Control Architectures for Autonomous Underwater Vehicles

Kimon P. Valavanis, Denis Gracanin, Maja Matijasevic,
Ramesh Kolluru, and Georgios A. Demetriou

Autonomous Underwater Vehicles (AUVs) share common control problems with other air, land, and water unmanned vehicles. In addition to requiring high-dimensional and computationally intensive sensory data for real-time mission execution, power and communication limitations in an underwater environment make it more difficult to develop a control architecture for an AUV. In this article, the four types of control architectures being used for AUVs (hierarchical, heterarchical, subsumption, and hybrid architecture) are reviewed. A summary of 25 existing AUVs and a review of 11 AUV control architecture systems present a flavor of the state of the art in AUV technology. A new sensor-based embedded AUV control system architecture is also described and its implementation is discussed.

Introduction

Research in Autonomous Underwater Vehicles (AUVs) is a part of the ongoing research efforts in the area of air, land, and water unmanned vehicles. Unmanned vehicles (remotely operated or autonomous) eliminate the need for human physical presence and, therefore, reduce human exposure in hazardous environments. Remotely operated unmanned vehicles use telerobotics and telepresence for navigation and control. In autonomous unmanned vehicles there is no human operator; thus, they function based on built-in machine intelligence and an on-board control system. The design of the control system and the underlying control system architecture is the major problem in the development of autonomous unmanned vehicles, due to high-dimensional sensory data, computation-intensive processing, and real-time execution constraints. The problem is even more complex for underwater autonomous unmanned vehicles due to power and communication limitations. An overview of air, land, and water unmanned vehicles clarifies the relationship between them.

General Overview of Unmanned Vehicles

Unmanned Vehicles (UVs) refer to and include unmanned aerial vehicles, unmanned ground vehicles, and unmanned underwater vehicles. Detailed information, UV historical background, technology, and updated listings may be found in [1].

Unmanned Aerial Vehicles (UAVs) emerged during the World War II and have been mainly intended for military usage. Lately, there is an increased interest in advanced UAVs, as evidenced by four very recent major projects: The Pioneer UAV, used by the U.S. Navy in Operation Desert Storm and still in use by the U.S. Marine Corps; the Predator UAV, operational in Bosnia; the Teledyne Ryan Tier II+ Global Hawk, a high-altitude, long-endurance, all weather aerial reconnaissance UAV currently performing test flights; and the Outrider (from Alliant TechSystems) tactical UAV currently under development, which is expected to be used by the U.S. Army, Navy, and Marine Corps Air Ground Task Force (see [1], pp. 24-81). Further on, the Swiss military authorities have developed the ADS 95 Ranger UAV System to support the Swiss Armed forces in intelligence data gathering and artillery operations [2].

Unmanned Ground Vehicles (UGVs), including teleoperated and autonomous vehicles (mobile robots), have been used in military and civil applications. UGV research and development has been dominated by DARPA (Defense Advanced Research Projects Agency) and NASA (National Aeronautics and Space Administration). The DARPA initiative started with the development of the first mobile robot, Shakey, and also includes the Autonomous Land Vehicle and the DARPA Demo II Program. NASA sponsors the development of unmanned vehicles for planetary surface exploration, from the Jet Propulsion Laboratory Mars Rover to the most recent Mars Pathfinder. Recent UGV design and development has been enhanced to build UGVs capable of operating in Intelligent Vehicle Highway Systems (see [1], pp. 83-117).

Unmanned Underwater Vehicles (UUVs) include remotely operated vehicles and autonomous underwater vehicles. UUVs in general may be used for inspection, drilling, mine countermeasures, survey, observation, underwater cable burial, and inspection of power plant conduits [3].

Remotely Operated Vehicles (ROVs) are tethered[1] to a support (surface) ship by an umbilical cable that relays control signals and power down to the vehicle and returns images and other sensor data to the support ship. ROVs are particularly valuable when the location of the undersea destination is uncertain, or when the ocean conditions endanger a manned mission. ROVs are continuously controlled and navigated by a human operator. This requires a high-bandwidth, low-latency communication link with the remote vehicle [4]. Also, the ROV umbilical cable constrains the vehicle to operations in close proximity to the support ship. However, as stated in [5], as the range of operation becomes longer and water deeper, the drag exerted by the tether

This work has been partially supported by NSF Grants BES-9506771 and BES-9712565. *The authors are with the Robotics and Automation Laboratory, Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA, USA (kimon@cacs.usl.edu, {dg, maja, rxk, gad}@acim.usl.edu).*

[1]Some near-surface ROVs, e.g. ORCA, have an on-board diesel engine and use radio-link for controls.

 0272-1708/97/$10.00©1997IEEE

Table 1. Selected AUV Configurations (Part 1)

Control architecture/ Implementation	Power source	Propulsion	Shape/ Max. speed/ Max. depth	On-board equipment	Application
1. ABE, Woods Hole Oceanographic Institution, USA (1992)					
2-level hierarchical (distributed)/ MC68HC11, T800 transputers, SAIL network	lead-acid gel, alkaline cells, lithium cells	3 stern, 2 vertical, 2 horizontal thrusters	3-body open frame structure/ 2 knots/ 6000 m	fluxgate compass, magnetic heading, angular rate sensor, CCD cameras, broadband 300 kHz transponders, low frequency long-baseline transponders	long-term seafloor survey
2. Aqua Explorer 1000, Tokai University and Kokusai Denshin Denwa Co. Ltd., Japan (1992)					
NA/ MC68040/4MB, VxWorks, 3 DSPs + image processor, VME bus	lead-acid	2 horizontal, 1 vertical brushless DC thrusters	flat-fish/ 2 knots/ 1000 m	cameras, obstacle avoidance sonar, VCR recorder, laser, rate gyroscope, altimeter, depthometer, accelerometers, acoustic transponder, acoustic link	seaflor and telecommunication cables inspection
3. ARCS, International Submarine Engineering Research and Bedford Institute of Oceanography, Canada (1984)					
hybrid (hierarchical + modified subsumption)/ MC68030	Ni-Cd battery, Al fuel cells	brushless DC thruster	torpedo/ 5 knots/ 400 m	inertial navigation unit, doppler sonar	under-ice mapping
4. Aurora, International Submarine Engineering Research and Simon Fraser University, Canada, (1994)					
hybrid (hierarchical + modified subsumption)/ dual PC-104 microprocessors	lithium battery	brushless DC thruster	streamline oval/ 3.5 knots/ 3000 m	ring laser gyro, doppler sonar, ultrashort baseline system, CTD sensors, side-scan sonar, obstacle avoidance sonar, TV camera, telemetry link	under-ice mapping
5. UUV/AUVC, Texas A&M University, USA (1988)					
3-level hierarchical controller, knowledge based system for learning and recovery/ 16 Sun SPARC processors	lead-acid and 2 diesel generators	NA	torpedo/ 10 knots/ NA	active sonar, collision avoidance sonar, GPS navigation, radar warning receiver	testbed
6. Dolphin, International Submarine Engineering Research, Canada (1983)					
hybrid (hierarchical + modified subsumption)/ MC68030	Ni-Cd, Ag-Zn batteries, diesel	turbo-diesel engine	torpedo/ 15 knots/ 3.6 m	DGPS, single beam side-scan sonar, multi-beam echosounder, forward looking sonar, multi-beam echo sounder, cameras	mine-hunting and hydrography
7. EAVE III,University of New Hampshire and Autonomous Undersea Systems Institute, USA (1987)					
4-level hierarchical/ MC68020/4MB based on VME bus, MC68000 based MSEL-CPU, pSOS, and VxWorks	lead-acid	6 brushless DC thrusters	2 buoyancy tubes for computers, 2 battery tubes/ 1.5 knots/ 200 m	obstacle avoidance sonar, acoustic depth sensor, altitude sonar, pressure depth sensor, water temperature sensor, fluxgate compass, acoustic long baseline and short baseline navigation, acoustic telemetry, RF telemetry	data acquisition, surveillance, multiple AUV communication and cooperation
8. Eric, University of Technology Sydney, Australia (1994)					
3-layer subsumption/ MC68332	NA	4 thrusters	3-spheres and frame/ NA/ NA	optical sensors, acoustic sensors	testbed
9. LDUUV, Naval Undersea Warfare Center, USA (1994)					
hierarchical/ 6 Sun SPARC processors (scaled down from TAMU AUVC)	Ag-Zn	NA	torpedo/ 12 knots/ 300 m	3 axis accelerometers, shear probes thermistor, CTD sensors, side-scan sonar, autonomous way point navigation	military, testbed

becomes significant. The thrusters, and thus the vehicle itself, must become larger and the cable thicker, and the energy that goes into cable maintenance becomes a major factor. The removal of the tether to the support ship frees the vehicle from a substantial drag, minimizes entanglement problems, and removes constraints in depth and maneuverability. In [6], there is also an explicit list of long-, medium-, and light-work ROVs, inspection and observation ROVs, low-cost ROVs, custom-built ROVs, and military ROVs.

Untethered underwater vehicles may be divided into two groups based on their maneuvering characteristics [5]: cruising vehicles, used for surveys, search, object delivery, and object location, which run continuously during their mission; and hovering vehicles, used for detailed inspection and physical work on and around fixed objects, which hold stations in the water column to perform their tasks. In terms of control, cruising vehicles typically require only three degrees of control—longitudinal, yaw, and pitch—while hovering vehicles require that dynamic thrust be generated to produce forces in three orthogonal directions and moments in yaw and sometimes pitch [5]. For all ROVs, tethered as well as untethered, problems of endurance and communication become critical, and ample intelligence must be provided within the vehicle to achieve the mission goal and accomplish useful work.

Autonomous Underwater Vehicles

An Autonomous Underwater Vehicle (AUV) is an unmanned untethered underwater vehicle that carries its own power source and relies on an on-board computer and built-in machine intelligence to execute a mission consisting of a series of preprogrammed instructions (potentially) modifiable on-line by data or information gathered by the vehicle sensors [1].

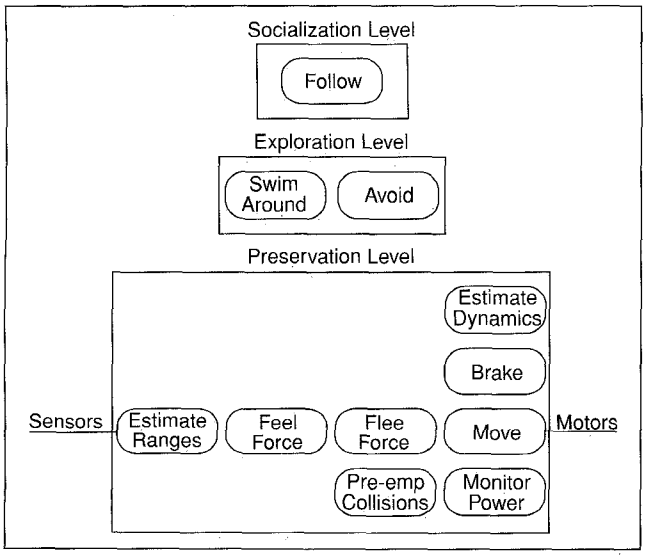


Fig. 3. Eric control architecture.

Examples of this architecture include the Ocean Voyager II [14] (hierarchical-subsumption), the Omni-Directional Intelligent Navigator (ODIN) [3] (hierarchical-heterarchical), and the Phoenix [20] (hierarchical-subsumption).

Review of Selected AUVs and Control Architectures

This section summarizes the control system architectures of 11 AUVs and their operational and functional principles. The summary provides an overview of the state-of-the-art available technology, as well as a suitable introduction for a sensor-based embedded control architecture proposed later in this article. It should be noted, though, that this selection of vehicles only reflects the authors' preference, as opposed to an exhaustive review of all AUVs developed to date. This section is complemented with Table 1, which presents the configuration of 25 operational AUVs.

Autonomous Benthic Explorer (ABE)

The ABE has been developed at the Woods Hole Oceanographic Institution [10]. It utilizes a distributed, hierarchical control architecture, based on two different layers with different computational capabilities, as shown in Fig. 1. The top layer has low computation capability and very low power requirements, while the lower layer has large and expandable computational power and higher power requirements. The system is distributed, with nodes communicating serially. Each sensor and actuator is associated with a node which contains its own single chip microcomputer. System design is modular, and modules are divided into levels according to computational requirements. ABE has been designed primarily for deep-water, short-range, long-duration missions.

Autonomous Underwater Vehicle Controller (AUVC)

The AUVC has been developed at the Texas A&M University as a control system for the unmanned underwater vehicle (Naval Surface Warfare Center). Later, it has been used for the large diameter unmanned underwater vehicle (Naval Undersea Warfare Center) [21]. The AUVC for the most part follows the hierarchical architecture modeling approach, as represented in Fig. 2. The AUVC has a very complicated software system, consisting of 18 software modules. The software system is composed of a Mission Management component, a Diagnosis component, a Control Systems component, and a Fault Tolerant Computing Environment component. The AUVC is capable of mission planning/replanning, path planning, energy management, collision avoidance, threat detection and evasion, failure diagnosis and recovery, radio communication, and navigation.

Eric

The Eric AUV has been developed by the Key Center Robotics Laboratory at the University of Technology, Sydney [17]. Eric follows the general style of subsumption architecture, with enhancements, as shown in Fig. 3. Three different levels of competence describe the capabilities of the system: the preservation level, the exploration level, and the socialization level. The preservation level performs obstacle avoidance, the exploration level performs high-level navigation, and the socialization level enables object following behavior. Object following is accomplished as a summation of forces. Attraction forces turn the AUV toward an object, and repulsive forces keep the AUV from colliding with the object. Eric's architecture overcomes the flaws associated with pure subsumption architecture through the use of formalization, which includes methods of guaranteeing robustness, identifying ambiguities by the use of naming rules, the standardization of symbols, etc., resulting in a reliable and robust architecture.

Experimental Autonomous Vehicle (EAVE) III

The EAVE III was developed at the Marine Systems Engineering Laboratory of the University of New Hampshire and Autonomous Undersea Systems Institute [11]. The vehicle uses a time-ordered, modular, hierarchical control architecture, as shown in Fig. 4. The architecture is divided into four different levels: Real-time, system, environment, and mission. The lowest, real-time level executes sensor management, data processing, computation of error signals, and computation of effector setting. The system level controls and maintains the vehicle and

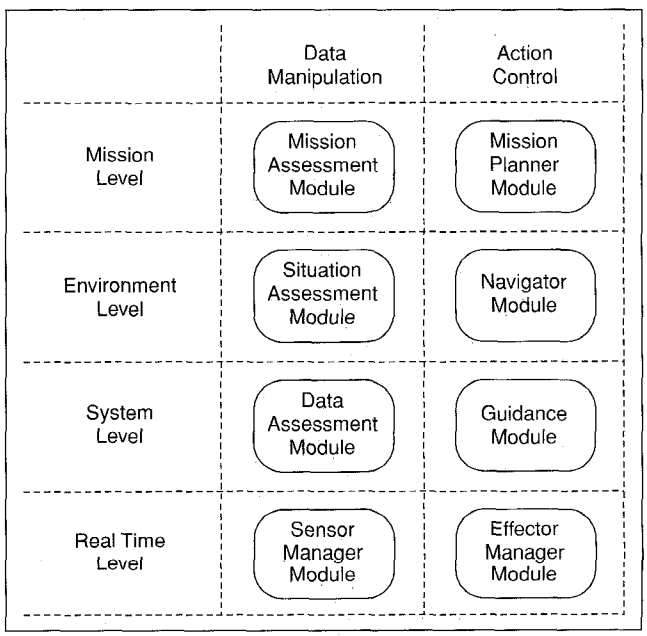


Fig. 4. Experimental Autonomous Vehicle III control architecture.

determines how accurately the desired path has been followed. The environment level deals with the vehicle's physical environment. The mission level is a high-level planner that handles aspects particular to the mission and the tasks to be accomplished. The control system design allows additional sensors and software to augment the system. The EAVE open space-frame design makes it an extremely flexible and adaptable testbed.

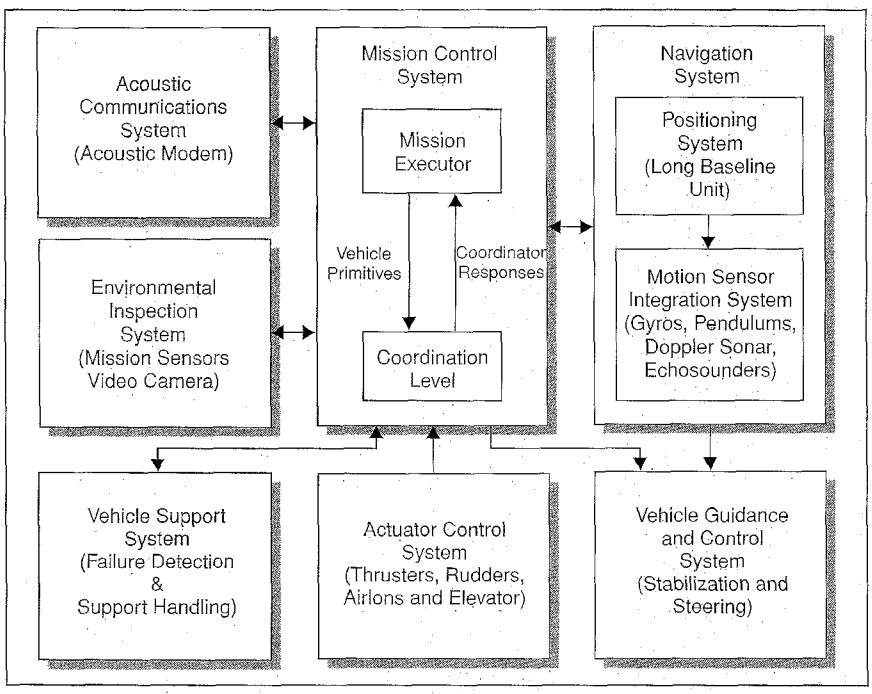


Fig. 5. Marine Utility Vehicle System control architecture.

Marine Utility Vehicle System (MARIUS)

The MARIUS AUV has been developed by a multi-disciplinary team of scientists from Denmark, France, and Portugal, under the Marine Science and Technology (MAST) Programme of the Commission of the European Communities [12]. The vehicle uses open, distributed hardware/software hierarchical control architecture, as shown in Fig. 5. The vehicle control system is divided into an organization, a coordination, and a functional level. The levels consist of several modules: the vehicle support system, the actuator control system, the navigation system, the vehicle guidance and control system, the acoustic communications system, the environmental inspection system, and the mission control system. The vehicle control system has dynamic configuration capabilities, and allows the use of reactive behaviors, thus enhancing system integrity and safety.

Ocean Technologies Testbed for Engineering Research (OTTER)

The OTTER has been developed at the Monterey Bay Aquarium Research Institute and Stanford University [13]. The vehicle implements an object-based task level control (OBTLC) architecture modeled as a three-level hierarchical architecture, as shown in Fig. 6. At the lowest, servo level, classical control theory is employed to design control laws which input sensor signals and output motor control at a fixed sample rate. The middle, task level acts as the encoder of logical and sequential actions, which define a specific task. At the top, organization level, there is human interaction with the AUV, via a graphical display and a virtual-reality interface. The OBTLC reduces the requirements on both data latency and communication rates between the operator and the remote system. The OBTLC enables integration of the decision and judgment capabilities of the human mind with the speed and accuracy of the machine computation and control.

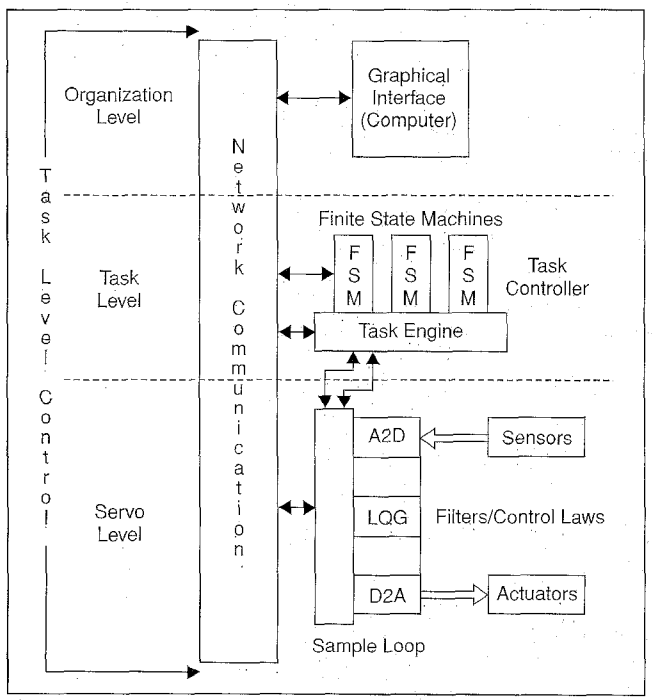


Fig. 6. Ocean Technologies Testbed for Engineering Research control architecture.

Ocean Voyager II

Ocean Voyager II has been developed at the Florida Atlantic University [14]. This vehicle utilizes a distributed hierarchical control system architecture, consisting of subsystems composed of sensors and actuators, with a corresponding microcontroller, as shown in Fig. 7. The microcontrollers act as nodes in a distributed control network. The nodes are connected together by a serial communications network. The nodes function as independent subsystems, so the system is highly modular. The distribution of the control system into subsystems increases the reliability and capability, while the complexity decreases.

Odyssey II

The Odyssey II has been developed at the Massachusetts Institute of Technology [18]. The vehicle adopts subsumption architecture (state-configured layered control), shown in Fig. 8. The vehicle dynamic controller resides at the bottom level of the

architecture. This level commands the actuators to achieve a desired vehicle state, as specified by the layered control level. The layered control level of the vehicle architecture is a state table, responsible for stepping through discrete mission phases. The control system is built around a "vehicle data structure," through which all the vehicle code elements interact. The vehicle data structure keeps track of the vehicle state: input, output, and adjustable settings, with associated time-stamps. Integration of the "vehicle data structure" into the control scheme enhances system integrity, modularity, and augmentation.

Omni-Directional Intelligent Navigator (ODIN)

The ODIN has been developed by the Autonomous Systems Laboratory, University of Hawaii [3]. Its architecture is hybrid, using both hierarchical and heterarchical architecture, as shown in Fig. 9. The supervisory level handles mission parameters on the basis of lower-level information, and three separate block functions: sensory data base, knowledge base, and planner. Each block has multiple layers, demonstrating an increase in intelligence with the increase in the number of layers. The ODIN intelligent control architecture is flexible, overcoming the limitations associated with purely hierarchical architecture. Also, it does not require excessive amount of communication, which is a major limitation of heterarchical architectures. The ODIN architecture is flexible and suitable for parallel computing processes.

Phoenix

The Phoenix AUV has been developed at the Naval Postgraduate School, Monterey [20]. The vehicle control is based on a three-level hybrid software control architecture, as shown in Fig. 10, comprising strategic, tactical, and execution levels. The hierarchical architecture is used between the three levels, and the subsumption architecture is used at the execution level. The strategic level uses Prolog as a rule-based mission control specification language. The tactical level is a set of C language functions that interface with the Prolog predicates and return Boolean val-

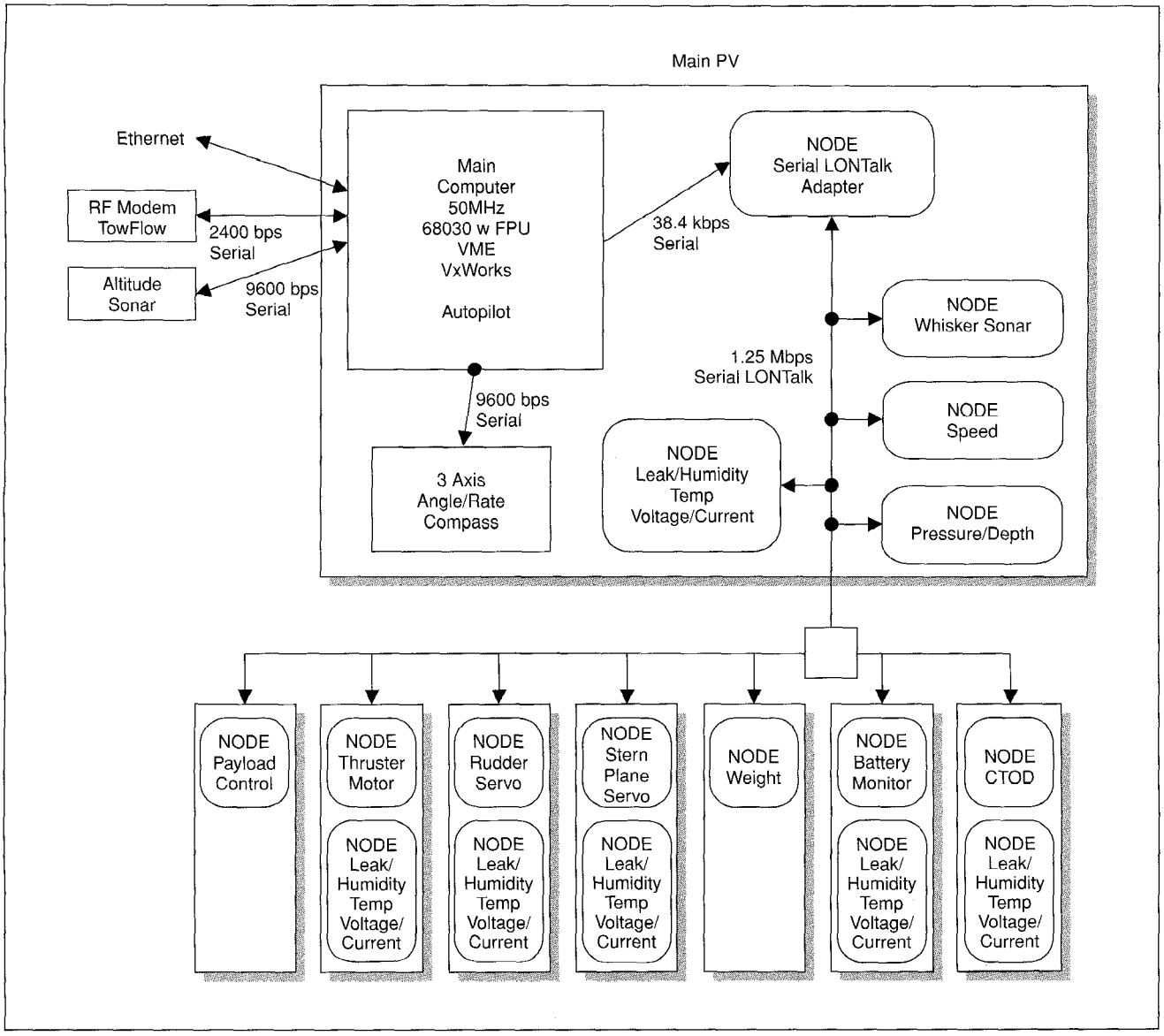


Fig. 7. Ocean Voyager II control architecture.

ues when called. The tactical level interfaces with the real-time execution level controller using asynchronous message passing through network sockets. The execution level operates synchronously and operates vehicle actuators and sensors in response to higher-level commands. The architecture incorporates error detection and recovery procedures. Reactive error recovery can be handled, in association with control performance evaluations, at the tactical level. The strategic level also handles reactive behaviors by transitioning to states that command error recovery procedure, when errors are detected.

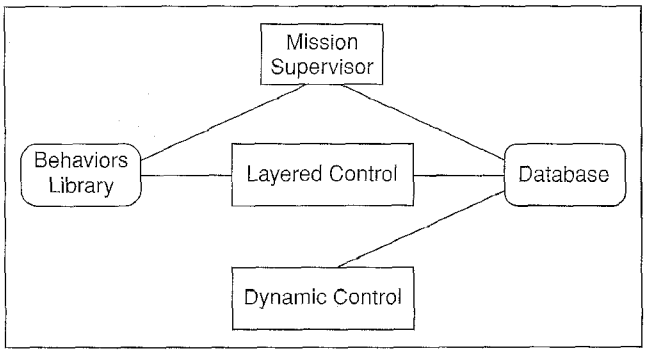


Fig. 8. Odyssey II control architecture.

Sea Squirt

The Sea Squirt has been developed at the Massachusetts Institute of Technology [19]. The vehicle control architecture is based on enhanced subsumption architecture (state configured layered control), represented in Fig. 11. The Sea Squirt's control architecture is divided in two levels: a higher level, implementing the state table, and a lower level, the layered control structure. State-configured layered control overcomes synchronization problems associated with layered control architectures in general, by adding a higher level of control that activates only the behaviors appropriate to a specific phase of the mission. A state table specifies the state of the vehicle, and is responsible for ensuring that the behaviors are activated at the right time and with the right priority. This minimizes the number of behaviors active, making behavior coordination and synchronization much easier compared to a classical layered control implementation.

Proposed Sensor-Based AUV Control System Architecture

The proposed control system is based on a two-level hybrid control architecture, comprising of a supervisory control level and a functional control level, as shown in Fig. 12. The hierarchical control architecture is used between the two levels, and the heterarchical architecture is used at the lower functional level. The overall system functionality is state-configured, based on state diagrams that define specific AUV missions/operations.

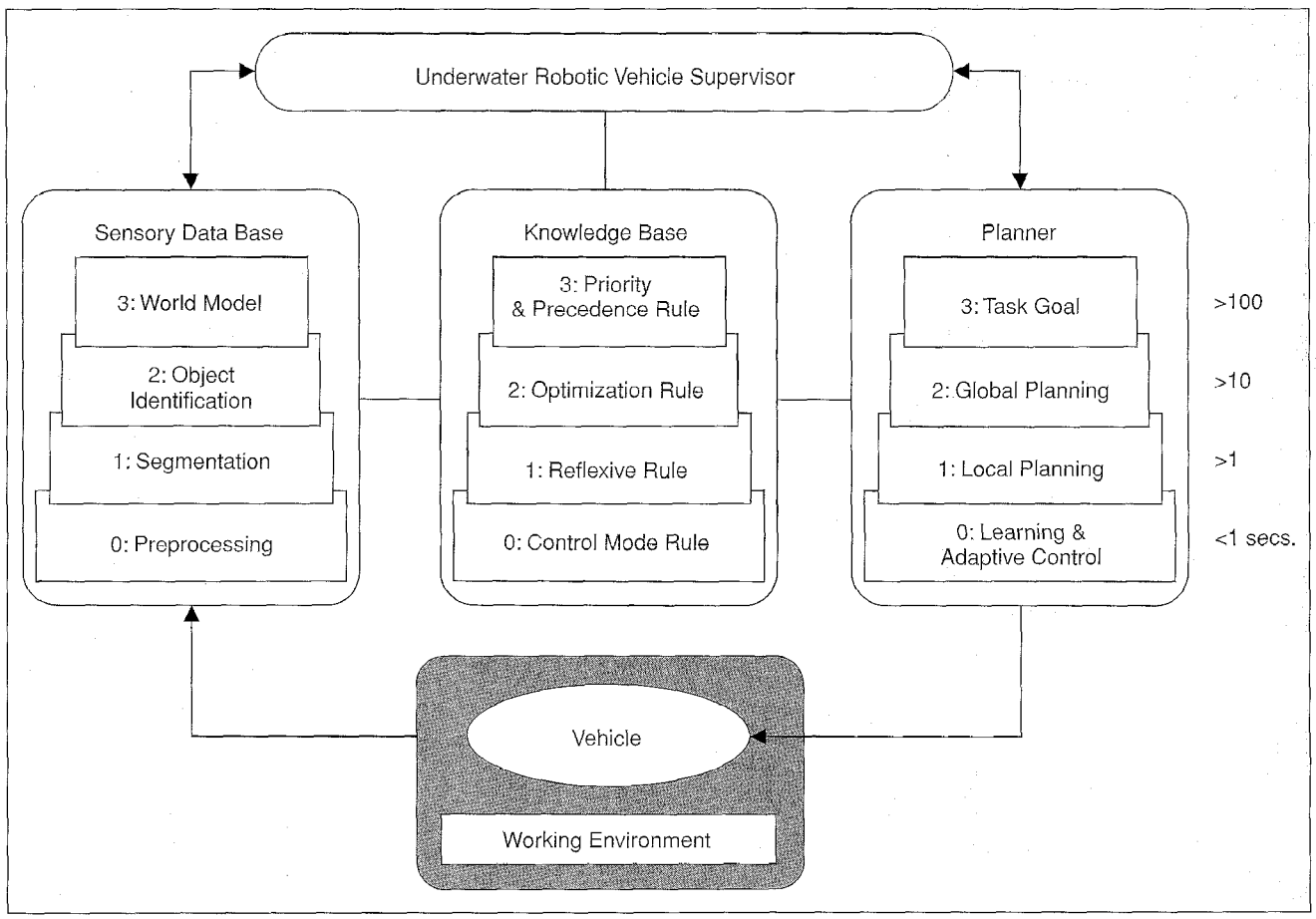


Fig. 9. Omni-Directional Intelligent Navigator control architecture.

Table 2. Microcontroller Control vs. Embedded-PC Control		
Implementation	**Microcontroller-Based**	**Embedded-Based**
Define architecture	Define the product architecture, including both hardware and software.	Define the product architecture and select appropriate off-the-shelf embedded CPU and expansion modules to match system requirements.
Assemble remote hosted (for microcontroller) and self-hosted (for embedded-PC) development system	Select, purchase, and assemble an appropriate remote-hosted development system that includes a workstation for software development, in-circuit emulator (ICE), compiler, and debugger to match the selected CPU.	Configure a development system using development kits for the actual modules to be embedded and appropriate disk and display interface modules.
Design hardware/software	Write the required operating software and design all required CPU and I/O hardware.	Obtain appropriate compatible operating system or real time executive, libraries and drivers. Write required software, design any required unique hardware interfaces using the industry standard bus specifications.
Integrate/debug hardware/software	Debug the OS on the remote-hosted development system. Use an ICE to bring up the target system CPU and basic system resources. Download and test the application software from the remote-hosted development system.	Test and debug the application directly within the self-hosted development system booted from floppy or hard disk.
Put software in EPROM (for microcontroller) and in SSD (for embedded-PC)	Burn EPROM, equivalent to the downloaded and tested software, install in the target system.	Install Solid State Disk (SSD) support software to create programmed byte-wide memory devices containing the equivalent of the system's boot floppy diskette. Install these SSD devices in the development system and verify proper operation.
Final test	Test and debug the ROM version of the application software in the target system.	Remove any modules not needed in the final embedded system. The self-hosted development system has now become the target system.

The state diagram residing at the supervisory control level determines the sequence of AUV tasks/operations throughout the various phases of a mission and is also responsible for transfer of operations from one phase to another.

The supervisory control component is responsible for the coordination of the overall AUV navigation, guidance, and control. It monitors and coordinates the order of module task execution (of the functional component). The functional component is responsible for specific tasks/operations occurring in a mission. Each module is designed to perform a well-defined set of tasks; all modules have their own local memory. The functional level modules directly control the vehicle's actuators, sensors, and all hardware components residing directly on the vehicle.

The described control architecture is modular, and the functionality of each module is determined based on specific tasks performed. All modules share a common communication bus for data storage and retrieval. There is no direct communication between individual modules. Information exchange is accomplished through shared variables.

This architecture offers the following advantages as a whole, compared to other reviewed architectures.

- It utilizes the shared memory module as a communication medium between the various modules, thus eliminating other more complex means of communication between the modules. This also eliminates any overhead generated by the communication protocols that would have been used otherwise.
- It uses state diagrams to specify missions, and thus makes it easy to change missions on the fly, or to reprogram the operation.
- The functionality of the architecture is based on software functions that run on the various modules. Since modules operate independently, the software of a specific module can easily be modified, improved, and changed altogether without affecting the functionality of the other modules.

A major advantage of this scheme is that all modules can directly access the shared memory without having to go through the supervisor or an intermediate level. This minimizes data transfer latency between modules and across levels.

Supervisory Control Component

The state-configured supervisory control level consists of the Master Controller (MC). In state-configured control, only the goal-oriented modules pertinent to the specific phase of the mission are active. The others remain inactive. Thus, power consumption requirements are minimized, and coordination among the modules is simplified. The responsibility for ensuring that the modules are activated at the right time and with the right priority is delegated to the state diagram of the MC. A similar approach was used by Bellingham and Consi in [19]. The main differences of our design are: i) the communication between the various modules is accomplished by using shared memory variables, and ii) the overall operation of the system is coordinated by the MC.

The shared memory is controlled by the MC. Since access to shared memory variables is controlled using semaphores and automatic logging of variables needs time stamps, every variable is accompanied by a semaphore variable and a time stamp variable. Every module has its own local memory (for execution purposes). Every module keeps a local copy of any shared memory variable that it needs to access. Before accessing a shared memory variable, a module must set its semaphore and reset it after the variable data is transferred to the local copy of the variable. Copying the shared variables to the local memory before proceeding with the execution of the module functions rather than manipulating the shared variables in the shared memory minimizes the time a module accesses the shared memory, thus allowing other modules to proceed with their execution and accessing of the shared memory. Fig. 13 shows the internal structure of the MC. The MC contains a main/central processor (Intel Pentium) and a local memory unit (for execution purposes), and the following three software processes that coordinate the operation of the AUV.

- *The Shared Memory Management (SMM) process* facilitates the variable transfers between the modules and the shared memory. Exchanged information includes, for example, the parameters supplied by the sensor control module, the current map file generated by the map generator module, etc. It is also responsible for the signals that are issued by the modules. These signals include requests for variables, acknowledgments, and so forth. The SMM is also responsible for periodic backup of all information stored in the shared memory variables, after a specific phase has completed its operation. This is necessary for recovery purposes and for reviewing the overall mission performance of the AUV.
- *The Interrupt Handling (IH) process* monitors the interrupt variables that are sent by the Monitoring and Recovery module, in case of a software or hardware malfunction. If an interrupt occurs, IH performs the necessary action, usually aborting the operation, and initiating an appropriate recovery function.
- *The Mission Scheduler (MS) process* coordinates the operation of the AUV by transferring control among the modules. The transfer of operation is based on state diagrams, representing the mission of the AUV.

The three processes of the MC also access the shared memory. A set of flags, associated with the modules of the system, is kept in the shared memory. Every time the state diagram reaches a new mission phase, it sets the flags associated with the modules that need to be activated. The modules periodically "look" at these flags and according to their status become active or remain inactive. After a phase is completed, all of the flags are reset by the MC. The backup function, in the MC, backs up all information stored in the shared memory variables, after a specific phase has completed its operation. This is necessary for recovery purposes and for reviewing the overall mission performance of the AUV. The system can retrieve information and make decisions for recovery techniques based on previously stored information.

The means of communication among the system modules is a shared memory system (Fig. 12). In such a system, data integrity and synchronization need to be maintained at all times. This is usually done using mutual exclusion by employing semaphores [9]. To avoid corruption of the shared memory, trusted functions are used to guarantee that only relevant parts of the shared memory are accessed by various processes (modules).

The shared memory system is accessible by all the modules of the system. The MC is responsible for coordination and management of the shared memory. The management and use of shared memory variables are easy, flexible, and powerful. The various

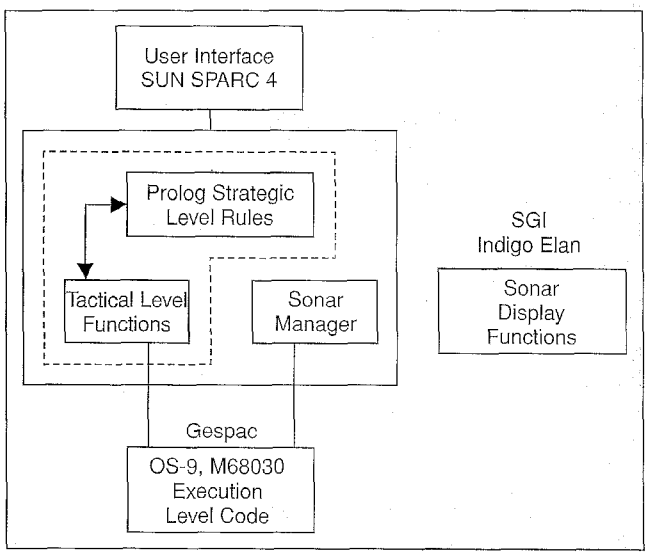


Fig. 10. Phoenix control architecture.

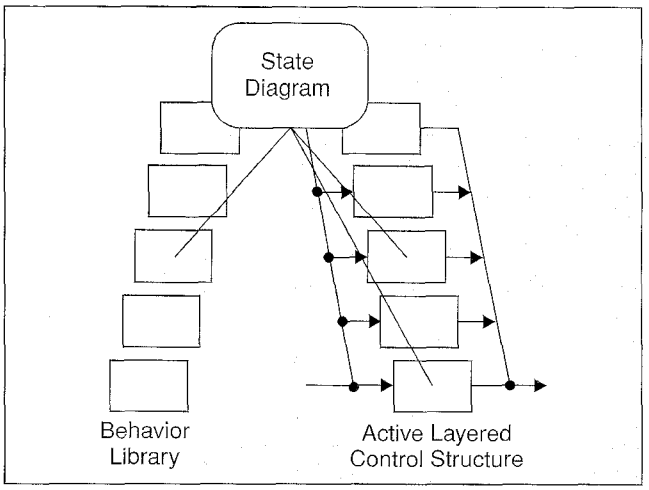


Fig. 11. Sea Squirt control architecture.

modules need not know about the functions of other modules. The only information a module needs is the data stored in pertinent variables in shared memory.

Functional Component

The Functional Component is composed of functionally independent modules, described as follows.

- *The Sensor Control (SC)* module controls the activities of the sensors and receives information from the sensors. It makes necessary calculations and generates results that are needed by the rest of the modules of the system and sends the results to the shared memory.
- *The Map Generation (MG)* module generates maps of the 3-D environment based on data obtained by the sensors and on any information previously known about the environment.
- *The Navigation and Task Execution (NT)* module is responsible for generating collision-free paths, generating trajectories over the paths, and avoiding obstacles. The generated paths can be either global (from point A to point B) or local (sub-point paths between A and B). This module is also responsible for performing mission tasks (i.e., pick object), by controlling the motors, fins, thrusters, gyros, servos, and end effector(s) of the vehicle.
- *The Object Recognition and Classification (RC) module* obtains information generated by the Sensors Control module and classifies the objects detected by the sensors. After classification, the objects are stored in the Knowledge Base (KB) of the AUV. The KB database is located in a physical storage device (i.e. a solid state disk (SSD)).

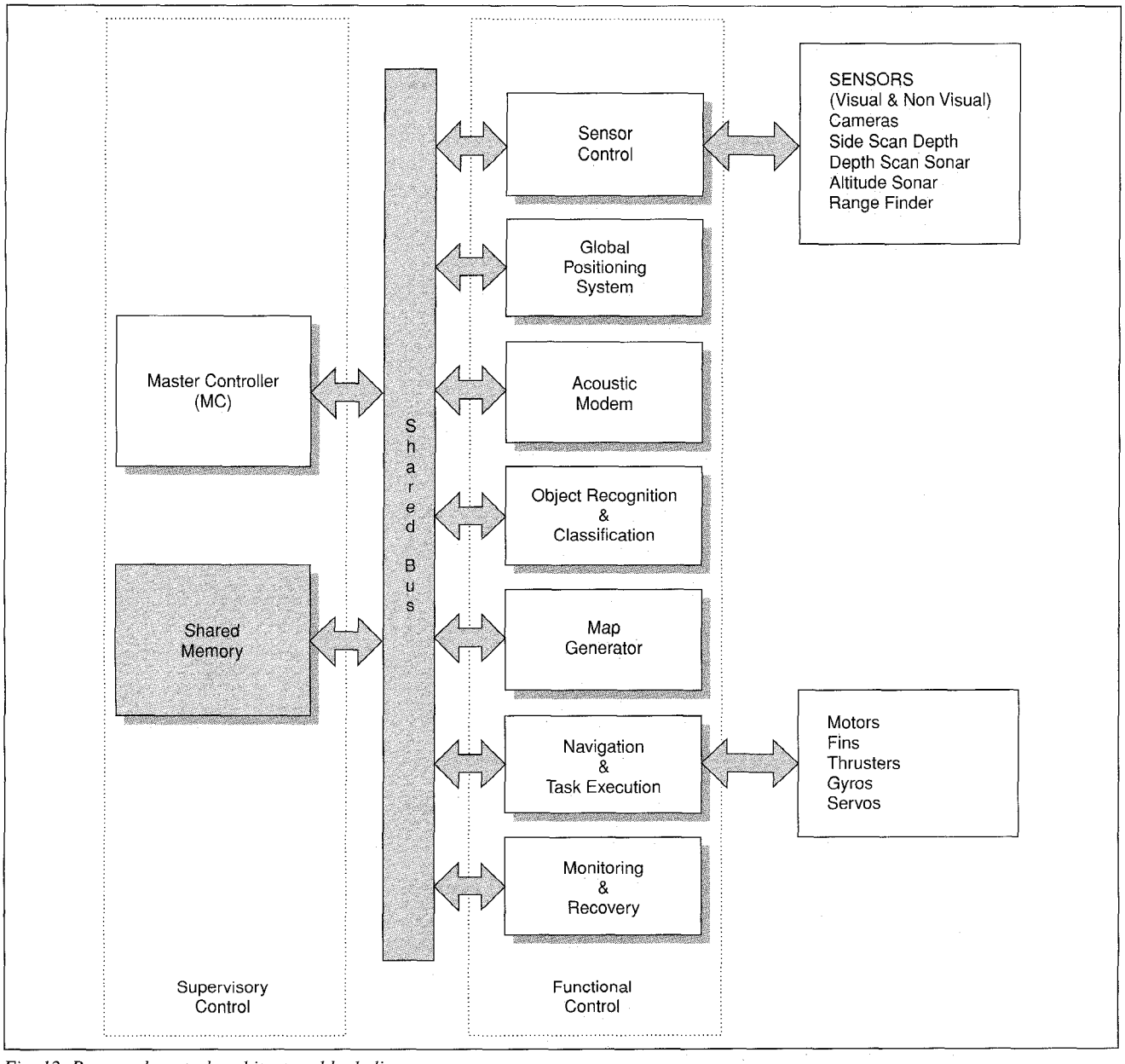


Fig. 12. Proposed control architecture block diagram.

- *The Global Positioning System (GPS) module* calculates the correct location of the vehicle in relation to the world (environment).
- *The Monitoring and Recovery (MR) module* monitors the overall functionality of the system. It locates malfunctions in either software or hardware, and initiates recovery procedures when necessary. Recovery procedures are also initiated if the vehicle runs into situations in which normal operation is impossible.
- *The Acoustic Modem (AM) module* communicates with operators or computers external to the vehicle. This allows monitoring of the system and issuing new missions (in the form of state diagrams) by external sources.

Proposed AUV Control System Architecture Implementation

There exist two main approaches to design and implement a hardware and software control system architecture. One approach follows a microcontroller-based system design; the other follows an embedded control system design. However, regardless of the particular implementation, from 4-bit/8-bit single chip microcontrollers to high performance RISC processors, embedded control system design minimizes risks, cost, as well as the development time, and provides a ready platform for running application-specific software. Table 2 compares the development procedure involved in both microcontroller and embedded designs and justifies the authors' preference to use an embedded controller to implement the proposed AUV architecture.

The proposed architecture is a state-configured embedded control architecture, so standard software and hardware components are utilized for development of a control system. While from the software perspective standards are rather mature (programming languages, communication protocols), from the hardware aspect the diversity of microprocessor and microcontroller architectures has prevented the emergence of any real standards for embedded system hardware. Only industrial computer buses such as VME, Multibus, and STD offer some degree of consistency. A detailed comparative evaluation of the emerging STD 32 and CompactPCI single board computer (SBC) technology has been performed, from the perspective of implementation of the AUV control architecture.

Having reviewed the existing technology, it has been decided to use the real-time QNX operating system for software development, and the Single Board Computers (SBCs) with the STD 32 standard as a hardware. It needs to be noted that this architecture can also be implemented using the microcontroller design method. The microcontroller method, though, would be more expensive and more complex. The use of SBCs allows the system to be easily expandable, easily upgradable, modular, reliable, and very inexpensive.

The STD bus has been the standard bus for industrial control systems since the 1970s. The STD-32 Bus implements a small, industrial strength, scalable, and versatile architecture suitable for demanding real-time control and data acquisition applications where small system size and cost are important. STD 32 is an open, well-designed standard with a wide range of processors, peripherals, industrial I/O, enclosures, and complete systems from numerous manufacturers.

The STD 32 Bus can run at 32 Mbytes per second for very high-speed data processing applications. Other performance characteristics include: multiprocessing, with centralized arbitration logic to monitor access to the bus, that allows the implementation of multiple processors in a single STD 32 system. The 32-bit throughput of the bus is crucial to inter-processor communication in real-time multiprocessing applications; 32-bit addressing and pipelining dramatically improves throughput for block data transfers by reducing bus cycle time and increasing bus bandwidth; high-speed Direct Memory Access (DMA) over the backplane streamlines the operation of data-intensive applications; slot-specific interrupts expand the number of available system interrupts for servicing systems requests.

A critical component in an STD 32 system is the backplane. The backplane design incorporates several important features including increased backplane signal impedance. A higher backplane signal impedance means that "cleaner" signals are sent across the backplane. That is, ringing and reflections are minimized. This is especially important during signal transitions between the TTL threshold regions of 0.8V and 2.0V. Fig. 14 shows a typical STD-32 system. The system shown here is the STD 32 STAR System, available from Ziatech Corporation. It is a simple yet extremely powerful approach to the design of real-time control computers. It includes multiple PC-compatible CPU cards in a single card cage (backplane).

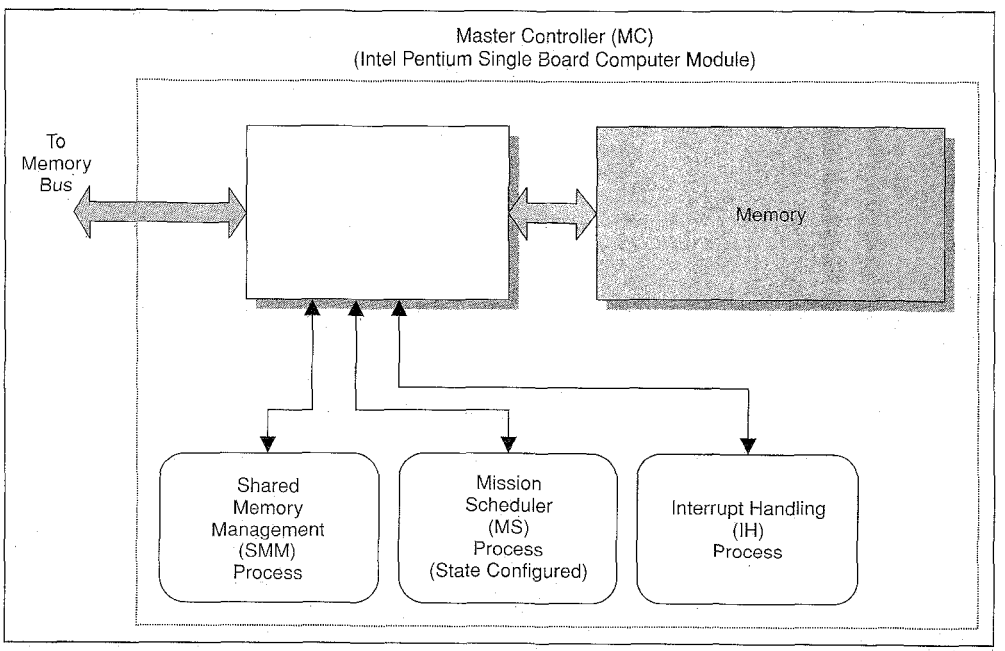


Fig. 13. Master Controller block diagram.

Each CPU has its own memory and operating system, but shares backplane memory, disks, video, and I/O with other CPUs in the system.

The STD 32 has been preferred over CompactPCI for the following reasons:

- **Multiprocessing:** CompactPCI does not support multiprocessing, which is critical for the performance of the proposed system, where each module is a separate processor. STD 32, in contrast, supports up to seven processors. Each CPU functions as a module of the overall system. This multi processor system is designed using the STD 32 STAR system that works under the QNX real-time operating system. All processors are functioning independently of the rest of the system, and the exchange of information is done through the shared memory. This eliminates the overhead of any communication protocols between the various single-board computers. Further, each of the processors in the STD 32 STAR system has Direct Memory Access (DMA) without multiplexing to the shared memory (common memory). This eliminates any delays associated with accessing the shared memory since all processors can simultaneously (without multiplexing) access the memory. Exchanging data using the shared memory as a means of communication is a fast approach compared to networking.
- **Power consumption:** CompactPCI only supports Pentium and higher processors, with an increased power consumption, which is at a premium in the case of an AUV. STD 32 can support any type of processor ranging from the low-end 386 to the high-end state-of-the-art Pentium family of processors.
- **I/O Requirements:** CompactPCI has been primarily designed to operate as a high-speed computing core for applications with modest I/O requirements only. It design goal was to be used to enhance the STD 32 bus and the VME bus. In the current situation, the system continually processes a large volume of input stream from the sensors, thereby increasing the demand on I/O requirements.

Table 3 provides a comparative evaluation of certain features of the STD 32 and Compact PCI single-board computers, which justify the selection of the STD 32 as the hardware platform for implementing the AUV control architecture. However, the main rationale for selecting the STD 32 hardware platform is the fact that STD 32 supports multiprocessing with up to seven processors having DMA to a shared memory without multiplexing.

Table 3. Comparison of STD 32 and CompactPCI Single-Board Computers

Feature	STD 32	CompactPCI
Multiprocessing	Up to seven processors	N/A
Clock frequency	8 MHz	33 MHz
Data transfer types	8-bit / 16-bit / 32-bit	32-bit / 64-bit
Maximum bandwidth	32 MB/s	133 MB/s / 266 MB/s
Processors	386 - Pentium family	Pentium family
Power consumption	Variable, processor-dependent	High
Fault tolerant	Yes	No
Hot swap capability	Yes	No
Peripheral slots	Up to 14 if only 1 processor is used	Eight (expandable with PCI to PCI bridges)

The STD 32 Star System uses the QNX operating system (OS) as its real-time OS. QNX is a scalable, multitasking, real-time OS with POSIX capabilities; two features often required for embedded control applications. Thus, the QNX real-time OS has been chosen.

For embedded control applications, the modularity of the OS allows the developer to omit unneeded system processes. With the addition of the small QNX networking module, an embedded system can become a network-transparent extension of a larger QNX environment for distributed applications, booting either from ROM or from the network. Device drivers exist for many popular embedded PC ROM environments.

With its micro-kernel, message-passing architecture, QNX can take a network of computers and present them to applications as a "single logical machine," regardless of how many physical computers are joined by the network. Applications developed for this "single logical machine" will run without changes even as the number of computers is scaled to suit the scope of the application. This scalability is possible because QNX allows applications to be designed as a team of cooperating, communicating processes on a single machine. When run on a QNX network, these processes can be configured to run throughout the network, while QNX provides network-transparent messaging between these processes. The networking allows any process to use any resource on any computer on the network. Multiple redundant network links between network nodes provide protection from network failures as well.

Currently, the embedded control architecture design has been completed at conceptual level. It is now being simulated using

Table 4. Notation and Coordinates for Underwater Vehicles

DOF	Name	Forces/ Moments	Linear/ Angular Velocities	Position & Euler Angles
1	Surge (x-axis motion)	X	u	x
2	Sway (y-axis motion)	Y	v	y
3	Heave (z-axis motion)	Z	w	z
4	Roll (rotation about x)	K	p	ϕ
5	Pitch (rotation about y)	M	q	θ
6	Yaw (rotation about z)	N	r	ψ

Petri nets to ensure its functionality. The Phantom S2 ROV (from Deep Ocean Engineering) has been acquired, and is being modified to convert it to an AUV.

Conclusions

Unmanned underwater vehicles have been proven as a useful tool for performing missions relevant to the ocean industry. However, while ROVs are being extensively used by the offshore oil industry (mainly) and other industries as well, AUVs have yet to establish their own niche in the market. Although AUV technology has overcome many obstacles, there are still major challenges related to power sources, navigation, and mission management.

In this article, a flavor of the state of the art in AUV technology is provided, several AUV control architectures are presented, and a comprehensive table summarizing operational features of 25 AUVs is included. Then, in an attempt to establish a standard hardware and software platform for an AUV control architecture, an embedded control hybrid architecture, based on the QNX real-time operating system and the STD 32 SBC, is proposed.

A World Wide Web (WWW) Site: The Autonomous Underwater Vehicle Resources WWW Page (http://www.acim.usl.edu/AUV/) contains a collection of links to institutions, departments, and companies involved in AUV related research and development. It provides an extensive list of references and resources available on the Internet for AUV information. The list presents all WWW resources known to the authors; however, it should not be considered as a complete general reference. Wherever applicable, a short description or annotation has been added. The WWW page has been constantly updated and expanded with new links as they become available.

Appendix: AUV Kinematics and Dynamics Equations of Motion

Consider a marine vehicle (an autonomous underwater vehicle, AUV, or a remotely operated vehicle, ROV). Define a moving coordinate reference frame $X_0\ Y_0\ Z_0$ attached to the vehicle, called the *body fixed* reference frame, with origin coinciding with the vehicle's center of gravity and with X_0, Y_0, Z_0 serving as the vehicle's principal axes of inertia. Define a fixed/world coordinate frame XYZ, called the *earth fixed* reference frame [3].

A marine vehicle's position and orientation in the 3-D space is determined as a function of six degrees of freedom (DOF) with parameters as defined in Table 4. The first three coordinates and their time derivatives determine the vehicle's position and translational motion along the x-, y-, z- axes, while the last three and their time derivatives determine the vehicle's orientation and rotational motion.

Based on Table 4, for any marine vehicle, the following vectors are defined:

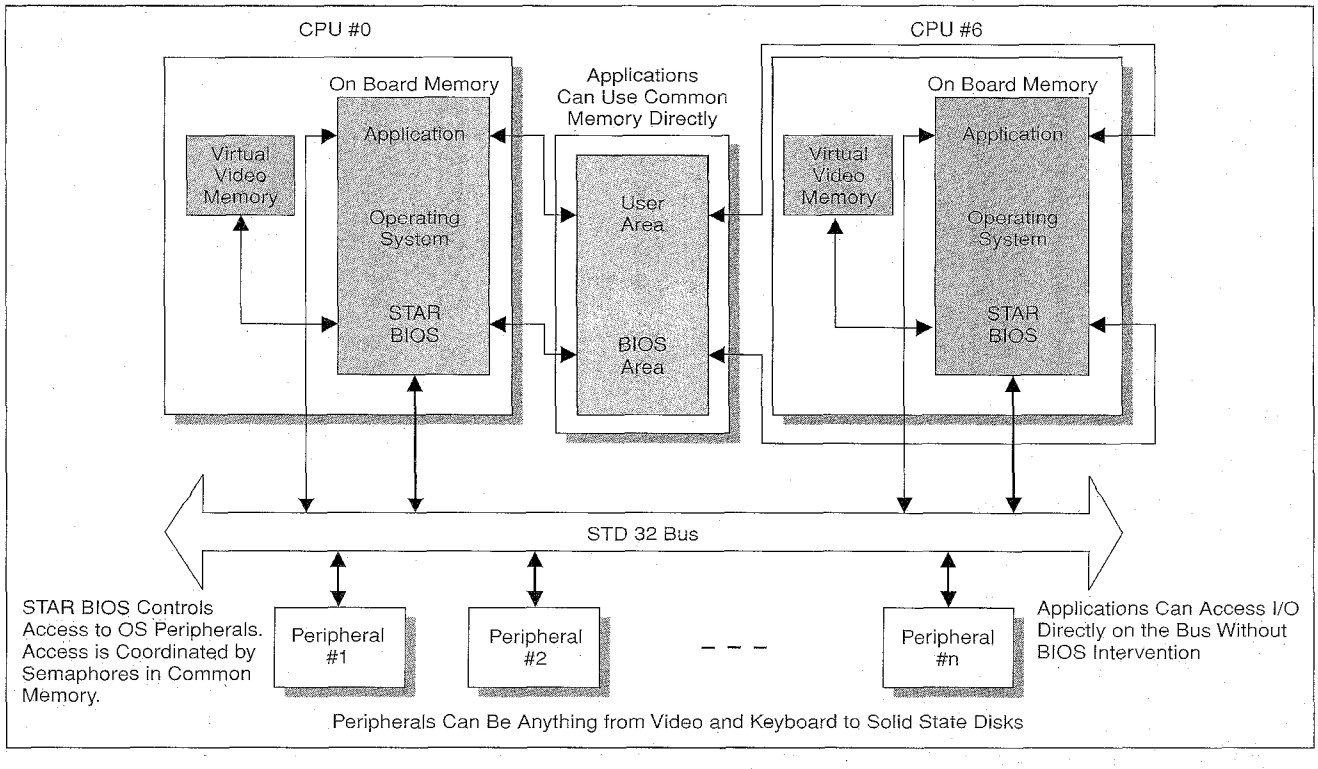


Fig. 14. STAR System architecture.

$$\eta_1 = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \eta_2 = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}, \quad \eta = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$$

$$\nu_1 = \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \quad \nu_2 = \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \quad \nu = \begin{pmatrix} \nu_1 \\ \nu_2 \end{pmatrix}$$

$$\tau_1 = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \quad \tau_2 = \begin{pmatrix} K \\ M \\ N \end{pmatrix}, \quad \tau = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix}$$

The vector $\eta = (x \quad y \quad z \quad \phi \quad \theta \quad \psi)^T$ defines the vehicle's position and orientation with the respect to the earth fixed reference frame; the vector $\nu = (u \quad v \quad w \quad p \quad q \quad r)^T$ defines the vehicle's linear and angular velocity with respect to the body fixed reference frame; the vector $\tau = (X \quad Y \quad Z \quad K \quad M \quad N)^T$ defines the forces and torques with respect to the body reference frame.

After defining the three principal rotation matrices (rotations) about the *x*-, *y*-, *z*-axis respectively, $C_{x,\phi}, C_{y,\theta}, C_{z,\psi}$, the linear velocity transformation which determines the vehicle's path relative to the earth fixed reference frame is:

$$\dot{\eta}_1 = J_1(\eta_2)\nu_1, \quad \nu_1 = J_1^{-1}(\eta_2)\dot{\eta}_1$$

where:

$$J_1(\eta_2) = C_{z,\psi}^T C_{y,\theta}^T C_{x,\phi}^T, \quad J_1^{-1}(\eta_2) = J_1^T(\eta_2)$$

The orientation of the body fixed reference frame with respect to the earth fixed frame is given by:

$$\nu_2 = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + C_{x,\phi}\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + C_{x,\phi}C_{y,\theta}\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}$$

which is rewritten as:

$$\nu_2 = J_2^{-1}(\eta_2)\dot{\eta}_2$$

with $J_2(\eta_2)$ undefined for $\theta = \pm 90°$ and $J_2^{-1}(\eta_2) \neq J_2^T(\eta_2)$. Since ν_2 cannot be integrated to obtain angular coordinates, η_2 is used instead. Therefore, the vehicle's kinematic equations may be expressed in the following form:

$$\begin{pmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{pmatrix} = \begin{pmatrix} J_1(\eta_2) & 0 \\ 0 & J_2(\eta_2) \end{pmatrix}\begin{pmatrix} \nu_1 \\ \nu_2 \end{pmatrix} \Leftrightarrow \dot{\eta} = J(\eta)\nu$$

where:

$$J_1(\eta_2) = \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\phi\sin\theta\sin\psi & -\cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

$$J_2^{-1}(\eta_2) = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{pmatrix}$$

$$J_2(\eta_2) = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{pmatrix}$$

The body-fixed frame 6-DOF nonlinear dynamic equations of motion are represented in compact form (including vehicle thruster forces, hydrodynamics damping, and lift and restoring forces) as follows:

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau$$

$$\dot{\eta} = J(\eta)\nu$$

where M is inertia matrix (with added mass), $C(\nu)$ is matrix of Coriolis and centripetal terms (with added mass), $D(\nu)$ is damping matrix, $g(\eta)$ is vector of gravitational forces and moments, and τ is vector of control inputs. The earth-fixed 6-DOF nonlinear dynamic equations of motion are obtained by applying:

$$\dot{\eta} = J(\eta)\nu \Leftrightarrow \nu = J^{-1}(\eta)\dot{\eta}$$

$$\ddot{\eta} = J(\eta)\dot{\nu} + \dot{J}(\eta)\nu \Leftrightarrow \dot{\nu} = J^{-1}(n)\left(\ddot{\eta} - \dot{J}(\eta)J^{-1}(\eta)\dot{\eta}\right)$$

Assuming that $J(\eta)$ is bounded away from $\theta = \pm 90°$.

$$M_\eta(\eta) = J^{-T}(\eta)MJ^{-1}(\eta)$$

$$C_\eta(\nu, \eta) = J^{-T}(\eta)\left(C(\nu) - MJ^{-1}(\eta)\dot{J}(\eta)\right)J^{-1}(\eta)$$

$$D_\eta(\nu, \eta) = J^{-T}(\eta)D(\nu)J^{-1}(\eta)$$

$$g_\eta(\eta) = J^{-T}(\eta)g(\eta)$$

$$\tau_\eta(\eta) = J^{-T}(\eta)\tau$$

so the earth-fixed vector representation is:

$$M_\eta(\eta)\ddot{\eta} + C_\eta(\nu, \eta)\dot{\eta} + D_\eta(\nu, \eta)\dot{\eta} + g_\eta(\eta) = \tau_\eta$$

References

[1] S.W. Kandebo, *AW&ST and AUVSI 1997-98 International Guide to Unmanned Vehicles*. New York: McGraw-Hill, Inc., 1997.

[2] —, "The Swiss ADS 95 Ranger UAV System," *Unmanned Systems*, vol. 15, pp. 16-24, Winter 1997.

[3] J. Yuh, ed., *Underwater Robotic Vehicles: Design and Control*. Albuquerque, NM: TSI Press, 1995.

[4] D.R. Blidberg, "Autonomous Underwater Vehicles: A Tool for the Ocean," *Unmanned Systems*, vol. 9, pp. 10-15, Spring 1991.

[5] A.S. Westneat, D.R. Blidberg, and R.W. Corell, "Advances in Unmanned, Untethered Underwater Vehicles," *Unmanned Systems*, vol. 1, no. 3, pp. 8-13, 1983.

[6] J. Simons, ed., *Remotely Operated Vehicles of the World*. Ledbury, England: Oilfield Publications Limited, '96/7 ed., 1997.

[7] J. Yuh, "Autonomous Underwater Robot Design," 1996 World Automation Congress, Tutorial 2: Underwater Robotic Systems ASL96-01, Autonomous Systems Laboratory, Dept. of Mechanical Engineering, University of Hawaii, Honolulu, HI 96822, 1996.

[8] E. Coste-Maniere, H. H. Wang, and A. Peuch, "Control Architectures: What's Going On?," in *Proceedings of the International Program Development in Undersea Robotics & Intelligent Control (URIC): A Joint U.S./Portugal Workshop*, (Lisboa, Portugal), pp. 54-60, March 2-3, 1995.

[9] K. Ganesan, S.M. Smith, K. White, and T. Flanigan, "A Pragmatic Software Architecture for UUVs," in *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, (Monterey, CA), pp. 209-215, June 2-6, 1996.

[10] D.R. Yoerger, A.M. Bradley, and B. Walden, "System Testing of the Autonomous Benthic Explorer," in *Proceedings of the IARP 2nd Workshop on: Mobile Robots for Subsea Environments*, pp. 159-170, May 3-6, 1994.

[11] D.R. Blidberg, S. Chappel, J. Jalbert, R. Turner, G. Sedor, and P. Eaton, "The EAVE AUV Program at the Marine Systems Engineering Laboratory," in *Proceedings of the IARP 1st Workshop on: Mobile Robots for Subsea Environments* (Monterey, CA), pp. 33-42, Oct. 23-26, 1990.

[12] A. Pascoal, C. Silvester, P. Oliveira, D. Fryxell, and V. Silve, "Undersea Robotics Research at IST: The AUV MARIUS Programme," in *Proceedings of the International Program Development in Undersea Robotics & Intelligent Control (URIC): A Joint U.S./Portugal Workshop*, (Lisboa, Portugal), pp. 111-118, March 2-3, 1995.

[13] S.M. Rock, H.H. Wang, and M.J. Lee, "Task-Directed Precision Control of the MBARI/Stanford OTTER AUV," in *Proceedings of the International Program Development in Undersea Robotics & Intelligent Control (URIC): A Joint U.S./Portugal Workshop*, (Lisboa, Portugal), pp. 131-138, March 2-3, 1995.

[14] S.M. Smith, "An Approach to Intelligent Distributed Control for Autonomous Underwater Vehicles," in *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology*, (Cambridge, MA), pp. 105-111, July 19-20, 1994.

[15] F. Janabi-Sharifi and W.J. Wilson, "An Intelligent Assembly Robotics System Based on Relative Pose Measurements," *Journal of Intelligent and Robotic Systems*, vol. 12, no. 1, pp. 49-86, 1995.

[16] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14-23, 1986.

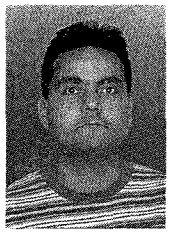
[17] A.J. Boswell and J.R. Leaney, "Using the Subsumption Architecture in an Autonomous Underwater Robot: Expostulations, Extensions and Experiences," in *Proceedings of the IARP 2nd Workshop on: Mobile Robots for Subsea Environments*, pp. 95-106, May 3-6, 1994.

[18] J.G. Bellingham and J.J. Leonard, "Task Configuration with Layered Control," MIT Sea Grant College Program, Autonomous Underwater Vehicles Laboratory Report MITSG 94-24J, The MIT Press, Cambridge, MA, 1994.

[19] J.G. Bellingham and T.R. Consi, "State Configured Layered Control," in *Proceedings of the IARP 1st Workshop on: Mobile Robots for Subsea Environments*, (Monterey, CA), pp. 75-80, Oct. 23-26, 1990.

[20] A.J. Healey, D.B. Marco, R.B. McGhee, B.P. Brutzman, R. Cristi, and F.A. Papoulias, "Coordinating the Hovering Behaviors of the NPS AUV II Using Onboard Sonar Servoing," in *Proceedings of the IARP 2nd Workshop on: Mobile Robots for Subsea Environments*, pp. 53-62, May 3-6, 1994.

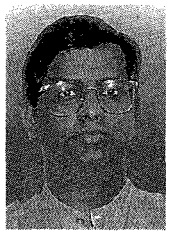
[21] D. Barnett, S. McClaran, E. Nelson, M. McDermott, and G. Williams, "Architecture of the Texas A&M Autonomous Underwater Vehicle Controller," in *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, (Monterey, CA), pp. 231-237, June 2-6, 1996.



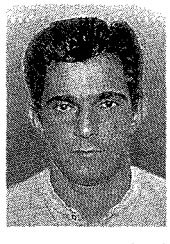
Kimon P. Valavanis was born in Athens, Greece, in 1957. He received the Diploma in Electrical Engineering, Division of Electronic Engineering (five years of study) from the National Technical University of Athens (NTUA), Athens, Greece, in June 1981. Dr. Valavanis received the M.Sc. and Ph.D. degrees from Rensselaer Polytechnic Institute (RPI) in electrical engineering and computer and systems engineering in 1984 and 1986, respectively. Since January 1991, he has been with The Center for Advanced Computer Studies, The University of Southwestern Louisiana (USL), where he is currently professor of computer engineering and associate director for research at the USL A-CIM Center. Dr. Valavanis is a Senior member of IEEE.

Denis Gracanin was born in Rijeka, Croatia, in 1963. He received the B.Sc. and M.Sc. in electrical engineering from the University of Zagreb, Croatia, in 1985 and 1988, respectively. He also received the M.Sc. and Ph.D. degrees in computer science from the University of Southwestern Louisiana, Lafayette, LA, in 1992 and 1994, respectively. Between 1985 and 1991 he was a lecturer assistant at the Telecommunications Department, University of Zagreb, Croatia. He is currently a research scientist at the A-CIM Center, University of Southwestern Louisiana, LA. His research interest are in the field of virtual reality, intelligent robotic systems, Petri nets, and automated manufacturing systems. Dr. Gracanin is a member of IEEE, ACM, AAAI, APS, and SIAM.

Maja Matijasevic received the B.S. and M.S. degrees in electrical engineering (telecommunications) from the University of Zagreb, Croatia, in 1990 and 1994, respectively. Since 1991 she has been affiliated with the Telecommunications Department of the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia. She is currently working toward her Ph.D. as a research associate at the Center for Advanced Computer Studies of the University of Southwestern Louisiana in Lafayette, LA. Her main research interests include computer and communications networks, multimedia, and virtual reality.



Ramesh Kolluru received his Ph.D. in computer science in 1996 from the University of Southwestern Louisiana (USL), Lafayette, LA. He graduated with honors from USL with a masters' degree in computer science in 1995, and from Osmania University, India, with a B.S. in mechanical engineering in 1992. He is currently a research scientist at the A-CIM Center, USL. He is closely affiliated with the Robotics and Automation Laboratory of the USL, and is responsible for implementation of the research agenda at the Center. His research interests include the areas of intelligent control, sensor-based control architectures, actuators and sensors, modeling and design of complex mechanisms, and factory automation. He is a member of the IEEE, RIA, ACM, and a life member of the Phi Kappa Phi Honor Society.



Georgios Demetriou received his B.Sc. degree in electrical engineering and his M.Sc. degree in computer engineering from the University of Southwestern Louisiana, Lafayette, LA, in the 1991 and 1994, respectively. He is currently working toward his Ph.D. in computer science at the Center for Advanced Computer Studies of the University of Southwestern Louisiana, Lafayette, LA. His main research interests include robotics, control architectures, teleoperations, and computer communications.