1995-06

Evaluation of the NPS PHOENIX Autonomous Underwater Vehicle Hybrid Control System

Healey, A.J.; Marco, D.B.; McGhee, R.B.; Brutzman, D.P.; Cristi, R.

Proceedings of the American Control Conference, Seattle, Washington, June 1995.
https://hdl.handle.net/10945/44823

Proceedings of the
American Control Conference
Seattle, Washington • June 1995

Evaluation of the NPS PHOENIX Autonomous Underwater Vehicle Hybrid Control System

A. J. Healey, D. B. Marco, R. B. McGhee, D. P. Brutzman, R. Cristi

Naval Postgraduate School
Monterey
California, 93943

ABSTRACT

This paper describes recent work with the NPS PHOENIX vehicle in the further development of the intelligent control software incorporating hover control behaviors. Of particular interest is the use of the TRITECH ST1000 and ST725 high frequency sonars to provide data about the environment. Vehicle positioning is proposed to be based in a local relative sense, augmenting global positioning by LBL transponders. Motion behaviors around a target area have been implemented including diving and pitch control under thruster power; heading control at zero speed; lateral and longitudinal positioning, as well as the automatic initiation of filters as needed for target tracking. A tri-level controller architecture is discussed as part of an ongoing evaluation for coordinating the task based control of vehicle robotic behaviors.

INTRODUCTION

For several years the Naval Postgraduate School has been engaged in development of advanced control technology for unmanned vehicles that will have useful roles to play in the future actions of the US Navy. One of these roles is in the support of minewarfare missions involving search and find operations against lethal targets. While some have contended that the use of tethered ROV's, are all that is required, it is our contention that the tether is too restrictive and must be eliminated. The consequence is that communications with the vehicle are at low rate through acoustic links and some level of autonomy will be needed to maintain vehicle control. Building an ever increasing level of automatic capability into a vehicle is of interest to us. In particular, under a new NSF grant, we are concerned with the ease of reconfiguration of control software code as missions become more complex or vehicle capabilities change. To that end, we have defined a tri-level software control architecture comprising Strategic, Tactical, and Execution levels. The three levels separate the software into easily modularized functions encompassing everything from logically intense discrete state transitioning to the interfacing of *asynchronous* data updates with the real time *synchronized* controllers that stabilize the vehicle motion to commands.

In our controller architecture, the Strategic level uses 'Prolog' as a rule based mission control specification language. It's inference engine cycles through the predicate rules to manage the discrete event logical aspects of mission related decisions. It transitions states, and generally develops the commands that drive the vehicle through its mission. Error recovery procedures from failures in the mission tasks or the vehicle subsystems are included as transitions to 'error' states that ultimately provide commands to the servo level control for appropriate recovery action.

The Tactical level, currently written in 'C' is set of functions that interface with the 'Prolog' predicates returning TRUE / FALSE in response to commands and queries, and which are also interfaced to the real time Execution level controller by asynchronous communications using script type message passing through a non-blocking socket.

The Execution level then commands the vehicle subsystems to activate behaviors that correspond to those commanded. Communication from the Tactical level to the Execution level takes place through a single socket. By the design of this hierarchical control system, the Tactical level runs asynchronously and retains the mission data file and the mission log file in global memory. It sends the command scripts to the Execution Level and requests data for the evaluation of state transitions. The architecture is a hybrid between

the true hierarchical control of NASREM [1] and the purely reactive schemes of subsumptionists [2, 3]. In this way, control of mission can be retained, while reacting to unanticipated events is also enabled.

In new work with the NPS PHOENIX - a newly renovated version of the NPS AUV II - we have extended the flight control experiments that were conducted and reported previously [4]. We have now developed the thruster control behavior of the vehicle. Experiments with coordinated actions between the high frequency sonar and thrusters in position control of the vehicle involve controlled sweeping of the sonar and positioning the vehicle to and from a wall. Results have indicated that it may be possible in the near future to use these sonars to drive a vehicle to a target or between obstructions.

This paper will describe some results of these experiments and discuss our evaluation to date concerning the control ideas put forth.

SYSTEM OVERVIEW

The Vehicle

The NPS PHOENIX, shown in Figure 1, has been recently outfitted with the tri-level controller and a sonar suite consisting of a Datasonics PSA 900 sonar at 200KHz. to derive altitude above bottom signals, and two Tritech high frequency sonars that are mechanically scannable through 360 degrees. The ST 1000 sonar is a 1 degree pencil beam profiling sonar that is best suited to measurement of the time of flight for the first strong return. The ST 725 sonar is a 20 degree high beam 2.5 degree wide that may be mechanically scanned either through full 360 degrees, or through a reduced sector, to capture a wider image of obstacles around the vehicle. This sonar returns intensity as a function of range in bins for any given heading.

These devices may be addressed for control purposes through a serial port where ASCII characters are used to command operational functions such as *'send one ping and analyze the return structure'* (either range to first return or intensity in a series of range bins), or *'turn by one step'*. By issuing a sequence of such commands, the sonar head may be made to *self center, continuously rotate while pinging and return the data stream,* or to *sweep over a defined sector and return the data stream.*

Additionally, the vehicle has now been equipped with cross body thrusters. Two vertical thrusters are for heave and pitch control, and two transverse thrusters are for heading and lateral movement control, in addition to the two propulsion motors at the stern and eight fin surfaces for flight control. The vehicle has a dry interior and a wet nose, a length of 2.13 meters and a dry weight of 175 kg. Sufficient energy storage (1100 Whr.) is provided by 4 lead acid gel batteries for approximately 3 hours of operational testing.

Control

In previous work [5], waypoint following in a transit phase of a mission was demonstrated in a swimming pool test area where competent behaviors were demonstrated composed of

a) Forward Speed Control,
b) Fin_Steering
c) Fin_Depth_Control
d) Waypoint_Following
e) Bottom_Following, and
f) Obstacle_Avoidance.

These control functions were implemented with a)-c) and f) running simultaneously, but subsumed by the guidance laws implemented in d); and, with c) subsumed by e). The control laws implemented have been based on PD, and Sliding Mode methods as explained in [6].

Control laws for these functions are readily accomplished entirely in the Execution level with digital control algorithms running at 0.1 sec. update rate. Now, however, new, more complex functions are being enabled using active control of thrusters and sonar. These are,

g) Submerge_and_Pitch_Control
h) Heading_Control,
i) Longitudinal_Positional_Control,
j) Speed_Control using command generators
k) Lateral_Positional_Control,
l) Center_Sonar,
m) Ping_and_Get_Sonar_Range (each sonar),
n) Ping_and_Get_Sonar_Range_ and_Step_One each sonar),
o) Step_Sonar (no ping, each sonar),

p) Initiate_Filter_For_Sonar_Range (Needed For Smoothed Range and Range Rate Estimation),
q) Reinitialize_Filter,

Most of these functions need a given subset of the actuator system to be active under the operation of either an open loop command or a feedback control law. Some of the functions use orthogonal sets of actuators and may be commanded without conflict. Some use the same actuators to control different functions and thus may be additive. This means, for example, that vertical thrusters may be used via control laws to control depth as well as pitch, and lateral thrusters to control heading as well as lateral position and side slip speed. In combination with propulsion motors, most functions including Submerge_and_Pitch_ Control, Longitudinal_Speed_Control and Longitudinal_ Position_Control, as well as Heading_Control, may now be commanded reliably. Heading_Control and Submerge_and_ Pitch_Control; and virtually any multiple combination of a) to o) above that would not cause a conflict of actuator control or sensor usage, are available.

Activation of orthogonal behaviors are instituted using a script composed of flags and set points that are a way of communicating between Tactical Level 'C' functions and the real time control loop of the Execution Level control. At each pass through the control loop, a read is made from the communications socket and a ladder check for particular 'case of' flags determines which set of sensors and actuators and control laws are to be activated during the computation cycle. The same technique is used to flag the activation of sonars, and filtering actions, and similarly for flags to indicate which data stream is to be written in return.

Reactive behavior in our controller can be handled in the Execution level control loop through command overrides following a sensor read, as, for instance, a new obstacle detection requiring an emergency surface or obstacle avoidance (flinch) response. At a higher degree in the Tactical level, reactive error recovery can be handled by resetting key parameters associated with control performance evaluations. An example is the resetting of a control gain if a particular function cannot be stabilized. Reactive behavior is also handled at the Strategic level by transitioning to states that command an error recovery procedure such as to surface if, for example, a particular action is not observed to be taken after a pre-specified time out.

While the work of [3] has developed GAPPS rules that are more like our Strategic level rules, but, in the end would also provide mode commands to vehicle servos, our work is developed around a rule based control to sequence mission related tasks [7, 8] according to a mission plan that could (if one prefers to view it this way) represent a hierarchy of state machines with transitioning from one to another state as mission phases are completed. The middle level of our tri-level architecture is then used to generate the scripts required to produce in the vehicle the requisite behavioral action. The Tactical Level functions deal with the interfacing between asynchronous mission control commands and the real time control computations of the 'sense - compute - send' cycle within the Execution level vehicle motion control loop.

We believe, as in Wang et. al. [9], that the key to successfully developing underwater robots is to have a fully functional Execution level first with attention paid to the proper functioning of the servo loops addressing the sensors and actuators as keyed to the control functions that must be capably instituted. The behaviors a) through o) are now stably implemented in the NPS PHOENIX vehicle through attention to appropriate digital control loops in the Execution level. In principle, once developed to a satisfactory point, the Execution level controller of any vehicle would not require any change as mission requirements change. The tri-level control system has been implemented and run with the PHOENIX vehicle in the Hover tank at NPS in a complex mission with results given in [12].

THE CONTROL NETWORK

The control system, illustrated in Figure 2, is currently implemented in hardware using three networked processors. All Execution level software is written in 'C' and runs on a GESPAC M68030 processor in a separate card cage inside the boat. Connected in the same card cage is an ethernet card and an array of real time interfacing devices for communications to sensors and actuators indicated in the details of Figure 3. The Execution level control code

containing a set of functions in a compiled module called **'exec.c'** is downloaded first and run to activate any mission. It starts the communication s socket on the GESPAC side and waits for the higher level controller to start.

Strategic Level

The Strategic level 'Prolog' rules are compiled and linked together with the supporting Tactical level 'C' language functions into the single executable process called **'Mission_Control'**, that is run in a SUN Sparc 4 laptop computer and linked through ethernet and a non-blocking socket to the Gespac processor. Upon starting, it first opens the SUN side of the communications socket, initiating the ethernet link between both SUN and GESPAC processors, then sending sequenced control commands to the vehicle. All vehicle control functions, with the exception of the transmission of sonar imaging data, communicate by message passing through that socket.

Tactical Level Software

A second SUN process called the **'Sonar Manager'** is opened which runs asynchronously in the SUN and with equal priority to the **'Mission_Control'**. This process is linked through a separate socket to the GESPAC for the purpose of the reception and handling of sonar imaging data. This process is activated if and when sonar is activated by the Strategic level rules. The **'Sonar Manager'** captures data that is sent out from the Execution level as soon as it has been acquired, and then processes and passes the data to be displayed on the IRIS Graphics workstation for visualization purposes.

The introduction of the additional process called **Sonar Manager** and it's separation from the **'Mission_Control'** Tactical level functions has been found to be important and a necessary first step toward a more general Concurrent Tactical Level that was foreseen by the earlier RBM architecture[7] and explained recently by Kwak and Thornton,[10]. The need for concurrency of multiple processes lies fundamentally with the fact that sonar data is obtained *asynchronously* with bounded but unknown latency and the servo control functions cannot wait for the sonar port data to arrive.. While it is perfectly normal to send control set point commands *asynchronously* to stable control loops, waiting for sonar returns could hold up the servicing of the inner servo loop commands to actuators. Thus in our solution to this problem, we have defined the additional **Sonar Manager** process to always read the socket onto which sonar data is written so that the socket is immediately free for another sonar write without delay and the servo loop is made independent of direct involvement with the sonar. As an unpleasant alternative, we have found that without the **Sonar Manager**, all the Tactical level functions would have to be modified to include a check to read sonar data if there. This would have been a cumbersome addition of much unnecessary code writing.

Execution Level Software

The structure of the Execution level software is illustrated by Figure 3 which indicates that it is composed of software at the hardware interface (software drivers) as well as software for vehicle control. After initialization of power systems and sonars, and the basic driver settings, the PIA card pins that control the on/off feature of power supplies, thruster power, screw power, and sonar power, a simple timing loop is entered and reentered at a fixed update rate (in our case 0.1 sec.) during which the following takes place,

1. read the socket 'A' for behavior based mode command flags and control set points,

2. read the sensors,

3. selecting appropriate 'C' code control functions for computing and sending control values to actuators, using multiple 'case of ' checks for distinguishing the commands,

4. writing selected data to memory or sockets 'A' or 'B' as appropriate, and

5. checking time for any time based events and waiting for the next timing interrupt to maintain integrity of the digital control loop,

Specific control laws as built into callable modules of code are easily selected according to the communication flags, provided that they exist in the first place.

SONAR MANAGEMENT AND CONTROL

Keeping the vehicle stationary and sweeping the ST1000 sonar through 360 degrees has enabled us to acquire maps of the scene within which the vehicle operates. Also, for many of the motion control behaviors listed above, the estimation of range rate is very important and requires that sonar management be effected at the execution level. For instance, to find a target we take the point of view of submarine officers that until you have three consecutive pings returning range consistently, there is no target out there. What this means here is that we command a heading on the sonar, tell it to ping and return range three times, and if three consecutive range data are close to each other, we associate a target with that fact. At that point, we start a Kalman Filter (constant gain) that will smooth the range data and estimate range rate. Range and range rate data is then acquired at the control loop rate (short ranges within 6 meters can be obtained that quickly). A third order filter is used in the standard notation [11] as ,

Range Dynamics Model

$$\mathbf{x}_{k+1} = \Phi \mathbf{x}_k + q_k$$

$$y_k = \mathbf{H}\mathbf{x}_k + v_k$$

with the update of estimate from

$$\hat{\mathbf{x}}_{k+1} = \Phi \hat{\mathbf{x}}_k + \mathbf{K}(y_k - \mathbf{H}\hat{\mathbf{x}}_k) \quad k = 1,...$$

Use of the innovation $(y_k - \mathbf{H}\hat{\mathbf{x}}_k)$ in the filter is key to the elimination of sonar dropout and spiking signal returns. For values larger than a defined threshold that is context dependent (say 0.3m), those returns are ignored in the filter update and the filter state is propagated uncorrected. If multiple returns are outside the threshold, we can say that the filter has lost lock on the target, at which time, re-acquiring of a target may become necessary. In our experiments to date, it has always been easy to acquire a target, still, the use of a technique such as a median filter to ensure that at least three consecutive pings return consistent range is recommended before any decision is made to start a filter and begin positioning behaviors. In open ocean, re-acquiring a target would involve initiating a new sweeping motion of the sonar.

One of the recommendations arising from our more recent work is that the **'sonar manager'** be interfaced to the Tactical level of the software architecture as it is clear that adaptive changes to the sonar power setting could become a necessary feature of practical sonar management. For instance, at close range, good results cannot be obtained when sonar power levels are high. Sonar power (gain in TRITECH terminology) must be tailored to the range of returns in addition to the usual Time Varying Gain function.

Positioning the vehicle with short range sonar returns has been documented in [12] from which the results in Figure 4 are reproduced.

SOFTWARE ARCHITECTURAL EVALUATION

It is not an easy task to evaluate a given control system architecture. The theoretical design for stability and robustness leads to selection of parameters that are used in the control functions of the Execution level. We are going beyond that now and are interested in the organization of control software. Some software controllers will be successful for fixed purpose tasks, but here, we have a multipurpose flexible control requirement and, because we are talking about control software, we are led to ask the following questions,

1) Does the controller permit easy evaluation of response and change to control parameters to 'tune' the low level servos?

2) Can this be done while testing is ongoing in real time?

3) Can new sensors be added to the vehicle with little change to the control software?

4) What levels of code and how many functions have to be changed for this new sensor to be added?

5) How many rules (code statements) must be changed if the mission is altered to eliminate, or to add, a new phase?

6) How is the control code modified to test just the performance of a particular existing sensor or actuator set?

7) How easy is it to change the data record for a different set of sensors?

8) How easy is it to change the conditions that define the transition signals?

There are perhaps many more questions that should be considered, dependent on the particular control system software used. The evaluation of our controller is ongoing.

CONCLUSION

The conclusion of our work to date has indicated that complex behavior can be readily coordinated through Strategic level rules, that are easily modified. These act as state transitioning mechanisms and the communication through Tactical level software to the Execution level controllers is a simple but convenient way of commanding competent functions of the vehicle. The design of well behaved control laws and functions at the Execution level is essential as a primary part of the design an is effected through careful attention to the digital control loop design. Human interfacing within the controller can take place at any level.

The independent coordination of sonar for range finding on a bearing, or for imaging over a particular sector of bearings is needed to derive motion commands for the vehicle. Smooth vehicle motion can be achieved in an underwater environment free from current and wave action provided that attention is given to the processing of the sonar data, but time delays in processing sonar data is a difficult problem to handle and is still under research. We would anticipate, however, that in the future, sonar based relative navigation without the use of LBL could be possible in structured or feature rich scenes. The work is continuing.

ACKNOWLEDGMENT

The authors wish to recognize the financial support of the National Science Foundation under Grant No.BCS-9306252 as well the Naval Surface Warfare Center, Annapolis Division, the Naval Explosive Ordnance Disposal Technology Center, and the Naval Postgraduate School Direct Research Foundation, all partly contributing to the conduct of this work,

REFERENCES

[1] Albus, J., "System Description and Design Architecture for Multiple Autonomous Undersea Vehicles" National Institute of Standards and Technology, Technical Note 1251, September 1988

[2] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot" *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23, 1986.

[3] Bonasso, R.P., Barrat, J. "A Reactive Robot System for Find and Visit Tasks in a Dynamic Ocean Environment", Proceedings of the 8th UUST, University of New Hampshire, Durham, NH. September 27-29, 1993 pp. 69-80

[4] Healey, A.J., Marco, D. B., " Experimental Verification of Mission Planning by Autonomous Mission Execution and Data Visualization using the NPS AUV II." Proceedings of IEEE Oceanic Engineering Society *Symposium on Autonomous Underwater Vehicles, AUV-92* Washington DC., June 2-3, 1992.

[5] Healey, A.J., Marco, D. B., "Slow Speed Flight Control of Autonomous Underwater Vehicles: Experimental Results with NPS AUV II" *Proceedings of the 2nd International Offshore and Polar Engineering Conference*, San Francisco, July 14-19 1992.

[6] Healey, A. J., Lienard, D., "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles", *IEEE Journal of Oceanic Engineering* Vol. 18, No. 3, July 1993 pp. 1-13

[7] Byrnes, R. B."The Rational Behavior Model: A Multi Paradigm, Tri-Level Software Architecture For Control Of Autonomous Vehicles", Ph.D. Dissertation, Naval Postgraduate School, Monterey CA. March 1993

[8] Byrnes, R., Kwak, S. H., McGhee, R. B., Healey, A. J., Nelson, M. L.,"Rational Behavior Model: An Implemented Tri-Level Multilingual Software Architecture for Control of Autonomous Vehicles" Proceedings of the 8th UUST, University of New Hampshire, Durham, NH. September 27-29, 1993 pp. 160-179

[9] Wang, H., Marks, R.L., Rock, S.M., Lee, M. J.," Task Based Control Architecture for an Untethered Unmanned Submersible" Proceedings of the 8th UUST, University of New

Hampshire, Durham, NH. September 27-29, 1993 pp. 137-149

[10] Kwak, S. H., Thornton, F. P. B., "A Concurrent Object-Oriented Implementation for the Tactical Level of the Rational Behavior Model Software Architecture for UUV Control ", *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, July 19-20, Cambridge, Mass. 1994 pp. 54-60.

[11] Gelb, A. ed. "Applied Optimal Estimation" MIT Press 10th printing 1988 ISBN # 0-262-57048-3.

[12] Healey, A. J., et. al. "Tactical / Execution Level Coordination for Hover Control of the NPS AUV II Using Onboard Sonar Servoing" *Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology*, July 19-20, Cambridge, Mass. 1994 pp. 129-138.

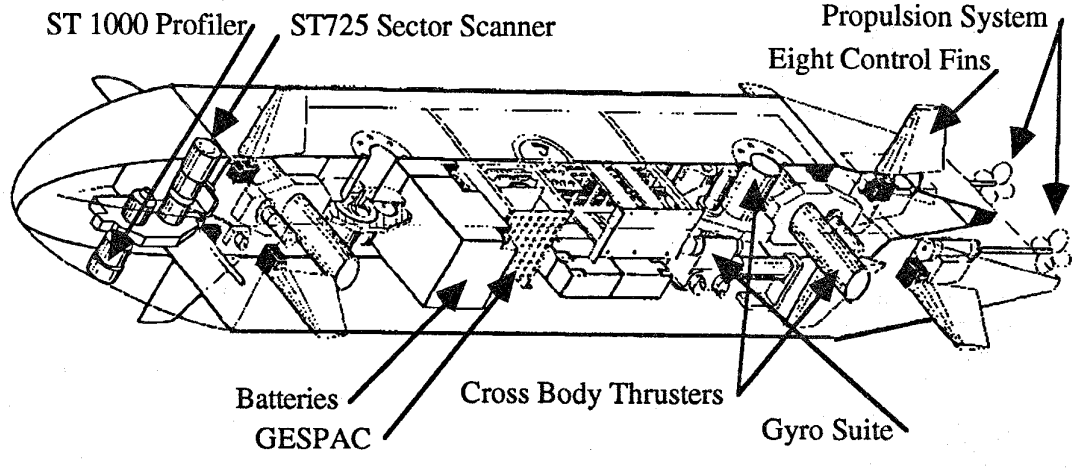

Figure 1 Outline Of The NPS PHOENIX Vehicle

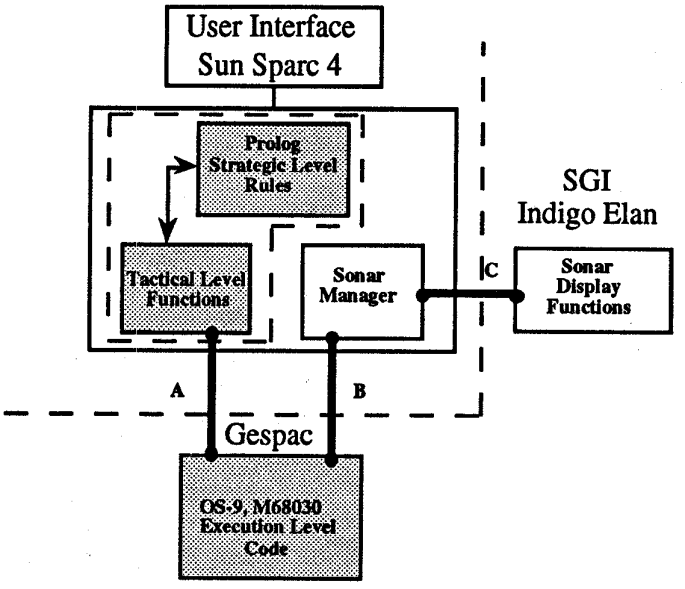

Figure 2 Outline Of The Phoenix Networked Controller

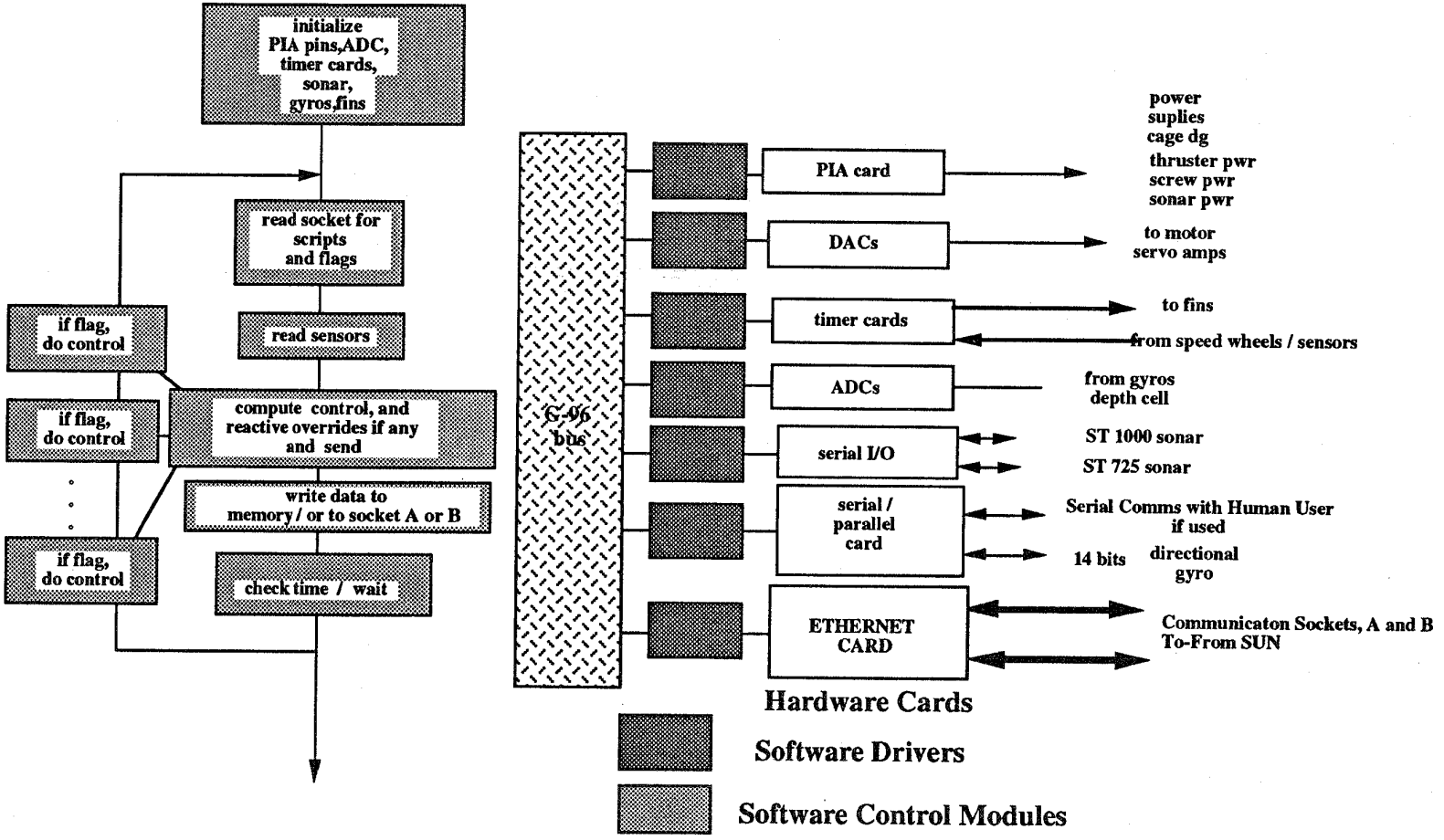

Diagram of the Software / Hardware Interface of the Execution Level of the NPS AUV II

Figure 3 The Structure Of The Execution Level Software

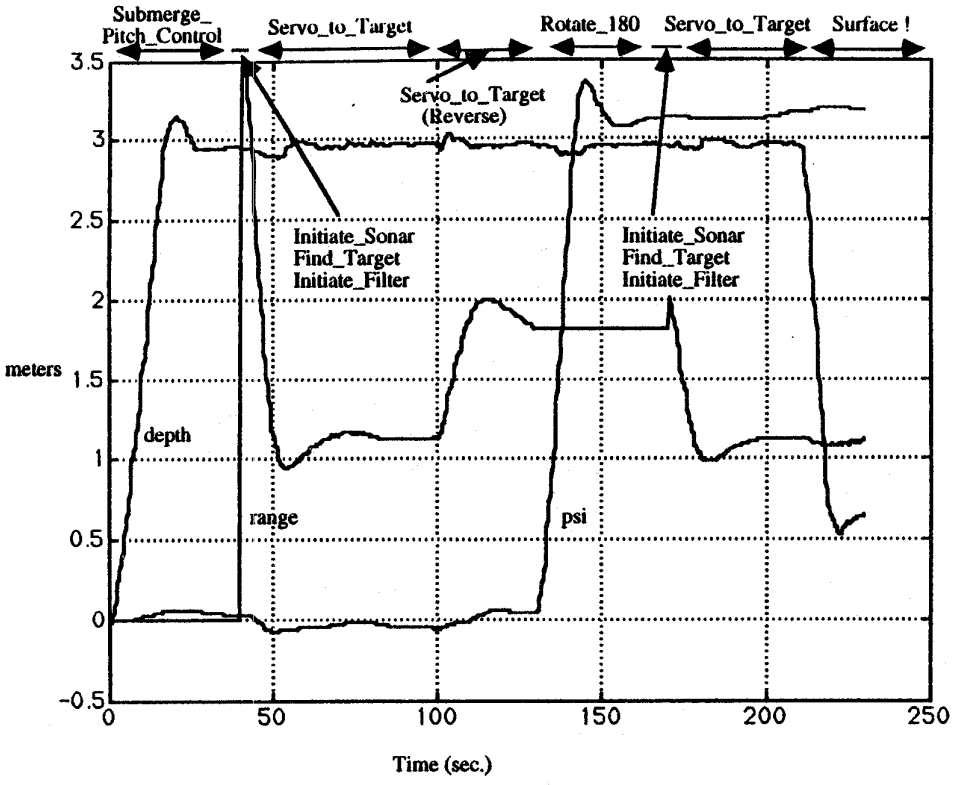

Figure 4 Mission Results [12]