

The Backseat Control Architecture for Autonomous Robotic Vehicles: A Case Study with the Iver2 AUV

AUTHORS

Donald P. Eickstedt

Scott R. Sideleau

Naval Undersea Warfare Center,
Division Newport

I. Introduction

Recent advances in autonomous underwater vehicle (AUV) technology have led to their use in a number of military and civilian applications, including anti-submarine warfare, mine countermeasures, environmental sampling, and oil field surveys among others. This increase in interest and funding has predictably led to an increase in the number of commercial companies offering AUVs. The number of commercial AUVs of all size classes and the number and types of sensors available for these AUVs have never been greater. Each of these companies, for understandable competitive reasons, has developed its own proprietary vehicle control software. The major customers for many of these companies are oil field survey operations that typically have low interest in any sort of adaptive or cooperative vehicle maneuvering and therefore have low interest in providing these capabilities as basic vehicle options. This situation has led to a disconnect with researchers developing applications requiring adaptive AUV autonomy, who, for the most part, view underwater vehicles as commodity platforms designed to carry application-specific

ABSTRACT

In this paper, an innovative hybrid control architecture for real-time control of autonomous robotic vehicles is described as well as its implementation on a commercially available autonomous underwater vehicle (AUV). This architecture has two major components, a behavior-based intelligent autonomous controller and an interface to a classical dynamic controller that is responsible for real-time dynamic control of the vehicle given the decisions of the intelligent controller over the decision state space (e.g., vehicle course, speed, and depth). The driving force behind the development of this architecture was a desire to make autonomy software development for underwater vehicles independent from the dynamic control specifics of any given vehicle. The resulting software portability allows significant code reuse and frees autonomy software developers from being tied to a particular vehicle manufacturer's autonomy software and support as long as the vehicle supports the required interface between the intelligent controller and the dynamic controller. This paper will describe in detail the components of the backseat driver architecture as implemented on the Iver2 underwater vehicle, provide several examples of its use, and discuss the future direction of the architecture.

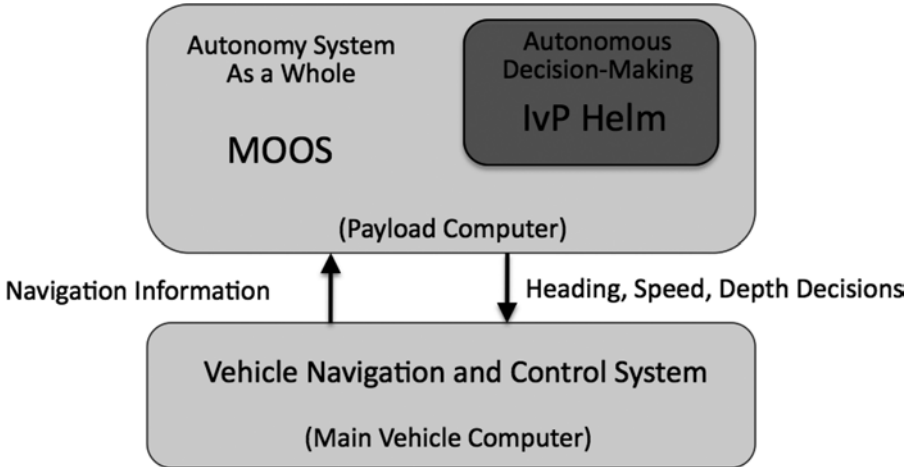
Keywords: Autonomous underwater vehicle (AUV), Marine robotics, Autonomy

sensor suites. For this group, code portability and rapid reconfiguration on different vehicles are major benefits. This disconnect can be resolved by viewing intelligent vehicle control and dynamic vehicle control as being separable components of an overall vehicle control architecture. For our purposes, the intelligent control component is responsible for determining the proper decision over the state space of desired vehicle course, speed, and depth. The dynamic control component is responsible for obtaining and maintaining the desired state by manipulation of the actuator surfaces and thrusters as well as for passing vehicle state information (e.g., position, heading, and speed) to the autonomy system. The general architecture just

described has been designated the backseat driver architecture (Benjamin et al., 2009b), where the intelligent controller can be seen as residing in the backseat and the dynamic controller in the frontseat. In this arrangement, the dynamic controller typically resides on the manufacturer's main vehicle computer (MVC) while the intelligent controller may reside in a totally separate processor on the vehicle, communicating with the MVC via a network or serial link using a standard interface. Although this architecture can be used with any intelligent controller implementation, Figure 1 shows it being implemented using the Mission Oriented Operating Suite (MOOS)-Interval Programming (IvP) autonomy system. Figure 2 shows a more detailed view

FIGURE 1

The backseat driver architecture: The key idea is the separation of vehicle autonomy from vehicle control. The autonomy system provides heading, speed, and depth commands to the vehicle control system. The vehicle control system executes the control and passes navigation information, e.g., position, heading, and speed, to the autonomy system. The backseat paradigm is agnostic regarding how the autonomy system is implemented, but in this figure, the MOOS-IvP autonomy architecture is depicted (Benjamin et al., 2009b).



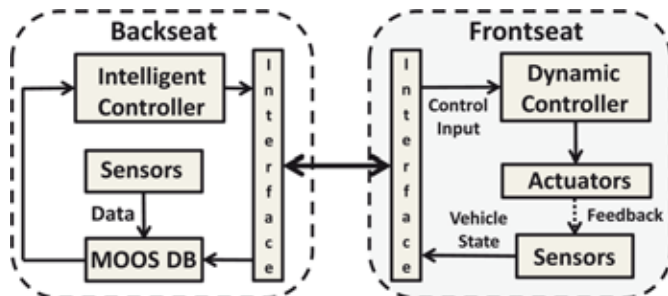
of the data flow between the frontseat and the backseat.

The backseat control architecture allows the intelligent control software to be rapidly portable across many different vehicles, with the only additional requirement being the need for an in-

terface module that implements the backseat/frontseat interface for a particular vehicle. Although this paper is focused on the implementation of the backseat driver architecture on the Iver2 AUV, it should be noted that it has also been successfully implemented

FIGURE 2

A more detailed view of the backseat architecture. Vehicle state data flow from the frontseat to the backseat over the interface where it is posted in the MOOSDB. Data from other sensors with connectivity to the backseat computer are also posted in the MOOSDB. Various processes and virtual sensors running under MOOS on the backseat use this data to compute information for use by the behaviors in the intelligent controller. The intelligent controller then sends the desired vehicle motion over the interface to the frontseat, thereby completing the control feedback loop.



on the Bluefin 21, Hydroid Remus, and FAU Ocean Explorer AUVs.

The remainder of this paper will describe the implementation of the backseat control architecture on the Iver2 AUV manufactured by Ocean Server Technology based in Fall River, Massachusetts. Section II will describe the major components of the implementation, including a detailed description of the Iver2 vehicle and the MOOS-IvP autonomy that supports the intelligent controller. Section III will describe the backseat control interface module that supports the interface between the intelligent controller and the MVC. Section V will show several vehicle runs made using the backseat control interface. Section IV describes the layered backseat/frontseat approach to vehicle safety. The paper will conclude by discussing the need for a standardized backseat interface.

II. System Architecture

In this section, we present the general system architecture for the Iver2 AUV equipped with the backseat driver architecture. This architecture consists of the Iver2 vehicle equipped with both an MVC supplied by the manufacturer as well as a separate payload computer that hosts the intelligent controller running under the MOOS autonomy middleware (Newman, 2009). In this implementation, the intelligent controller is 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 (Benjamin, 2002; Benjamin and Curcio, 2004). The MOOS module iOceanServerComms (described in detail in Section III) provides the backseat/frontseat interface via a serial link with the MVC.

A. The Iver2 AUV

The Iver2 AUV (see Figure 3) is a small man-portable AUV manufactured in Fall River Massachusetts by Ocean Server Technology, Inc. This vehicle was chosen by the Marine Autonomy Group (MAG) at the Naval Undersea Warfare Center in Newport, RI, based primarily on the ability to rapidly launch the vehicle from the shore or a variety of small or large craft as well as its ability to carry a variety of sensors relevant to our work in adaptive and collaborative autonomy. The vehicle's small size allows significantly more frequent experimental usage compared with larger vehicles needing ships with large capacity winches for launch and recovery. The MAG currently operates three Iver2 vehicles.

FIGURE 3

The Iver2 AUV equipped with the YSI 6600 V2 sonde, Woods Hole Oceanographic Institute acoustic MicroModem, and a 16-element towed hydrophone array. The payload computer stack consists of a PIII-800 CPU, and two 8-channel D/A boards. This model Iver2 navigates on the surface using GPS and underwater using a compass and speed table. This vehicle is designed for adaptive and collaborative autonomy experimentation applications, including anti-submarine warfare and rapid environmental assessment.



1) *General Characteristics.* The Iver2 vehicles in the MAG are approximately 65 inches in length and weigh approximately 50 pounds. The MAG vehicle is 5 inches longer than the standard vehicle and has been lengthened to provide space for future electronics additions, including a precision clock, inertial measurement unit, and Iridium satellite transceiver. The vehicle can operate at speeds of up to 4 knots using its rechargeable batteries, which have 600 W-h of capacity. Vehicle endurance is anywhere between 4 and 12 h depending on the vehicle's speed and hotel load (e.g., analog-to-digital [A/D] conversion for the acoustic array). The vehicle also comes equipped

with integrated depth (pressure) and altitude sensors.

2) *Navigation.* Each of the Iver2 vehicles in the MAG is equipped with a GPS receiver and three-axis digital compass providing roll, pitch, and yaw. One of the MAG vehicles was recently equipped with a Doppler Velocity Logger (DVL) to provide closed-loop speed control. The other two MAG vehicles have open-loop speed control using speed tables based on thruster revolutions per minute (RPM). Navigation solutions are computed on the MVC using dead reckoning and sent to the backseat over the backseat/frontseat interface. In the near future, all MAG vehicles will be equipped with inertial measurement units to provide closed-loop speed control in cases where the DVL is not equipped or cannot be used (e.g., in deep water).

3) *Communications.* Each of the Iver2 vehicles in the MAG is equipped with IEEE 802.11 wireless (WiFi) capabilities for communications while on the surface as well as a Woods Hole Oceanographic Institute acoustic MicroModem for communications while underwater. In the MAG configuration, the WiFi link is operated by the MVC, and the acoustic modem is operated by the autonomy system on the payload computer. The acoustic modem is used both for providing status messages to the topside as well as for reception of command and control messages from the topside. In the near future, each MAG Iver2 will be equipped with an Iridium satellite transceiver which will provide status messages while the vehicle is on the surface as well as provide for the reception of command and control instructions

when the vehicle is out of WiFi or acoustic communications range.

- 4) *Sensors.* In the current MAG configuration, each Iver2 vehicle is equipped with a YSI, Inc., 6600 V2-4 sensor bulkhead allowing the insertion of many different types of oceanographic sensor probes. The current MAG vehicles are equipped with conductivity-temperature, dissolved oxygen, and turbidity probes. Data from these probes are provided to the backseat autonomy system via the backseat interface in real time, allowing the vehicle to perform real-time adaptive environmental sampling. The DVL on the DVL-equipped Iver2 has an integrated downward-looking acoustic Doppler current profiler capable of profiling up to 30 depth bins, each of which can be up to 1 m in length. In addition to the oceanographic sensors, each vehicle is equipped with a 16-element acoustic line array cut for acoustic sampling at 1 kHz. Each array is equipped with inline preamplification. The array on each vehicle feeds acoustic data to a set of 8-channel A/D converter boards described in Section II-A.5, allowing the vehicle to perform real-time underwater target tracking and localization experiments.

- 5) *Payload Computer.* The payload computer stack in the MAG Iver2 consists of a CPU board running an Intel PIII-800 processor with 512 MB of main memory and a 100-GB hard drive. Communications with the MVC is via both a standard 10-Mbps Ethernet connection and a standard RS-232 serial link. The Ethernet link is primarily used for communication with the payload computer through the ve-

hicle's WiFi connection while the serial connection is used for the backseat control interface. In addition to the CPU board, the payload computer stack is equipped with two General Standards 24DSI-12 8-channel A/D converter boards providing 16 channels of simultaneous sigma-delta sampling for the acoustic line array.

- 6) *Safety Equipment.* The MAG Iver2 vehicles are all equipped with Sonotronics EMT-01-2 acoustic pingers to facilitate vehicle recovery in the event that the vehicle remains underwater beyond the mission execution time (e.g., vehicle becomes entangled or otherwise stuck).

Ocean Server Technology, Inc., also provides a safety tow float option (patent pending), which is a small towed device capable of deploying a programmable buoyancy airbag to bring the vehicle to the surface. The MAG Iver2 vehicles will be equipped with these safety tow floats in the near future when executing missions that do not require the use of the towed arrays.

B. The MOOS-IvP Autonomy Architecture

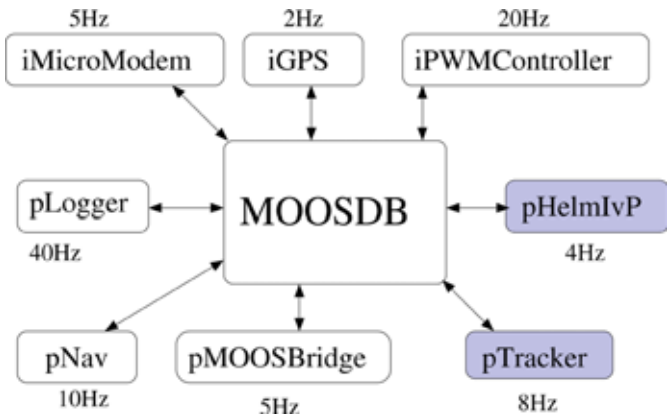
The autonomy system on the MAG Iver2 vehicles is implemented within the MOOS-IvP architecture for autonomous control, developed and maintained under a GNU general public license (GPL) by MIT, Oxford University, and NUWC. MOOS-IvP is composed of the Mission Oriented Operating Suite (MOOS), an open source software project for communication between, and nested control and coordination of, software processes running on the individual nodes of a network of autonomous platforms. MOOS-IvP is fully portable and platform independent, but typically implemented under GNU/Linux. MOOS-IvP also contains

the IvP Helm, a behavior-based helm that provides the core of the behavior-based control architecture. The IvP Helm runs as a single MOOS process and uses multi-objective optimization with the IvP model for behavior coordination (Benjamin, 2002). At any given time, multiple behaviors may be active and competing for influence on the vehicle motion. MOOS-IvP, as an open source project, is comprised in part of the core MOOS middleware capabilities and other essential applications, distributed by the Mobile Robotics Group at Oxford University. Additional MOOS modules, the IvP Helm, and a suite of basic IvP Helm behaviors are packaged on top of the core Oxford modules and distributed as MOOS-IvP from servers at MIT. More can be found at www.moos-ivp.org. A number of publicly available tools exist to extend the core autonomy capabilities described here. More information on these tools can be found in Benjamin et al. (2009a). Also see Benjamin et al. (2006a), Benjamin et al. (2006b), and Benjamin et al. (2007) for other examples of MOOS-IvP on autonomous marine vehicles.

A MOOS community contains processes that communicate through a database or bulletin board process called the MOOSDB, as shown in Figure 4. The MOOSDB process is the core of the MOOS architecture and handles all communication between the processes using a publish-and-subscribe architecture. Thus, all processes may publish variable name/value pairs to the MOOSDB. Once a variable has changed, all processes that have registered for subscription to the variable will receive a notification, following which they may request the new variable value if needed. The MOOS processes include all necessary control functions as well as sensing and processing

FIGURE 4

A MOOS community contains processes that communicate through a database or bulletin board process called the MOOSDB. The MOOSDB process is the core of the MOOS architecture and handles all communication between the processes using a publish-and-subscribe architecture. MOOS may be composed of processes for data logging (pLogger), data fusion (pNav), actuation (iPWM Controller), sensing (iGPS), communication (pMOOSBridge, iMicroModem), and much more. They can all be run at different frequencies as shown.



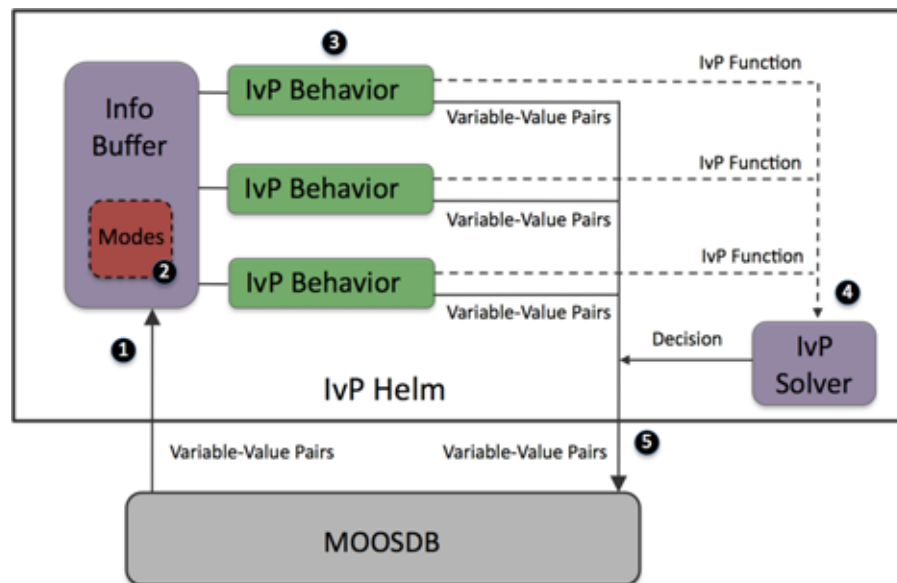
modules, with the MOOSDB providing the unified interface standard that enables the fully autonomous integration of sensing, modeling, processing, and control. MOOS ensures that a process executes its Iterate method at a specified frequency and handles new mail on each iteration in a publish-and-subscribe manner.

Each iteration of the helm contains the following steps: (1) Mail is read from the MOOSDB. It is parsed and stored in a local buffer to be made available to the behaviors. (2) If there were any mode declarations in the mission behavior, file they are evaluated at this step. (3) Each behavior is queried for its contribution and may produce an IvP function and a list of variable pairs to be posted to the MOOSDB at the end of the iteration. (4) The objective functions are resolved to produce an action, expressible as a set of variable-value pairs. (5) All variable-value pairs are published to the MOOSDB for other MOOS processes to consume. This is illustrated in Figure 5. Examples of adaptive behaviors are described in Eickstedt et al. (2006).

Adaptive behaviors for oceanographic sampling with the autonomous kayaks can be found in Eickstedt et al. (2007).

FIGURE 5

The pHelmIvP Iterate Loop: (1) Mail is read from the MOOSDB. It is parsed and stored in a local buffer to be made available to the behaviors. (2) If there were any mode declarations in the mission behavior file, they are evaluated at this step. (3) Each behavior is queried for its contribution and may produce an IvP function and a list of variable-pairs to be posted to the MOOSDB at the end of the iteration. (4) The objective functions are resolved to produce an action, expressible as a set of variable-value pairs. (5) All variable-value pairs are published to the MOOSDB for other MOOS processes to consume (Benjamin et al., 2009b).



The common theme between MOOS and the IvP Helm is the creation of a larger autonomy system from independent components, often from different developers and institutions. MOOS modules are independent applications communicating through a publish-subscribe protocol. IvP Helm behaviors are independent behaviors coordinated by the IvP Helm state space declarations and multi-objective optimization. This further de-coupling of autonomy components is motivated by the same reason as the de-coupling of autonomy from physical platforms as discussed above. It allows one to field an autonomous vehicle by procuring from the component level versus the system level. This allows contributions from a greater diversity of developers with complementary areas of expertise. No module is so critical that it cannot be replaced by a new version as one is

developed. This often has a positive effect in keeping the short- and long-term costs of a system under control. Furthermore, the independent relationship of modules allows a system of modules to be *nested*, i.e., constructed of both public and non-public modules side by side. The non-public modules may be proprietary, classified, or both. They are not visible to the independent developers of the MOOS-IvP public modules.

III. The Backseat Control Architecture

A. Overview

The iOceanServerComms MOOS module forms the interface between the main vehicle control (i.e., MVC or frontseat) computer and the vehicle autonomy (i.e., backseat) computer. Regular updates from the vehicle's sensors (e.g., compass, GPS, altimeter, oceanographic sensors) received across the interface, in addition to any data collected from sensors directly connected to the payload computer, are placed in the MOOS database. The IvP Helm reads the necessary sensor data, based on the running behaviors, from the MOOS database to make its multi-objective decision over the vehicle's state space of heading, speed, and depth for execution on the next control cycle. The desired state space decision is then transmitted by the iOceanServerComms module to the frontseat computer where the dynamic controller will attempt to achieve the desired state. Typically, this interface is run at a rate of 1 Hz, which is sufficient for control of a vehicle with a maximum speed of 4 knots. On the MAG Iver2 vehicles, the backseat/frontseat interface uses a standard RS-232 serial connection in a three-pin configuration, which causes the vehicle autonomy system to appear as

another simple sensor to the frontseat control computer. Other underwater vehicles, such as the Bluefin 21 AUVs, link their control and payload computer via standard IEEE 802.3 Ethernet. Regardless of the connection method employed by the manufacturer, the backseat interface, in this case iOceanServerComms, is responsible for the transmission and processing of messages encoded using the National Marine Electronics Association (NMEA) string format for sensor messages. The following subsections will describe the contents of each message in detail. Further information about the exposed remote helm controls supplied by Ocean Server Technology is available in DeArudra and Crowell (2009).

B. Data Request Message

The data request message (\$OSD) allows iOceanServerComms to request specific sensor messages from the frontseat computer. Since the MAG Iver2 vehicles are equipped with YSI 6600 oceanographic sensors (providing CTD [conductivity, temperature, depth], turbidity, and dissolved oxygen data), we can currently request updates of five different message types: compass, GPS, vehicle state, vehicle power (i.e., battery controller data), and YSI sensors. The data request message is easily expandable by the vehicle manufacturer to facilitate data transmission from other sensors that require connectivity to the vehicle control system. For example, recently an integrated Doppler Velocity Logger (DVL) by Sontek/YSI was added as an option on Iver2 vehicles; the ability to query the frontseat computer for DVL data was quickly added. An example of the NMEA encoded string to request data from the Iver2 frontseat across a serial connection is shown in Table I in the Appendix, where each of the

fields, if present as shown, indicate that data is requested from the frontseat.

The vehicle control system responds with the requested data messages, followed by an acknowledgement (\$ACK) that the command was received and processed successfully or that the request was received and not processed due to an error. In response to the acknowledgement, iOceanServerComms updates the MOOSDB variables FRONTSEAT_ACK and FRONTSEAT_ACK_ERROR, which can be used for communications monitoring by the end user or an intermediate MOOS application. If a given data message fails the checksum test when decoded by the iOceanServerComms module, it is then discarded. By default, iOceanServerComms requests and parses the four default NMEA messages (compass, GPS, vehicle state, power). The additional data messages can be enabled by toggling the appropriate entry (YSI, DVL, and CTD, respectively) in the iOceanServerComms MOOS configuration block. The data polling rate on the MAG vehicles is 1 Hz, but frequencies of up to 20 Hz have been successfully tested on the bench and in the water. The user must balance the density of data to postprocess and the size of the log files generated when considering the rate of data polling (i.e., iOceanServerComms MOOS AppTick).

The iOceanServerComms module parses the NMEA messages and updates the appropriate MOOS variables as described in Tables II through VIII shown in the Appendix.

C. Echoed Variables

To allow for intermediate manipulation by other MOOS processes and to facilitate the use of other sensors that may be more accurate on the reporting of particular data (e.g.,

heading), iOceanServerComms does not directly post to the MOOSDB certain critical navigation variables that the IvP Helm monitors. Instead, iOceanServerComms publishes data to unique MOOS variables and requires that the user employ the MOOS utility module pEchoVar to re-post data to the correct variables. This method allows, for example, data from an inertial measurement unit (IMU) to be posted to the NAV_HEADING variable instead of the COMPASS_HEADING from the frontseat computer. On the MAG Iver2 vehicles, pEchoVar currently re-posts the variables published by iOceanServerComms, as shown in Table II in the Appendix.

D. Helm Control Variables

The IvP Helm calculates the objective functions of the active helm behaviors and makes a multi-objective decision on the desired vehicle course, speed, and depth for the next control cycle. The helm then updates the variables in the MOOS DB shown in Table III in the Appendix.

When the IvP Helm has control (i.e., the MOOS variables DEPLOY = TRUE and MOOS_MANUAL_OVERRIDE = FALSE) and the iOceanServerComms command variable is set (i.e., VEHICLE_UNDERWAY = TRUE), iOceanServerComms reads the following variables from the MOOS database and transmits the NMEA-encoded \$OMS command message to the frontseat as shown in Table IV in the Appendix.

The maximum dive/surface angle, for safety reasons, is set to the Ocean Server recommendation of 30° by iOceanServerComms. The backseat timeout, or when the frontseat control system should take control of the vehicle after not receiving a new \$OMS message, is configurable via the Comm

Timeout parameter in the iOceanServerComms module MOOS configuration file.

The backseat continues to transmit \$OMS messages as long as the IvP Helm is engaged (MOOS variable IVPHELM_ENGAGED = ENGAGED). The iOceanServerComms module monitors the status of IVPHELM_ENGAGED and also notes the time the last update was received. If the configurable timeout is reached before an update to IVPHELM_ENGAGED is received, iOceanServerComms assumes the IvP Helm process has relinquished control or has stopped due to an unexpected error and blocks the further sending of servo control messages until another update to IVPHELM_ENGAGED is received. Control may also be released by iOceanServerComms when the IvP Helm reports it has completed executing all of its autonomy behaviors (IVPHELM_ENGAGED = DISENGAGED), for example, after a backseat mission timeout has occurred. When disengaged, the frontseat control system continues to execute the last received heading, depth, and speed until the frontseat timeout (specified in the \$OMS message itself) expires, after which the frontseat control system resumes control and finishes execution of its preprogrammed mission (e.g., to park the vehicle at a desired location).

E. Primitive Control Message

The Ocean Server \$OMP message allows the backseat autonomy system to directly control the fin and propulsion settings of the vehicle. Although arguably antithetical to the backseat driver control philosophy, this enables the development of a custom dynamic controller and navigation fusion MOOS module that, in turn, facilitates control

of a heavily modified Iver2 vehicle (i.e., one which no longer matches an existing frontseat control profile). The primitive control message (see Table XI in the Appendix) requires the population of the MOOS variables shown in Table XI to produce a valid \$OMP message. The \$OMP message follows the same timeout rules as the \$OMS message.

IV. Layered Backseat/ Frontseat Safety Coordination

A layered approach to vehicle safety rules is a critical part of the backseat control architecture. In this layered approach, safety rules can be activated on both the front seat and backseat payload computers, with the triggering of backseat safety rules resulting in the relinquishment of vehicle control to the frontseat. The safety rules that are run in the IvP Helm are considered the primary safety rules, with the rules run on the frontseat considered to be emergency backup in case of a backseat failure. Typically, the frontseat safety rules use hard vehicle safety limits designed to avoid vehicle loss while the backseat safety rules use soft limits designed to catch situations where the vehicle has maneuvered outside the expected operational envelope.

A. Backseat Safety

The configurable operating region behavior (BHV_OpRegion) is used to provide the IvP Helm with a three-dimensional zone of operation and maximum mission time. If the vehicle exceeds the operating region bounds (position, maximum depth, minimum altitude) or the maximum mission time, the IvP Helm commands the vehicle to a heading, depth, and speed of zero and then

disengages (IVPHELM_ENGAGED = DISENGAGED), causing the default front seat mission to resume. The default frontseat mission will typically park the vehicle at a predetermined location for pickup. In addition to the BHV_OpRegion, iOceanServerComms also provides monitoring of the IvP Helm heartbeat variable (see Section III), periodically updated by the IvP Helm to indicate it is still operational. If the predetermined timeout has expired, indicating that the IvP Helm is not responding, control is relinquished to the frontseat computer, where the default mission is executed.

B. Frontseat Safety

The Ocean Server Technology frontseat control software, Underwater Vehicle Controller (UVC), enables the operator to set a number of safety rules, including:

- Maximum depth from surface
- Minimum depth off bottom
- Maximum mission time
- Maximum dive angle over time
- Detection of no forward progress
- Detection of no upward progress when surfacing
- Detection of no downward progress when diving
- Leak detection

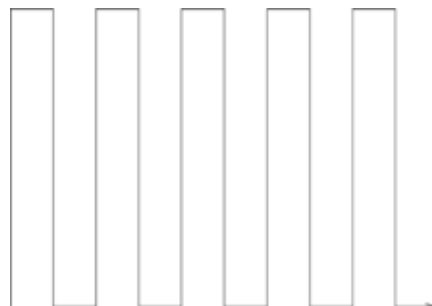
If an enabled safety rule is triggered, then the frontseat controller aborts its current mission, stops processing servo (\$OMS) and primitive (\$OMP) commands from the backseat driver, and executes its safety return path mission, which is a series of surface way points previously designed to park the vehicle at a safe location on the surface where it can then be retrieved. The frontseat safety rules always take precedence over backseat driver commands as the backseat autonomy is, again, only a “sensor” to the vehicle control computer.

V. Results

The iOceanServerComms backseat driver interface module running under MOOS-IvP has been successfully tested on the MAG Iver2 vehicle in a number of in-water applications. This section will illustrate several of those missions, including one adaptive mission and one non-adaptive mission. Non-adaptive missions follow a fixed path regardless of the data collected by the sensors. A typical example of a non-adaptive mission would be the commonly used ladder survey mission. Figure 6 shows an illustrative example of this type of pattern, which is used to sample a predefined area. The path traveled by the AUV is determined before the mission and is fixed, with the data being downloaded for off-line analysis. This type of survey is simple to set up and execute. A non-adaptive mission is unlikely to be optimal, however, with respect to either total mission time or data quality, with the data either being over-sampled or under-sampled. An adaptive variation of this

FIGURE 6

A plot of a typical non-adaptive AUV mission. In this case the desired vehicle path is a standard horizontal ladder survey pattern with legs of fixed width. The path of the AUV in this type of non-adaptive mission is determined before the mission begins and does not change during the mission. An adaptive innovation to a ladder survey would be to sample the data to determine the optimum horizontal spacing of the survey legs to satisfy both mission time and data quality metrics.



mission would adjust the width of the legs on the ladder pattern based on the sampled data itself to ensure that mission time is optimized, with the data being sufficiently sampled. Adaptive maneuvering can occur during a mission in the horizontal plane or the vertical plane or simultaneously in both. An adaptive mission would also allow real-time mission changes upon the detection of unique features such as upwellings or fronts. The backseat driver architecture supports both modes of operation transparently.

Figure 7 shows a non-adaptive mission executed in March 2010, off the coast of Key Largo, Florida, using the iOceanServerComms interface. The purpose of this mission was to test an algorithm for determining the optimum sampling points to sample a given area given a constraint on total mission time. No *a priori* information is known about the parameters of the survey area except for its extent. The sampling points are chosen using a simulated annealing algorithm. In this mission, a MOOS process named pSurvey runs the optimization algorithm over the desired survey area upon startup. pSurvey computes each of the sampling points, which are then used as input to a simple Waypoint IvP behavior that outputs the desired course for the vehicle to travel to each of the sampling points in turn. In this mission, a total mission time of 2 h was used, and the vehicle was run at a constant speed and depth. As can be seen in the plot, the Iver2 vehicle successfully navigated each of the sampling points.

Figure 8 shows a plot of a more complicated adaptive sampling mission, with adaptation occurring in the vertical (depth) plane. The mission goal for the vehicle is to run horizontally in a fixed direction while changing its depth

FIGURE 7

A plot of an Iver2 space filling mission off the coast of Key Largo, Florida, in March 2010. In this mission, the vehicle computes the sampling points (red circles) that optimally sample the given area (black rectangle) using a simulated annealing algorithm with a simulated sampling time of 1 s at each point and a total mission time of 2 h. After computing the points, the vehicle successfully visits each in turn.

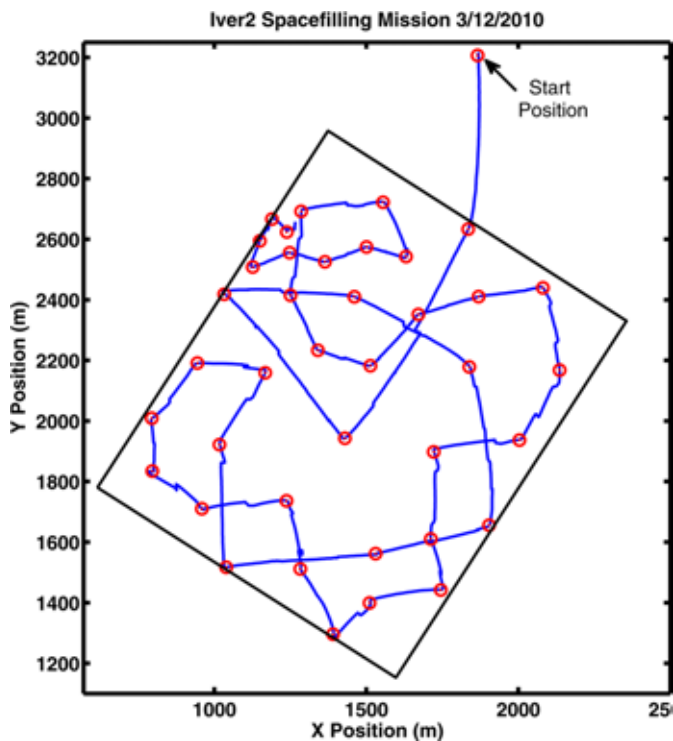
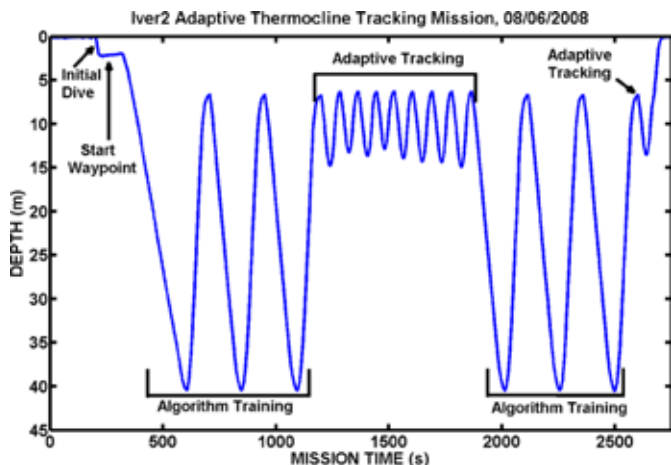


FIGURE 8

A plot of the Iver2 depth during an adaptive thermocline tracking mission in the Mediterranean off of Pianosa Island in August 2008. As shown in the plot, the vehicle first samples the entire water column to gather statistics on the sound speed profile and then adaptively changes the yo-yo depth to focus sampling on the thermocline, which was at approximately 10 m depth. This is an example of adaptive maneuvering in the vertical plane.



adaptively to try and capture the thermocline. Thermocline data can be valuable for oceanographic models attempting to provide forecasts of ocean parameters such as sound speed. In this implementation, a MOOS process named pSSGrad estimates the sound speed gradient using real-time sound speed data from the YSI sonde on the Iver2. A MOOS process named pThermoTrack then uses the sound speed gradient information to determine the top and bottom depths for a yo-yo behavior, with the goal being to sample the thermocline without resorting to doing a yo-yo over the entire depth range. This mission was run in 80 m of water in the Mediterranean Sea off of Pianosa Island, Italy, in August 2008. As shown in the plot, the vehicle first samples the entire water column to gather statistics on the sound speed profile and then adaptively changes the yo-yo depth to focus sampling on the thermocline, which was at approximately 10 m depth.

VI. Conclusion

In this paper we have described the backseat driver architecture for robotic vehicles in detail and its implementation on a commercially available AUV. Although this architecture was developed for marine vehicles, it is also suitable for operating other types of robotic vehicles such as land or aerial robots. The driving force behind the development of this architecture was a desire to free autonomy software developers from dependence on any particular AUV and therefore any particular manufacturer's autonomy software, leaving the software developers free to reuse their code on whichever particular AUV is suitable for the task at hand. In fact, we estimate that over 90% of current MOOS-IvP modules are reused on

the Iver2, Bluefin 21, FAU Ocean Explorer, Hydroid Remus, and other vehicles.

An expanded discussion of code reuse can be found in Benjamin et al. (2009b) and includes the following:

- *Diversity of Contributors.* Increasingly, an autonomy system contains many components that touch many areas of expertise. This would be true even for simple applications, but is compounded when considering the variety of sensors and missions. A system that allows wide code reuse is also a system that allows module contributions from a wide set of developers or experts. This has a substantial impact on the issues mentioned below of lower cost, higher quality and reliability, and reduced development time.
- *Lower Cost.* One immediate benefit of code reuse is the avoidance of repeatedly reinventing modules. A group can build capabilities incrementally, and experts are free to concentrate on their area and develop only the modules that reflect their skill set and interests. Perhaps more important, code reuse gives the systems integrator *choices* in building a complete system from individual modules. Having choices leads to increased leverage in bargaining for favorable licensing terms or even non-proprietary terms for a new module. Favorable licensing terms arranged at the outset can lead to substantially lower long-term costs for future code maintenance or augmentation of software.
- *Higher Performance Capability.* Code reuse enhances performance capability in two ways. First, since experts are free to be experts without reinventing the modules outside their expertise and provided by others, their own work is more

likely to be more focused and efficient. They are likely to achieve a higher capability for a given finite investment and given finite performance time. Second, since code reuse gives a systems integrator *choices*, this creates a meritocracy based on optimal performance-cost ratio of candidate software modules.

- *Higher Performance Reliability.* An important part of system reliability is testing. The more testing time and the greater diversity of testing scenarios, the better. And of course the more time spent testing on physical vehicles versus simulation, the better. By making core components of a code base public and permitting reuse by a community of users, that community provides back an enormous service by simply using the software and complaining when or if something goes wrong. Certain core components of the MOOS-IvP code base have had hundreds if not thousands of hours of usage on a dozen or so platform types in a variety of situations.
- *Reduced Development Time Line.* Code reuse means less code is being redeveloped leading to quicker overall system development. More subtly, since code reuse can provide a systems integrator choices and competition on individual modules, development time can be reduced as a consequent. An integrator may simply accept the module developed the quickest, or the competition itself may speed up development. If choices and competition result in more favorable license agreements between the integrator and developer, this in itself may streamline agreements for code maintenance and augmentation in the long term.

One issue that needs to be addressed, and is being addressed at NUWC, is the development of a standard for the backseat driver interface including message formats and interface handshake protocols. Even though all the vehicles mentioned have a backseat driver interface, each manufacturer has implemented its own message format and interface control protocols. An interface standard would allow rapid development of interface modules for new vehicles as well as increase code portability and reuse of the autonomy modules.

Acknowledgments

The authors would like to acknowledge the staff at Ocean Server Technology, Inc., for their assistance in the development and testing of the backseat driver interface for the Iver2 vehicle as well as the NATO Undersea Research Center and the crew of the *R/V Leonardo* for hosting the Iver2 team during the GLINT 2008 exercise.

Lead Authors:

Donald P. Eickstedt
Scott R. Sideleau
Naval Undersea Warfare Center,
Division Newport, Newport, RI
Email: donald.eickstedt@navy.mil;
scott.sideleau@navy.mil

Appendix

TABLE I

Data request message.

| \$OSD,C,G,S,P,Y,D,T*cc | |
|-------------------------------|--|
| \$OSD | Message header |
| C | Compass request |
| G | GPS request |
| S | State (of vehicle) request |
| P | Power (or battery controller) request |
| Y | YSI sensor request |
| D | Doppler Velocity Logger request |
| T | Conductivity, temperature, depth request |
| cc | Standard NMEA checksum (XOR of message contents) |

TABLE II

Echoed variables.

| Published by iOceanServerComms | Published by pEchoVar |
|--------------------------------|-----------------------|
| COMPASS_HEADING | NAV_HEADING |
| COMPASS_DEPTH | NAV_DEPTH |
| COMPASS_PITCH | NAV_PITCH |
| COMPASS_ROLL | NAV_ROLL |
| COMPASS_YAW | NAV_YAW |
| FRONTSEAT_X | NAV_X |
| FRONTSEAT_Y | NAV_Y |
| FRONTSEAT_LAT | NAV_LAT |
| FRONTSEAT_LONG | NAV_LONG |
| FRONTSEAT_SPEED | NAV_SPEED |
| FRONTSEAT_ALTITUDE | NAV_ALTITUDE |

TABLE III

Helm control variables.

| Published by pHelmVp | Units |
|----------------------|-------------------|
| DESIRED_HEADING | Degrees true |
| DESIRED_SPEED | Meters per second |
| DESIRED_DEPTH | Meters |

TABLE IV

Backseat control message.

| \$OMS,H,D,A,S,T*CC | |
|---------------------------|--|
| \$OMS | Message header |
| H | Desired heading |
| D | Desired depth (feet) |
| A | Maximum pitch angle |
| S | Desired speed (knots) |
| T | Desired timeout (s) |
| *CC | Standard NMEA checksum (XOR of message contents) |

TABLE V

GPS message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|-------------------------------|
| GPS_TIME | UTC time of day |
| GPS_WARNING | Satellite fix good? |
| GPS_SPEED | Estimated vehicle speed (m/s) |
| GPS_HEADING | True heading (degrees) |
| GPS_DATE | UTC date |
| GPS_MAGNETICVARIATION | Magnetic variation (degrees) |
| GPS_LATITUDE | Latitude (decimal degrees) |
| GPS_LONGITUDE | Longitude (decimal degrees) |
| GPS_UPDATE_RECEIVED | Time of last valid fix |

TABLE VI

State message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|-----------------------------------|
| FIN_T_YAW | Top fin's yaw (servo setting) |
| FIN_B_YAW | Bottom fin's yaw (servo setting) |
| FIN_L_PITCH | Left fin's pitch (servo setting) |
| FIN_R_PITCH | Right fin's pitch (servo setting) |
| FIN_MOTOR_SPEED | Motor thrust (servo setting) |
| IVER_FRONTSEAT_WP | Frontseat mission's waypoint |
| FRONTSEAT_LAT | GPS or dead-reckoned latitude |
| FRONTSEAT_LONG | GPS or dead-reckoned longitude |
| FRONTSEAT_SPEED | Vehicle speed (knots) |
| FRONTSEAT_X | Vehicle X MOOS coordinate |
| FRONTSEAT_Y | Vehicle Y MOOS coordinate |

TABLE VII

Power message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|--------------------------|
| BATTERY_PERCENT | Percent charge remaining |
| BATTERY_WATTHRS | Watt-hours remaining |
| BATTERY_WATTS | Wattage draw |
| BATTERY_VOLTS | Voltage level |
| BATTERY_AMPS | Current draw |
| BATTERY_TIME | Time to empty/full |
| BATTERY_STATE | Charging or discharging? |
| BATTERY_LEAK | Leak detection Boolean |

TABLE VIII

YSI message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|-------------------------------|
| YSI_DATE2 | Date of YSI controller |
| YSI_TIME | Time of YSI controller |
| YSI_TEMP | Temperature (degrees C) |
| YSI_SPCOND | Specific conductivity (mS/cm) |
| YSI_SALINITY | Salinity (ppt) |
| YSI_DEPTH | Depth (m) |
| YSI_TURBIDITY | Turbidity (NTU) |
| YSI_ODO% | Percent dissolved oxygen |
| YSI_ODO | Dissolved oxygen (mg/L) |
| YSI_BATTERY | Voltage of YSI controller |
| CTD_SOUND_VELOCITY | Calculated sound velocity |

TABLE IX

DVL message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|---------------------------------|
| DVL_SPEED_X | Speed in the X direction (m/s) |
| DVL_SPEED_Y | Speed in the Y direction (m/s) |
| DVL_SPEED_Z | Speed in the Z direction (m/s) |
| DVL_DISTANCE_X | Distance in the X direction (m) |
| DVL_DISTANCE_Y | Distance in the Y direction (m) |
| DVL_DEPTH | Depth from surface (m) |
| DVL_ALTITUDE | Altitude off the bottom (m) |

TABLE X

CTD message.

| Published by iOceanServerComms | Purpose |
|--------------------------------|----------------------------------|
| CTD_CONDUCTIVITY | Specific conductivity (mmhos/cm) |
| CTD_TEMPERATURE | Temperature of water (degrees C) |
| CTD_SALINITY | Salinity of water (ppt) |
| CTD_SOUND_SPEED | Sound speed in water (m/s) |

TABLE XI

Primitive control message.

| Subscribed to by iOceanServerComms | Units |
|------------------------------------|-----------------------------|
| DESIRED_RUDDER | Servo control value (0–255) |
| DESIRED_ELEVATOR | Servo control value (0–255) |
| DESIRED_THRUST | Servo control value (0–255) |

References

- Benjamin**, M. 2002. Interval programming: a multi-objective optimization model for autonomous vehicle control. Ph.D. thesis, Brown University. Providence, RI.
- Benjamin**, M., Curcio, J., Leonard, J., Newman, P. 2006a. Navigation of unmanned marine vehicles in accordance with the rules of the road. In: Proc. IEEE Int. Conf. on Robotics and Automation, Orlando, Florida. pp. 3581-3587.
- Benjamin**, M., Grund, M., Newman, P. 2006b. Multi-objective optimization of sensor quality with efficient marine vehicle task execution. In: Proc. IEEE Int. Conf. on Robotics and Automation, Orlando, Florida. pp. 3266-3232.
- Benjamin**, M.R., Battle, D., Eickstedt, D.P., Schmidt, H., Balasuriya, A. 2007. Autonomous control of an autonomous underwater vehicle towing a vector sensor array. In: Proc. IEEE Int. Conf. on Robotics and Automation, Rome, Italy. pp. 4562-4569.
- Benjamin**, M.R., Curcio, J. 2004. COLREGS-based navigation of unmanned marine vehicles. In: Proc. IEEE AUV2004 Conf., Sebasco Harbor, Maine. pp. 32-39.
- Benjamin**, M.R., Newman, P.M., Schmidt, H., Leonard, J.J. 2009a. Extending a MOOS-IvP autonomy system and user's guide to the IvP build toolbox. Technical Report TR-2009-037. MIT Computer Science and Artificial Intelligence Laboratory.
- Benjamin**, M.R., Newman, P.M., Schmidt, H., Leonard, J.J. 2009b. An overview of MOOS-IvP and a brief user's guide to the IvP helm autonomy software. Technical Report TR-2009-028. MIT Computer Science and Artificial Intelligence Laboratory.
- DeArudda**, J., Crowell, J. 2009. Remote helm guide. Technical Report. Ocean Server Technology, Inc.
- Eickstedt**, D.P., Benjamin, M.R., Schmidt, H., Leonard, J.J. 2006. Adaptive control of heterogeneous marine sensor platforms in an autonomous sensor network. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Beijing, China. pp. 5514-5521.
- Eickstedt**, D.P., Benjamin, M.R., Wang, D., Curcio, J., Schmidt, H. 2007. Behavior based adaptive control for autonomous oceanographic sampling. In: Proc. IEEE Int. Conf. on Robotics and Automation, Rome, Italy. pp. 4245-4250.
- Newman**, P. 2009. Under the hood of the MOOS communications API. Technical Report. Oxford University.