Trends in Software Environments for Networked Robotics

Davide Brugali[1], Moisés Alencastre-Miranda[2], Lourdes Muñoz-Gómez[2], Debora Botturi[3], and Liam Cragg[4]

[1] Universitá degli Studi di Bergamo, Italy `brugali@unibg.it`
[2] Tecnológico de Monterrey - Campus Estado de México, México {`malencastre, lmunoz`}`@itesm.mx`
[3] Department of Computer Science University of Verona, Italy `botturi@metropolis.sci.univr.it`
[4] University of Essex, Department of Computer Science, Colchester, CO4 3SQ, UK. `lmcrag@essex.ac.uk`

1 Introduction

Networked Robotics [SG02, SG03] is a kind of Human-Robot Interaction [HRI04]; the human operator collaborates with the robot for the execution of difficult tasks for which the robot cannot achieve a high level of autonomy due to the complexity of the task or the structure of the operating environment.

The peculiarity of Networked Robotics is the physical distance between the robot and the human operator that prevents the operator from seeing the robot during the execution of its tasks. Therefore, Networked Robotics strongly relies on information and communication technology for the efficient transmission of the operator's commands and of the robot sensory feedbacks.

Mainstream application fields related to teleoperation of physical devices are Space Robotics and Remote Surgery, where effective human-robot interaction builds on dedicated and reliable communication networks (e.g. ATM or satellite) and specialized hardware devices (e.g. exoskeleton master linkage). In teleoperation applications of haptics, a loop is closed between the human operator's motion "inputs" and forces applied by the haptic device via a communication link, robot manipulator, and the environment. Key challenges for the advancement of teleoperation technology include the development of high performance control paradigms, advanced mechanism and quantitative measures on the teleoperation system including haptic displays, and new software methodologies that permit a very careful and precise design of high performance architectures at a reasonable effort in terms of cost and development time.

The adoption of teleoperation technology in other application areas, such as factory automation, rescue robots (earthquake/terrorism), and nuclear decommissioning is enabled by the use of standard PC-based user interfaces as robot console and of conventional networks (e.g. Internet and wireless LANs) as communication medium between the robot and the human operator.

While PC-based and Internet-based systems provide widely accessible, and cost effective mechanisms for teleoperation technologies, they are subject to indeterminate transmission delay and data loss. Software engineering environments can aid in overcoming the drawbacks of adopting Internet as a communication mechanisms by facilitating the appropriate sharing of autonomy and exchange of information within a distributed system between the human operator and the robotic system. When Networked Robotics is employed in tasks where the complexity of the task or the structure of the operating environment prevents full robot autonomy, such systems may still benefit from the application of intelligence to increase the efficiency of user commands.

Furthermore, distributed software environments allow the construction of Networked Robotics systems that seamlessly scale up from one-to-one interaction (one human operator and one robot) to many-to-many interaction (multiple human operators and multiple robots). Users can collaborate between them in this distributed interaction in order to work together in some robotics application (e.g. teleoperation of several robots in a common task). Users can be located in different places. Commonly, robots are in the same global environment (e.g. a factory). However, several robots can be sharing the same local environment (e.g. production area, offices, outdoor area, etc), or each one can be in a different local environment.

2 Opportunities to Exploit Software Development Techniques in Networked Robotics

The chapters in Part IV of this book describe Networked Robotics systems that have been build exploiting several software development techniques, such as domain analysis, components and frameworks which have been thoroughly discussed in the previous parts of this book. In the following sections we analyze three Networked Robotics paradigms from three points of view:

1. The operator's role. Depending on the task to be executed, the operator can participate to the robot's operations by reacting to its feedbacks and sending new commands immediately, or he can elaborate the robot's strategy and behavior off-line.
2. The robot autonomy. Networked Robots can have autonomy levels that range from full autonomy (the robot is given a task which it then accomplishes safely), to shared autonomy (the operator influences or decides the robot's behaviour and strategy), and to limited autonomy (the operator controls the robot directly).

3. The robot-operator information exchange. From the communication point of view, the human-robot interaction is characterized by the amount and quality of information that has to be exchanged in order to perform the task correctly and to exploit the communication medium efficiently.

The three points of view serve as convenient schema to highlight strengths and limitations of each Networked Robotics paradigm. For each paradigm a suitable software development approach is illustrated which is described in details in a subsequent chapter.

Finally, the Sidebar *Java3D for Web-based Robot Control* by I. Belousov illustrates some examples on how to exploit the Sun Microsystems Java3D library for robot teleoperation.

2.1 Direct Control

The simplest model of interaction between the robot and the human operator is the Master-Slave control. The operator sends to the remote robot (e.g. a manipulator arm) low-level commands (e.g. increment a joint's position) and receives sensory feedbacks (e.g. images from a camera mounted on the arm). This model of interaction is also known as "move-and-wait" strategy: the operator induces a small movement and waits to observe the results of the movement before committing to further action.

The operator plays the role of remote controller in the human-robot feedback control loop. The dynamic of the task is assumed to be quite slow since teleoperation of a robot requires intense mental attention. The advantage of having a direct control is that the operator can feel the effects of his commands, since there is a direct correspondence between action and sensing.

The robot autonomy can be very limited as the robot is under the full control of the remote operator. Usually the robot is at least able to react to unexpected events (e.g. a collision) in a simple and predefined way, such as stopping the command execution and informing the remote operator. The direct control paradigm is highly versatile, since it does not require the robot to build and use a detailed world model, nor it assumes the environment to be static or structured.

The main drawback of direct control is that the communication load is high and that the operator feels the control of the robot under important communication delays very uncomfortable. This is due to two characteristics of Internet communication: *unpredictable time delays* and *low bandwidth.*

Unpredictable time delays means that data are received with variable time interval that irregularly changes. As a consequence, the control loop between the robot and the remote operator may become instable. The operator is instinctively induced to repeat or magnify a command to the robot if he does not feel its effects within a given time interval. At the other side of the communication link, the robot behavior is affected by the duration of a command or the temporal distance between two consecutive commands.

Low bandwidth means that the network traffic affects the flow of data that can be profitably exchanged between the robot and the operator. TV images are substantially delayed, and the size and quality of the images are in many cases insufficient for the operator to perform the required task, e.g. estimating the position and distance between moving objects in the scene. In order to reduce remote control traffic, feedback data can be compressed without loosing relevant information. When video feedback is not essential, information on the robot's operating state can be transmitted to the remote operator in the form of relevant events occurred in the scene (e.g. the object was grasped or dropped, an unexpected object appeared in the working environment, the robot has reached the target position, etc.).

Chapter *Advanced Teleoperation Architecture*, by Andrea Castellani et al. presents an approach to develop Internet-based teleoperation systems that deals with the problems of unpredictable delays and low bandwidth. The approach builds on distributed middleware technology and in particular on the real-time extension of the Common Object Request Broker Architecture. The proposed software framework for bilateral, i.e. force reflecting, teleoperation system takes into account the main features of a haptic system: real-time behavior, distributed resources and general purpose structure. The framework support the interconnection of different devices, the exploitation of different control algorithms, the processing of data from several and different sensors and the use different connection media. RT-CORBA has been selected for its characteristics that make it a suitable middleware for teleoperation systems: standard object interface in support of incremental modularity; hidden object location, implementation and transport protocols independence, suitable for a distributed software infrastructure; platform (programming language, CPU architecture) independent, thus capable of using optimal module implementation; real-time support; optimization of memory management, network protocols, code generation; support of different transport protocol; performance optimization, i.e. the run-time scheduler maps client requests to real-time threads priorities.

2.2 Shared Control

Virtual Reality is often used as a convenient tool for visually reconstructing the robot's environment and behaviour from raw sensory data (e.g. the distance measurements of the robot's sonars), from aggregate data (e.g. the 3D shape of objects detected with a stereo vision system) or from geometric data of a previously known environment (e.g. the blueprint of a known building). Virtual Reality models can be effectively defined to represent industrial environments, which are usually well structured, static or their dynamic is known. Nevertheless, accurate VR models are difficult to define and maintain. Typically, they are build off-line using automatic reconstruction techniques based on stereo vision or active vision. Sensory information are affected by uncertainty and are usually incomplete.

A Virtual Environment (VE) is a 3D visual simulation that represents the computational geometric model of a real environment created with computer graphics techniques. A VE includes several virtual objects that represents the real objects that are found in the real environment. Some of that virtual objects can be virtual robots. Visual simulation can run much faster than the real robot, thus the operator can predict the final state of the robot before the command has been received and executed. The operator can then use this information for preparing the next command and issuing it in advance in order to compensate the transmission delay.

For each task to be performed by one or more robots, one or more operators analyse the problem, plan the sequence of actions that solves it, verify the robot behavior in simulation, and finally, transmit the batch of commands to the remote real robot. The VE simplifies the task analysis and command definition, since each operator is no more constrained by the single point of view of the robot's video camera. Instead, operators can observe the virtual robots, their tools, and the surrounding objects from every position in the space by changing the viewpoint to an arbitrary position, zooming in/out of the scene to an unlimited extent, and using semitransparent images. Once defined, the batch of commands can be executed repeatedly every time the same task has to be performed. During the task execution, the operator plays the role of task supervisor. If the environment characteristics or the task specifications change, the operator must define a new sequence of commands and transfer it to the remote robot.

The robot autonomy becomes now significant for the correct execution of the batch of commands received from the remote operator. The feedback control loop is completely local, thus the robot must be able to react to unexpected events in a more robust way. In order to increase robustness, commands can be transmitted to the robot along with the predicted final state, in order to allow the robot to compare its state with the predicted state and check for consistency. If significant differences are identified, the robot notifies the remote operator that an unexpected event occurred.

The information exchange between the operator and the remote robot is not only limited to strings of textual commands but involves code fragments in a script language and high level data structures. The goal is to avoid or at least minimize the operators' intervention during the robots' operations. The exploitation of VEs to reconstruct the robot's environments allows to the operators to work off-line in tasks where the robots have more autonomy (e.g. testing planning methods on the visual simulation of a known environment), monitoring the right execution and also to save communication bandwidth. The robot can simply transmit information related to variations in the scene (such as the position of a moving object) instead of a video stream.

Chapter *A Multi-Robot-Multi-Operator Collaborative Virtual Environment* by Moises Alencastre-Miranda et al. addresses the issues involved in developing a Collaborative Virtual Environment (CVE) for robotic applications, that is a system that allows multiple users in different geographical locations to interact, share, communicate and collaborate in a given task at the same time in a common virtual environment. In particular, the chapter illustrates specific software development design choices related to:

1. How to model a shared sense of space, presence and time.
2. How to implement this sharing model on a distributed networked environment using a software middleware.
3. How to synchronize distributed applications that interact in the virtual environment and that operate robots in the real environment.
4. How to promote modularity and interoperability of the CVE.

2.3 Mobile Control

The two Networked Robotics paradigms described in the previous sections are characterized by different mixes of robot autonomy and operator's intervention which determine the information exchange between the operator site (the Client) and the robot site (the Server).

In most distributed systems, the information that can be passed between the Client and the Server is limited to a small number of primitive data types (e.g. strings, integers, bytes, etc.). Code Mobility allows clients and servers to exchange full-fledged software components (mobile objects) carrying over both the data and the code that manipulates it. Mobile objects might be passive components or threads that start running when the downloading has been completed. An object can migrate at run-time from the Client's to the Server's execution environment (Mobile Agent) or vice-versa (Code on Demand).

The main advantage of moving the code through the network is to perform the computation where the resources are located (data, physical devices, human operators). The goal is to reduce the interactions between distributed subsystems and thus the network traffic. While Code on Demand offers the opportunity to send knowledge and action to a remote location as a single software component, a Mobile Software Agent encapsulates knowledge, action and past experience in an autonomous, asynchronous, dynamic and potentially intelligent (i.e. able to learn) software entity, able to move multiple times within a distributed but unified execution environment (which supports the execution of individual mobile agents but protects an underlying network of host computers and robots). The flexibility, autonomy and intelligence which mobile agents offer directly translates into increased flexibility, autonomy and intelligence in the host architecture. In addition, the mobility of mobile agents within such an architecture can be used specifically to provide a range of functional benefits, including increased system adaptability and fault tolerance, and the ability to dynamically allocate control.

The adaptability of a distributed architecture can be improved by employing mobile agents to automatically move and integrate updated control code at run time, and in parallel, in multiple networked robots; fault tolerance can be improved by using mobile agent mobility to move important mission level data away from failing system components, or store such data to persistent media such as hard drives for later retrieval; while dynamic allocation of control can be achieved by relocating control in a distributed multiple telerobot architecture in the event of significant changes in network topology (caused by possible robot failures or additions); so as to maximize control and reduce the effects of communication delay between a user and the controlled robots, as found in Internet based telerobot control architectures.

Typically the operator interacts with the robot through the Internet using a desktop PC or even a mobile device, such as a laptop or a Personal Digital Assistant (PDA). If the robot is accessible through a Web portal, code mobility allows the operator to download the front-end to the robot that best fits the computational capabilities of its client device, e.g. an Applet with a VRML environment for a desktop PC or a Middlet with a simple graphical interface for a PDA. For example, the operator can supervise the robot activity using a PDA to visualize the current values of the robot's state variables, such as its current task, location, operating status, the battery level.

Usually, when the operator needs to upgrade the robot's functionality, the robot must be stopped and a new version of the software control application installed. Code mobility can simplify software upgrading as it allows to replace single software components while the application is running. Furthermore, it can be performed remotely. For example, the operator could upgrade the sensory data fusion algorithm by sending the mobile object that encapsulates it to the remote robot. This is useful when a new type of loom inspection requires a different video images processing algorithm.

The information exchange between the operator and the remote robot could be enhanced using code mobility since this technology can reduce network traffic and improve communication reliability. If the needed relevant data (e.g. a binary information) is much smaller than what the robot will answer to the operator request (e.g. a video stream), the algorithm (encapsulated in a mobile object) that elaborates the information can be transmitted to the robot's place obtaining a significant optimization. This is the case of the operator who has to achieve a succession of possible interdependent interactions with the remote robot (for example successive loom inspection, sensor data filtering and interpretation) before getting the relevant data (e.g. the identification of a malfunctioning component). This kind of interaction can be improved by sending a mobile object to the robot's site that handles the composite request at once and gets back the result.

Code mobility enhances the concept of autonomy transfer, as it allows the operator to transfer the code implementing the control algorithm that generates the sequence of batch commands or that carries on abstract plans along with the procedures to decompose them in simple robot actions.

The benefit of code mobility for autonomy transfer can be appreciated in those application scenarios for which the maximum level of flexibility is required, such as when the robot operates in ever changing environmental conditions (e.g. rescue robotics) or when the robot functionality must be customized for remote training of students or technicians.

The robot can play the Client and Server roles interchangeably. It plays the role of Server when it receives mobile objects from the operator's site, and it plays the role of Client when it downloads mobile objects from other sites or resources.

Chapter *Modularity and Mobility of Distributed Control Software for Networked Mobile Robots* by Liam Cragg et al. presents the authors' experience with a number of implementations, namely autonomous, telecontrol and AI robot architectures with which they have explored the properties of mobile agent mobility. Although many advantages of mobile agents are claimed in literature, our investigation has systematically examined whether mobile agents are applicable and beneficial in the multiple robot domain. In particular the chapter addresses the following issues:

1. Which kind of functionality does a mobile agent implement?
2. Which support does this technology provide to the software development process?
3. How to enforce secure and reliable interactions between the mobile agent and the robot?
4. How to test and debug the agent's code that will interact locally with the remote robot?

3 Conclusions

The development of Networked Robotics systems involves the combination of several technologies, such as computer graphics, computer network, distributed computing, and multithreading execution in addition to the most sophisticated techniques to build autonomous, fault-tolerant, and efficient robots. Advanced software environment play a key role in the development of Networked Robotics systems as they can simplify the integration of heterogeneous distributed technologies and enhance the performance of the overall system. The three approaches presented in the following three chapters face similar problems but from different perspectives. The challenge for future research is to explore their synergism in order to exploit them in more complex scenarios.

References

[HRI04] Special Issue on *Human-Robot Interaction*, IEEE Transactions on Systems, Man and Cybernetics, Part C, Volume: 34, Issue: 2, May 2004

[SG03] Siegwart, R. Goldberg, K. Ed., Special Issue on *Internet and Online Robots*, Autonomous Robots, Volume 15, Number 3, November 2003

[SG02] Siegwart, R. Goldberg, K. Ed., *Beyond webcams, An Introduction to Online Robots*, MIT Press, Cambridge, MA, 2002.