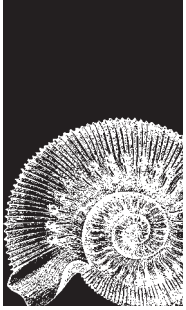

JMR

Journal of Maritime Research, Vol. VIII. No. 1, pp. 3-18, 2011

ISSN: 1697-4840

HARDWARE IN THE LOOP SIMULATION BENCHMARK FOR AUTONOMOUS MARINE VEHICLES

A. Moreno[1,2], D. Moreno[1,3], D. Chaos[1,4] and J. Aranda[1,5]

Received 10 December 2010; in revised form 17 February 2011; accepted 18 March 2011

Abstract

The use of autonomous vehicles is becoming ubiquitous due to the versatility and flexibility that they display in the execution of individual and cooperative task, coupled with the fact that their use avoids placing human lives at risk. Closely related to these autonomous systems are the simulation tools. These tools are essential to test the correct design and behaviour of the modelling and control algorithms theoretically. A poor design could have dramatic consequences to the vehicle itself, the rest of vehicles and even the environment in a real scenario. In this work a benchmark for unmanned vehicles is presented. The main objective is to have a generic framework in which to develop and test control algorithms for coordinated and cooperative tasks between different kinds of vehicles. The tool is constructed in a modular way, so as any kind of vehicle can be simulated (or tested) with a slight modification of the program. The benchmark can run continuous and discrete (DEVS and non-DEVS) simulations and it is constructed over LabVIEW as Hardware-In-The-Loop (HIL) simulation platform. Using the same system for simulation and real experiments reduces the cost of hardware tests and facilitates enormously the portability of the theoretical design to the real world. This tool is oriented both for researchers and students, to test their own control algorithms both theoretically and experimentally.

Key Words: benchmark, simulation, control, autonomous vehicles, DEVS, LabVIEW.

[1] Universidad Nacional de Educación a Distancia , Calle Juan del Rosal 16, 28040 Madrid, Spain. [2] PhD Student, Email: amoreno@bec.uned.es, Tel. +34 913987147, Fax. +34 913987690. [3] PhD Student, Email: dmoreno@dia.uned.es, Tel. +34 913987942, Fax. +34 913987690. [4] PhD Student, Email: dchaos@dia.uned.es, Tel. +34 913987157, Fax. +34 913987690. [5] Full Professor, Email: jaranda@dia.uned.es, Tel. +34 913987148, Fax. +34 913987690.

INTRODUCTION

The use of autonomous vehicles in different research and commercial areas has been increasing in the last few years for reasons that have to do with autonomy, flexibility, and the new trend in miniaturization. Moreover, their use in collaborative tasks allows for the realization of complex missions, often with relatively simple systems; see for example the many works related in the robotics field, (Yamaguchi, 2003) for example.

Closely related to these autonomous systems are the simulation tools. These tools are essential to test the correct design and behaviour of the control algorithms theoretically, as many of these autonomous systems has delicate and expensive instrumentation or are working on dangerous conditions. For example, a poor control design of a vehicle could have dramatic consequences to the system itself, other vehicles and even the environment in which it works. For this reason, coupled with the fact that real tests have high cost on time and money, construction of a test bed satisfies the need for a simulation environment that researchers and students can use to implement and analyse cooperative and non-cooperative control algorithms for different kind of unmanned vehicles (aerial, terrestrial and marine) theoretically and experimentally.

There are many works related with the development of tools for simulation in the autonomous vehicles or robotics field. In (Rasmunsens & Chandler, 2002) a simulator for aerospace vehicles is constructed in MATLAB/Simulink to test cooperative control algorithms. The vehicles are 6 DOF and with embedded flight software. In (Luke et al, 2005) a single-process, discrete event simulation core and visualization library written in Java is developed, designed to be flexible enough to be used for a wide range of simple simulations, but with a special emphasis on swarm multi-agent simulations of many agents (up to millions). (Vaughan, 2005) proposes a simple benchmark for multi-robot simulator performance. For a deeper survey in existent simulation environments, the readers are referred to (Craighead et al, 2007) which presents a survey of computer based simulators for unmanned vehicles, covering a wide spectrum of vehicles. This report surveys 14 widely available simulators, showing the main characteristics for an adequate simulation environment.

These works deal with the simulation field, without testing the actual systems. The aim of our work is to have a modular and easily reconfigurable test bed, so as any kind of vehicle can be simulated and tested with a slight modification of the program by hardware-in-the-loop (HIL) simulations. In this sense there exist some works that include hardware-In-The-Loop (HIL) simulations to test the actual hardware and software with the same tool, without the need of field experimentation. In (Jung and Tsiotras, 2007) the modelling and experimental identification results for a small unmanned aerial vehicle (UAV) are presented. A hardware-in-the-loop (HIL) simulation environment is developed to support and validate the UAV autopilot hardware and software development in MATLAB/Simulink. In (Ridao et al, 2004) a multi-

vehicle, real-time, graphical simulator based on OpenGL that allows hardware-in-the-loop simulations is developed for unmanned underwater vehicles (UUV).

Based on previous works, the software framework presented in the current document is expected to be a modelling and control simulation benchmark for unmanned aerial vehicles (UAVs), unmanned marine vehicles (UMVs) and autonomous underwater vehicles (UAVs), that allows the end user to define and customize models and controls of the overall simulation or instead exchange them by hardware such as real vehicles (HIL simulations). Therefore we have the same tool for simulation and real experiments, facilitating enormously the portability of the theoretical design to the real world and reducing significantly the costs of real tests.

The benchmark runs continuous and discrete (DEVS and non-DEVS) simulations to overcome all the spectrum of possible designs. One step forward respect to previous simulations tools for autonomous vehicles is this possibility of running DEVS simulations. The DEVS M&S formalism (Zeigler et al., 2000) provides several advantages to analyse and design complex systems: completeness, variability, extensibility, and maintainability. Furthermore, in a near future will offer options to include environmental disturbances within the simulation. The recreation scenarios vary from unique vehicle to multi-vehicle study. These scenarios are designed to test bottom-line challenges of control of autonomous vehicles. Hereafter, they are described and divided among two different control layers: individual and multiple vehicle autonomous control.

SIMULATION PLATFORM

As it has been commented, the autonomous vehicle simulation platform runs simulations based on discrete events (DEVS and Non-DEVS) along with continuous simulations. Moreover, these simulations can be performed centralized, distributed, remotely within real or virtual time context. In this section we explain the platform architectural and conceptual design, starting from the base on which is built, stepping in the architecture and communication protocol, and ending with the system modelling and configurations.

Basis

As if it was a building, the construction materials are made of DEVS models (also non-DEVS and continuous models, as well as Hardware) and both the simulations protocol and models follow rules based on its formalism. The platform is built with LabView programming tools.

DEVS

The Discrete Event System Specification is a general formalism for discrete event system modelling based on set theory (Zeigler et al., 2000). It allows representing any

system by three sets and five functions: input set (X), output set (Y), state set (S), external transition function (δ_{ext}), internal transition function (δ_{int}), confluent function (δ_{con}), output function (λ), and time advanced function (ta). The DEVS M&S formalism provides several advantages to analyse and design complex systems: completeness, verifiability, extensibility, and maintainability. DEVS can reproduce Discrete Time System Specifications (DTSS) and approximate continuous modelling paradigms (Differential Equation System Specification (DESS)). That is, DEVS is able to describe discrete event, discrete and continuous systems. Thus, simulation tools based on DEVS are potentially more general than other tools including continuous simulation tools (Kofman, 2004). Furthermore, DEVS conceptually separates models from the simulator, making possible to simulate the same model using different simulators working in centralized, parallel or distributed execution modes. Recently, a working group of the Simulation Interoperability Standards Organization has developed a standard (Zeigler et al., 2008) to support interoperability of DEVS models implemented in different platforms as well as with legacy simulations.

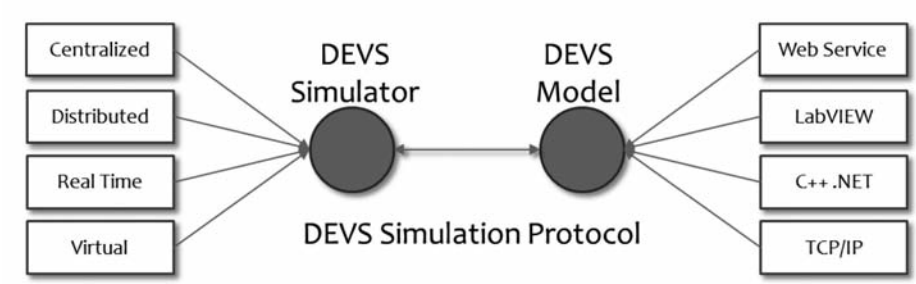

Figure 1: Conceptual Architecture of the Standard.

LabVIEW

LabVIEW's virtual instrument (VI) is a graphical programming language specifically designed for developing instrumentation, diagnostics, and data acquisition systems. Many engineering and scientific disciplines, both professional and academic, have adopted LabVIEW, which has resulted in a broad collection of libraries and legacy code.

Next, we highlight the characteristics of National Instruments software LabVIEW that convinced us to select LabVIEW as our software development framework:

— Hardware integration (HIL).
— Visualization and graph management.
— Real Time.
— Management of communication interfaces (USB, BT, TCP/IP, WS, etc.).
— Data display and user interfaces.

— Multicore programming.
— Multiple Targets and OSs.
— Multiple Programming approaches and interoperability with other languages, applications and paradigms (MATLAB/Simulink, DLL, .NET, ActiveX, etc.).

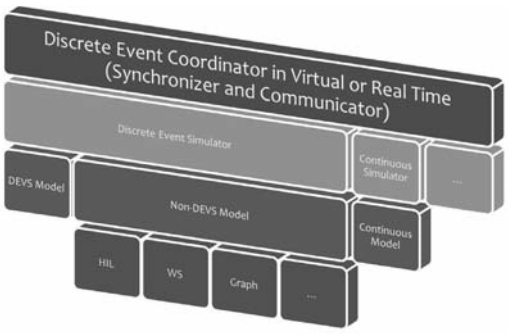

Figure 2: Architecture,

Architecture

The architecture is made of three layers: coordination, simulation and modelling. Each layer interacts with the upper or lower layer depending on the simulation protocol step. As seen on Figure 3 the coordinator synchronizes and communicates the simulators that may simulate discrete models, sample a continuous model or other action. A discrete event simulator can simulate a DEVS or a Non-DEVS model. A non-DEVS model can perform any action discretely, such as Hardware or remote Communication via USB, BT, WS or TCP/IP. While a continuous simulator samples a continuous model at discrete intervals.

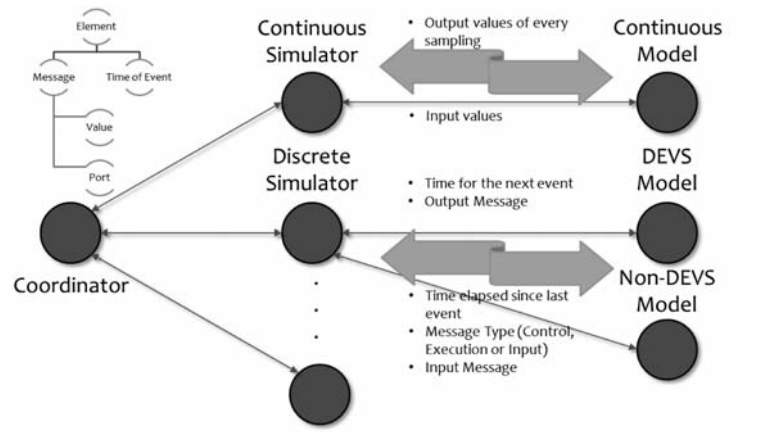

Figure 3: Communication Protocol.

Communication Protocol

The communication between coordinator and simulators is achieved by a common structure given by a message and a time event that indicates the time of execution of the next event. In summary, the coordinator communicates with the simula-

tor to send incoming messages, run by time or by a control message and associated event time. Whereas, simulators communicate with the coordinator to send outgoing messages with the time of the next event. Figure 4 illustrates this message exchange protocol with more detail.

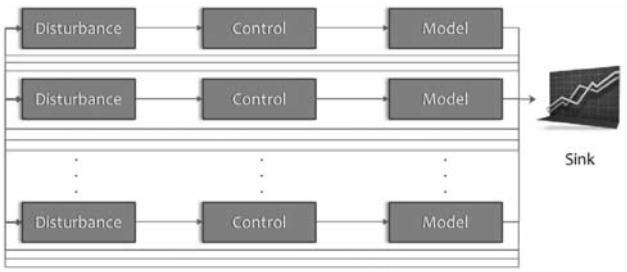

Figure 4: System Modelling.

System Modelling

The system modelling of an autonomous vehicle experiment scenario is achieved by defining three types of models per individual. A model that prototypes a UAV, UMV, or an AUV through a discrete event or continuous paradigm. It can also communicate with a real vehicle. The control module performs an intelligent navigation control depending of the overall control strategy over the previous model. Finally, the disturbance model injects alterations in the communications and bounds the field of vision of the vehicle.

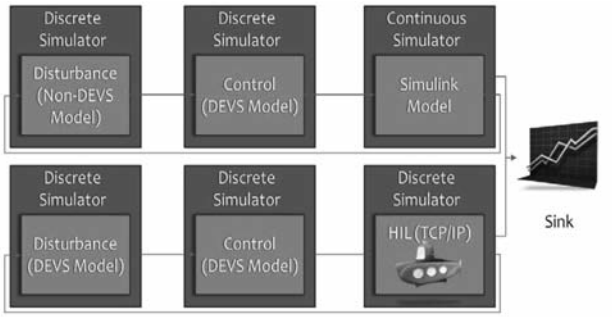

Figure 5: Example Configuration.

Possible Configurations

Each simulation component may be defined by a DEVS model, non-DEVS model, continuous model, or an interface that interacts with hardware, MATLAB Simulink, DLL, NET, etc. An example configuration is depicted by Figure 6. Also, a model can be built by the user according to a specific interface for each type. Therefore, a simulation module can behave in any manner while meeting the constraints of the simulation protocol that has been defined. Soon, our tool will provide customized interfaces for automatically link modules to standard communication interfaces such as SOAP Web Services.

GRAPHICAL USER INTERFACE

One key point of the simulation framework is providing a visual environment that meets the following characteristics; high degree of usability, wide functionality, and offer a pleasant and customizable interface. The platform GUI is divided in two

modules. On the one hand, a scenario editor, which provides a graphical interface for configuring an autonomous vehicle experiment. On the other, the simulation graphical environment, that illustrates the on-going experiment results.

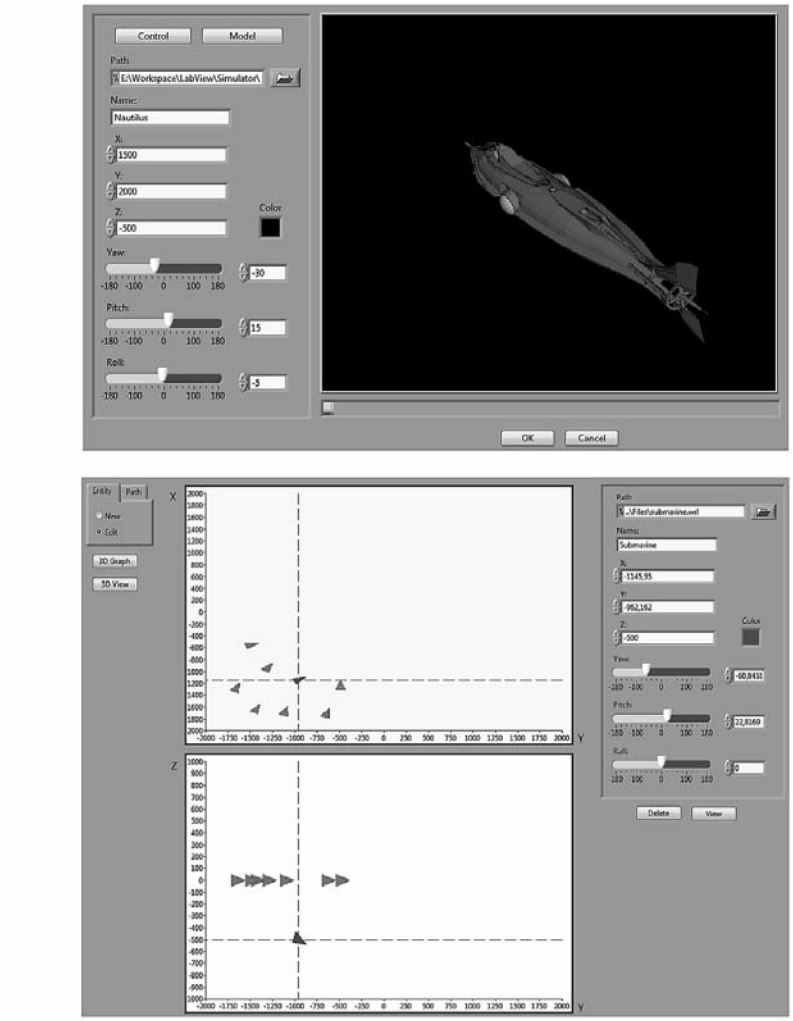

Figure 6: Scenario editor captures (a) and (b).

Scenario Editor

This tool offers the end user a wide range of possibilities to configure each autonomous vehicle entity. First, select and customize the vehicle's model and control strategy. Secondly, tweak the initial conditions parameters, as seen on Figure 6.a, coor-

dinates (X, Y, and Z) and Euler angles (yaw, pitch, and roll). Besides, it doesn't merely allow the user to configure each vehicle individually, but also gives an overall perspective of the world dimensions and the other vehicles of the mission (Figure 6.b).

Simulation Environment

The goal of this module of the graphical interface is to symbolize the outcomes of the simulation in progress. That is, two and three dimensional graphs that represent each vehicle's trajectory and current position over the simulation world, Figure 7.a. Additionally, illustrates a 3D environment with the autonomous vehicles 3D models aimed to denote reality in a more reliable manner, Figure 7.b. The sea surface effects have not been plotted to allow the visualisation of the AUV model in the Figure 7.b.

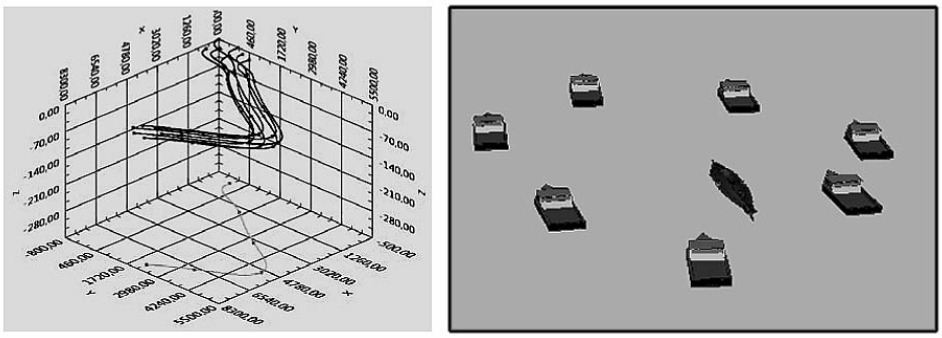

Figure 7: Simulation results (a) and capture of the simulation (b).

RUNNING OF THE SIMULATOR TEST BED

This section describes the results of simulations and real tests that illustrate the potential of the test bed developed for coordinated and cooperative control of unmanned vehicles. The first example shows how the tool developed allows the simulation of multiple and different vehicles for cooperative missions. The second example shows how the "hardware in the loop" allows the immediate conversion of a simulation example into a real test, substituting the vehicle model by the input and output interface of the vehicle.

Coordinated control of multiple unmanned vehicles

The formation control or swarm control is one of the most emerging topics in robotics field. In this example we consider an arbitrary number of underactuated surface marine vessels that must keep a certain formation during the tracking of an underwater target. For the sake of simplicity we consider a centralized strategy where each agent measures the range from the target to itself. The localization of the underwater

vehicle is done via trilateration algorithms (Alcocer, 2009). The formation control is based on a coordinated path following approach in which some time restrictions, maintained by the advance speed control, are introduced. For target trajectory prediction, an Information Filter is used, as defined in (Bar-Shalom et al., 2001). The details are omitted.

The desired positions for the agents depend on the formation required for the task. As it was mentioned, the objective in the underwater target tracking task is to keep the AUV localized by a mobile surface sensor network. As (Moreno-Salinas et al, 2010) describes, the vehicles must keep a regular distribution around the target, the size of which depends on the target depth. The agents can take any position around the target whereas they keep the desired regular formation. This characteristic provides more flexibility to the formation control and allows a simpler tracking control.

The simulation results are shown in Figure 7. We can observe how the regular formation is maintained along the target tracking task keeping the target projection in the centre of the vehicle formation, and therefore keeping the optimal formation to obtain the best accuracy possible, as described in (Moreno-Salinas et al, 2010).

From this example it is clear that with the developed simulator it is easy to carry out complex simulations where a high and arbitrary number of vehicles of different nature are involved in a cooperative and/or coordinate task.

Hardware in the loop: Tracking control of a hovercraft.

An interesting problem arises when any theoretical control that works fine in simulation must be tested in a real scenario. For this purpose, our simulator allows the hardware in the loop simulations, so with an immediate step we can change the simulation scenario by a real scenario, using a real vehicle instead of a theoretical dynamic model.

In this example a non-lineal tracking control law for a hovercraft is used. This kind of vehicle is non holonomic, and its study is very interesting from a theoretical point of view, as its behaviour is very similar of a surface craft. This non lineal control law is used to follow a circular trajectory, first in a simulation scenario, where a model of the hovercraft is run (from a previous identification process, see (Chaos, 2010)). Then it is tested in a real scenario, where the theoretical model is substituted by the vehicle communication interface in the simulator. The details of the control law are omitted due to space considerations, for a deeper analysis see (Chaos, 2010). This control law is used only as an illustrative example, because, as it was commented in previous sections, any control law (designed or existing) can be implemented for any vehicle, using the control law editor.

In Figure (8.a) we can observe the simulation results, how the hovercraft makes the trajectory tracking with high accuracy with the control law designed. In Figure

(8.b) the results in the real scenario are shown, where we can notice that the behaviour of the hovercraft is similar to the theoretical one. In Figure 9 a caption of the simulation environment is shown.

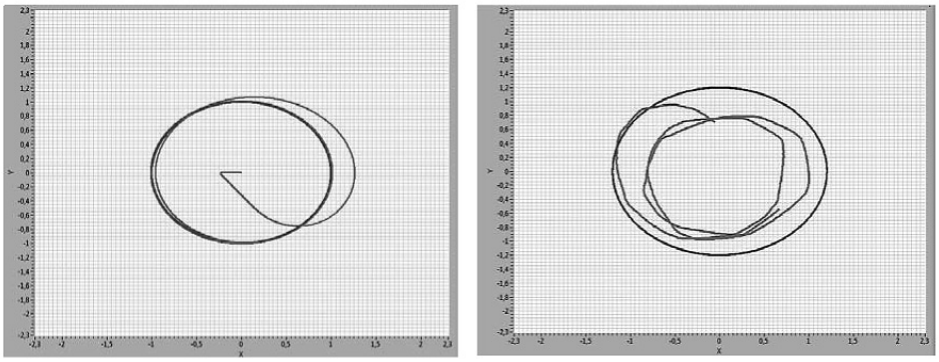

Figure 8: Simulation and HIL simulations results for tracking of a hovercraft.

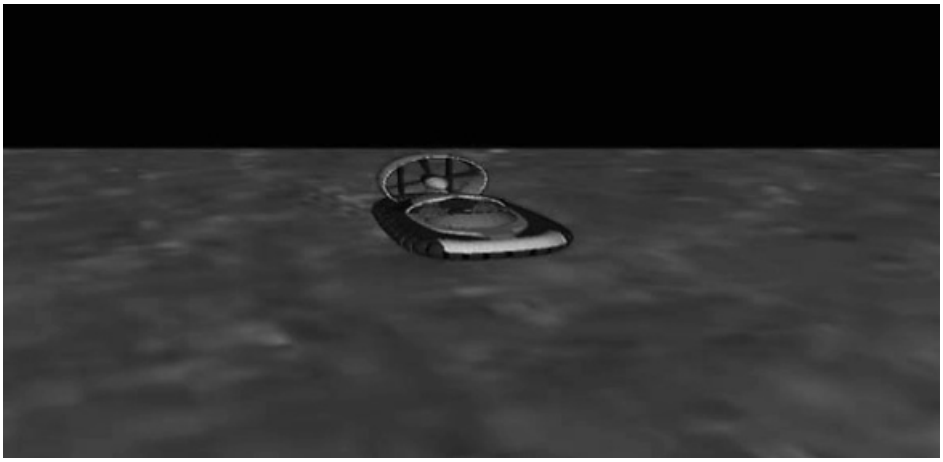

Figure 9: Capture of the simulation environment.

Therefore, we can test our control laws or models both in a theoretical and in a real framework in simple and fast way, without the need of implementing different systems or programs.

CONCLUSIONS

In the present work a modular and easily reconfigurable test bed have been developed, so as any kind of vehicle can be simulated (and tested) with a slight modification of the program. The test bed runs continuous and discrete (DEVS and no-

DEVS) simulations to overcome all the spectrum of possible designs. One step forward respect to previous simulations tools for autonomous vehicles is the possibility of running DEVS simulations. It is constructed in LabVIEW to allow HIL simulations. Therefore the same tool is used for simulation and real experiments, facilitating enormously the portability of the theoretical design to the real world and reducing significantly the costs of real tests.

Future work will aim at: i) extending the vehicle models to a higher variety of vehicles used in cooperative and coordinated tasks, ii) including in the simulator environmental perturbations, as winds, currents, waves, etc, to improve the simulation platform as well as new control problems, as collision avoidance, fail detection, localization, etc, iii) extending the tool to construct a remote laboratory via web for educational purposes, in which the students could interact with indoor vehicles together with the simulation platform.

REFERENCES

Alcocer; A. (2009). Positioning and Navigation Systems for Robotic Underwater Vehicles. PhD Thesis, Instituto Superior Tecnico, Lisbon, Portugal, 2009.

Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2001). Estimation with Application to Tracking and Navigation. John Wiley, New York, NY; 2001.

Jung, D. and Tsiotras, P. (2007). Modeling and Hardware-in-the-Loop Simulation for a Small Unmanned Aerial Vehicle. American Institute of Aeronautics and Astronautics, Conference and Exhibit, 7 - 10 May 2007, Rohnert Park, California.

Craighead, J., Murphy, R., Burke, J. and Goldiez. B. (2007). A Survey of Commercial & Open Source Unmanned Vehicle Simulators. 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10-14 April 2007.

Chaos, D. (2010). Nonlinear control of underactuated nonholonomic marine vehicles. PhD Thesis. Department of Computer Science and Automatic Control.

Kofman, E. Discrete event simulation of hybrid systems. SIAM Journal on Scientific Computing, vol. 25, no. 5, pp. 1771–1797, 2004.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K. and Balan, G. (2005). MASON: A Multiagent Simulation Environment. Journal Simulation, Volume 81, Issue 7, July 2005.

Moreno, D., Pascoal, A. M.; Alcocer, A; Aranda, J. (2010). Optimal Sensor Placement for Underwater Target Positioning with Noisy Range Measurements. 8th IFAC Conference on Control Applications in Marine Systems, CAMS 2010, Rostock, Germany, September 2010.

Rasmussen, S.J., Chandler, P.R. (2002). MultiUAV: a multiple UAV simulation for investigation of cooperative control. Winter Simulation Conference (WSC'02), vol. 1, pp.869-877

Ridao, P., Batlle, E., Ribas. D. and Carreras, M. (2004). NEPTUNE: A HIL Simulator for Multiple UUVs. OCEANS 04 Conference, 524 - 531 Vol.1, 9-12 Nov. 2004

Risco-Martín, J. L., Moreno, A., Cruz, J. M., and Aranda, J. 2009. Interoperability between DEVS and non-DEVS models using DEVS/SOA. Spring Simulation Multiconference.

Vaughan, R. (2005). Massively multi-robot simulation in stage. Swarm Intelligence, Volume 2, Numbers 2-4, 189-208, Springerlink.

Yamaguchi, H. (2003). A distributed motion coordination strategy for multiple nonholonomic mobile robots in cooperative hunting operations. Robotics and Autonomous System, 43, pgs 257-282, 2003 Elsevier Ltd.

Zeigler, B. P., Kim, T., and Praehofer, H. (2000). Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, 2000.

Zeigler, B. P., Mittal, S. and Hu, X. (2008). Towards a formal standard for interoperability in M&S/System of Systems integration. GMU-AFCEA Symposium on Critical Issues in C4I, 2008.

HARDWARE EN EL PUNTO DE REFERENCIA DE SIMULACIÓN DE LOOP PARA VEHÍCULOS AUTÓNOMOS MARINOS

RESUMEN

El uso de vehículos autónomos está presente en numerosas áreas de robótica debido en parte a la versatilidad y flexibilidad que demuestran en la ejecución de diversas tareas de forma tanto individual como cooperativa, junto al hecho de que su uso evita el poner vidas humanas en peligro para la realización de tareas que entrañen cierto peligro. Estrechamente ligado a los vehículos autónomos se encuentran las herramientas de simulación. Estas herramientas son esenciales a la hora de comprobar el correcto diseño y funcionamiento, tanto de los algoritmos de control como de los modelos usados, de una forma teórica. Esta etapa de simulación es totalmente necesaria en el desarrollo de algoritmos de control, ya que un diseño deficiente puede provocar consecuencias dramáticas para el vehículo, los vehículos vecinos o el medio en donde se desarrolle la tarea. En este sentido en este trabajo se presenta un banco de pruebas, que pueden ser tanto teóricas como experimentales, para vehículos autónomos. El principal objetivo es tener un marco en el cual desarrollar y probar diferentes algoritmos de control para tareas cooperativas y coordinadas entre diferentes clases de vehículos. La herramienta está construida de forma modular de tal manera que cualquier clase de vehículo autónomo pueda ser simulado sólo con ligeras modificaciones del programa principal. Los procesos a simular podrán ser continuos, y discretos (DEVS y noDEVS), estando construido sobre el programa gráfico LabVIEW como una plataforma de simulación "harware-in-the-loop". De esta manera se usa el mismo sistema para simulación y tests reales, reduciendo el coste de las pruebas sobre el hardware y facilitando enormemente la portabilidad de los diseños teóricos al mundo real. La herramienta esta orientada para su uso tanto por investigadores como estudiantes, de forma que puedan probar sus propios algoritmos de control de forma teórica y experimental.

MÉTODOS

El software desarrollado en el presente trabajo pretende ser una herramienta de simulación para modelado y control de diferentes clases de vehículos autónomos. Puede ejecutar simulaciones continuas y discretas (DEVS y noDEVS). Una ventaja adicional frente a otros simuladores es la posibilidad comentada de realizar simulaciones DEVs. El formalismo DEVs proporciona diversas ventajas para analizar y

diseñar sistemas complejos como integridad, variabilidad, extensibilidad y mantenibilidad. La recreación de escenarios varía desde un solo vehículo a múltiples vehículos de diferentes clases. Las características principales de la plataforma de simulación son:

Arquitectura: La arquitectura del sistema se compone de tres capas: coordinación, simulación y modelado. Cada una de las capas interacciona con las otras dependiendo del paso del protocolo de simulación. El coordinador sincroniza y comunica los simuladores. Los simuladores discretos podrán ejecutar simulaciones DEVS y no-DEVs.

Protocolo de comunicación: la comunicación entre el coordinador y simulador es realizada mediante una estructura común dada por un mensaje y tiempo de un evento que indica el tiempo de ejecución del próximo evento. En resumen, el coordinador se comunica con el simulador para mandar mensajes de entrada, por tiempo o por mensajes de control y tiempo de evento asociado. Mientras tanto, los simuladores se comunican con el coordinador para mandar mensajes de salida.

Modelado: el sistema de modelado de un escenario experimental de un vehículo autónomo esta basado en tres tipos de modelos. Un modelo para el vehículo a través de paradigmas discretos o continuos, o la comunicación con el vehículo real (simulaciones HIL). El modulo de control que implementa el control de navegación dependiendo de la estrategia de simulación deseada a priori. Finalmente, el modelo de perturbación que introduce alteraciones en las comunicaciones y limita el campo de visión de los vehículos.

Configuraciones posibles: cada componente de la simulación debe ser definido como un modelo DEVS, no DEVS o continuo, o como un interfaz de comunicación para interactuar con el hardware, MATLAB, DLL, .NET, etc.

Interfaz de usuario: la interfaz de simulación debe poseer las siguientes características: alto grado de usabilidad, alta funcionalidad, y ofrecer una interfaz agradable y fácil de manejar. La interfaz esta dividida en dos módulos. El editor de escenarios que proporciona un interfaz gráfico para la configuración del experimento y el entorno gráfico de simulación que va mostrando en tiempo real el resultado de la simulación mediante modelos en 3D.

CONCLUSIONES

En este trabajo se ha desarrollado un banco de pruebas modular y fácilmente reconfigurable, de tal manera que se pueda simular diferentes clases de vehículos autónomos simplemente con ligeras modificaciones del programa. El programa ha sido construido de una forma jerárquica, con la comunicaciones entre vehículos modeladas de forma explicita, incluye herramientas gráficas y los datos son salvados para un posible estudio de los mismos "fuera de línea". La herramienta puede ejecutar simulaciones tanto continuas como discretas (DEVS y noDEVS), de forma que se

abarque todo el espectro de posibles situaciones de diseño. Un elemento diferenciador respecto otros simuladores de vehículos autónomos es este hecho de poder hacer simulaciones DEVS. Además ha sido construido sobre LabVIEW para permitir simulaciones "harware-in-the-loop". De esta manera tenemos una única herramienta para la simulación teórica y los test experimentales sobre los componentes hardware, facilitando enormemente la portabilidad de los diseños teóricos al mundo real y reduciendo significativamente los costes de los tests experimentales.