Standardization and integration in robotics: case of Virtual Reality tools

Nicolas Mollet
IIT - Italian Institute of Technology
Robotics, Brain and Cognitive sciences Dpt.
Intaro Team
Via Morego, 30 - 16012 Genova, Italy
nicolas.mollet@gmail.com

Luca Giulio Brayda
IIT - Italian Institute of Technology
Robotics, Brain and Cognitive sciences Dpt.
Intaro Team
Via Morego, 30 - 16012 Genova, Italy
luca.brayda@gmail.com

Baizid Khelifa
IIT - Italian Institute of Technology
Robotics, Brain and Cognitive sciences Dpt.
Intaro Team
Via Morego, 30 - 16012 Genova, Italy
baizid.khelifa@iit.it

Ryad Chellali
IIT - Italian Institute of Technology
Robotics, Brain and Cognitive sciences Dpt.
Intaro Team
Via Morego, 30 - 16012 Genova, Italy
ryad.chellali@iit.it

Abstract

Robotics needs realistic simulators and environments to display not only physical interactions between robots and the environment, but also more and more complex processes. Sensing procedures, communications and other processes like cognitive reasoning may be first run on adapted and competitive simulation platforms to deal with optimal design. Virtual Reality (VR) left the ghetto of visual rendering tool to become a more complete mean to handle basic simulation problems and to enable aggregation of complex functions for complex interactions. VR and robotics have met through Teleoperation. Historically, the first Teleoperation systems were used as interfaces to control remotely robots by providing input tools and sensory feedback information. With technological improvements, VR shifted to be the main kernel for first, studies on advanced robotics systems studies like bio-inspired robots behavior, multi-robot systems, etc., and second, to provide advanced multi-modal interfaces for Real Time interactions with a useful level of abstraction. We present in this paper some basic Robotics problems and the way VR can solve them.

The first part of this paper discuss about the benefits of standardization and VR for complex robots systems. Then, we present some existing solutions and we analyze their drawbacks and advantages. Finally, we introduce an application where we take advantage of those features, in terms of efficiency in the development cycle and of reusability.

1. Issues about standardization and benefits from VR tools in Robotics

Nowadays, Robotics is going through a fundamental period: *real* robots start to be largely spread for general public. We are not concerned with classical robot toys we have been finding for a long time (Figure 1): *puppet* robots which are simply remotely controlled, with limited movements and neither memory nor learning capabilities. We are concerned with robots having their own part of autonomy, mobility and sometimes fantastic articulated skeletons. Those robots can *evolve* with time, can have communication skills and even implement some learning base. Several factors and indicators can be identified to confirm this affirmation. The state-of-the-art technology is advanced enough to create complex polyarticulated robots, particularly Humanoids (Aldebaran, RoboErectus, Robonova, Bioloid, etc.) not being used in Research laboratories only. Furthermore, embedded chipsets are now able to achieve high performance in terms of computation and memory, which commonly allows to perform image/video analysis or even to support voice or face recognition. More important for the large diffusion of robots, technology allows now manufacturers to produce low-cost robots, or at least at reasonable prices for industrialization and access for general public. Sony's Aibot©was a precursor in this area, and nowadays many robot-toys like Pleo©from Ugobe, Robosapien©from WowWee and Robonova©from Hitec can be found. Furthermore, from a psychological point of view, people are now ready to *adopt* robots in their daily life. For example, Japanese people are

already more likely to adopt robots like androids, but behind this cultural heritage there's an evolution in the world towards a global acceptance. This is not always the case of course, even in Japan, as it was recently experienced by the iFBot©from Business Design Laboratory. This robot was designed to offer a kind of company (the robot provides important communication skills with an efficient face recognition) to lonely people, but finally it was not a success because of acceptance issues. However, nowadays general purpose and micro-computers are embedded in more and more devices, while children make use of them very early and their learning curve is impressively steep. The *living* computer is only the next step, and the robot toys are its representation. By mainly targeting the youngest people, the acceptance phase is easier, and it is going to simply become natural with time.

Figure 1. Two famous robots illustrating the evolution towards *real robots*: the Omnibot 2000©by Tomy, 1985 and the Aibo©from Sony, 1999.

Nevertheless the large diffusion of robots highlights some fundamental interaction issues. Of course, those general public robots are already different from the remotely controlled robots, as they provide some parts of autonomy and so on. But the interactions are still very limited and still suffer from precision problems (identification, voice recognition, etc.). So, many researches are focused on Cognitive aspects, in order to create a collaboration with robots for the interaction process rather than a *direct control*. The idea is that a collaboration with the robots would allow, from the robot's point of view, an interpretation of the request from the human, so it would improve the interaction efficiency and decrease the errors of control. This is equivalent to add *semantic* information to *syntactic* information, as it is being currently done by coupling Natural Language Understanding with Speech Recognition software [11]. This field of research is very active currently. Furthermore, we can underline that many of those robots provide some tools for write *code* and command robot's behavior or even movements. However, every manufacturer provides its own tools for use and development: SDK, interface, language or 3D environment are always dedicated to a particular robot or at least to robots within the same manufacturer, such as the *Robotic Invention System* for *Lego Mindstorm* [4] for example. If we take the case of researchers, or even of the general public, such an approach does not allow a native compatibility between heterogeneous robots from different manufacturers. Of course, economical reasons of manufacturers justify this attitude, which, on the other side, prevents a large integration and diffusion of different robots in the home environment.

So, the standardization from the connectivity and interoperability's point of view is the first necessary step for the large diffusion. In fact, a parallel comparison with the history of computers can be done: in the late 70's and 80's, standards have appeared, in particular with the Personal Computer (the *PC*) from IBM in 1981. At this time, and later, many hardware architectures and many dedicated Operating Systems (OS) were designed and implemented. But the *PC* became a standard for many manufacturers: from the standpoint of OS, clearly the role of Microsoft was fundamental with Ms Dos, OS/2, and Windows series for the general public access. Below the OS, the most important novelty was the ability to integrate various components easily, and to create a *platform* where all of those heterogeneous components could communicate and work together: this feature is exploited today, because in order to build a PC several components from several manufacturers are used. Thus both aspect, hardware and software, have achieve a standardization process, and have conducted to the democratization of computers and to their large diffusion. Nowadays, robots are envisaging the same scenario of PCs, and future standards will allow a kind of *plug 'n play* robots in a *full compatible environment* (for communication, interaction between and with the robots, etc.). By now, however, there's only a natural competition between potential standards, as several solutions are available. To this regard, VR can play a big role in the standardization process. We will show that VR is the best candidate for achieving efficient, low cognitive load interaction with robots.

Virtual Reality find strong roots in Teleoperation, so in the discipline which aims to provide a control on (distant) robots. Generally speaking, VR is a particular way of visualazing: visualization is fundamental for humans, as it offers the possibility to represent (thus to understand) any complex situation. While handling a robot, through VR it is possible to see what it sees, to be empathic or even to *be* the robot. Furthermore, VR can offer a very important abstraction layer for coding or interacting with robots. For these reasons, it should not be a hazard for VR to play a funda-

mental role in future standards. More specifically, we can identify in fact several reasons to use VR as a fundamental and standard tool for Robotics:

- *Simulation*. Many operators are currently being trained with simulators. However, when handling real situation, the *mismatch* between the real and the virtual world can be large. As a proposed solution, real robots can be substituted by simulated data, exchanged with the virtual world. First, teleoperators can train on the virtual world, optionally retrieving the "fake" real data. So the same world is seen both at simulation and in real time, thus reducing the experience mismatch. Second, VR also offers the possibility for coders to test their programs on robot's behaviors with such simulations.

- *Abstraction*. The virtual world is a representation of the real world, so virtual robots can offer an abstraction of low-level configurations or commands. For example, we can abstract several complex and dedicated robots by a unique robot with abstracted commands, which are then assigned to all the robots. So virtual robots can also complete semantically simple functions (i.e. "go there"), which are present at this virtual level and which are then translated into low-level, complicated tasks (motion, communication). The abstraction level is fundamental in Teleoperation and in representation in general, as it decreases the cognitive load.

- *Discrimination and Augmentability*. Useless information from the real world can be discarded, useful information can be emphasized, and possibly the most interesting parts of reality can be augmented (for example a virtual target can be added for both robots and teleoperators, via VR or AR) or shown as they are in reality.

- *Scalability*. If multi-robot agents are considered, VR is preferable because of the huge amount of information sent by the agents to all the operators. Supposed that agents are connected through a peer-to-peer or a centralized network, then agents continuously send data, possibly with videos. While data can be stored in a server for later use and visualization, operators need to see lower and lower resolution worlds as the number of agents increase. VR allows that. This is a consequence of the *discrimination* point previously introduced.

- *Transparency between real and virtual*. VR can be used to add objects, as previously introduced. Such addition can also take advantages from the transparency between virtual and real: as the virtual world represent the *reality*, the robot's and man's representation of the real world, any addition of pure virtual objects have consequences in the reality. For example, the addition of visible, but still virtual, limits for partially unknown area which nedds to be explored. Nonetheless, the use of virtual sensor,such as a virtual telemeter, can give virtual abilities to a real robot, as far as the real world is known and modeled in the virtual world. This can be useful if one wants to artificially add data to the environment and study data redundancy or sufficiency to solve a certain problem.

- *Prototyping*. By combining the abstraction layer and the link between virtual and real robots, coders can obtain a very interesting system to obtain real-time abstracted information on the real robots. The idea is that with such a knowledge, and if of course the VR environment is not limited to compiled programs, the coders can adapt and modify their code in real-time.

- *Navigation in space and time*. Let us consider the case where several teleoperators have to control groups of robots to explore an unknown area. Once part of the environment is known, it is possible for the operators to stop the robots (or to let them keep on doing their tasks) and to navigate into the past-explored rooms, to analyze the environment from different points of view and at different paste time lags. VR can also show the possible future trajectories of moving robots, anticipating future results, showing , as an example, where a robot will stop working because of low batteries for example.

The next part of the paper presents some VR solutions which aims to provide such a standardization. The goal is to highlight current solutions being potentially combined to offer first an easiest integration of new robots in an existing system with standards of communication and interactions, and second an efficient and reusable development of their behaviors and controls.

2. State of the art and analyze of existing solutions

We first present and analyze the state of the art in standardization and in particular standardization with VR, then we make a particular analysis of two solutions we that consider very pertinent.

2.1 Existing solutions

Efficiently applying VR for managing just a single robot, but using multiple ergonomic devices, has been very recently addressed in [5]. In this work VR and AR are helpful because their ability of simplifying the granularity of

information the human operator is needing, while performing a certain task. To achieve this, the operator can make use of several devices (keyboard, mouse, a force-feedback device), each one having a defined priority on the others, optionally available. This implies that the operator is directly concentrating on a single robot: as such, this strategy cannot be applied to multiple robots. Concerning the software framework, in this work the virtual robot can be remotely commanded via the ARITI platform [15], which is a fundamental feature to be addressed for Telepresence, possibly extensible to Telecollaboration in the case of multiple users. While this is interesting, we emphasize that there is the need to extend such works with more than one robot: the effort to address a unified framework for handling multi-robot agents is far from reaching the winning post. To solve this issue, we observe that one could proceed top-down: first, by finding a suitable software platform, possibly addressing a sufficient abstraction for (multi-) robot control (including VR) at the minor cost; second, by using (which implies buying) the available robots on the market supporting, at the lowest hardware-software level, such platform. This solution is analyzed in [3], where open source platforms are considered as the way to handle both in the industrial and research world. For industrial applications, the case of the ARIA platform from ActiveMedia is typical, because an open source, free of charge, cross-operating system platform is available for developers. While this potentially leads to many solutions, *de facto* the restriction comes from the fact that the same company is building robots, thus only ARIA-supported robots can be used. As stated in Section 1, this is normal for industrial-driven software and hardware production, but would introduce limitation in research, where the range of supported hardware should be as large as possible. In this sense a bottom-up approach is, for our purposes, more suitable. First, only software platforms possibly addressing many and heterogeneous sets of robots should be chosen: in fact, using VR to add an abstraction layer is as much useful as different kind of robots are available. Second, a platform allowing to use VR is necessary. This requirement sensibly narrows down the available choices. Third and last, the global cost of hardware and software should be considered, in terms of initial software licences (which should ideally drive the choice towards free of charge softwares), software development and maintenance (open source software would be the best in this case), cross-platform and cross-language development, and finally hardware cost. Several robot platforms exist, which try to be as much versatile as possible: MIRO [17] and RT-Middleware [1] represent an effort to integrate hardware from different manufacturers and to address distributed control; the OROCOS [6] platform also includes libraries to perform dynamics and kinematics computation and Bayesian filtering for position estimation. However, these platforms still lack of the VR environment, which we think is necessary for an ideal abstraction layer: projects such as the Player/Stage [8] and Gazebo [9] include a simulation environment. Logically, the use of Virtual Reality well suits with higher level functionalities: a virtual environment can be used to efficiently symbolize the behavior of a robot with few but very useful data; the same works vice versa: through VR, commands can be given to the robots if behavioral routines are present on the platform. For this reason it would be desirable to access, together with simulation, to a behavior coordination layer. To the best of our knowledge, only two platforms provide such features: Microsoft Robotics Development Studio (MRDS) and URBI. We describe them more in depth in 2.2. For the sake of completeness Other available platformsare CLARAty [14], similar to URBI because software modules for sensors and actuators are not modular - i.e. it is not exchangeable between processors and other micro-controllers, ORIN[13] and ORCA [10], which are modular like MRDS, but are open source and still less widespread on the market, and finally OPENR [7], which supports full modularity but suffers from being available on a limited number of devices.

2.2 Focus on URBI, Webots and MRDS

As previously discussed, we have identified MRDS and URBI as the only two platforms providing all the relevant features for standardization, abstraction, visualization and simulation. We support our findings in the following. By construction, URBI doesn't provide a 3D environment, this being not its main purpose, so we focus on an analysis on MRDS and URBI combined with Webots (such a combination is used for example in [16]). As a guideline such analysis of those systems, we compare the potential features of

- Simulation tools and robot creation. Simulation tools are very important to validate tasks before its usage on the real robot: debugging of course, correcting algorithm (for trajectory or collisions for example), etc. Robot creation deals with the ability of the platform to propose or to integrate and design new robots.
- Flexibility within robot application and Operating System. It means that the system should be able to work with any operating system and suitable for any type of robotic application.
- Languages and simplicity. Related to this property the system must be developed with different programming languages, that give different access level, from the hard-coder to people without any particular advanced programming skills.
- Task synchronization and hardware configuration. Refers first to the ability of synchronization between

different assigned tasks to the robot and also to synchronize commands between robots. On another hand, it's the capability to add services for new hardwares to extand the robot's capabilities.

From the viewpoint of simulation tools and robot creation, the MRDS has a very high level of physics simulation and realistic native 3D representation. On the other hand, Webots covers a large area of robot creation and has also a good physics simulation environment, relying on ODE (Open Dynamics Engine). Concerning flexibility with respect to the Operating Systems, URBI and Webots are available under several OS: Windows, Linux and Mac OX. On another hand, MRDS is natively limited to Windows. But the use of Mono[1] (Open source compatibility project for Microsoft's .Net) allows its usage under Linux, MacOSX and Unix as well.

Another important and useful feature is the development languages and the simplicity for a particular system. Both solutions can be used with a large variety of useful languages. URBI can be used with C++, Java, and python, while Webots can be used with C, C++ and Java [16] [12]. MRDS can be used with C-Sharp and in fact with all the .NET packages (Python, etc.). From a usage point of view, URBI offers a scripting URBI language which is very useful (as it's clearly dedicated to robotics), while MRDS is more focused on the Visual Programming Language (VPL) and set of tools for code generation (C-Sharp).

Finally, an important feature is the ability of task synchronization and simplicity of hardware configuration. The MRDS has a large range of predefined services to make very easily the configuration of a lot of hardware devices, such as cameras, sensors, motors and so on. Those services offers a very important abstraction layer, guaranteeing reusability. MRDS offers an efficient mechanism to synchronize tasks such as Set - such as commands to several robots - or Get - such as sensors information values. The whole system supports Concurrency and Coordination Runtime (CCR) properties. Furthermore, MRDS natively permits the real-time synchronization between virtual and real robots. Finally, MRDS integrate already several robots, and several components. Manufacturers are often working on an integration of their new robots in this solution, and that's a very important advantage. URBI also propose efficient solutions of synchronization between robot commands [2]. From the hardware's point of view, URBI is also largely integrable and integrated on hardware. It also provides many interesting human-machine interaction components (equivalent to some MRDS services), like voice recognition, voice synthesis or face detection as an example.

As a conclusion, for a software engineer, using URBI+Webots should be functionally equivalent to using MRDS (not considering other issues such as money or updates availability). For a non-programming user, MRDS is more accessible as it's a whole solution which works *out of the box*. Finally, we have need to consider the current effort from manufacturers to propose native integration of their robots in MRDS, and the large presence of Microsoft on computers. From this point of view, MRDS is already an important candidate in this actual standardization process, in particular for general public.

[1] http://www.mono-project.com

3 Example of application

We are currently working on a project which use multi and heterogeneous robots. We have decided to use the MRDS solution for this project, but this choice is not putting out the use of URBI (combined or not) with Webots. In fact, this project will certainly combine the two solutions in a near future, according to the needs, and in order to propose relevant developments and abstractions being compatible with those two actual major solutions.

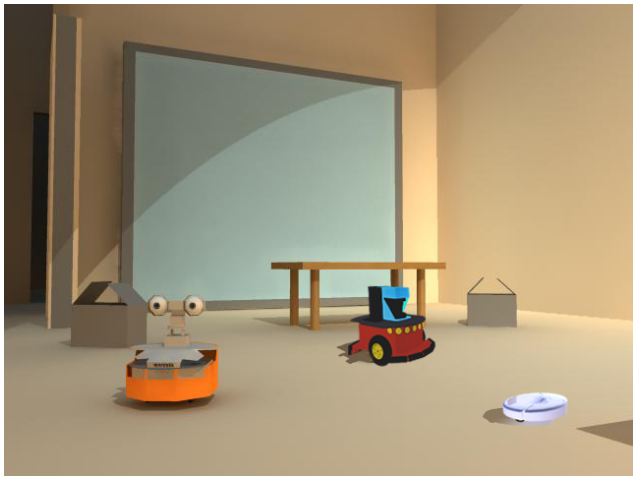

Figure 2. Several heterogeneous robots explore an area. All are integrated in MRDS.

According to this paper, our long discussion on standardization and VR tools, this project illustrates several of the advantages we identified. For example, we integrated heterogeneous systems on the same development platform. The figure 2 shows this actual integration of some heterogeneous robots, from 3 different manufacturers (with different interaction abilities, different sensors, and all of them with dedicated low-level interfaces), in the Collaborative Virtual Environment of MRDS. Clearly, the idea is that we plan to plug any new robot easily on our development platform, by using its direct integration (equivalent to a *driver* for a device we plug to a computer, and also equivalent to the

traditional *plugin* approach for software developments). By entering the platform, this new robot is automatically able to communicate with others (thanks to the MDRS' standardized communication protocols and tools for example), and of course the existing interaction tools will allow its control without any code development. It means that teleoperators can control it directly with their usual interaction modalities. Moreover, the algorithms for direct controls, for action metaphors, or even for robots' behaviors, can exploit those standardized interaction abilities, those standardized *interfaces*, in order to integrate this new robot in a more global process, like it can be with algorithms for multi-robots projects. We can consider for example algorithms which organize robots to reach a target, by using abstracted displacement commands. Those one are here the same for any robots. Then, our platform manage the transformation of those high level commands according to the particular robot. On another hand, the robots abilities are abstracted towards generic commands. We offer to the teleoperators the ability to act on the group of robots, as a virtual global entity, by providing a high level of abstraction. With this standardized approach, any new robot can automatically integrate this group management. Finally, we can also underline that, of course, there's a strict Real-Time surjection from the Virtual to the Real world: the first can be richer, while the robots in the second provide a continuous feedback and do the actions required by their virtual avatar.

4 Conclusion

To conclude, we have first discussed about the interest of standardization in Robotics. Clearly, this step will be fundamental in the general public access to Robotics, and it seems that the *game* is getting close to the solution. We presented the existing alternatives, and we insisted on the VR aspects as we identified this tool as fundamental for interaction, representation and abstraction of robots. Finally, we illustrate with a very small example our current usage of those new standards. It clearly shows that such standards are already working.

References

[1] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon. Rt-middleware: distributed component middleware for rt (robot technology). *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3933–3938, Aug. 2005.

[2] J.-C. BAILLIE. Urbi: Towards a universal robotic 3ody interface. *Humanoid Robots, 2004 4th IEEE/RAS International Conference on Volume 1, Issue , 10-12 Nov. 2004 Page(s): 33 - 51 Vol. 1*, 1(1):33–51, Nov. 2004.

[3] P. Barrera, G. Robles, n. José M. Ca F. Martín, and V. Matellán. Impact of libre software tools and methods in the robotics field. *SIGSOFT Softw. Eng. Notes*, 30(4):1–6, 2005.

[4] D. Baum. *Dave Baum's Definitive Guide to Lego Mindstorms*. APress L. P., 1999. Illustrator-Rodd Zurcher.

[5] P. Boudoin, C. Domingues, S. Otmane, N. Ouramdane, and M. Mallem. Towards multimodal human-robot interaction in large scale virtual environment. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 359–366, New York, NY, USA, 2008. ACM.

[6] H. Bruyninckx. Open robot control software: the orocos project. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 3:2523–2528 vol.3, 2001.

[7] M. Fujita and K. Kageyama. An open architecture for robot entertainment. In *International Conference on Autonomous Agents*. ACM Press, 1997.

[8] B. Gerkey, R. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30 - July 3, 2003*, pages 317–323, 2003.

[9] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 3:2149–2154 vol.3, Sept.-2 Oct. 2004.

[10] A. Makarenko, A. Brooks, and T. Kaupp. Orca: Components for robotics. In *International Conference on Intelligent Robots and edheterogeneous environments. Communications Magazine, IEEE, Systems (IROS)*, page 163168, 2006.

[11] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[12] O. Michel. Cyberbotics ltd.webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems, Volume 1 Number 1 (2004), ISSN 1729-8806*, 1(1):39–42, 2004.

[13] M. Mizukawa, H. Matsuka, T. Koyama, and A. Matsumoto. Orin: Open robot interface for the network, a proposed standard. In *SICE*, 2000.

[14] I. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin. Claraty and challenges of developing interoperable robotic software. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 3:2428–2435 vol.3, Oct. 2003.

[15] S. Otmane, M. Mallem, A. Kheddar, and F. Chavand. *Tltravail Robotis et Ralit Augmente: Application la Tloperation via Internet*. PhD thesis, Universit d'Evry-Val D'Essonne, 2000.

[16] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. In *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 11, NO. 2, APRIL 2007*, pages 265–286, 2007.

[17] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar. Miro - middleware for mobile robot applications. *Robotics and Automation, IEEE Transactions on*, 18(4):493–497, Aug 2002.