NISTIR 4875

A SUBMARINE SIMULATOR DRIVEN BY A HIERARCHICAL REAL-TIME CONTROL SYSTEM ARCHITECTURE

Hui-Min Huang
Ron Hira

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Unmanned Systems Group
Robot Systems Division
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

Philip Feldman

Advanced Technology &
Research Corporation
Laurel, MD 20707

NIST

NISTIR 4875

A SUBMARINE SIMULATOR DRIVEN BY A HIERARCHICAL REAL-TIME CONTROL SYSTEM ARCHITECTURE

Hui-Min Huang
Ron Hira

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Unmanned Systems Group
Robot Systems Division
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

Philip Feldman

Advanced Technology &
Research Corporation
Laurel, MD 20707

July 1992

U.S. DEPARTMENT OF COMMERCE
Barbara Hackman Franklin, Secretary

TECHNOLOGY ADMINISTRATION
Robert M. White, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

A SUBMARINE SIMULATOR DRIVEN BY A HIERARCHICAL REAL-TIME CONTROL SYSTEM ARCHITECTURE

Hui-Min Huang and Ron Hira
Robot Systems Division
National Institute of Standards & Technology
Gaithersburg, MD 20899

Philip Feldman
Advanced Technology & Research Corporation
Laurel, MD 20707

ABSTRACT

The Robot Systems Division (RSD) at the National Institute of Standards and Technology (NIST) has been developing a generic reference model architecture for intelligent control, known as the Real-time Control System (RCS), for the last two decades. Much of the previous work has been in the of area industrial robots and autonomous vehicles, which led to the understanding that simulation and animation are an integral part of control system design. This paper illustrates the use of simulation and animation in an RCS design for submarine automation. The automation of submarine operations involves complex system functionalities and requires an enormous amount of intelligence to be built into the software to enable a submarine to operate in an unstructured and often hostile environment semi-autonomously. Visualization provides the designer immediate feedback of his or her design. This paper describes an example of fusing simulation and animation with RCS.

1. INTRODUCTION: THE SUBMARINE AUTOMATION PROJECT

The Robot Systems Division (RSD) of the National Institute of Standards and Technology (NIST) has been supported by the Defense Advanced Research Projects Agency (DARPA) Submarine Technology Program (STP).* Under this program NIST and the Advanced Technology and Research Corporation (ATR) are collaborating on the development of a series of demonstrations and a software architecture for submarine automation (Quintero and Barbera 1992; Huang and Hira 1992) based on the hierarchical real-time control (RCS) architecture (Albus 1991, Barbera *et al.* 1984). This paper reports current achievements with emphasis on two elements in the total software architecture; specifically, simulation and animation.

2. THE PHYSICAL SYSTEM AND ITS OPERATIONS

This project utilizes a SSN 637 class nuclear powered submarine data model to demonstrate submarine automation implemented with the RCS reference model architecture. A steam turbine drives a propeller system to provide the main thrust for the ship. Electro-hydraulic activated control surfaces, including a rudder system, a set of stern planes and a set of fairwater planes, provide maneuvering control. Depth control may be achieved by regulating the control surfaces or the buoyancy via various ballast and trim tanks. Trim tanks exist at both the bow and stern of the ship. By pumping water between these tanks, the pitch of the

* ARPA Order No. 7829

submarine can be adjusted. Sonar is used by the submarine to interrogate the surrounding environment to find safe passage while maneuvering under ice.

3. THE SCENARIO

Descriptions of typical mission scenarios can be used to highlight the operational problems to be solved and to identify goals for the control system under development. The overall scenario for this project is for a submarine to transit from the port of San Diego to the North Pole to perform a rescue mission. However, the current focus has been to concentrate on maneuvering control for a submarine during under-ice passage (perceived as navigation through the Bering Strait). Sonar data are processed to estimate the ice distribution, from which clear paths are determined. Path commands are broken down into finer detail and carried out by the responsible subsystems with proper levels of authority. At the lowest level, the actuator systems are activated and coordinated to drive the submarine according to the desired goals.

The scenario also includes navigation in stealth mode while encountering sudden salinity perturbations. These salinity aberrations may occur from fresh water runoffs, where rivers of fresh water cause the water density to drop suddenly. A drop in the density of the sea water will cause the ship to have negative buoyancy. The submarine will start sinking, because its weight is now greater than the weight of the water it displaces. Salinity and temperature variations occur frequently under ice which create significant problems related to depth control and signature management (the ability to avoid detection, while the ship is maneuvering).

Simulation of this scenario enabled the demonstration of a number of features of RCS, including human interaction and decision aiding, the results of which are described in (Huang and Hira 1992).

4. A BRIEF DESCRIPTION OF THE DEMONSTRATION SYSTEM ARCHITECTURE

The centerpiece of the computer demonstration system architecture is referred to as the Real-time Control System (RCS) architecture originated at NIST and developed by Albus (Albus et al. 1982) and Barbera (Barbera et al. 1984). RCS features a hierarchical software structure. Controllers are employed at each level, each controller containing a behavior generation (previously called task decomposition) function which performs decision making and execution processes. This function is supported by the required sensory processing and world modeling functions to form closed-loop control.

The execution of RCS is based on the decomposition or partitioning of system goals into a series of intermediate sub-goals. This process is performed by all the controllers and is referred to as task decomposition (Huang and Hira 1992; Huang, Quintero, and Albus 1991). The real-time aspect of control involves each controller sampling inputs and computing a very limited set of outputs to approach the sub-goals during each cycle of execution. This implementation employs a clock to regulate the execution cycle to satisfy system stability and response criteria.

Figure 1 shows two implementation examples (Depth and Dive/Rise) of RCS generic controllers and the shared memory communication. An RCS controller decomposes an input command into a set of simpler output commands for its subordinates. This decomposition is guided by status messages received from: its subordinate(s), the present state of the world, feedback from sensors, and operator input. Each controller: generates output commands to subordinate controllers, reports status to its supervisory controller, makes requests for sensory processing, and updates relevant world model data. This internal processing framework is identical in all controllers. Refer to (Huang and Hira 1992) for detail on generic controller templates and functions.

Each controller is essentially a closed-loop control unit. The integration of the system occurs through a communication mechanism that (using a shared memory buffer structure) moves data sets between modules. This provides two benefits. First, the addition, deletion, or relocation of modules is simplified. Second, the buffering of data in shared memory simplifies the handshaking method needed for communication, shown in Figure 1.

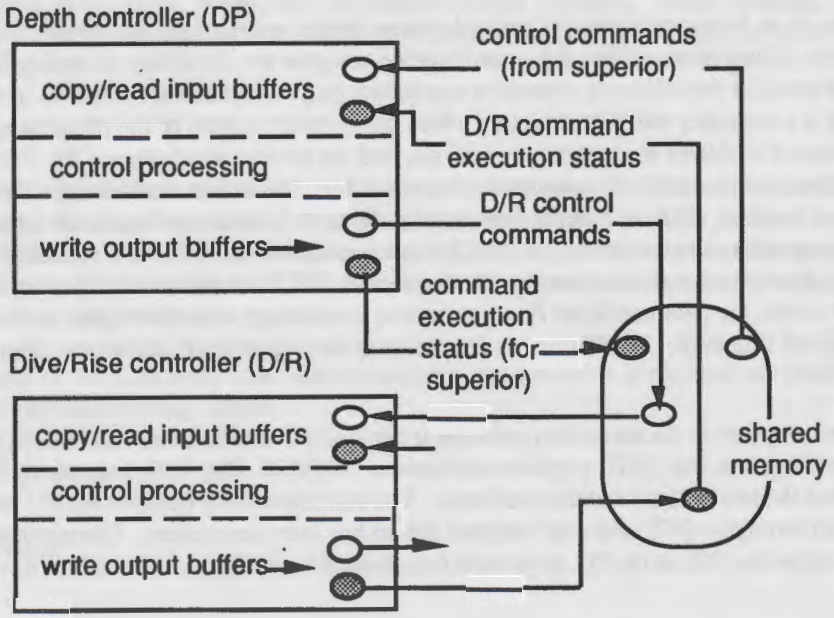

FIGURE 1. RCS Controllers and Shared Memory Communication

Human limitations in the management of information must be considered in the design of large and complex systems. To address this issue, strong emphasis was placed on developing generic processing structures where the operation of all the modules is made to look the same, so that the modules appear to be replicas of one processing format. The overall software structure for this implementation is shown in Figure 2.

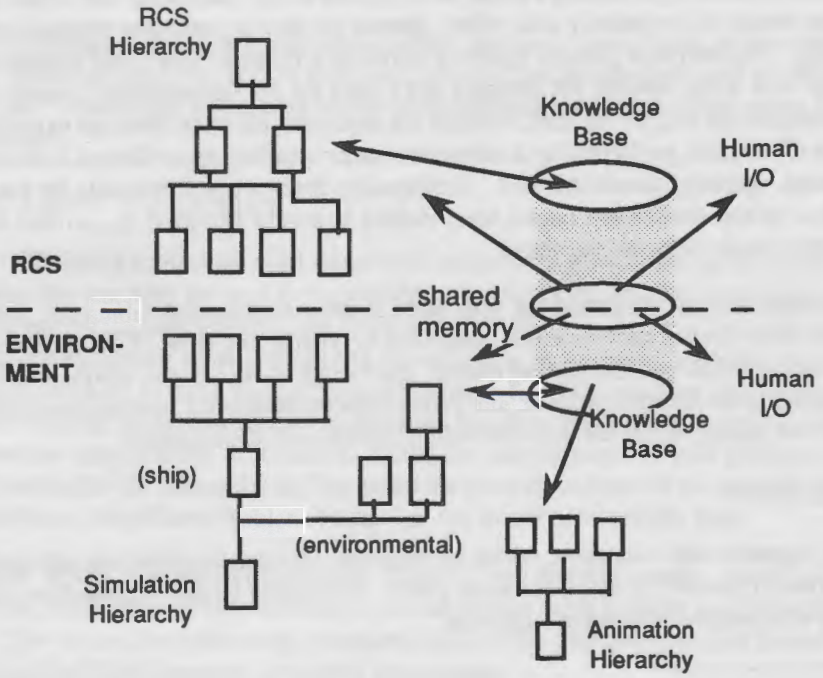

FIGURE 2. Software Structure

5. SIMULATOR STRUCTURE

Simulation is an important aspect of control system design, particularly for complex large scale automation problems. Simulation enables the control system designer the flexibility of testing "what if" scenarios without the need for prohibitively expensive equipment (e.g., a submarine). Parallel to the RCS control structure, there is a simulator software structure which provides simulation of the physical systems and their control interfaces, the objects the systems operate on, and the environment (Figure 2). Simulator modules contain a similar communication framework as the controllers. Note that the simulator hierarchy is drawn from the bottom level up. This is done for two reasons. First, the lowest level actuator simulators possess a one to one correspondence to the lowest level RCS actuator controllers. Second, a consistent flow of dependency is shown from the top of the drawing (the top level of RCS) to the bottom (the top level of simulation). In other words, the command/data flow propagates consistently from the highest level RCS controller to the highest level simulator. In addition, an animation system graphically shows the simulation results in real-time.

The control and part of the simulation software is cyclically executed on an Intel 80386 processor based PC†. A Silicon Graphics, Inc. (SGI) graphics workstation (4D/VGX 220, VME bus based) hosts the animation software and the rest of the simulation software. The communication between the PC bus and the VME bus is facilitated through a BIT3 memory mapped bus to bus interface system. Human interaction may be injected, from either the SGI or the PC, to provide human decision aiding or to introduce environmental disturbances.

6. DOMAIN EXPERT INTERACTION FROM A SYSTEM IMPLEMENTATION PERSPECTIVE

The fundamental issue to be addressed in system development is the understanding of the problem to be solved. It is surprisingly easy to assume that the problem has been completely defined in the specifications. Implementors must understand the problem and interact with submarine operations experts throughout the implementation process, to ensure that the final product meets the actual need.

The domain experts interacted with are ex-submarine commanders. Although they are very knowledgeable, it was not a trivial process to extract the information from them. It would have been futile to present them with the task of furnishing a series of IF-THEN-ELSE statements that could be incorporated into the computer model with relatively little effort. Instead we used an interview process to extract the necessary knowledge. The interview process typically started in a rather arbitrary and unstructured way. The interviews began with some familiar yet arbitrary story lines for the commanders to easily critique and/or extend. The scenarios are stepped through, changes are made and the story lines are extended until a complete description of the tasks performed by a submarine under a variety of conditions is obtained. This was an iterative process, spanning several months. Additionally, the operations manuals for the 637 class submarine, as well as related movies and books, were studied as part of the effort to correlate the scenarios to specific hardware systems on board the vessel.

The information obtained facilitated the generation of the demonstration submarine. Implementing the RCS ship maneuvering system demonstration, described in (Huang and Hira 1992), included: developing a hierarchy, developing a task tree, converting domain knowledge to RCS plans, developing controllers, etc. RCS implementation is an iterative process, and in this implementation the process essentially began at the bottom levels of the system, where the software would interact with the actuators.

7. SIMULATION

Simulation, together with animation, serves an important function by providing safe and cost effective ways to test software (Tarnoff, Jacoff, and Lumia 1992). Simulation feeds the controllers information that they would otherwise receive from actual hardware.

---

† References to company or product names are for identification only and do not imply Government endorsement.

The focus of our work to date has been on the submarine maneuvering system, which involves primarily the simulation of the helm, depth, and propulsion control functions. Other systems, including: trim, hover, hydraulic, compressed air, and ventilation have been simulated. However, the latter systems are not all fully integrated.

7.1 Actuators

At the bottom level, the simulator units are comparatively simple. The rudder simulator detects the amount of voltage that its corresponding controller is sending. The simulator looks at its previous state (which was stored from the last execution time in shared memory), determines the next state, converts that into encoder counts, and places the result into the area in memory where the controller would expect to find it as a sensor input signal on the next cycle.

Simulations of the following ship maneuvering system actuators have been emphasized because of their importance to maneuvering control:

- Main Ballast
- Stern Planes
- Fairwater Planes
- Rudder
- Main Propulsion Turbines.

7.2 Submarine Dynamics

A ship simulator was implemented at the next higher level, which integrates all of the low level actuator simulations. As mentioned in Section 4, in simulation, commands do not move down from a higher authority through successive layers until the actuator is reached as they do in the control hierarchy. Rather, information from the simulators across the bottom level move upwards in abstraction through successive layers of integration, also including the "simulated environment" that surrounds the system; specifically in this case, the ocean, sea floor and ice keels (layers of sea ice built up underwater as a result of water flow).

In the ship simulator, submarine dynamics including: position, speed, depth, bubble angle, and heading are computed. There are two ship depths simulated, namely, the actual depth and the depth that the ship depth sensor reads. The former is regarded as the environmental simulation, whereas the latter is regarded as the ship simulation.

7.3 CMAC Applied To Ice Mapping

In order to efficiently control the submarine while maneuvering under ice and to make intelligent path planning decisions for reaching the goal point, a mapping of the ice distribution is required, see (Huang and Hira 1992). The map needs to be updated and interrogated on-line in real time. The submarine has fourteen forward looking ice avoidance sonar beams, one upward sonar to measure the depth below the ice on the surface, and one downward looking sonar to measure distance from the sea floor (see section 7.5). Only the fourteen forward looking beams have been implemented for this software demonstration. The ice map may be updated, from the current sonar information. Using the current map, the path planning algorithm computes the best heading for the submarine and transmits the recommendation to the control system. However, mapping the ice poses a significant problem because of the sparse nature of the data.

A Cerebellar Model Articulation Controller (CMAC) type neural network, created as a model of the human cerebellum, see (Albus 1975), was used in the submarine work in order to efficiently build a map of the ice profile. The ice profile is then used in conjunction with the goal point for path planning. The following describes how the CMAC network is utilized for mapping.

CMAC has a number of beneficial functions for mapping sparse data. First, as with most neural networks, CMAC is a generalizer, or pattern classifier, as shown in Figure 3. In other words, a *region* of input space (a shaded square), $\Re^n$ is mapped to a *single* output which points to an address in an array. The generalization (multiple points in a region are mapped to a single output) is achieved by use of a hashing function. The value placed in the array cell is determined by a training function.

The submarine sonar simulation has two phases, the training phase for creating and updating the map, and the interrogation phase for path planning. For this simulation, the input vector to the neural network was the absolute x, y, and z coordinates of the sonar beam and the training vector was a boolean value of *ice-detected* or *no-ice-detected*. The training phase included sequencing through each of the fourteen forward looking sonar beams and updating the map, as shown in Figure 4.

After the map has been trained, or updated, the path planning algorithm may interrogate the CMAC network. The path planning algorithm checks for the existence of ice in a given direction by sending the CMAC network a position in three space. CMAC responds with the contents of the cell pointed to by the network. If the cell has a value of *ice-detected*, then the interrogation is iterated with a new position until a *no-ice-detected* region is located.

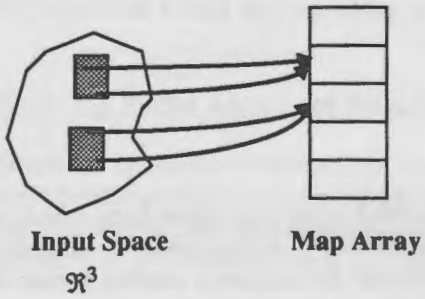

FIGURE 3. CMAC Mapping

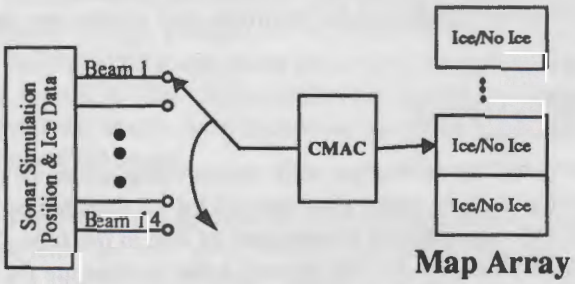

FIGURE 4. CMAC Ice Training

7.4. Environment

The following environmental aspects are simulated: randomly generated ice profile, randomly generated sea floor, and sea water salinity.

The ice keels and sea floor were generated fractally‡ by the use of a mid-point displacement technique. Basically, a grid of points that form a continuous sheet are recursively displaced such that the next level of detail is produced by randomly displacing a new point placed between two old points. This allows the sheet to extend infinitely in all directions with the variation in the sheet limited only by the behavior of the random

number generator used. The properties of the random number generators used were chosen such that the sea floor terrain is smoother than the ice keels.

Sea water simulation includes a density attribute, which allows one to simulate a fresh water runoff as a drop in density. To provide the submarine with changes in salinity, and therefore unforeseeable changes in water density and submarine depth, the environmental simulation needs to be capable of storing salinity data as well as the depth information of the sea floor and ice keels. This is accomplished by adding elements to the arrays that represent the terrain sheets. Additional fields can represent salinity, temperature gradients, etc. In this implementation, the salinity level of the water surrounding the submarine can be altered from the keyboard, allowing an operator to perturb the behavior of the submarine without waiting for it to run into a simulated fresh water pocket.

7.5 Sonars

In the submarine sonar simulation, sonar data are generated by tracing out along the path of each sonar beam, from the submarine's own position into the ice environment. The ice sheet is laid out on a grid, with points spaced at 100 meter intervals. The sonar beams are traced out from their origin and tested for an intersection with the plane formed by the three points in the ice sheet each beam is currently under. The point of intersection is then calculated and tested to see if it lies within the polygon defined by the three points. If it does, this is recorded as a return for that beam. The strength of the return is then determined, based on the angle of interception and the distance of the point of intersection to the submarine. The coordinates of the intersections are then stripped of their height values and used to generate a two-dimensional display that shows the relative position of ice to the submarine's own position and orientation. Sonar data are loaded into the CMAC neural network, which can then be interrogated by the planning algorithm to produce an ice avoidance heading. The CMAC network remembers the position of ice that was detected in previous sonar sweeps, and therefore is able to provide a map to the path planner which can recommend a heading that is less likely to result in a collision than if the sonar information were used raw. This heading is placed into shared memory for use by the control system.

8. ANIMATION

To provide the three dimensional graphical representation of the control and simulation systems, another, entirely separate, software system was implemented on the graphics system (SGI 4D/VGX). The task of the graphics system is twofold: to portray the information generated by the control and simulation systems, and to provide the user with a graphical interface which allows the conditions of the submarine and its environment to be altered during the course of execution.

8.1 Graphics System Software Structure

The graphics system software structure, shown in Figure 5, resembles the control system in utilizing: hierarchical layout, command/status, shared memory, etc. Therefore, the entire graphics system execution can be viewed as another set of control activities. Details for some graphic modules, denoted by asterisks, follow.

---

‡ As Lauwerier (Lauwerier 1991) stated: "A fractal is a geometrical figure in which an identical motif repeats itself on an ever diminishing scale."

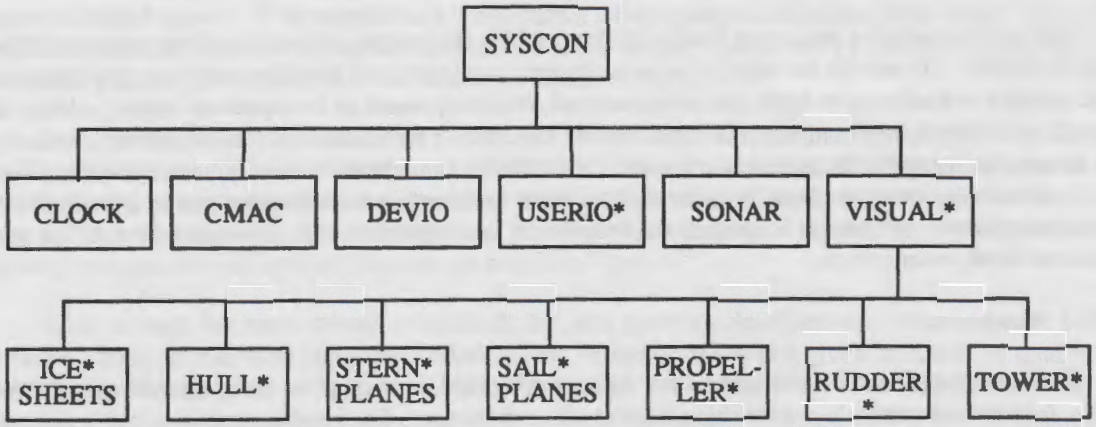

FIGURE 5. Animation Software Structure

VISUAL: This unit concerns itself with the actual drawing of the real-time graphics. As opposed to the behavior of some of the other units in the control and simulation areas, the commands to the visual unit are quite simple, basically INIT (initialize) and RUN. The INIT phase sets up the graphics hardware and initializes the graphics software parameters (defining colors, for example). The RUN command causes the graphics to look at the areas in shared memory that affect the behavior of the graphics. This includes such elements as the submarine orientation and position, rudder, sail plane and fairwater plane positions, and the ice sheet and sea floor arrays of XYZ positions.

USERIO: As with the VISUAL unit, the commands coming into this level are INIT and RUN. In this unit however, the tasks are to take keyboard input and store it in the command buffers or global memory for the proper graphics software units to execute. This is limited to mouse interaction for eyepoint movement, or to adjust salinity values during the simulation under normal operations. When debugging, however, there are keyboard equivalents for all aspects of submarine behavior. This allows the submarine to be "flown" manually through given test sequences.

Draw Submarine (Under VISUAL, comprises: HULL, STERN and SAIL PLANES, PROPELLER, RUDDER, and TOWER): The data describing the submarine hull contour and control surfaces are prestored. The hull is made up by connecting the "bulkheads" of the submarine together with cylinders. Each bulkhead has the same number of vertices, so a single graphics drawing routine can handle them all. During each execution cycle, the ship position, orientation, speed, and viewpoint input (from keyboard or mouse) are used to compute and display the entire scene; i.e., the submarine in a correct view and location relative to the environment (including the ice and the sea floor).

The fastest drawing routine in the SGI graphics library, tmesh (Silicon Graphics Computer Systems 1990), is used to draw the hull. In execution, the hull contour data are accessed one subset at a time until completed. They are used as the vertices for drawing the tmesh strips to be wrapped around the cylinders. The shading of the submarine is handled by assigning each vertex a specific shading value, determined by incrementing and decrementing a shading parameter.

Draw Ice Sheets (Under VISUAL): Ice sheets are drawn from the vertices of the fractally computed ice keel segments. The color of the ice at any given point is determined by two variables. The first is the thickness of the ice. The further down (i.e., larger negative value) the vertex is, the darker the color drawn. The second factor is the distance from the eyepoint. The color of the ice (and sea floor) is incrementally converted to the color of the infinite background as a function of distance. In other words, at a distance of 0.0 m, the color of the object would be the color of ice. At a distance of halfway to the maximum viewing

distance, the color would be half background and half ice. At the maximum distance, the color is purely that of the infinite background. Past this point, the terrain is not drawn, since it would not be seen. It is calculated though, as can be seen by moving the eyepoint towards terrain which has yet to be rendered. Note that the sonar and neural net still function normally.

SYSCON: The top coordinating unit for the graphics system. Again, since the purpose of the graphics is primarily to portray visually data that have been generated in the simulation, the coordinating aspects are limited mostly to the initialization process.

8.2 Interaction With The Graphics System

Information comes into the graphics system from three areas:

- The shared memory exchange between the control system and the graphics system, including the ship position and orientation. Using the information obtained, the system constructs a snapshot of the simulation at any given instant in time.
- The keyboard is sampled in a similar manner to the shared memory exchange. For example, entering CTRL-A places the command into the system to abort the graphics program, which provides orderly termination of the software. Each unit terminates and reports back to its superior, until the entire system is halted.
- The mouse XY position and button status are integrated with a graphical user interface (GUI) menu and eyepoint position selection. The user, by holding the left mouse button down, has complete freedom to point the eyepoint in any direction. If, instead, the right button is pressed with the cursor over a GUI slider control, the value represented by the position of the slider is placed in shared memory as a user request to modify a value in the environmental simulation. The control system then acts on this environmental change, and modifies the behavior accordingly.

An advantage of sampling shared memory between the simulation and the graphics becomes apparent in simulation speed. The simulation and control cycle is set at 30 milliseconds, and runs on a computer other than the graphics system - in this case a 386 PC. On the other hand, the graphics system may take more or less than that amount of time to draw a given image. This variation is due to the number of polygons in the scene and the size of these polygons. Typically, the graphics refresh rate is 80 to 125 milliseconds. The control and simulation systems do not get any animation information from the graphics system. Therefore, there is no need for the control and simulation systems to wait for the graphics to finish drawing a picture to the screen before they go to the next cycle. Rather, the graphics system draws a picture of what is currently in memory. In this way, the simulation and control systems can run as fast as necessary, in order to maintain simulation fidelity and control response, while the graphics lags behind at most one screen refresh cycle.

8.3 Animation Displays

The animation consists of the following:

- submarine model with all the control surfaces,
- ice sheets and sea bottom,
- current sonar data display, and
- an ice distribution and path suggestion display.

These images require that the computer generate a high level of scene density. Where possible, graphic conditions are precalculated. The submarine, sea floor, and ice shading are precalculated using a simplified radiosity algorithm**. Depth cueing is accomplished by progressively subtracting the difference between

** Radiosity algorithms are rendering techniques that "compute the global interreflection of light in an environment composed of diffuse surfaces," (Sillion 1991).

vertex elements (ice, sea floor, and submarine colors) and the background color (dark blue) as a function of distance. In the case of this system, it was decided that we should actually be able to see under water, and so the transparency of the water and the light transmitted by the ice are wildly unrealistic.

An example animation screen is portrayed in Figure 6.

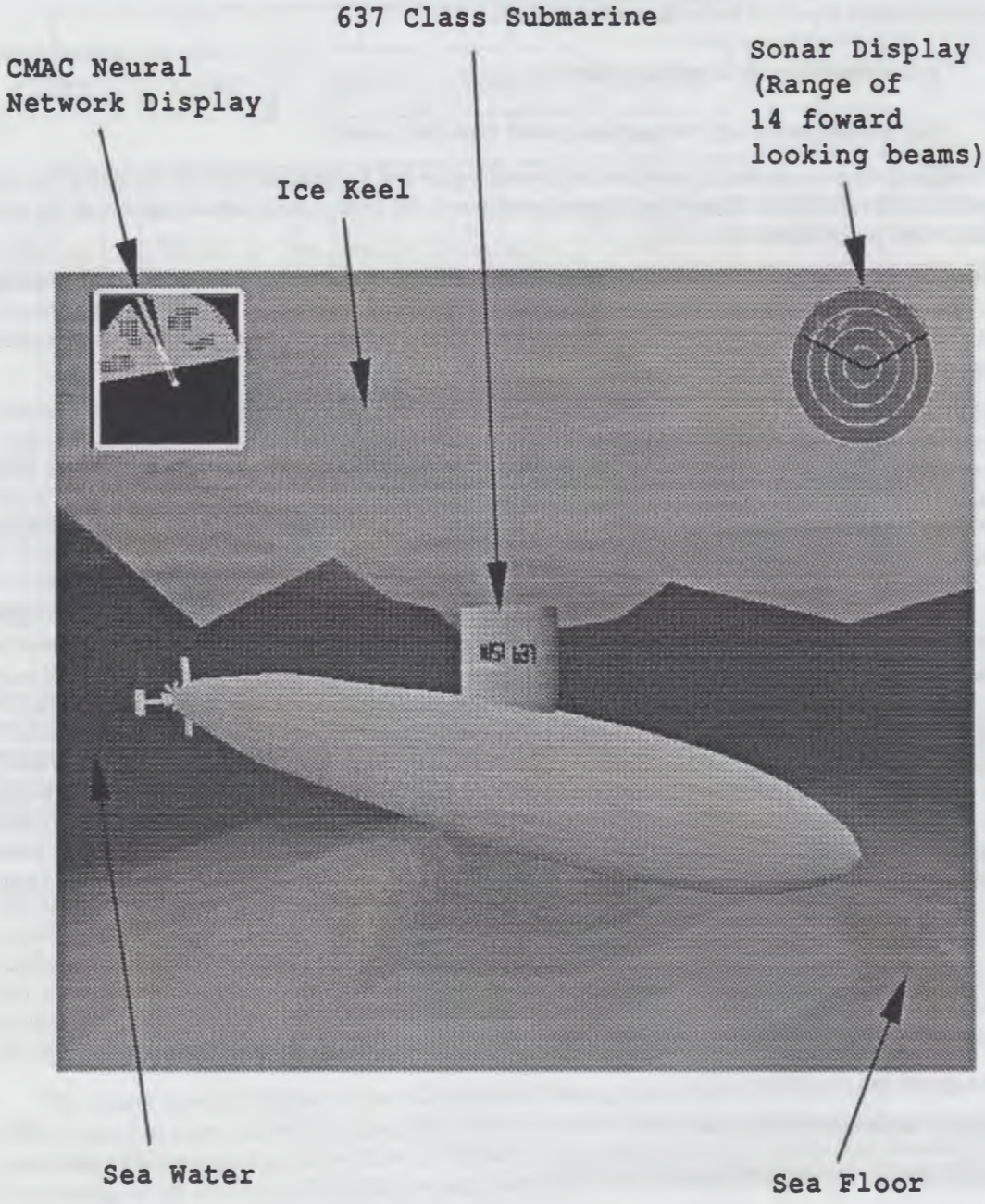

FIGURE 6. Example Animation Display

9. SOFTWARE DEMONSTRATION

A user inputs a system level command RUN_MISSION to the RCS to start the operation. All the controllers and the simulation modules are executed cyclically. The submarine essentially moves ahead with a pre-specified under-ice maneuvering speed and maintains a given depth. The submarine transits toward its goal position while performing ice avoidance. The simulated ship maneuvering data are sent to the graphics workstation through shared memory for display.

The user can interact with the environment by changing the salinity, the ship control surface positions, etc.

The controllers can detect and flag severe errors, e.g., sudden fluctuations in water density. With the RCS running in interactive mode (Huang and Hira 1992), warnings can be displayed on the graphic workstation together with a list of options for error recovery. The user can click on the selection and the data will be sent back to the controllers for execution. The animation shows the effect of the selected error recovery option.

10. CONCLUSION

We have described the implementation of the simulation and animation software for a submarine maneuvering system demonstration. Simulation and animation are two important components in the total Real-time Control System (RCS) methodology for the development of intelligent systems. Simulation and animation software interact with the control system in real-time to facilitate the conceptualization and testing of the control system under development. In the next phase of this project, work will continue in enhancing the simulation and animation systems in the following areas: higher fidelity ship dynamic simulation, better graphic user interface (GUI) capability, and more useful graphic displays such as depth profile and ship diagnostic status displays.

ACKNOWLEDGEMENTS

The authors extend their appreciation to Mr. Richard Quintero of NIST, Dr. Anthony Barbera, Ms. M.L. Fitzgerald, Mr. Clyde Findley, Mr. Nat Frampton, and Mr. Mark Routson of the Advanced Technology and Research Corporation for their participation in various aspects of this project.

REFERENCES

Albus, J.S. 1975. "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)." *Transactions of the ASME*, (Sep.): 220-227.

Albus, J.S., McLean, C., Barbera, A., and Fitzgerald, M. 1982. "An Architecture for Real-Time Sensory-interactive Control of Robots in a Manufacturing Environment." In *Proceedings of the 4th IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology* (Gaithersburg, MD, Oct. 26-28).

Albus, J.S. 1991. "A Theory of Intelligent Systems." In *MANUFACTURING AND AUTOMATION SYSTEMS: TECHNIQUES AND TECHNOLOGIES, CONTROL AND DYNAMIC SYSTEMS, ADVANCES IN THEORY AND APPLICATIONS*, Volume 45, Academic Press.

Barbera, A.J., Fitzgerald, M.L., Albus, J.S., and Haynes, L.S. 1984. "RCS: The NBS Real-Time Control System." In *Proceedings of the Robot 8 Conference and Exposition, Volume 2 - Future Considerations*, (Detroit, MI, June 4-7).

Huang, H. and Hira, R. 1992. "Applying the NIST Real-time Control System Reference Model to Submarine Automation: A Maneuvering System Demonstration." NIST Interagency Report, to be published. NIST, Gaithersburg, MD.

Huang, H., Quintero, R., and Albus, J.S. 1991. "A Reference Model, Design Approach, and Development Illustration toward Hierarchical Real-Time System Control for Coal Mining Operations." *MANUFACTURING AND AUTOMATION SYSTEMS: TECHNIQUES AND TECHNOLOGIES, CONTROL AND DYNAMIC SYSTEMS, ADVANCES IN THEORY AND APPLICATIONS*, Volume 46, Academic Press.

Lauwerier, H. 1991. *Fractals*, Princeton University Press, Princeton, NJ.

Quintero, R., Barbera, A.J. 1992. "An RCS Methodology for Developing Intelligent Control Systems." NIST Interagency Report, to be published. NIST, Gaithersburg, MD.

Silicon Graphics Computer Systems, 1990. *Graphics Library Reference Manual, C Edition*, Silicon Graphics, Inc., Mountain View, CA.

Sillion, F. 1991. "Detection of Shadow Boundaries for Adaptive Meshing in Radiosity." In *Graphics Gems* II, Arvo, J., ed. Academic Press, Inc., San Diego, CA, 311-315.

Tarnoff, N., Jacoff, A., Lumia, R. 1992. "Graphical Simulation for Sensor Based Robot Programming." *Journal of Intelligent and Robotic Systems*, Vol 5, 49-62.

NIST-114A
REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4875

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
JULY 1992

TITLE AND SUBTITLE

A SUBMARINE SIMULATOR DRIVEN BY A HIERARCHICAL REAL-TIME CONTROL SYSTEM ARCHITECTURE

AUTHOR(S)
Hui-Min Huang, Ron Hira and Philip Feldman

PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

Advanced Technology &
Research Corporation
Laurel, MD 20707

7. CONTRACT/GRANT NUMBER
ARPA Order No. 7829

8. TYPE OF REPORT AND PERIOD COVERED
Nov '90 to Mar '92

SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Defense Advanced Research Projects Agency
3701 N. Fairfax Drive
Arlington, VA 22203-1714

SUPPLEMENTARY NOTES

ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Robot Systems Division (RSD) at the National Institute of Standards and Technology (NIST) has been developing a generic reference model architecture for intelligent control, known as the Real-time Control System (RCS), for the last two decades. Much of the previous work has been in the of area industrial robots and autonomous vehicles, which led to the understanding that simulation and animation are an integral part of control system design. This paper illustrates the use of simulation and animation in an RCS design for submarine automation. The automation of submarine operations involves complex system functionalities and requires an enormous amount of intelligence to be built into the software to enable a submarine to operate in an unstructured and often hostile environment semi-autonomously. Visualization provides the designer immediate feedback of his or her design. This paper describes an example of fusing simulation and animation with RCS.

KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
animation, automation, intelligent control, neural networks, simulation, software architecture

AVAILABILITY

[ ] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[ ] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
15

15. PRICE
A02

ELECTRONIC FORM