An Application Interface for UCHILSIM and the Arrival of New Challenges

Juan Cristóbal Zagal, Iván Sarmiento, Javier Ruiz-del-Solar

Department of Electrical Engineering, Universidad de Chile,
Av. Tupper 2007, 6513027 Santiago, Chile
{jzagal,isarmien,jruizd}@ing.uchile.cl,
http://www.robocup.cl

Abstract. UCHILSIM is a robot simulator recently introduced in the RoboCup Four Legged League. A main attractive of the simulator is the possibility of reproducing with accuracy the dynamical behavior of AIBO[1] robots as well as providing good graphical representations of their surroundings on a soccer scenario. Learning over virtual environments can be performed with successful transfers of resulting behaviors to real environments. Previous version of the simulator had a major drawback: Only the UChile1 team could make use of it since the developed system had high dependency on the team code. In this paper we present results of a development work which was envisioned on the first presentation of UCHILSIM; an application interface for allowing any OPEN-R software code to be directly used over the UCHILSIM simulator. The possibility of having this kind of tool opens a great field of developments and challenges since more people will develop OPEN-R software, even without having the robotic hardware but the simulator. Other recent improvements on our simulator are briefly presented here as well.

1 Introduction

Simulation is becoming a hot topic on robotics. An increase in the number of simulation related publications can be observed over the main journals of the field. New robotics simulators are emerging into the field either for research or commercial applications, ranging from general purpose simulators to specifics to certain robotic task, such as grasping or arm soldering trajectory design. New developments on robotic simulators are being recognized by the research community, examples of these are the robotic grasping simulator GraspIt! [7] that won the NASA Space Telerobotics *Cool Robot of the week* prize on February 2002, the general purpose robotic simulator WebBots [6] that recently won the second place on of EURON[2] *Technology Transfer Award*, and a publication related to our UCHILSIM simulator won the *RoboCup Engineering Challenge Award* on RoboCup 2004[3].

[1] AIBO and OPEN-R is a trademark or registered trademark of Sony Corporation.

[2] EURON is the European Robotics Research Network, http://www.euron.org

[3] The robot soccer world federation http://www.robocup.org

Under our perspective the main attractive of simulation in robotics is to enhance learning, although this doesn't seems to be the main reason for its current expansion on the field. Industrial robots on the automotive industry for example offer simulators of their products which allow users to get familiar with them by practicing their kinematics before running the real hardware. Researchers use simulators for testing new approaches on arbitrary scenarios and to use multiple robot agents without having the actual hardware. Applications of simulation on this field are very broad, and the generation of good simulators is being enforced by the increase on computer power as Moore's law establish [8], as well as the exponential improvements on graphics hardware power [4]. Main criticism to robot development under simulation given by Brooks [2] several years ago cannot still be defended while facing the current achievements of computer simulations. But overall a main justification to fight towards simulation on robotics is given by the emerging supporting theories such as the Theory of Mind [9]. The authors have also proposed some theoretical basis such as Back to Reality [14].

Towards Improving our Simulator

UCHILSIM [12] is a dynamics robotic simulator introduced for the RoboCup Four Legged League; the simulator reproduces with high accuracy the dynamics of AIBO motions and its interactions within a soccer scenario. The simulator has shown to be a useful tool for learning into virtual environments with successful behavioral transfers to reality. In [13] experiments are shown on the generation of dynamics AIBO gaits form simulation to reality, in [14] experiments are presented about learning to kick the ball using the Back to Reality approach and the UCHILSIM simulator.

We believe that this is a relevant kind of tool to promote on future developments of RoboCup. Aiming at improving further our simulator such that it can become a general use platform for the four legged league, we have generated a list of main requirements to fulfill by a simulator: (1) Use a generic and flexible definition of robots, (2) allow to incorporate other user defined objects into the simulation, (3) allow multiple robots to share a common virtual environment, (4) use of different robotics platforms, (5) use a fast and realistic dynamics engine, (6) provide good graphics and visualization methods for the desired task, (7) use a fast and robust collision detection system, (8) provide good programmatic interfaces in order for anybody to use the system, and (9) run over multiple host platforms. Prior to this publication UCHILSIM satisfied points 1 to 7, however there was no programmatic interface for allowing any generic OPEN-R software to run over UCHILSIM. This paper deals precisely with this point. Here we present an application interface for the UCHILSIM simulator which will allow spreading the use of this tool.

The reminder of this paper is as follows, section 2 present the implemented interface for UCHILSIM, section 3 present examples of using and testing this interface, section 4 discuss possible applications of this tool, on section 5 we describe briefly some recent improvements on the simulator and finally on section 6 we present conclusions and envision future challenges for this system.

2 An Application Interface for UCHILSIM

The UCHILSIM simulator has been restricted to the use of the UChile1 RoboCup four legged team. This restriction was expressed on the form of several code dependencies among the simulator and the team source code. In order for any OPEN-R developer to make use of the simulator it would have involved rewriting a large amount of code for each particular application.

The idea of building an interface for the simulator was announced on the first UCHILSIM publication [12], however there were just some ideas at that time. Among these ideas we considered first to construct a simulator programmatic interface by writing a large number of primitive functions for accessing the simulated hardware similarly as one does when using the OPEN-R Sony Software Development Kit [10]. Writing such programmatic interface would have been almost equivalent to generate a complete SDK for our simulator. A main drawback of this approach is that it would involve for any user to rewrite its particular application using the set of functions provided by a parallel SDK. Fortunately we found another alternative at a lower level, before going into its details we should describe briefly the OPEN-R SDK for which it was implemented.

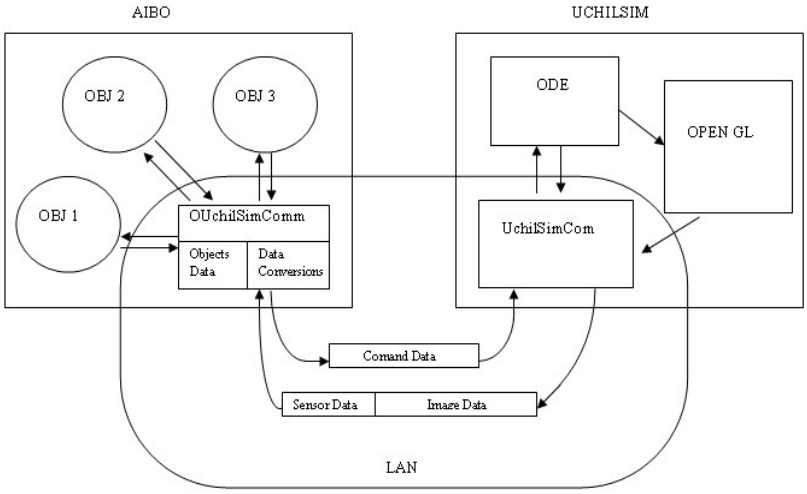

Fig. 1. Diagram showing how the OUChilSimComm and the UChilSimComm interface objects exchange command, sensor and image data trough the network.

2.1 Description of the Target SDK

The OPEN-R SDK is an interface proposed by Sony in order to promote the developments of robots software and hardware, refer to [10]. The interface enhances the development of modularized pieces of software which are called OPEN-R objects.

The objects are implemented as independent processes which run concurrently and intercommunicate by means of messages. The connections among objects are described by communication services described on a boot time readable file. This is a very important characteristic since it allows objects to be replaceable components at an operative level.

Under OPEN-R the interface to the system layer is also implemented by means of inter object communication. There is a specific object provided on OPEN-R called *OVirtualRobotComm* which is in charge of providing a low level driver interface of data with the robot hardware by means of exchanging command, sensor and image data with other objects, this relation is established trough the same configurable communication service file.

2.2 Description of the New Interface

The idea is to replace the low level object interface OVirtualRobotComm provided under OPEN-R by another OPEN-R object designed for interfacing data with a simulated robot under UCHILSIM instead of a real robot. The interface object that we have developed is called *OUChilSimComm*. This object is designed to run either over an AIBO robot or on a host computer by using OPEN-R Remote Processing [10]. Although this object runs embedded in the space of OPEN-R objects, it should interchange data with the UCHILSIM simulator which runs on a host computer. This communication is performed by network TCP/IP connection among OUChilSimComm and an interface developed at the simulator side. We call this interface as *UChilSimComm*. Figure 1 shows a diagram of the relations among these interface modules. Command data is collected at the OUChilSimComm module and then dispatched to the simulator across the network; similarly sensor data and image data are packed by the UChilSimComm interface at the simulator side using fixed sized data structures. Then data is exchanged using TCP/IP connections either across platforms or over the same host machine (using the local host IP). There are many choices for implementing that since OUChilSimComm runs over the robot or on a host computer as well as the other OPEN-R modules.

Data Structures and Packets: The interface between the module and the simulator uses two different and independent network connections, one for the sensor and image data and another for the command data. Data packets used for image and sensor data are of fixed length while packets used for transmitting command data are of variable length. OUChilSimComm maintains a buffer of sensor and image data which is constantly updated with data coming from the simulator. This data is dispatched to the calling objects as requested. The command data which is received from the objects is immediately dispatched to the simulator. The following is a description of each data flow and how the structures are arranged. This structures slightly differ depending on the robot model being used (ERS7/ERS210).

Sensor Data: The digital sensor data is generated at UCHILSIM with a similar rate as the existing on the real robot. After each dynamic integration step, the actual joint

sensor values are collected from the virtual robot joints. Simplistic values are given to the acceleration sensors, as well as for the switch sensors. A data transmission packet is filled with all these values containing a header with timestamps related to the data. When the packet is received by the OUChilSimComm object a *OSensorFrameVectorData* OPEN-R structure [10] is constructed by calling the corresponding data constructor provided with OPEN-R, and then filling the corresponding fields with the incoming data. If this data structure is requested by other objects then it is dispatched, otherwise the data is stored into a limited size buffer.

Image Data: The digital image data is generated at UCHILSIM with a similar rate as the existing on the robot camera. After each new YUV image frame is acquired from the simulator a data structure equal to *OFbkImage* [10] is generated, and then the data is split into three packets for the simplicity of network transmission, each packet contains their corresponding time stamps and sequence identifiers. When the packet is received by the OUChilSimComm object the OFbkImage structure is reconstructed and then the *OFbkImageVectorData* [10] structure is updated by directly incorporating OFbkImage data. The OFbkImageVectorData is constructed by using an existing OPEN-R constructor.

Command Data: The commands are generated by any running OPEN-R object and then transmitted to the OUChilSimComm module. The OPEN-R data structure which contains these data is *OCommandVectorData* [10]. Once this structure is received the task of OUChilSimComm is to extract the joint command reference values and timing data, disregarding any LED command. Then a transmission packet is generated containing a header which indicates command type, number of data frames and timing data. When the packet arrives UCHILSIM (trough UChilSimComm interface), the corresponding joint commands are executed as position references for the motors located at each joint, these reference values are taken by the corresponding PID controller located at each simulated joint.

3 Using and Testing the Interface

As it can be seen the interface is implemented at the system level rather than at the programmatic level, and therefore the developers don't need to perform modifications on their code, just to re define the communication services and to recompile their own code to the host computer in case this is desired to be used. The user should modify the stub.cfg file replacing the OVirtualRobotComm service connections with the OUChilSimComm service. Then on the target directory it should make sure that CONECT.CFG file contains the right connections.

From the interface side, a configuration file should be updated indicating the corresponding network connections where the robot and UCHILSIM process are located. It should also be specified the corresponding robot model.

The presented interface has been successfully tested with the simple source code examples provided with OPEN-R, such as *MovingLegs7* [10]. The test was performed

running all processes on a single host computer (2.5 GHz processor, 512 Mb ram, no graphics accelerator). We have also tested our own source code by applying simple vision related tasks such as ball following behaviors.

4 Possible Applications of this Tool

We believe that one of the major applications of this type of tool is that it will allow more developers to enter RoboCup and in general to program software for robots by directly using OPEN-R without having to access directly the robot. Another main advantage of this tool is to accelerate developments into the four legged league by providing a standard test bed for new ideas. Certainly with this simulator it is no longer necessary to worry about destroying the hardware or even about the long time required for specific experiments. Since any OPEN-R code can be used, with this application interface team code can be evaluated using more statistically proven strategies such like making two teams to compete for very long trials, this kind of tool can be established as a standard way of testing code prior to real competitions for example. It is also important to test and to compare specific parts of team code especially given the trend of the league on the modularization and specialization on the functions of vision, localization and strategy. Since the interface is based on the idea of modular objects which can communicate along a network it is possible to have extensive distributed systems which share a common dynamical environment. This can be even extended to the use of Internet. The idea of parallelism can be exploited further and we can use parallel computing for example for evolving team behaviors.

5 Other Recent Improvements on UCHILSIM

Multitasking and Task Scheduling: An important limitation of the simulator was the high processor consumption due to graphics. Therefore we have implemented separated processing threads on the simulator, one is specific for the graphics and the other is specific for the dynamical computations. This allows us to have always fast dynamics while having just a best effort result over the graphical representation. A consequence of this is that the graphics seen by the user might appear blinking when the window size is too large; however there is always good speed for the dynamics. Eventually the graphic representation can be totally disconnected from the simulation; this might be useful for experiments on which there is no need of having the robot camera, such as offline locomotion learning tasks. Another implemented alternative is to have hard control of the different tasks such as graphics and dynamics computations which should be executed. In this respect we have implemented a task scheduler which allows us to control specific timings for the different tasks.

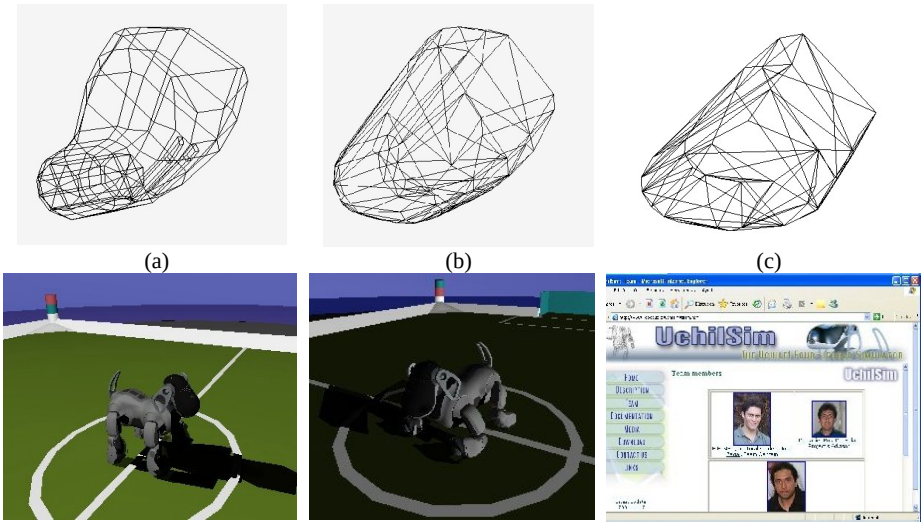

Fig. 2. Mesh improvements on a portion of the ERS-7 robot leg, on (a) the original mesh is presented, (b) is the result of computing the convex hull of the mesh and finally (c) shows the result of reducing the level of detail by using GLOD library tools. On (d) and (e) examples are shown of the implementation of the Shadow Volumes CG technique. On (f) a screenshot of the new website of UCHILSIM is shown.

Graphics and Mesh Improvements: We have incorporated the computer graphics technique of Shadow Volumes [11]. The attractive of this technique is that it allows producing precise shadows on real time. Since overlapping shadows generate areas of varying intensities this allows to reproduce the effect that can be observed when we have strong sources of light over the soccer scenario; the robot produce shadows in the directions opposite to the different sources of light. Figure 2 shows some examples of using this technique. Other improvements that we have introduced, proposed in [1], are CMOS filters and introduction of camera aberration over the AIBO lenses. We have implemented an tested also a set of offline tools for optimizing the robot meshes which are provided by Sony. In particular the interest for the simulator is to have representative and simple shapes for collision detection. By using the quick hull algorithm[4] in conjunction with the GLOD library [3] we are able to considerably reduce the amount of points which are used for describing a given shape. Figure 2 shows result of applying this tool, a given limb segment originally consist of 178 vertex, after applying convex hull we get a model of 84 vertex, finally after applying GLOD tools we get just 68 vertex on our model.

[4] please refer to http://www.qhull.org

6 Conclusions and Further Challenges

It was presented an application interface for the UCHILSIM simulator. We envision the arrival of new challenges related to the optimization of this tool and the use that other teams might give to it. From our team perspective there are still some tasks to fulfill; these are the development of a networking interface, improvement of sensor models and probably a sound interface. People at our group for example is currently quite motivated on performing experiments with distributed simulation using parallel computing for producing behavioral learning over large search spaces.

Acknowledgements

We thank Paul Vallejos, Carlos Gortaris and Alvaro Neira for their collaboration. This research was partially supported by Departamento de Investigación y Desarrollo (Universidad de Chile) under project ENL 05/03.

References

1. Asanuma, K., Umeda, K., Ueda, R., Arai, T.:Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics In: 7th International Workshop on RoboCup 2003, Lecture Notes in Artificial Intelligence, Springer, Padua, Italy (2003)
2. Brooks, R.A.: Flesh and Machines: How Robots will Change Us. Phanteon, USA, February (2002)
3. Cohen, J., Luebke, D. Duca, N., Schubert, B.: GLOD: A Geometric Level of Detail System at the OpenGL API Level. IEEE Visualization 2003, Seattle, WA (2003).
4. Kelly, F., Kokaram, Anil.: Graphics hardware for gradient-based motion estimation. Embedded Processors for Multimedia and Communications, San Jose, California (2004) 92-103
5. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
6. Michel, O.: Webots: Professional Mobile Robot Simulation. In: International Journal of Advanced Robotic Systems, Vol. 1, Num. 1 (2004) 39-42
7. Miller, A., Allen, P.: GraspIt! A versatile simulator for Robotic Grasping, IEEE Robotics and Automation Magazine, December (2004) 110 -122
8. Moore, G.E.: Cramming More Components Onto Integrated Circuits. Electronics Journal, April 19 (1965)
9. Scassellati, B.: Theory of Mind for a Humanoid Robot. First IEEE/RSJ International Conference on Humanoid Robotics (2000)
10. Sony Corporation.: OPEN-R SDK Programer's Guide and Level2 Reference Guide. Published by Sony Corporation (2004)
11. Watt, A., Watt, M.: Advanced Animation and Rendering Techniques, Theory and Practice. Addison-Wesley, New York (1994)
12. Zagal J.C., Ruiz-del-Solar J.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: 8th International Workshop on RoboCup 2004, Lecture Notes in Artificial Intelligence, Springer, Lisbon, Portugal, (2004)
13. Zagal J.C., Ruiz-del-Solar J.: Learning to Kick the Ball Using Back to Reality. In: 8th International Workshop on RoboCup 2004, Lecture Notes in Artificial Intelligence, Springer, Lisbon, Portugal (2004)
14. Zagal J.C., Ruiz-del-Solar J., Vallejos, P.: Back to Reality: Crossing the Reality Gap in Evolutionary Robotics. IAV 2004 the 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal (2004)