

Communications with Thetis, a Real Time Multi-vehicles Hybrid Simulator

Olivier Parodi, Vincent Creuze, Bruno Jouvencel

▶ **To cite this version:**

Olivier Parodi, Vincent Creuze, Bruno Jouvencel. Communications with Thetis, a Real Time Multi-vehicles Hybrid Simulator. ISOPE: International Society of Offshore and Polar Engineers, Jul 2008, Vancouver, Canada. pp.326-331. lirmm-00311712

HAL Id: lirmm-00311712

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00311712

Submitted on 20 Aug 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Communications within Thetis, a Real Time Multi-vehicles Hybrid Simulator

Olivier Parodi, Vincent Creuze, Bruno Jouvencel
Robotic Department, LIRMM – University of Montpellier 2 - CNRS
Montpellier, France

ABSTRACT

The purpose of this paper is to present the communications aspect in *Thetis*, a real time multi-vehicles hybrid simulator for heterogeneous vehicles.
This simulator allows *hardware in loop* (HIL) simulations including virtual sensors which allows to provide a representation of a virtual world, and with the support of communication devices, which allow overall communications between vehicles.
After a short state of the art, we introduce the main mechanisms of our simulator. Then we present the modeling of the phenomena which are encountered when AUVs (Autonomous Underwater Vehicle) communicate with aquatic modems, and the messages distribution system considering the assumptions we have made. Finally, we present results of our Hardware In Loop Simulation in which *Taipan 2* and *Taipan 300* (our 2 AUVs) exchange messages in different spatial configurations.

KEY WORDS:
Simulation; Modeling; Multi-vehicles simulator; Real-time systems.

INTRODUCTION

Development and control of an autonomous vehicle is far from being an easy task; this is especially true for underwater robots because of the difficulty to supervise it during the experiments. In order to avoid long time and expensive design and implementation during which there is a possibility to lose or damage the material, it is necessary to test all the sub-systems of the robots before launching the robot in the harsh underwater environment. Moreover scenarii, in which many autonomous vehicles are simultaneously used, are nowadays seriously considered. Indeed this kind of mission allows to deploy heterogeneous vehicles with different sensors in order to grasp the environment more efficiently and faster than with a single vehicle. Currently there are few experimentations in which many AUVs are simultaneously deployed. Possible reasons are the cost of such experimentations, the unavoidable logistic burden, and the theoretical aspect of the most control laws, which rarely consider the computation time or the underwater communications restrictions.
Hence, simulation tools play an important role: they help us to test and validate control laws and software architecture, and to detect preliminary inconsistencies within the scenarii. Moreover, these technologies limit the required human resources, decrease the number and difficulty of necessary real experiments, the cost and the time spent. There are different types of simulators. A useful classification is proposed in (Ridao, Batlle, Ribas, Carreras, 2004). Simulators are classified in 4 categories: the offline simulators, the online simulators, the hardware in loop simulators and the hybrid simulators.
Offline Simulators allow to design the control of robots in a fisrt approximation. Matlab/Simulink is often used for this kind of simulation because of the availability of toolboxes (such as the one proposed by Fossen described in (Fossen, 2002) and because of the easy implementation of mathematical models. But we have to keep in mind that the temporal aspect of the simulation is not taken into account and it potentially makes the control algorithm inoperative when it is transferred on a real robot. Thus it is not possible to validate control architecture or a sensor referenced command with such a simulator.
Online Simulators belong to another type which allows to take into account the temporal consistency of the simulation. Indeed, in this type of simulation, 1 second of simulated time actually corresponds to 1 second in real time. This is the case in the *SubSim* simulator (Tobias Bielohlawek, 2006). However the algorithms are still not executed on the robot itself, and the temporal behavior of the computer used for the simulation can be different from the one onboard the robot.
In *Hardware In Loop* simulations, the control algorithm is executed on the robot itself, but the commands sent to the actuators are routed towards the simulator instead of the real robot (Suriano, Moriconi, 2007). The simulator then considers the actuation commands in order to compute the evolution. However, in this sort of simulation the external world is not taken into account, except for the dynamic properties of the environment. Only the proprioceptive sensors are simulated and the overall algorithms suite cannot be fully tested.
Hybrid simulators are HIL simulators where real and virtual systems interact together in an virtual environment. It is therefore necessary to simulate an environment (static or dynamic) in which the robot will be fully functionally operative. Therefore, it is possible to test all the algorithms of the machine, from low level control of sensors or actuators to the whole architecture. This approach has been used by several authors such as in (Ridao, Batlle, Ribas, Carreras, 2004), in which, authors present Neptune, their real time graphic multi-vehicles simulator, allowing to performe online, HIL and hybrid simulation.
In order to elaborate a command and a strategy to make our 2 AUVs cooperate together, we are particularly interested in HIL or hybrid

simulators which are able to simulate several heterogeneous vehicles. Several studies have been made on this topic, in the scope of underwater robotics.

First works have been done by the DARPA Naval Technology Office in 1988 (Albus, 1988). This simulator allows the cooperation between many underwater platforms which are driven by a real time intelligent sense-decide-act system. The simulation of sensors and environment are taken into account.

In (Lane et al., 1998), a distributed simulator for underwater vehicles called *core simulation engine (CSE)* has been developed. This system is equipped with operator training capabilities, mission feasibility assessment and mission replay. A subscription method and a run time infrastructure allowed the distributed programming.

The Cooperative AUV Development Concept (CADCON) simulator (Chappell, Komerska, 2001) uses a client-server model in which it is possible to handle interactions between vehicles controlled by the simulation clients through an environment simulator. This simulator system focuses on the high level communication and does not deal with the dynamics and control of the heterogeneous vehicles.

In (Ridao et al., 2001), a simulator called DEVRE, composed of a set of processes running on a local area network, has been developed. The simulation runs onto 3 computers which are the onboard AUV computer, a human machine interface computer and a third computer used to compute the dynamics and to represent a virtual world. In this simulator multi-vehicles simulation and communications between the AUVs are not allowed.

In (Hornfeld, 2001) they have also developed a HIL simulator for their AUV DeepC. This simulator provides real-time models (vehicle and environment), tools for planning and control in a virtual environment. Although it seems to be a very challenging project, it is difficult to find publications about the intern characteristics of this simulator.

In (Kobayashi et al., 2001) a simulator has been developed which is deployed on a 4 computers network. The first one is the control unit which is in charge of computing the control variables using the data from the sensors simulator. This computer is the same as the one onboard AUV. The second one allows to monitor and gives remote command for teleoperation. The third one is the AUV simulator which is in charge of computing the vehicle motion from the actuator command and external conditions. Finally the fourth one generates and submits the necessary data about execution of AUV simulation.

(Carlson, Beaujean, An, 2004) treats the vehicles as nodes of the communication system in the network's routing protocol. This simulator is designed specifically to support the development and testing of mobile ad hoc network routing protocols. It supports multiple heterogeneous vehicles simulation, considering multiple communication links.

The Subsim simulator (Tobias Bielohlawek, 2006) focuses on a system with dynamics which executes a control algorithm, simulates and visualizes sensor data, actuator behavior and camera pictures.

All the above mentioned simulators present several very helpful specificities, through several models (environment, vehicles…) which are able to accurately cope with reality. Some of them are specialized in HIL simulation, although others are mostly designed for multi-AUVs applications, while others are focusing on accurate environment simulation. Among all of these references no one has addressed the problem of temporal decoupling between the command computed by the onboard computer and the simulator(s). Yet this is a critical problem on addressed in next section. On the other hand, from the end-user point of view, the source codes and the associated documentations are not always available, and in this context it is not convenient to reuse the existing development. Most of these simulators are not based on a distributed architecture, meaning that it is not possible to add another computer when the load of the existing ones becomes too important. This leads some teams to simplify the used models at the expense of the quality of the obtained results.

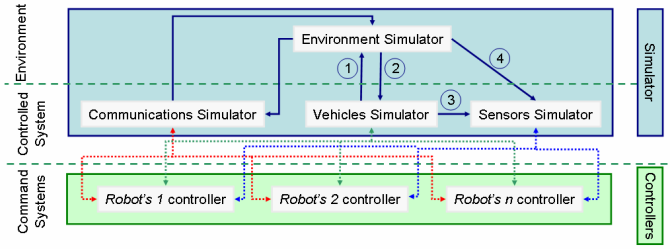


Fig. 1. Simplified architecture of *Thetis*: there is a logical sequence between 3 simulators even if there are independent processes on different computers. The cycle duration of this sequence must be largely lower than the period of the controller. The temporal decoupling is only effective (and that's enough) between the simulator and the controllers. The link between the communications simulator and the environment simulator is event-driven (when a communication between vehicles occurs).

Moreover the considered systems are always AUVs (sometimes heterogeneous) but there is no mention about cooperation capabilities between a surface vehicle and an underwater one (for example). Yet, we find more and more scenarii in which an UAV (Unmanned Aerial Vehicle), an ASV (Autonomous Surface Craft) or a land vehicle are required to cooperate.

For all these reasons we have developed *Thetis*, a hybrid simulator which provides the necessary environment to experiment HIL simulation with support of communications between the vehicles, and allows the use of exteroceptive sensors.

Building such a simulator which concurrently provides all these functionalities is a necessary tool to envisage such complex scenarii.

This is a challenging engineering multi-disciplinary task because of the number of specialists who must collaborate (acoustician, computer specialist, control engineer etc…).

For this reason the aim of this simulator architecture is not to provide a set of "perfect" models: in this paper we describe a simulator architecture allowing us to deal with the problem of heterogeneous coordinated vehicles, under communication constraints. Moreover, the open source *Thetis* project is built to allow for many different specialists to develop their own models.

On the other hand this simulator architecture is distributed, in order to allow the implementation of complex models, without being restricted by computational burden, thus guaranteeing the real-time aspect.

Finally this simulator allows to interchange the used models, in function of the desired accuracy.

As this paper focuses on the architectural aspects of the simulator, the different models, already implemented, are deliberately basic, except for the vehicles, where full hydrodynamic models are considered, as exposed in (Fossen, 2002).

This paper describes the mechanisms and structure of the simulator, focusing on inter-vehicles communication. For details on previous development, please refer to (Parodi, El Jalaoui, Andreu, 2008) where we focus on the connectability aspects of *Thetis* to our real robots.

So first we present the architecture of our simulator: *Thetis*, specifying the models we consider. Then we present the messages distribution system taking into account the assumptions we have made. Finally, we present results of a simulation where Taipan 2 and Taipan 300 exchange messages in different spatial configurations.

THETIS: HYBRID SIMULATOR

Thetis is a new real-time hybrid simulator allowing for considering heterogeneous multi-vehicles scenarii, including communication physical restrictions.

Critical Concepts

This simulator is composed with a set of mechanisms which respects several properties.

One of the most important is the capability of the simulator to ensure a *temporal decoupling* between the control loop and the simulation loop. This warrants the coherence between simulation results and expected real behavior. For example, if for any reason, the onboard computer is unable to provide the actuator commands at an appropriate time (unexpected delay), then the simulator has to exhibit the natural consequence of this delay on the vehicle behavior (open loop).

Indeed, in *Thetis*, there is no logical synchronization between the simulation loop and the controller computation onboard the robots.

Another important concept is the *upgradeability* of the simulator. We are able to add some new components (sensors, vehicles...) in a very easy way. This warrants us the possibility of using this simulator with different types of vehicles. Indeed, the *modularity* of this simulator favors the interventions of different specialists. Thus, a sonar specialist will be able to modify the sonar model, without considering the global simulator functioning. Then, the system designer can also transfer this model computation onto another computer if he considers it is too resource-consuming for the original one. Finally, a very interesting aspect of this architecture is its *portability*. Indeed it is able to work with different robots and thus be connected with different control software architectures. We only need to adapt the interface between the 2 systems (simulator and controller), in order to make them work together. Moreover, the *distributed* aspect of *Thetis* affords the faculty of the simulator to be divided and executed on different computers, linked on a dedicated local area network via UDP/IP protocol. This avoids to overload computers and to affect the real time capability of the systems.

Implementation and Technical Considerations

All the concepts exposed in the previous section are supported by using 3 mechanisms and the architecture is based on 4 simulators.

- First an *XML-based specifications exchange* (XML for Extensible Markup Language) allows to structure the parameters of the different models (modem, radio, fins, motor …), and configuration files, while promoting the modularity and the portability. Indeed modularity is obtained using the XML format, which allows to specify the components parameters, and modifying them in a very easy way. For example, if we consider 2 sensor devices, measuring the same physical quantity, the intrinsic parameters (time-response, accuracy, rate…) are described in 2 different XML files, while the physics of the phenomenon remains the same, and is described in a unique other XML file. Now, the replacement of a sensor by another is done by calling a XML file in place of the other. All the system components description follows the same idea (fins, acoustical modem, GPS, propeller,...). Moreover, many components could use the same formalism to describe their intrinsic parameters without using all of them. For example, we can imagine that an underwater communications specialist needs a very accurate model describing its modem; he should use all the parameters of the XML file. Another user, who doesn't need such an accuracy level, will use the same XML file, but only a part of these parameters in order to supply its own model. The validation and extensible properties of the XML language make it an ideal base to enrich the model parameters files. Thus the robots (in fact the different components as sensors, actuators, communication devices, etc...) used in this simulator are described in XML formalism. To do this, we suggest a set of different tags allowing to describe the components of the robots in an easy way.

- Then portability and temporal decoupling is favored by using *sockets* and *local shared memories* widely. Thus it is possible to run several processes on different computers each of them interacting with others using these mechanisms.

- In order to ensure real-time performances as well as effective temporal decoupling, the overall simulation system is in fact an application divided in 4 simulators. The first one is a *vehicle simulator* which allows the simulation of the robots dynamics. The second one is a *sensors simulator* allowing the simulation of the various sensors of the robots. The third one is a *communications simulator* in charge of computing the behavior of communications device (signal processing for an acoustic modem, for example) and of distributing the different messages to the concerned vehicles (in fact those which are within range of communications). Lastly the fourth one is a static *environment simulator* (no evolution with time at yet). It allows the simulation of exteroceptive sensors, it is in charge of computing the possible collisions and of computing the different acoustic signals propagation, the latency and distortions of the communication signals. All these simulators are interconnected on a dedicated UDP/IP network. This avoids overloading the network in order to guarantee real-time performances and to avoid packets losses (potentially possible with UDP/IP network). The connections between these blocks are detailed and explained on figure 1.
  Only the sensors, the communications and vehicles simulators are connected to the real robot, the environment simulator being connected only to the 3 other ones. All these simulators work under Linux RTAI. Information about network configuration is described in a shared XML file. All the XML files describing the components of the system are loaded at the initialization and thus allow to instantiate the different objects of the simulator.

- Finally we have created a set of libraries containing a set of classes enabling us to build the various objects of the simulation system. All these classes are documented with doxygen tool (doxy, 2007) and are available online (www.lirmm.fr/~parodi/thetis).

Vehicles Simulator

The vehicle simulator is in charge of computing the robots dynamic evolution. We present this simulator structure on figure 2. It is composed of 2 independent processes communicating via a local shared memory. Before starting the simulation, this simulator connects itself to the environment simulator to obtain all the physical constants (environment viscosity and density, gravity…).

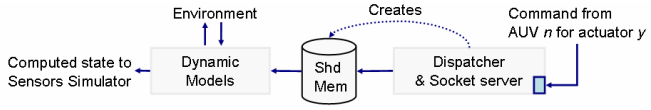


Fig. 2. Architecture of the vehicles simulator.

The first process is the *Socket Server and Dispatcher* process, which is in charge of ensuring the communication between the simulator and the robots controllers. The robots send actuators commands calculated by the onboard computer to the simulator through UDP socket. Then the

data received are decoded and written to a shared memory initially created by this process.
The second process is the actual cycle of vehicles simulation (called *Dynamic Models* on the figure 2). During this cycle all the forces and torques applied to the robots are computed in order to determine the vehicles accelerations and, after integration steps, vehicles attitudes and velocities are computed. The computed data are sent to the environment simulator in order to verify the absence of collisions and if occurs to correct positions. Afterwards, the computed data are sent to the sensors simulator. This cycle is presented on figure 3. The initial position and attitude of the vehicles are given at the initialization (this data is provided by the XML configuration files), then the dynamic model evolves according to the actuation effects and the environment.

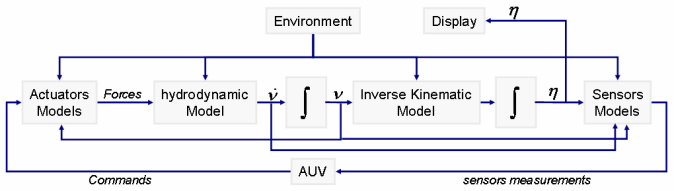


Fig. 3. Simulation cycle of the vehicles simulator (classically, ν expresses the vehicle velocity in the vehicle frame; η denotes the position and attitude of the vehicle, in the earth-fixed frame).

Sensors Simulator
The sensors simulator is in charge of providing to the real robot (onboard computer) virtual data from the simulation. Presently the models of proprioceptive and simple exteroceptive (Temperature, Conductivity…) sensors are implemented. Complex exteroceptive sensors (sonar and camera) will be the next steps of our work.
This simulator is based on the execution of 3 independent processes, communicating via 2 local shared memories. The structure of this simulator is presented on figure 4.

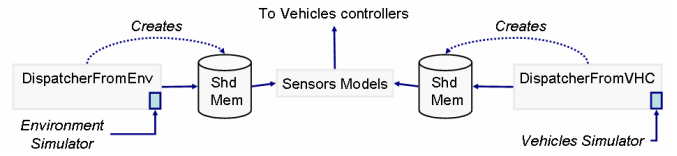


Fig. 4. Architecture of the sensors .

As in the vehicles simulator, there are 2 dispatcher processes (*DispatcherFromVHC* and *DispatcherFromENV*) which are respectively in charge of listening to messages from vehicles simulator (vehicles state: velocities, accelerations, attitudes...), and messages from environment simulator (parts of maps determined according to the position and the range of the robot's sensors), and of decoding and writing data into 2 local shared memories (one for each process). The third process is the sensors simulator which computes output of each sensor for each robot. These outputs are computed according to each sensor model (which parameters are described in the XML file), data from the vehicles simulator (systems state), and at last to the data from the environment simulation. Once these outputs are computed, they are specifically sent to the concerned robots.

Environment Simulator
The environment simulator is in charge of providing the stored geophysical maps around a given geographical point (Temperature, Salinity, Local current, Plankton density…), computing signal propagation (when communications between vehicles occurs), signal distortion and latency, detecting collisions. At each cycle of vehicle simulation, the environment simulator is called and sends back the value of the local current and the occurrence of a collision.
Even if many things seems to be computed by the environment simulator, it is worth noting that only the calculation of the signal propagation may be resource-consuming depending of the implemented model. The maps management is light because we only consider static maps (no evolution with time). It is the same for the collision detection algorithm: the complexity of calculations depends mainly of the number of vehicles.
As the others this simulator is based on several independent processes, each of them sharing its own data with shared memories as we can see on figure 5. As before we find again 3 dispatcher processes (*DispatcherFromCOM*, *DispatcherFromVHC* and *DipacherFromSEN*) which are in charge of listening to messages from the different simulators. The main program consists in a loop which computes sequentially the different tasks previously mentioned. The computed data are sent to the other simulators during this cycle. It has to be noted that the computed communication signals are only sent to the communications simulator when (considering latency and bandwidth) and if necessary (if the vehicles are not within range of communications no messages will be delivered). Moreover the rate of communication is considered to be constant in the communication area. When a communication occurs in this area, other communications using the same frequency potentially invalidate the communication channel, and the messages are not delivered.

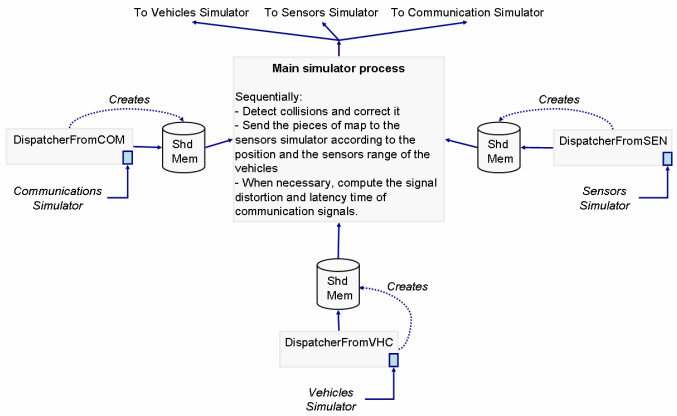


Fig. 5. Architecture of the environment simulator

Communications Simulator
We are currently implementing a messages distribution mechanism for inter-robot communications. This mechanism takes into account the delay, the rate and the losses caused by the type of communication device in use, and the propagation medium. It allows to teste some control laws for coordinated AUV flotilla.
The communications simulator is in charge of simulating the signal processing done by the communication devices. When a vehicle has to emit a message (radio or acoustic waves), it sends a request to its concerned communication device (radio, wifi, acsoutic modem…). In the simulation case, these data are not sent to the physical communication device, but routed to the communications simulator (Parodi, El Jalaloui, Andreu, 2008) for more information about the connectivity of the simulator with our AUVs). The communications simulator receives a frame which contains the emitting AUV identifier, the message to be sent in clear (no RAW data) and the identifier of the communication device to be used for this issue.
The model of the involved communication device is used to generate the emitted signal. Then, the computed analog signal is sent to the environment simulator. Presently the model of the communication device is not implemented yet and we send the clear message. The

required parameters (frequency, emission intensity, communication rate…) necessary for the environment simulator in order to be able to provide its propagation model, are initially loaded from the XML configuration file, related to the involved communication device.
Indeed, for our application, we don't need to accurately model the signal propagation in the environment. On the other hand, the simulator architecture will allow to interested people to implement their own models.
As mentioned earlier, the environment simulator computes the shape of the propagated signal and its characteristics. Then they are delivered to the communications simulator, after a time representing the time of the signal propagation in the environment. This calculation is made between the emitter and all the other vehicles. This allows to determine the vehicles which could be reached (i.e. those which the signal reception is higher than the trigger point of their communication device). The communications simulator could transmit the messages to the concerned vehicles after having decoded the received signal thanks to the inverse model of the communication device.
In order to realize this, the communications simulator has a structure quite similar to others, presented fig. 6.

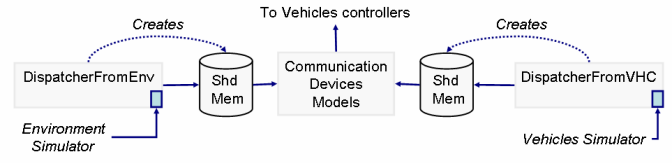


Fig. 6. Architecture of the communications simulator.

MODELS AND ASSUMPTIONS

In this chapter we explain the modeling which we have chosen to develop the simulator. Although it is not exclusively limited to simulate AUVs, it is the first model that we have implemented because the interest of our team (Spiewak, Jouvencel, Fraisse, 2006). Other models will be implemented later if needed. Hence, the simulated sensors suite is dedicated to underwater applications and the modeled environment is exclusively underwater. Obviously, all this can be easily modified in order to deal with heterogeneous robots and environment like coordination between AUVs, UAVs and surface crafts.

AUV Modeling

The robot model is used to compute the robot accelerations according to the actuation command vector. The modeling of the AUV is made up of the hydrodynamic, hydrostatic and dynamic phenomena of the robot on the one hand, and of the actuators model on the other hand. Here is a brief description of these models:

- *Hydrodynamic forces*: the simulator uses the 6 dof (*Degree of Freedom*) non linear equations of 6DOF AUV expressed in the body fixed frame (Fossen, 2002)

$$\dot{\nu} = (M_A + M_{RB})^{-1}(\tau - (C_{RB} + C_A + D)\nu - g(\eta)) \quad (1)$$

where ν denotes the system velocities (linear and angular), M_{RB} and M_A are the inertia matrix and the added mass matrix, τ denotes the force and torque produced by the thrusters and control surfaces, C_{RB} and C_A the rigid-body, the added Coriolis and centripetal matrixes, D denotes the damping matrix, g denotes the gravity and buoyancy force and torque, η denotes the position and orientation vector
The potential damping, the skin friction and the wave drift damping are not considered.
Only the damping due to vortex shedding is computed. Once the estimation of the accelerations is computed, we integrate a first time to obtain the velocities in the body fixed frame. It is then necessary to integrate a second time after shifting the frame so as to obtain the robot position and attitude in the inertial frame.

$$\begin{bmatrix} \dot{p}^n \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} R_b^n(\Theta) & 0_{3\times3} \\ 0_{3\times3} & T_\Theta(\Theta) \end{bmatrix} \begin{bmatrix} v_o^b \\ \omega_{nb}^b \end{bmatrix} \quad (2)$$

- *Thruster Model*: A steady state model is used for the thruster of the AUVs. We can consider that the response is instantaneous according to the input.
We use a bilinear model, i.e. a non linear function computing the thrust according to the angular speed of the propeller ω and the linear speed of the robot in the thruster direction. See (Ridao et al., 2004) for more details.

$$T = C_{T\|\omega\|\omega} \times \omega - C_{T\|\omega\|\nu} \times \nu \quad (3)$$

Other authors propose dynamic models (Whitcomb, Yoerger, 1999), (Fossen, 2002). The possibility of implementing such thruster models will be studied later.

- *fins Model*: the robots Taipan (Spiewak et al., 2006) have a cylindrical shape and are equipped with rudder at the stern and with 2 pairs of diving planes located at the bow and at the stern.
The lift F_Z is the projection of the resultant F on the axis, orthogonal to the fluid direction. The drag force is the projection on the axis, parallel to the fluid direction. These forces are modeled according to the following equations, expressed in a frame fixed on the control surface (Aucher, 1981) and (Santos, 1995):

$$F_{cz} = \frac{1}{2}\rho S V_0 C_{zs}$$
$$F_{xz} = \frac{1}{2}\rho S V_0 C_{xs} \quad (4)$$

Where ρ is the density of the fluid, S is the projected area of the fin, perpendicularly to the fluid direction; V_0 is the relative velocity of the body with the fluid, C_{ZS} is the lift coefficient corresponding to the axes of the surface; C_{XS} is the drag coefficient corresponding to the axes of the surface.
Once the physical action of the planes is expressed, we consider that the axis of rotation of the planes is situated at a distance d_a from the origin of the local frame of the robot.
The forces of the lift and drag (no evolution with time at yet) and the induced moment are therefore given by:

$$\begin{bmatrix} F_x \\ F_z \\ M_q \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\rho S_s V_0^2 (C_{cz}\sin\delta + C_{xs}\cos\delta) \\ -\frac{1}{2}\rho S_s V_0^2 (C_{zs}\cos\delta - C_{xs}\sin\delta) \\ F_z(lc_s\cos\delta - d_{as}) + F_x(lb_s\sin\delta) \end{bmatrix} \quad (5)$$

where: l is the distance between the leading edge of the planes and the system metacentre; b_S is the wingspan; c_S is the chord; S_S is the surface wing defined by $S_s = b_s c_s$; $l=0,2$ for the taipan class.

Sensors Modeling

The proprioceptive sensors provide measurements in order to estimate

the state variables of the robot, and its derivatives. These sensors are modeled by using directly the variables produced by the vehicles simulator. Afterwards, the sensors simulator provides the samples at the same frequency as the real sensor, taking care to limit its range and adjust its resolution. It is also necessary to add noise so as to obtain a more realistic simulation.
Presently a GPS (Trimble Lassen SKII), a velocity logs (RDI Workhorse Navigator Doppler Velocity Log), as well as an attitude and heading reference system (XSens MTi) are modeled.
As the physical phenomena driving the exteroceptive measurements are complex, the modeling process of the exteroceptive sensors is a non trivial task.
We are presently working on a simple sonar model using classic ray tracing method.

Environment Modeling
The environment is modeled using different elements: the topography, the temperature and salinity distribution, the environmental disturbances (currents). These data are included in a single function $[temperature, salinity, current] = f(x, y, z)$. Presently, only the topography, the temperature, salinity and currents distribution are implemented. This model considers these phenomena as stationary.

Communications Modeling
The messages distribution model previously mentioned is the same whatever the communication devices used or the environment propagation, despite the related parameters, contained in a XML file, initially loaded.
Nevertheless, we have started with a model developed for the propagation of sound waves in the aquatic environment, since the main subject of our team is underwater robotics.
The aquatic acoustic communications are constrained by intrinsic properties of water (important propagation time, low communication rate due to the use of low frequency, reflections, and the necessity to incorporate error-correcting code). Thus we have created a propagation model which takes into account several physic phenomena which represent the constraints that we actually meet in our experimentations. We mainly model the transmission losses (absorption and dispersion) and the sea relative noise level.

Transmission losses
Whatever is the environment in which the acoustic wave is propagating, it undergoes an attenuation which mainly depends on the frequency and the distance traveled by the wave. In fact this attenuation is due to 3 main phenomena:

- The absorption, which traduces the fact that a part of the energy is converted into heat along the traveled path and that we model like (L. Berkhovskikh, Y. Laysanov, 1982)

$$g(f) = \frac{0.11 f^2}{1+f^2} + \frac{44 f^2}{4100+f^2} + 2.75\times10^{-4} f^2 + 0.003 \quad (6)$$

  where f is the signal frequency in KHz and g the attenuation in dB/Km.
- The dispersion which represents the acoustic wave spread during its propagation (the surface of a sphere is growing up with the square of the distance from the emission point, independently of the wave frequency) :

$$TL_D = 10\log d^2 \quad (7)$$

  where d denote the distance in meters.
- The diffusion (phenomena whereby a beam is deflected in many directions) and which is not considered here.

Then, total loss is expressed as: $TL = g \times \frac{d}{1000} + TL_D$.

Noise Level
The oceanic noise is mainly due to human activities noise, surface agitation created by the wind, the rain, the animals and finally the thermal noise. We resume on fig. 7. the range of the frequency occupied by the different sources of the Wenz model and their typical acoustic intensity.

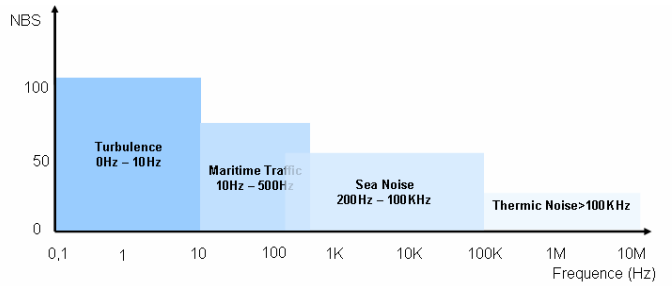


Fig. 7. Isotropic noise level from different sources.

The noise level generally decreases when frequency increases. Wenz Wenz, 1962) has proposed a model (eq. 8.) which allows us to approximate the average level of each noise component.

$$\begin{cases} f < 10Hz : NBS_{Turbulence} = 107 - 30\log f \\ 10Hz < f < 500Hz : NBS_{Trafic} = 76 - 20\left(\log\left(\frac{f}{30}\right)\right)^2 + 5(I_S - 4) \\ 200Hz < f < 1KHz : NBS_{LowSeaNoise} = 44 + \sqrt{21 v_{kt}} + 17(3 - \log f)(\log f - 2) \\ 1KHz < f < 100KHz : NBS_{HighSeaNoise} = 95 + \sqrt{21 v_{kt}} - 17\log f \\ f > 100KHz : NBS_{ThermicNoise} = -75 + 20\log f \end{cases} \quad (8)$$

where f is the central frequency of the considered band; I_S is a traffic cue (1: low 7: strong); v_{KT} the wind speed in knots.
It should be noted that the noise level NBS given here is a spectral isotropic noise level expressed in $dB/\mu P\sqrt{Hz}$. Indeed, the noise intensity is expressed as an energy density, i.e. for 1Hz-wide band frequency.
Hence the noise level (in dB) must be reported on the bandwidth (W expressed in Hz) of the used receiver system:

$$NB = NBS + 10\log W \quad (9)$$

Propagation
Using the following inequality (eq. 10.), it is possible to determine the radius of the sphere that is bounded by the reception threshold of the receiver AUV.

$$SL - TL - NB > threshold \quad (10)$$

where SL stands for Source Level of the transmitter AUV.
Thus it is possible to determine for each AUV (other than the emitter AUV) if it is in the signal propagation sphere. If not, the message will not be delivered. Else, we determine more precisely the reception level taking into account the attitudes of the 2 vehicles and the directivity diagram of the antenna (using a lookup table).
If the computed signal reception level is higher than the reception threshold of the modem, the message is delivered after a duration Δt which is computed taking account of the distance between the 2 antennas, depending of the size of the message and finally the modems communication rate (eq. 11):

$$\Delta t = \frac{d}{c} + \frac{Size}{D} \qquad (11)$$

where *Size* is the size of the message to be transmitted and *D* is the communication rate.
The low dynamic of the systems and the very small messages to be exchanged, allows us to consider that the message could be delivery at once; it does not seem to be a too strong hypothesis.
On the other hand, we determine the areas in which it is no longer possible to receive a message. Indeed if several AUVs are emitting at the same time on the same frequency, we consider that the message cannot be delivered to AUVs staying in the overlapping area, defined as the intersection of the different propagation spheres of the emitters.
The communication rate in these spheres is considered to be constant because we exploit the safe mode of our modems that allows to improve the quality of communication, despite the reduced bandwidth (20bit/s). It does not seem beneficial for us to implement a more sophisticated propagation model given the experimental conditions. Indeed our AUVs are spaced with a few hundred meters. Thus we don't need to implement a modal or parabolic algorithm for this kind of application. We are currently implementing a model based on the ray theory to increase our simulation accuracy.

PRELIMINARY SIMULATIONS

We have 2 AUVs which are currently being upgraded. The first one is Taipan 300; this is a small AUV which is designed for shallow water operations. It is 193cm long for a diameter of 15cm and a weight of 32 Kg (figure 8).



Fig. 8. Taipan 300 in front of the Salagou Lake in France

On the photo (see fig. 8.) the vehicle is fitted with a CTD (located at the front of the vehicle just behind the white nose), but we are currently replacing this device by an acoustic modem. This vehicle is moving at a speed of about 2m/s and its autonomy is about 3h. This AUV can reach a depth of 100m.
The second AUV is Taipan 2 which is a much more sophisticated vehicle with several sensors (see fig. 9.)

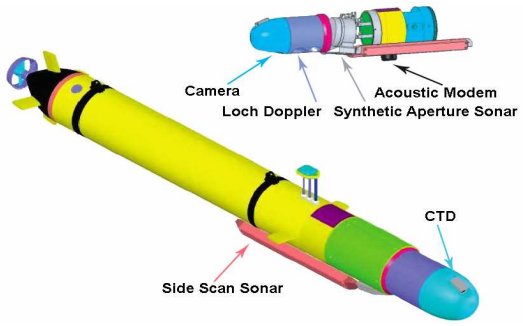


Fig. 9. Equipment of the new version of Taipan 2

These 2 AUVs carries a Tapac modem. These modems have a transmission power of 177 dB and a bandwidth of 6Khz. Their reception trigger level is equal to 15dB beyond environmental noise. Their radiation chart is omnidirectional. Its transmission frequency is 33 Khz (chirp modulation).
In this simulation we assume that the wind velocity is equal to 5 knot. Using the Wenz model we determine that the modems maximal communication range is equal to 1900 m, which correspond to constructor values.
In order to simplify the interpretation of the results, only one of the 2 vehicles is moving during the simulation. The first vehicle (Taipan 300) is moving along a straight line, restricted to the horizontal plane. Moreover, the pitch and roll angle are set to zero, that guarantees the vehicle to follow a straight line, forced to a constant depth.
The second AUV is fixed (actuators are set to 0) with a constant depth (the depth value is forced in the vehicle simulator).
In this simulation, Taipan 2 is in a fixed point and emits messages (constituted by the string "HELLO WOLRD TAIPAN 2 IS BROADCASTING A MESSAGE"). The trajectory followed by Taipan 300 is shown in figure 10. Each 350s a dot is drawn on the trajectory. The blue circle on the figure represents the transmission area of Taipan 2. Taipan 2 emits its message at t=60s, t=1500s et t=2300s. These instants are represented on the Taipan 300 trajectory by 3 squares.

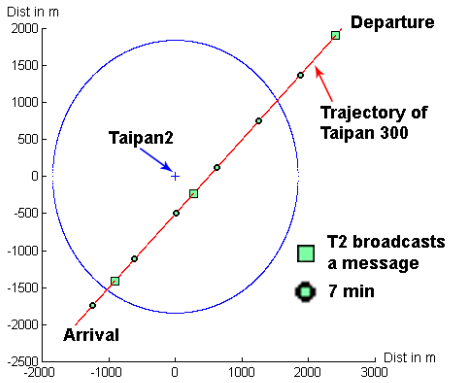


Fig. 10. Trajectory of Taipan 300

We can see that at t=0, Taipan 300 is out of the communication range of Taipan 2. At t=650s communication becomes possible. Finally at t=2382s Taipan 300 is out of the Taipan 2 communication range (figure 11).

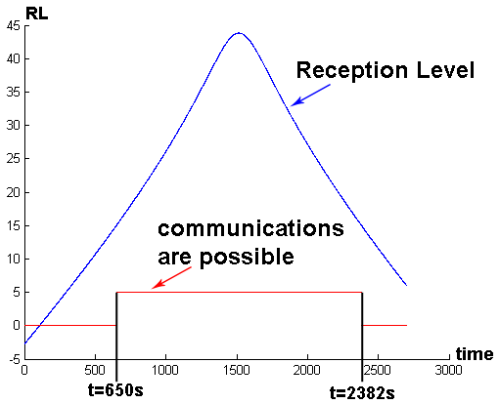


Fig. 11. Reception level during the trajectory of Taipan 300

Figure 12 represents a timeline with the date of emission/reception of messages from the vehicles. We assume that each transmission has a fixed delay and a null duration

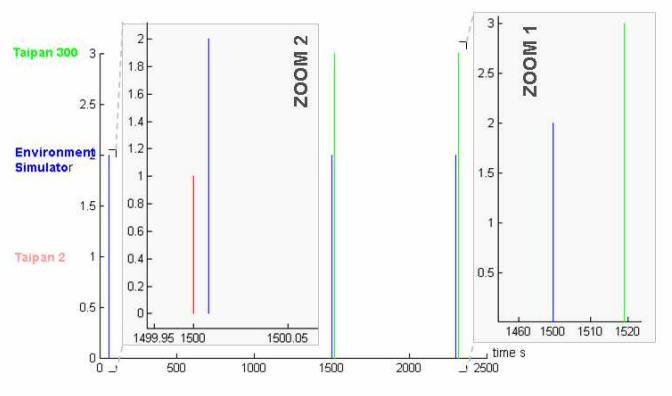


Fig. 12. Logs of the simulated mission

Figure 12 presents data logged by the 2 vehicles and the environment simulator. We distinguish these data on the graph by the height of their peak 1: Taipan 2, 2: Environment simulator, 3: Taipan300. On the "zoom 2" we can see that the first message sent by Taipan 2 is received by the environment simulator (few millisecond later) but never by Taipan 300 (no Taipan 2 peak after). The second and the third message are sent by Taipan 2 and received by taipan 300 after 20s ("zoom 1"). This duration corresponds to the latency and the necessary duration to send the message.

This simple simulation allows us to illustrate all mechanisms mentioned in the previous sections. AUV trajectories are enough to validate the environment and communications simulators.

CONCLUSIONS

In this paper we have presented the communication capabilities of the simulator *Thetis*. This HIL simulator plays an important role in the development of the controllers of our robots. *Thetis* is a real-time multi-vehicles simulator which needs to be completed (including complex exteroceptive sensors). We need to compare the results from the simulation with real experimentations. This is especially true for the underwater communications. Taipan 2 should be available in few months and we will be able to experiment the effective rate of communication between the AUV and a second acoustic modem located on a surface vessel.

Our contribution concerns the proposed simulator architecture. This architecture gathers all the important features that a simulator must include in order to allow simulation of flotilla. The models used are certainly less accurate than other within traditional simulators, but the structure of our simulator allows for easy evolution. The structure of our simulator allow us to test control laws of AUV involved in flotilla by taking into account all aspects of this field (e.g. including communication between vehicles).

The communications model we have presented is not very complex and accurate, but it could be useful for control design purpose. Interferences in communication are currently taking into account toward a function under development. This function verifies if the geographical zone of communications is sound proofed by an acoustical sensor. This functionality is very important in case of flotilla simulation. We have only evoked the use of this simulator with 2 robots in an underwater environment but it is generic enough to be used with other robots and thus we can imagine more complex scenarii like the coordination of AUVs and UAVs in order to achieve coastal monitoring.

REFERENCES

P. Ridao, E. Batlle, D. Ribas, M. Carreras (2004), "Neptune: a hil simulator for multiple UUVs", *OCEANS '04. MTS/IEEE TECHNO-OCEAN '04*

TI. Fossen (2002), "Marine Control Systems: Guidance Navigation and Control of Ships, Rigs and Underwater Vehicles", *Marine Cybernetics AS*

Suriano, Moriconi (2007), "A Distributed Simulator for the Development of the Unmanned Underwater Vehicles Control Software", *Robotics and Applications and Telematics*

Albus (1988), "System Description and Design Architecture for Multiple Autonomous Underwater Vehicles", *National Institute of standards ans Technology, Gaithersburg, MD, Technical Note 1251*

D.M Lane et al. (1998), "Mixing simulation and real subsystems for subsea robot development", *Proc; IEEE OCEAN'98, Nice, France, pp. 1382-1386*

T. Bielohlawek (2006), "SubSim – An Autonomous Underwater Vehicle Simulation System", *Ag Robotersysteme Fachbereich Iinformatik An Der Unuversität Kaiserslautern*

Chappell, Steven G.; Komerska, Rick J.(2001), "An Environment for High-Level Multiple AUV Simulation and Communication", *Proceedings of UI 2001*

Ridao, P., Battle, J., Amat, J., Carreras, M.(2001), "A distributed environment for virtual and/or real experiments for underwater robots," *Proceedings 2001 ICRA. vol.4, no., pp. 3250-3255 vol.4*

W. Hornfeld (2001), "DeepC, the German AUV Develoment Project", *status report of the STN ATLAS Elektronik GmbH*

Kobayashi et al (2001), "Development of an autonomous underwater vehicle maneuvering simulator," *OCEANS vol.1, no., pp.361-368 vol.1*

Carlson, E.A., Beaujean, P.-P., An, E. (2004), "Simulating communication during multiple AUV operations," *Autonomous Underwater Vehicles, vol., no., pp. 76-82*

O. Parodi, A. El Jalaoui, D. Andreu (2008), "Connectivity of Thetis, A Distributed Hybrid Simulator, with a mixed Control Architecture", *The Fourth International Conference on Autonomic and Autonomous Systems ICAS 08*

Doxygen, http://sourceforge.net/projects/doxygen/ accessed on 10/07

J.M. Spiewak, B. Jouvencel, P. Fraisse (2006), "A New Design of AUV for Shallow Water Applications: H160", *ISOPE'06: International Offshore and Polar Engineering*

LL. Whitcomb, DR. Yoerger (1999), "Development, Comparison and preliminary experimental validation of nonlinear dynamic thruster models", *IEEE journal of oceanic engineering*

M. Aucher (1981), "Dynamique des sous-marins", *Sciences et techniques de l'armement*

SA. Santos (1982), "Contribution à la conception des sous-marins autonomes : Architecture des actionneurs, architecture des capteurs d'altitude, et commandes référencées capteurs", *Thèse de l'Ecole nationale supérieure des mines de Paris*

L. Berkhovskikh, Y. Laysanov (1982), "Fundamentals of Ocean Acoustics », *N.Y: Springer*

G.M. Wenz (1962), "Acoustic ambiant noise in the ocean: spectra and source", *Journal of American Acoustic Society, vol 34, pp. 1936-1956*