NEPTUNE: A HIL Simulator for Multiple UUVs

P. Ridao, E. Batlle, D. Ribas and M. Carreras

Institute of Informatics and Applications. University of Girona,
Campus de Montilivi
Girona, CP:17071, Spain
{pere,bbtalle,dribas,marcc}@eia.udg.es

Abstract **- This paper overviews the field of graphical simulators used for AUV development, presents the taxonomy of these applications and proposes a classification. It also presents NEPTUNE, a multi-vehicle, real-time, graphical simulator based on OpenGL that allows hardware in the loop simulations.**

I. INTRODUCTION

Simulation, as a tool for UUV development, plays an important role at different phases of the development process. At 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 [9]. 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 modelled. Therefore, intelligent control architectures which relay on the robot-environment interaction cannot be simulated. In [5], a MATLAB/Simulink framework including world modelling is used for the comparison of different intelligent control architectures for AUVs . Although the framework allows the simulation of the sonar, other external sensors like computer vision cannot be reproduced. In [13], 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 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 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 [13], the simulation load is so heavy that 1 second of simulation lasts 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 the 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 plays 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 behaviour of the computer used for the simulation can be different from the one that 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 sensors, there is still 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, for instance, to simulate a mission where a robot navigates in a water tank while using virtual sonar to avoid virtual obstacles. 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 [12] and [8] among others.

Once the whole system is ready to work, 3D graphical simulations can still be used to operate, supervise and monitor (OM) the current operation of the vehicle [12,8], as well as for mission playback (P) and post mission analysis. Another common utility is their use for operator training (TR) in commercial ROV systems [18, 19].

This paper overviews the field of graphical simulators for AUV development and presents a new simulator called

0-7803-8669-8/04/$20.00 ©2004 IEEE.

NEPTUNE. The paper is organized as follows. In Section II, the architecture of a generic simulator is described. Section III presents the models used for robot and thrusters simulation. Then, Sections IV, V and VI present the models used for world, environment and sensor simulation. Section VII presents the NEPTUNE simulator. Section VIII presents a case of study, and finally, the conclusions are found in section IX.

II. SIMULATOR ARCHITECTURE

Figure 1 shows the main components of a generic underwater robot simulator. Light grey blocks represent physical components of the robotic system, white blocks represent their virtual counterparts, and dark grey blocks represent shared components working in real and virtual mode.

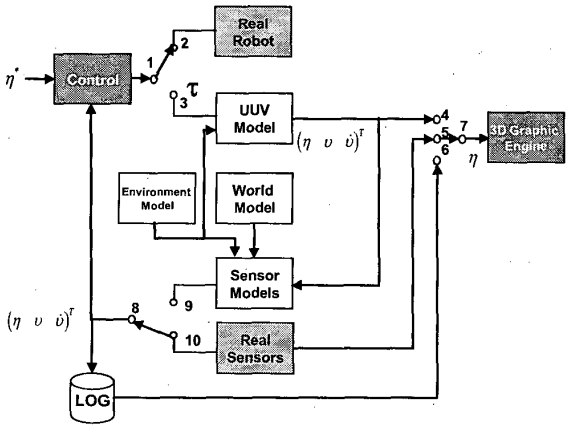

Figure. 1 Block diagram of the robot simulator

Hereafter, the components are briefly described:

- Real Robot: This component represents the real UUV. Its input is the generalized force and torque vector applied by the thrusters and the control planes.
- Real Sensors: This component represents de physical sensors used by the robot to sense its internal state, the state of the environment, or the underwater world.
- UUV Model: Set of nonlinear differential equations that describe the hydrodynamics of the UUV.
- World Model: The underwater world is composed by the topography of the ocean bottom, the presence of other underwater agents (fishes, robots,...) and objects (underwater wrecks, ROV panels, structures, ...). Hence, the world model is composed by a data representation of these elements.
- Environment model: The robot is exposed to different physical phenomenas like tides, currents, waves... which affect its performance in some way. Hence, the goal of this block is to simulate the environmental parameters which are representative for robot operation.
- Sensor Models: The robot state, the robot world and the robot environment are sensed through simulated robot sensors. This block contains the model of these sensors.
- 3D Graphical engine: This block is responsible for the graphical representation of the real/virtual scene of the robot operation.

Depending on the working mode of these blocks as well as their interconnections, different taxonomies can be achieved like: offline (OFF), online (ON), hardware in the loop (HIL) or hybrid simulation (HS), as well as an online monitoring (OM) or operator training (TR). The functionalities of such taxonomies have been introduced in the previous section. Table 1 shows the differences of such configurations in terms of data paths (interconnections of the switches shown in figure 1), relationship of the real and the simulated time, hardware used for the execution of the control software, relationship of the simulated and the real robot software, mission playback and training capabilities.

	Data Paths	Δt=Δk?	Hw	Sw	Sensors	Robot	Playback	3D
OFF	1→3 4→7 9→8	≠	D	D	S	S	No	Yes
ON	1→3 4→7 9→8	=	F	A	S	S	No	Yes
HIL	1→3 4→7 9→8	=	A	A	S	S	No	Yes
HS	1→2,3 4,5→7 9,10→8	=	A	A	S&R	S&R	No	Yes
TR	1→3 4→7 9→8	=	D	A	S	S	Yes	Yes
OM	6→7	=	D	-	-	-	Yes	Yes

Δt=Δk? *- Δt: Simulation time spent by and integration step; Δk: integration step.* ***Hw*** *- D: Robot hw differs from the hw used for the simulation; A: Actual robot hw is used during the simulation; F: a Functionally equivalent hw is used for the simulation.* ***Sw*** *- D: The actual robot software differs from the one used in the simulation; A: Actual robot software is used during the simulation.* ***Sensors/Robot*** *- S: Simulated sensors/robot are/is used; R: Real sensors/robot are/is used.*

Table 1 Simulators Taxonomy.

III. 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. While for guidance simulations a kinematics model can be enough, for more realistic control simulation a full hydrodynamics model should be used. In the second case, the model input

is the force and torque exerted by the thrusters. The thruster models allow the prediction of such force and torque. Hereafter, a brief description of these models is presented.

A. Kinematics Model

Let {E} and {B} be the earth and the body fixed frames respectively, given the robot velocity vector in the body frame the kinematics equation allows the computation of the robot position and attitude as follows:

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

Where ${}^{B}\dot{\upsilon}$, and ${}^{E}\eta$ are the acceleration and the position-attitude vector, and $J\left({}^{E}\eta\right)$ is the kinematics transformation matrix (see [9] for details). With this model the robot is assumed to achieve the desired velocity instantaneously.

B. Hidrodynamics Model

The non-linear hydrodynamic equation of motion of an underwater vehicle with 6 DOF [9], in the body fixed frame, can be conveniently expressed as:

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

where,

- ${}^{B}\tau$ is the force and torque exerted by the thrusters
- ${}^{B}\tau_{E}$ are the environmental forces and torques
- M_{RB} and M_A are the inertia matrix and the added-mass matrix
- ${}^{B}\dot{\upsilon}$ is the acceleration
- C_{RB} and C_A are the rigid-body and the added Coriolis and centripetal matrixes
- D is the damping matrix
- ${}^{B}G$ is gravity and buoyancy force and torque

Equation (2) relates the forces exerted on the UUV with the acceleration and the velocity experimented by the vehicle. In other words, if the forces acting on the robot are known, equation (2) allows the estimation of the robot acceleration and, through integration, the computation of the velocity vector. Once the robot velocity is known, equation (1) can be used for estimating the position and attitude.

C. Thruster Models

Thruster models are used for predicting the force exerted by the thrusters. Some of them are also able to estimate the torque. Hereafter, the main steady state and dynamic thruster models are presented.

i) Steady State Models

Using the steady state models, the thruster is considered to respond instantaneously with respect to its input. Two main models can be distinguished: (1) the *bilinear model* and (2) the *affine model*. The bilinear model is a nonlinear function that computes the thrust as a function of the angular speed of the propeller (ω) and the linear speed of the robot in the thruster direction (υ):

$$ T = C_{T_{|\omega|\omega}}|\omega|\omega - C_{T_{|\omega|\upsilon}}|\omega|\upsilon \qquad (3) $$

For low speed robots, the second term is often neglected and the previous model becomes the so called affine model:

$$ T = C_{T}|\omega|\omega \qquad (4) $$

ii) Dynamic Models

In [17] the authors review 3 different dynamical models for electrically driven motor thrusters. Model 1 is a first order nonlinear model with state variable ω and control input Q (motor torque):

$$ \dot{\omega} = \beta Q - \alpha\omega|\omega| \; ; \; \tau = C_{T}|\omega|\omega \qquad (5) $$

where α and β are two coefficients. Model 2 is a first order nonlinear model with state variables ω and v_p (axial fluid velocity at the propeller) and control input V_m (motor voltage):

$$ \dot{\omega} = k_1\omega + k_2 V_m - k_a Q \qquad (6) $$

$$ \dot{v}_p = \gamma T - \gamma\delta v_p|v_p| \qquad (7) $$

where k_1, k_2, k_a, γ and δ are constant coefficients. The rotational velocity ω can be used to compute the tangential velocity of the blade. Using the axial fluid velocity (v_p) and the tangential velocity (v_t) the relative velocity vector between the fluid and the blade wing (v_{total}) can be computed as well as the angle of attack (α):

$$ \begin{aligned} & v_t = 0.7r\omega \; ; \; \theta = \operatorname{atan2}(v_p, v_t) \; ; \; \alpha = p - \theta \\ & v_{total} = \sqrt{v_p^2 + \left(0,7r\omega\right)^2} \end{aligned} \qquad (8) $$

where p is the propeller pitch. Then, lift theory is used to estimate lift and drag:

$$ \begin{aligned} L &= \frac{1}{2}\rho a v_{total}^2 C_{L\max}\sin\left(2\alpha\right) \\ D &= \frac{1}{2}\rho a v_{total}^2 C_{D\max}\left(1-\cos\left(2\alpha\right)\right) \end{aligned} \qquad (9) $$

here, ρ, a, C_{Lmax} and C_{Dmax} are the fluid density, the thrusters duct area, the lift and the drag coefficients. Then, from the lift and drag the thrust and torque are computed:

$$ \begin{aligned} T &= L\cos\left(\theta\right) - D\sin\left(\theta\right) \\ Q &= 0.7rL\sin\left(\theta\right) - D\cos\left(\theta\right) \end{aligned} \qquad (10) $$

If a high gain servo velocity loop is used to compensate the motor electro-mechanic dynamics, then a simpler model with the state variable is v_p and control input ω is suggested. This model, called model 3, is formulated through equations 7 to 10.

IV. WORLD MODEL

In order to decide how to act, when the robot moves through the underwater world its sensors are used for sensing. To be able to simulate this behaviour, 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 underwater world encoding: (1) bathymetry and (2) 3D CAD-like models. A bathymetry

model is an elevation map consisting on a grid of altitudes [15]. 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 VII).

V. ENVIRONMENT MODEL

On the other hand, there are several physical variables that define the state of the environment: waves, wind, currents, temperature, and salinity within others. 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 [7] while wave models have been used in [15]. 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 [9] to simulate currents as a random walk process.

VI. SENSOR MODELS

A. Internal 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 corresponding 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 [15] 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 modelling 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.

B. External Sensors

External sensors are used for sensing the environment. Sonar and vision are examples of this type of sensors. Sonar 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 it is 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, returning, then, the corresponding value. In this case, the multipath can be simulated by using a source of non-zero mean high variance gaussian noise which is added with a particular probability depending on the environment type. This is the case of NEPTUNE which is described in section VII.

The use of realistic 3D graphical simulations allows nowadays making the 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 about the simulation of a cable tracking mission). Nevertheless, the simulation of computer vision has several drawbacks. Although OPENGL provides the functionality needed for fog simulation, it is not obvious how to simulate the turbulence commonly present in the underwater environment. The same happens with other phenomena like the forward and backward scattering. It should be noted that those processes must be simulated in real-time, which makes this problem even more challenging. Therefore, none of the studied systems (see table 2) simulate them. Instead, they simulate a theoretical vision sensor in a clear water. This is also the case of NEPTUNE.

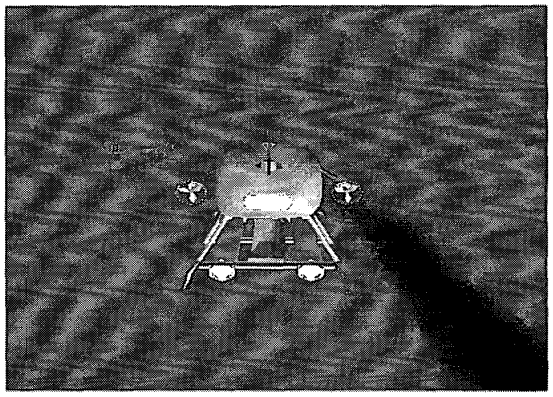

Figure 2. Virtual View of NEPTUNE

VII. 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 spheres of a particular radius of action. This model, together with a conic beam sonar model allows 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.

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]	B O	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
[7]	B	SMTIC	N	N	N	N	N	Hil	Y	3D	Y	EAVE; EST
[8]	N	C	Y	G_b	Y	H	A	HIL; OM	Y	3D	Y	ODIN; SAUVIM
[11]	O	N	Y	N	N	H	N	On; Hil	Y	3D	Y	PHANTOM
[12]	N	N	Y	G_b	N	H	N	On; Hil; Hs; P	Y	3D	Y	Twin Burger; PTEROA; MANTA
[15]	B	CW	Y	N	N	H	S_s	On; Hil	Y	3D	Y	OEX
[16]	B	MCWTD	Y	N	N	H	N	N	Y	3D	Y	Sea Squirt
NEPTUNE	B O		Y	G_c	Y	H	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; OFF: Offline Simulation; On: Online Simulation; Hil: Hardware in the loop Simulation; Hs: Hybrid Simulation; P: Mission Playback; OM: Online Monitoring; G_c: Conic beam geometric mode; G_b: Single beam geometric mode; N: Not detailed; Y: Yes;

NEPTUNE is a multi-vehicle Simulator (figure 2). 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. 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 and attitude within the robot frame ...). 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 and 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 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 (figure 3). In Table 2 the main properties of NEPTUNE as well as the main properties of several surveyed graphical simulators are reported.

VIII. A CASE OF STUDY

The URIS robot (figure 4) was developed at the University of Girona with the aim of building a small-sized AUV. The hull is composed of a stainless steel sphere with a diameter of 350mm, designed to withstand pressures of 4 atmospheres (30 meters depth). On the outside of the sphere there are two video cameras (forward and down looking) and thrusters (2 in X direction and 2 in Z direction). Due to the stability of the vehicle in pitch and roll, the robot has four degrees of freedom (DOF): surge, heave, sway and pitch. Except for the sway, the others DOFs can be directly controlled. The robot has two onboard PC-104 computers. One runs the low level and high level controllers on a QNX real time operating system. The other runs computer vision algorithms on a Linux operating system. Both computers are connected through an Ethernet network. An umbilical wire is used for communication, power and video signal transmissions.

For small-size, low-cost robots like URIS navigation is a really challenging problem to solve. Compared with the dimension of URIS, conventional sensors like DVLs or the transponders used for LBL and USBL are too big to be included. URIS uses the MT9 INS from XSens Technologies, which is a good low cost solution for attitude estimation. Nevertheless, as other low-cost INS it is not accurate enough for the localization through double integration of the acceleration, since the robot exhibits very small accelerations. Only extremely expensive and very big INS can be used for this purpose. For this reason, URIS navigation will be based on computer vision. While the Image mosaicking based navigation [10] is modified to be

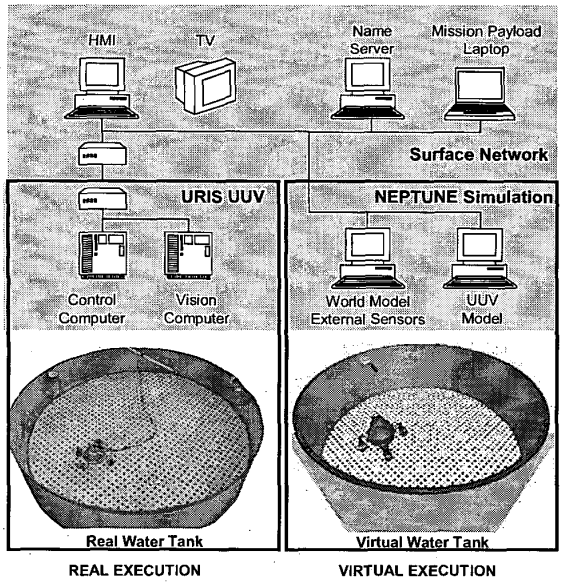

Figure 3 Networked architecture of URIS and NEPTUNE.

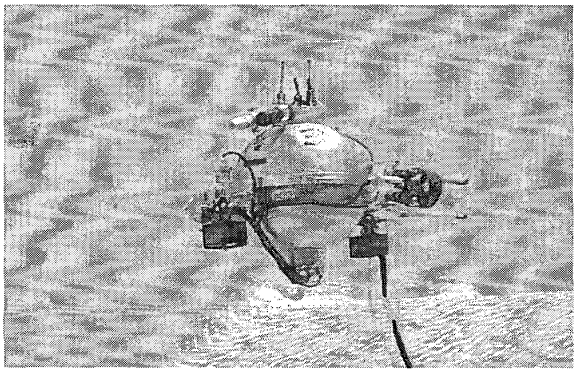

Figure 4 URIS UUV during a pool test.

m	30 [Kg]	$X_{\dot{u}} = Y_{\dot{v}} = Z_{\dot{w}}$	29.4462 [Kg]
${}^{B}x_{G}$	${}^{B}\vec{0}$ [m]	$K_{\dot{p}} = M_{\dot{q}}$	0.8744 [Kg.m^2]
		$N_{\dot{r}}$	1.5423 [Kg.m^2]
${}^{B}x_{b}$	^{B}(0 0 -0.0118) [m]	$X_u = Y_v = Z_w$	17.51 [N/(m/s)]
B	0	$K_p = M_q$	0.8090 [N/(m/s)]
$I_{xx} = I_{yy} = I_{zz}$	0.27 [Kg.m^2]	N_r	2.4 [N/(m/s)]
Note: The parameters not shown are equal to zero			

Table 3 Model parameters for URIS UUV.

Forward sense [N/rpm^2]		Backward sense [N/rpm^2]	
$C_{T1} = C_{T2}$	$1.43 \cdot 10^{-5}$	$C_{T1} = C_{T2}$	$1.48 \cdot 10^{-5}$
$C_{T3} = C_{T4}$	$1.29 \cdot 10^{-5}$	$C_{T3} = C_{T4}$	$1.25 \cdot 10^{-5}$

Table 4 Thruster coefficients for URIS UUV (affine model)

able to run in real time, a navigation system based on computer vision was especially developed to be used within our water tank. This localization system, briefly described in next section, was used during the experiments for the identification of the URIS hydrodynamic parameters which are shown in Table 3 and Table 4 (see [14] for details).

A. Lab Setup For URIS UUV

A small water tank located in the lab is used for experimenting with URIS. The tank is 4.5 m. diameter and 1 m. depth (figure 3). Although its small size, the dimensions are enough to run experiments in yaw, pitch and surge. Only heave experiments cannot be run, due to the lack of depth. An specially coded pattern is located in the bottom of the water tank. Using a down looking camera mounted in the robot, URIS is able to process the coded pattern images and estimate its position, attitude and velocity in real time [6]. Two more cameras are used in the setup. One provides a top view of the robot. The other, is an underwater camera located into the water tank.

B. HIL Simulation of URIS using NEPTUNE

In order to be able to run a HIL simulation of URIS several components are needed:

- URIS.UUV: this file (figure 5) contains the robot definition which is composed by the file name of the URIS model, the file name of the VRML file containing the graphical 3D model of the robot, the radius of the smaller sphere containing the robot, the initial position of the robot and a list of the sonar sensors included in the robot. For each sensor, the maximum range, the resolution and its position and attitude in the robot fixed frame are defined.

- URIS.DYN: this file (figure 6) defines the robot and thruster models. Here, the values for the matrixes of eq. 2 are defined for both senses: forward and backward. It defines de buoyancy, the buoyancy centre, the weight and the gravity centre. For each thruster the thrust and torque coefficients (eq.4) are defined together with the force and torque direction vectors.

- URIS.WRL: this is the VRML file generated using conventional software for 3D object modelling.

- URIS.MON: (figure 7) this file defines the components of the virtual world where the robot will move. It defines the VRML file which contains the 3D topography of the world. A list of objects which complement the world is also provided. Each of these objects has its own scale factor, radius of influence (for obstacle detection) and position-attitude vector.

Once all the files are available, a NEPTUNE project must be setup. The project defines the world definition file to be used

(URIS.MON), and the UUVs to be included in the simulation (URIS.UUV in this case, nevertheless, several UUVs can be included in this step). The final step consists on enabling the simulation. During the execution, it is possible to open as views as desired and to save the images of the views in a file in order to build videos of the simulation.

```
dynamic <uris.dyn>
model <uris.wrl>
scale <0.0030>
radius <12.6>
position <0.0 0.0 0.0 0.0 0.0 0.0>
sensor <downward><
 x_max <10.0>
 beta <0.3>
 inc_x <0.3>
 inc_z <0.2>
 position <0.5 0.25 0.5 0.0 -90.0 0.0>
>
sensor <forward><
 x_max <10.0>
 beta <0.3>
 inc_x <0.3>
 inc_z <0.2>
 position <0.0 0.0 1.25 0.0 -30.0 0.0>
>
```

Figure 5 URIS.UUV file.

```
Mf <59.4462 0.0 0.0 0.0 0.0 0.0
 0.0 59.4462 0.0 0.0 0.0 0.0
 0.0 0.0 59.4462 0.0 0.0 0.0
 0.0 0.0 0.0 1.1444 0.0 0.0
 0.0 0.0 0.0 0.0 1.1444 0.0
 0.0 0.0 0.0 0.0 0.0 1.8123>

Mb
<59.4462 0.0 0.0 0.0 0.0 0.0
 0.0 59.4462 0.0 0.0 0.0 0.0
 0.0 0.0 59.4462 0.0 0.0 0.0
 0.0 0.0 0.0 1.1444 0.0 0.0
 0.0 0.0 0.0 0.0 1.1444 0.0
 0.0 0.0 0.0 0.0 0.0 1.8123>

DLf
< 17.51 0.0 0.0 0.0 0.0 0.0
 0.0 17.51 0.0 0.0 0.0 0.0
 0.0 0.0 17.51 0.0 0.0 0.0
 0.0 0.0 0.0 0.8090 0.0 0.0
 0.0 0.0 0.0 0.0 0.8090 0.0
 0.0 0.0 0.0 0.0 0.0 2.4>

DLb
< 17.51 0.0 0.0 0.0 0.0 0.0
 0.0 17.51 0.0 0.0 0.0 0.0
 0.0 0.0 17.51 0.0 0.0 0.0
 0.0 0.0 0.0 0.8090 0.0 0.0
 0.0 0.0 0.0 0.0 0.8090 0.0
 0.0 0.0 0.0 0.0 0.0 2.4>

DQf
< 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0>

DQb
< 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0
 0.0 0.0 0.0 0.0 0.0 0.0>

b <300.0>
bc <0.0 0.0 -0.0118>
w <300.0>
gc <0.0 0.0 0>

motor <motor0><
CTf <0.00001434760> CTb <0.000014780836>
CQf <0.00000430428> CQb <0.0000044342508>
O <1.0 0.0 0.0>
PA <0.0 0.0 1.0>
>

motor <motor1><
CTf <0.00001434760> CTb <0.000014780836>
CQf <-0.00000430428> CQb <-0.0000044342508>
O <1.0 0.0 0.0>
PA <0.0 0.0 1.0>
>

motor <motor2><
CTf <0.00001294597> CTb <0.0000125>
CQf <-0.000003883791> CQb <-0.0000038835>
O <0.0 0.0 1.0>
PA <0.0 1.0 0.0>
>

motor <motor3><
CTf <0.00001294597> CTb <0.0000125>
CQf <0.000003883791> CQb <0.0000038835>
O <0.0 0.0 1.0>
PA <0.0 1.0 0.0>
>
```

Figure 6 URIS.DYN file

```
world<
    <water_tank.wrl>
    scale <2.0>
    resolution <0.01>
>
object<
  <underwater_camera.wrl>
  scale <0.05>
  radius <0.1>
  position <0.0 -2.0 -4.0 0.0 0.0 0.0>
>
```

Figure 7 URIS.MON file

IX. CONCLUSIONS

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 modelled through their hydrodynamic equation and thrusters are modelled using the affine model. The world is modelled using VRML as well as a bathymetry model and sonar is modelled using a geometric method.

Acknowledgments

Authors want to thanks Dani Ferrés, Narcís Palomeras and Emili Hernàndez for their help with the implementation of different versions of NEPTUNE.
This research was sponsored by the Spanish commission MCYT (DPI2001-2311-C03-01).

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 Manoeuvring and Control of Marine Crafts, Alborg (Denmark), pp. 141-146, August 2000.

[6] M. Carreras. et al.. "Vision-based Localization of an Underwater Robot in a Structured Environment", IEEE International Conference on Robotics and Automation ICRA'03, Taipei (Taiwan), September 2003.

[7] 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.

[8] 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.

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

[10] Garcia, R. et al., "Towards a Real-Time Vision-Based Navigation System for a Small-Class UUV," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas (USA), October 2003.

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

[12] 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.

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

[14] P. Ridao, A. Tiano, A. El-Fakdi, M. Carreras and A. Zirilli, "On the identification of non-linear models of unmanned underwater vehicles", Control Engineering Practice, Vol. 12, Issue 12 , Pages 1483-1499, December 2004.

[15] 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.

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

[17] 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.

[18] ROV Pilot Trainer Web page, http://www.underwater.pg.gda.pl/01_rov.htm, accessed on 31/08/2004.

[19] ROVolution Web page, http://rovolution.co.uk/grlhomepage.htm, accessed on 31/08/2004.