GRAPHICAL SIMULATORS FOR AUV DEVELOPMENT[1]

P. Ridao, M. Carreras, D. Ribas and A. El-Fakdi

Institute of Informatics and Applications. University of Girona. Spain

{pere,marcc,dribas,ael-akdi}@eia.udg.es

ABSTRACT

A long development time is needed from the design to the implementation of an AUV. During the first steps, simulation plays an important role, since it allows for the development of preliminary versions of the control system to be integrated. Once the robot is ready, the control systems are implemented, tuned and tested. The use of a real-time simulator can help closing the gap between off-line simulation and real testing using the already implemented robot. When properly interfaced with the robot hardware, a real-time graphical simulation with a "hardware in the loop" configuration, can allow for the testing of the implemented control system running in the actual robot hardware. Hence, the development time is drastically reduced. This paper overviews the field of graphical simulators used for AUV development proposing a classification. It also presents NEPTUNE, a multi-vehicle, real-time, graphical simulator based on OpenGL that allows hardware in the loop simulations.

1. INTRODUCTION

The use of simulation as a tool for UUV development plays an important role at different phases of the development process. At the first steps of the design, *offline simulators* are very helpful. MATLAB/Simulink is a good tool for such kind of simulations. Nowadays there are several toolboxes available to be used for marine robots, including UUVs [8]. Although these toolboxes are very useful for the simulation of robot models and control systems, they don't reproduce the external world and hence, external sensors like sonar cannot be modeled. Therefore, intelligent control architectures which relay on the robot-environment interaction, cannot be simulated. In [5], a MATLAB/Simulink framework including world modeling is used for the comparison of different intelligent control architectures for AUVs. Although the framework allows for the sonar simulation, other external sensors like computer vision cannot be reproduced. In [11], a MATLAB/Simulink framework is used for the simulation of a system for fault diagnosis and recovery applied to ROVs. The system uses OpenGL 3D graphics for world representation. Although not used for computer vision simulation, the system is able to reproduce different views including those of the onboard cameras. Other researchers [1] have developed a 3D simulator using C++ which allows for the simulation of sonar and vision. Using the texture mapping capability of OpenGL, they project an image mosaic of the real environment onto the bottom surface. This allows for a quite realistic simulation of vision systems. In the offline simulation, 1 second of simulation doesn't last for 1 second of the reality. In some cases, the simulated time is much smaller than the real time [1]. This is of particular interest, for instance, when algorithms like GAs are used, since it allows the system to find a fast solution. In other cases [11] the simulation load is so heavy that 1 second of simulation last for more that 1 second in the reality. In both cases, the temporal properties of the implemented algorithms are not taken into account. Hence, when the code is transferred to the actual robot, the temporal consistency must be checked for correctness.
The *online simulators* on the other hand, ensure time consistency between the simulated and the real time. Hence, the time properties of the simulated algorithm are taken into account within the simulation. Online simulation play an important role in UUV development since normally only few prototypes are available for development and testing. With online simulators it is possible to achieve concurrent engineering. Hence, different engineers are able to develop in parallel different aspects of the control systems on a totally virtual reproduction of the actual robot. Nevertheless, the algorithm is not executed in the actual hardware of the robot and therefore, the time behavior of the computer used for the simulation can be different from the one that

[1] This research was sponsored by the Spanish commission MCYT (DPI2001-2311-C03-01).

will be used during the robot control. *Hardware in the loop (HIL)* simulators are used for overcoming such an inconvenience. In this case, the developed software is executed on the actual robot hardware but the robot actions are routed towards the simulator instead than to the real actuators. In the same way, the sensor readings are simulated from the outputs of the online simulator. In all these sort of simulation systems, the fidelity of the simulation depends on the accuracy of the robot model, the world model and the sensor models being used. Nevertheless, even complex models are ideal reproductions of the observed reality. Although some researchers have introduced error models which are added to the output of the virtual sensor, there still is a gap between the simulated and the real mission. A step forward to the real execution performance is achieved with the use of *hybrid simulators (HS)*. A HS is a HIL simulator where the real and virtual systems operate together in an augmented reality environment. Then, it is possible to simulate a mission where a robot navigates in a water tank while using virtual sonar to avoid virtual objects. HS are of particular interest to test the expected performance that could be achieved when new sensors have to be acquired. This approach has been used by different authors like [10] and [7] among others. Once the whole system is ready to work, 3D graphical simulations can still be used to operate, supervise and monitor the current operation of the vehicle [10,7], or even for mission playback and post mission analysis.

This paper overviews the field of graphical simulators for AUV development and presents a new simulator called NEPTUNE. The paper is organized as follows. Section 2 presents the models used for robot and thrusters simulation. Then, Sections 3, 4 and 5 present the models used for world, environment and sensor simulation. Section 6 presents NEPTUNE and Table 1 proposes a classification. Finally the paper concludes in section 7.

2. UUV MODEL

The robot model estimates the robot movement as response to a given input. Depending on the simulation needs, different kind of models can be used. For guidance simulations a kinematics model can be enough:

$$ {}^{E}\dot{\eta} = J\left({}^{E}\eta\right){}^{B}\upsilon \; ; \; {}^{E}\eta = \int {}^{E}\dot{\eta}dt \qquad (1)$$

This formula and the formulas hereafter are represented with the same standard notation used in [8]. Please, refer to that reference for details about the formulas as well as the meaning of the variables. For more realistic control simulation a full hydrodynamics model is commonly used:

$$\begin{aligned} &{}^{B}\tau + {}^{B}\tau_{E} = \left({}^{B}M_{RB} + {}^{B}M_{A}\right){}^{B}\dot{\upsilon} + \\ &\left({}^{B}C_{RB}\left({}^{B}\upsilon\right) + {}^{B}C_{A}\left({}^{B}\upsilon\right)\right){}^{B}\upsilon + {}^{B}D\left({}^{B}\upsilon\right){}^{B}\upsilon + {}^{B}G(\eta) \end{aligned} \qquad (2)$$

In the second case, the model input is the force and torque exerted by the thrusters. This force can be modeled using a steady state model:

$$T = C_{T_{|\omega|\omega}} |\omega|\omega + C_{T_{|\omega|V_a}} |\omega| V_a \qquad (3)$$

In many practical applications, the advance speed at the propeller V_a can be considered *0* and hence, the thruster affine model is obtained:

$$T = C_T |\omega|\omega \qquad (4)$$

In [13] the authors show how important thruster dynamics is for hovering maneuvers. For this kind of simulations, dynamics models are needed. The same paper evaluates three dynamics models for electrically actuated motors. Refer to it for details.

3. WORLD MODELS

When the robot moves through the underwater world its sensors are used for sensing in order to decide how to act. To be able to simulate this behavior, a world model is needed. The topographical model of the world has two goals: (1) to represent the virtual world in the computer screen and (2) to act as the input to the virtual sensors. There are two principal methods for world encoding: (1) bathymetry and (2) 3D CAD-like models. A bathymetry model is an elevation map consisting on a grid of altitudes [12]. It is a way to encode a level curves map. Any surface that can be represented with a two variables function *h(x,y)*, can be represented by this kind of map. In fact, in [2] authors used mathematical functions to encode the bathymetry model. Nevertheless, bathymetry models cannot be used to represent some natural features like caves or some artificial structures that could be located on the ocean bottom. The other way to represent the virtual world is the use of CAD files. Most of the current 3D graphical packages are able to export the designs into VRML format. Hence, if this format is adopted it is very easy to edit new worlds [3]. Moreover, there are several libraries freely available able to parse the VRML language and represent the objects using OpenGL. Some times, both models are used together. This is the case of NEPTUNE (Section 6).

4. ENVIRONMENT MODELS

There are several physical variables that define the state of the environment: waves, wind, currents, temperature, and salinity. Temperature, for instance, has an important impact in the acoustics since the sound speed depends on it. Salinity, magnetic and termocline models have been reported in [6] while wave models have been used in

Reference	World Model	Environment Model	Models					Simulation	Distribution	Graphics	Multivehicle	Field robots
			Sensor									
				External								
			Internal	Sonar	Vision	UUV	Thruster					
[1]	N	N	Y	G_c	Y	H	A	OFF	N	3D	N	GARBI
[2]	B	N	Y	N	N	H	N	OFF	N	N	Y	PHOENIX SUV
[3]	BO	N	Y	G_b	N	H	A	On Hil P	Y	3D	N	PHOENIX
[4]	N	N	Y		N	H	A	Hil	Y	3D	N	ROMEO
[6]	B	SMTIC	N	N	N	N	N	Hil	Y	3D	Y	EAVE EST
[7]	N	C	Y	G_b	Y	H	A	HIL M	Y	3D	Y	ODIN SAUVIM
[9]	O	N	Y	N	N	H	N	On HIL	Y	3D	Y	PHANTOM
[10]	N	N	Y	G_b	N	H	N	On Hil Hs P	Y	3D	Y	Twin Burger PTEROA MANTA
[12]	B	CW	Y	N	N	D	S_s	On Hil	Y	3D	Y	OEX
[13]	B	MCWTD	Y	N	N	D	N		Y	3D	Y	Sea Squirt
NEP-TUNE	BO		Y	G_c	Y	D	A	On Hil Hs	Y	3D	Y	URIS GARBI

B: Bathymetry; O: set of 3D object models; I: Ice covert topography; H: hydrodynamics; S_s: Steady State; A: thruster affine model; S: Salinity field; M: Magnetic field; T: Temperature field; D: Density; C: Currents; W: Waves; On: Online Simulation; Hil: Hardware in the loop Simulation; Hs: Hybrid Simulation; P: Mission Playback; M: Online Monitoring; G_b: Single beam; G_c: conic beam; N: Not detailed;

Table 1 Classification.

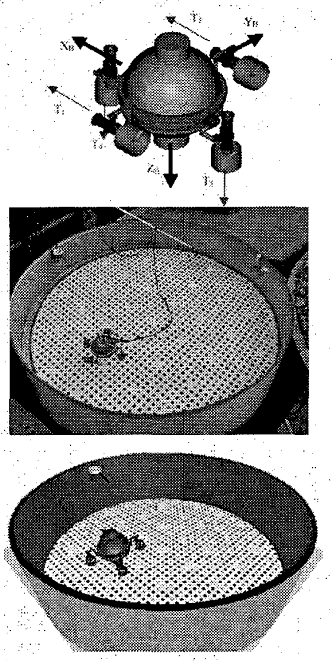

Fig. 1 From top to bottom: URIS robot; Lab Setup for URIS; Virtual World for URIS.

[12]. Wind and waves forces affects only to the vehicles in the surface. Hence, for underwater robots, currents play the major role. For computer simulations is realistic enough [8] to simulate currents as a random walk process.

5. SENSOR MODELS

UUV sensors can be classified into three categories: (1) internal sensors, (2) external sensors and (3) mission payload sensors. Internal sensors include the sensors used for measuring the state variables of the UUV as well as their integrals and derivatives. A common way to model these sensors is to use the correspondent output of the UUV model, sampled at the same frequency that the real sensors works, limited to the range of the sensor and using the same resolution. Some authors introduce noise to the measurement in order to do it more realistic. In [12] the authors used a magnetic deviation lookup table for simulating the compass measurement. They also introduced a bias in the measurement and gaussian distributed white noise. For the modeling of the LBL, they
proposed to use a sound speed profile together with a gaussian ray tracing method. In [4], the authors used a method for estimating the probability density function of the sensor noise after removing the outliers. External sensors are used for sensing the environment. Sonar and vision are examples of this type of sensors. Sonar sensors can be simulated at different levels: (1) using the sonar equation together with an acoustic ray tracing algorithm taking into account the sound speed profile and (2) using a geometrical method. First method is based on the physics of the sound propagation and it is able to reproduce effects like the multipath. The problem is that they are computationally intensive. For range detection in the immediate robot surroundings, a geometric method is very appropriate [3] since the sound speed remains constant. Commonly, geometric methods trace a ray from the sonar transducer to the environment and return the corresponding range [3]. An alternative method, considers each sonar beam as a cone with β degrees of aperture. Points belonging to this cone are explored in order to see if they impact with objects in the vehicle surroundings. In this case, the corresponding range is returned. This is the case of NEPTUNE which is described in section 6. The use of realistic 3D graphical simulations allows nowadays making a simulation of computer vision systems. The virtual views generated by

the 3D graphical engine can be used to grab images. Moreover, using texture mapping it is possible to use real images to generate the virtual scene. See [1] for a nice application to the simulation of a cable tracking mission.

6. NEPTUNE

NEPTUNE is a real-time graphical simulator with capabilities for on-line, hardware in the loop and hybrid simulation. Externally, the virtual world is based on two components: (1) a VRML file containing the topography of the scene and (2) a set of objects also defined in VRML. Internally, the topography of the scene is converted into a bathymetry grid and the objects are considered as spheres of a particular radius of action. This model, together with a conic beam sonar model allows for a very simple and fast geometric method for obstacle and/or collision detection. Of course, there is a basic assumption which considers the bottom surface as convex. NEPTUNE is a multi-vehicle simulator. Hence, more than one robot can be simultaneously simulated. Each robot is defined through three basic files. The first is a VRML file containing the robot geometry (Fig.1). The second is a file which contains the robot and thruster hydrodynamics coefficients. For simulation, the hydrodynamic model (eq.1) is used. Thrusters are simulated using the affine model (eq.4). Finally, the third file contains the file names of the previous two files plus a definition of the sensors included in the robot (number of sonar beams, position in the robot frame and attitude ...). A custom developed language has been used for the definition of the last two files. Then, it is very easy to adapt NEPTUNE to simulate new robots. At this moment, simulated sensors include: (1) range detection sonar, (2) vision and (3) internal sensors (position, attitude, speed, depth,). Since this is an ongoing project, ocean currents and waves are no yet supported, although we plan to introduce them in next versions. In order to allow for a real time performance, the application has been build as a distributed application including several processes: (1) the NEPTUNE main program (2) one robot dynamics process for each simulated robot (3) a name server. All the programs interact among them through a TCP/IP network. In Table 1 the main properties of NEPTUNE as well as the main properties of several studied graphical simulators are reported.

7. CONCLUSSIONS

Graphical simulators play a key role for UUV development. They can perform in different ways (offline, online, hardware in the loop and/or hybrid simulation as well as monitoring and/or mission playback) depending on the application. They also differ in the sort of UUV, world, environment and sensor models they use. NEPTUNE is a real-time 3D graphical simulator for running online, hardware in the loop and hybrid simulations. It is very flexible in the sense that new virtual worlds and new UUV models can be added in a very easy way. UUVs are modeled through their hydrodynamic equation and thrusters are modeled using the affine model. The world is modeled using VRML as well as a bathymetry model and sonar is modeled using a geometric method.

8. REFERENCES

[1] Antich J. and Ortiz A., "Experimental Evaluation of the Control Architecture for an Underwater Cable Tracker", 6th IFAC MCMC, Girona (Spain), pp.140-165, September 2003.

[2] Borges de Sousa J. and Göllü A. "A Simulation Environment For The Coordinated Operation Of Multiple Autonomous Underwater Vehicles", Winter Simulation Conference, Atlanta (USA), pp. 1169-1175, December 1997.

[3] Brutzman D.P., "A Virtual World for an Autonomous Underwater Vehicle", Phd. Thesis, Monterey (USA), Dec. 1994.

[4] Bruzzone Ga., *et al*, "A Simulation Environment for Unmanned Underwater Vehicles Development", MTS/IEEE Oceans 2001, Honolulu (USA), pp. 1066-1072, November 2001.

[5] Carreras M. *et al.*. "An overview on behaviour-based methods for AUV control", 5th IFAC MCMC, Alborg (Denmark), pp. 141-146, August 2000.

[6] Chappell S.G. *et al.*, "Cooperative AUV Development Concept (CADCON) An Environment for High-Level Multiple AUV Simulation", 11th International Symposium on Unmanned Untethered Submersible Technology, Durham (USA), pp.112-120, August 1999.

[7] Choi S.K., *et al.*, "Distributed Virtual Environment Collaborative Simulator for Underwater Robots", IEEE/RSJ Int. Conf. on Robots and Systems, Takamatsu (Japan), pp 861-866, November 2000.

[8] Fossen, T.I, "Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles", Marine Cybernetics AS, Trondheim, December 2002.

[9] Gracanin *et al.*, "Virtual Environment Testbed for Autonomous Underwater Vehicles", Control Engineering Practice, vol. 6, no. 5, pp. 653-660, 1998.

[10] Y. Kuroda, *et al.*, "AUV Test using Real/Virtual Synthetic World", IEEE Symp. on Autonomous Underwater Vehicle Technology, Monterey (USA), pp.365-372, June 1996.

[11] Omerdic, E. *et al.*, "Fault Detection and Accommodation for ROVS", 6th IFAC MCMC, Girona (Spain), pp.155-160. Sept. 2003.

[12] Song F. *et al.*, "Modeling and simulation of autonomous underwater vehicles: design and implementation", IEEE Journal of Oceanic Engineering, Vol. 28, Issue: 2, pp.283-296, April 2003.

[13] Tuohy S. T., "A Simulation Model for AUV Navigation," IEEE Oceanic Engineering Society Conference Autonomous Underwater Vehicles, Cambridge (USA), pp. 470-478, July 1994.

[14] Whitcomb L.L.and Yoerger D. R., "Development, comparison, and preliminary experimental validation of nonlinear dynamic thruster models", IEEE Journal of Oceanic Engineering, Vol. 24, Issue: 4, pp.481-494, October 1999.