HIL Simulator for AUV with ContrACT

Silvain Louis, David Andreu, Karen Godary-Dejean, Lionel Lapierre

HIL Simulator for AUV with ContrACT

Silvain LOUIS[1,2,3], David ANDREU[1], Karen GODARY-DEJEAN[1], Lionel LAPIERRE[1]

Abstract
Le processus de conception d'un système est composé de différentes étapes de validation. La validation est d'autant plus importante dans le contexte de systèmes embarqués critiques, et donc soumis à des contraintes temporelles strictes, comme c'est le cas pour la robotique sous-marine en milieu naturel. Notre méthodologie de conception démarre par l'étude théorique et finit bien sûr par les essais sur le terrain, mais le passage par la simulation est obligatoire afin de valider les concepts au plus tôt. De par l'augmentation de la complexité des missions et la difficulté qu'apporte l'expérimentation dans le milieu subaquatique, la simulation Hardware-in-the-Loop (HIL) est un outil très précieux. Nous avons donc conçu un simulateur HIL de notre robot, qui est utilisé pour la validation de nos missions dans le cadre de l'étude de la biodiversité marine du lagon de Mayotte. Il est basé sur le middleware ContrACT, lui-même installé sur le système d'exploitation temps réel Linux Xenomai. Notre simulateur se veut le plus transparent possible afin de minimiser son impact sur le contrôle, mais également le plus précis possible dans les modèles et la gestion du temps afin de rendre la validation la plus réaliste.

Keywords
Real time — Validation — HIL

[1] *Department of Robotics, LIRMM, Montpellier, France*
[2] *Mayotte University Center, Mayotte, France*
[3] *Marine biodiversity and usages, MARBEC, Montpellier, France*

Introduction

Nowadays mobile robotic systems are more and more complex and, as a consequence, their validation is difficult. This is particularly true for underwater robotics missions, where the robots have to fulfill specific task in dynamic, unknown and hard environment. The design process of such systems and missions starts with the theoretical study and finishs with on-the-field trials. The simulation stage is necessary to validate concepts earlier. Due to increase of mission complexity and difficulty of experiments in the underwater environment, the Hardware-in-The-Loop (HIL) simulation, close to reality, is a very useful tool. The HIL simulation replaces only the sensors and actuators by the simulator and keeps the robot's control architecture within the test loop.
At LIRMM[1] in collaboration with the research center MARBEC[2] and the CUFR[3] Mayotte, we are developing the robot JACK[4] for multiple missions, like fishes observation or coral study. We also developed KARST exploration missions. To validate the control architecture and the missions, we developed a HIL simulator for the JACK robot. The main feature of this simulator is its architecture. This one allows modular decomposition, accurate timing control and a controlled mode switching.

At first, this document presents the applicative, robotics and architectural context of the HIL JACK simulator. The place of the HIL simulator in the flow of our design process is explained. Then, it follows with the overall presentation of the HIL JACK simulator and its integration with the robot control architecture. And last, the JACK simulator architecture and particularly its real-time features are presented.

1. Context

1.1 The Environment: Lagoon of Mayotte

The main mission of CUFR Mayotte is environment observation of the Mayotte lagoon. For that, biologists use divers specialized in the fish counting, to get pertinent data on the population of different submarine ecosystems. This method is dangerous, difficult and expensive. Thus, the method developed by CUFR Mayotte, MARBEC and LIRMM is to use a robot to carry sensors related to their experiments.

We are focusing on 3 missions : The transect, the punctual observation and the species tracking. The transect is the basis of most of the marine biologist work when they need to take a census of species population (fishes in our cases). It consists on following a straight line and recording the fishes population along it. In this case, the robot is autonomously controlled to follow a dedicated feature (straight oriented line, or environmental feature). The punctual observation allows to observe a static element of the environment with different angles, while the robot is controlled to remain on the geodesic

[1] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, France
[2] MARine Biodiversity, Exploitation and Conservation
[3] Centre Universitaire de Formation et de Recherche
[4] http://ciscrea.net/produit/details/Mini+ROV+JACK+100+et+300+m

sphere centered on the observed element. The species tracking objective is to follow a dynamic element (e.g. animal) in the environment at a fixed distance, and to observe it under an angle chosen by the user.

This applicative context has an impact on the control architecture of the robot because of missions complexities, especially if we are interested in ensuring reliability at the architectural level. For example, during an experiment, the user can at any time change the mission. These unpredictable changes and especially the transient effects of these changes have a great importance on the architecture. Furthermore, in real missions, the robot could move closed to some fragile environment, which might be unknown and dynamic. To simulate this type of missions, this proximity requires a great precision in the JACK simulator and thus imposes constraints on the simulator architecture.

1.2 The Robot : JACK

To perform these missions, we choose to use the robot JACK as a basis. The EXPLORE team modified the robot [1] to enable high flexibility in electronics, mechanics and software. The mechanics and electronics flexibility of JACK consists mainly in the possibility of plugging multiple sensors (depending on the mission) without heavy modification. Software flexibility is underlying many aspects as modularity in the control architecture but also in the JACK simulator.

1.3 The middleware : ContrACT

The control architecture of JACK is based on the middleware ContrACT [2] which provides primitives and real-time mechanisms. This middleware runs on a real-time Linux patched with Xenomai. ContrACT is composed with 3 levels (Figure 1) : the decisional layer with Supervisors, the executive layer with operational Modules and between these two layers the Scheduler which manages the Modules execution depending on the Supervisors decisions.

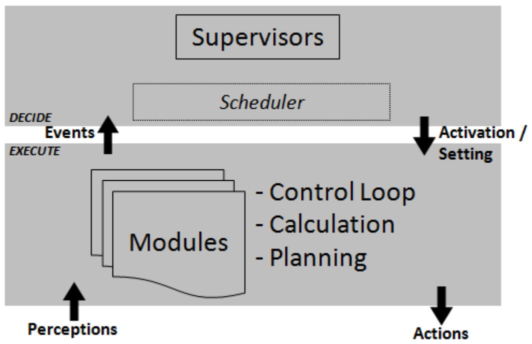

Figure 1. 3-layer ContrACT based architecture

The operational modules are the elementary blocks of the architecture that contain the basic functionalities. Modules are assembled into schemes. All modules of a given scheme run at the same frequency. The Scheduler triggers the execution of the modules at the frequency of the corresponding scheme, according to precedence constraints. Supervisors drive scheme selection and activation/deactivation, and have an event-triggered execution.

ContrACT modules are real-time Xenomai tasks, each of them with an execution period and a defined maximum duration. They communicate each other according to the publishers/subscribers mechanisms.

1.4 Notation

Let $\{U\}$ be the universal coordinate frame and $\{B\}$ be the body frame. In the sequel, η_U expresses the system state in $\{U\}$, ν_B is the system velocities in $\{B\}$ and $\dot{\nu}_B$ is the system acceleration as stated on Equation 1 and illustrated at Figure 2.

$$\begin{aligned}\eta_U &= [x,y,z,\Phi,\theta,\Psi]^T \\ &= [X_U, A_U]^T \\ \nu_B &= [u,v,w,p,q,r]^T \\ &= [V_B,\omega_B]^T \\ \dot{\nu}_B &= [\dot{u},\dot{v},\dot{w},\dot{p},\dot{q},\dot{r}]^T \\ &= \left[\dot{V}_B,\dot{\omega}_B\right]^T\end{aligned} \tag{1}$$

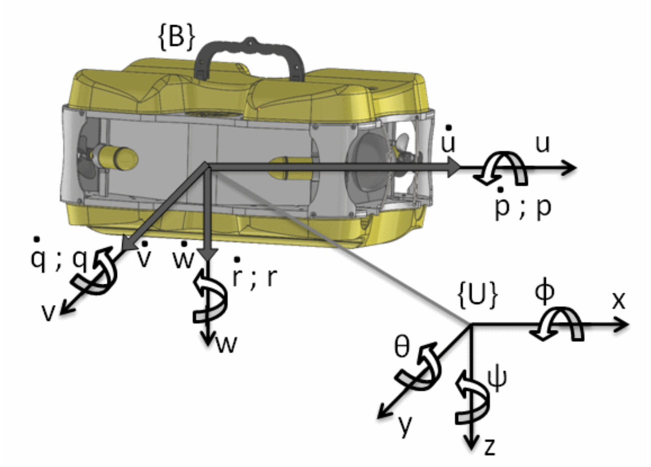

Figure 2. Definition of universal and body frame, for JACK

2. Methodology of validation

Our validation methodology consists of 4 steps (Figure 3).

Like all design processes, we start by a theoretical analysis to find the appropriate equations of the control. The first step is dedicated to the validation of the robotic functionalities, in terms of control performances (e.g. stability analysis). This can be done with, for example, the Matlab software, verifying the algorithms on some specific data inputs, coming from the JACK simulator. In this case, Matlab simulates the JACK hardware.

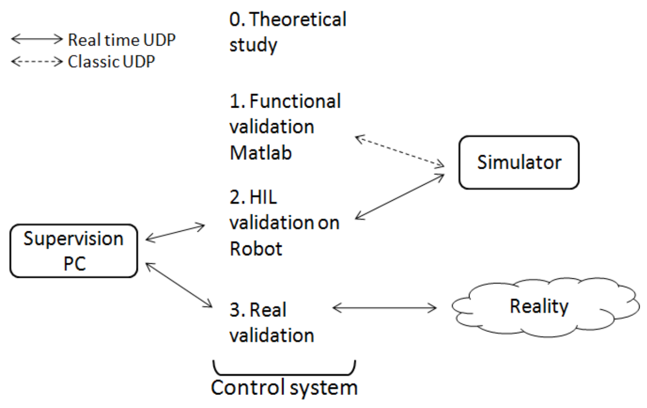

Figure 3. Validation methodology

The next and very important step is the HIL validation. This step validates the implementation of the algorithms / equations into ContrACT modules and their robotic functionalities [3] with external environment (Simulator). This step is necessary to verify the respect of the time constraints of processes (period and duration) within the simulated environment.

Finally, the last step is the validation in real environmental conditions. This last step is also important because simulators (even HIL) remain an approximation (more or less complete and accurate) of reality. The on-the-field experiments are useful to test the system with real constraints: reaction of algorithms to noise (even if also taken into account within the simulator), robustness to model uncertainties, disturbance rejection (sea current, ..) and behavior in case of failures.

In the sequel, we focus on the third step: the HIL validation. The next section is dedicated to the presentation of the JACK simulator.

3. Hardware in the Loop validation

Hardware In the Loop validation allows to consider the effective control architecture (its implementation on the target) within the simulation loop [4]. It also allows to take into account the hardware of the robot, except for the elements physically interacting on the environment. Sensors and actuators are replaced by virtual devices that simulate the perception and action on the simulated environment.

3.1 Simulation platform architecture

The simulation set-up (see Figure 4) relies on 3 entities: the robot with its embedded control architecture (hardware and software), the HMI operator dedicated computer and the simulation computer in charge of simulating the environment as well as interactions between the robot and the environment. The JACK robot (the hardware and the control algorithms of the embedded architecture) and the HMI operator computer are exactly the same than during the real missions.

The communication of the simulation platform relies on an Ethernet network, with communications based on the UDP/IP

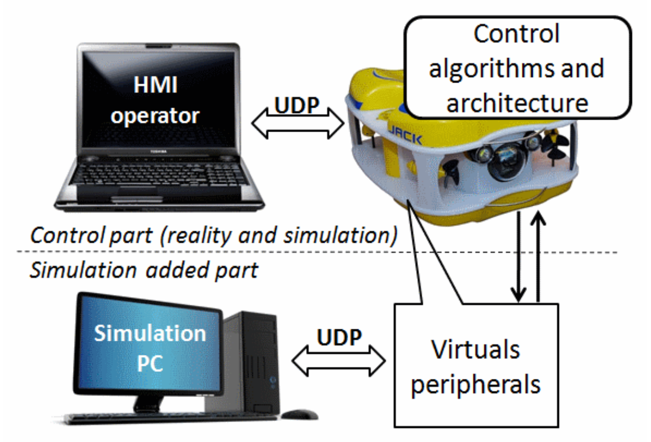
HMI operator
UDP
Control algorithms and architecture
Control part (reality and simulation)
Simulation added part
Simulation PC
UDP
Virtuals peripherals

Figure 4. JACK HIL simulation platform

stack. UDP messages contain all the variables from and to the simulator (actuators and sensors values). Real-time constraints are taken into account both from hardware and software points of view. Two distinct domains of collision are used to connect the robot embedded controller to the simulator and to the HMI operator. This avoid having to face medium access management on Ethernet which is a probabilistic network. The robot controller exploits one communication module for each link, favouring modularity as well.

Both logical links ensure temporal decoupling of associated processes, e.g. the simulation process runs asynchronously with the control one. This is essential since any temporal coupling between simulation and control would mask the lack of reactivity, the potential unstability, etc ... On the robot controller side, the two communication modules respectively run in a periodic way regarding simulation and in an aperiodic way regarding HMI operator. Indeed, communication with the simulation takes care of controlling robot entities (sensors ans actuators being in the simulator) at a fixed and given frequency. Regarding the simulator, it sends sensors data to the robot controller as soon as they are available, since some sensors, like sonar ones, do not have a fixed acquisition time (acoustic wave propagation depends on the distance to the environment elements). On the other side, communication with the HMI operator controller is event-based, ie the robot controller is waiting for messages which are sent only when the operator (on the HMI controller) reacts.

3.2 Switching between simulation and reality

To switch between simulation and reality contexts, only sensors and actuators dedicated modules have to be changed in the control architecture (by switching between experimentation configuration and simulation configuration), ie without modifying the control scheme. In both cases, data to be exchanged are the same, as well as inter modules communication mechanisms.

The communication scheme for the real mission (Figure 6) instantiates communication drivers with hardware modules (reading sensors, actuators update). The communication

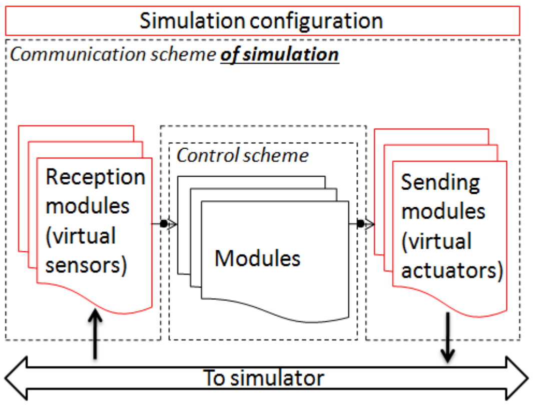
Simulation configuration
Communication scheme of simulation
Reception modules (virtual sensors)
Control scheme
Modules
Sending modules (virtual actuators)
To simulator

Figure 5. Simulation architecture

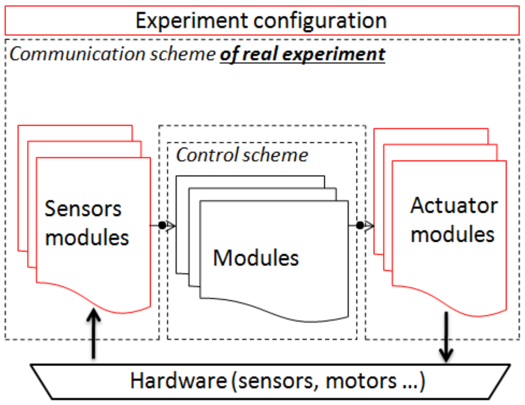

Figure 6. Real experiment architecture

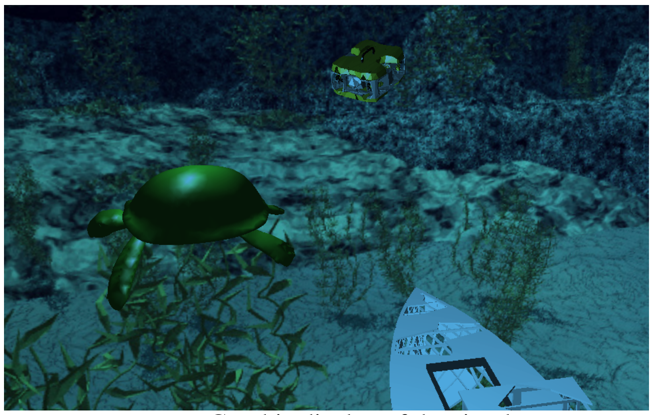

Figure 7. Graphic display of the simulator

Figure 8. Underwater environment CryEngine

scheme for the simulation (Figure 5) uses virtual peripherals according to which data are exchanged with the simulator.

4. The simulator

4.1 A two-part architecture

The Jack simulator architecture consists of two parts:

- A set of processes dedicated to all the computations related to simulation (integration, physical models (of the robot or the environment), sensors, actuators ...)
- The display rendering 3D (3D Obj models [5], light, camera simulation, visualization (robotlike or Godlike) [6]).

The first part is implemented using ContrACT, dealing with real time constraints. This allows to obtain good integration frequency and a desired precision in the scheme execution periods. The second part is implemented with QT5 and uses OpenGL for 3D display (Figure 7) [7].

This splitting of the simulator architecture allows to change the 3D generator without changing the physical simulator. It could be possible for example to use an underwater environment design with CryEngine[5] (Figure 8).

[5]http://cryengine.com/

This also allows for a remote display of the JACK simulator. Indeed, it is possible to separately deploy simulation processes on a dedicated computer [8] with sufficient performances to obtain accurate simulation results, and graphic display on another computer. The decoupling between the simulation part (high and precise frequency), and the display part (low frequency and event-based) allows greater precision of execution of the simulation part, which is the most important for the validation objective, since it avoids overloading the processor and/or delaying simulation processes.

Table 1 shows the mean and deviation of the actual scheme integration execution period which has a period order of 1ms with and without 3D display integration. We observe that without the 3D display, the execution is regular with an average of 0.9993ms (1ms required period) and only 2.75% of deviation.

The influence of the display rendering on the execution period is also shown Figure 9. We can see on this Figure the irregularity period of the scheme integration with 3D display. This happened because of the 3D display priority over that of integration scheme priority. This priority is dynamically defined by ContrACT scheduler (earliest deadline first algorithm) and so this phenomenon is not mastered. In this case, there is a significant impact because the 3D display process is potentially long.

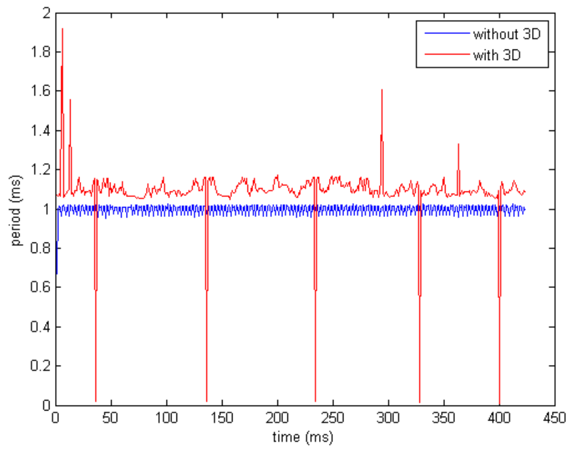

Figure 9. Effective period of scheme execution with and without 3D display

Table 1. Effective period of scheme execution

	Mean (ms)	Deviation (%)
With 3D display	1.0926	13.42
Without 3D display	0.9993	2.75

4.2 Real time simulation

Taking into account real time constraints in a simulator is critical [9] because the accuracy of the simulation is dependent on the integration period (dt) (Equation 2 and Figure 10). Except [10], few authors explicitly mention this constraint and its respect. For the following results, we use the same method of integration (Euler) (Equation 2) and we compare the results (drift position after 5s of simulation) in case of perturbations like the perturbations we mentioned regarding 3D display.

$$\begin{aligned} V_B &= V_B + (\dot{V_B} \cdot dt) \\ V_U &= M_B^U \cdot V_B \\ X_U &= X_U + (\dot{V_U} \cdot dt) \end{aligned} \tag{2}$$

We have considered that the robot follows a straight line with null orientation angles ($M_B^U = \mathbb{1}$) and we apply an acceleration in the body frame, with the form $\dot{V_B} = cos(t) + 0.2m/s^2$. This acceleration is first integrated in the body frame, transformed in the universal frame, and integrated again to get the position of the robot. The integration is performed according to two parameters: the period (1ms or 100ms) and the error period : 0%, 2.75% or 13.42% (in accordance with the previous test). With a dual integration, the position equation is $X_B = -cos(t) + \frac{0.2 \cdot t^2}{2} + c$. Here are the positions (Table 2) and errors (Table 3) compared to the theoretical value, after 20s of simulation. Figure 10 shows the position error evolution with different period errors.

We observe a strong dependence between the quality of the simulation and the frequency parameter (period and periodicity error). This confirms the need to use a tool that provides high and accurate execution frequency (particularly for the integration scheme, ie a realtime based tool).

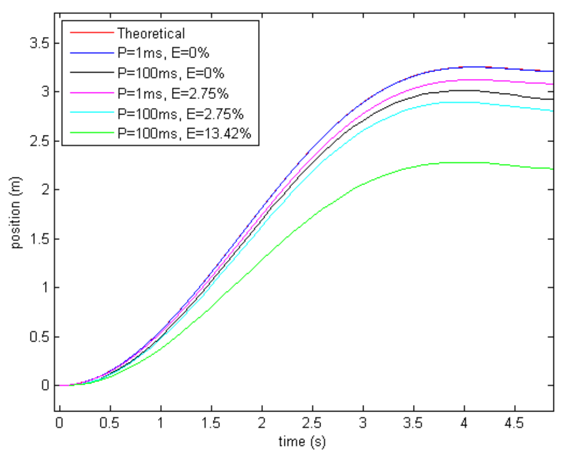

Figure 10. Simulated position with error on the periodicity

Table 2. Position (m) after 5s

	Theoretical	E=0%	E=2.75%	E=13.42%
P=1ms	40.59	40.58	38.97	
P=100ms		39.39	37.83	29.82

4.3 Software architecture of the simulator

To obtain a high and accurate execution frequency of the schemes of the simulator (particularly integration scheme), we use the middleware ContrACT.

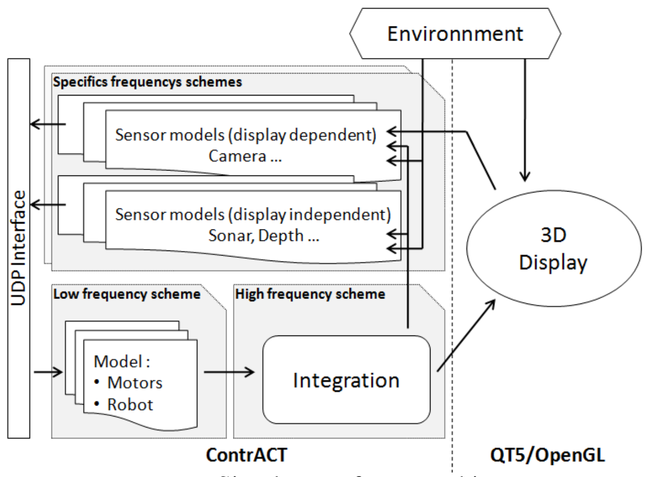

Figure 11. Simulator software architecture

ContrACT allows to manage schemes with different periods, like for instance the high frequency execution scheme containing models integration, the low frequency execution scheme corresponding to communications, and the aperiodic scheme dedicated to specific sensors as acoustic camera (frequency of which is sensor dependent). There are 3 type of simulation sensor schemes.

Table 3. Position error (%) during the first 5s

	E=0%	E=2.75%	E=13.42%
P=1ms	0.029	3.99	
P=100ms	2.96	6.80	26.55

The first type is the periodic scheme with the same period than the sensor. Each execution of the scheme, all the sensor modules simulate the corresponding phenomena and generate sensor data. For example, this is the case for temperature or depth sensors.

If the sensor algorithm is too complex and it can be cut in parts, there is the second type of scheme. The second type is the periodic scheme with a higher period than the sensor. Each execution cycle of the scheme, the module calculate a part of the data. For example, for a sensor like profilometric sonar, each parts corresponds to a distance measurement for a fixed angle. Then at each distance (ie for each angle), computed data is sent to the controller. For other sensors, like DVL, the data are sent to the controller only when all the parts are computed.

The third type is the aperiodic scheme. When the sensor measurement time is not constant, the period can not be fixed.This is the case for the acoustic camera. The distance measurement is dependent of the flying time, and so of the environment. To match this phenomenon, we also simulate the flight time in the simulator.

All these kind of schemes are taken into account, and of course the simulation dedicated omputer must be powerful enough.

5. Conclusion

This article briefly presents an HIL simulator of the underwater robot JACK. The HIL validation is a useful step (and tool) in the context of underwater robotics because of the complexity of on-the-field experiments. This is particularly true in our context where some of the experiments have to be done at Mayotte. This simulator can now be used by the development team before every on-the-field experiment to previously validate the desired behavior. It offers accurate and real time simulation. Thanks to its modularity, the HIL simulator can be improved modifying model's precision, adding new phenomena like currents of water, etc. It is also possible to add other robots to deal with flotilla for more complex missions, taking into account the model of acoustic communication.

Acknowledgments

Thanks to Robin PASSAMA for his ContrACT explanations, Sebastien VILLEGER^3 and Thomas CLAVERIE^2 for their PhD supervision and their collaboration on this subject. Thanks to Ecole doctorale de Montpellier and CUFR for grants.

References

[1] S LOUIS. Rapport de stage industriel de fin d'études. Technical report, 2014.

[2] R Passama. Environnement de développement : ContrACT.

[3] C.J Cannell, D.J Stilwell, and J.A Austin. A simulation tool to support the development of adaptive sampling algorithms for multiple autonomous underwater vehicles. *Autonomous Underwater Vehicles*, pages 127 – 133, 2004.

[4] O Parodi. *Simulation hybride pour la coordination de véhicules hétérogènes au sein d'une flottille*. PhD thesis, Université de Montpellier, 2008.

[5] D.M Lane, G.J Falconer, G.W Randall, N.D Duffy, J.T Herd, P Chernett, J Hunter, M Colley, J Standeven, V Callaghan, J Smith, J Evans, A Woods, J Penrose, G.A Whittaker, D Smith, and I Edwards. Mixing simulations and real subsystems for subsea robot development specification and development of the core simulation engine. In *OCEANS '98*, pages 1382 – 1386, 1998.

[6] T Bielohlawek. An autonomous underwater vehicle simulation system. 2006.

[7] A Göllü. A simulation environment for the coordinated operation of multiple autonomous underwater vehicles. *Proceeding of the 29th conference on Winter simulation*, pages 1169 – 1175.

[8] P Ridao, J Battle, J Amat, and M Carreras. A distributed environment for virtual and/or real experiments for underwater robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3250–3255, 2001.

[9] O Parodi, L Lapierre, and B Jouvencel. A real-time multivehicules hybrid simulator for heterogeneous vehicules. In *International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[10] R Bono, M Caccia, and G Veruggio. Simulation and control of an unmanned underwater vehicle. In *International Conference on Intelligent Robots and Systems (IEEE)*, pages 1573–1578, 1995.