ASSET: A Testbed for Teleoperation Systems

NANCY RODRIGUEZ, JEAN-PIERRE JESSEL
IRIT
University Paul Sabatier,
118 route de Narbonne, 31062 Toulouse
FRANCE
rodri@irit.fr,jessel@irit.fr http://www.irit.fr

Abstract: - Development of a teleoperation system means development of several subsystems, such as control devices modules, communications and graphic user interface. Each one of these components is studied, conceived, and built generally in an independent way with particular software and methods. The complexity of the system can involve a very expensive development and integration of components to complete the system. In this case an environment of evaluation very easily configurable and domain-adaptable is a useful tool. In this article, we propose ASSET, a reusable framework for the development of systems of teleoperation. Our system allows to test simulation models, 3D scenes, behaviors of autonomous entities and new devices. This system was used with modules of behavioral simulation developed in the laboratory to study the assistance of autonomous robots within the framework of a teleoperation mission.

Key-Words: - teleoperation testbed, robotics, simulation, interaction devices, framework

1 Introduction

A teleoperation system makes possible to an operator to execute a task, without being at the same place than the machines are. Since the operator is not physically on the site of task execution, the teleoperation eliminates the risks associated with jobs such space exploration, toxic substances handling, etc. At the same time, in order to the operator to work efficiently, it is necessary to give him all the information that he will need to carry out his task, considering also that the communication between the user and the system of teleoperation must be fluid, with information easy to interpret. To achieve these requirements, a teleoperation system is a complex system, formed by divers subsystems such as interaction devices, acting devices, control modules, event handling, communications, graphics user interfaces and computer facilities to support computations. Each of these subsystems is a subject of research, design, development and testing, and for this reason, teleoperation system construction requires considerable effort and resources, the same if there exist some tools to help in the development. In fact, because these tools are domain specific, not compatible between them, each researcher works in a different environment that affects the integration process later. On the other side, if adequate tools don't exist, the researcher must build them and he can't only focus on his particular problem of study because he must solve other problems, for example, a researcher who builds a device driver does not need to spent his time on building a graphical user interface.

Our proposed solution to reduce some of this difficulties, is to develop a general testbed for teleoperation systems, easily adaptable to different contexts. This system, called ASSET (Architecture for Systems of Simulation and Training in Teleoperation), is a modular system that abstracts mechanisms presented in all teleoperation applications such as network communications, graphics user interface, communications between user and simulation, etc. These mechanisms are provided as a set of reusable components, easily configurable, at which it is possible to add or change modules in a dynamic way. This allow to use the system as a testbed for evaluation of 3d scenes, IA behaviors, new devices or interactions techniques. These features enables us to say that ASSET may be useful to a wide range of research.

2 Background

For the most part, teleoperation systems are still in the domain of research products, developed to solve a specific problem of teleoperation domains such as devices control, user interfaces and simulation. It also exists an interest in the development of more flexible systems, to meet the specific needs of new projects. Coupled Layer Architecture for Robotic Autonomy (CLARAty) [1], joined the efforts of NASA's precedents works like the Telerobot Testbed Demonstration System of Jet Propulsion Laboratory [2], a system conceived to be used to develop, implement, and evaluate the performance of advanced concepts in autonomous, tele-autonomous, and teleoperated control of robotic manipulators. CLARAty's goal is to develop and implement a comprehensive control architecture for multiple, disparate, interacting, planetary rovers. The

control of these systems will use the architecture to implement artificial intelligence techniques for autonomous sequence planning, error handling, and recovery during surface operations in an unknown terrain. A very important and explicit objective of this system is to have the resultant architecture exported to other NASA rover systems, providing a common software environment. Up to now, the prototyping and implementation of the CLARAty architecture is still in its early stages. Obviously, in order to address the needs of space telerebotics research, very expensive, sophisticated and high performance computing systems are required. For this reason, several projects to build economical and acceptable platforms for teleoperation research have been developed. For example, as part of the research in teleoperator control interface design, a WindowsNT/PC-based teleoperation system prototype was developed in the Mississippi State University[3]. This system links a virtual environment manipulator interface to a PUMA robot, to study human factor issues in the development of teleoperator interfaces. This experience shows that near real-time teleoperator control can be achieved using low-cost common PC hardware. With the same interest, Ghiasi have proposed a reusable framework designed to enable the manipulation of devices via the World Wide Web for web-base teleoperation that made use of open source products and simple APIS as Java and Python[4]. This framework attempts to reduce the level of skill required to successfully develop a teleoperation device providing mechanisms to interact with, and providing tools that allow to build different GUIs and to make extensions or modifications to existing functionality. The system flexibility allow its use as a higher level tool, as in the work described by Balcisoy [5]. This work presents a framework for testing the design of objects in an augmented reality context. The definition of modeling object geometry is extended with modeling object behavior to allow users to experiment with a large set of possibilities without having extensive knowledge on the underlying simulation system. This approach shows that users can decrease the time spent on prototype evaluation and have a realistic testing environment.

Another important axis in teleoperation research is simulation, because it allows researchers, designers and users to construct robots and task environments in a quick and inexpensive way, compared to real systems cost, and it allows the study of geometries, kinematics, dynamics and motion planning. In his dissertation Anderson [6] described and compared in a detailed manner most utilized robotics simulators: ARS MAGNA is an abstract robot simulator that provides an abstract world in which a planner controls a mobile robot. Experiments may be controlled by varying global world parameters, such as perceptual noise, as well as building specific environments in order to exercise particular planning features. However it proves that it is inadequate when an attempt is made to adapt them to a new domain or to a new type of agent. The Michigan Intelligent Coordination Experiment (MICE) testbed is a tool for experimenting with coordination between intelligent systems under a variety of conditions. MICE simulates a two-dimensional grid-world in which agents may move, communicate, and affect their environment. MICE is essentially a discrete-event simulator that helps to control the domain and a graphical representation, but it provides relatively few constraints on the form of the domain and on the agent's abilities. RSIM is a SGI-based graphical robot simulator from the University of Melbourne, it makes discrete time simulation of an arbitrary linked robot arm, with full kinematics and dynamics. There is a discrete-time controller and a standard C interface so that users can create and test different controlling algorithms. This robot simulator currently works only on SGI machines.

Research in virtual environments (VE) has included research in VE construction toolkits, VE software architectures and VE training projects as NPSNET [7] which is a multi-user distributed virtual environment used to recreate complex military missions. The Distributed Interactive Virtual Environment (DIVE) is a virtual reality system allowing many users to explore the 3D space and interacting with each other [8]. VIPER, a system developed in our team [9] is also a generic, multi-user distributed virtual reality platform that is able to run on heterogeneous physical architectures. Minimal Reality Toolkit (MRToolkit) is a set of software tools for the production of virtual reality systems and other forms of three dimensional user interfaces, it includes device drivers, support programs and a language for describing geometry and behavior [10]. Bamboo is a component framework enabling the development of virtual environments, that dynamically loads language-specific plugins into and out of memory[11]. Each plugin is part of a "module," which is a directory structure that can also contain any data files used by itself (i.e. geometry, texture, sound). Typically, modules are archived and subsequently downloaded at runtime via HTTP from one or more developer-specified web servers over the Internet.

In our system, we are interested in providing a tool for helping development of teleoperation systems. This system unifies the advantages of the different system revised: a modular design to make extensions and modifications easily, the ability to separate process from the domain, virtual environments as a graphic user interface, simulation, built with commonly available tools. Our system is highly configurable and allows reusing the components already built and facilitating the integration because the interface functional of each

module remain constant. An additional objective in the ASSET implementation is to have a not expensive system, light weight, object-oriented and developed with open source products. For this reason, we have choosen Java and Java3d to develop our system, so that it can run on any platform without further changes. Java3d additionally offers a high-level API for designing 3D systems.

3 ASSET System

ASSET (Architecture for Systems of Simulation and Training in Teleoperation) is a set of reusable Java components which offers to the programmers the services related to teleoperation missions, providing a testbed for experimenting with behaviors, simulation models and devices. Asset is designed to facilitate the research and development of teleoperation applications by providing modules that allow dynamic integration, simulation for facilitates experimentation and distribution issues as dead-reckoning techniques for minimizing the sending data step.

3.1 Architecture

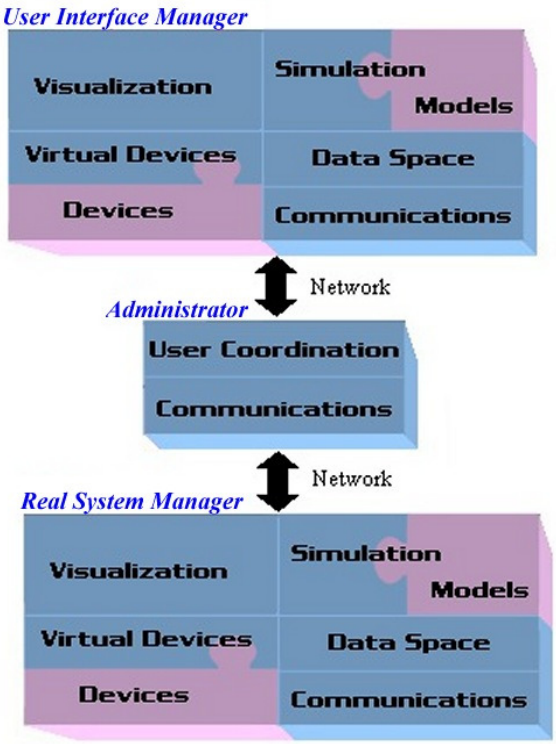

Fig.1

As shown in Fig.1, the ASSET's architecture consists on three modules:

User Interface Manager: User Interface Manager (UIManager) is responsible for the communication between the system and the user, it handles the local simulation and the interaction devices. Communication between this module and the others modules of the system is managed by the Communications and Events component.

Real System Manager: It is the module of control of the real system. It is very similar to the UIManager, but it controls the sensors and effectors. Real System Manager (RSManager) executes the commands and manages coherence between the real state and the simulated state to update the user's simulations.

Administrator: The administrator coordinates the interactions between the participating entities, users and robots. It has a Communications and Events component to transmit the commands to effectors and the information from the real system to the users, and a Coordination component to solve conflicts raised by different orders.

3.2 Mechanisms

One key feature in ASSET is the management abstract of the application's basic components (i.e. devices, communications, state). In that direction in addition to the architecture, ASSET defines the mechanisms which allow the interaction between the different modules and the different components of each module:

3.2.1 Data space and Event Handling

To set the communication between the various components of a module, we have defined a data space which maintains device information, commands, and network messages. The data space notifies the occurrence of a written event (when a component adds a message) allowing devices, simulation objects or communication units to recover and to process them. This capability allows to have a domain and devices independence.

3.2.2 Simulation

Simulation facilitate experimentation because it allows to evaluate a system in inaccessible environments or to face rare events. Indeed, by replicating simulation and models in every host participant, we have rapid feedback and filtration of invalid commands. In ASSET, there is a simulation in each UIManager as in the RSManager. The simulator in the UIManager makes possible to give feedback to the user without delay while simulator in the RSManager avoids the transmission of data at the end of each interval of simulation. The RSManager compares after simulation update if the real system state is

different from the simulation state, and in this case, it sends the accurate to update the simulation of the UIManager. The simulation component in ASSET offers the services of 3D visualization, collision detection and reading of Java3D and VRML2.0 models. One very interesting feature in ASSET simulation is the possibility to define the behavior for each simulation object. This allows having entities with different degrees of autonomy in the simulation.

3.2.3 State

To know if the real state and the simulated state are different, ASSET uses the conditions defined by the user for his application. The user defines the set of variables that constitute the state of the system and, for each variable, he defines the maximum error value. If there are one or more variables that have reached their maximum error value, simulation must be updated. Because the state of the system and the acceptable difference between two states are defined by means of a configuration file, it is possible to easily calibrate the system. On the other side, the type of the variables and the concept of distance can be modified by the user because the system instantiates dynamically the classes developed by the user to manage his variables.

3.2.4 Devices management

In ASSET we have defined virtual devices which offer a set of common services that can be implemented with various physical devices. The virtual device is a mediator between the real device and the system ensuring the independence between the application and the specific devices.

With the architecture and mechanisms defined in the ASSET system, we have achieved important requirements like extensibility, adaptability and a highly configurable system that can integrate available resources. For example, for testing behaviors, the developer only needs to provide a geometric model and a control class for the entity. The same, for testing a new device, the developer only needs to implement the basic services defined in the virtual device. This feature allows reducing the time of development and letting the developer to focus in his work to optimize it. To present our idea we worked on EVIPRO, a system for facilitating cooperative teleoperation research.

4 A Demonstration Scenario

To help the user to accomplish teleoperated tasks, an autonomous or teleautonomous robot can participate. Assistance robots complete human faculties and allow the system to take advantage of the computer's capacities to realize repetitive tasks and also physically hard work, and to use as better as possible the expert dexterity to look and to react at the right time [12]. In project EVIPRO, we are studying man-machine cooperation to carry out teleoperated missions in a system using virtual reality and adaptative tools where drawbacks are compensated with an autonomous robotic system [13]. The goal for the human users and the autonomous robots is to achieve a common task in the virtual environment. The system EVIPRO consists on a reactive system (ASSET) that understands the events of the dynamic environment, and a system of behavioral simulation on the basis of autonomous robots called A^3 whose mission is to help the user. The scenario it is the following:

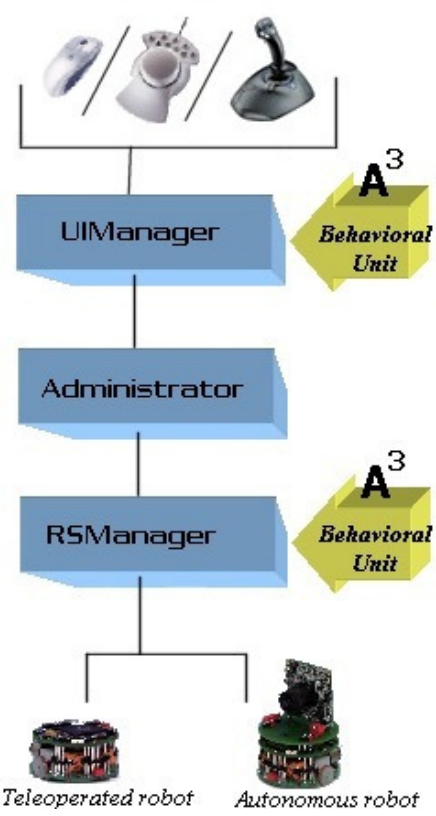

Fig.2

Fig.2 shows the EVIPRO system with an user who cooperates with an autonomous robot. The user addresses commands to khepera robot 1 by means of an interaction device (spacemouse, joystick) and the user interface (UIManager). These commands are first sent to the local simulator to produce user feedback. If the result of that action triggers the simulation to another valid state, the commands are sent to the control module of the real system by the Administrator's way. The RSManager updates the simulation allowing behavioral unit to know the state of each simulation object, and to react as well as possible to the new environment. This unit filters information to obtain the needed data for making a

choice of behavior and update his state. Finally, the two khepera robots execute their commands. It is important to note that the autonomous robot is only controlled by the behavioral unit associated with the simulator of the RSManager, allowing system A3 to only work with valid global data. The unit in UIManager just feeds visual simulation.

The construction of this system allows us to verify the architecture and the ideas stated in ASSET. Because with Reflection package of Java, it is possible to dynamically load classes to our system, it allows to integrate new classes and models without modifying the system. Nevertheless, some difficulties have arised, being the most important collision detection. In fact, in Java3D it is possible to know when collision is produced but it is very difficult to determine collision information. At the moment we are using a simple algorithm of collision detection based on surrounding spheres while we work in the development of a new collision detector.

5 Conclusion

We have developed ASSET, a support environment that serves as a testbed for teleoperation systems development. In this experimental environment, the developer can test new simulation models, behaviors and devices. This feature facilitates the utilization of ASSET in the creation of new systems and in making rapid tests and prototypes. Furthermore, to achieve platform independence, the implementation of ASSET is based on Java and Java3D. We have used our system in the building of a sample application, and we have been able to demonstrate the benefits of its flexibility, in particular in the dynamic integration of behaviors in the simulation and in the construction of applications independent of specific devices. Future work will cover several aspects. At the current time, the most promising directions appear to include: testing heterogeneous objects with different levels of autonomy, studying interaction devices and techniques (spacemouse, joysticks, gant, stereoscopic views), and allowing multiusers interaction for training, Finally, improvements of the collision detection engine are pursued. Another valuable feature to be integrated will be the task planning to allow the operator to use the result of his experiences in simulation to execute a task in the real system without permanent control.

References:

[1] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, The CLARAty Architecture for Robotic Autonomy *Proceedings of the 2001 IEEE Aerospace Conference*, Montana, USA, 2001

[2] JPL Telerobot Testbed-Lessons Learned, http://robotics.jpl.nasa.gov/tasks/testbed/accomplishments/lessons/testbed-lessons.html

[3] David B. Kaber, Rong Zhou, Dezhen Song, Design and Prototyping of an Economical Teleoperation Testbed for Human Factors Research: Cost, Resource Requirements and Capability Assessment, *Proceedings of the 25th International Conference on Computers & Industrial Engineering*, New Orleans, USA, 1999

[4] S. Ghiasi, M. Seidl, B. Zorn, A Generic Web-based Teleoperations Architecture: Details and Experience, *SPIE/Telemanipulator and Telepresence Technologies* VII, Boston, USA, 1999

[5] Selim Balcisoy, Marcelo Kallmann, Pascal Fua, Daniel Thalmann, A framework for rapid evaluation of prototypes with Augmented Reality, *ACM VRST 2000 - Symposium on Virtual Reality Software and Technology* , Seoul, Korea, 2000

[6] John Anderson, *Constraint-Directed Improvisation For Everyday Activities*, Department of Computer Science, University of Manitoba, Canada

[7] Michael Macedonia, Donald Brutzman, Michael Zyda, David Pratt, Paul Barham, NPSNET: A Multiplayer 3D Virtual Environment over the Internet, *Proceedings of the AM-1995 Symposium on Interactive 3D Graphics*, 1995

[8] Hagsan O., Interactive Multiuser VEs in the DIVE System, *IEEE Multimedia Magazine*, Vol. 3, No. 1, 1996

[9] Torguet P, Ballet O, Jessel JP, Gobetti E, Duchon J, Bouvier E., CAVALCADE a system for collaborative virtual prototyping, *Journal of Design and Innovation Research - Special Virtual Prototyping*, Vol. 2, No. 1, 2000

[10] MR Toolkit, http://www.cs.ualberta.ca/~graphics/MRToolkit.html

[11] K. Watsen and M. Zyda, Bamboo - A Portable System for Dynamically Extensible, Real-time, Networked, Virtual Environments, *VRAIS'98 - IEEE Virtual Reality Annual International Symposium*, Atlanta, USA 1998

[12] Hirohiko Arai, Tomohito Takubo, Yasuo Hayashibara and Kazuo Tanie, Human-Robot Cooperative Manipulation Using a Virtual Nonholonomic Constraint. *Proceedings 2000*

IEEE International Conference on Robotics and Automation. 2000

[13] Heguy,O., Rodriguez,N., Luga,H., Jessel,J.-P., Duthen Y., Virtual Environment for Cooperative Assistance in Teleoperation, *WSCG'2001 - International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, Plzen, Czech Republic, 2001

[14] Nathan Smith, Christopher Egert, Elisabeth Cuddihy, Deborah Walters. Implementing Virtual Robots in Java3D using a Subsumption Architecture, *WebNet 1999 - World Conference on the WWW and Internet*, Hawaii, USA, 1999