A Behavior-Based Approach to Adaptive Feature Detection and Following with Autonomous Underwater Vehicles

Andrew A. Bennett, *Associate Member, IEEE,* and John J. Leonard, *Member, IEEE*

***Abstract*—This paper presents a technique for adaptively tracking bathymetric contours using an autonomous underwater vehicle (AUV) equipped with a single altimeter sonar. An adaptive feature mapping behavior is developed to address the problem of how to locate and track features of unknown extent in an environment where *a priori* information is unavailable. This behavior is implemented as part of the layered control architecture used by the AUV Odyssey II. The new adaptive feature mapping behavior builds on previous work in layered control by incorporating planning and mapping capabilities that allow the vehicle to alter its trajectory online in response to sensor data in order to track contour features. New waypoints are selected by evaluating the expected utility of visiting a given location balanced against the expected cost of traveling to a particular cell. The technique is developed assuming sensor input in the form of a single, narrow-beam altimeter sensor attached to a non-holonomic, dynamically controlled survey-class AUV such as the Odyssey II. Simulations of the Charles River basin which have been constructed from real bathymetry data are used as test missions. The 7-m contour line of a prominent trench in the river serves as the target feature. The adaptive contour following behavior tracks the contour despite navigation error and environmental disturbances, supplying the capability of autonomously detecting and following distinctive bathymetric features using a point sensor. This behavior provides a foundation for future research in tracking of dynamic features in the water-column and for concurrent mapping and localization over natural terrain using a point sensor.**

***Index Terms*—Adaptive sampling, autonomous underwater vehicles, intelligent control, navigation.**

I. Introduction

THIS PAPER presents a behavior-based approach to the problem of adaptively mapping contour features using a survey-class autonomous underwater vehicle (AUV) such as the Odyssey II [3] equipped with a single downward-looking sonar. The approach was developed to address the challenge of how to use a survey class AUV to adaptively map distinctive bathymetric features such as trenches. A prominent trench in the Charles River (shown in Fig. 2) was used as the basis for a simulation study of how to perform adaptive feature mapping using a behavior-based approach to high-level control.

Manuscript received March 10, 1999; revised August 5, 1999. This work was supported by the Department of Commerce under Grant NA-46-RG-0434, by NASA under Grant 3843-GC94-0231, by the Henry L. and Grace Doherty Assistant Professorship in Ocean Utilization, and by the National Science Foundation under Grant BES-9733040.

The authors are with the Department of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139-4307 USA.

Publisher Item Identifier S 0364-9059(00)03370-7.

Adaptive contour following is one example of a larger class of problems known as feature relative navigation which require the AUV to determine its trajectory online in response to sensor data. Feature relative navigation tasks are difficult because they require the integration of techniques from intelligent control, navigation, and mapping, in real-time to enable the AUV to perform its mission safely and efficiently. Behavior-based control approaches are attractive for feature relative navigation tasks because of their low computational requirements and potential for real-time response in dynamic environments. However, the types of missions performed with AUV's to date have been restricted to reactive tasks such as bottom following, homing, and obstacle avoidance. By considering the task of adaptive feature mapping, in this paper we aim to apply a behavior-based control approach to a more complex type of mission than has been attempted previously.

The desirable aspects for adaptively mapping a feature in a predesignated area such as the Charles River are:

- quickly finding instances of a given feature type;
- adequately sampling the feature once it is found;
- efficiently locating all instances of a feature of a given size within the assigned region.

The technique developed should retain these attributes when there is navigation uncertainty such as a compass or speed sensor bias and external disturbances such as currents.

The structure of this paper is as follows. Section II presents the problem we are considering from the standpoint of potential applications. Section III provides background for the problem in terms of previous research in high-level control of AUV's. Section IV defines the problem formally and states the assumptions used in our simulation study. Section V describes our new technique for adaptive contour following, and Section VI presents simulation results that demonstrate the success of the method both with and without navigation error and external currents. Finally, Section VII presents our conclusions and discusses future research. The Appendix gives the parameter values for the behavior that were used in the experiments.

II. Motivation

The full potential of AUV's to revolutionize ocean sampling [17] cannot be realized until methods are developed to allow AUV's to react to sensor measurements in real-time as they are being acquired. New information processing and intelligent control techniques are required to enable AUV's to navigate relative to features of interest in the ocean environment and to max-

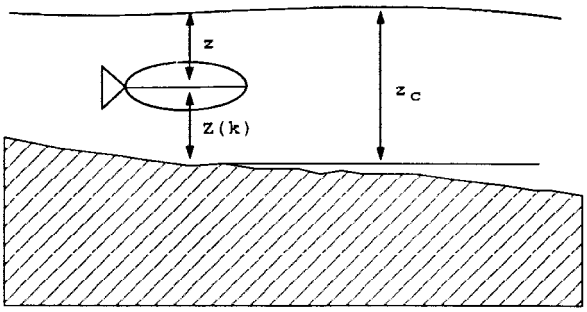


Fig. 1. Definition of variables for measured altitude $Z(k)$, measured depth z, and depth of desired contour z_c.

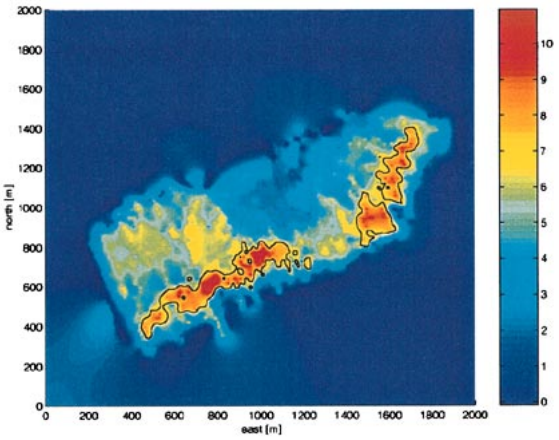


Fig. 2. Charles River basin with 7-m contour highlighted. This is a bird's-eye view of the river. The origin of the coordinate system in the lower left-hand corner of the figure was chosen for convenience. MIT is located north of this section of the river (at the top of the page) and the Back Bay area of Boston is to the south (at the bottom of the page).

imize the amount of sensor data obtained from these features. Three scenarios in which the capability to navigate relative to *contour* features are: 1) adaptive mapping of a region; 2) adaptive mapping of a dynamic feature; and 3) geophysical navigation using natural terrain features.

For example, automated survey of ice keels in the Arctic presents one instance of the problem of adaptive region mapping. Recently oil companies have expressed interest in drilling for oil in the Sea of Okhotsk off the eastern coast of Russia [27]. 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. Consequently, ice keels carve trenches into the seabed each winter as the pack ice moves about. These ice keels are likely to cut oil pipelines laid on the seabed, resulting in an ecological disaster. It would be desirable for AUV's to survey the extent and distribution of both the ice keels and the trenches dug out by ice keels in the area to provide information on how deeply new oil pipelines should be buried. To perform this mission, an AUV should be capable of locating and mapping the extent, depth, and distribution of ice keels and trenches so that researchers can develop an understanding 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.).

The Haro Strait experiment, performed by MIT, Woods Hole Oceanographic Institution, and the Institute of Ocean Sciences in July 1996 [29], presents a good motivating scenario for adaptive mapping of dynamic ocean 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, geophysical navigation using natural terrain features, is a technique whereby a vehicle matches sensor measurements to an *a priori* map to determine its position. Kamgar-Parsi has presented techniques for matching observed terrain contours with an *a priori* map [20]. To enable vehicles equipped with a single altimeter to navigate in this way, the vehicle must be able to follow a contour line on the bottom to obtain enough measurements from different parts of the contour to be useful for matching to the map.

In each of these three scenarios, it is desirable for the AUV to detect and track contour features based on the data measured by the vehicle sensors. These tasks are beyond the current state-of-the-art of what has been demonstrated to date in high-level control of AUV's.

III. Related Research

High-level control architecture design has been recognized as an important problem both in autonomous underwater vehicle research and in the fields of mobile robotics and artificial intelligence (AI) in general [24]. Three main schools of thought have emerged: 1) planning-based systems; 2) behavior-based systems; and 3) hybrid systems, which incorporate characteristics of the first two categories.

Planning-based systems typically rely on a central world model which is created by processing of sensor data [15], [18], [22], [25]. The information in the central model is used by a *planner* to determine a course of action necessary to satisfy the vehicle's goals. Planners can be powerful if sufficient time and information are available. Their chief drawback is the inability to handle rapidly changing environments in a timely fashion [10], [11].

Behavior-based systems 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 operate over a limited time horizon. 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.

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. The reader is directed to Kortenkamp *et al.* [21] for a more detailed discussion.

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 [2], a voting scheme [26], or spreading of activation [23].

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 diverse body of work, ranging from internalized plans [26] to contingency plans [16] 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 [7].

The Draper Laboratory's planning-based system developed for the DARPA Autonomous Minehunting and Mapping Technology project presents a good example of a recent planning-based AUV controller [28], while the Naval Postgraduate School's Tri-level Control Architecture developed for the Phoenix AUV is a representative hybrid architecture [19]. Since this paper describes a new type of behavior-based system, the rest of this section focuses on the historical development of behavior-based AUV control.

A. The Subsumption Architecture

Brooks' seminal work in the development of the *subsumption architecture* at the MIT AI lab during the 1980's came as a reaction to the "traditional" high-level control architectures which were then prevalent [9]. In traditional high-level control, a task is broken into *functional units*, each executed in turn after the other [see Fig. 3(a)]. This form of decomposition assumes that each task is composed 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

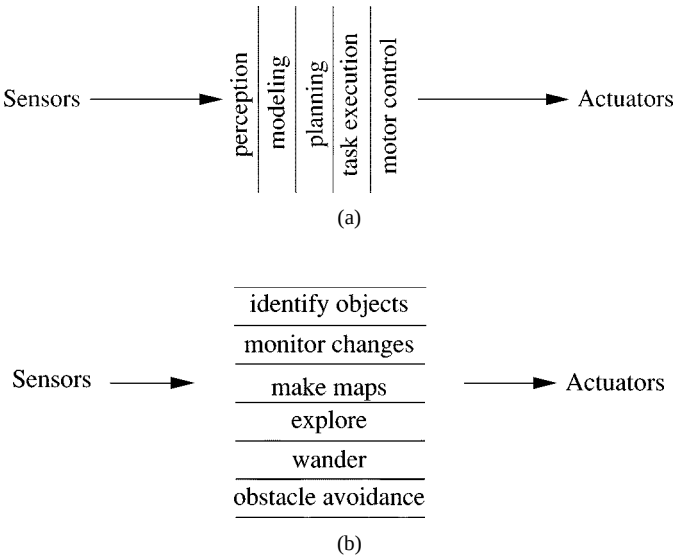


Fig. 3. (a) Traditional functional decomposition of intelligent control for a mobile robot system versus (b) behavior based decomposition of a mobile robot [9].

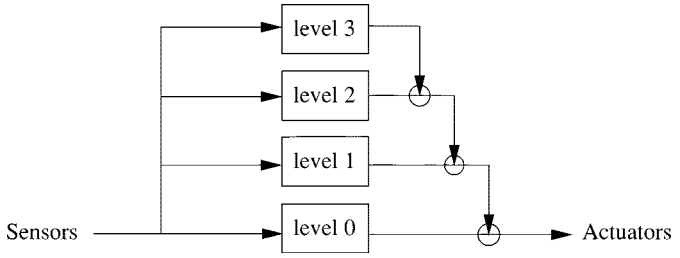


Fig. 4. Subsumption is a series of competence layers. Each layer can suppress the layer below it [9].

[Fig. 3(b)]. 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.

In the development of a behavior-based control system, one first creates a complete root control system, referred to as the *zeroth* layer, or level 0 competence. This is thoroughly debugged and tested. Once proven, another layer of competence is added and is called the first-level competence. Level 1 can monitor the same input data as level 0 and can also monitor its output (see Fig. 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.

In practice, this form of control quickly reveals 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 second shortcoming

is commonly referred to as *situatedness* [7]. 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 in which the vehicle as a whole will function. 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 low computational requirements [2], [7], [24]. Kortenkamp *et al.* provide a detailed discussion of both the profound impact of behavior-based control on mobile robotics, and an assessment of some of its shortcomings [21].

B. Layered Control

Bellingham and Consi proposed the concept of *layered control* as an adaptation of the subsumption architecture suitable for high-level control of AUV's. Layered control is essential a simplified version of the subsumption architecture [2]. The approach has been validated on the Odyssey II AUV's performing numerous field deployments [3], [29]. The approach grew out of an effort to address the problem of scaling. The principal difference between layered control and the subsumption 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.

In the 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 performed by a module which is external to the behaviors. The results of sensor processing are made available to all behaviors via a central data structure. (Such a data structure is often called a "blackboard" [14].) 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. In this way, many of the problems of interconnections and scaling found with a purely subsumptive system are avoided.

To extend layered control to handle multi-phase missions and avoid the problem of situatedness, state configured layered control (SCLC) [4] was proposed. In SCLC, multiple sets of simple behaviors are 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. These issues are discussed in more detail in Section V.

C. Adaptive Sampling

The problem of adaptive sampling with AUV's is closely related to the high-level control problem. The goal is to control the trajectory of the AUV to obtain a sampling pattern that will most efficiently characterize a given phenomenon. 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 and temporal capabilities (i.e., maneuvering, speed) of the sensor platform.

Bellingham *et al.* [6], [33] 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 and endurance of the AUV. The resulting path was designed to be the optimal 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 to maximize the spatial and temporal information obtained using coordinated fleets of AUV's.

Singh [30] and Burien [12], [13] 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, MA (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.

The adaptive feature mapping behavior presented in this paper differs from these approaches because it is feature-based. 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 versus total mission time.

IV. Problem Statement

Motivated in particular by the under-ice mapping scenario discussed already in Section II, we address the challenge of how to use the AUV Odyssey II equipped with just a sonar altimeter to adaptively map the trench in the Charles River in Boston, MA, as shown in Fig. 2. A bathymetric feature like the Charles River trench would enable repeatable tests of different adaptive sensing strategies in the future.

A survey class vehicle, such as the AUV Odyssey II [3], is assumed to be the type of AUV employed to map the feature. The choice of vehicle class has a direct impact on the problem approach due to the nature of dynamic control of survey-class AUV's (i.e., minimum speed at which the vehicle is controllable and a finite turning radius). The results in this paper were ob-

tained using the full vehicle software and dynamic simulation environment for the Odyssey II developed by Bellingham [3], and hence the software developed in the simulation study would translate directly to the vehicle for future field testing.

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. By considering adaptive feature mapping using a point sensor, we can develop techniques which can be generalized in the future to apply to conductivity, temperature, and depth (CTD) measurements of dynamic ocean features, ice keels, and ocean fronts. One of the challenges for extending these techniques to track dynamic features is that in such an experiment it would be difficult to know the ground truth and to compare the results of different experiments.

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 [3]. The sensor model is based on the characteristics 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 II in the Charles River. The bathymetry of the test area is derived from sonar altimeter data collected in the Charles River basin by a group of MIT students during the summer of 1993.

Incorporating these assumptions, the problem can be summarized as follows (see Fig. 1). The AUV moves in the horizontal plane at constant depth z and has position and heading $\boldsymbol{x}(k) = (x(k), y(k), \psi(k))$, where k is the current step of discrete time. The pitch θ and roll ϕ of the vehicle are assumed to be maintained at small values (on the order of plus or minus a few degrees) by the vehicle's dynamic controller. The vehicle is controlled by specifying its forward speed $u(k)$ and yaw rate $r(k)$. The speed and yaw rate are constrained by: $0 \leq u_{\min} \leq u(k) \leq u_{\max}$ and $|r(k)| \leq r_{\max}$. A navigation system is available that produces position estimates for the vehicle $(\hat{x}(k), \hat{y}(k))$ with known error characteristics and an altimeter sensor that produces measurements of the height of the vehicle off the bottom $Z(k) = f(x, y) - z + v(k)$, where $f(x, y)$ is a single-valued height function that represents the bottom bathymetry and $v(k)$ is a noise process. The vehicle is assumed to have a "waypoint controller" that steers the vehicle through a sequence of desired waypoints $\boldsymbol{x}_i$ for $i = 1 \cdots n$. The problem for the adaptive feature mapping behavior is to process the altimeter and navigation sensor measurements to choose the sequence of waypoints that the vehicle should follow to accomplish the objectives of detecting and tracking instances of a desired contour line consisting of all locations $\{(x_i, y_i)\}$ such that $f(x_i, y_i) = z_c$, where $Z(k) + z = z_c$.

V. Approach

To address this problem, we developed a new behavior for adaptive feature mapping which represents an extension of layered control that incorporates planning and mapping capabilities within an "adaptive behavior." Before implementing the approach, which is described in more detail below, we first attempted to solve the problem with two less complex approaches: a simple reactive technique, aligned with Brooks' philosophy of behavior design, and a state-configured layered control approach, which attempted to decompose adaptive feature mapping into a multi-phase mission. Both attempts were unsuccessful.

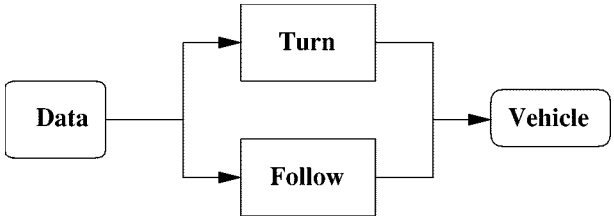


Fig. 5. First implementation (simple layered control approach). In this implementation, there were two vehicle behaviors: follow, which held a steady course over a feature, and turn, which reversed vehicle heading, alternating left and right, in an attempt to re-acquire the feature.

The simple reactive system consisted of two behaviors, "follow" and "turn," as shown in Fig. 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. Figs. 6 and 7 show two representative runs. The simple reactive behavior approach had no memory. It found a feature through random motion, and it tried to track a feature with a very simple strategy. The problem with this approach was that it experienced difficulty in re-acquiring the feature once it lost track of it. Approximately half of the simulated missions performed with the behavior ended unsuccessfully in this way.

To improve the vehicle performance, the second implementation adopted a state-configured layered control approach [1]. The task of adaptive contour following was decomposed into multiple phases and simple rules were created for transitioning between the active behavior. A spiral search phase was added to complement the detect and follow phases of the simple reactive system (Fig. 8). The result, illustrated in Fig. 9, was that the vehicle did much better in not losing a feature once it found it, but the actual trajectory that resulted was inefficient. The vehicle was prone to repeatedly revisit positions of the feature that were previously mapped.

Our experience with these implementations led to a new strategy for designing behaviors, which was embodied in the creation of the `trench_finder` adaptive behavior. The approach maintains the attractive features of the layered control structure, but greatly expands the capabilities of an individual behavior. The novel aspect of our approach is that the additional decision-making capability is not at some higher level but is instead embedded in the low-level behavior itself in a restricted fashion. Specifically, basic mapping and planning capabilities are added to the behavior. The result is a large increase in vehicle capability and functionality through an incremental increase in the complexity of the intelligent control system. We refer to the new behavior as an "adaptive behavior." This strategy of behavior design provides a technique for using layered control for more complex missions without imposing penalties of scaling and undesirable interactions between many simple behaviors.

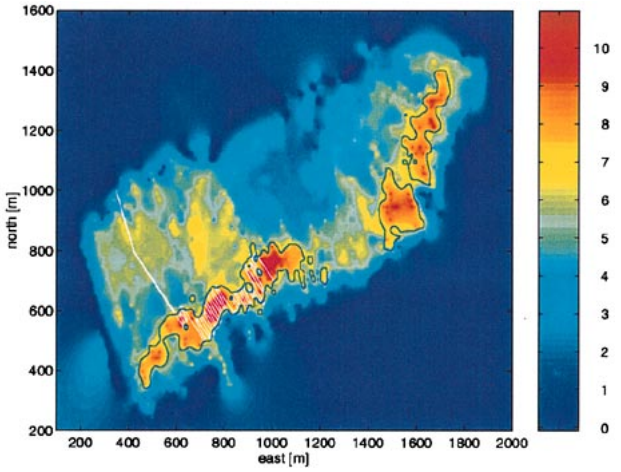


Fig. 6. Example of a successful trench finding mission using a simple reactive system. The vehicle started in the upper left of the figure at location [300N, 1000E]. Detections of the contour of interest triggered heading changes by the vehicle, resulting in a path that zigzags its way across the trench. The commanded vehicle speed was 1.0 m/s.

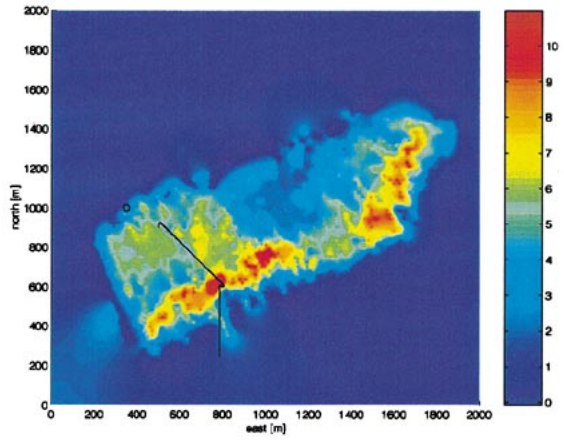


Fig. 7. Example of an unsuccessful trench finding mission using a simple reactive system. The vehicle started at location [500N, 900E]. With the different starting point from the mission in Fig. 6, the vehicle loses the feature after two turns and travels outside the search area.

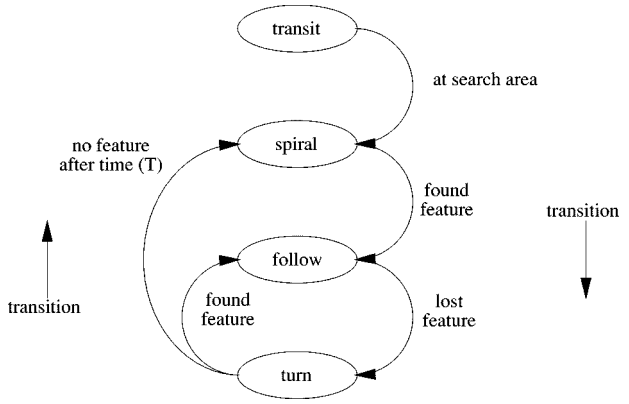


Fig. 8. Second implementation (SLCL approach). An initial transit phase and a spiral search phase were added and decision rules were defined to transition between these states.

In a pure SCLC system, the states are predefined before the mission. This can lead to two key problems: 1) contingency planning and 2) adaptability. Mataric observes that the solution

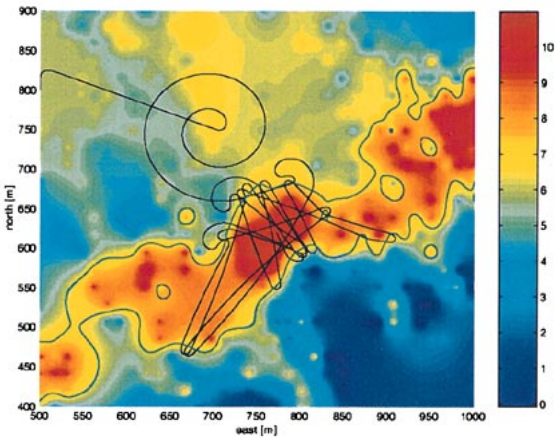


Fig. 9. SCLC implementation, in which a spiral search phase is executed whenever the vehicle loses the feature. The vehicle started at location [500N, 800E]. The mission duration was 160 min.

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" [24]. 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 Fig. 9 has sufficient states to address the basic problem at hand, that is, 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 ensure 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 predetermined mapping function, but instead changes itself in response to current and past sensory information, that is, 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 concept of an *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 adaptive behavior has (or has access to) memory, a capacity for planning, and can contain multiple states internally (see Fig. 10). 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

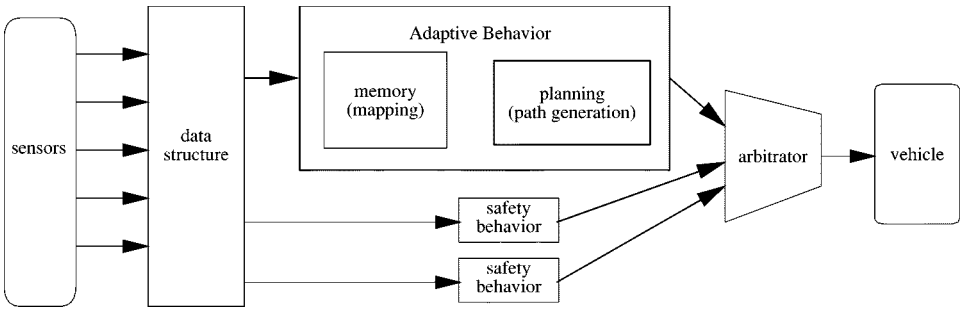


Fig. 10. Schematic of an adaptive behavior incorporated into a layered control system.

advance of their use, making them predictable in the field. By doing this, a user can employ sophisticated prepackaged 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 that is lower than any safety behaviors that may be employed during the mission (e.g., power monitoring, obstacle avoidance, etc.). The output is restricted to vehicle heading, depth, and speed, while inputs are restricted to a few mission-specific global parameters, such as desired contour depth and bounds on the search area.

To keep track of previous locations where the vehicle has traveled and where the feature has been detected, the `trench_finder` behavior maintains two different data structures, the *visitation map* and the *feature map*. Each map consists of a set of cells arranged in a regular grid. A grid-based map has the advantages of simplicity and flexibility, it is easy to search and manipulate, and can accommodate multiple sensor modalities and feature types. The *visitation map* tracks all locations that have been visited by the vehicle during the mission so far. The *feature map* keeps track of the locations where the feature of interest has been detected, which is subset of the set of all the *visited* cells. A cell size of 10×10 m was chosen because it is approximately twice the turning radius of the AUV Odyssey II.

The next step is for the behavior to use the information in these two maps to *plan* where to go next. The map is used as it is created 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.

New waypoints are selected by evaluating the expected utility of visiting a given location balanced against the expected cost of transitioning to a particular cell. Preference is given 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 $\{\overline{V}\}$ ($C \cap V = \emptyset$). This is illustrated in Fig. 11. This set of candidate cells $\{C\}$ is then ranked according to distance and orientation relative to the AUV, with preference given to those cells in the immediate path.

The `trench_finder` ranks these either according to a lookup table based on distance (see Fig. 12) or a cost function

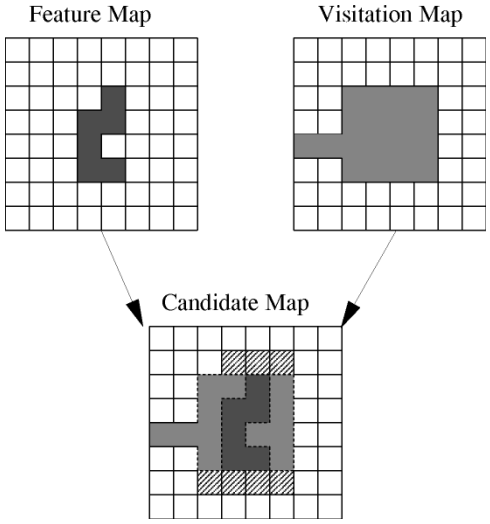


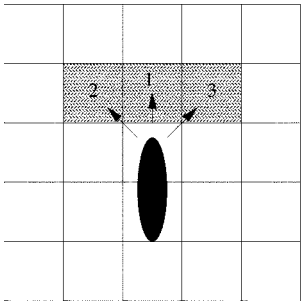
Fig. 11. Waypoint candidate generation process. The *feature map* on the upper left tracks all known candidate features. The *visitation map* on the upper right tracks all places where the vehicle has been. Candidate waypoints are determined by examining the intersection of the two maps. Based on the 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 Fig. 12.

which can apply different weights to heading and speed changes. The cost function is

$$W = \min\left\{\alpha\sqrt{(X_{\rm wp}-X_{\rm veh})^2+(Y_{\rm wp}-Y_{\rm veh})^2} + \beta\left|\theta_{\rm veh}-\tan^{-1}\left(\frac{Y_{\rm wp}-Y_{\rm veh}}{X_{\rm wp}-X_{\rm veh}}\right)\right|_{0<\Delta\theta<\pi}\right\} \quad (1)$$

where $(X_{\rm wp}, Y_{\rm wp})$ designate the candidate waypoint, $(X_{\rm veh}, Y_{\rm veh})$ designate the vehicle position, and α and β are weights which alter the relative cost of turning versus distance to candidate waypoints. When using the cost function, the vehicle determines where to go next by compiling a list of candidate waypoints and then ranks these candidates by cost value W, low to high, and proceeds to visit each waypoint in sequence.

Each time the adjacency map is updated, the list of candidate cells is also updated and reranked 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.



(a)

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

(b)

Fig. 12. (a) 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 or based on a function of distance and location relative to the current vehicle location and orientation. (b) Sample pregenerated 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.

This occurs when the list of candidate cells is empty ($\{C\} = \emptyset$). Seeking new features is performed by first setting a waypoint in the center of the largest unmapped region. Upon arrival at this location, a spiral search pattern is performed until either: 1) a new feature is found or 2) 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.

The internal construction of the `trench_finder` consists 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. 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 implementation differs from SCLC [1] because each state is not a full behavior, and the transition rules are isolated from the user. The adaptive behavior is controlled by a small number of parameters and will behave in a well-understood manner.

The states are illustrated in Fig. 13 and summarized below.

`initialize`: Execution of `trench_finder` begins by initializing various parameters used during the mission. (These are described more fully in the Appendix).

`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 the 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 in the center of the search area.

`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 coarse survey can be set by the user to be at a search interval of 10–100 times the standard search interval.

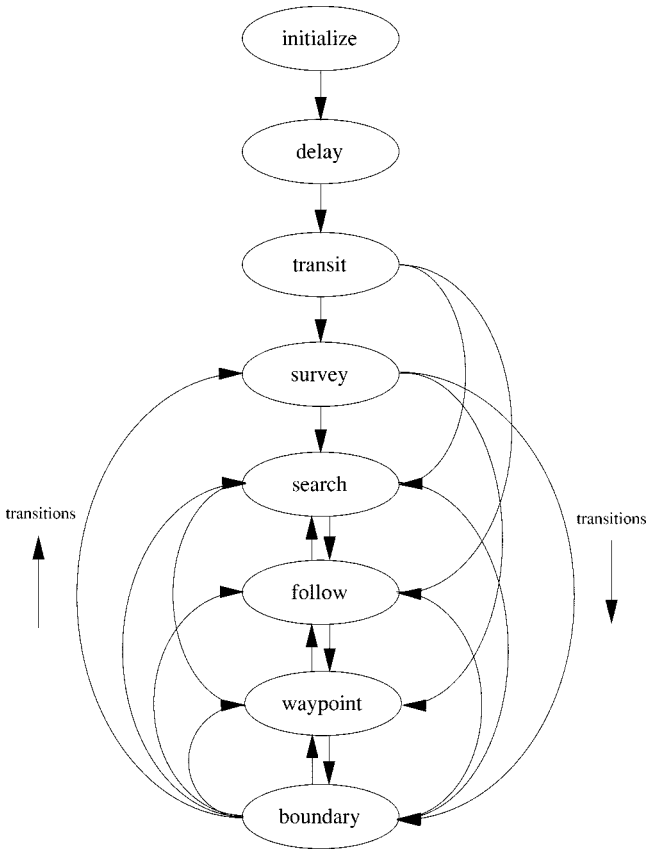


Fig. 13. Flow of control within the adaptive contour following behavior. At each time step, one of the states above is active. Transitions between states are determined based on current navigation and sensor information and the content of the visibility and feature maps that have been created thus far during the mission.

`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 next. Waypoint transitions to the follow state when a feature has been detected. If no further

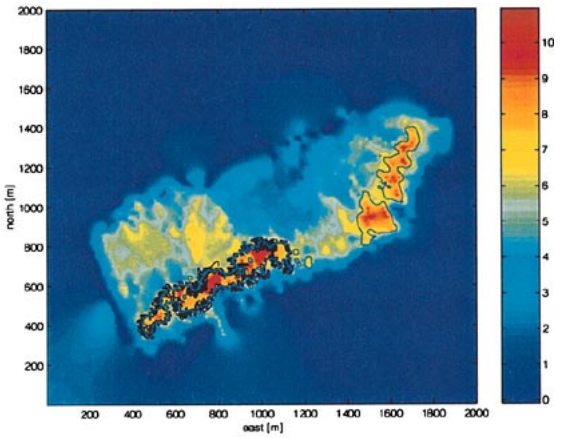


Fig. 14. Vehicle path for Mission 1. The vehicle is started near the feature at location [800N, 700E] to save transit time, but has no *a priori* information about the feature. The mission duration was 5 h (based on an assumed battery capacity). The commanded vehicle speed was 1.0 m/s.

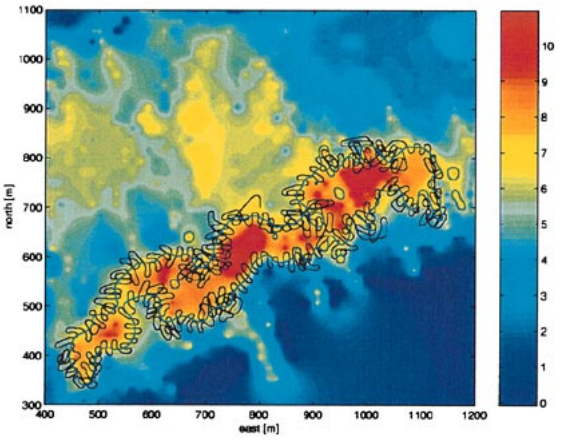


Fig. 15. Close-up of vehicle path for Mission 1.

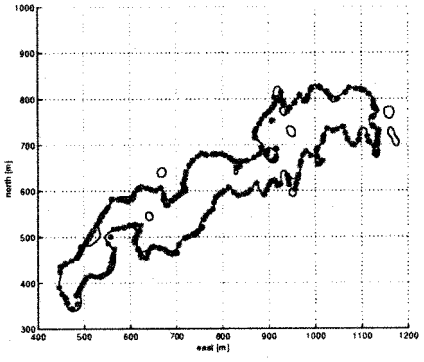


Fig. 16. Feature map generated by the adaptive contour follower for Mission 1.

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 reentry into the search area.

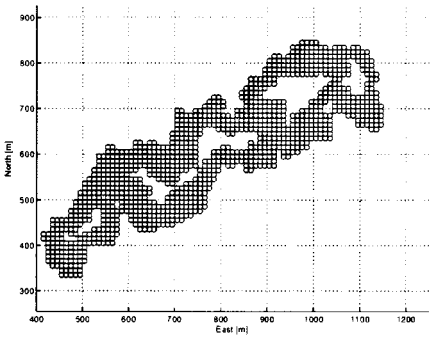


Fig. 17. Visitation map generated by `trench_finder` for Mission 1.

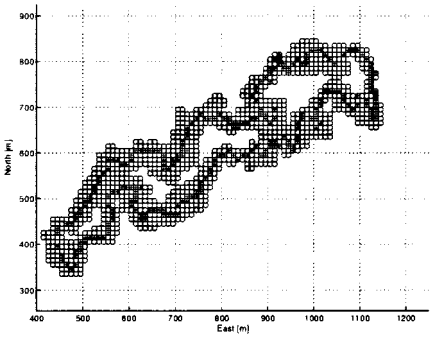


Fig. 18. Feature map ("+") overlaid onto the visitation map ("∘") generated by `trench_finder` for Mission 1. Note the unvisited candidate adjacency cells along the eastern edge of the feature.

All state transitions are governed by a set of predefined transition rules that depend on the current vehicle state, navigation information, and sensor data. This information is combined with the on-board maps to determine transitions to other states. 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). The values of the parameters used in the behavior for the simulations reported below are provided in Table I in the Appendix.

VI. Results

Samples of the performance of the `trench_finder` behavior are shown in Figs. 14–23. Figs. 14–18 and Figs. 19–23 show typical simulation results initiated in two different parts of the Charles River, in which accurate navigation information is provided to the vehicle. Fig. 14 shows the vehicle track from the perspective of the entire basin.The vehicle launch point was located 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 Fig. 15, 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.

Fig. 16 shows the vehicle's feature map. The "*" symbols denote the cells that contain candidate features. Fig. 17 is the visitation map of all cells that the vehicle has sensed. Finally, Fig. 18 shows the superposition of the two maps. The end of the

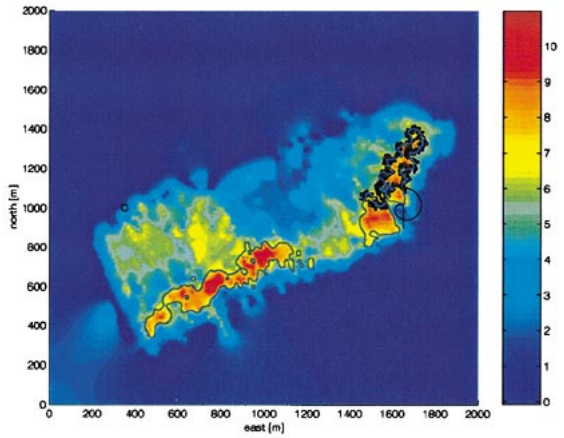


Fig. 19. Vehicle path for Mission 2. The starting location was [1650N, 900E].

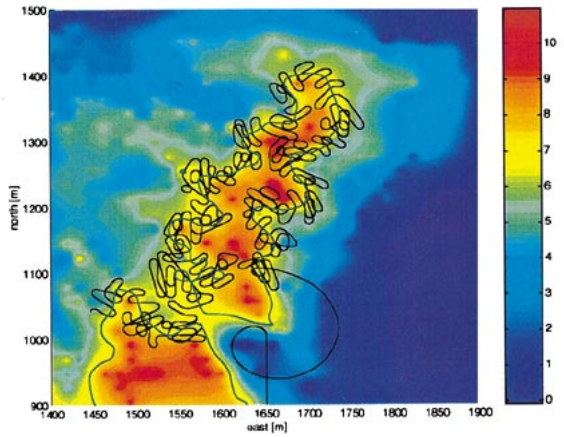


Fig. 20. Close-up of vehicle path for Mission 2.

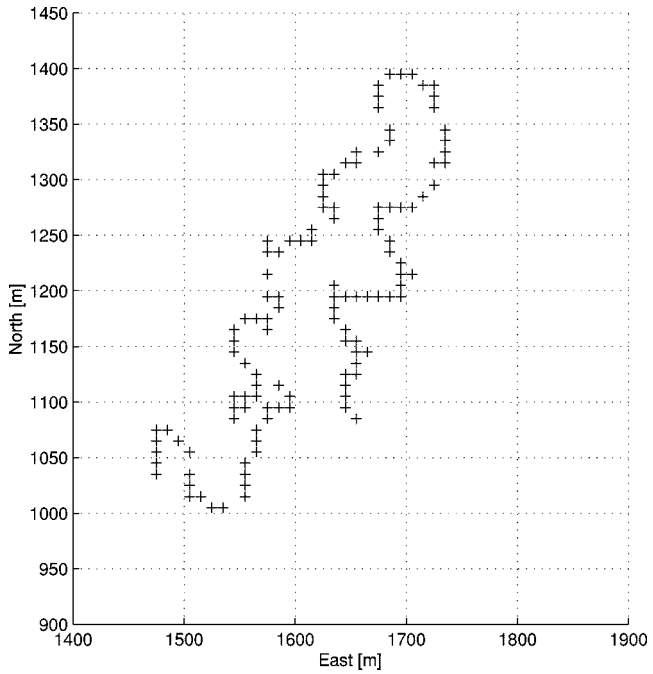


Fig. 21. Feature map for Mission 2.

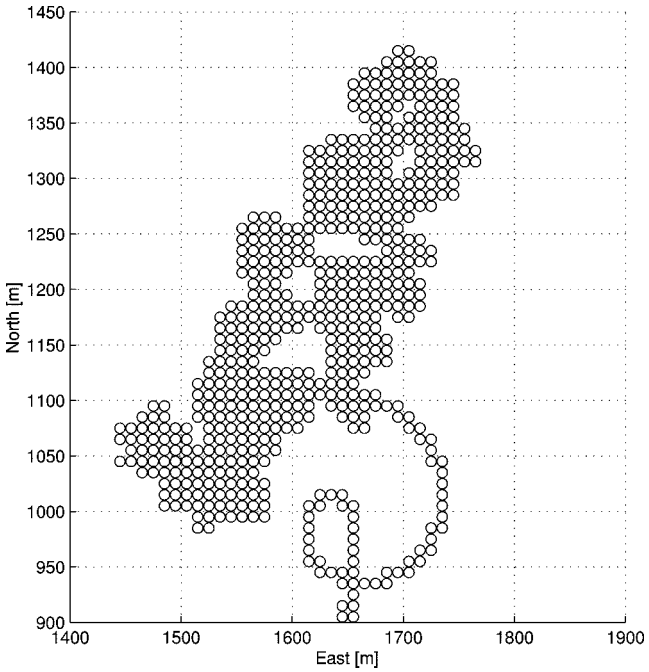


Fig. 22. Visitation map for Mission 2.

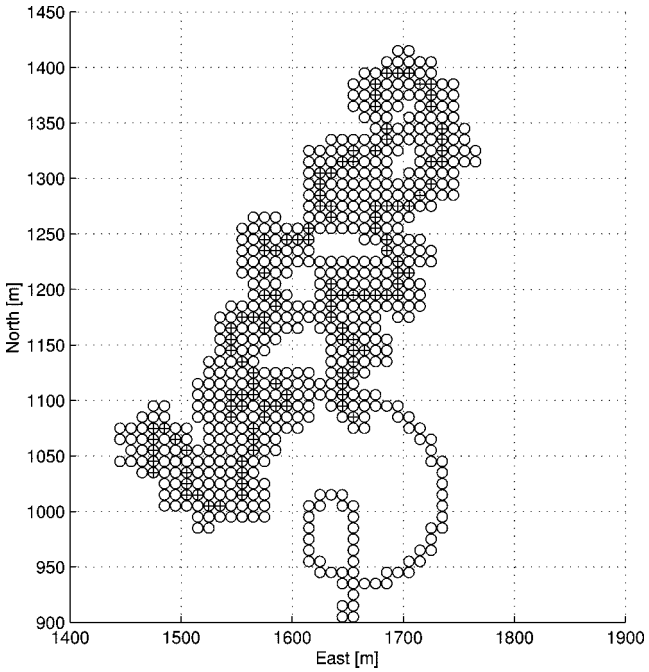


Fig. 23. Overlay of feature map and visitation maps for Mission 2.

mission occurred before all candidate cells had been exhausted. (The duration of the missions was restricted to 5 h to simulate a limited battery life.) Unsearched cells (candidate waypoints) can be noted at 650 north by 840 east and all along the feature cells at 1140 east. We can see from Fig. 15 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 Figs. 19–23, 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. In Fig. 20, the vehicle path can be seen in more detail. Note that the northeast trench is composed of two pits with a small saddle in

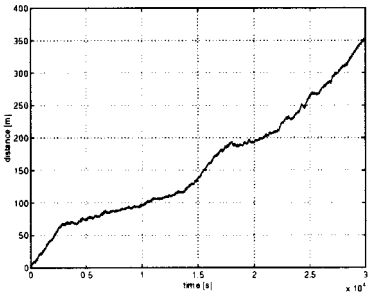


Fig. 24. Error growth in dead reckoning navigation system for Mission 3 (shown in Fig. 25). Note that, while the dead reckoner believed the vehicle was approximately 350 m away from the true position, the vehicle itself had actually lapped the feature and was working its way around the feature for a second time.

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. Figs. 21–23 are the *feature map*, *visitation map*, and the combination of the two (as used by the waypoint planner), respectively.

These results show that the adaptive feature mapping behavior works well as long as accurate navigation information is available. The next important question to ask is how well does this technique perform in the presence of navigation error and external currents? This section considers this question by showing the performance of the technique in one challenging run with fairly high levels of error. A more detailed performance analysis is contained in [8].

The form and effect of navigation error depends upon the type of navigation system employed. For AUV's such as the Odyssey II, navigation is either an externally referenced system, such as a long baseline system, or an internal system such as an inertial navigation system (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.* [32]). 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 turning 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).

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 within the current itself. The INS system cannot measure such a steady-state drift due to the lack of acceleration. A dead-reckoner which relies on a speed sensor that measures water speed relative to the vehicle will similarly be misled by the lack of relative motion between the vehicle and the current. Navigation errors build up quickly when the `trench_finder` behavior relies solely upon such a dead-reckoning system.

Figs. 24–26 show the combined effects of a $-1.1°$ heading bias, a 0.05-m/s speed sensor bias, and an external current of

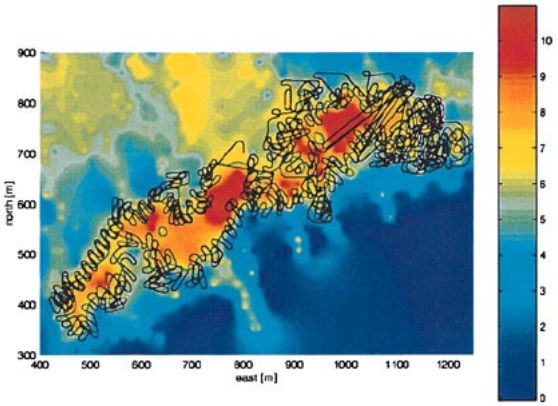


Fig. 25. Actual vehicle path for Mission 3, in which there was a heading bias of $-1.1°$, a speed bias of -0.05 m/s, and an external current of 0.004 m/s N, 0.11 m/s E. The vehicle started its mission at [700N, 800E] and ended at approximately [400N, 450E]. The total mission duration was 8.3 h. The total path length traveled by the vehicle was 24.5 km.

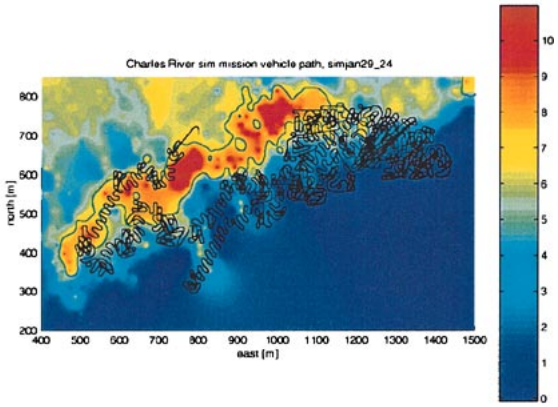


Fig. 26. Dead-reckoned vehicle path from Mission 3 (compare with Fig. 25). Note that while the dead-reckoner navigation error continued to increase, the `trench_finder` behavior continued to successfully map the feature.

0.004 m/s north and 0.11 m/s east. Mission duration was 8 h 20 min, starting at [700N, 800E] and finishing at approximately [400N, 450E], 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 previously mapped part of the trench during the last portion of the mission. This demonstrates `trench_finder`'s ability to maintain contact with a feature despite disturbances.

Additional simulations which illustrate different cost function weightings, cell sizes, and variation of other parameters are contained in [8].

VII. Conclusions and Future Research

This paper has presented a behavior-based approach to adaptive feature mapping. The concept of an "adaptive behavior" that incorporates mapping and prediction functions within a single behavior as part of the Odyssey II layered control architecture was introduced. The behavior was demonstrated to successfully track the 7-m contour in the Charles River despite the presence of navigation error and currents. The key step was to incorporate a grid-based map and a waypoint generation scheme within the adaptive feature mapping behavior. The adaptive behavior

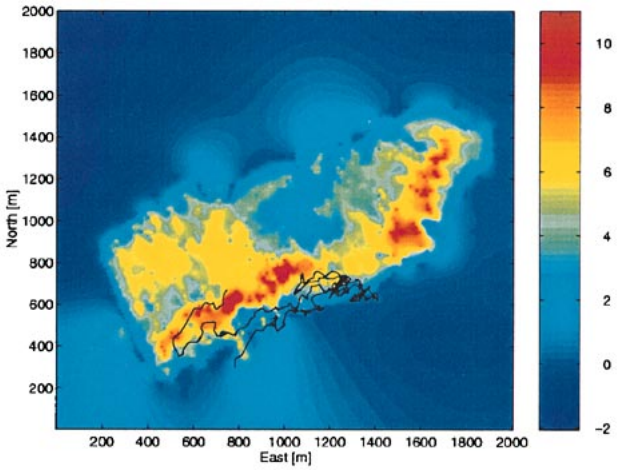


Fig. 27. Resulting estimated contour from the mission shown in Fig. 26. The correspondence between the extracted contours at the beginning and the end of the mission suggest the viability of using this method to reset the vehicle position and correct for dead-reckoning error growth.

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 paper of adaptively mapping the Charles River trench.

The simulation study described here is only one step in the development of new adaptive feature mapping capabilities for AUV's. Future testing with an actual vehicle is essential to verify the validity of the technique. An important characteristic of the work presented is that it was performed using the full operating version of the Odyssey II simulation and control software environment [5]. The new behavior could be tested on an actual Odyssey II vehicle if an opportunity for field trials became available.

Efforts to perform this task using "traditional" behaviors (without mapping and prediction) were unsuccessful [8]. 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 SCLC 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. One of the positive lessons learned from this work is that if one expands the definition of "what is a behavior" to incorporate simple planning and mapping capabilities, then more complex behavior is possible. By keeping the planning and mapping functions bound inside this adaptive behavior, the complexity is isolated from other behaviors.

The new adaptive feature mapping behavior provides an important building block for realization of improved mapping and sampling capabilities for AUV's. One exciting possibility for extension of this work is to consider the problem of tracking dynamic water column features [8]. 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 ensures that the vehicle will extend its search in the appropriate direction if or when the feature is lost and needs to be re-acquired. To predict where a given feature will be, the vehicle must have more information about the feature. This means that the adaptive behavior must also have some internal model of the feature and its expected behavior.

TABLE I
INPUT PARAMETERS FOR THE SIMULATIONS

Symbol or variable	Nominal value
search zone boundaries: $\begin{bmatrix} X_{west} & Y_{south} \\ X_{east} & Y_{north} \end{bmatrix}$	$\begin{bmatrix} 200 & 200 \\ 2000 & 1600 \end{bmatrix}$
Search mode	Spiral
Search Interval	15.75 m
Detection type (scalar, gradient)	Scalar
Detection threshold	7.0 m
Deadband	$\pm$0.02 m
Waypoint adjacency search radius	2 cells
Data filtering	Moving window
Map cell size	10 $\times$ 10 meters
Waypoint search method	LOOKUP
Contour vs. region search	CONTOUR
Search origin	[800N, 700E]

An alternative direction for future research is to consider the problem of using contours as navigational references. One possibility is that an AUV could use the new behavior presented here to be able to relocalize itself by finding and tracking a distinctive contour. This capability is suggested in Fig. 26, in which the vehicle overruns the same terrain as encountered during the initial part of its mission, executing nearly the same trajectory. This is further supported by Fig. 27, which displays the contour extracted by processing the data from the mission. This can be considered a type of "emergent behavior" resulting from fairly simple decision rules which demonstrates one of the desirable characteristics of using a behavior-based control approach. Concurrent mapping and localization using natural terrain features embodies one of the ultimate goals of mobile robotics research [31]: Can an AUV build a map of an unknown environment based on contour features and use that map for effective navigation?

APPENDIX
PARAMETER DEFINITIONS AND VALUES USED IN THE SIMULATION

The values of the `trench_finder` behavior inputs are listed in Table I and are described below:

- *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 $1/2\pi$ 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. 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 [8]. 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 ± 0.02 m, 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, while smaller cells yield a more detailed map, but at the cost of more vehicle time necessary to generate the map. A cell size of 10 m was used for the results in this paper.
- *Waypoint Search Method:* The waypoint search method determines how the vehicle ranks candidate waypoints. Methods available are precompiled lookup tables (either vehicle heading-dependent or heading-independent) or a cost/utility function.
- *Contour versus 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, a starting point near the center of the search area is selected automatically [8].

ACKNOWLEDGMENT

The authors would like to thank J. G. Bellingham, C. Chryssostomodis, and all the members of the MIT Sea Grant AUV Laboratory for their support.

REFERENCES

[1] J. G. Bellingham and T. R. Consi, "State configured layered control," in *Proc. IARP 1st Workshop on Mobile Robots for Subsea Environments*, Monterey, CA, Oct. 1990, pp. 75–80.
[2] J. G. Bellingham, T. R. Consi, R. Beaton, and W. Hall, "Keeping layered control simple," in *Proc. AUV'90*, 1990.
[3] 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 Conf. Autonomous Underwater Vehicles*, Cambridge, MA, 1994, pp. 148–156.
[4] J. G. Bellingham and D. Humphrey, "Using layered control for supervisory control of underwater vehicles," in *Proc. ROV '90*, Vancouver, BC, Canada, 1990, pp. 175–179.
[5] J. G. Bellingham and J. J. Leonard, "Task configuration with layered control," in *Proc. IARP 2nd Workshop on Mobile Robots for Subsea Environments*, Monterey, CA, May 1994, pp. 193–302.
[6] J. G. Bellingham and J. S. Willcox, "Optimizing AUV oceanographic surveys," in *IEEE Conf. Autonomous Underwater Vehicles*, Monterey, CA, 1996, pp. 391–398.
[7] A. A. Bennett, "Combining planning with reactive architectures in an autonomous underwater vehicle," in *Proc. Int. Symp. Unmanned Untethered Submersible Technology*, Sept. 1993, pp. 437–445.
[8] A. A. Bennett, "Feature relative navigation for autonomous underwater vehicles," Ph.D. dissertation, MIT, 1997.
[9] R. A. Brooks, "Visual map making for a mobile robot," in *IEEE Conf. Robotics and Automation*, 1985, pp. 824–829.
[10] ——, "Elephants don't play chess," *Robotics and Autonomous Systems*, pp. 3–15, June 1990.
[11] ——, "Intelligence without reason," Massachusetts Institute of Technology, Technical Report A.I. Memo No. 1293, April 1991.
[12] E. Burien, "Search methods for an autonomous underwater vehicle using scalar measurements," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.
[13] E. Burien, D. Yoerger, A. Bradley, and H. Singh, "Gradient search with autonomous underwater vehicles using scalar measurements," in *Proc. AUV '96*, 1996, pp. 86–98.
[14] N. Carver and V. Lesser, "The evolution of blackboard control architectures," *Expert Systems with Applications*, pp. 1–30, January 1994.
[15] R. Chatila and J. P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *IEEE Int. Conf. Robotics and Automation*, St. Louis, MO, March 1985, pp. 138–145.
[16] J. H. Connell, "SSS: A hybrid architecture applied to robot navigation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1992, pp. 2719–2724.
[17] T. Curtin, J. G. Bellingham, J. Catipovic, and D. Webb, "Autonomous ocean sampling networks," *Oceanography*, vol. 6, no. 3, pp. 86–94, 1993.

[18] G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *Autonomous Mobile Robots: Control, Planning, and Architecture*, S. S. Iyengar and A. Elfes, Eds. Los Alamitos, CA: IEEE Computer Soc. Press, 1991, pp. 254–277.
[19] A. J. Healey, R. B. Marco, and R. B. McGhee, "Autonomous underwater vehicle control coordination using a tri-level hybrid software architecture," in *Proc. IEEE Int. Conf. Robotics and Automation*, April 1996, pp. 2149–2159.
[20] B. Kamgar-Parsi, L. Rosenblum, F. Pipitone, L. Davis, and J. Jones, "Toward an automated system for a correctly registered bathymetric chart," *IEEE J. Oceanic Eng.*, vol. 14, pp. 314–325, Oct. 1989.
[21] D. Kortenkamp, R. P. Bonasso, and R. Murphy, Eds., *Artificial Intelligence and Mobile Robots*. Cambridge, MA: MIT/AAAI Press, 1998.
[22] J. E. Laird and P. S. Rosenbloom, "Integrating, execution, planning, and learning in soar for external environments," in *Proc. AAAI-90*, 1990, pp. 1022–1029.
[23] P. Maes, "Learning behavior networks from experience," in *Toward a Practice of Autonomous Systems: Proc. First Eur. Conf. Artificial Life*. Cambridge, MA: MIT Press, 1991, pp. 48–57.
[24] M. J. Mataric, "Behavior-based control: Examples from navigation, learning, and group behavior," *J. Experimental and Theoretical Artif. Intell.*, vol. 9, no. 2/3, 1997.
[25] H. P. Moravec and D. W. Cho, "A Bayesean method for certainty grids," in *AAAI Spring Symp. Robot Navigation*, 1989, pp. 57–60.
[26] D. Payton, D. Keirsey, D. Kimble, J. Krozel, and K. Rosenblatt, "Do whatever works: A robust approach to fault-tolerant autonomous control," *J. Appl. Intell.*, vol. 3, pp. 226–249, 1990.
[27] United Nations Environment Programme. (1997) Northeast Russian Petroleum Service Agency Facts. [Online]. Available: http://www.grid.unep.no/arci0025.htm
[28] R. A. Regan, "Autonomous minehunting and mapping at-sea testing," in *IEEE Oceans*, 1996, pp. 807–812.
[29] H. Schmidt, J. Bellingham, M. Johnson, D. Herold, D. Farmer, and R. Pawlowcisz, "Real-time frontal mapping with AUVs in a coastal environment," in *IEEE Oceans*, 1996, pp. 1094–1098.
[30] H. Singh, D. Yoerger, R. Bachmayer, A. Bradley, and W. Stewart, "Sonar mapping with the autonomous benthic explorer (ABE)," in *Proc. Int. Symp. Unmanned Untethered Submersible Technology*, Sept. 1995, pp. 367–375.
[31] C. M. Smith, J. J. Leonard, A. A. Bennett, and C. Shaw, "Feature-based concurrent mapping and localization for autonomous underwater vehicles," in *IEEE Oceans*, 1997.
[32] J. Vaganay, J. G. Bellingham, and J. J. Leonard, "Outlier rejection for autonomous acoustic navigation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 1996, pp. 2174–2181.
[33] J. S. Willcox, Y. Zhang, J. G. Bellingham, and J. Marshall, "AUV survey design applied to oceanic deep convection," in *IEEE Oceans*, 1996, pp. 949–954.

Andrew A. Bennett (S'95–A'97) received the B.S.M.E. degree from the Massachusetts Institute of Technology (MIT), Cambridge, in 1985, the M.S. degree from Stanford University, Stanford, CA, in 1986, and the Ph.D. degree from MIT in 1997.

He is a Senior Engineer at I. S. Robotics, Somerville, MA. His work addresses the design and implementation of high-level control algorithms, electro-mechanical systems, and sensor systems for a variety of autonomous systems.

John J. Leonard (S'87–M'87) received the B.S.E. degree in electrical engineering and science from the University of Pennsylvania, Philadelphia, in 1987 and the Ph.D. degree in engineering science from the University of Oxford, Oxford, U.K., in 1994.

He is an Assistant Professor of Ocean Engineering at the Massachusetts Institute of Technology (MIT), Cambridge. His research addresses the problems of navigation and mapping for autonomous underwater vehicles. From 1991 to 1996, he was a Postdoctoral Fellow and Research Engineer in the Underwater Vehicles Laboratory of the MIT Sea Grant College Program.

Dr. Leonard is a member of the IEEE Robotics and Automation and Oceanic Engineering Societies and the Acoustical Society of America.