E. Laird and P. S. Rosenbloom. Integrating, execution, planning, and learning in soar for external environments. In *Proceedings, AAAI-90*, pages 1022–1029, 1990.

[63] J-C. Latombe. *Robot Motion Planning.* Boston: Kluwer Academic Publishers, 1991.

[64] J. P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1120–1123, Milan, Italy, 1987.

[65] J. J. Leonard. *Directed Sonar Sensing for Mobile Robot Navigation.* PhD thesis, University of Oxford, 1990.

[66] J. J. Leonard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation.* Kluwer Academic Publishers, Boston, MA, USA, 1992.

[67] E. R. Levine, D. N. Connors, R. Shell, T. Gagliardi, and R. Hanson. Oceanographic mapping with Navy's large-diameter UUV. *Sea Technology*, pages 49–57, June 1995.

[68] E. R. Levine, R. G. Lueck, P. L. Donaghay, D. N. Connors, T. Gagliardi, R. C. Hanson, and R. R. Shell. Turbulence and optics sampling from an autonomous underwater vehicle. In *Proceedings AUV '96*, 1996.

[69] E. Levinson and C. San Giovanni Jr. Laser gyro potential for long endurance marine navigation. In *PLANS 80: Position Location and Navigation Symposium Record*, pages 115–129, Atlantic City, NJ, USA, December 1980.

[70] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983.

[71] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–670, 1979.

[72] L. Lucido, B. Popescu, J. Opderbecke, V. Rigaud, R. Deriche, Z. Zhang, P. Costa, and P. Larzabal. Segmentation of bathymetric profiles and terrain matching for underwater vehicle navigation. In *Proceedings of the second annual World Automation Conference*, Montpellier, France, May 1996.

[73] J. E. Lupton. Hydrothermal plumes: Near and far field. *AGU Monograph 91*, pages 317–346, 1995.

[74] P. Maes. Learning behavior networks from experience. In *Toward A Practice of Autonomous Systems: Prodeedings of the first European conference on artificial life*, pages 48–57. MIT Press, 1991.

[75] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *AI Journal*, 2, 1992.

[76] Elbert S. Maloney. *Dutton's Navigation and Piloting.* Naval Institute Press, Annapolis, MD, USA, 1985.

[77] M. Mataric. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3), 1997. special issue on Software Architectures for Physical Agents.

[78] M. J. Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *From Animals to Animats: Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, USA, 1992. MIT Press.

[79] J. C. Maxwell. On hills and dales. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 40(4):421–425, December 1870.

[80] M. B. May. Gravity navigation. In *Record of the 1978 Position Location and Navigation Symposium*, pages 212–218, San Diego, CA, USA, November 1978.

[81] C. McNeill. *Orienteering: the Skills of the Game.* Crowood Press, 1989.

[82] P. H. Milne. *Underwater Acoustic Positioning Systems.* E. F. N. Spon, London, England, 1983.

[83] T. M. Mitchell. Plan-Then-Compile Architectures. *SIGART Bulletin*, 2(4), 1991.

[84] H. P. Moravec and D. W. Cho. A baysean method for certainty grids. In *AAAI Spring Symposium on Robot Navigation*, pages 57–60, 1989.

[85] G. Muhr. NOAA Great Lakes Environmental Research Laboratory home page, 1997. http://www.glerl.noaa.gov/.

[86] J. Myers, C. Holm, and R. MacAllister, editors. *Handbook of Ocean and Underwater Engineering.* New York: McGraw-Hill, 1969.

[87] L. R. Nackman. Two-dimensional critical point configuration graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):442–450, July 1984.

[88] Naval Undersea Warfare Center (NUWC) Newport Division. NUWC DIVNPT UUV home page, 1997. http://www.nuwc.navy.mil/.

[89] NIMA. *Mapping, Charting and Geodesy Handbook.* National Imagery and Mapping Agency, 1996. http://www.nima.mil/.

[90] D. Pagac, E. M. Nebot, and H. Durrant-Whyte. An evidential approach to probabilistic map-building. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 745–750, Minneapolis, MN, USA, April 1996.

[91] J. G. Paglia and W. F. Wyman. DARPA's Autonomous Minehunting and Mapping Technologies (AMMT) Program: An Overview. In *Proc. IEEE Oceans Conference*, pages 794–799, Ft. Lauderdale, FL, USA, September 1996.

[92] D. Payton, D. Keirsey, D. Kimble, J. Krozel, and K. Rosenblatt. Do whatever works: A robust approach to fault-tolerant autonomous control. *Journal of Applied Intelligence*, 3:226–249, 1990.

[93] D. Polvani. Magnetic guidance of autonomous vehicles, part 2. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 257–264, 1987.

[94] M. B. Priestley. *Spectral Analysis and Time Series.* Academic Press, 1981.

[95] United Nations Environment Programme. Northeast Russian petroleum service agency facts, 1997. http://www.grid.unep.no/arci0025.htm.

[96] A. H. Robinson, J. L. Muehrcke, A. J. Kimerling, and S. C. Guptill, editors. *Elements of Cartography, 6th ed.* Wiley, 1995.

[97] S. Russell and P. Norvig. *Artificial Intelligence A Modern Approach.* Prentice Hall, 1995.

[98] D. T. Sandwell. Geophysical applications of satellite altimetry. *Reviews of Geophysics Supplement*, pages 132–137, 1990.

[99] H. Schmidt, 1995. Personal communication.

[100] H. Schmidt, J. Bellingham, M. Johnson, D. Herold, D. Farmer, and R. Pawlowcisz. Real-time frontal mapping with AUVs in a coastal environment. In *Proc. IEEE Oceans Conference*, pages 1094–1098, 1996.

[101] J. T. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[102] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, Academic Press 4:298–351, 1983.

[103] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research*, 2(3):46–75, 1983.

[104] J. T. Schwartz and M. Sharir. On the piano movers' problem: V. the case of a rod moving in three-dimensional space amidst polygonal obstacles. *Communications on Pure and Applied Mathematics*, 37:815–848, 1983.

[105] H. Singh. *An Entropic Framework for AUV Sensor Modeling.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[106] H. Singh, D. Yoerger, R. Bachmayer, A. Bradley, and W. Stewart. Sonar mapping with the autonomous benthic explorer (ABE). In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 367–375, September 1995.

[107] W. K. Stewart. *Multisensor Modeling Underwater with Uncertain Information.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1988.

Massachusetts Institute of Technology Artificial Intelligence Laboratory AI-TR 1143.

[108] R. B. Tilove. Local obstacle avoidance for mobile robots based on the method of artificial potentials. In *Proc. IEEE Int. Conf. Robotics and Automation*, page 566, 1990.

[109] B. H. Tracey. Design and testing of an acoustic ultra-short baseline navigation system. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1992.

[110] D. Tucker. NASA Ames IMG: Dante II walking robot, 1995. http://maas-neotek.arc.nasa.gov:80/dante/.

[111] S. T. Tuohy. *Geophysical Map Representation, Abstraction, and Interrogation for Underwater Vehicle Navigation.* PhD thesis, MIT, 1993.

[112] S. T. Tuohy, J. J. Leonard, J. G. Bellingham, N. M. Patrikalakis, and C. Chryssostomidis. Map based navigation for autonomous underwater vehicles. *International Journal of Offshore and Polar Engineering*, 6(1):9–18, March 1996.

[113] D. Turcotte and G. Schubert. *Geodynamics: Applications of Continuum Physics to Geological problems.* New York: John Wiley and Sons, 1982.

[114] C. Tyren. Magnetic anomalies as a reference for ground-speed and map-matching navigation. *The Journal of Navigation*, 35(2):242–254, May 1982.

[115] C. Tyren. Magnetic terrain navigation. In *Proc. Int. Symp. on Unmanned Untethered Submersible Technology*, pages 245–256, 1987.

[116] S. Udpa. *Collision Detection and Avoidance in Computer Controlled Manipulators.* PhD thesis, California Institute of Technology, 1977.

[117] J. Vaganay, J. G. Bellingham, and J. J. Leonard. Outlier rejection for autonomous acoustic navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2174–2181, April 1996.

[118] S. A. Vere. Organization of the basic agent. *SIGART Bulletin*, 2(4), 1991.

[119] T. H. Waterman. *Animal Navigation.* Scientific American Library, 1989.

[120] R. B. Whitmarsh, L. M. Pinheiro, P. R. Miles, M. Recq, and J. C. Sibuet. Thin crust at the western Iberia ocean-continent transition and ophiolites. *Tectonics*, 12:1230–1239, 1993.

[121] J. S. Willcox, Y. Zhang, J. G. Bellingham, and J. Marshall. AUV survey design applied to oceanographic deep convection. In *Proc. IEEE Oceans Conference*, 1996.

[122] B. Yamauchi. Mobile robot localization in dynamic environments using dead reckoning and evidence grids. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1401–1406, Minneapolis, MN, USA, April 1996.