OPEN GRAPHICS SYSTEM FOR 3D-MAPPING VALIDATION ON ROBOTICS

J.V. Catret, M. Mellado & D. Puig

Departamento de Ingeniería de Sistemas y Automática (DISA)
Universidad Politécnica de Valencia (UPV)
Camino de Vera, s/n, 46022 – Valencia, Spain
Tf: +34-963879575, Fax: +34-963879579,
{martin,jvcatret,dpuig}@isa.upv.es; http://www.isa.upv.es

ABSTRACT: An innovative product for 3D-mapping validation on robotics is introduced on this paper as result of the application of a multi-robot programming and simulation software to that purpose. The software, called *Virtual Robot Simulator (VRS)*, has an open design that facilitates the integration of user's applications by means of external components that give access to its main functionalities. For example, realistic and smart graphics representation of 3D-maps can be easily generated using simple functions. In fact, the user can concentrate the efforts on implementing sensor models, signal filters, 3D-mapping methods and so on, and leave the arduous work of graphics representation to *VRS*. Then *VRS* acts as a "true" virtual cell composed of robots with sensors moving on their environment and reflecting the actions of the user's application. Even more, the specific physical sensor and robot is transparent to the user when decides to take the best of the hierarchical software structure offered to manage this kind of systems. A practical example is commented on the paper.

1. INTRODUCTION

During last years, much research on robotics has been strongly focused on sensor-based 3D-mapping, both, for mobile robots [1], [2], [3] and robot-arms [4], [5], [6], [7]. The main efforts usually lay on the sensor model, the data fusion technique or the 3D-mapping method. But one of the problems for researchers is how to evaluate the results of the obtained 3D-maps and also how to compare their results with other published results even considering that physical and hardware platforms can be different. Obviously, the most appropriate way should include a realistic graphical representation of the 3D-map. Unfortunately, to cover graphical aspects in addition to the already hard work of sensor control and robot control complicates greatly the labor of the researchers. This unaffordable work produces some times that the work finishes only on the theoretical part, making difficult to understand the scope of the obtained results. This gap can only be beaten if the researchers have access to a graphical testing platform with a simple interface that allows them to integrate easily their work with a comfortable interface. The work presented on this paper tries to cover this gap.

In this paper, the use of a graphical software application as platform to evaluate methods for 3D-mapping is explained. *Virtual Robot Simulator (VRS)*, created originally for remote robot control, programming and simulation, has become a useful tool for this purpose. On the next section, the origin and main features of *VRS* are shown. Then, the open software architecture is explained, making special focus on how to integrate user's applications on *VRS* and a hierarchical structure offered as help for the user's development. Users can make their own applications and integrate their results in *VRS* in such a way that they can avoid to develop the graphical work and the robot simulation, served both from a *VRS External Access Library (VREAL)*. Even more, the access to robot systems and I/O devices can be easily managed with the software components of *VRS*. In the last section, an example is presented to justify the simplicity of use but the high capability of the application and its result on *VRS* is shown and explained.

2. VIRTUAL ROBOT SIMULATOR (VRS)

VRS is a C++ application developed on MS Visual C++ environment by the DISA-UPV Tele-Robotics Group, with a representation based on the OpenGL graphic library under Windows operating systems (Figure 1). *VRS* can be applied on multi-robot systems for their off-line programming and simulation as well as for on-line programming and monitoring. Its begging, on 1998, was motivated by the need to have a remote application for graphical programming, monitoring and simulating multi-robot cells on the control system GENERIS (Generalised Software Control System for Industrial Robots) [8] developed by European Commission Joint Research Centre (JRC - Ispra/Italy).

The easy and friendly user interface of *VRS* reduces down notably learning time, even for new users on this kind of applications, in such way that becomes adequate for educational, research and industrial purposes. For example, the investigation of new virtual reality trends, such as haptic systems, is going on based on *VRS*. *VRS* has been also successfully used for robot off-line programming on rapid prototyping in industrial applications, thanks to its flexible architecture that allows data integration from and to other applications.

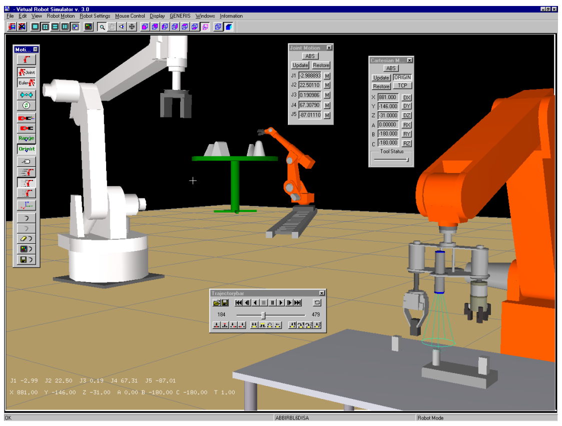

***Fig. 1**. Graphic Interface of VRS*

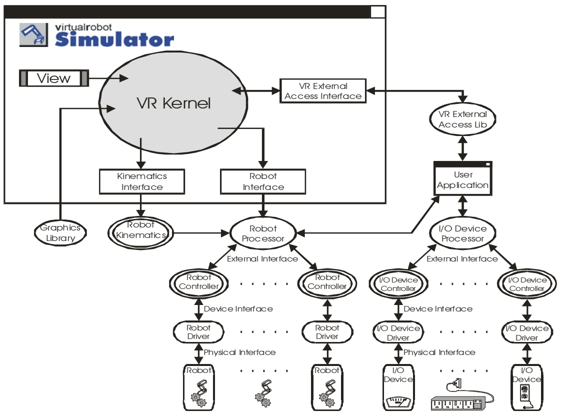

***Fig. 2.** VRS Open Software Architecture*

3. VRS OPEN ARCHITECTURE

The *VRS* software architecture is based on four parts (Figure 2), the *VR* Kernel, the user interface, a set of internal modules required as external access interface and a set of external components that the user can replace with his/her own components according to simple coding and compiling constraints.

The *VR* Kernel has been designed through a hierarchical structure of classes. This hierarchy has been implemented following the principle of inclusion, where an entity is modelled as a set of smaller parts. The external components (note that each one requires an interface internal module for its integration with *VRS*) are Dynamic Link Libraries loaded on memory during execution time. The user has available a set of coding templates (and compiling choices) to develop new components that can be used instead of the original ones, in such a way that *VRS* can be adapted to the user's requirements without being compiled again. As an example, a new mechanical structure can be easily studied just developing its specific kinematics module, although most of the industrial robots and even some redundant robots are already solved with the original kinematics component. In the same way, new robots can be easily integrated on *VRS* just developing the proper controllers to link the applications with their drivers.

3.1. Virtual Robot External Access Library (VREAL)

But the most powerful tool to adapt *VRS* to the user's requirements is the *Virtual Robot External Access Library* (*VREAL*). *VREAL* is implemented as a Dynamic Link Library in order to make possible the interaction between user's application and *VRS*.

VREAL provides an interface formed by a set of almost 50 functions (in addition to constant and type definitions) whose objective is to allow the user's application to manage elements defined on *VRS* (robots and environment) and even create new elements on *VRS*, such as objects generated from sensor data. The functions are organized in 6 independent sets:

- Functions to initialize and to close *VREAL*.
- Functions to load on *VRS* and close the environment and the robots.
- Functions to edit the location of the environment and the robots.

- Functions to manage loaded robots in *VRS*, such as moving the robot in both Cartesian and Joint spaces, asking for robot status, selecting tool and tool center point (TCP), operating the tool and much more.
- Functions to handle an auxiliary list of figures that the user's application can create and modify on *VRS*.
- A function to handle some functionality of the user interface of *VRS*, mainly to configure the view parameters (zoom, point of view, reference view point, …).

With the help of *VREAL*, the robot, environment and 3D-maps representation is transparent to the user, who only must concentrate the effort on its own application. Hence, the user leaves the usually arduous graphical part to *VRS*, which acts as a "true" virtual cell with robots and environment, reflecting the actions of user's application.

For a 3D-mapping application in robotics, the developer should manage both sensors and robots. Even more, the researcher usually would like to study and compare the performance of the methodology with different sensors and/or robots. Once again this is frequently a laborious work because big changes on the software can be required even if the techniques are sensor and robot independent. In this way, the *VRS* external components to manage robots and I/O devices are designed to become transparent the physical robots and sensors[1].

Users have access to a generic robot processor and a generic I/O device processor, which are linked respectively to the robot driver and the I/O device driver via software controllers. The controllers are just format translators, which adapt the generic interface served to user's application to the specific driver interface given by the vendor. To apply the same user's application to new physical sensors or robots, just new controllers must be developed (once again, coding templates are offered).

4. APPLICATIONS AND RESULTS

The structural design explained on previous section has been used as support for the following application example. Consider the case where a 3D probabilistic model of an Ultrasonic Sensor (US) has to be evaluated. The scanning of a part in a robotics cell is considered as the evaluation test, using a simple 3D-mapping method. The robotics cell is composed of an ABB IRBL6 5 axes robot-arm with the S2 control system with a Honeywell 940 US sensor (150-600mm range, ±1mm accuracy, 17mV/mm resolution) attached on the robot tool. The sensor is connected to a data acquisition board PCLAB 812 compatible (12-bits resolution). The configuration of the application example is shown on Figure 3. The part to be scanned by means of a horizontal swept is shown on Figure 4.

As a simple example, the possible code for an application centered mainly on the implementation of the US sensor model and the 3D-mapping is going to be commented. The application requires from *VREAL* and the external components Robot Processor and I/O Device Processor only the calls to the functions shown on Table 1.

Therefore, the user must implement only seven functions on the Robot Controller component and only four functions on the IO Device Controller component to adapt these functions to his/her specific drivers.

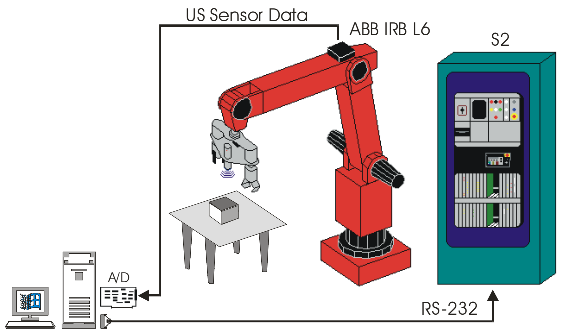

Fig. 3. *Prototype Scheme*

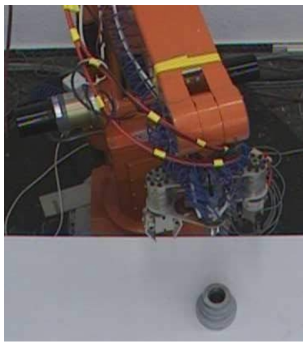

Fig. 4. *Part to be scanned*

[1] In this section, an I/O device means the data acquisition board where the sensor is connected while a robot means the robot control system connected to the computer.

Table 1. Functions required for the Application Example

Concept	VREAL	Robot Processor	I/O Processor
Initialization	`alInitialize` `alClose`	`rpOpenConnection` `rpCloseConnection`	`ioInit` `ioClose`
Configuration	`alLoadRobot` `alLoadEnvironment` `alSetTCP`	`rpEnableAxes` `rpSetUserRefFrame` `rpSetToolOffset` `rpSetSpeedScaleFactor`	`ioSetConfig`
Proper Process	`alPointToPointMove` `alAddPoint`	`rpPointToPointMove`	`ioGetAnalogueValue`

For such a simple example, the expected result on *VRS* has the form illustrated on Figure 5. Note from the code that during execution, the application is controlling, that is, moving, both robots, the real one and the virtual one on *VRS*. It is not difficult to improve the code in order to monitor the sensor value obtained for each measure.

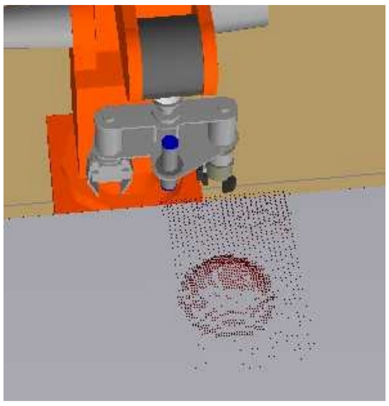

Fig. 5. *Result for Simple 3D-Mapping on VRS*

5. CONCLUSIONS

A full operational graphics application for programming, simulating and monitoring multi-robot cells has growth to be an original software for 3D-mapping validation. The application, named *Virtual Robot Simulator (VRS)*, is successfully applied on different robotics industrial and RTD projects. *VRS* has also become to be a suitable and convincing tool for research and educational purposes on a so widely studied topic as 3D-mapping on robotics, as justified in the paper by means of two practical examples. The exposition is made on base of its open design with external components that can be easily integrated with new trends procedures. Further more, to find out new application fields for *VRS*, like computer vision, is still an open point.

6. ACKNOWLEDGEMENT

This work has been partially funded by projects from JRC (13787-1998-03 F1EP ISP ES), Generalitat Valenciana (AE98-01), UPV (PII-19990576) and FEDER-CICYT (Ref. 1FD97-2158-C04-0x TAP).

7. REFERENCES

[1] R. Gutierrez-Osuna, Jason A. Janet, Ren C. Luo. "Modeling of Ultrasonic Range Sensors for Localization of Autonomous Mobile Robots". IEEE Transactions on Industrial Electronics, Vol.45, No. 4, pp 654-662 (1998)

[2] F. Blanes, G. Benet, P. Perez, J. Simo. "Map Building in an Autonomous Robot Using Infrared Sensors". IFAC International Symposium on Intelligent in Components and Instrumentation for Control Applications, Buenos Aires, Argentina (2000)

[3] A. Kutz. "Constructing Maps for Mobile Robot Navigation based on Ultrasonic Range Data". IEEE Transactions on Systems, Man, and Cybernetics. Vol .26, No. 2 (1996)

[4] E. Kruse, R. Gutsche el al. Effective, "Iterative, Sensor Based 3-D Map Building Using Rating Functions in Configuration Space". IEEE ICRA, pp. 1067 - 1072 (1996)

[5] P. Renton, M. Greenspan et al, "Plan-N-Scan: A Robotic System for Collision Free Autonomous Exploration and Workspace Mapping". Journal of Intelligent and Robotic System, No. 24 (1999)

[6] Y. Yu and K. Gupta. "On Sensor-based Roadmap: A Framework for Motion Planning for a Manipulator Arm in Unknown Environments". IEEE/RSJ International Conference on Intelligent Robot and System, pp. 1919 - 1924 (1998)

[7] Y. Yu & K. Gupta. "Sensor-Based Motion Planning for Manipulator Arms: An Eye-in-Hand System". IEEE ICRA San Francisco 2000

[8] E. Ruiz. "GENERIS:The EC-JRC Generalised Software Control System for Industrial Robots". International Journal of Industrial Robot, Vol. 26, No. 1, 1999