Autonomous Control of an Autonomous Underwater Vehicle Towing a Vector Sensor Array

Michael R. Benjamin
NAVSEA Division Newport
Newport RI 02841
Dept of Mechanical Engineering, MIT
Cambridge MA 02139
Email: mikerb@csail.mit.edu

David Battle
Don Eickstedt
Dept. of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge MA 02139
Email: davidb,eicksted@mit.edu

Henrik Schmidt
Arjuna Balasuriya
Dept. of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge MA 02139
Email: henrik,arjunab@mit.edu

Abstract— **This paper is about the autonomous control of an autonomous underwater vehicle (AUV), and the particular considerations required to allow proper control while towing a 100-meter vector sensor array. Mission related objectives are tempered by the need to consider the effect of a sequence of maneuvers on the motion of the towed array which is thought not to tolerate sharp bends or twists in sensitive material. We describe and motivate an architecture for autonomy structured on the behavior-based control model augmented with a novel approach for performing behavior coordination using multi-objective optimization. We provide detailed in-field experimental results from recent exercises with two 21-inch AUVs in Monterey Bay California.**

I. Introduction

A. Motivation

The primary motivation for this work is to achieve the ability to deploy large sets of autonomous mobile marine platforms over a wide area of the ocean environment and over a long period of time with little or no human supervision. Concerns over effective coverage, communication range and safe operation of the platforms are all primary motivations of an effective form of autonomous control. The long duration and unpredictable nature of the environment require the vehicles to adapt their missions and behave autonomously as events unfold. Conversely, practical concerns of marine operations over large areas require an element of operator predictability over the course of time. These two characteristics can be at odds with each other in practice, but can be tempered by effective periodic communication through a network of fixed and mobile nodes co-deployed in a coordinated manner designed to balance individual platform and network objectives.

In the work described here, we focus on autonomous control and communications with a single AUV fitted with a vector sensor array (Fig. 1) operating in an environment with other AUVs having access to one or more fixed or mobile gateway buoys, all equipped with acoustic modems. We have implemented and describe here a set of autonomy behaviors sufficient for long term deployment, and dynamic re-deployment from ship or shore station, and report results from recent in-water deployment exercises in Monterey Bay.

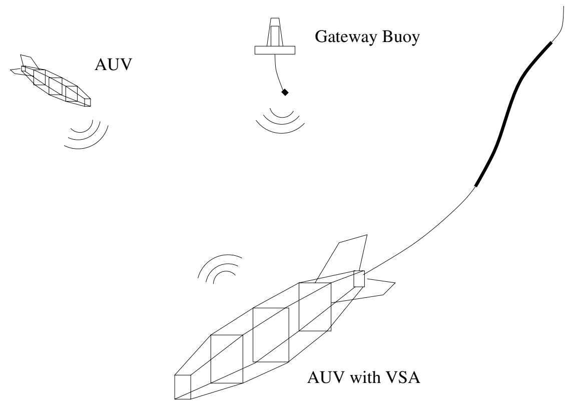


Fig. 1. A set of autonomous underwater vehicles is deployed over a large area and time period to monitor underwater marine traffic. One vehicle is equipped with a vector sensor array (VSA) to provide contact and track solution information to be shared via acoustic link to other AUVs and back to shore via a gateway buoy, which itself could be mobile. The VSA-equipped node or platform may also be redirected by other nodes via acoustic link to deploy in another region based on information obtained by other nodes. The VSA-equipped vehicle is responsible for efficiently meeting its deployment obligations while avoiding sets of maneuvers that would compromise the safety of the sensing and communication structure of the array.

The towed vector sensor array is one of the main elements of this larger objective, which aims to exploit the latest advances in AUV technology to deploy an autonomous vehicles with endurance of the order of several months. A critical element of this approach includes sensor scalability to the smaller platform sizes afforded by AUVs and efficient energy use to obtain the required levels of persistence. Vector sensor technology facilitates scalability by virtue of its higher theoretical gain for a given aperture length.

B. Background of Behavior-based Control

In behavior-based systems, platform control is the result of a set of independent, specialized modules working together to choose appropriate vehicle actions. It has been viewed as an alternative to the traditional sense-plan-act control loop where decision-making and planning are performed on a single world model that is built up and maintained over time. Commonly cited virtues of behavior-based systems include:

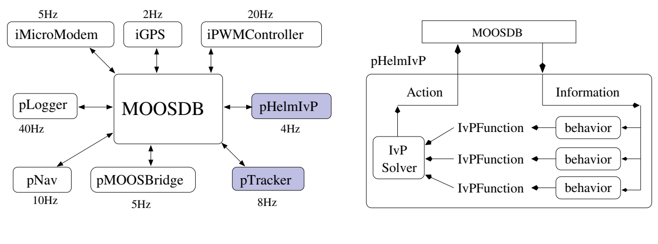
the ease of development of the independent modules, the lack of a single complex world model, and the potential for a highly reactive vehicle with certain behaviors triggered by the appropriate events in a dynamic environment. The origin of these systems is commonly attributed to Brooks' "subsumption architecture" in [1]. Since then, it has been used in a large variety of applications including: indoor robots, e.g., [2], [3], [4], [5], land vehicles, e.g., [6], and marine vehicles, e.g., [7], [8], [9], [10], [11]. Action selection is the process of choosing a single action for execution, given the output of each behavior. Action selection has well known difficulties as reported in [4] and [6] for which multi-objective optimization is thought to provide some countermeasure.

II. Technical Approach

A. The MOOS-IvP Autonomy Architecture

This work uses the MOOS-IvP architecture for autonomous control. MOOS-IvP is composed of the Mission Oriented Operating Suite (MOOS), a open source software project for coordinating software processes running on an autonomous platform, typically under GNU/Linux. MOOS-IvP also contains the IvP Helm, a behavior-based helm that runs as a single MOOS process and uses multi-objective optimization with the Interval Programming (IvP) model for behavior coordination, [12]. See [13] and [14] for other examples of MOOS-IvP on autonomous marine vehicles.

A MOOS community contains processes that communicate through a database process called the MOOSDB, as shown in Fig. 2(a). MOOS ensures a process executes its "Iterate" method at a specified frequency and handles new mail on each iteration in a publish and subscribe manner. The IvP Helm runs as the MOOS process pHelmIvP (Fig. 2(b)). Each iteration of the helm contains the following steps: (1)



(a) A MOOS Community (b) The pHelmIvP process

Fig. 2. The IvP Helm runs as a process called pHelmIvP in a MOOS community. MOOS may be composed of processes for data logging (pLogger), data fusion (pNav), actuation (iPWMController), sensing (iGPS), communication (pMOOSBridge, iMicroModem), and much more. They can all be run at different frequencies as shown.

mail is read from the MOOSDB, (2) information is updated for consumption by behaviors, (3) behaviors produce an objective function if applicable, (4) the objective functions are resolved to produce a single action, and (5) the action is posted to the MOOSDB for consumption by low-level control MOOS processes. The behaviors used in this work are discussed in Section V.

B. The Back-Seat Driver Paradigm

While low-level control tasks such as navigation, depth-keeping and vehicle safety were relegated to the AUV main vehicle computer, all high-level control inputs were derived from a separate vehicle payload computer running the MIT MOOS-IvP system. This computer incorporated a Linux-based CPU for general-purpose computational tasks, such as the multi-objective helm, as well as a specialized 8 GFlop vector processor to support heavy signal processing loads such as spectrum analysis and broad-band beamforming.

III. General Properties of the Helm and Behaviors

A. General Properties of the Helm

The IvP Helm, as a software module, can be viewed as an interface between the set of behaviors and the MOOSDB and the other system components connected to MOOSDB. Its primary function at run time is to arbitrate between behaviors by soliciting objective functions from each, over a common decision space, and performing multi-objective optimization. Besides posting the resulting decision to the MOOSDB, the helm also posts other variable-value pairs based on requests of the behaviors generated during the behaviors' function-producing iteration. This includes state information that one behavior may communicate to another across iterations, such as a condition that marks the completion of one behavior and triggers the activation of another as described by the CONDITION and ENDFLAG parameters below.

The helm, upon startup, reads a configuration, i.e., mission, file with sets of parameters. The key parameters for behaviors used in experiments reported here are described in the following sections. An important component of our research objectives is to allow behaviors to be adaptive not only to environment events, but also to periodic high-level commands from field control. This means a mission file is not a static script with a start and completion, but more aptly described as a state space with an initial state and conditions for migrating between states based on events; mission-control, environment or otherwise.

B. Universal Behavior Parameters

The following parameters describe properties inherited by all behaviors described in later sections.

PRIORITY: The priority weight of the produced objective function. A behavior may also be implemented to dynamically determine its own priority weight.

DURATION: The time duration before the behavior is marked completed. If none provided, the behavior will not time-out. The clock begins when the behavior first becomes active.

CONDITION: A condition that must be satisfied for the behavior to be active. It is a equal-separated pair such as DEPLOY=true. If more than one condition is given, they all must be satisfied. The variables are MOOS variables and the helm automatically subscribes to a variable appearing as a behavior condition.

RUNFLAG: A variable and a value posted while the behavior is active. It is a equal-separated pair such as TRANSITING=true. More than one runflag may be provided and can be used to satisfy or block the conditions of other behaviors.

ENDFLAG: A variable and a value posted when the behavior has completed. The circumstances causing completion are unique to the individual behavior, but if any behavior has a DURATION specified, the endflags are posted upon time-out, which occurs when the duration specified by the DURATION parameter has been exceeded. The value of this parameter is a equal-separated pair such as ARRIVED_HOME=true.

UPDATES: A MOOS variable from which updates to behavior parameters are read from after the behavior has been initially instantiated and configured at the helm startup time. Any parameter and value pair that would have been legal at startup time is legal at runtime. This is one of the primary hooks to the helm for mission control; the other being the behavior conditions described above.

IV. Network and Field Control

A key component of our objective of ubiquitous, autonomous mobile marine sensing platforms is the periodic interface to humans via Network and Field Control (NAFCON) detailed briefly here.

A. The WHOI Acoustic Modem and NAFCON Interface

The AUVs are equipped with acoustic modems from the Woods Hole Oceanographic Institute (WHOI) that implement an adaptive decision feedback equalizer with integrated Doppler and error-correction to afford 80-bps frequency hopped-FSK mode communications. The modem provides the user with the tools necessary to create a simple time-division, multiple-access (TDMA) network for master-slave polled systems, or a random-access peer-to-peer network. Each communication transaction includes a short network packet called a cycle-initialization. This specifies the source, destination and data rate of the packet to follow. In this experiment, Compact Control Language (CCL) developed by WHOI is used to handle the bandwidth limitations underwater.

In these experiments, two primary types of messages were sent by the NAFCON to the AUV; deploy and prosecute. The deploy message instructs the AUV as to where it should operate for efficient use of the field. The message includes latitude, longitude, and depth of deployment. A prosecute message is sent from NAFCON when a target of interest has been detected and localized to an area of uncertainty. The goal of the prosecution is to better localize, classify or track the target. Similarly, the AUV reports to the NAFCON by sending status reports, contact reports and track reports.

The CCL messages are decoded by the modem driver and published as a variable in the MOOSDB (NAFCON_MESSAGES). The pNAFCON MOOS process subscribes to the NAFCON_MESSAGES string variable and extracts details of the message sent by the NAFCON. Based on the message contents, pNAFCON publishes mission control messages and set flags in the MOOSDB to allow the helm to activate relevant behaviors.

B. Autonomy States and State-Transitions

The vehicle mission specification used in these experiments was comprised of a set of states, behaviors operating in those states, and events from the field and NAFCON for transitioning between states. The states are shown in Fig. 3 and described below.

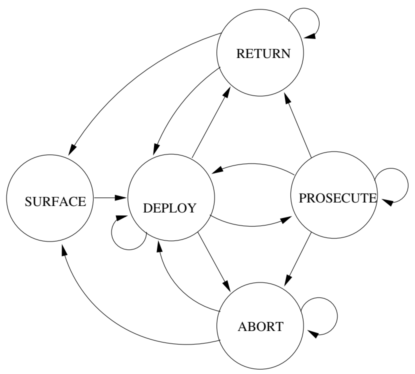


Fig. 3. The five main vehicle autonomy states. Each state is transitioned into with the receipt of a NAFCON message of the same name. Each state is comprised of one or more vehicle behaviors responsible for achieving the objectives of that state.

DEPLOY-STATE: Entered on receipt of the DEPLOY message. The vehicle transits to its deployment location and begins to loiter waiting for a prosecute message.

PROSECUTE-STATE: In this mode, the vehicle transits toward the currently estimated current position of the target. If the vehicle gets within a predetermined range of the estimated target position, it begins a loiter maneuver waiting for an acoustic detection. If an acoustic detection is made, the vehicle will maneuver to track the target until it moves outside its operational area, it loses the target, or a RETURN or ABORT message is received.

ABORT-STATE: The vehicle will transit toward its abort waypoint using the waypoint behavior described in Section V. When the waypoint is reached, a SURFACE message will be generated and the vehicle will enter the SURFACE state. The non-monotonic radius and duration-limit parameters are utilized to handle errant cases where the vehicle misses the abort point.

SURFACE-STATE: SURFACE: The system initially begins in this state waiting for a DEPLOY message. The SURFACE message is also internally generated when the vehicle reaches either its return or abort waypoints.

RETURN-STATE: The vehicle will transit toward its return waypoint using the waypoint behavior described in Section V. When the waypoint is reached, a SURFACE message is generated and the vehicle will enter the SURFACE state. The non-monotonic radius and duration-limit parameters are utilized to handle errant cases where the vehicle misses the return point.

Each state is comprised of one or more vehicle behaviors responsible for achieving the objectives of that state. These behaviors are described next.

V. Helm Behaviors

A. The Waypoint Behavior

The waypoint behavior is for transiting to a set of specified waypoints. The objective function produced by this behavior is defined over the 2D action space given by possible heading and speed choices (Fig 5). The following parameters are defined for this behavior:

POINTS: A list of x,y pairs given as points in 2D space; units in meters.

SPEED: The desired speed, in meters/second, at which the vehicle travels through the points.

CAPTURE_RADIUS: The radius tolerance, in meters, for satisfying the arrival at a waypoint.

ORDER: The order in which the waypoints are traversed. Either "reverse" or the default "normal".

LEAD: For track-line following, this is the distance, in meters, from the *perpendicular intersection point* to the next waypoint. The *perpendicular intersection point* is the point on the line given by the current and previous waypoint that is closest to the current vehicle position. The vehicle steers toward this point. If this point extends beyond the next waypoint, the steering point is exactly the next waypoint. If heading toward the first waypoint, this steering point is just that waypoint.

REPEAT: The number of times the vehicle will traverse through the set of waypoints, proceeding to the 1st waypoint after the nth waypoint has been hit.

NM_RADIUS: The non-monotonic radius is the capture radius distance within which a detection of increasing distances to the waypoint is treated as a waypoint arrival. As a rule of thumb, a distance of twice the arrival radius is used (Fig. 4).

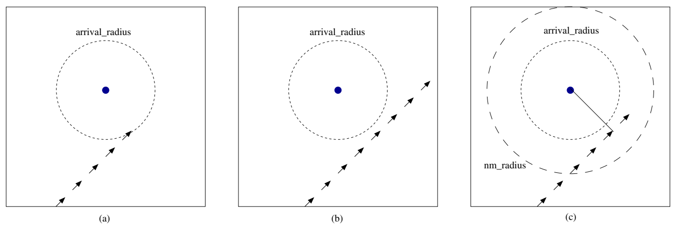


Fig. 4. (a) a successful waypoint arrival by achieving proximity less than the capture radius. (b) a missed waypoint likely resulting in the vehicle looping back to try again. (c) a missed waypoint but capture declared anyway when the distance to the waypoint begins to increase and the vehicle is within the *non-monotonic radius*.

B. The Loiter Behavior

This behavior is used for transiting to and repeatedly traversing a set of waypoints forming a convex polygon. Typically the polygon is a hexagon forming a desired loiter region. Measures are described below to ensure this behavior robustly handles dynamic exit and re-entry modes when or if the vehicle diverges from the loiter region due to external events. And it is dynamically reconfigurable to allow a mission control module to repeatedly reassign the vehicle to different loiter regions by using a single persistent instance of the behavior. The following parameters are defined for this behavior in addition to the general behavior parameters described previously:

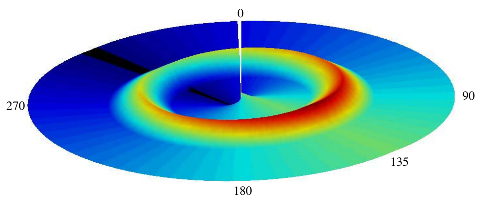


Fig. 5. The objective function produced by the waypoint behavior is defined over possible heading and speed values. Here is an objective function favoring maneuvers to a waypoint 135 degrees from the current vehicle position and favoring speeds closer to the mid-range of capable vehicle speeds. Higher speeds are represented farther radially out from the center.

POLYGON: A list of x,y pairs indicating points in 2D space. Units are in in meters and must describe a convex polygon; As an alternative to listing a sequence of points, a orbit-style polygon can be given by four values the x and y position, the radius, and the number of points on the polygon.

SPEED: See the waypoint behavior.

CAPTURE_RADIUS: See the waypoint behavior.

CLOCKWISE: If "true", the behavior will influence the vehicle in a clockwise direction around the polygon.

NM_RADIUS: See the waypoint behavior.

ACQUIRE_DIST: Distance between the vehicle and polygon that will trigger the behavior into *acquire* mode. Concerns both the cases when the vehicle is inside as well as outside the polygon; the re-acquire algorithms are different however.

When the behavior is active, it is in either one of two modes; the *acquire* mode or *normal* mode. In the normal mode it is merely proceeding to the next waypoint on the polygon. In the acquire mode, each iteration begins by determining a polygon re-entry vertex, to minimize the angle to the following waypoint. The acquire point depends on the chosen direction of polygon traversal, as shown in Figure 6.

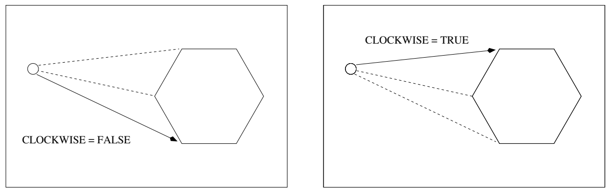


Fig. 6. In the *acquire* mode, the polygon points are evaluated for suitability in terms of a smooth entry trajectory. Only the "viewable" points, those viewable if the polygon were an opaque object and the viewer were at the current vehicle location, are contenders. The contenders are rated on the follow-on angle given the desired clockwise or counter-clockwise loiter direction. Larger follow-on angles are preferred as shown.

When in the acquire mode and *outside* the polygon, the chosen vertex is the one most tangential in either the clockwise or counter-clockwise direction as shown in the figure. When the vehicle is *inside* the polygon, the chosen vertex is the one which forms the most obtuse angle between the current vehicle position, the vertex, and the follow-on vertex. Unlike the case when outside the polygon, the chosen vertex changes as the vehicle makes progress back to the polygon perimeter. The effect is for the vehicle to spiral out to the perimeter for the smoothest re-entry into a normal loitering path (See Fig. 11).

The circumstance most common for triggering the acquire mode is the initial assignment to the vehicle to loiter at a new given region in the X,Y plane. This assignment *could* occur while the vehicle happens to already be within the polygon for a number of reasons. Furthermore, the vehicle could be driven off the polygon loiter trajectory due to environmental (wind or current) forces or the temporary dominance of other vehicle behaviors such as collision avoidance or tracking of another vehicle. Once the behavior enters the acquire mode, it remains in this mode until arriving at the first waypoint (defined by the arrival and non-monotonic radii settings), and switches to normal mode until the acquire mode is re-triggered or the behavior run conditions are no longer met.

C. The Memory-Turn-Limit Behavior

The objective of the Memory-Turn-Limit behavior is to avoid vehicle turns that may cross back on its own path and risk damage to the towed array. Its configuration is determined by the two parameters described below which combine to set a vehicle turn radius limit. However, it is not strictly described by a limited turn radius; it stores a time-stamped history of recent recorded headings and maintains a *heading average*, and forms its objective function on a range deviation from that average. This behavior merely expresses a preference for a particular heading. If other behaviors also have a heading preference, coordination/compromise will take place through the multi-objective optimization process. The following parameters are defined for this behavior:

MEMORY_TIME: The duration of time for which the heading history is maintained and heading average calculated.

TURN_RANGE: The range of heading values deviating from the current heading average outside of which the behavior reflects sharp penalty in its objective function.

The heading history is maintained locally in the behavior by storing the currently observed heading and keeping a queue of n recent headings within the MEMORY_TIME threshold. The heading average calculation below handles the issue of angle wrap in a set of n headings $h_0 \ldots h_{n-1}$ where each heading is in the range $[0, 359]$.

$$\text{heading\_avg} = \text{atan2}(s, c) \cdot 180/\pi,$$

where s and c are given by:

$$s = \sum_{k=0}^{n-1} \sin\left(h_k \pi/180\right)), \qquad c = \sum_{k=0}^{n-1} \cos\left(h_k \pi/180\right)).$$

The vehicle turn radius r is not explicitly a parameter of the behavior, but is given by:

$$r = v/((u/180)\pi),$$

where v is the vehicle speed and u is the turn rate given by:

$$u = \text{TURN\_RANGE}/\text{MEMORY\_TIME}.$$

The same turn radius is possible with different pairs of values for TURN_RANGE and MEMORY_TIME. However, larger values of TURN_RANGE allow sharper initial turns but temper the turn rate after the initial sharper turn has been achieved.

D. The Go-To-Depth Behavior

This behavior will drive the vehicle to a sequence of specified depths and duration at each depth. The duration reflects the time at depth *after* achieving that depth. The behavior examines the current vehicle depth and declares the target depth achieved if it is within the delta given by CAPTURE_DELTA. The behavior also stores the recorded depth from the prior behavior iteration, and if the target depth is between the prior depth and current depth, the depth is considered to be achieved regardless of whether the current depth is within the CAPTURE_DELTA. The following parameters are defined for this behavior:

DEPTH: A list of depth-duration pairs. The duration applies from the time the depth is first achieved.

REPEAT: The number of times the vehicle will traverse through the evolution of depths. The default value is zero.

CAPTURE_DELTA: The depth difference between current and target depth required to declare a target depth achieved.

E. The Periodic-Speed Behavior

This behavior will periodically influence the speed of the vehicle while remaining neutral at other times. The timing is specified by a given period length in which the influence is on, and a gap length specifying the time between periods. It was conceived for use on an AUV equipped with an acoustic modem to periodically slow the vehicle to reduce self-noise and reduce communication difficulty. The following parameters are defined for this behavior:

PERIOD_LENGTH: Period duration during which the behavior produces an objective function over the desired speed.

PERIOD_GAP: Time duration in seconds between periods.

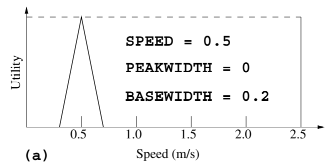
PERIOD_SPEED: The desired speed in meters/second.

PERIOD_PEAKWIDTH: The width of the peak in meters/second in the speed objective function. See Fig. 7.

PERIOD_BASEWIDTH: The width of the base, in meters/second in the speed objective function. See Fig. 7.

VI. Experimental Results

The results reported in the section are from field exercises in Monterey Bay in August 2006. The tow platform for the series of experiments described was an MIT 21 inch diameter, 158 inch-long AUV custom built by Bluefin Robotics in Cambridge Mass (Fig. 8). The vehicle possessed a single ducted thruster capable of propelling the vehicle and array



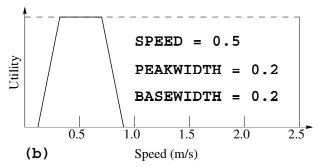


Fig. 7. In (a) the preference is a for a particular speed and a slight tolerance in either direction. In (b) the preference is for a particular range of speeds with a slight tolerance either way.

at better than 3 knots. The navigation system of the vehicle consisted of a GPS receiver (for surface fixes), a Leica digital 3-axis fluxgate compass, Crossbow Attitude and Heading Reference System (AHRS), and bottom-locking RDI- Teledyne Workhorse Navigator 300 kHz Doppler Velocity Log (DVL). Dead-reckoning was achieved by fusing the above sensor inputs via a proprietary Bluefin algorithm.

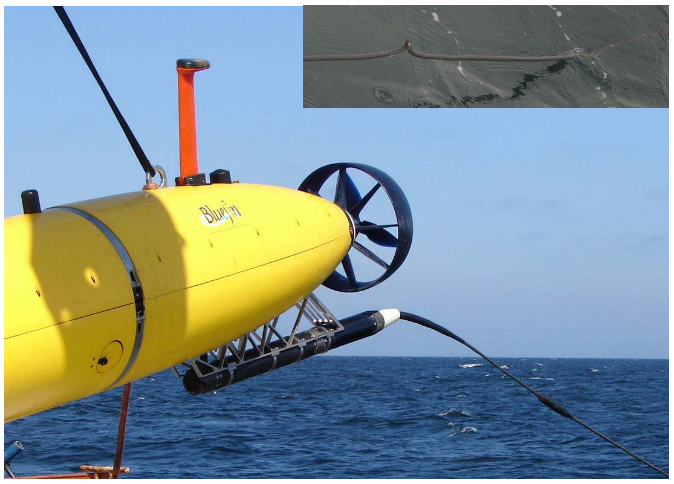


Fig. 8. The MIT 21-inch Bluefin autonomous underwater vehicle modified to tow a vector sensor array. The inset photo shows a severe twist incurred on the array after vehicle calibration runs in the first few days of field exercises. Vehicle control during the calibration runs was not conducted using the autonomy architecture described here, and several sharp (less than 40 meter radius) turns were performed on the vehicle that likely contributed to the twist shown here and several others. The inset photo was shot the day prior to the main photo which reflects the repair to the array twist with a cast-like patch halfway down in the photo. No such severe twists or minor twists were observed when the autonomy methods reported in this paper were used. Photos from Monterey Bay exercises August 2006.

The prototype array discussed here was designed to support a speed range from 0 to approximately 4 knots. Zero speed operation, wherein the array floats up behind the vehicle (which may be bottomed), invites the possibility of low-power detection modes and extended endurance. The particular VSA tested in MB'06 consisted of 47 meters of lightweight tow cable with integral optical fiber communications, an 8 meter vibration isolation module (VIM), 30 meters of acoustic array, and 15 meters of stability-enhancing drogue. See Figs. 8 and 9. In engineering the array and its mechanical interface to the Bluefin 21 vehicle, particular attention was payed to the issue of stability, which is typically poor for low-tension (low-drag/low-speed) systems of this type. Such instabilities are well known to be speed dependent and, in this case, dictated the selection of array materials in addition to a restricted maneuvering envelope for the AUV.

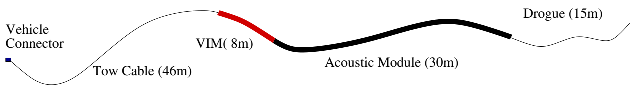


Fig. 9. The vector sensor array is roughly 100 meters in total length composed of four sections leading out from the AUV: a 47 meter tow cable section, an 8 meter vibration isolation module, a 30 meter acoustic module, and a 15 meter drogue at the end.

The tow cable section of the VSA was composed, in part, of a thin steel tube surrounding fiber optic data cables. This array section was thought to be vulnerable to damage if twists or kinks were to occur during operation. We report here two deployments using the MOOS-IvP autonomy module. Results from deployments are described in Figs. 10 and 11.

VII. Conclusion

This paper has investigated the problem of controlling an autonomous underwater vehicle towing a 100-meter vector sensor array. We described a novel method of behavior-based control using multi-objective optimization and a novel set of vehicle behaviors that were demonstrated in field exercises to effectively control an AUV under different mission circumstances over hours of operation. This paper also provides, to our knowledge, the first ever demonstration of such a system on a physical marine platform. Further research is ongoing to explore the robustness of this method in more complex navigation scenarios, both in simulation and on the water.

Acknowledgements

This work was supported by the Office of Naval Research (ONR), under Contract N00014-05-G-0106 Delivery Order 008, "PLUSNET: Persistent Littoral Undersea Surveillance Network", and by Dr. Don Wagner at ONR.

References

[1] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, April 1986.
[2] R. C. Arkin, "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior," in *Proceedings of the IEEE Conference on Robotics and Automation*, Raleigh, NC, 1987, pp. 264–271.
[3] R. C. Arkin, W. M. Carter, and D. C. Mackenzie, "Active Avoidance: Escape and Dodging Behaviors for Reactive Control," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 1, pp. 175–192, 1993.
[4] P. Pirjanian, "Multiple Objective Action Selection and Behavior Fusion," Ph.D. dissertation, Aalborg University, 1998.
[5] J. Riekki, "Reactive Task Execution of a Mobile Robot," Ph.D. dissertation, Oulu University, 1999.
[6] J. K. Rosenblatt, "DAMN: A Distributed Architecture for Mobile Navigation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1997.
[7] A. A. Bennet and J. J. Leonard, "A Behavior-Based Approach to Adaptive Feature Detection and Following with Autonomous Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 2, pp. 213–226, April 2000.
[8] M. Carreras, J. Batlle, and P. Ridao, "Reactive Control of an AUV Using Motor Schemas," in *International Conference on Quality Control, Automation and Robotics*, Cluj Napoca, Rumania, May 2000.
[9] R. Kumar and J. A. Stover, "A Behavior-Based Intelligent Control Architecture with Application to Coordination of Multiple Underwater Vehicles," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Cybernetics*, vol. 30, no. 6, pp. 767–784, November 2001.

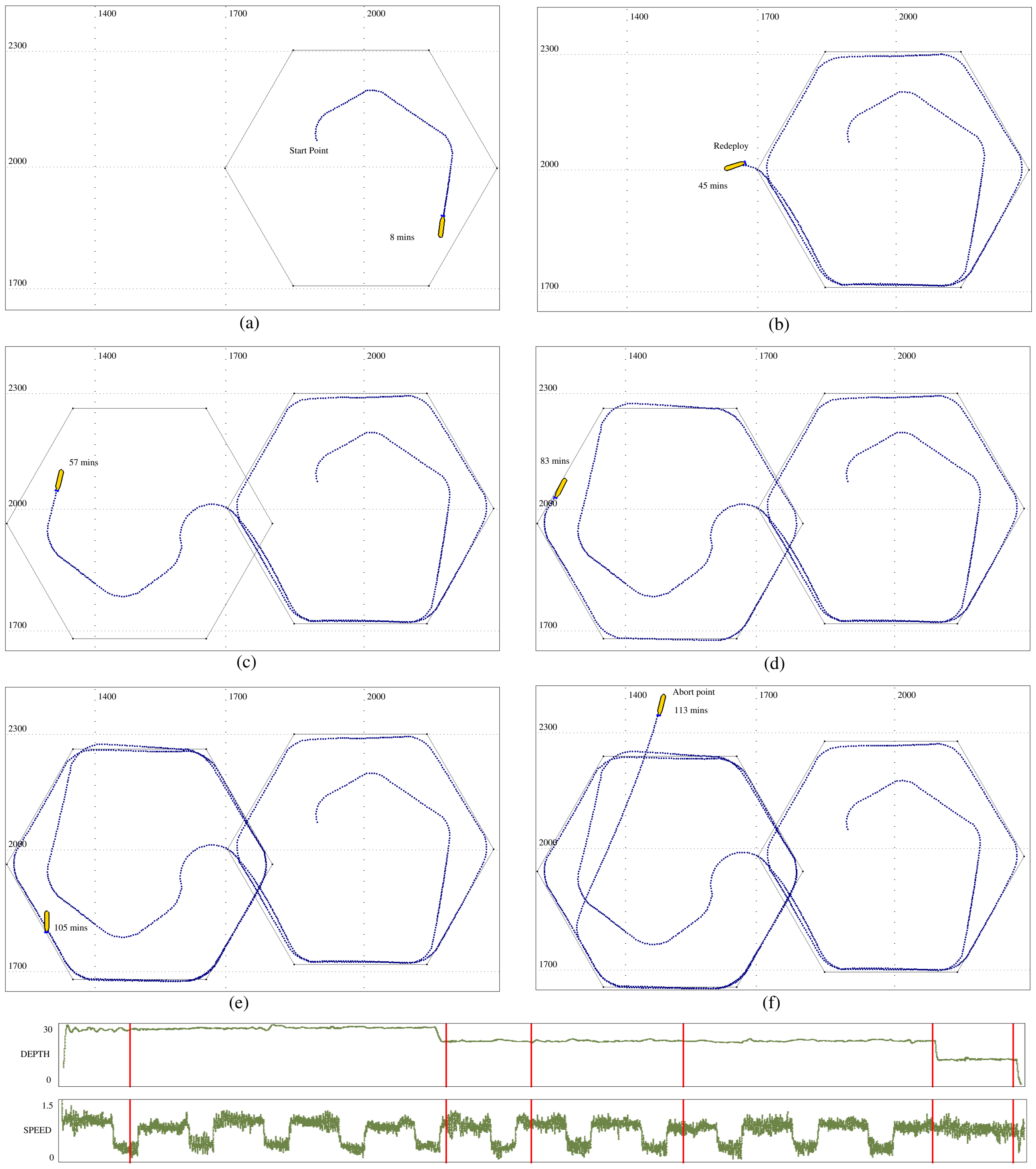


Fig. 10. Monterey Bay experiments with MIT 21-inch Bluefin AUV towing a vector sensor array. In (a) the AUV has completed its spiral out to shown hexagon loiter region. In (b) the AUV has been loitering for 45 minutes and receives a command to redeploy. In (c) the AUV has redeployed and completed its spiral out to the shown hexagon loiter region. In (d) the vehicle has nearly completed its first traversal of the hexagon loiter region. In (e) the vehicle receives a new command to redeploy to an abort point. In (f) the vehicle approaches the abort point and surfaces. The depth and speed of the vehicle are plotted on the bottom versus time with the six frames in (a) thru (f) marked accordingly.

[10] J. K. Rosenblatt, S. B. Williams, and H. Durrant-Whyte, "Behavior-Based Control for Autonomous Underwater Exploration," *International Journal of Information Sciences*, vol. 145, no. 1-2, pp. 69–87, 2002.

[11] S. B. Williams, P. Newman, G. Dissanayake, J. K. Rosenblatt, and H. Durrant-Whyte, "A decoupled, distributed AUV control architecture," in *Proceedings of 31st International Symposium on Robotics*, Montreal, Canada, 2000, pp. 246–251.

[12] M. R. Benjamin, "Interval Programming: A Multi-Objective Optimization Model for Autonomous Vehicle Control," Ph.D. dissertation,

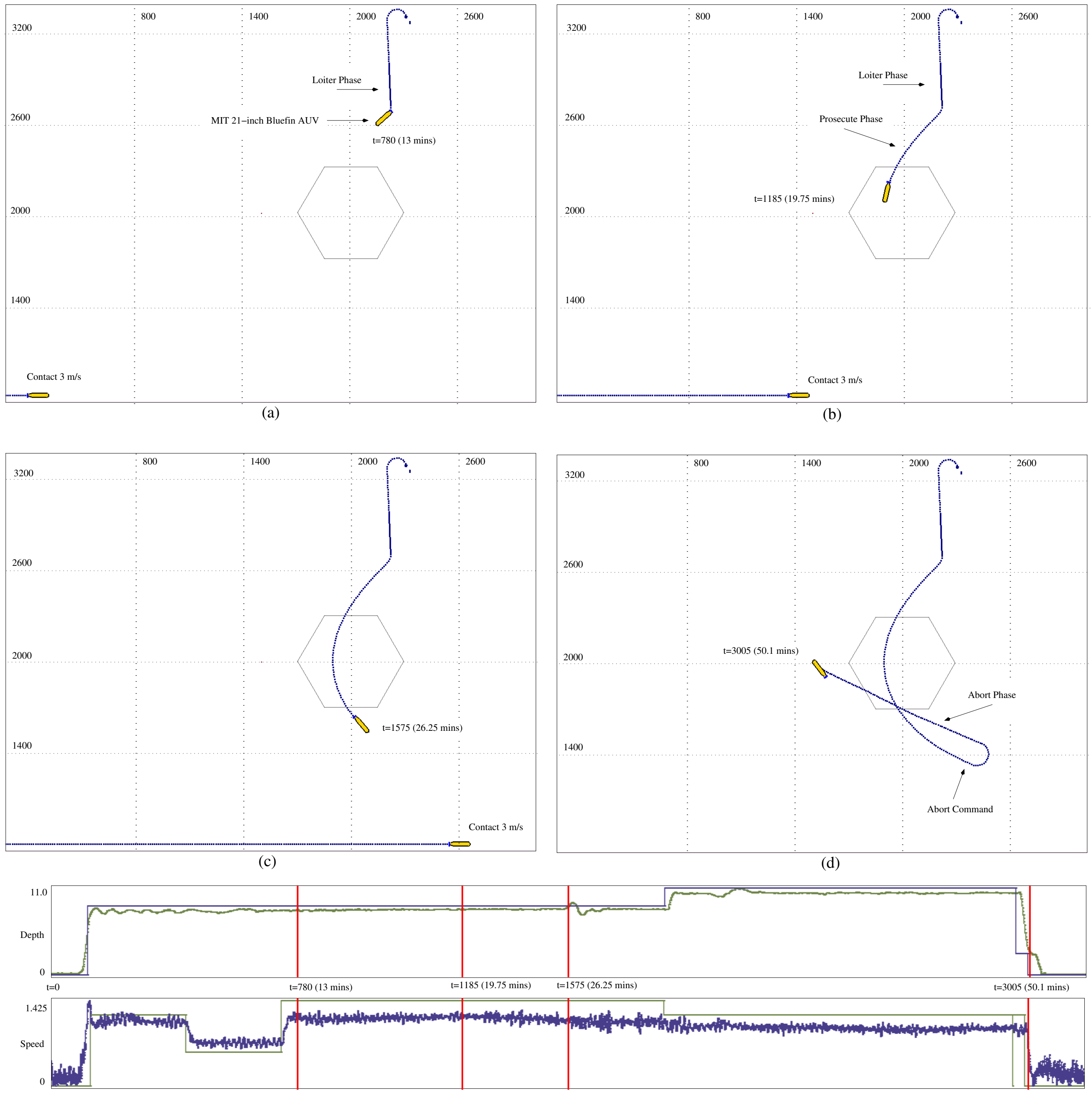


Fig. 11. Monterey Bay experiments with MIT 21-inch Bluefin AUV towing a vector sensor array. In (a) the vehicle has been deployed to a loiter region shown by the hexagon and has been commanded to traverse the region in a clockwise manner and thus proceeded to the eastern most vertex of the hexagon as an initial entry trajectory. During transit it was at 8 meters of depth and cycling between speeds of roughly 1.2 m/s and 0.7 m/sec as shown by the depth and speed plots below. Before it arrived on station at the loiter region, it received a new message from field control via acoustic link to prosecute a contact with the position and trajectory shown. In (b) the vehicle has been in the prosecute phase for roughly 5 minutes and has closed range considerably. The desired speed of the vehicle is higher in the prosecute phase to close range quickly. In (c) the vehicle has been in the prosecute phase for roughly 13 mins and the contact is now opening range to the vehicle. The message to abort the mission and return to a commanded abort point is imminent. The abort command will require the vehicle to perform nearly a 180 degree turn. In (d) the vehicle has arrived at the abort point and come to surface. The transit to the abort point was at 10 meters of commanded depth and the same commanded speed as during the loiter phase as the two time plots below indicate.

Brown University, Providence, RI, May 2002.

[13] M. Benjamin, J. Curcio, J. Leonard, and P. Newman, "Navigation of Unmanned Marine Vehicles in Accordance with the Rules of the Road," in *International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, May 2006.

[14] M. Benjamin, M. Grund, and P. Newman, "Multi-objective Optimization of Sensor Quality with Efficient Marine Vehicle Task Execution," in *International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, May 2006.