Simulation and teleoperation tools offer many advantages for the training or learning of technological subjects, such as flexibility in time-tables and student access to expensive and limited equipment. In this paper, we present a new system for simulating and teleoperating robot arms through the Internet, which allows many users to simulate and test positioning commands for a robot by means of a virtual environment, as well as execute the validated commands in a real remote robot of the same characteristics. The main feature of the system is its flexibility in managing different robots or including new robot models and equipment. © 2005 Wiley Periodicals, Inc.

1. INTRODUCTION

The teaching of practical subjects or professional courses in Robotics requires traditionally expensive equipment and many times is insufficient to be used by many students simultaneously, as for example robots and their controllers. Moreover, it is possible that this equipment can be damaged if it is used improperly. Another problem derived from requiring a laboratory with the appropriate equipment to carry out the practices is the need for the students to go to such laboratories according to some strict schedules.¹ ² Nowadays, it is being demonstrated that new technology such as the Internet, the teleoperation of systems and the virtual reality allows the student to carry out practical exercise from other places (like at home) without the above-mentioned problems by using virtual laboratories.³ ⁵ Other advantages of these systems of remote practice are that they allow a great number of students to have access to expensive and scarce equipment, and that they can also offer group-working or self-evaluation mechanisms.⁶ ⁸

There is a great amount of Java-based teleoperation applications thanks mainly to the portability of
this language. Many of the applications for Robotics are for simulating or teleoperating educational equipment\textsuperscript{6–8} or simple robots.\textsuperscript{3,9–11} In the field of industrial robot arms there are fewer applications,\textsuperscript{4,5,12} and generally they are designed for specific robots. Moreover, very few of them are based on a open architecture, which offers the required flexibility to change the robot being used or to add new robots without modifying either the user-interface or the architecture of the system.\textsuperscript{13,14} With regard to the simulation, there are not many Java-based applications for industrial robots that offer a realistic virtual environment, and the majority represent only wired models or simplified structures.

This paper presents a new system for training in Robotics that allows, on the one hand, the simulation of movement commands in a virtual robot, and on the other, the teleoperation of an equivalent of a real robot arm through the Internet by several users. This system is focused on the training of kinematics and the trajectory design for industrial robot arms. The user interface is always available for executing simulations in the virtual environment, although the teleoperation capabilities may be disabled or the user may not have any privileges for using the system.

The main feature of our system, in contrast to others, is that it offers the flexibility of managing different robots or including new robot models and equipment, as well as other kinds of passive objects in the workspace, without changes in the user-interface and the system’s architecture. The system developed also has other interesting features, such as a feedback to the user based on updating a graphic simulation while the robot is moving, the use of high-level communication protocols, and the modeling and the virtual representation of the robots, passive objects and scenarios based on Java 3D. This system displays a more realistic simulation than the majority of the proposed Java-based systems for simulating robots, and the simulated robot can be very similar to the real robot. It also offers a user interface which is very user-friendly for operators who are not specialists in Robotics.

Java 3D is an API (Application Programming Interface) from Sun’s Java (TM) that provides 3D graphics.\textsuperscript{15} It is similar to other APIs, such as OpenGL. The main advantages of using Java 3D, in contrast to other APIs for Java or other platforms such as VMRL, which has been used traditionally for Robotics, is that Java 3D is a standard extension of Sun’s Java which provides good integration with other Java extensions and a good performance in many systems.\textsuperscript{5,16} Java 3D uses the scene-graph approach, which is a hierarchical approach for describing objects and their relationships among one another.\textsuperscript{17}

The paper is structured as follows. The following section describes the different aspects of the system’s architecture, such as equipment, functional structure, communications and feedback options. Afterwards, Section 3 explains how the robots are modeled, using Java 3D. The user-interface features are shown next, in Section 4. Section 5 briefly describes some experiences in teaching, and finally, some important conclusions are shown in Section 6.

2. SIMULATION AND TELE-OPERATION SYSTEM

The different aspects of our system’s architecture for simulation and teleoperation of robot arms are detailed in what follows. This system, called Robolab, is the evolution of another environment previously developed by us and includes many new features.\textsuperscript{25} The system is accessible from http://www.disclab.ua.es/robolab/.

2.1. Equipment

Figure 1 shows the different elements that compose the system. There are two well-defined parts separated by the Internet: the user’s computer and the laboratory equipment. The user’s computer requires only an Internet access, an up-to-date web client and Java and Java-3D runtime components as software.\textsuperscript{15} This allows users to use different kinds of computers or operating systems to access to the system. Several clients can connect to the system simultaneously from different users’ computers.
In the laboratory, the only pieces that require any considerable investment are the robots and their controllers. At present, we are using two robots: a Scorbot ER-IX (Intelitek) robot arm, with 5 DOF with an electric gripper, and a PA-10 (Mitsubitshi), which is a robot with 7 DOF plus a tool. However, the system can manage other robot arms, or even, operate more than one robot at a time, as described later on in this article.

The other components of laboratory equipment are as follows. First, the main server is a PC that includes the web server from where the user can download the Java applet with the interface to simulate and teleoperate the robots. This computer also manages the users’ access and accounts. The teleoperation server is in another PC that validates the commands to the robot received from a user’s computer, translates them to the appropriate language, and sends them to the robot’s controller. There can be more than one teleoperation server to manage more robots. The video server is an AXIS 2400 that allows users to receive video streams as feedback during the teleoperation processes. Besides, a Siemens S7-200 PLC, which is directly connected to the main server through a serial interface, switches several devices on/off, such as the lighting system, the robot, the cameras and the internal servers. Finally, a computer with Linux is used as a firewall to increase the system’s security.

2.2. Functional Structure

With regard to the operating of the system’s software, there are three main blocks to be considered, as shown in Figure 2.

The client software is a Java applet that can be loaded from the main server. In this applet, the main part is the module that manages the simulation and the teleoperation thanks to a 3D model of the corresponding robot which is based on a library of Java-3D classes which will be described in Section 3. The simulation is drawn on a graphic virtual environment by the user-interface module. These two modules are coordinated by an event manager which, on one hand, controls the interface devices events (keyboard, mouse, joystick, etc.) and, on the other hand, controls the communication events. The features of the client-applet will be described later on in Section 4.

The main server is supported by M.S. Windows NT or 2000 and the M.S. Internet Information Server running on a PC. The web server offers the web page where the Java applet for the client-applet and the manuals, exercises, etc. are found. A user can download the exercises and the applet, and use the applet to do a simulation of the robot commands, without any login restrictions. Thus, the system can be used to practice concepts although the teleoperation capabilities are not available in that moment.

When the user gets a command list which has been validated by the simulation, he can request a teleoperation from the web server. At this moment, an ASP module (Microsoft Active Server Pages) takes over the control, and verifies the user’s identity. If the user is registered in the user database, the ASP module creates a Java module that acts as a communication interface or bridge between the client-applet and the corresponding teleoperation server. To select the adequate teleoperation server and robot, the client-applet sends some configuration parameters to the web server when requesting the teleoperation. It also is possible to specify the robots that each user can teleoperate in the user database.

Finally, there can be one or more teleoperation servers running on one or more PCs in the system. A teleoperation server is a Java program that attends connections from the main server. A connection includes a command list to be executed in the robot and the corresponding feedback data. When a teleoperation server receives a command list, it does a simulation of the commands in order to verify that they are correct. This simulation is based on the same library of Java-3D classes that the client-applet uses, and guarantees the correct use of the robots (Section 3). The library includes not only the robot models but also the corresponding command interfaces for languages and protocols of the robots’ controllers. Only if the simulation is correctly done, are the commands translated to the robot language and sent to the robot’s controller.

To manage the access from several users to the same robot, the teleoperation server attends to them in a sequential order, returning the robot to the home
position before executing the command list of each user. Besides, each user has a maximum time of use configured in the main server’s database.

The aspects of the communications and protocols are described in the next section.

2.3. Communications

On the one hand, for the communication between a client-applet and the main server (Figure 2), the protocol HTTP is used. Although this is a heavily high-level protocol, it offers many advantages, as described below. In any case, the exchanged information is compact, facilitating its quick exchange.

The main advantage of using a high-level protocol like HTTP is that any connection between a client-applet and the main server is possible, independently of the networks and firewalls to be crossed, provided that the necessary ports are enabled. This is not always guaranteed with IP sockets, due to the need for IP addresses that are either public or visible and firewalls or routers can translate or block some addresses. Another important advantage is the option of using HTTPS (Secure HTTP) to obtain a secure connection in which the data are signed and encrypted. This option only requires the correct configuration of the web server and the port-setting in the client-applet, without having to change the version of the programs.

The data exchanged between a client-applet and the main server are codified as URL strings to be sent in HTTP. These data include information like the user’s login, configuration parameters or the command to be executed by the robot.

On the other hand, the communication between the main server and the teleoperation servers is done through IP sockets, since, in this case, the computers are in the same private LAN and there are no problems of addressing or security.

After a client-applet has been connected to the main server and the login authentication process has been successful, a direct communication between the client-applet and the teleoperation server is established, over the HTTP and IP sockets protocols. This communication allows the exchange of commands from the client-applet to the teleoperation server and feedback data in the opposite direction, and is based on the simple protocol shown in Figure 3. The client-applet sends a list of commands, which must have been previously tested in a simulation. Each command of the list is composed of a type-identifier which represents the order to be executed (home, move to, etc.), the joint values associated with the command (if they are necessary), and the times associated to the movements of each joint.

It should be pointed out that the system does not perform a control-loop of the robots through the Internet, and only tested command lists are sent to the teleoperation server to be executed remotely by the robot and its controller, in order to get real movements. For this reason and to facilitate the users’ access to the system from any place, the communication is based, up to now, on standard Internet protocols instead of real-time protocols with time management. However, we are working on this kind of protocol in order to make the feedback better, as described in the next section.

2.4. Feedback Options for Tele-Operation

The system permits two options for performing the feedback to the user during a teleoperation: an online video stream from the robot workspace or a graphical updating of the 3D simulation in the client-applet based on information about the robot’s current state received from a teleoperation server. The two options can be configured and activated independently in the client-applet.

The video feedback is the most usual way of doing teleoperation. However, an appropriate video feedback requires a sufficient bandwidth for the connection between the user and the teleoperated system. In an educational environment, many students usually have access to the Internet through the telephone network and, in such cases, a video stream does not allow correct teleoperation at all times. For
this reason, we have developed the graphical updating procedure,\textsuperscript{24} which supposes a new feature in the systems for teleoperating robot arms, in contrast to the common video feedback.

With graphical updating, the teleoperation server replies to a command list sent by a user with a flow of packages that contains data feedback concerning the position of the different joints of the robot arm while the movement is being done, as shown in Figure 3. This data feedback is used by the client-applet to update the robot-model and its graphical representation in the user-interface. To reduce the information to be transmitted, the position values are only sent when the robot makes a significant change of position. This way of doing the feedback, requires a lower bandwidth than the video stream does. At present, we are working on improving this kind of feedback by using a protocol which considers time management like RTP, or other UDP-based protocols, in order to discard out-of-date joint values.

While the data feedback is included in the protocol between the client-applet and the teleoperation server (see previous section), the video feedback is done independently by means of an authenticated connection between the client-applet and the video server.

3. SIMULATION AND ROBOT MODEL

The most important feature of our system is the flexibility to define new robot models and include them in the client-applet and the teleoperation server. In this way, new robots can also be incorporated into the laboratory equipment to be teleoperated. To model a robot, a library with a set of different Java classes has been created by us from the Java 3D API. These classes allow us define the structure, the graphical representation and the kinematics model of the desired robot, besides other parameters.\textsuperscript{13} This library does not take into account the robot’s dynamic model.

With this library (Figure 4), a robot is composed of different chained links (RLink objects) and joints (objects extended from RJoint) in a structure called scene-graph. A modeled robot also can have a tool represented by an RTool object. In addition, RShape objects are used for graphical representation of RJoint and RTool objects.

An RShape object is a basic object that stores the graphic representation each of the robot’s components. Besides, it can also be used to represent the scene’s static elements. It is basically composed of a Java 3D transformation-group that allows the definition of the location and orientation, and a graphic object. This graphic object can be either a basic Shape3D object of Java 3D, which represents a simple element to be displayed (for example, a joint), or a subscene-graph of Java 3D with a complex form (for example, a static object in the virtual scene, such as a work-table).\textsuperscript{17} The graphic object can be created by programming with Java 3D functions, or can be loaded from a local file with 3D data (WRML; 3D Studio or other formats) or from an URL. In addition, the RShape object also defines the basic methods for collision-detection.

RElement defines an abstract class which represents a generic element of a robot. As shown in Figure 4, the RJoint and RLink classes extend the RElement class. An RElement is basically composed of a transformation-group and an RShape object. The transformation-group defines the location and orientation with regard to the previous element of the robot, and the RShape object is used only in the RLink and RTool classes to define their graphical representation. For these two classes, the RElement can also include a material definition. The RLink class represents a physical element of a robot, whereas the RJoint class represents a robot’s generic joint. In contrast to an RLink object, an RJoint does not require any graphic element associated to it. An RJoint object includes properties to define the minimum, maximum and home values, as well as several ways of moving the joint.

RJoint is an abstract class which is the base for other more specific classes: i.e., RRotatoryJoint, which represents a rotational movement, RPrismatic-
Joint, which represents a movement along an axis, and RTool, which represents a tool at the end of a robot. A tool has a movement, which can be considered as one DOF when modeling it. Thus, it is modeled as a joint. This abstract class can be extended to get specific tools, for example, a gripper modeled with RGripper. New kinds of joints and tools can also be incorporated into the library by the user.

As an example, Figure 5 shows a model that corresponds to a Scorbot ER-IX robot, which has 5 rotary joints (q1 to q5), 5 links (L0 to L4) and a gripper. The figure also shows the scene-graph obtained by linking the first five RElements. The nodes T are Java 3D transformation-groups, nodes B are necessary Java 3D branch-groups that act as the root of a sub-graph, and nodes S are Java 3D graphic objects to be displayed in the graphical simulation.

Finally, all the elements that compose a specific robot are included in an object derived from RRobot class, which represents the complete robot model. This object also has high-level operations for moving the robot, calculating the inverse or direct kinematics or detecting the robot’s collisions either with itself or with the environment, in addition to the functions which are necessary for the teleoperation. These functions are basically a command interpreter that processes the commands received and translates them into the robot-controller’s language, and a module for communicating with the controller.

Thus, a robot is modeled as an RRobot object in a transparent way for the client-applet and the teleoperation server. In the applet, the object offers the simulation and graphical representation. In the teleoperation server, an RRobot object also offers the necessary command interface and protocol to communicate with the robot’s controller, in addition to the simulation.

To include a new robot model into the system to provide simulation capabilities through the client-applet, the RRobot class must be extended with the specific kinematics and graphic characteristics of the robot. For the kinematic characteristics, we must specify elements such as the java scene-graph, which is composed of the necessary linked RElements that represent the robot structure, and the maximum and minimum joint values. If a real robot is available for teleoperation, the functions of the teleoperation server (command interpreter and controller’s language translator) must also be specified.

Apart from the models for simulating or teleoperating robots, it is also possible to define models for scenarios and passive objects in the robot’s workspace. These models can be incorporated into the client-applet in run-time from a local file or from an URL, thanks to a file-loader. The file-loader performs the job of taking a file or stream in a standard format such as VRML and turning it into a Java 3D. We are now working on the updating of the position of the passive object by the client, with real information from the workspace in the laboratory, so that the robot can manipulate and interact with objects. In a teleoperation server, the scenarios and objects represent the real robots’ workspace so that any possible collisions can be correctly detected.

4. USER-INTERFACE

Figure 6 shows the aspect of the interface that the client-applet offers with two different robot models. In the first case, the client-applet simulates and displays a Scorbot ER-IX robot, and, in the second case, the client-applet represents a Mitsubishi PA-10 robot. In the both cases, a workspace with the same objects is included in the simulation, in addition to the robot.
Figure 6. User-interface of the Java applet.
The interface shows the robot simulation, the transformation matrix with the position and the orientation of the end tool, the values of the robot joints, and the list of tested movement commands entered by the user, in addition to other options. It is also possible to modify the point of view of the graphic representation with a mouse.

The applet is able to use different devices to define a position or movement for the robot. Thus, it is possible to use either, a keyboard, a mouse or a commercial joystick for games, with a force-feedback, to move the simulated robot easily and directly. In the case of a joystick, the collisions detected by the simulation are transmitted to the user as a sensation of resistance by the joystick. Furthermore, the user can specify the exact joint values, the transformation matrix that indicates the Cartesian coordinates of the end tool and the duration of the movements, using some dialogue windows (Figure 7). This latter option is more appropriate for doing exercises with specific values for the trajectories. The robot’s tool can also be activated through its function.

Thanks to the simulation, the students can practice and carry out movements, and store the validated movements in the command list of the interface in the same order. It is possible to read a previously saved list from a local file, or execute the commands of the list step by step. The client-applet is signed to validate its authenticity and allow it access to local files.

When the user activates the teleoperation option, the command list is sent to the laboratory equipment to be executed by the robot. To use the teleoperation, the user has to configure some teleoperation settings, such as the user’s name and password, the URL of the main server and the address and port of the teleoperation server in a dialog window. The interface gives the user three options for performing the feedback of a teleoperation, as described above in Section 2.4.: either an online video stream or a graphical updating of the 3D simulation in the client-applet, or both.

Another interesting feature of the applet is the possibility of loading objects and scenarios into the simulation in run-time, from a local file or from a URL, as described in Section 3. Thus the scenario that virtually represents the workspace of the laboratory and some common objects can be available for the users in the main server’s web. Figure 6 shows an example of the interface with a scenario where there is the ground plane, a work-table and other static objects in addition to the modeled robot.

Finally, it should be pointed out that we have also developed an application version of the applet. With this version, the user does not need to have to access to the web server to download the applet. However, this version is mainly oriented to robot simulation, and it has not all the teleoperation features of the applet version.

5. EXPERIENCE IN TEACHING

The authors of this article have been using this type of simulation and teleoperation technology for teaching subjects on Robotics for several years, and systems like the one described above have been used in practice groups as part of a virtual laboratory. To evaluate the virtual laboratory’s acceptability and its effect on learning, we have carried out several statistical studies during the last three academic years. We shall now comment briefly on the most important conclusions of our studies.

Although there are a great number of students who happily accept the simulation and teleoperation tools that offer them a flexible time-table for their experiments, many others prefer to have a real laboratory at the university where they can work in coordination with their classmates and have the support of a teacher. In other words, the students consider the virtual laboratory to be a valuable complement to the teacher and traditional teaching, but never a substitute for them. Thus it is very interesting to investigate web environments that offer not only virtual tools but also ones that allow the students to share their experiences and their results, while having a virtual teaching support. Our research team is now working along these lines.
It has also been shown that the remote access to costly tools and resources like robots is positive and interesting, since it makes practice more attractive and real in comparison to a mere simulation.

Finally, there is no doubt that a simple interface with a good simulation, like the client-applet described in this article, helps the students to save time in learning to use the tool and allows them to concentrate on the more important aspects of the course. In any case, it must be assured that the software required for the student’s computers is accessible and easy to install and run. To do so, Java is a very good option.

6. CONCLUSIONS AND CURRENT WORKING-LINES

In this paper, we have presented a new system for the simulation and teleoperation of industrial robot arms that includes several important features in comparison to other systems. Our system is mainly oriented to the teaching of technical courses and allows many students to test their simulated commands in a real robot through the Internet. Students can test practical concepts about the robot’s kinematics and evaluate the trajectories for the manipulator designed in the class.

The user-interface applet is always available for executing simulations in the virtual environment, although the teleoperation capabilities may be disabled or the user may not have any privileges for using the system. It is only necessary to have the required Java libraries installed.

The main feature of the system is its flexibility in changing the robot model used in the simulation or in adding new robots to be teleoperated in the laboratory, without any changes being required in the user-interface. The library of classes created for modeling robots, which is based on Java 3D, facilitates the specification of new robot models and their inclusion in the system. The system can even manage more than one real robot. The client-applet is also able to incorporate scenarios and passive objects in run-time, and the simulation detects collisions of the robots with their environments and with themselves. Thus, in contrast to the majority of existing systems for teleoperating industrial robots, the architecture of our system is not specific for any particular robot, and it is easy to include new models for real robots, as well as their corresponding teleoperation capabilities.

One of the main advantages of using a system like the one presented in this paper for teaching robotics is it prevents the equipment from being damaged as a result of incorrect use. Our system only permits teleoperation if the commands to be executed by the robot have been tested beforehand with the simulation that the client-applet offers. Moreover, the teleoperation server always verifies the command lists with its own simulation engine before the final execution.

The teleoperation function can be used from almost any computer that is connected to the Internet thanks to a high-level protocol designed over HTTP, in contrast to other existing systems that use socket-based communications and, thus, can have problems with intermediate firewalls or proxy servers. Furthermore, the option that enables the graphic representation in the client-applet to be updated with the real state of the robot, permits an on-line feedback with simple accesses to the Internet, with a lower bandwidth requirement than a video stream.

Due to the fact that the system does not consider a control loop of the robots through the Internet, but rather that tested command lists are sent to the teleoperation server to be executed remotely by the robot and its controller, special real-time protocols with time management have not been considered necessary. However, we are now working with these kinds of protocols to improve the feedback based on the graphic simulation updating.

When Internet is used as a transporting network, it is very important to include security instruments to protect the transmitted information and the system. The system we have presented here not only has user-authentication and accounting, but is also able to use HTTPS, which guarantees that the information exchanged between the client and the server through the Internet is not accessible to third parties. The client-applet is signed to validate its authenticity to the user.

Finally, we should point out that the user-interface offers different methods of defining the movements of the robot. Thus, it offers both, an easy and direct control of the simulation by means of a mouse or a joystick with force-feedback, and a precise specification of the robot’s position by indicating the joint values or the transformation matrix.

At present, we continue developing the system and we are working on including new features. The first is 3D recognition of basic objects in the workspace and their inclusion in the simulation as 3D models, with the ability of manipulating them with the robot. This important feature has been considered by some authors in their systems. The second feature is the force feedback to the user. This feedback is
based on two sources: the forces generated virtually by the simulation when collisions are detected, and the force read by a sensor in the robot’s end tool while manipulating objects.

ACKNOWLEDGMENTS

We are grateful to the “Agencia Valenciana de Ciencia y Tecnología” of the Generalitat Valenciana regional government for their financial support to the research project CTIDIA/2002/108.

REFERENCES