Underwater Experiments Using A Reactive System For Autonomous Vehicles

R. Peter Bonasso

The Autonomous Systems Laboratory
The MITRE Corporation
7525 Colshire Drive, Mclean, Virginia 22102
pbonasso@mitre.org

Abstract

This paper describes a situated reasoning architecture, originally used with ground mobile robots, which is shown to easily integrate control theoretic algorithms, navigation heuristics and human supervision for semi-autonomous robot control in underwater field environments. The control architecture produces reaction plans that exploit low-level competences as operators. The low-level competences include both obstacle avoidance heuristics and control-theoretic algorithms for generating and following a velocity/acceleration trajectory. Experiments with an undersea remotely-piloted robot in a test tank at the Deep Submergence Laboratory at Woods Hole, MA are described. The robot performed both pilot-aided and autonomous exploration tasks robustly during normal changes in the task environment. The architecture was implemented in the GAPPS/REX situated automata programming language. The guaranteed constant cycle time of the synchronous REX circuits allowed for rapid tuning of the parameters of the control-theoretic and heuristic algorithms to obtain smooth, safe motion.

Introduction

We are interested in programming robots to carry out tasks robustly in field environments (Bonasso et al. 1990). Field environments are those in which events for which the robot has a response can occur unpredictably, and wherein the locations of objects and other agents is usually not known with certainty until the robot is carrying out the required tasks. We expect the agent to be able to deal with its own mechanical and sensor limitations (e.g., wheel slippage, limited sensor sampling rates), and with natural changes in the flow of events (e.g., normally moving obstacles or other agents, transition from day to night). But when confronted with events for which it has no response (e.g., a meteor shower or runaway train), we expect the agent only to safely cease operations. Situated reasoning research (e.g., (Brooks 1986, Chapman & Agre 1986, Kaelbling 1987, Firby 1989, Schoppers 1989) addresses just such requirements for intelligent robots.

In early 1989, the MITRE Corporation and the Woods Hole Oceanographic Institute (WHOI) agreed to conduct joint research to investigate the use of situated reasoning techniques to control autonomous and tethered underwater vehicles for the purpose of deep ocean exploration. Yoerger and Slotine of MIT and WHOI's Deep Submergence Laboratory (DSL) had developed an adaptive control technique for trajectory generation and following for undersea vehicles (Yoerger & Slotine 1991). Bonasso of MITRE's Autonomous Systems Laboratory (ASL) had developed a situated reasoning architecture featuring reaction plans which used subsumption competences as robust operators (Bonasso 1990) for ground mobile vehicles. We desired to investigate the hypotheses that the architecture could be used for robust operations in underwater environments, and that it would allow for the integration of extant control-theoretic algorithms.

Experiments with the architecture were to be carried out on the WHOI Remotely Piloted Vehicle (RPV) in a test tank. The control-theoretic algorithms along with transponder-based navigation algorithms execute on a surface computer with which the reactive software was to communicate over an RS232 link (see Experiments and Figure 5).

Software Architecture

As stated above, the architecture consists of reaction plans which use subsumption competences for operators. For the exploration tasks in the tank, such plans are relatively simple: the usual goal is to navigate to coordinates as specified externally by the pilot or internally by the robot. As will be discussed, our programs result in synchronous circuits, so that the desired goal point can be different every cycle, thus allowing the human or the robot to set a new goal as often as desired. The most complex plan used was a series of waypoints for autonomous exploration.

The reason even these simple plans are sufficient at a given level of detail for these tasks with the actual vehicle is that the operators such as "move to x,y,z" are competent. That is, the operators can deal with variations in the environment as part of their design, thus unburdening the reaction plan of those considerations. Figure 1 shows the concept for the layered competences in our architecture. In our work we commit to layered control and to the combining of directives between layers. But we do not commit to the original subsumption language, asynchrony of finite state machines, or the use of only inhibition and suppression techniques (see Brooks 1986). Thus we use the term layered competences rather than subsumption.

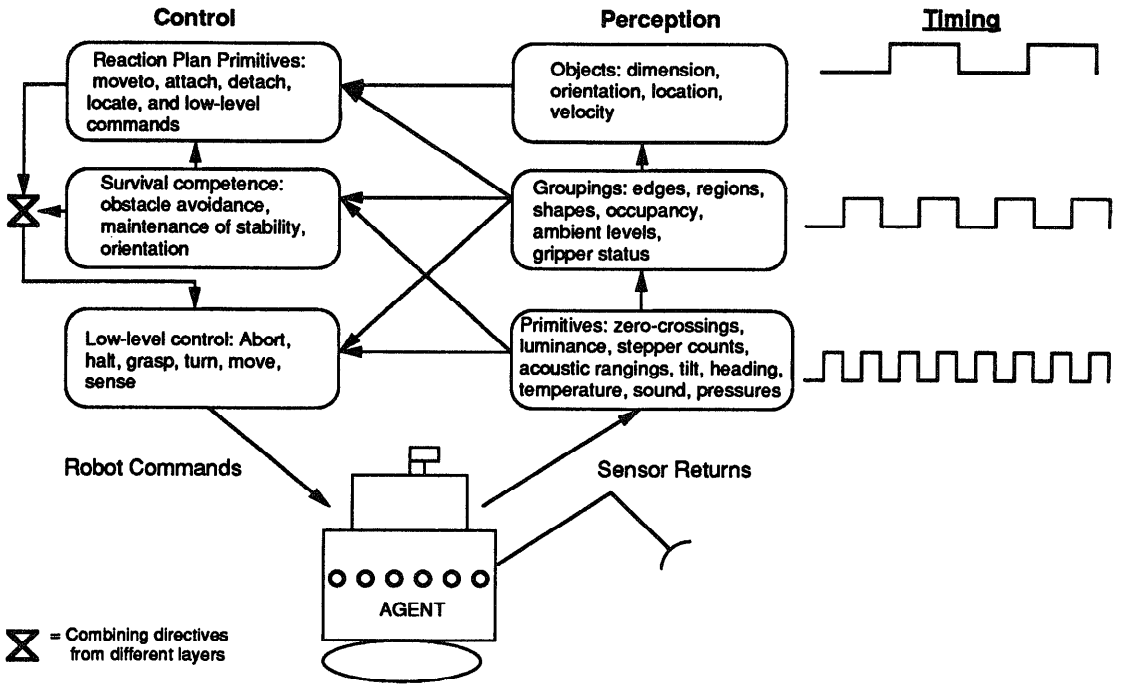

Figure 1. Notional Architecture of Layered Competences

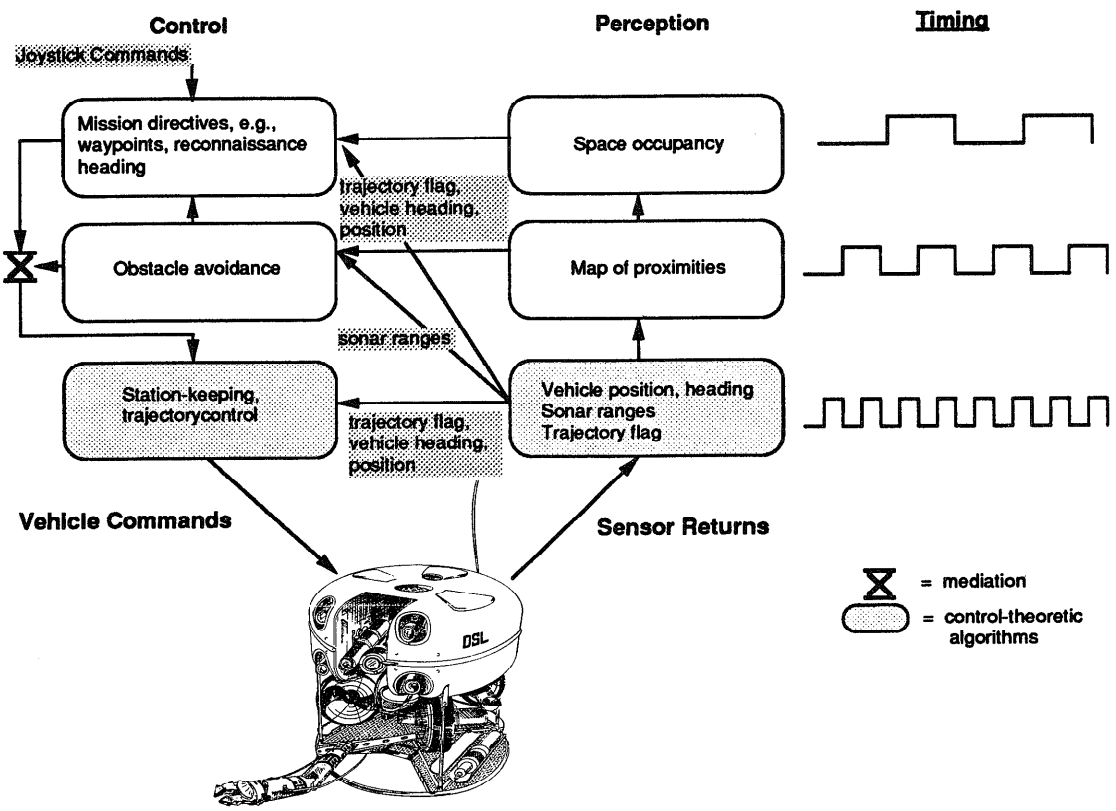

Figure 2. Instance of the Architecture for Underwater Tasks

Figure 1 is notional. For instance, the lowest level perception results could be made available to the highest level reasoning. And there may be more than three layers of competence, though in practice, we have used the three layers shown.

There are three hallmarks of these layered competences. The first is the basic trait of the subsumption architecture: higher level competences subsume those of the lower-levels. For example, once the robot has a competence to avoid obstacles on the fly, any high-level navigation vector which is generated by, say, a "move to x,y,z" reaction plan operator, will be adjusted to insure avoiding a previously unseen obstacle. This merging of directives (shown by the valve icon in Figure 2) can involve sophisticated algorithms as in our use of navigation templates (Slack 1990) for the robust motion of a land mobile robot.

The second hallmark is that the lowest level dictates the smallest cycle time, and higher-level cycles are multiples of that time. Our implementation generates synchronous circuits which at each strobing or tick of the circuit guarantees outputs for the lowest-level competence. Subsequent ticks produce additional outputs from higher levels in the architecture. This insures that the lower levels can be configured to effect emergency reactions tailored to the fastest problematic events in the environment, and yet will be blended into higher-level outputs as they become available.

The third hallmark is that levels of perception processing roughly match the levels of reactive competence, i.e., that perception at each level is task-driven. Thus, in the implementation, there may be global structures to allow for search efficiency, but task-related perception algorithms, if not individual representations, can exist at each level. For example, in one of our land mobile robot developments, the concept of (aware-p ?class-of-things) is used as part of a "locate object" competence. If the predicate is not true, then the robot's database of objects must be updated via a directed sensor search algorithm. For some classes of things, the agent becomes aware simply by receiving raw data, such as the signal from a bumper contact switch.

In the case of the RPV experiments, the specific architecture is shown in Figure 2. Here we have made the control-theoretic algorithms a competence for reaching and keeping station at a desired coordinate position. These algorithms handle well the dynamics caused by vehicle weight changes and shifting currents. The control-theoretic competence is actually treated as a kind of actuator which receives a goal point and sends a trajectory status flag which indicates whether the vehicle has completed following the computed trajectory to the goal point.

The highest level generates goal points from the plan and monitors the vehicle position until it achieves the goal point, or the trajectory following is complete. Due to limitations in the transponder system or dead bands in the thrusters, the vehicle can fail to achieve a goal position within a given tolerance. In these cases, the highest level competence (human or autonomous) can select an adjusted goal point which will take these errors into account.

The obstacle avoidance competence monitors the readings from a set of proximity detection sonars, and if any reading is less than a given "danger close" distance, it uses a proximity map generated from the sonars to determine a goal point which will move the vehicle away from obstacles appropriately. The algorithm basically sums the vectors of proximities from the sonars into a resultant which is crossed with the negative unit vector.

The mediation at the valve icon is between the mission goal point and the obstacle avoidance goal point. The reaction plan described below allows the mission goal point to be passed on to the low-level control as long as it does not conflict with the goal point that obstacle avoidance is trying to achieve. Though the mediation computation can be complex, in our experiments to date, simply having the obstacle avoidance competence usurp the mission directive when a collision was imminent resulted in a useful behavior.

When the task is to have the RPV wander around the tank on its own, space occupancy computed from the sonars is used to look for the unoccupied area farthest from the vehicle's current position and to select a goal point along the resulting heading.

Implementation

In considering an implementation, we wanted the resulting code to run synchronously. If the software executing on the AI computer had guaranteed constant cycle times, then since the RPV surface computer executed constant time control-theoretic algorithms, we could isolate the software operations from sensor and actuator physics for analyzing the behavior of the robot. Constant time cycles can be achieved by making all computation synchronous. We also wanted to insure that the formal semantics of any abstract representations still held in the on-board software. Synchronous operations helps maintain consistent semantics about the agent's most recent perception of the world and the formal rationale for carrying out the next action.

The desire for synchronous computations and consistent semantics led us to implement the situated reasoning software in the GAPPS/REX language (Kaelbling 1988). The language brings to bear formal semantics about the relationship between an agent's internal states and those of the environment (Rosenschein & Kaelbling 1986). The accompanying robot programming environment can be used to develop the software, while the resulting synchronous circuits guarantee constant cycle times.

With this language, the architecture dictates a programming methodology as follows. First write GAPPS goal reduction rules for the invocation of competences. The rules in Figure 3 can be used to safely achieve a desired goal point as read from a joystick operated by a pilot. Figure 4 shows similar rules for an internally generated goal point.

After writing these rules, the next step is to write the REX code for the functions that make up the competence execution. In Figure 3, for example, the (no-obstacles)

predicate uses the raw sonar readings to look for danger, while the (avoid-obstacle-coords) function computes the goal-point from the proximity map. In Figure 4, (set-wander-heading) and (set-wander-coords) use the space occupancy data. Though both functions perform essentially the same computation, the REX optimizer will unify the "wires" of the resulting virtual circuit to eliminate redundancies.

Finally, one builds the reaction plan using the goal expressions in the reduction rules or operators which produce reduction rules via regression (Kaelbling & Rosenschein 1990). The simple plans for the tasks outlined herein can be built using a prioritization of goals. In GAPPS (prio-and g_1 ... g_n) tries to satisfy n goals, but failing that tries to satisfy n-1 goals, etc. The plan for aiding a human piloting the RPV from a joystick (see Figure 3) is:

(prio-and (maint not-crashed)(ach joystick goal point))

An example plan to have the RPV robot roam safely around an area (see Figure 4) is:

(prio-and (maint not-crashed)(ach wander))

```
(defgoalr (maint not-crashed)
  (if (no-obstacles)
    (do anything)
    (ach avoid nearest obstacle)))

(defgoalr (ach joystick goal point)
  (if (orm (not-equalm (input-trajectory-flag))
           !*trajectory-complete*)
       (at-joystick-goal-point))
    (do anything)
    (and (do new-command !*true*)
         (do world-x (first (joystick-inputs)))
         (do world-y (second (joystick-inputs)))
         (do world-z (third (set-joystick-z)))
         (do rpv-heading (set joystick-heading))
         (do rpv-speed !*max-speed*)))

(defgoalr (ach avoid nearest obstacle)
    (and (do new-command !*true*)
         (do world-x (first (avoid-obstacle-coords)))
         (do world-y (second (avoid-obstacle-coords)))
         (do world-z (third (avoid-obstacle-coords)))
         (do rpv-heading (set-joystick-heading))
         (do rpv-speed !*caution-speed*)))
```

Figure 3 . GAPPS reduction rules for aiding robust piloting. The top level goal is (prio-and (maint not-crashed) (ach joystick goal point)). Terms with asterisks are global parameters which are formed into a structured memory location by the ! symbol. Ach and maint are abbreviations for achieve and maintain respectively. Orm is a REX machine for disjunction. The do commands essentially send the specified values to a vector of outputs. These outputs are sent via RS232 to the vehicle trajectory controller.

```
(defgoalr (ach wander)
    (if (notm (RPV-at-wander-angle))
      (ach turn to wander angle)
      (ach wander set point)))

(defgoalr (ach turn to wander angle)
    (and (do new-command? !*true*)
         (do rpv-heading (set-wander-heading))))

(defgoalr (ach wander set point)
  (if (orm (not-equalm (input-trajectory-flag))
           !*trajectory-complete*)
       (at-joystick-set-point))
    (do anything)
    (and (do new-command !*true*)
         (do world-x (first (set-wander-coords)))
         (do world-y (second (set-wander-coords)))
         (do world-z (third (set-wander-coords)))
         (do rpv-speed !*max-speed*)))
```

Figure 4. GAPPS Rules for a wander behavior. The heading is selected by the function (set-wander-heading). Once the RPV is pointed at the goal position, (set-wander-coords) provides outputs to the low-level trajectory controller. When used in the top-level goal of (prio-and (maint not-crashed) (ach wander)), robust roaming is exhibited by the vehicle.

Experiments

For our experiments we selected pilot-aided and autonomous explorations in the WHOI test tank. Figures 5 and 6 (derived from Yoerger & Slotine 1991) depict the hardware and computational layout of DSL's RPV in the test tank. This test facility is designed to allow visiting researchers to conduct experiments in an underwater environment without the problems and cost associated with actual ocean experiments. The reaction plans and first two levels of competence ran on a microExplorer, and resulting goal points and headings were sent to the surface computer via RS232 link and then to the vehicle via the tether. The control-theoretic competence generated velocity and acceleration trajectories on board, and returned the trajectory-complete flag along with position and heading data to the surface computer which passed that data on to the microExplorer.

The architecture concepts were originally implemented in GAPPS/REX on a HERO 2000 mobile robot and on a Denning MRV-3 sentry robot for navigation among indoor and outdoor obstacles (Bonasso 1991). Both robots use ultrasonics as the primary sensors, the HERO using a single sonar reflecting off a rotating mirror; the MRV-3 using a ring of 24 sonars. The RPV was originally equipped with a front spot-beam sonar but with no other sensors which could detect proximate obstacles. The experience with the ground mobile robots thus motivated the design and construction of a Proximity Obstacle Detection System (PODS) for the RPV. This system uses

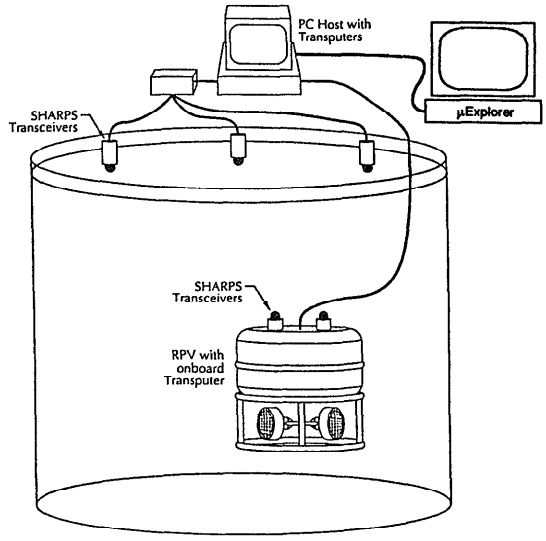

Figure 5. The experiments used a test bed vehicle operating in a test tank. The vehicle is equipped with a distributed Transputer based computer system and a precision navigation system.

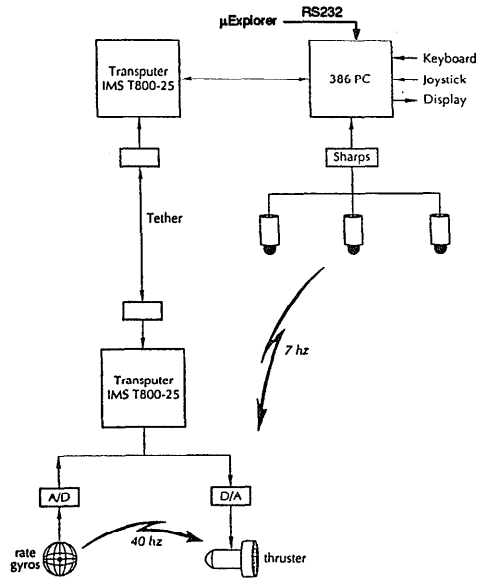

Figure 6. The distributed computing system utilizes IMS T800 Transputers both on the vehicle and the 386 PC host. The software development environment uses INMOS C and standard INMOS utilities to load code through the network. The link between the Transputers, which normally cannot work over long distances, has been modified to an RS422 standard.

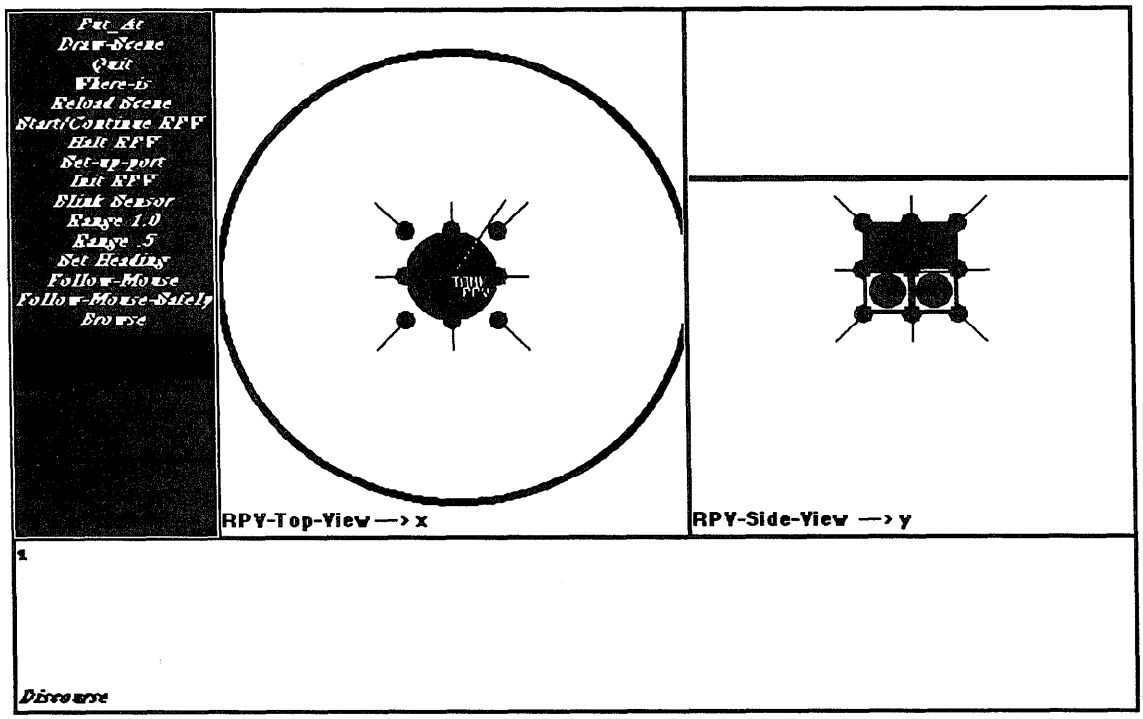

Figure 7. Pilot Interface Display. The small circles with lines represent the sonars and their orientation. The large single line in the top view indicates the vehicle's heading.

14 altimeter sonars positioned on the vehicle at the vertices and faces of an imaginary cube circumscribing the vehicle.

Before the PODS construction was complete, we conducted an early experiment with the RPV using a reaction plan for waypoint following in the test tank at WHOI. On-board station-keeping kept the vehicle at each waypoint within the 0.015 meters required by the plan except in cases where the waypoint was extremely close to one or more of the transponders. For those cases, the reaction plan could be adjusted to add a factor to the waypoint coordinates to compensate for the error.

Once the PODS was integrated on-board the vehicle, we used the architecture to successfully develop a pilot support system. A human pilot uses a joystick to direct the vehicle to navigate to and orient on a porthole in the RPV test tank. The pilot attempts to point the camera mounted on the RPV at the porthole in order to view the scene beyond. In all runs, the low-level control competence kept the vehicle oriented and on station while the obstacle avoidance competence kept the vehicle from colliding with the side of the test tank during the operation. Thus, the pilot simply pushed the joystick in the direction he wanted the vehicle to go, and held the joystick against its stops in that direction while viewing the camera monitor.

The less than immediate response from the vehicle during such piloting made holding the joystick tedious, so we reverted to a simple mouse interface shown in the microExplorer display in Figure 7. On each Rex cycle, the mouse data was used to specify the high-level goal point and heading, and the display was updated with the actual vehicle position data. With this display and the camera monitor, the pilot did not need to have the vehicle in sight (which is the usual pilot mode for deep sea operations) to successfully explore the tank environment. The piloting method with this display was to first specify a heading and observe the camera monitor. When an object of interest appeared, such as a porthole, the mouse was positioned in the display at or beyond the tank's edge along the heading line of the vehicle. The pilot adjusted this goal point each cycle via observations from the monitor. Since the vehicle autonomously maintains a safe distance from any obstacles, including the walls, a pilot need not be concerned with a safe positioning of the mouse. In all cases, the vehicle either touched the wall with negligible force before recovering, or stayed within the plan-specified 0.6 meters "danger-close" distance +- 0.015 meters.

To obtain the above robust behavior, some tuning of the maximum trajectory speed and the danger-close distance was needed to insure that there was sufficient time from the reading of the sonars to the output of an obstacle avoidance goal point so as to avoid collisions. This was easily achieved in two to three trials in each experiment, since the time in question is the constant time of the Rex cycle plus the constant time of reading the input data (the PODS was set to update registers on the surface navigation computer on a continuous basis, thus providing essentially memory-mapped I/O from that system to the Rex circuit via the RS232 port). The computation time of the control-theoretic algorithms is not included since the vehicle halts in its current trajectory when a new goal point is received. Obtaining the position and sonar inputs took on the order of one second, while the Rex circuit cycle ran at about 3 HZ. A maximum safe trajectory speed of 0.7 meters per second resulted in the best performance (the trajectory control accelerates to the maximum velocity and decelerates when approaching the goal point).

The second set of experiments concerned the wander behavior. Again in all cases, once the maximum safe trajectory speed and best danger-close distance was determined, the vehicle wandered autonomously around the tank area, successfully avoiding the sides of the tank and any obstacles placed in the tank (such as an access ladder and pylons used in an unrelated experiment placed in the tank midway in our experiments). This demonstration is the three-dimensional underwater analogy to the outdoor wander behavior of our MRV-3 ground mobile robot, and thus seems to confirm the hypothesis that the architecture used in a ground mobile environment translates well to a three-dimensional underwater environment.

Related Work

The Sea Grant work at MIT (Bellingham et al. 1990) also involved the use of a subsumption architecture for control of an undersea vehicle. The researchers attempted to include the control of the non-linear dynamics of their Sea Squirt vehicle in the original subsumption framework, but implemented all the algorithms in software. Soon they found that they had to move the closed-loop control of the vehicle dynamics outside the subsumption framework to reduce the memory and throughput requirements imposed on the central computer. This group also modified the subsumption architecture such that no communication between layers is allowed, conflicts being resolved through a strict prioritization scheme. Our work, on the other hand, begins with an architecture that provides for the "outboarding" of a layer of competence (i.e., the same type of dynamics control as in the Sea Grant work), but allows for any combination of communication and mediation between layers that experiments and analysis may deem appropriate and practical.

(Schudy & Duarte 1990, and Rodseth 1990) report on two other architectures for underwater vehicles. The former is a complex integration of control-theoretic algorithms with a blackboard-based planner; the latter, a modular system implemented with object-oriented programming, which communicates state variables between modules. The former system is currently only a concept, but seems too complicated to exhibit real-time response without extensive tuning of the parts. The latter system, used successfully on a small underwater vehicle, does not appear to use control-theoretic closed-loop control, but rather a set of ad hoc heuristics for the task and environment at hand.

Conclusions

The Woods Hole experiments support the hypotheses that the situated reasoning architecture described in this paper can be used for robust operations in underwater environments, and that it does allow for the easy integration of extant control-theoretic algorithms. The architecture, along with the GAPPS/REX programming environment enabled us to port ideas used with ground mobile robots to an underwater robot, which then performed its tasks robustly during normal changes in the task environment. We attain desired synchronous operations and formal semantics by implementing the architecture in GAPPS/REX. This programming environment supports flexible combinations of competing behaviors as appropriate for the task. Further, the pilot-aiding task showed that the architecture easily integrates natural, artificial, and analytical intelligence. However, the tasks in the above experiments have involved simple plans and have not required extensive sensor processing. A planned follow-on ocean mapping experiment, which will require more extensive reaction plans and will use more sophisticated sensors, should do much to confirm or deny these preliminary indications.

Acknowledgments

We wish to acknowledge the excellent efforts put forth by Nathan Wilson and Leslie Kaelbling in making the necessary changes to GAPPS/REX which support the architecture described in this paper. As well, the technical support from David Mindell of WHOI, who programmed most of the RPV software, was immeasurable in achieving useful findings from the research RPV and tank at Woods Hole.

References

Bellingham, J.G.; Consi, T.R.; Beaton, R.M,; and Hall, Wm. 1990. Keeping Layered Control Simple. In Proceedings of the Symposium on Autonomous Underwater Vehicle Technology, 3-8. Washington D.C. IEEE. Cat#90CH2856-3.

Bonasso, R.P.; Hwang, V.S.; Sanborn, J.C.; and Stoney, W.E. 1990. Investigating Robot Safety and Robustness In An Autonomous Systems Laboratory. In Proceedings of the 1990 Space Applications of AI, Robotics, and Automation, 105-108. Kobe, Japan. AIAA.

Bonasso, R.P. 1990. Creature CO-OP : Achieving Robust Remote Operations With A Community of Low-Cost Robots. In Fifth Conference on AI for Space Applications, 257-269. Huntsville, AL. NASA Pub 3073.

Bonasso, R.P. 1991. Integrating Reaction Plans and Layered Competences Through Synchronous Control. In Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia. Morgan Kaufman.

Brooks, Rodney A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2:14-23.

Chapman, David and Agre, Philip E. 1986. Abstract Reasoning As Emergent From Concrete Activity. In Proceedings of the Workshop on Planning & Reasoning About Action, 268-272. Portland, OR. Morgan Kaufman.

Firby, James R. 1989. Adaptive Execution in Complex Dynamic Worlds. PHD diss., Dept. of Computer Science, YALEU/CSD/RR #672, Yale University.

Kaelbling, Leslie Pack. 1987. An Architecture for Intelligent Reactive Systems. *Reasoning about Actions and Plans*, 395-410. Morgan Kaufman.

Kaelbling, Leslie Pack. 1988. Goals As Parallel Program Specifications. In Proceedings of the Seventh National Conference on Artificial Intelligence, 60-65. Minneapolis-St. Paul, Minnesota. AAAI Press.

Kaelbling, Leslie Pack and Rosenschein, Stanley J. 1990. Action and Planning In Embedded Agents. *Robotics and Autonomous Systems* 6(1&2): 35-48.

Rosenschein, S. J. and Kaelbling, Leslie Pack. 1986. The Synthesis of Digital Machines with Provable Epistemic Properties. In Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge, 83-98. Monterey, CA. Morgan Kaufman.

Rodseth, J.O. Software Structure For A Simple Autonomous Underwater Vehicle. In Proceedings of the Symposium on Autonomous Underwater Vehicle Technology, 23-26. Washington, D.C. IEEE. Cat#90CH2856-3.

Schudy, R.B., and Duarte, C.N. 1990. Advanced Autonomous Underwater Vehicle Software Architecture. In Proceedings of the Symposium on Autonomous Underwater Vehicle Technology, 9-21. Washington D.C. IEEE. Cat#90CH2856-3.

Schoppers, M. J. 1989. In Defense of Reaction Plans As Caches. *AI Magazine*. 10(4): pp 51-60.

Slack, Marc G. 1990. Situationally Driven Local Navigation for Mobile Robots, JPL Pub. 90-17, NASA.

Yoerger, Dana R. and Slotine Jean-Jacques E. 1991. Adaptive Sliding Control of An Experimental Underwater Vehicle. In the Proceedings of the IEEE International Conference on Robotics and Automation. Sacramento, CA. IEEE Computer Society Press.