

## *OTTER*: The Design and Development of an Intelligent Underwater Robot

**HOWARD H. WANG, STEPHEN M. ROCK**

*lazarus@sun-valley.stanford.edu, rock@sun-valley.stanford.edu*

*Aerospace Robotics Laboratory, Department of Aeronautics and Astronautics, Stanford University, Stanford CA 94305*

**MICHAEL J. LEE**

*lemi@mbari.org*

*Monterey Bay Aquarium Research Institute, Pacific Grove CA 93950*

Received ??. Revised ??.

Editors: J. Yuh

### **Abstract.**

Recent advances in sensing and intelligent control technologies open a whole new dimension in underwater autonomy. However, before truly-capable, autonomous underwater robots can be created for subsea intervention and exploration, many research issues must be first investigated and developed experimentally on testbed platforms.

*OTTER* is an underwater robot designed to be used as a testbed for autonomous technologies. Both *OTTER*'s hardware and software systems are configured to support simultaneous development and testing of different concepts for underwater robotics by independent researchers. A general control-software framework enables common access to all subsystems and avoids the duplication of basic robotic functionality jointly required by all projects. Additionally, the new autonomous technologies enabled by the results of individual research are mutually compatible and can be easily integrated into a single robotic system. Examples of new technologies demonstrated on the *OTTER* underwater robot include control from a real-time vision-sensing system, coordinated arm/vehicle control, and control from 3D graphical user interfaces.

**Keywords:** Autonomous underwater robot, task-level control, control/software architecture, vision-sensing system, coordinated arm/vehicle control, 3D graphical user interfaces

### **1. Introduction**

The Stanford University Aerospace Robotics Laboratory (ARL) and the Monterey Bay Aquarium Research Institute (MBARI) are working together in a joint program on technologies that will enable underwater robotic systems to become more useful tools to both science and industry. In support of this program, several robotic submersibles have been built as testbed platforms on which to develop, test, and demonstrate new autonomous technologies. These testbed systems needed to be rapidly reconfigurable and accessible to individ-

ual researchers working simultaneously on independent topics.

This article discusses the design and development of the latest testbed called *OTTER*, an Ocean Technology Testbed for Engineering Research. First, some background in autonomous underwater robots and in the technologies being investigated using *OTTER* are given. Then, a discussion of the philosophy driving *OTTER*'s design follows. Next, *OTTER* is described in detail at the system and component levels. Finally, a few of the autonomous technologies that have been experimentally demonstrated to date with *OTTER* are discussed.



Fig. 1. The *OTTER* Underwater Robot.

### 1.1. Background

Many autonomous underwater vehicles (AUVs) have been constructed by research institutions around the world, but they are far from being common tools for daily use by science and industry in subsea operations. Fundamental issues in intelligence, control, and power remain open research, as well as engineering, topics. Earlier efforts focused on large AUVs such as the DARPA (ARPA) vehicle (Brancart, 1994) which can carry large payloads and dive to great depths. Smaller AUVs have been designed to cruise for long distances like the *Odyssey* (Bellingham *et al.*, 1992) or to stay submerged for long durations like *ABE* (Yoerger *et al.*, 1991). Others have been built to support various research objectives such as control from sonars, cooperating multiple vehicles or adaptive non-linear vehicle controllers (e.g. *Phoenix* (Healey *et al.*, 1994), *EAVE III* (Blidberg *et al.*, 1990), and *ODIN* (Choi *et al.*, 1995)). There are also notable international AUV research programs in Europe and Japan as well (e.g. *VORTEX* (Perrier *et al.*, 1993), *MARIUS* (Pascoal, 1994),

*Aqua Explorer 1000* (Kato *et al.*, 1993), and *Twin Burger* (Fujii *et al.*, 1993)).

Working underwater robots with intervention and station-keeping capabilities have largely been limited to remotely-operated vehicles (ROVs) like MBARI's *Ventana*. These robots accomplish their missions under tight manual control provided by skilled and experienced pilots communicating through a high-bandwidth tether. By adding on-board intelligence and automatic guidance to human-controlled underwater robots, the level at which a person interacts with these systems can be elevated. Since higher-level communication reduces the requirements on data transfer rates and has greater tolerance for time delay, bandwidth-limited acoustic modems (Catipovic *et al.*, 1994) can replace tethers as links to control the remote systems. Eliminating the tether not only removes a major source of operating costs, but also increases the portability and maneuverability of the overall system. Maintaining human intelligence in the control loop preserves much of functionality of ROVs for untethered systems. The technologies being addressed by the research in the MBARI/ARL program apply to this class of semi-autonomous underwater robots (SAUVs).

### 1.2. Motivation

The *OTTER* testbed has been designed and built to support a variety of research in technologies for small underwater robots (see Figure 1). Individuals working in each research area share *OTTER* as a common platform to test and develop their concepts experimentally. *OTTER*'s design parameters and specifications have evolved directly from the combined requirements of the different technical areas under investigation. These areas are briefly described below to help motivate the decisions made in the design and construction of *OTTER*.

**Control from Local Sensors** Underwater robots working in the open ocean may not have access to global sensors to find their inertial state or the states of the environment in which they are operating. However, local sensors carried on board are often important sources of robot-relative state information.

Table 1. General Characteristics of OTTER.

Dimension	2.1 m long, 1 m wide, 0.5 m high
Displacement	150 kg dry (approx. 450 kg wet)
Drive Thrusters	746 W (1 HP) Brushless VR motors, 26 cm propellers, 5:1 gearing
Maneuvering Thrusters	186 W (1/4 HP) Brushless VR motors, 15 cm propellers, 3.5:1 gearing
Arm motor	373 W (1/2 HP) Brushless VR motor, 60:1 harmonic drive
Maximum Depth	1000 m estimated
Maximum Speed	4 kts estimated
Power	Nickel-Cadmium batteries, 750 W hrs on board
Control	Full 6 DOF (roll and pitch statically stable)

One local sensor used extensively by humans, but to a limited extent so far by robots, is vision. Recent advances in vision-processing hardware make it feasible to reduce the vast amount of data, inherent in video signals, to usable information at rates fast enough for feedback control. Along with new algorithms to distill relative-state information from raw vision-hardware outputs, it is now possible for underwater robots to servo directly off of signals from on-board video cameras.

**“Smart” Sensors and Actuators** The revolutions in the computer and electronics industry have not been fully utilized by underwater robots. In particular, Very Large-Scale Integrated circuit (VLSI) technology can be used to miniaturize digital electronic components and reduce power consumption. Entire microcontroller boards are now small enough to be packaged with each sensor and actuator to create intelligent components.

“Smart-sensor” systems with their own microprocessors can return filtered measurements of the sensed states as well as estimates on their rates of change. “Smart actuators” provide command access at higher levels of control and are able to respond to desired force, velocity, or position commands instead of the usual current or voltage commands. Microprocessors help segment subsystems into individual black boxes such that internal changes within each box does not severely impact the rest of the system. Smart components help separate functionality from implementation, allowing the sensor or actuator expert to work independently from the systems integrator.

**Hydrodynamic Interactions** Submersible hydrodynamics has been widely studied, and there exists proven methods to develop good models useful for the control of streamlined underwater

vehicles. Many researchers have also worked on the dynamics and control of robotic manipulators.

Currently, most robotic arms mounted on underwater vehicles are teleoperated by humans, reducing the need to understand and quantify the complex hydrodynamic interactions between the arm and vehicle for control. However, for small, untethered submersibles, the impact of unknown hydrodynamic forces acting on the arm in response to its motion can degrade the performance of the combined arm/vehicle system. Models of the hydrodynamic interactions between the arm and vehicle need to be developed and integrated with an automatic control system to produce a robot that truly coordinates arm control with vehicle control.

**Human Interfaces** For semi-autonomous robots, the interface through which a human interacts with the remote mechanism is an important component of the system. As much information as possible about the current state of the system must be made available to the operator in order to make goal-oriented decisions.

Since the communication link for untethered, underwater robots is, at best, an acoustic modem, issues in presenting low-bandwidth and time-delayed data become important. Humans can interpret visual information more easily than textural information, so computer-generated graphics can be used to provide the operator with an environment in “estimated reality” that is augmented with stored information accumulated during a mission or known a priori. Possible user interfaces for remote systems span from simple 2D-graphics solutions to full immersion in 3D virtual reality.

**Robotic Programming** Programming AUVs, or autonomous vehicles in general, is the sub-

Table 2. Summary of *OTTER* Subsystems.<sup>1</sup>

Subsystem	Component
Structure	Aluminum pressure housings Welded steel frame Free-flooding fiberglass shell Removable 1 DOF manipulator Foam and fibreglassed redwood for flotation
Actuation	8 Ducted, oil-filled thrusters 2 DOF, Pan/tilt device for underwater cameras Arm motor with harmonic drive
Power	Nickel-Cadmium batteries DC-DC power converters Relay bank
Sensing	Stereo CCD cameras Real-time image processing Fluxgate compass Two-axis inclinometer Systron Donner MotionPak (3 linear accelerometers/3 rate gyros) Pressure transducer (depth sensor) Sonic High-Accuracy Ranging and Positioning System (SHARPS) Manipulator torque and acceleration sensors Leak detectors and battery monitors
Computation—Hardware	On-board real-time VME computer On-board ADC and DIO boards Off-board real-time VME computer Off-board UNIX workstations
Computation—Software	VxWorks <sup>TM</sup> real-time operating system ControlShell <sup>TM</sup> software framework Virtual-reality user interface X-Window user interface
Communications	Ethernet or serial link through tether NDDS <sup>TM</sup> communications protocol Serial daisy-chain motor communications network

ject of study for many people working on control/software architectures. Custom software tends to work only with very specific hardware. For robots, hardware consists of both the computer system that is executing the software and the devices that the software is controlling (e.g. motors, sensors, etc.). Even when written modularly, it is almost impossible to exchange software between different hardware platforms without arduous modification.

Different programming languages may be more appropriate at different levels of control. High-level programming languages are evolving away from the standard, text-based, coding methods to the paradigm of visual programming where one creates graphical diagrams of information flowing into and out of reusable processing blocks.

Distributed computation with many processors interconnected by a communications network will inevitably become the backbone of many computer architectures for underwater robots. The challenge of managing the added complexity is the burden that balances the increase in computing power provided by distributed architectures. What has nominally been a research topic in computer-software engineering is now another aspect of robotic design and development.

### 1.3. Design Philosophy and Principles

To support the various research areas of interest described above, MBARI and ARL have built *OTTER*, a multi-purpose, submersible research

testbed. The overriding goal governing its hardware and software design was to create a flexible framework upon which students can integrate, develop and test new technologies in a single robotic system. In addition, a goal of the project was to be compatible with systems which had been developed at Stanford University for space robotics and with a new ROV, *Tiberon*, which was being developed at MBARI.

*OTTER*'s main function as a laboratory test platform, operated by students in a test-tank environment, is reflected in its design. While some components may withstand depths of up to several thousand feet, *OTTER* was neither designed nor constructed for operations in the harsh environment of the open ocean or to be used in actual science and industrial applications. Without the constraint of building ocean-worthy, operational hardware, rapid prototyping of new subsystems results in a faster development cycle to keep pace with the quick evolution of technology inherent in the research environment. In a sense, the design and development of *OTTER* never ceases. Each new student may add advanced capabilities and hardware, and during the process, disable and remove older ones belonging to prior students and research projects.

Thus, the key attributes that will allow *OTTER* to survive and evolve through generations of students are the generality, flexibility, and extensibility of its fundamental software/hardware architecture. Genericness of the foundation is key to future expansibility. In a developmental testbed, this is especially important since it is often impossible to predict what may or may not be required by future users. While certain subsystems may be custom built, e.g. sensor arrays and manipulator packages, the framework that interconnects them should be generic. This philosophy is reflected in the choice of the VME bus for the hardware backplane of the computer system and of VxWorks for the real-time operating system. Because both VxWorks and VME are popular in the robotics and other industries, many products and much support exist and are constantly created in response to new technological advances in areas such as microprocessor chips. With these choices, *OTTER*'s core architecture is not in danger of becoming ob-

solete, and new technologies developed outside the domain of the underwater robotics program can be quickly and easily integrated.

In terms of the overall physical configuration of the system, *OTTER* was designed to be handleable by two people which is desirable from a logistical viewpoint. Many of the science applications driving the technology research require stationary in situ operations necessitating precision position control and the ability to maneuver at low speeds. The general mission scenario conceived for this class of SAUVs consists of a transit phase from a home base to the site of operations where the SAUV would perform some specific task under human direction, and finally, a transit home. Thus, the need to control six degrees-of-freedom during station-keeping operations drove the chosen thruster configuration, while the general geometry of *OTTER* and the shape of the hydrodynamic fairing was selected to optimize motion along one degree-of-freedom during transit.

What have not been major concerns in *OTTER*'s design, so far, are power storage and consumption. Increased energy density of high-technology batteries and other forms of on-board energy production such as fuel cells have not been a focus of the research. Advances in that arena are being pushed by other industries like electric vehicles which have invested heavily into their development. The decision was made to use the most powerful, easily accessible, off-the-shelf computers and software systems available, even if they used more energy than optimal. This is because computers and electronics are advancing rapidly. It is assumed that by the time the technologies developed on *OTTER* are ready to be moved to operational systems, high, powerful levels of computational capability will be available in low power.

## 2. *OTTER* Testbed

The *OTTER* testbed was designed and built at MBARI's facility in Moss Landing, California. *OTTER* is the third-generation testbed designed by the program. Many design decisions on *OTTER* resulted from lessons learned during the operation of the first two platforms. Even though *OTTER* mainly operates in a test-tank environment, *OTTER*'s major structural components—

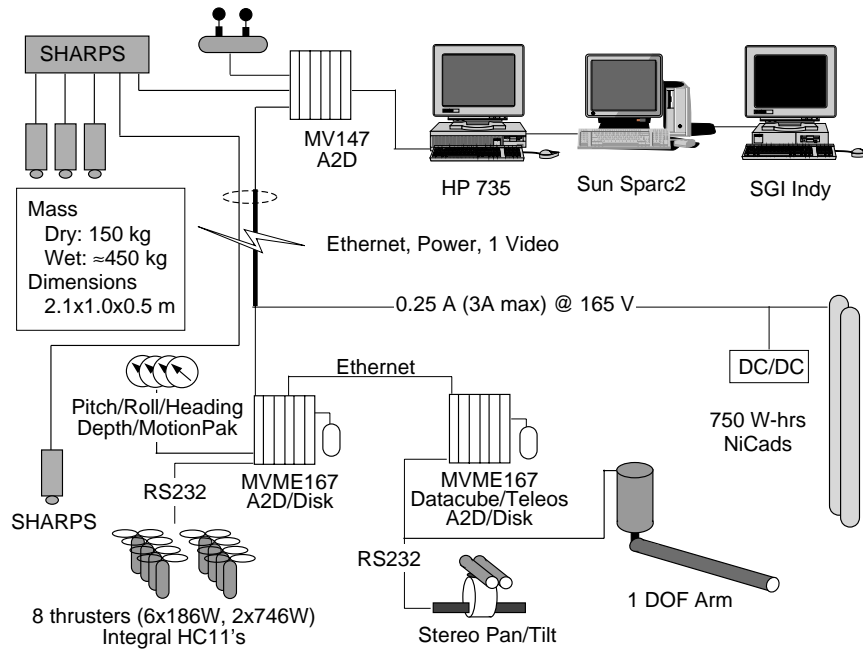


Fig. 2. Components of the *OTTER* Robotic System.

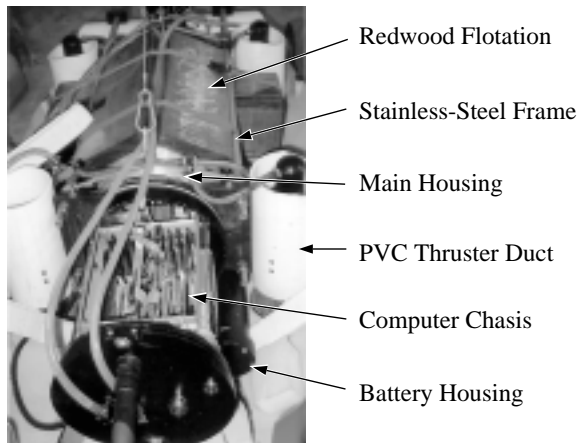


Fig. 3. *OTTER* without top shell.

e.g. housings for electronics, batteries, thrusters— have been designed and tested to withstand depths up to 1000 meters. In addition to providing power to trickle charge on-board batteries, a tether is used to facilitate development and testing. Although an acoustic modem is scheduled to be installed on *OTTER*, the maturity level of high-speed acoustic communications is such that an acoustic modem can not be used reliably without further development by experts in the tech-



Fig. 4. Removable rear propulsion unit.

nology. However, the high-level control of *OTTER* was designed to accommodate a limited-bandwidth and time-delayed link. *OTTER* has been operated with a serial link running at 38.4 kBaud (state-of-the-art acoustic modems have operated at rates up to 20 kBaud). A test tank at

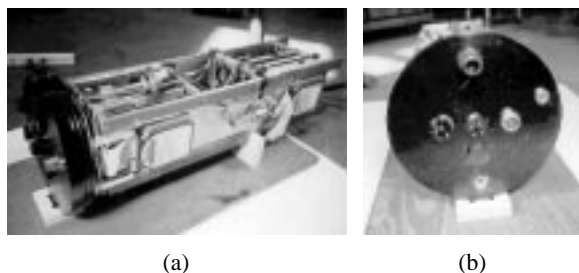


Fig. 5. The computer chassis attached to the endcap is shown in (a). (b) shows the endcap with tether, junction box, and video connectors.

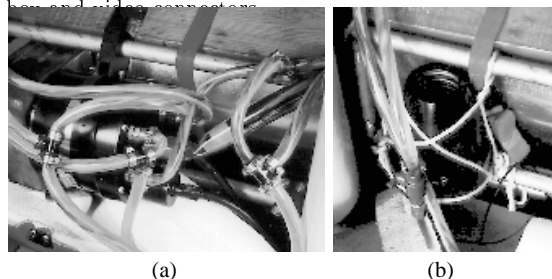


Fig. 6. (a) shows the oil-filled, electrical junction box and (b) shows the oil pressurization and compensation device.

MBARI's Moss Landing facility is used for testing and operation of the *OTTER* system. The tank is 11 meters in diameter and 3 meters high and provides a benign environment to demonstrate and develop many of the advance technologies being studied.

Table 1 contains some general system parameters and characteristics, and the major subsystems that comprise *OTTER* are listed in Table 2. Figure 2 shows a schematic of the subsystems.

## 2.1. Structure

*OTTER*'s main structural element is a cylindrical pressure housing 0.36 m in diameter by 1.25 m long made of aluminum (Figure 3). The main housing holds the on-board computer systems and many of the sensors. Two smaller 0.12 m in diameter by 1.25 m long aluminum cylinders hold the NiCad batteries and are mounted underneath the main housing. Surrounding the pressure housings are eight ducted thrusters attached to a welded stainless-steel frame which runs the length of the vehicle. The ducts are made from PVC piping. Covering the vehicle is a two-piece fiberglass shell that provides a fairing for improved hydrodynam-

ics. Fiberglass-covered redwood pieces and foam mounted within the shell are used to trim the vehicle's buoyancy. Because the shell is free flooding, the effective mass and inertia of the robot underwater is significantly higher than in air.

The main-drive thrusters and aft-lateral thruster form a removable unit that provides access to the main housing (Figure 4). The main housing consists of a hollow cylinder with two endcaps. Two 6U VME cages are mounted in-line to form a computer chassis attached to the rear endcap (Figure 5). The entire endcap/chassis unit can be removed for benchtop operation. The tether attaches to the endcap through an underwater connector providing power connections for battery charging, two twisted pairs used for serial communications or video, and a BNC for video or Ethernet. Connectors for two underwater video cameras also penetrate through the rear endcap.

Analog and digital signals and power lines of various voltages also pass from main housing to an electrical "junction" or breakout box mounted externally (Figure 6). Signal and power wires are connected to each individual actuator as well as to the batteries. The wire bundle connecting the box to the main housing and all of the actuator wire bundles are encased in Tygon tubing filled with mineral oil. An oil-compensation system pressurizes all of the tubes, actuators, and the junction box. The non-conductive oil acts to relieve the external pressure on the wire bundles and motor housings while underwater. Even though operating with a pressurized-oil system can be rather messy, the benefits of such a system should there be a leak is reflected by its widespread use on operational underwater systems.

## 2.2. Actuation

In performing basic research in "smart" actuators, the MBARI/ARL program has designed and constructed an actuator with custom motor-control electronics integrated with a brushless DC motor. All of the actuators on *OTTER* (thrusters, pan/tilt, manipulator) use the same motor electronics with motors of different sizes. The motors themselves are based on variable-reluctance (VR) technology. These low-cost motors are brushless motors that do not have any permanent mag-

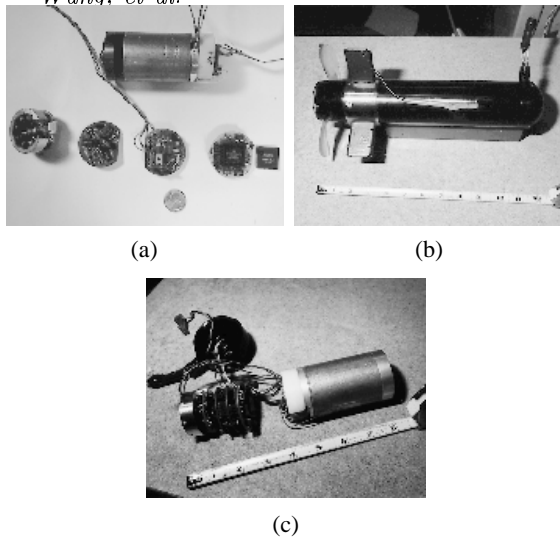


Fig. 7. Components of the motor electronics stack. In (a), four individual electronics boards and a motor make up an actuator. (b) shows an assembled motorstack. A completed thruster is shown in (c). The same motor electronics are also used for the pan/tilt and manipulator actuators.

nets and provide a high power-to-weight and high torque-to-inertia ratios.

**Integrated Microprocessor Stack** The actuator modules integrate the motor, the motor control, and the communication logic into one unit. Four components form each motor electronics stack (Figure 7): a microprocessor/digital electronics board, an analog-amplifier electronics board, a DC-DC power-converter board, and a power-amplifier board. All of the electronics have been pressure-rated to 1000 meters because the electronics operate immersed in mineral oil for pressure compensation of the housing. Since optical encoders do not work well in oil, a custom capacitive encoder is used to determine motor shaft position to commutate the motor as well as to provide position and velocity information for closed-loop control. Pressure compensation also allows a low-pressure seal to be used, resulting in much lower seal friction at the shaft.

A Motorola HC11 microprocessor and an Altera Application Specific Integrated Circuit (ASIC) are used in combination to control the VR motors. Altera's ASIC chip is an example of VLSI user-programmable chip technology newly available to robotic researchers. All of the logic nec-

essary to decode position and velocity of the motor and for commutation is implemented using the ASIC. ASICs are programmable logic devices that can implement the logic from hundreds of standard integrated circuits in one user-programmable chip. One can submit a digital-circuit diagram, boolean-logic equations, basic TTL circuits created with a graphical interface, or even a matrix of desired outputs for given inputs as specifications to a compiler that automatically creates the logic burned into an ASIC.

For control of the actuator, the Altera is used to convert the encoder signals from the motor into position and velocity values that are passed onto the HC11. The HC11 runs various closed-loop controllers and sends torque commands back to the Altera which does the actual commutation. The HC11 is programmed to control the motors in one of three modes: open-loop torque, closed-loop velocity, and closed-loop position. A fourth controller based on force feedback has been used in experiments on a test stand but has not yet been implemented on *OTTER*.

The version of the HC11 that is used has only 2 kBytes of on-chip EEPROM to store its programs, and thus, software efficiency is extremely important. Therefore a simple, but effective, custom serial protocol was created for the control computer to communicate to each thruster at 31.25 kBaud. Three daisy-chained, full-duplex serial loops are used. Two of the loops each connect four thruster motors with the control computer, and the pan/tilt and arm motors are linked by the third. The control computer acts as the loop master and sends commands sequentially to each motor on the loop. Every HC11, and therefore actuator, has a unique identifying number burned into EEPROM that can be considered its "address" on the loop. As soon as an actuator has determined that a packet is being sent on the serial loop for itself, the actuator immediately sends back a packet of data containing information such as motor velocity and position. Thus, an actuator only talks on the serial line when it is spoken to. In the custom communications protocol, the first byte of each packet is the id of the destination actuator. With this scheme, an actuator can start its reply to the loop master delayed only by the length of time it takes to transmit one char-



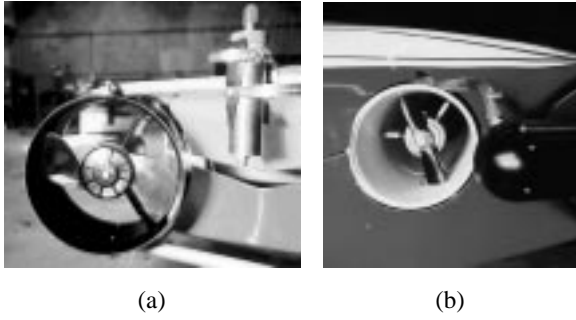


Fig. 8. The port-drive thruster is shown in (a), and the aft-lateral thruster is shown in (b). The propeller mounted in (a) is modified from a trolling motor propeller, while the one in (b) was originally designed for model aircraft. Also, the SHARPS transceiver is visible in (a).

acter. Packets can be of arbitrary length, but are typically only six bytes long.

Theoretically, 390 six-byte packets can be sent by the master of the loop per second at 31.25 kBaud with a two-character time delay to delineate packets. The thrusters are controlled at 20 Hz, pan/tilt at 30 Hz and manipulator motor at 230 Hz with the increasing sample rates indicative of the increasing bandwidth of the dynamics of each system.

**Thrusters** Propulsion is provided to the vehicle by two drive thrusters and six maneuvering thrusters (Figure 8). The drive thrusters consist of 746 W (1 HP) VR motors with 5:1 planetary-gear reducers. The maneuvering thrusters use smaller 186 W (1/4 HP) VR motors with 3.5:1 planetary-gear reducers. The drive thrusters are mounted in the horizontal plane at the rear of the vehicle and are intended to provide the high-powered propulsion required to move rapidly in the forward direction during transit operations. Also mounted in the horizontal plane are the fore-lateral and aft-lateral thrusters, which together with the drive thrusters are used to control the yaw ( $\Psi$ ) and the  $x, y$  positions of the vehicle (see Figure 9). The remaining four maneuvering thrusters are mounted in the vertical direction with one on each “corner” of the vehicle. These thrusters provide control of the pitch ( $\Theta$ ), roll ( $\Phi$ ), and vertical ( $z$ ) degrees-of-freedom of the vehicle.

Experiments have shown good correlation between thrust produced and propeller angular-velocity squared in steady conditions (see Figure 10). Taking advantage of this relationship, a calibration curve for each thruster describing

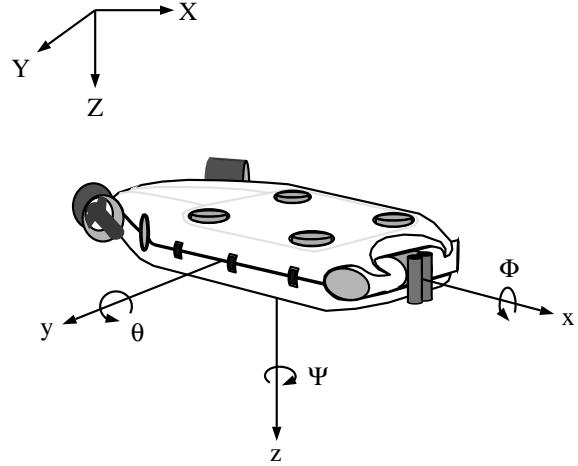


Fig. 9. The *OTTER* Coordinate System. ( $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ ) are inertial linear coordinates with the  $\mathbf{X}$  vector pointing North and  $\mathbf{Z}$  pointing down. ( $\Psi, \Theta, \Phi$ ) are the  $\mathbf{ZYX}$  Euler angles that represent *OTTER*'s orientation with respect to the global system. ( $\mathbf{x}, \mathbf{y}, \mathbf{z}$ ) are body-fixed coordinates.

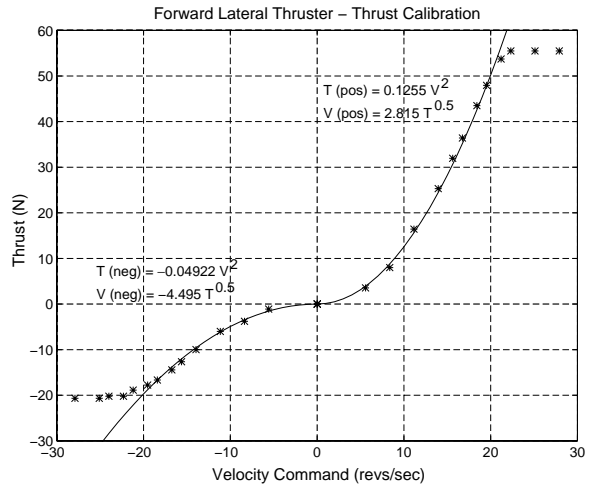


Fig. 10. Typical Thrust vs Velocity Curve. Note the high asymmetry of the thrust curve for the positive and negative directions and the low saturation levels. For each thruster, a calibration map is used to convert commanded thrust to commanded velocities.

the relationship between thrust and the square of the output-shaft rotation rate is generated. These thrust/velocity relationships are used in the vehicle control to map commanded thrusts to commanded motor velocities sent to the thrusters. Currently, the resolution of the velocity command is eight bits.



Fig. 11. Underwater cameras on a pan-tilt mechanism.

Experiments have also been conducted in using duct-mounted strain gages to provide feedback for local, closed-loop force control. Although preliminary results show great promise for this method of control, a new mechanical design will be required to provide a robust and clean force signal for feedback control.

*OTTER* uses two types of commercial, off-the-shelf propellers: one designed for small, marine trolling motors, and the other for model aircraft (see Figure 8). These propellers have been shaved to fit within the thruster ducts. Unfortunately, these propellers are highly asymmetric and optimized to provide thrust in a single direction. Although taking the thrust asymmetry into account through a mapping function helps the performance of the control system, the combination of these propellers and the thruster motors is suboptimal both in terms of efficiency and maximum producible thrust.

**Pan/Tilt** The pan/tilt device controls the direction of the optical axes of *OTTER*'s stereo camera pair. The unique mechanism uses two 186 W (1/4 HP) VR motors that cooperatively drive—through 60:1 harmonic-drive reducers—a single cable that controls both the pan and tilt degrees-of-freedom (Figure 11). When the motors turn in the same direction, tilt motion is achieved. Differentially driven at the same speed, the motors produce pure pan motion. The pan/tilt motors can be controlled in a closed-loop position mode by commanding 16-bit positions or in a velocity mode.



Fig. 12. Single-link manipulator mounted on *OTTER*.

The pan/tilt device is integrated as part of the front-lateral thruster duct to keep the vehicle as compact as possible. Because of this, the tilt degree-of-freedom is accomplished by rotating the camera around the duct. This leads to more of a “nod” effect than a true tilt, resulting in a translation of the image planes when tilting the cameras. Because the stereo arrangement offsets each camera from the pan axis, the image planes also translate slightly while panning. These translations must be taken into account by the vision system (Section 2.4).

**Manipulator** To support coordinated-arm/vehicle-control experiments, a single-link arm can be mounted at the front of *OTTER* (Figure 12). The arm is 7.1 centimeters in diameter and 1 meter long. This arm, being used for initial experiments, has the same general physical characteristics as a prototype, all-electric, 4-DOF manipulator that has been designed for *OTTER* for use in future research in automated manipulation. The single-link arm is mounted at the fore-port corner of the vehicle frame and tilted down at an angle of 60 degrees from the horizontal plane. This configuration was chosen because it places the arm in the region most likely to be the workspace of the 4-DOF manipulator. With the arm mounted in this way, all of the vehicle's degrees-of-freedom are affected by the hydrodynamic forces generated by the arm as it moves. The actuator used by the arm is a 373 W (1/2 HP) VR motor that drives the arm through a 60:1 harmonic-drive reducer.

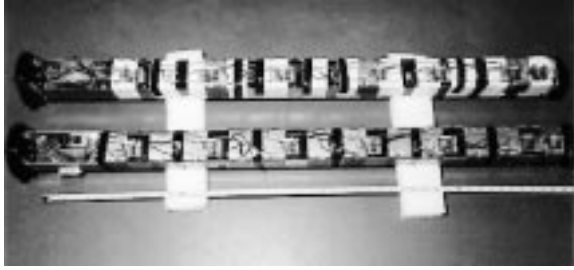


Fig. 13. On-board NiCad batteries. 60 D-sized NiCad in each cannister provide a total of 750 W hrs of on-board power.

Measurements of output-shaft torque and lift moment at the arm hub are sensed using a specially-designed beam element connecting the output shaft to the base of the arm. This beam element is outfitted with two full strain-gage bridges, one sensitive to moments about the axis of the drive shaft, and the other sensitive to bending moments about an axis through the hub and normal to both the arm and drive-shaft axes. These sensors measure the output-shaft torque due to in-line hydrodynamic forces and the lift moment due to transverse forces generated from the shedding of vortices.

2.3. Power

Two battery cannisters, each holding six packs of ten D-sized nickel cadmium (NiCad) cells, are mounted on-board OTTER providing 750 watt hours of power at 160 volts DC (see Figure 13). Each of the 120 D-cells stores 5 amp hrs at 1.33 volts. Mechanical relays are switched on to engage the battery system, and under-voltage and over-temperature sensors built into each battery pack are monitored by the control system.

The tether is used to trickle charge the NiCads at 165 volts and 0.25 amps. In the current configuration, the tether is also directly connected to the DC-DC converters powering the main computer cage. The main computer controls the battery relays as well as a bank of solid-state relays that provide power to the rest of the vehicle. When the main computer is booted, the control program can switch on and off individual relays through digital I/O. This configuration provides a large margin of

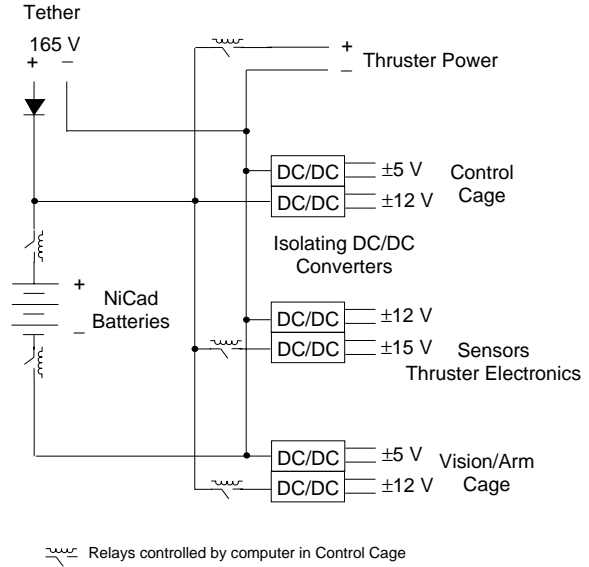


Fig. 14. Schematic diagram of OTTER's power distribution system. In the tethered configuration, power from the tether directly drives the control computer cage. Once the control computer is booted, the relays that engage the batteries and send power to the other computer cage and sensor systems can be switched on. If tether power should fail, all relays switch off and the batteries disengage automatically.

safety during tethered experiments. When power to the tether is cut off, the main computer shuts off automatically causing all of the relays to open and disconnecting the batteries. To remove the tether, a slight modification will be made to the design so that the batteries can be engaged by an external switch.

Internally, Vicor high-efficiency, solid-state DC-DC converters are used to change 160 volts provided by the batteries/tether to ± 5, 12, and 15 volts for on-board computers and electronics (see Figure 14). In addition to converting voltages, the Vicors electrically isolate different subsystems from one another. The increased safety provided by the electrical isolation of subsystems cannot be understated. All signals through the tether are isolated either through isolation power supplies, video isolation transformers, or Ethernet transceivers. By isolating all sources of power from each other and from earth ground, it takes two faults or shorts to create a potentially dangerous current path. Ground-fault detectors are used to monitor the integrity of the electrical system.

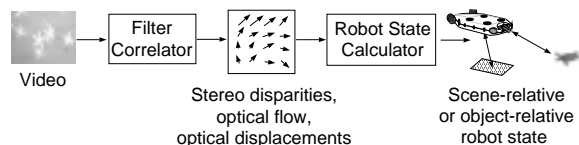


Fig. 15. Schematic diagram of *OTTER*'s real-time vision system.

#### 2.4. Real-Time Vision System

In the real-time vision system, video signals are digitized and stored in frame buffers. Buffered images are then filtered and correlated. The vision hardware provides correlations between sections of any mix of two live images or stored images at 30 Hz. From the correlations, algorithms have been developed to find positions and velocities of objects or the background relative to the cameras (Marks, 1995). Figure 15 shows the information flow for the vision-sensing system.

The real-time vision-sensing system physically consists of two CCD video cameras, a mechanical pan/tilt unit, several image-processing boards, a real-time computer, and an UNIX workstation. The UNIX workstation acts as an user interface to the vision system. The image-processing boards and real-time computer reside on board *OTTER* in a VME card cage and perform all of the vision-sensing computation.

#### Cameras and Video Digitization Hardware

Two Pulnix 840-N, low-light CCD cameras are housed in pressure vessels mounted in a stereo configuration on the pan/tilt device at the front of *OTTER* (see Section 2.2 and Figure 11). These cameras are gen-locked and produce RS-170 video streams (black-and-white, interlaced, 30-Hz frame rate). The horizontal field-of-view of each camera is 52 degrees in air and is reduced to approximately 36 degrees during underwater operations due to the change in the refraction index.

Image digitization is accomplished using a Datcube DIGICOLOR color digitizer and display device. The DIGICOLOR is used to digitize simultaneously two RS-170 (NTSC) monochrome, analog-video streams. The resulting, digital-image frames are 512 by 484 pixels with an 8-bit intensity level. Also, the DIGICOLOR is used to display processed video with the vision-processing results

overlayed on top of the image. Images produced by the DIGICOLOR can be sent up the tether to the operator via a twisted pair.

Digital-image frames are stored using a Datcube FRAMESTORE board. The FRAMESTORE is capable of simultaneously storing and outputting three digital-image frames. The FRAMESTORE is used during station keeping and mosaicking to hold the reference images against which live video is correlated (see Section 3.1).

**Special Image Processing Hardware** The two image-processing boards, known as the PRISM3 system and made by Teleos Research, Inc., are used for filtering and correlation. The first board, the Convolver, filters two digital-video streams at frame rate (30 Hz) using a signum of the Laplacian-of-Gaussian filter (Marks, 1995), (Marks *et al.*, 1994a).

The second board, the Correlator, computes correlations between filtered-image frames. The Correlator is extremely fast—36 correlations of a 32-pixel square area require less than 100 microseconds. Correlations from the Correlator are processed using a Motorola MVME-167, a 68040-based, single-board computer. The disparity or change in position of identical features between two images are used in algorithms to calculate positions or velocities of objects or the displacement of the cameras, and therefore of the robot, from a previous location.

When the two images are from a stereo pair, the disparity between the images is a measure of range to what the cameras see. If one image is held static, then the disparity between that image and a live image corresponds to the displacement of the robot from where the static image was recorded. The disparity between two images from a single camera taken successively in time relates to relative-velocity information.

This vision hardware is another example where VLSI technology is enabling something only previous achievable in non-realtime software simulations.

**Graphical Interface to Vision System** Key parameters that affect vision-sensing-system functionality and performance are viewed and can be modified at run time using a graphical interface

running on an HP-735 UNIX workstation. Parameters that can be changed include the vision-sensing mode, Laplacian-of-Gaussian filter width, correlation thresholds, display characteristics, and tracking and mosaicking controls.

## 2.5. Motion, Position and Attitude Sensors

In order to sense and control the position and attitude of the vehicle, a variety of devices are used. Integration of redundant information provided by the various sensors to produce a best estimate of vehicle state is an on-going topic of research.

**Heading (Yaw)** A KVH-Industries, ROV-1000 fluxgate compass provides measurements of heading. The compass has an accuracy of 1 degree and a resolution of 0.1 degrees in a magnetically-clean environment. However, the heading signal is corrupted slightly (about  $\pm 2$  degrees of noise) by electromagnetic noise from the power electronics and cooling-fan motors within the main pressure housing. The ROV 1000 is calibrated by running a built-in program that maps the local magnetic environment. The heading is calculated internally at 10 Hz and can be provided digitally or through an analog signal.

The analog signal wraps around between maximum and minimum output voltage when the compass passes through 0 degrees heading. Because this signal is continuous and it takes some time for the wrap-around to complete, the wrap-around can be interpreted by an automatic control system as a rapid change in heading and pulse the thrusters controlling yaw erroneously. However, "de-glitching" logic is built into the software that filters the yaw state to provide a smooth transition through 0 degrees.

**Pitch/Roll** The pitch and roll of *OTTER* are measured using a two-axis inclinometer made by Spectron Systems Technology, Inc. This unit provides analog measurements of pitch and roll with a range of  $\pm 45$  degrees from horizontal and a resolution of 0.01 degree. The inclinometers produce clean pitch and roll angles, but introduce their own dynamics at higher frequencies ( $> 5$  Hz).

## Linear Acceleration and Angular Rates

The MotionPak sensor package from the Systron Donner Inertial Division of BEI Electronics Company uses three quartz-flexure accelerometers and three solid-state gyros to provide inertial measurements of linear acceleration and angular rate. The linear accelerations are not being used at this time, although future research may examine the possibility of fusing these measurements with those from the vision-sensing system or a long-baseline, acoustic position system to provide linear-positioning information at high bandwidths. The angular rates from the MotionPak are used to provide damping when controlling attitude.

The accelerometers have a resolution of 1 micro-g with a range of  $\pm 5$  g's. The rate gyros have a resolution of 0.002 deg/sec and a range of  $\pm 100$  deg/sec. However, the biases of the signals are highly temperature dependent and thus drift over time. The accelerometers are more adversely affected than the rate gyros. Rate-gyro calibration is done periodically during experiments by measuring the biases of the signals with the vehicle motionless.

**Depth** *OTTER*'s depth can be inferred from hydrostatic-pressure measurements from a pressure transducer. The pressure measured is directly proportional to the depth below the surface. This value from sensor is used instead of the depth value from the SHARPS position-measurement system (see below) for two reasons. First, because of its simplicity, the pressure transducer is more reliable. Also, the sensor is more accurate, especially when the vehicle is operating close to the plane of the SHARPS transponders. Since the depth sensor is an analog device, measurements can be obtained at the full, 100 Hz sample rate of the vehicle controller rather than the 3 Hz update rate available from SHARPS.

**Position** The inertial  $X, Y$  position of *OTTER* is measured using SHARPS (Sonic High Accuracy Ranging and Positioning System) made by the Marquest Group, Inc.<sup>2</sup> SHARPS is a long-baseline acoustic system which uses three inertially-fixed acoustic transducers to find the position ( $X, Y, Z$ ) of a fourth. Three transducers are mounted around the perimeter of the test tank, and a fourth

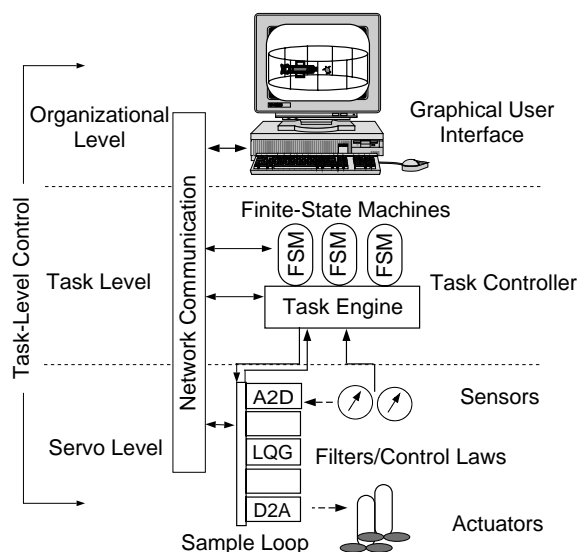


Fig. 16. OTTER Control Architecture.

is mounted on *OTTER*. SHARPS can be deployed in open waters over a 100 meter range.

The SHARPS transducers are connected to a IBM PC compatible through separate umbilicals.<sup>3</sup> The position of the transducer on *OTTER* is calculated by the PC and sent via a serial line to a real-time computer system that broadcasts the position over Ethernet to the on-board computers and off-board graphical user interfaces. SHARPS provides position updates at about 3 Hz with two centimeters of accuracy. However, because the  $Z$  signal from the pressure sensor is continuous and more robust, only the  $X$  and  $Y$  values from the SHARPS system are used by the control system.

## 2.6. Control Architecture

*OTTER*'s control architecture allows it to be controlled at the *task level* using high-level commands such as "go there", "track that object", etc., which the robot is able to execute automatically. The control architecture called "Task-Level Control" (TLC) developed for *OTTER* is based on similar architectures used in ARL's other experimental robotic platforms (Schneider and Cannon, 1993), (Ullman et al., 1993).

This architecture, discussed in greater detail in (Wang et al., 1993), divides the control of a remote system into a three-layer hierarchy (Fig-

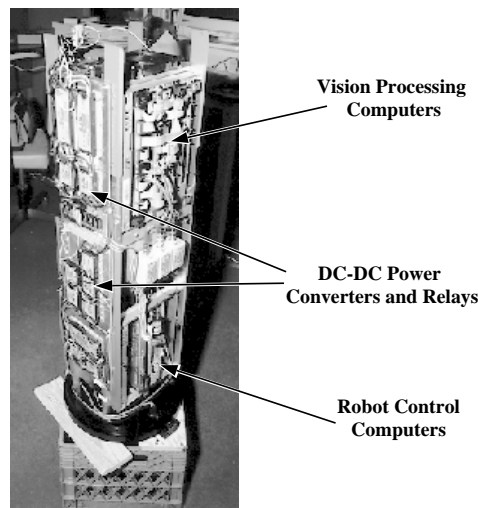


Fig. 17. On-board VME cages.

ure 16) similar in organization to the one described by Saridis (Saridis, 1979). At the lowest layer or *servo level*, classical control theory is employed to design control laws that take in sensor signals and output motor control at a fixed sample rate. The encoding of logical and sequential actions that define a specific task takes place at the middle layer or *task level*. Tasks are "programmed" using finite-state machines (FSMs). FSMs generally modify the control laws and signal-flow paths of the servo layer to achieve different robot behaviors or control modes.

The top level or *organizational level* is where the human or other higher, intelligent agents sequence and monitor task execution to achieve mission goals. Typically, the entities at the top-layer must interpret processed sensor data to form a judgment of the progress of the robot during mission execution and adjust robot actions to accommodate in-situ anomalies as they occur. In the *OTTER* control architecture, the human interacts with the remote system at the organizational level via graphical displays and virtual-reality interfaces (see Section 2.7).

**Computer Hardware** The control of *OTTER* is distributed among a network of on-board, real-time computers. The user interacts with the on-board computers through a variety of UNIX workstations including Suns, HPs, and SGIs. An off-board, real-time computer serves both as a

communications relay between the robot and the workstations and as a port for analog manual inputs such as joystick controllers.

Two independent, on-board VME card cages are used for robot control and vision processing (see Figure 17). Each cage has a Motorola MVME-167 (68040, 25MHz) single-board computer with attached hard disk (200MB). In addition, *OTTER*'s control VME cage has an analog/digital board with 16-bit analog-to-digital converters and digital IO. A serial-interface board for actuator communications and an analog-filter board for sensor signals also reside on the same bus. The processor in the second, on-board VME cage controls both the manipulator and pan/tilt motors as well as runs the vision-processing algorithms mentioned in Section 2.4. The computers in the two cages are connected via Ethernet.

A third real-time processor is used in an off-board VME cage. During operations when communication with the *OTTER* is performed serially, this processor acts as the gateway between topside and on-board computers running Serial Line Internet Protocol (SLIP) at 38.4 kBaud. This processor, a MVME 147 (68030, 20MHz), also uses a 12-bit analog-to-digital converter board to read joystick inputs for sending manual commands to *OTTER* when it is running in manual mode. However, the main function of this processor is to take serial data from the SHARPS PC (see Section 2.5) and broadcast vehicle-position information over the network.

A Sun Sparc 2, an HP 735, and a SGI Indy comprise the suite of UNIX workstations that are used both for developing software and for running the graphical interfaces through which users command *OTTER*. Real-time code can be developed on any of machines using cross compilers. Although 3D, graphical user interfaces (GUIs) exist for all of the workstations, the implementation of each GUI is quite different due to the incompatibility of the accelerated graphics hardware/software provided by the different computer manufacturers (see Section 2.7).

By using a variety of computer architectures, the program remains compatible with the development environments used by various research organizations including the ones forming this program. The interoperability of the software produced by

this program makes the results more readily transferable to other research groups. There are also reasons in choosing to use one type of platform over another. For example, advanced graphics research is typically done with Silicon Graphics computers because they are optimized for that facility.

**Control Software** To implement the Task-Level Control architecture described earlier, a set of software tools have been assembled to create a TLC system. If one views the definition of the control/software architecture as a blueprint of the robot control system, then the toolset serves not only as the instruments with which system will be created, but also as the framework that connects and supports the different software components which make up the system.

Developers using a set of well-designed development tools benefit from the corporate knowledge that the tools can provide. When a person, perhaps a new member of the research team, wants to learn and possibly create or modify an existing robot control system, the typical thing to do is to read and interpret massive amounts of code that someone else has already written. Alternatively, with a set of software tools, a person first learns how to use the tools and how they are applied to create software components that conform to the blueprint (control architecture). In the beginning, the details of components themselves, e.g. a Butterworth filter or a PD control law, are not as important as how to create the components and how to interconnect them. Later, new components are added that are usually a part of the results of a person's fundamental research (e.g. adaptive, neural-net controllers, vision-processing blocks, etc.).

By using a common software framework, independent researchers working on the same robotic platform share the components that implement the basic robotic functions. When a hardware and/or software change is made on the fundamental configuration, this change is automatically propagated to all of the users of the robot just by modifying the shared components. In addition, the new capabilities added by individuals through their research are easily merged together to create more complex applications since the software components that implement each capability already fit together in the same framework.

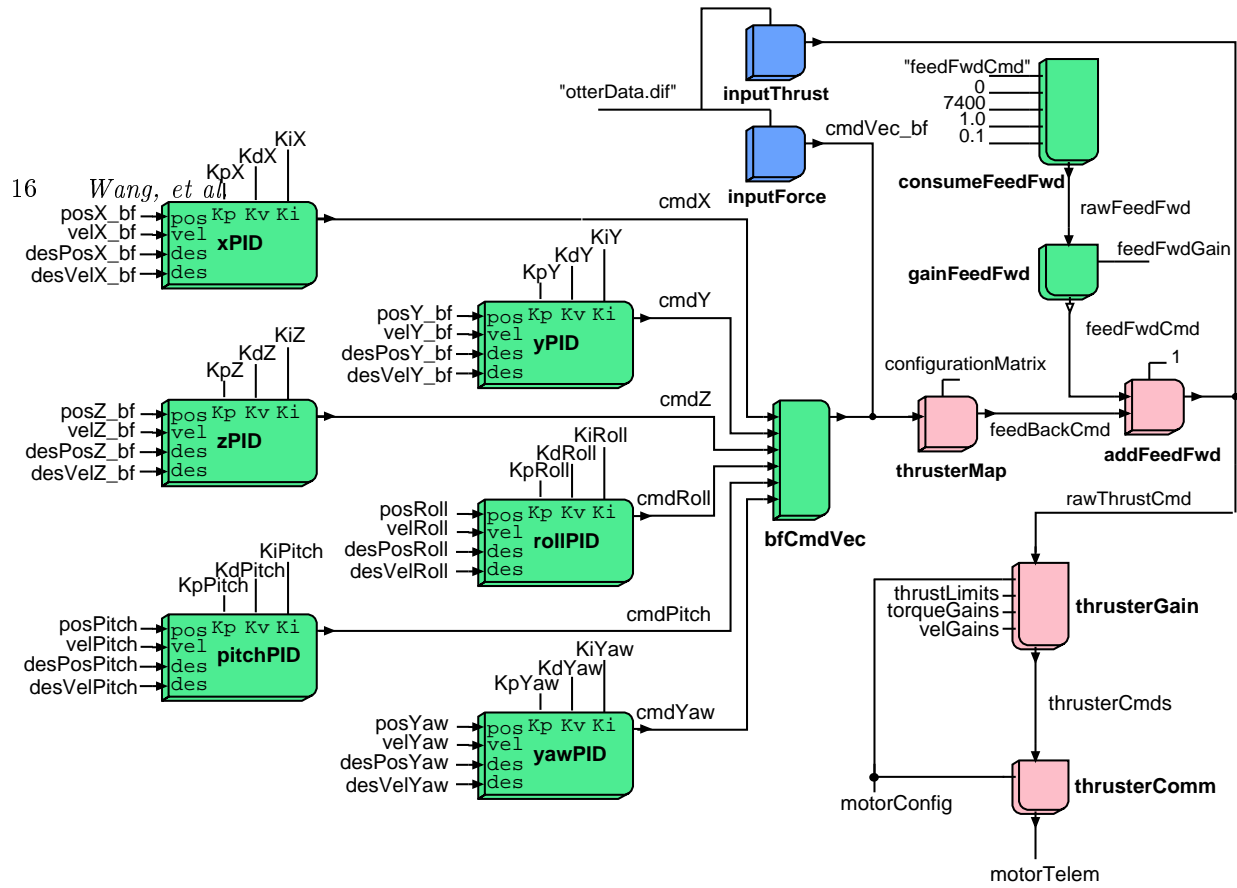


Fig. 18. Data flow diagram for *OTTER* thruster controllers.

**Real-time Operating System** The most basic piece of the toolset is a real-time operating system. Simple robotic systems may be controlled using only a single program that implements all of the logic, sensor processing, safety monitoring, control laws, and motor drivers. However, as the systems become more complex, currently executing processes running on a single or multiple processors need to communicate with each other, synchronize their execution, and share data. Device drivers are required to separate what services a specific piece of hardware provides (e.g. serial data inputs and outputs, etc.) from how that piece of hardware does it (e.g. control registers, interrupt service routines, etc.). Since a physical system is to be controlled, real-time constraints must be satisfied, and the control software must be efficient and executed deterministically. A real-time operating system provides many of these services that robot control programs need.

*OTTER* uses VxWorks, a popular multi-tasking, real-time operating system within the general robotics community, as its foundation upon which low-level control loops and high-level

coordination logic are built. VxWorks has a good network interface compatible with UNIX workstations, and thus, can support distributed control applications like the *OTTER* controller. Using VxWorks, the user can literally “login” to the on-board computer systems. Programs can be loaded off of local, SCSI hard disks or off of a workstation’s disk that has been NFS mounted.

**ControlShell Framework** On top of VxWorks, a large set of associated tools, based on past work done by students at ARL, is used as the basic framework to create TLC controllers for *OTTER*. Real-Time Innovations, Inc., is a company that was created to continue the development of software tools first written to support Ph.D. research (RTI, 1995). *ControlShell* is a Computer-Aided Software Environment (CASE) for real-time system-software development (Schneider *et al.*, 1994), (Schneider *et al.*, 1991). It provides graphical tools to help create and link software components in an object-oriented framework for real-time applications. ControlShell also provides data man-



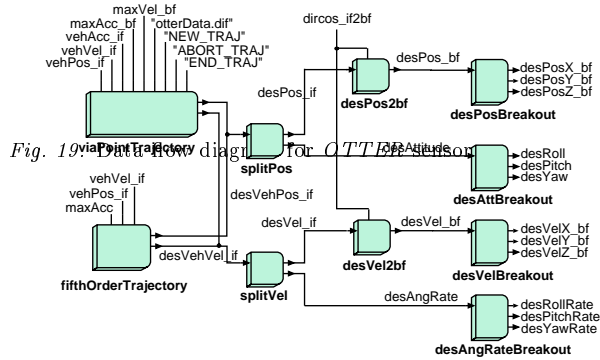
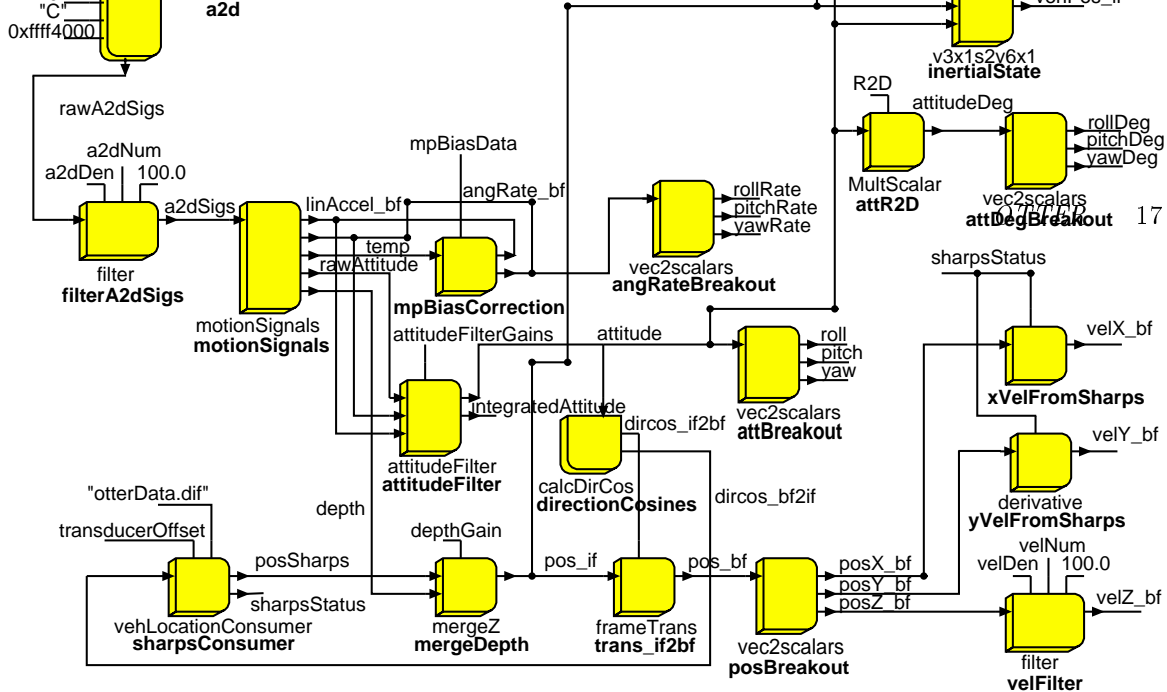


Fig. 20. Data flow diagram for OTTER's trajectory generator.

agement and flow control during execution of the applications.

With ControlShell, software *components* are developed in C++ that represent the most basic modules of a control system. Once the inputs and outputs of a component are defined, ControlShell automatically generates skeleton code that becomes a reusable component when the programmer fills in the working details. Examples of components are digital filters, analog-to-digital device drivers, PID and LQR controllers, estimators, and basically, any piece of a control-system block dia-

gram that is executed periodically in a sampled-data digital system.

With a graphical editor, one connects the inputs and outputs of each component to define how the data flows through the components (see Figures 18, 19, and 20). Then various *configurations* are defined by grouping different components together. These configurations are, in essence, different control modes for the system. Later, when the system is running, one can switch between predefined configurations and thus change the control mode of the system. Several configurations

Table 3. Various control configurations for *OTTER*.

Configuration	Description
SENSORS_ONLY	Sensor processing only, no control to thrusters.
MANUAL_FORCE	Manual force input converted to individual thruster commands.
MANUAL_THRUST	Manual input to each thruster.
ATTITUDE_DEPTH	Automatic control of attitude and depth only.
POSITION	Position control using SHARPS and attitude sensors.
COORDINATED_ARM_VEHICLE	Control using feedforward from arm controller and position feedback.
VISION_STATIONKEEP	Position control using vision feedback.
VISION_MOSAIC	Control to create video mosaics using vision feedback.
VISION_TRACK	Control to track objects using vision feedback.

defined for *OTTER* are shown in Table 3. Independent researchers sharing the system create custom configurations from the existing components and from ones that they have added themselves through their work.

An important aspect of control system development and “tuning” is the ability to change control parameters such as gains or locations of poles and zeros while the robot is running to observe the difference in performance. ControlShell provides a menu system that gives immediate access to all of the variables in the control system. Using the menu to modify control system parameters, one can quickly tune the values for maximum performance.

**Finite-State Machines** The coordination level of the TLC control architecture (see Figure 16) is programmed using *finite-state machines* (FSMs) created graphically using another ControlShell graphical editor. An FSM is a visual language that relates *transitions* between *states* of a system to *events* that trigger the transitions. High-level tasks can be encoded as FSMs by identifying discrete, intermediate states that lead to the completion of the task and by defining events that trigger the transitions from state to state. In essence, FSMs are used by the robot as maps to follow to reach a goal based on what state the robot is in and what event has just occurred. As there are may be many possible paths to reach point B from point A, the FSM can encode many different sequences of actions to complete the same task.

Events can be generated in different ways. A program can monitor the progress of the robot doing a task and generate events when different stages of task completion are reached. Safety monitors can run in the background generating emer-

gency events such as ones that indicate low batteries or water leakage, or events may be directly generated by an user causing the robot to change control modes.

A diagram of the main FSM controlling *OTTER* appears in Figure 21. Additional tasks can be added to the main FSM by adding FSM “functions” or “subroutines” like the *RetrieveObject* FSM subroutine shown in the figure. Each FSM subroutine is itself a conglomeration of states and transitions encoding a procedure to accomplish a subgoal.

**Real-Time Communications** Concurrently running programs are fundamental elements that make up a single application in control of a complex robotic system. These programs may reside on the same computer, on different computers connected by a local network, or even across computers located miles apart. Sharing data about the system state, sending commands from one program to another, and synchronizing actions of independent modules require the elements to communicate with each other.

RTI’s *Network Data Delivery Service* (NDDS) (Pardo-Castellote and Schneider, 1994) is used for communications between the distributed programs running on both real-time and UNIX computers that make up the robotic controller network. NDDS makes it easy to share data and event information among systems in a computer network. It uses a subscription-based paradigm to send packets between producers and consumers of data. Processes running on *OTTER* produce information (e.g. vehicle position and attitude) that is consumed by the graphical user interfaces (GUIs) running on the workstations. On

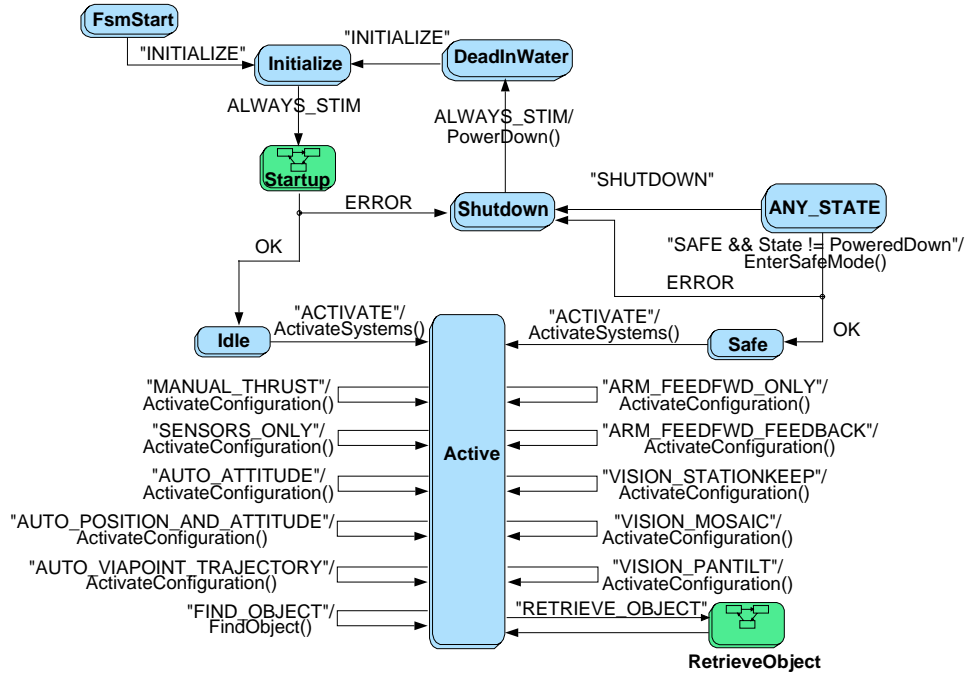


Fig. 21. OTTER’s FSM controller. Note that the icons for “states” like *Startup* actually represent multiple states and can be considered a finite-state machine subroutine.

the other hand, the GUIs produce user commands that are consumed by *OTTER*. NDDS supports multiple consumers and producers of the same data, thus allowing the seamless integration of robot dynamic simulators and multiple command interfaces within the same control network. Because in NDDS neither consumers nor producers of data care about from whom they receive data or to whom they send data, it is easy to connect a new program module to the robotic application without modifying the existing modules.

**OTTER Low-Level Control System** Diagrams of the components and data flow that make up the core of the *OTTER* control system appear in Figures 18, 19, and 20 (not shown are the diagrams which integrate the components that implement vehicle control from vision nor the ones that implement coordinated arm/vehicle control).

The desired and actual state in the body-fixed coordinate system (see Figure 9) are fed into individual PID controllers for each degree-of-freedom. *OTTER* uses aircraft/spacecraft body-fixed coordinates, i.e. *x* out the nose, *y* off to the right, and *z* pointing down and **ZYX** Euler angles ( $\Psi, \Theta, \Phi$ )

to represent orientation. In the inertial coordinate system, **X** points North, **Y** points East, and **Z** is positive downward. Thus when submerged, depth is positive.

From the controllers, a vector of commanded body forces and torques are mapped into commanded forces for each of the eight thrusters. The eight thruster forces are then adjusted for the characteristics of each thruster and transformed into velocity commands.

Sensor data, read from analog-to-digital converters or from serial ports, are converted from raw signals such as voltages into processed data such as angular rates or depth and digitally filtered to remove high-frequency noise. Signal vectors, e.g. *attitude*, are separated to be passed to the rest of the control system as individual values, e.g. roll, pitch, yaw angles. Sensor-processing components also create derived signals such as the direction-cosine matrix representing vehicle orientation and rates of signals for which there is no direct rate measurement such as position.

Under automatic position control, *OTTER* can be moved by sending it a *MOVE* command with a desired, final inertial state or with a set of inertial waypoints through which to pass. Using the com-

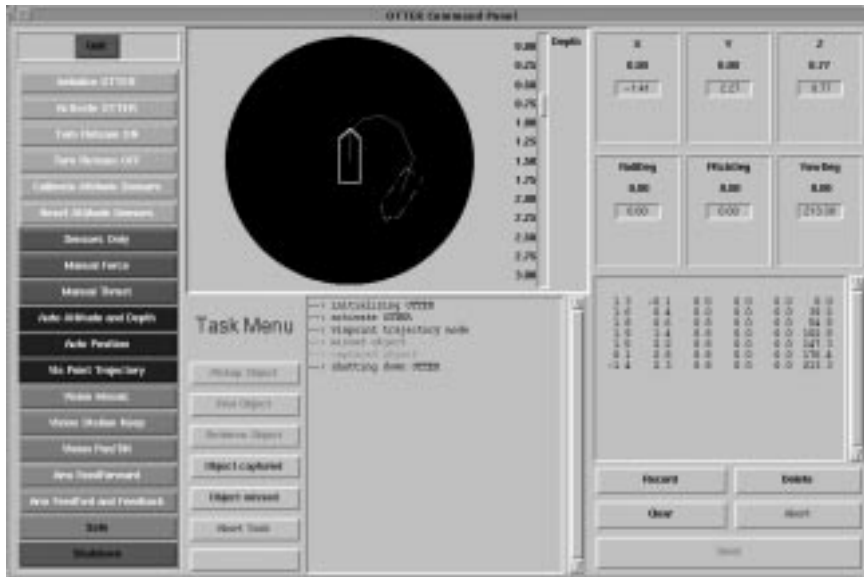


Fig. 22. A 2D graphical interface based on Tcl/Tk. The dotted vehicle is the ghost icon of *OTTER* that a user would drag to command motions.

ponents shown in Figure 20, either a smooth 5th-order trajectory or a splined viapoint trajectory is generated beginning at the current position. At each time step after the trajectory has begun, desired inertial angles and positions for that time are transformed into the body-fixed frame and fed into the PID controllers. As a result, *OTTER* can be commanded to move smoothly and in a coordinated fashion.

## 2.7. Graphical User Interface

The user interface is an important component in the TLC architecture. With telemetry from the robot arriving in limited quantities and perhaps greatly delayed, techniques must be developed that intelligently combine the telemetry with stored information known a priori about the robot and its operating environment, presenting to the operator a best estimate of the current status of the remote system. The user can then draw conclusions, make plans, and issue task commands based on what he or she sees in this presentation of “estimated-reality”.

Several user interfaces have been created for *OTTER* on many different types of computers. Some offer full 3D, virtual environments and others only give an iconic representation of vehicle

state. Depending on the speed of the computer, especially in displaying complex graphics, it was often found that a simple interface could be more intuitive to use than a 3D interface. While the 3D graphics provided more realistic scenes of *OTTER* in its work setting, the lagging response to user commands of the graphics running on slower computers hindered the intuitive feel extended by 3D environments. However, when a faster computer with dedicated graphics hardware is available to run the same 3D interface, the natural advantages of the realism offered by computer-generated graphics is restored.

The earlier, simple 2D graphical interfaces were created in C using basic X-Window graphics and Motif<sup>4</sup> widgets. GUI builders—software applications that provide tools to create, place and otherwise manipulate widgets such as buttons or sliders, etc.—helped to layout the interface and generated the basic code automatically. Then additional code to connect the graphical interface and *OTTER* was added using the NDDS communications library (see Section 2.6).

Currently, these types of interfaces are being generated using the Tcl/Tk scripting language and graphics toolkit that have been developed recently to make writing GUI applications easier (Ousterhout, 1994). Because Tcl/Tk uses

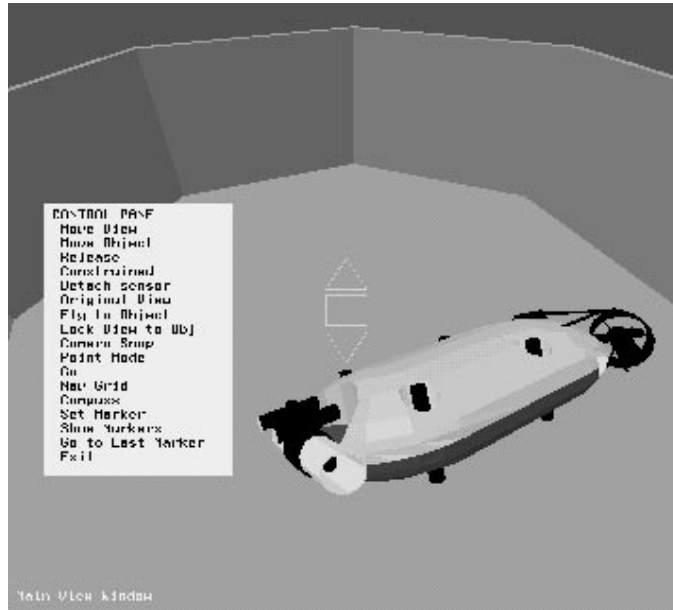


Fig. 23. The VEVI 3D Virtual Environment. *OTTER* is shown in the test tank with a menu of interface commands.

scripts stored in plain text files to configure the GUI, after one creates the basic functions required to connect to the underwater robot, changes to the GUI can be made simply by modifying the script. No recompiling of code is required, thus making changes easier and faster to implement. Figure 22 shows a 2D interface written using Tcl/Tk.

Several 3D virtual environments also have been developed for controlling *OTTER*. For the HP and Sun workstations, libraries based on the PHIGS graphics standard were used to generate 3D models embedded in an X-Window environment. While PHIGS is adequate to produce realistic models of robots, it does not support some advanced graphics features such as texture mapping or ray tracing. For research into developing an user interface that is able to integrate information known a priori about the operating environment with data accumulated during mission execution, the team uses the advanced graphics capability provided by Silicon Graphics computer systems. For example, if the benthic bathymetry of where *OTTER* is operating has been mapped already during a previous visit, then as *OTTER* wanders about creating mosaics of the surroundings, the image mosaics can be returned to the user and integrated with the virtual environment by using texture mapping.

Currently, a sophisticated interface based on NASA Ames Research Center's Virtual-Environment Vehicle Interface (VEVI) (Fong, 1993) running on a SGI Indy is used to control *OTTER* (Fleischer *et al.*, 1995). The VEVI-based interface is a full virtual-reality interface that can be displayed on head-mounted units or on a special monitor to be viewed in stereo using LCD glasses (see Figure 23). Versions of this interface have been used to control NASA's *TROV* experiments in Antarctica, the *Dante* walking robot in Alaska, and the *Marsokhod*, Russia's Mars land-rover prototype, during trials in Hawaii. Being compatible with NASA allows the team to run experiments cooperatively with NASA in controlling remote, semi-autonomous systems like *OTTER* across long distances to accomplish real science missions. NASA hopes to understand the ramifications of using autonomous robots for scientific exploration of space by first investigating their use in analogs of these missions underwater.

Other types of telemetry information can also be displayed by user interfaces. When the vision-sensing system is operated in an object-tracking mode and it locks onto an object, a graphical icon representing the object is placed in the interface, indicating to the user in an intuitive manner where

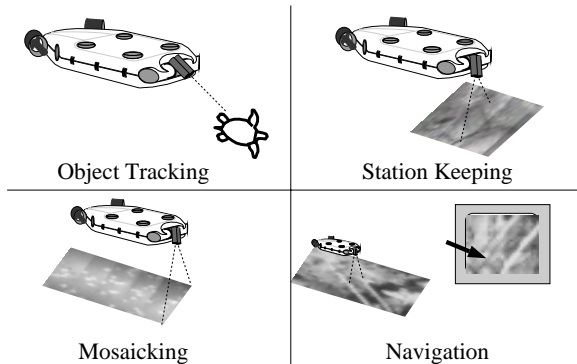


Fig. 24. Experimental tasks using vision feedback.

the object is located relative to *OTTER*. Conceivably, additional sensor information can be stored with the icon such as actual video footage of the object.

Commanding *OTTER* to move to new positions is accomplished by dragging a “ghost” image of the robot in the interface using a mouse or a 6-DOF input device called a spaceball. As the ghost is moved about, waypoints for the path can be recorded as well as the final desired position. These waypoints are sent down to *OTTER* where an on-board trajectory generator creates a smooth path passing through the waypoints for the vehicle to follow.

### 3. Advanced Underwater Technologies

Using the *OTTER* underwater robot as a test platform, several experiments have been completed to demonstrate the autonomous-control technologies being investigated by the MBARI/ARL program. In the following section, some results of the control from local vision and coordinated arm-vehicle control experiments will be presented. More details of the work presented in this section can be found in the dissertations of the two ARL students responsible for the research (Marks, 1995), (McLain, 1995).

#### 3.1. Visual Sensing and Control

A large effort has been invested in studying optical-vision systems as local sensors for the automatic control of underwater

robots (Marks, 1995), (Marks *et al.*, 1994a). Using the vision-sensing system described in Section 2.4, a number of vision-dependent autonomous tasks (see Figure 24) have been experimentally demonstrated on *OTTER*.

**Object Tracking** Scientists using MBARI’s ROV *Ventana* often observe marine animals for hours at a time while they move or drift about in the Monterey Bay. Piloting the ROV is task that requires high concentration and takes exceptional skill gained through long-term experience to do proficiently. Often two or more pilots must trade off controlling the ROV every hour or so to rest. By enabling the capability of autonomous tracking of these creatures by the ROV, the workload on the pilots can be significantly decreased.

Using the stereo cameras with the vision-sensing system, *OTTER* has demonstrated the ability to track arbitrary objects underwater, maintaining both relative position and orientation (Marks *et al.*, 1993), (Marks *et al.*, 1992), (Marks *et al.*, 1991). Objects are segmented from the background by using stereo disparity to calculate their relative positions from the vehicle. Higher object-tracking performance is achieved by combining vehicle control with control of a high-speed camera pan/tilt system (Wang *et al.*, 1992). The high bandwidth and limited range of the pan/tilt system complements the unlimited range but slower dynamics of the vehicle.

**Station Keeping** Scientists also like to view static scenes with the ROV for long periods for both biological and geological research. During this time, the pilot must fight constantly changing disturbances from unseen currents or the tether to keep the same scene in view of the cameras. Thus, automatic station keeping is another desirable ability for underwater robots. In most operating environments, there is no global sensor system producing the inertial position of the robot for use in a feedback controller such as an acoustic long-baseline system. At best, underwater robots can use their on-board attitude and pressure sensors to hold orientation and depth automatically.

However, the live images recorded by the on-board cameras contain sufficient information to determine camera, and therefore robot, displacement from a desired static scene. With the vision-

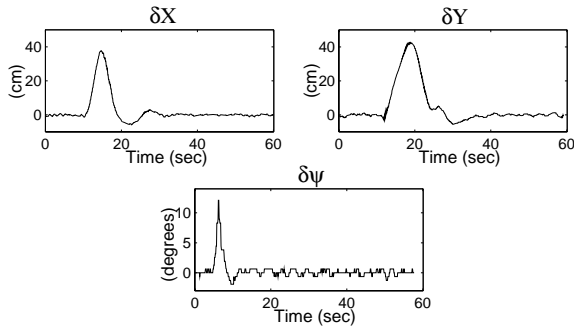


Fig. 25. *OTTER* station keeping using vision. Plots of the  $X$ ,  $Y$ , and  $\Psi$  motions of *OTTER* while station keeping using the vision sensor. Large disturbances are imparted to each degree of freedom which are then rejected by the control system.

sensing system, displacement information can be extracted and fed into the underwater-robot controller for automatic station keeping. This is accomplished by storing an image of the area relative to where the robot is to hold station. Then, a number of algorithms have been developed to compare the live image with the stored image to determine robot’s positional error in the 3 linear degrees-of-freedom and a single rotation error in the plane of the image. These errors are used by the automatic control system to provide drift-free station keeping (Marks *et al.*, 1994b). In Figure 25, the reaction of *OTTER* to large disturbances while using vision for station keeping is shown.

**Creating Video Mosaics** Another application of the vision system is to generate mosaics of the ocean floor automatically from individual frames of video (Marks *et al.*, 1995). Large-area mosaics can already be produced from video or still shots through post-processing techniques. However, with a vision system controlling the robot to move to the next spot where a frame should be recorded, mosaics can be generated “on-the-fly” and displayed to user in real-time. This mosaic is not only helpful to the pilots in determining the position of the robot relative to its surroundings, but also can aid the scientist to decide what to do or where to go next to accomplish the mission. A mosaic generated by *OTTER* of the bottom of a test tank (dressed with rocks and pipes) using 18 individual images can be seen in Figure 26. To cre-

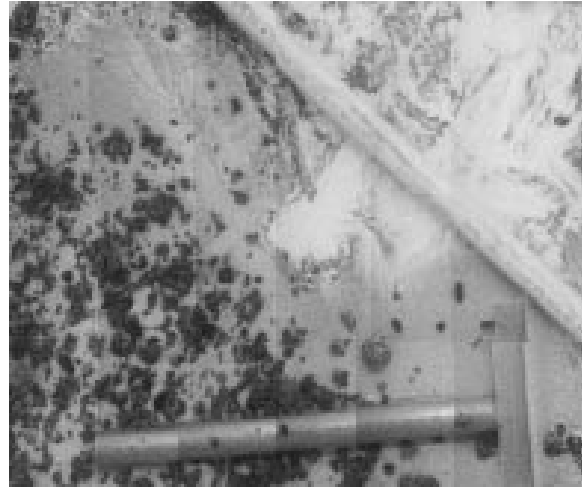


Fig. 26. 3x6 mosaic created automatically by *OTTER*.

ate this mosaic, *OTTER* was driven by the vision-sensing system back and forth across the tank to each position where an image was required.

### 3.2. Coordinated Arm-Vehicle Control

Robots without intervention capabilities are mainly sensing and transportation platforms. However, once a manipulator is added to the robot, entirely new capabilities are opened for investigation. Current underwater manipulators are typically teleoperated using “hand-in-glove” methods with the robot fixed relative to the object that they are manipulating, e.g. sitting on the ocean floor.

Ultimately, the scientist or pilot should be able to point at objects presented on the live video image or in a computer-generated user interface and have the arm/vehicle system automatically pick up or otherwise manipulate them. To do this, the robot must actively manage the hydrodynamic and hydrostatic interactions between the manipulator(s) and vehicle in order to be able to place its end effector precisely. These interactions are complex with both kinematic and dynamic parameters varying as the manipulator changes configuration. Understanding these interactions is the first step in creating a system that can automatically pick up artifacts off of the ocean floor.

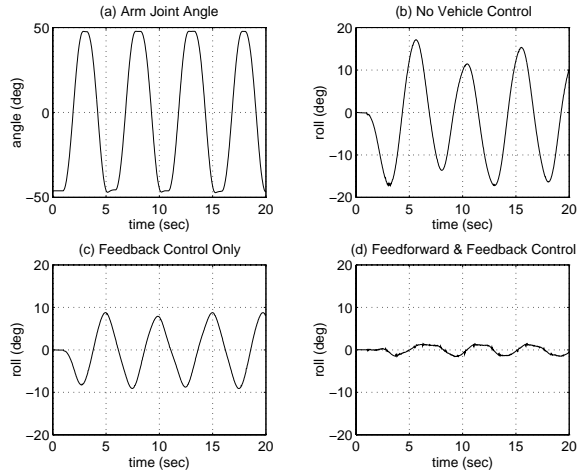


Fig. 28. Performance of coordinated arm/vehicle control. As the arm is rotated 90 degrees several times, plots of *OTTER*'s roll motion is shown. In (b), the vehicle's thrusters are turned off. In (c), the vehicle is using only feedback control on attitude and position. The dramatic improvement in performance using feedforward control based on a model of arm hydrodynamics and feedback control can be seen in (d).

Fig. 27. Manipulator hydrodynamic test facility at Stanford ARL.

By accurately predicting the forces on the vehicle caused by moving the manipulator underwater, counter forces can be applied by the thrusters to keep the vehicle motionless during manipulation. A test-tank facility at Stanford is used to study experimentally the hydrodynamics of a simple single-link underwater arm (Figure 27). From basic fluid theory and using both visualization of flow patterns and direct measurements of forces and torques acting on the arm, a hydrodynamic model of the single-link arm has been developed (McLain, 1995). This model is used to help control the arm/vehicle system in a coordinated manner during manipulator motion.

The severity of the hydrodynamic interaction of a manipulator with the vehicle and its correction through the use of new algorithms that coordinates the control of the arm and vehicle have been demonstrated with *OTTER*. For these experiments, a single-link, one DOF arm is mounted on *OTTER* (see Section 2.2 and Figure 12).

As the arm is swung through 90 degrees multiple times, the effect of the arm motion on vehicle roll and yaw degrees of freedom is significant. Without any control on the vehicle, the generated forces can cause the vehicle to roll over  $\pm 18$  degrees and to drift over 15 degrees in yaw. Even with the vehicle under feedback control to maintain both position and orientation, the disturbance of the arm produced over  $\pm 9$  degrees of error in roll and 8 degrees of drift in yaw.

However, by using the hydrodynamic model to calculate and send feedforward force/torques to the vehicle's thrusters that counter act the hydrodynamic disturbance force/torques produced by the arm, the overall performance of the system is significantly improved. In Figure 28, the *OTTER*'s roll motion while moving the arm is shown with no control, with only vehicle feedback control, and with the addition of feedforward control taking hydrodynamic disturbance forces into account. Although not shown, the addition of feedforward control improved the performance of all of the other degrees-of-freedom as well.

#### 4. Conclusions

The joint MBARI/ARL research program is investigating various technologies for small, semi-autonomous underwater robots. The *OTTER* underwater robot is used as a research platform to develop low-level algorithms and high-level strategies for control as well as to test new, innovative actuators and sensors. *OTTER*'s design capitalizes on the emergence of VLSI ASIC technology to build small, low-cost, low-power, custom digital electronics for intelligent sensors and actua-



tors. *OTTER* uses a generic framework for the control/software architecture that provides general access to all robotic subsystems. Independent researchers are able to share common software components while developing custom code for individual research projects. The ease of reconfigurability of both hardware and software components are key reasons for *OTTER*'s success as an engineering-research testbed.

Examples of the technologies demonstrated experimentally on *OTTER* are direct robotic control with a sophisticated vision-sensing system and coordinated arm/vehicle control using a model of manipulator hydrodynamics. In addition, control of *OTTER* through computer-generated, graphical user interfaces shows the semi-autonomous abilities of the *OTTER* system to carry out human high-level commands by the on-board intelligent controller.

### Acknowledgements

The authors would like to acknowledge the contribution of all of the students, engineers, and researchers of the MBARI/ARL team, and especially Dick Burton, *OTTER*'s mechanical designer. Funding for the research performed in the joint MBARI/ARL program has been provided by the Monterey Bay Aquarium Research Institute, the National Aeronautics and Space Administration, the Office of Naval Research, and the National Science Foundation.

### Notes

1. UNIX is a registered trademark of AT&T. VxWorks is a trademark of Wind River Systems, Inc. ControlShell and NDDS are trademarks of Real Time Innovations, Inc. X Window System is a trademark of the Massachusetts Institute of Technology.
2. Marquest Group, Inc., has become Imetrix, Inc., and now markets SNAP as a SHARPS compatible system.
3. A tetherless version of SHARPS is available.
4. Motif is a trademark of Open Software Foundation, Inc.

### References

J.G. Bellingham, C.A. Goudey, T.R. Consi, and C. Chyrssostomidis. A small long-rang autonomous vehicle for deep ocean exploration. In *Proceedings of the Second International Offshore and Polar Conference*, pages

- 461–467. International Society of Offshore and Polar Engineers, June 1992.
- D. R. Blidberg, S. Chappell, J. Jalbert, R. Turner, G. Sedor, and P. Eaton. The EAVE AUV program at the marine systems engineering laboratory. In *The First Workshop on: Mobile Robots for Subsea Environments*, pages 33–42. International Advanced Robotics Programme, October 1990.
- C. P. Brancart. The evolution of the AUV: The DARPA/ARPA UUV program. In *Proceedings of IARP Second Workshop on Mobile Robots for Subsea Environments*, pages 181–192, Monterey, May 1994. IARP.
- J. Catipovic, D. Brady, and S. Etchemendy. Development of underwater acoustic modems and networks. *Oceanography*, 6(3), 1994.
- S.K. Choi, J. Yuh, and G.Y. Takashige. Development of the omni-directional intelligent navigator. *IEEE Robotics and Automation Magazine*, 2(1):44–53, March 1995.
- S. D. Fleischer, S. M. Rock, and M. J. Lee. Underwater Vehicle Control from a Virtual Environment Interface. In *Proceedings of the Symposium on Interactive 3D Graphics*, Monterey, CA, April 1995. SIGGRAPH.
- T. W. Fong. A Computational Architecture for Semi-autonomous Robotic Vehicles. In *Proceedings of AIAA Computing in Aerospace 9 Conference*, San Diego, CA, October 1993. AIAA.
- T. Fujii, T. Ura, and U. Kuroda. Mission execution experiments with a newly developed AUV the “Twin Burger”. In *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Technology*, pages 92–105. University of New Hampshire, Marine Systems Engineering Laboratory, September 1993.
- A.J. Healey, D.D. Marco, R.B. McGhee, D.P. Brutzman, R. Cristi, F.A. Papoulias, and S.H. Kwak. Tactical/execution level coordination for hover control of the NPS AUV II using onboard sonar servoing. In *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, pages 129–138, Cambridge, MA, July 1994. IEEE.
- N. Kato, Y. Ito, J. Kojima, K. Asakawa, and Y. Shirasaki. Guidance and control of autonomous underwater vehicle Aqua Explorer 1000 for inspection of underwater cables. In *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Technology*, pages 195–211. University of New Hampshire, Marine Systems Engineering Laboratory, September 1993.
- R. L. Marks, H. H. Wang, R. C. Burton, and M. J. Lee. SPOTTER: A submersible platform and ocean technology testbed for engineering research. Project report, Stanford Aerospace Robotics Laboratory and Monterey Bay Aquarium Research Institute, 1991.
- R. L. Marks, T. W. McLain, D. W. Miles, S. M. Rock, G. A. Sapilewski, H. H. Wang, R. C. Burton, and M. J. Lee. Monterey Bay Aquarium Research Institute/Stanford Aerospace Robotics Laboratory Joint Research Program Summer Report 1992. Project report, Stanford Aerospace Robotics Laboratory and Monterey Bay Aquarium Research Institute, 1992.
- R. L. Marks, S. M. Rock, and M. J. Lee. Automatic object tracking for an unmanned underwater vehicle using real-time image filtering and correlation. In *Proceedings of*

- IEEE Systems, Man, and Cybernetics*, France, October 1993. IEEE.
- R. L. Marks, M. J. Lee, and S. M. Rock. Visual sensing for control of an underwater robotic vehicle. In *Proceedings of IARP Second Workshop on Mobile Robots for Subsea Environments*, Monterey, May 1994. IARP.
- R. L. Marks, H. H. Wang, M. J. Lee, and S. M. Rock. Automatic visual station keeping of an underwater robot. In *Proceedings of IEEE Oceans 94 Osates*, Brest, France, September 1994. IEEE.
- R.L. Marks, S.M. Rock, and M.J. Lee. Real-time video mosaicking of the ocean floor. *IEEE Journal of Oceanic Engineering*, 20(3):229–241, July 1995.
- Richard L. Marks. *Experiments in Visual Sensing for Automatic Control of an Underwater Robot*. PhD thesis, Stanford University, Stanford, CA 94305, June 1995.
- Timothy W. McLain. *Modelling of Underwater Manipulator Hydrodynamics with Application to the Coordinated Control of an Arm/Vehicle System*. PhD thesis, Stanford University, August 1995. Also published as SUDAAR 670.
- John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley Publishing Company, Inc., Menlo Park, CA, 1994.
- G. Pardo-Castellote and S. A. Schneider. The Network Data Delivery Service: Real-Time Data Connectivity for Distributed Control Applications. In *Proceedings of the International Conference on Robotics and Automation*, San Diego, CA, May 1994. IEEE, IEEE Computer Society.
- A.M. Pascoal. The AUV MARIUS: Mission scenarios, vehicle design, construction and testing. In *The Second Workshop on: Mobile Robots for Subsea Environments*, pages 127–140. International Advanced Robotics Programme, May 1994.
- M. Perrier, V. Rigaud, E. Coste-Manière, D. Simon, and A. Peuch. VORTEX: a versatile testbed vehicle for control algorithm evaluation. In *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Technology*, pages 29–36. University of New Hampshire, Marine Systems Engineering Laboratory, September 1993.
- RTI. Real-time innovations, inc, 1995. 954 Aster, Sunnyvale, CA. email: info@rti.com.
- G. N. Saridis. Toward the realization of intelligent controls. In *Proceedings of the IEEE*, volume 67:8. IEEE, 1979.
- S. A. Schneider and R. H. Cannon. Experimental Object-Level Strategic Control With Cooperating Manipulators. *The International Journal of Robotics Research*, 12(4):338–350, August 1993.
- S. A. Schneider, M. A. Ullman, and V. W. Chen. ControlShell: A Real-Time Software Framework. In *Proceedings of the 1991 IEEE International Conference on Systems Engineering*, Dayton, OH, August 1991.
- S. A. Schneider, V. W. Chen, and G. Pardo-Castellote. ControlShell: A Real-Time Software Framework. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service and Space*, volume II, pages 870–7, Houston, TX, March 1994. AIAA, AIAA.
- M. A. Ullman, R. H. Cannon, and S. M. Rock. A Modular System Architecture for Multi-Manipulator, Free-Flying Space Robots. In *Proceedings of the International Symposium on Intelligent Control*, pages 90–97, Chicago, IL, August 1993.
- H. H. Wang, R. L. Marks, S. M. Rock, M. J. Lee, and R. C. Burton. Combined Camera and Vehicle Tracking of Underwater Objects. In *Proceedings of Intervention/ROV '92*, San Diego CA, June 1992.
- H. H. Wang, R. L. Marks, S. M. Rock, and M. J. Lee. Task-Based Control Architecture for an Untethered, Unmanned Submersible. In *Proceedings of the 8th Annual Symposium of Unmanned Untethered Submersible Technology*, pages 137–147. Marine Systems Engineering Laboratory, Northeastern University, September 1993.
- D.R. Yoerger, A.M. Bradley, and B.B. Walden. The autonomous benthic explorer. *Unmanned Systems*, 9(2):17–23, Spring 1991.
- Howard H. Wang** received his B.S. degree in Aerospace Engineering from the University of Michigan in 1989 and the M.S. degree in Aeronautics and Astronautics from Stanford University in 1990. Currently, he is a Ph.D. candidate at Stanford University working on task-level control of semi-autonomous underwater robots. This work is part of a joint collaboration between the Stanford Aerospace Robotics Laboratory and the Monterey Bay Aquarium Research Institute to study advanced underwater-robot technologies.
- Stephen M. Rock** received his S.B. and S.M. degrees in Mechanical Engineering from the Massachusetts Institute of Technology in 1972, and the Ph.D. degree in Applied Mechanics from Stanford University in 1978. Dr. Rock joined the Stanford faculty in 1988, and is now a Professor in the department of Aeronautics and Astronautics. Prior to joining the Stanford faculty, Dr. Rock led the Controls and Instrumentation Department of Systems Control Technology, Inc. In his eleven years at SCT, he performed and led research in integrated control; fault detection, isolation and accommodation; turbine engine modeling and control; and parameter identification. Dr. Rock's current research interests include the development and experimental validation of control approaches for robotic systems and for vehicle applications. A major focus is both the high-level and low-level control of underwater robotic vehicles.
- Michael J. Lee** received his S.B. degree in Electrical Engineering from the Massachusetts Institute of Technology in 1975 and the M.S. degree in Electrical Engineering from Stanford University in 1979. Mr. Lee joined the Hewlett Packard Co. in 1976 and held several engineering and management positions including Department Manager of the Control Systems Department at HP Laboratories. In 1987, Mr. Lee joined MBARI as one of its founders with the position of

Technical Director. Mr. Lee was an Adjunct Profes-

sor in Electrical Engineering at Boise State University from 1977 through 1978, and is currently a Consulting Associate Professor at Stanford University.