An Advanced Interactive Interface for Robotics E-Learning

doi:10.3991/ijoe.v4i4.561

C. A. Jara, F. A. Candelas and F. Torres
University of Alicante, Spain

***Abstract*—Virtual and remote laboratories have improved learning and training in the academic community. They allow students to acquire methods, skills and experience related to real equipment in an intuitive and cost-effective way. The purpose of this paper is to present the development and the implementation of an e-learning environment in the field of Robotics. The main aim of this application is to allow students to simulate and to teleoperate a robot arm in an easy and user-friendly way, although it also includes many novel advanced features. The application has been developed using *Easy Java Simulations* (EJS), an open-source tool for people who do not need complex programming skills.**

***Index Terms*—Educational technology, simulation software, telerobotics, virtual and remote laboratories.**

I. INTRODUCTION

Theoretical lessons do not provide enough robotic knowledge for students. Laboratory work offers them practical issues to improve their robotic experience. However, many problems exist in giving students sufficient educational robotic laboratories. These include expensive equipment and limited time. As a solution to this, new technologies such as *Virtual and Remote Laboratories* in telerobotic systems provide a distance teaching framework which meets the student hands-on learning needs. They solve the problems mentioned above, offering a great number of advantages:

- Remote practices: students can take part without having to be present in the laboratory.
- Learning in a free and flexible way in contrast to a fixed and regular class schedule.
- Remote access to real equipment that allows the management of robots through the Internet from anywhere at anytime.
- Expensive systems can be used, which would generally be impossible in common situations.

This paper presents a virtual and remote laboratory called *RobUaLab.ejs* for training and learning in robotics. In general, this application allows a complete simulation of a robot arm by means of a virtual environment based in a Java applet. Path planning algorithms validated in the simulation can be executed in a real remote robot through the Internet. In addition, this telerobotic system developed has other interesting features such as: 1) feedback to the user both on online video and graphical updating of the 3D simulation; 2) the use of high level protocols (HTTP/HTTPS) to have a transparent communication; and 3) a very realistic graphical interface (Fig. 1).

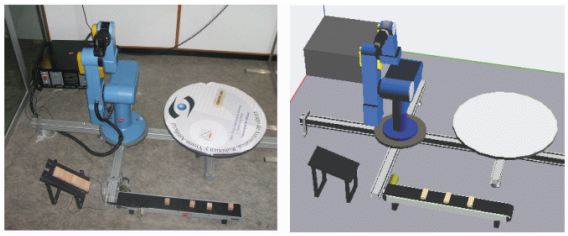

Figure 1. Real-Virtual environment

Nowadays, there are other robotic e-learning systems developed for an educational purpose. Among them, it is worth mentioning the following:

- ARITI [1]: a telerobotic system that allows to control a robot with an interface based on *Virtual Reality* (VR) and *Augmented Reality* (AR).
- UJI Robot [2]: a multi-robot architecture system that gives access to a robot arm by means of an interface which uses VR and AR.
- Robolab [3]: an open architecture for simulating and tele-operating different robot arms through the Internet.
- VISIT [4]: a telerobotic application which has advanced robotic technologies.

In contrast to the above mentioned systems, the virtual and remote laboratory presented here has the advantages of an application based on *Easy Java Simulations* (EJS) [5]: full portability and an interactive graphical user interface based on VR and AR. In addition, it allows one to manage many robotic functions which are novel in a free Java application like this.

The e-learning application presented is being used in the course "Robot and Sensorial Systems" in the Computer Science degree at the University of Alicante. Moreover, this system belongs to a network performed by different virtual and remote laboratories from Spain, coordinated by Prof. Sebastian Dormido called "AutomatLab" (http://lab.dia.uned.es/automatlab). With this virtual platform, students and teachers can experiment with real equipment and share knowledge by means of a collaborative environment based in *eMersion* [6].

This paper is organized as follows. Section 2 discusses the technical implementation of this robotic system, which includes the hardware components, software architecture and communication protocols. Afterwards, Section 3 describes the main features of the virtual laboratory developed. The remote capabilities are shown in Section 4. Next, some experiments performed with the application are explained in Section 5. Finally, the conclusions and some future work lines are shown in Section 6.

II. TECHNICAL IMPLEMENTATION

A. Hardware Components.

The different hardware components are shown in Figure 2. There are two clear parts linked by the Internet: the user's computer and the laboratory equipment. The *User's PC* requires only Internet access, a web browser, Java and Java 3D runtimes components as software. This allows users to use different kinds of computers or operating systems in order to run the application from anywhere at anytime.

In the laboratory, the only pieces that require a considerable investment are the robot arm, its controller and the automatic conveyor belt. For the development of this robotic lab, a robot *Scorbot ER-IX* (Intelitek) of 5 DOF with an electric gripper is used for teleoperation. The rest of hadware components are simple computers.

The *Main Server* is a PC which includes the web site from where the user can download the Java application to simulate and to teleoperate the robot. The *Tele-operation Server* validates the commands that the robot receives from a user's computer, translates them to the appropriate robot language, and sends them to the robot controller. The *IP camera* allows users to receive video streams by means of the HTTP protocol as feedback during the teleoperation processes. Finally, a *PLC* connected with the *Main Server* permits remote power control of the laboratory .

B. Software architecture.

With regard to the software design, there are three main blocks to be considered: the *Client Applet*, the *Main Server* and the *Tele-operation Server* software (Fig. 3). The *Client Applet* is an EJS application that can be downloaded from the *Main Server*. In this program, the main parts are the robot model which manages the 3D simulation, which is based in the EJS library of Java 3D, and the functions used in the teleoperation tasks.

When the user gets a path planning validated by the simulation, he can request the teleoperation module from the application. At this moment, a *PHP module* takes over the control, and verifies the user's identity. If the user is registered in the user database, the *PHP module* creates a socket communication which acts as a bridge between the *Client Applet* and the *Tele-operation Server*.

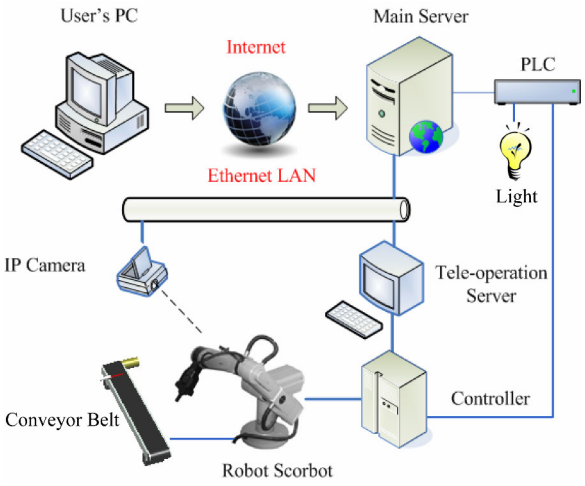

Figure 2. Hardware components

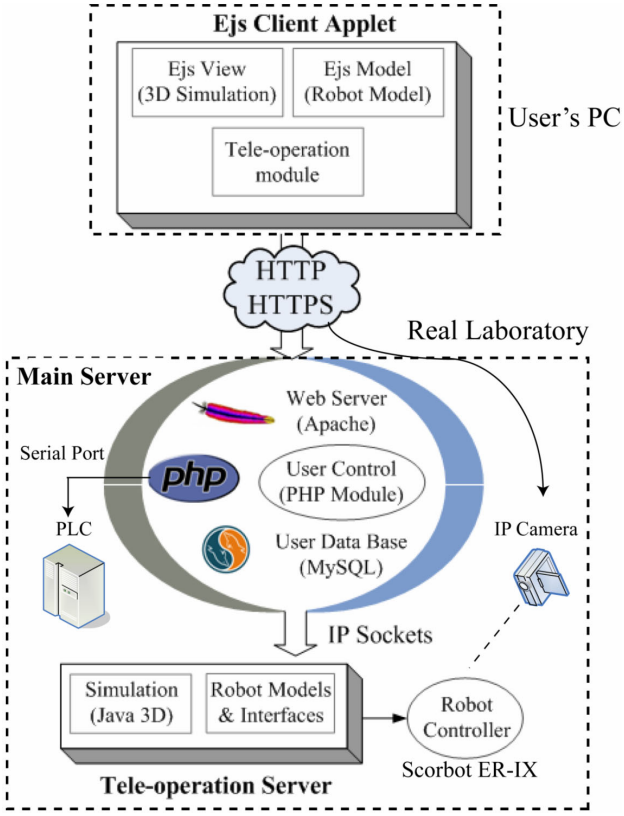

Figure 3. Software architecture and communication protocols

The *Tele-operation 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 the *Tele-operation 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 robot model as the client application and guarantees the correct use of the robot.

C. Communication protocols

The protocols HTTP and HTTPS are used in the communication between the *Client Applet* and the *Main Server* (Fig. 3). The main advantage of using high-level protocols is that any connection between a client and the web server is possible, independently of the networks and firewalls to be crossed, provided that the necessary ports are enabled. This simplifies the use of the application since users do not have to configure any network device or firewall.

The data exchanged between the client and the *Main Server* is codified as URL strings to be sent in HTTPS. These data include information such as the user login, configuration parameters or the command to be executed by the robot arm. On the other hand, the communication between the *Main Server* and the *Tele-operation Server* is done through TCP sockets and UDP packets because the computers are in the same private LAN. After the client has been connected to the *Main Server* and the login authentication process has been successful, a communication between the client and the teleoperation server is established over the HTTPS and TCP/UDP protocols. This communication allows the exchange of high level commands from the client applet to the *Tele-*

operation Server and feedback data in the opposite direction (Fig. 4).

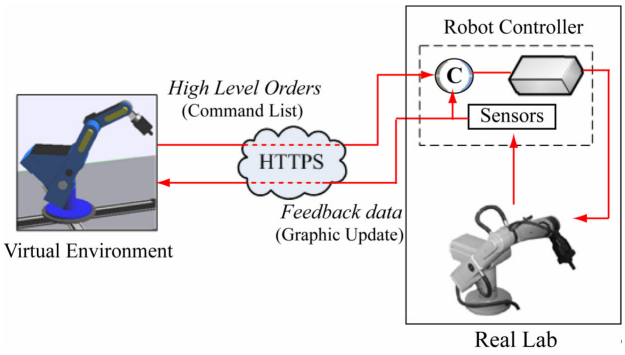

Figure 4. Scheme of the teleoperation with real-time feedback

The client sends a list of commands, which are previously tested in a simulation. Each command of the list is composed of a type-identifier which represents the order to be executed, the joint values associated with the command, 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 robot through the Internet (Fig. 4). Only tested command lists are sent to the *Tele-operation Server* to be executed remotely by the robot arm, in order to get real movements.

Finally, the HTTP protocol is also used to remote control both the *PLC* and the *IP camera*.

III. THE VIRUTAL LABORATORY

The virtual lab developed implements a large amount of options suitable for robotic e-learning. Students will be able to learn complex robotic concepts by means of a VR environment in an easy way. This section describes the main features of the virtual part of the applet and all the possibilities which are implemented for the user experimentation.

A. *User interface*

The appearance of the user interface is shown in Figure 5. The lower part on the left shows a 3D representation of the workspace where the robot arm is displayed. This robotic simulation has been developed using the Java 3D capabilities of EJS and represents a complete virtual model of the real environment. On the right of the application, there are some control display panels where users can view the time evolution of some model variables: position and speed (*Pos_Speed* panel), acceleration and actuator torque (*Dynamics* panel), transformation and Jacobian matrices (*DataC* panel), and the dynamic equation matrices (*DataD* panel). The control menu located in the upper part of the diagram allows users to save the experiments performed both in image format and in Matlab m-file format (*eJournal* option). This permits users to share result experiments with other users by means of the collaborative environment where the application is embedded. Finally, the upper part of the left contains several button controls with options which will be explained in the next subsection.

B. *Virtual laboratory's options*

The virtual environment developed allows users to experiment with a lot of options. Many of them are novel in a free Java application like this. Among them, it is worth pointing out:

- *Kinematics*: users can move the robot specifying both the exact joint values (*direct kinematic*) and the Cartesian coordinates of the end effector (*inverse kinematic*). Denavit-Hartenberg systems, transformation and Jacobian matrices can be seen in the user interface. In addition, the application detects possible singularities in the robot workspace.

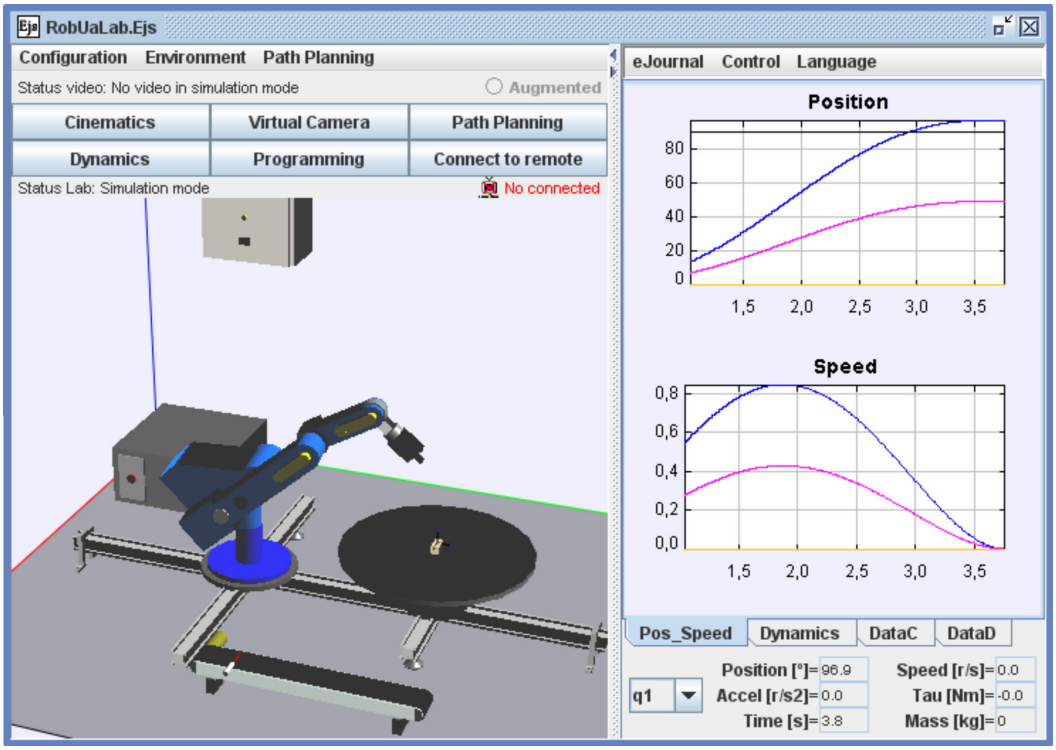

Figure 5. User interface

- *Path Planning*: users can practice and carry out movements of both joint trajectories (synchronous, asynchronous, splines and 4-3-4 polynomial trajectory) and Cartesian trajectories (line). The simulated trajectories can be stored in a command list and simulated sequentially. The user can also import and export trajectories to the software from a text file easily (Fig. 6).
- *Environment modeling*: users can introduce specific virtual objects in the workspace to do *pick & place* operations.
- *Dynamics*: users can evaluate the torques in the actuators when the virtual robot is simulating a task. They can modify dynamic parameters such as link masses, inertias and viscous friction from the robot and realize how the dynamics change.
- *Off-line-programming*: users can programme Java routines in the simulation. They can create variables, mathematical operations and order movements (Fig. 7). The trajectories simulated in the routines are stored in the command list to simulate sequentially. As well as in the path planning, users can import and export programs from a text file. A complete experimental example of a virtual off-line-programming routine will be shown in Section V.
- *Virtual Camera*: users can view a virtual workspace projection of an eye-in-hand virtual camera. This option will be used to perform visual servoing applications in future developments.
- *Visualization data*: as mentioned before, users can view in real-time all the values about the position transformation, kinematic and dynamic models of the virtual robot (Fig. 8).

IV. REMOTE CAPABILITIES

The application presented allows controlling remotely real equipment through the Internet. These remote experiences enhance the accessibility of experimental setups providing a distance teaching framework which meets the student hands-on learning needs. The next subsections explain the remote capabilities of the system and the way to access them.

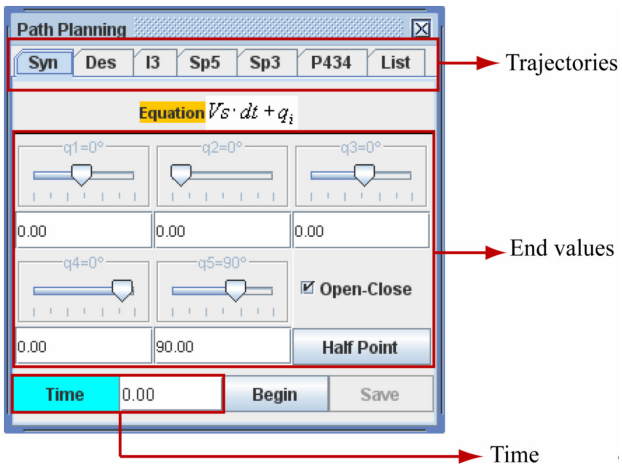

Figure 6. Path Planning interface

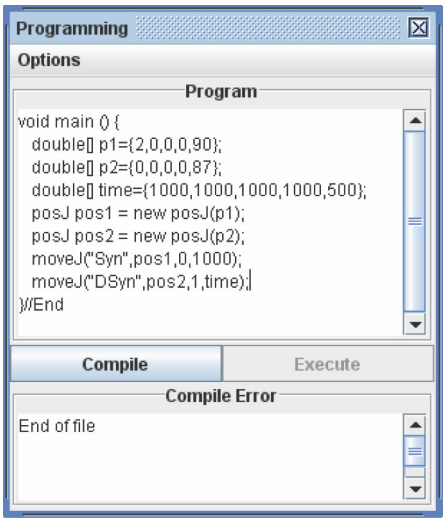

Figure 7. Off-line-programming interface

A. *Schedule system*

The application is embedded in a user restricted environment. Authorized students can download the applet at anytime from anywhere and experiment with only the options of the virtual laboratory. Remote access to real equipment is controlled by a schedule system. Thus, users can make a reservation of the real lab specifying the experiment timetable (day and time). This action creates a new line in the *User Data Base* from the *Main Server* (Fig. 3) with the user and experiment data (name, password, start and end of time of remote access). A thread process installed in the web server checks that users are in their correct timetable when they experiment with the real lab. In this way, only one user can control the robotic plant at the same time and it avoids multiple user connections.

B. *Teleoperation options*

The application allows the execution of high level tasks permitting users to interact with the real plant in a friendly and easy way. This remote experimentation is based on the high level protocols HTTP and HTTPS. This way, users do not have to open any port or firewall for the teleoperation and they only need a common Internet connection.

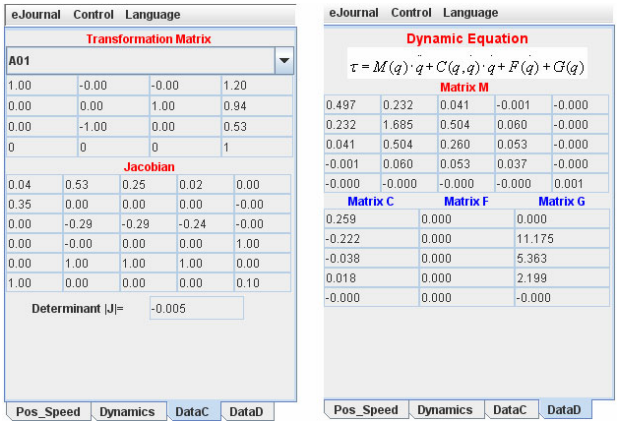

Figure 8. Visualization data about robot kinematics and dynamics

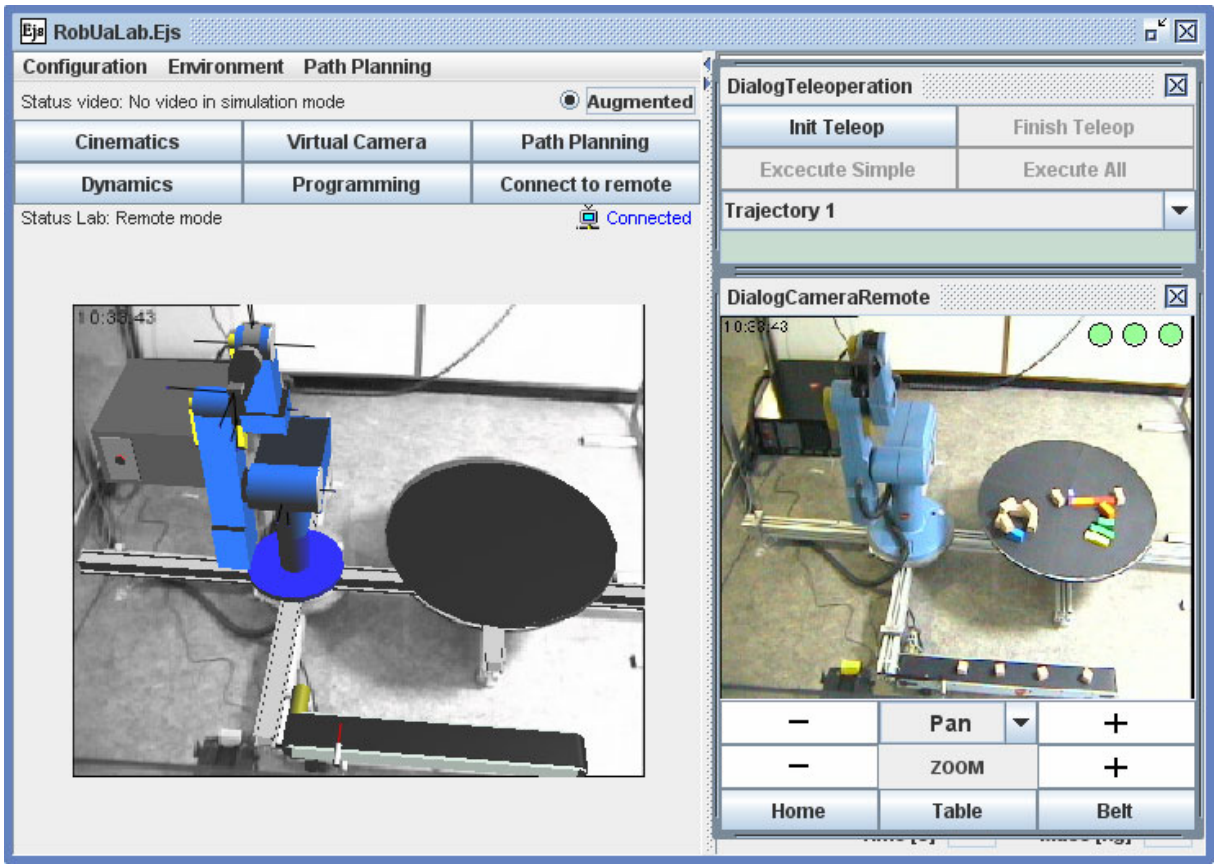

Figure 9. Remote experimentation using augmented reality

The teleoperation options implemented in the application allows the remote control not only of the robot, but also of some electronic devices of the real laboratory. They are the following:

- *Remote PLC/Camera* control: authorized users can control from the applet both some PLC control parameters (switch on/off both the light and the robot controller) and the real camera projection (pan, tilt and zoom).
- *Remote robot control*: according to a schedule, users are able to execute remotely in the real robot the command list stored in the virtual simulation. As mentioned before, the path planning sent to the real robot is previously checked in the *Teleoperation Server* which detects the possible collisions of the robot-arm with its environments and with itself.
- *Feedback options*: the application gives the user two options for performing the feedback of a teleoperation: an online video stream and graphical updating of the 3D simulation with the current position of the real robot.
- *Augmented Reality*: the real information from the robot scenario is complemented with some virtually generated data from the virtual environment (Fig. 9). Virtual projection is combined with the current state from the remote laboratory taking into account current IP camera setting and the 3D environment. This feature helps to improve user performance and provides more information to control the robot.

V. EXAMPLES OF EXPERIMENTS

In this section, two experimentation examples about the powerful of the virtual and remote laboratory developed, will be shown. One of them about the off-line-programming tool of the virtual environment, and the another one about the remote capabilities of the application.

Off-line-programming allows users to develop Java routines in the simulation. They can create variables, mathematical operations and objects in Java language. In addition, a Java library included in the applet provides classes and methods in order to move the virtual robot. Thus, a student can program tasks in the virtual environment using this library. For a detailed description of the available classes and methods for off-line-programming, see Table I.

The common structure that students must follow to program an off-line task is the following:

- Declaration of the positions (joint or Cartesian values) and time in form of double variables and arrays.
- Creation of the positions objects by means of classes *posJ* and *posC*. The parameters for these objects are the double variables and arrays initialized before.
- Declaration of the movement commands. Users can use the methods *moveJ* for joint movements or *moveC* for Cartesian movements. The parameters for these methods are the objects *posJ* and *posC* created before.

TABLE I.
CLASSES AND METHODS OF THE PROGRAMMING JAVA LIBRARY

Classes/Methods	Description
posJ (double[] position)	Class for define joint positions. Parameter: double array
posC (double[] position)	Class for define Cartesian positions. Parameter: double array
moveJ (String traj, posJ p, int grasp, double time)	Method for robot joint movements Parameters: type of trajectory, position, open/close grasp, time.
moveC (String traj, posC p, int grasp, double time)	Method for robot Cartesian movements Parameters: type of trajectory, position, open/close grasp, time
home_robot (int grasp)	Method to move the robot at home position Parameter: open/close grasp
open()	Open the grasp Parameter: -
close()	Close the grasp Paramenter: -
belt()	Switch on the conveyor belt until an object detection Paramenter: -

When a program is compiled and executed successfully, the trajectories simulated by the virtual robot are stored in the command list. This allows users to experiment remotely in the real robot with the path planning algorithm validated in the simulation. Next, a programming example of a pick-and-place task (subsection A) and a remote path planning execution (subsection B) will be described.

A. *Pick-and-place experiment*

The programming experiment consists of doing a pick-and-place operation of an object located in the conveyor belt. Figure 10 shows the corresponding program of this task. As mentioned before and it can be seen in Figure 10, there are three different parts in the program's structure: 1) declarations of the positions; 2) creation of the objects *posJ*; and 3) definition of the order movements. In these last methods, users can specify the trajectory that they want to use. In the example proposed, the task is performed by means of a 4-3-4 polynomial trajectory (parameter "434" in the method *moveJ*), a smooth robot movement with acceleration and speed continuity. This path planning divides the robot trajectory in three parts: actuator acceleration (a fourth-order equation), time of maximum actuator velocity (a third-order equation) and actuator deceleration (a fourth-order equation too). The temporal evolution of the robot joint position during the pick-and-place experiment is shown in Figure 12.

Figure 11 shows the states of the virtual robot during the execution of the pick-and-place experiment. The image sequence represents each of the joint positions programmed in the routine. Before the first position of the virtual robot, the conveyor belt is switched on to detect the object. This order correspond with the method "belt()" of the Java program (Fig. 10).

The grasped object has its respective mass and inertia. However, their real values are very low. In order to realize how the dynamic changes, the object's mass was increased to five kg. In this way, the temporal evolution of the effective torques will change when the virtual robot grasps the object. As it can be seen in the image sequence, the virtual robot transports the object in the trajectories 3, 4 and 5. Figure 12 shows how the robot's dynamics changes during this time. The robot torques increase considerably in comparison with the rest of the path planning due to the high object's mass and inertia.

B. *Remote path planning experiment*

This subsection shows the remote execution of a path planning experiment. As mentioned in Section IV, the teleoperation is based on high level order movements over the HTTPS protocol. Thus, path planning algorithms validated in the simulation will be able to be sent for remote execution in the real plant.

In order to ensure the correct use of the robot, the application applies a security system based on the following criteria:

- The *Robot Model* of the virtual lab (Fig. 3) checks that the trajectory simulated does not exceed the maximum velocity and acceleration allowed in the robot.
- The virtual environment validates that the trajectory simulated has not any singularity during its execution.
- The *Teleoperation Server* software checks that the trajectory sent to the real robot does not have any collision with other objects from the workspace or with itself.

In this way, after simulating a path planning algorithm validated in the virtual lab, users are able to execute it remotely.

The experiment proposed consists of a synchronous trajectory of four seconds long. Figure 13 shows the states of the real robot together with the user interface during the execution of the path planning experiment. The right image represents the IP camera video stream, and the left image shows the graphical updating of the 3D simulation.

As it can be seen in the image sequence, 3D graphical updating is slightly delayed regarding the real image. This fact is due to feedback data from the current position of the real robot is performed through HTTPS requests and this protocol is always put down at some delay time.

Despite the above mentioned, authors have assessed communication delays to evaluate the quality of this educational tool. The remote experiment was performed with a PC located in the same province by means a common Internet connection. The system was able to receive ten different joint coordinate values from the path planning teleoperated in the real robot, a synchronous trajectory of four seconds long.

```
void main () {
 double[] p1={26.5, 77.3, 28.4, -9.7, -63.9};
 double[] p2={26.5, 96.2, 12.3, -12.5, -63.9};
 double[] p3={26.5, 64.2, 26.4, 4.5, -63.9};
 double[] p4={95.3, 54.5, 26.4, 4.5, -63.9};
 double[] p5={95.3, 65.7, 26.4, 4.5,- 63.9};

 posJ pos1 = new posJ(p1);
 posJ pos2 = new posJ(p2);
 posJ pos3 = new posJ(p3);
 posJ pos4 = new posJ(p4);
 posJ pos5 = new posJ(p5);

 belt();
 moveJ("434",pos1,-1,0.1,0.1);
 moveJ("434",pos2,0,0.1,0.1);
 moveJ("434",pos3,1,0.1,0.1);
 moveJ("434",pos4,-1,0.1,0.1);
 moveJ("434",pos5,-1,0.1,0.1);
 home_robot(0);
}//End
```

DECLARATION OF THE POSITIONS

CREATION OF THE OBJECTS posJ

ORDER MOVEMENTS

Figure 10. Off-line-program of the pick-and-place experiment

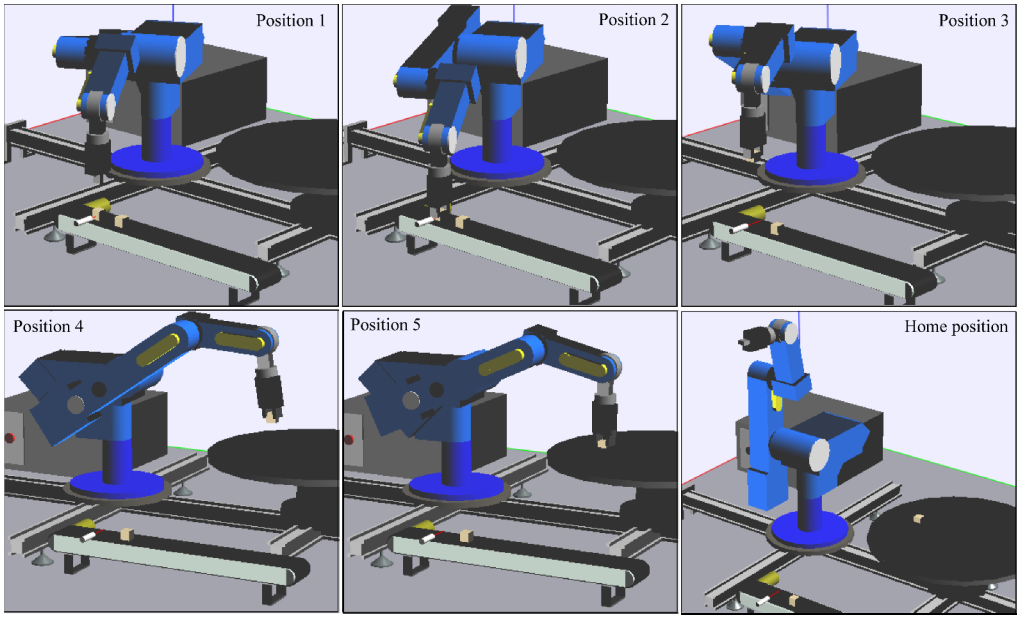

Figure 11. States of the virtual robot during the execution of the pick-and-place experiment

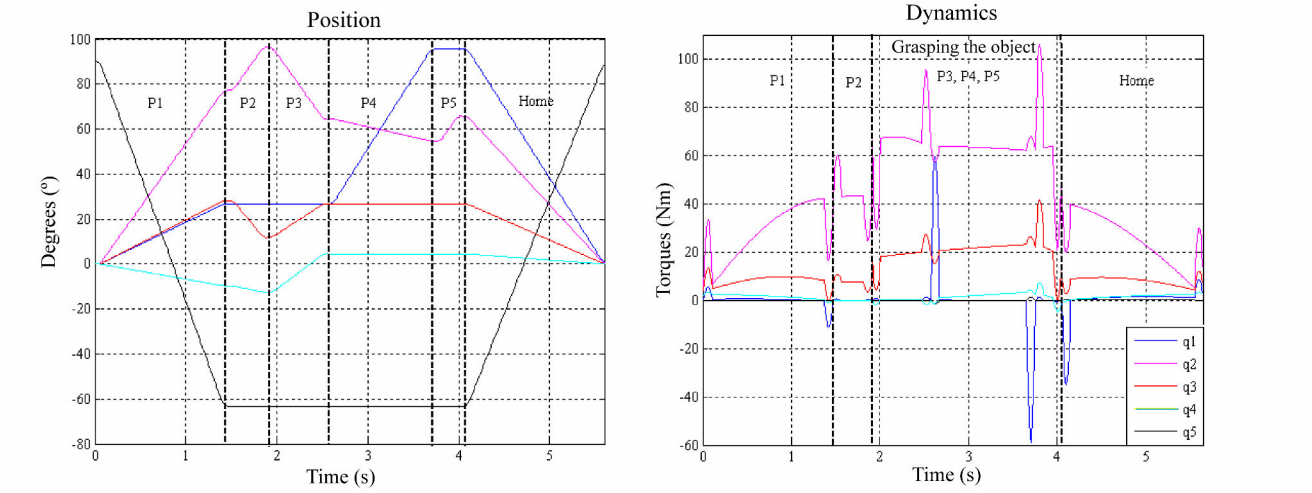

Figure 12. Temporal evolution of the joint position and torques

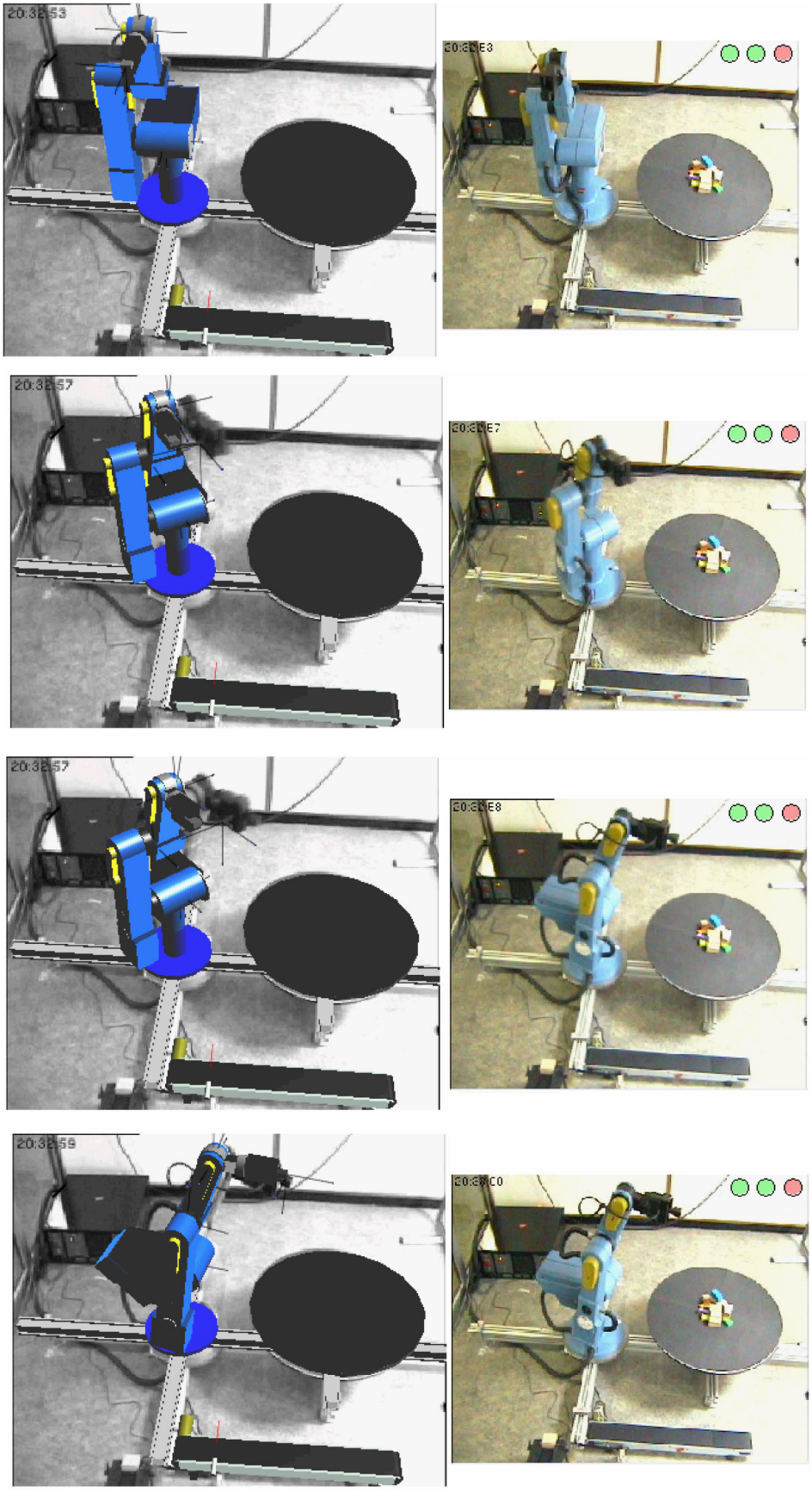

Figure 13. Remote path planning experiment using augmented reality

VI. CONCLUSIONS AND CURRENT LINES OF WORK

In this paper, the virtual and remote laboratory RobUaLab.ejs, for the simulation and teleoperation of an industrial robot arm, has been presented. Our system is mainly oriented towards the training and e-learning of robotic concepts.

The application has been developed using EJS, an open-source tool designed for the creation of interactive simulations. In this way, the procedure to transform the robotic system in an interactive virtual laboratory has been easier to do than the majority of programs available. It has not been necessary to learn specific programming skills and a big investment of time has not been needed to create the application.

With the virtual lab developed, students can learn robotic concepts such as direct/inverse kinematic, path planning, dynamics and programming. The user interface is very user-friendly, and the graphical simulation very realistic.

The remote capabilities of the application allow users to experiment with real equipment. Remote experimentation of high level tasks based on AR encourages students to learn robotic concepts and provides them with a realistic hand-on experience.

Finally, the system presented collects a lot of interesting virtual and remote features (complete robot simulation, robot dynamics, remote power and robot control, augmented reality, etc.), which are difficult to find together in a free Java applet like this. The advanced features which contains the application only are usually available in professional or specific software tools.

At present, new features are being incorporated into the virtual laboratory, such as a system for 3D recognition and modeling of the objects that the robot arm handles. The virtual environment only allows users to pick and place operations with virtual objects introduced from the applet (see subsection III.B). However, it is very interesting that users can also teleoperate the handling of objects with the robot arm. In this way, the virtual environment will represent the objects from the real workspace.

Figure 14 shows the recognition algorithm proposed. The Java applet will recognize real objects from the IP camera images. Therefore, pick-and-place operations programmed in the virtual lab will be able to be performed in the real plant.

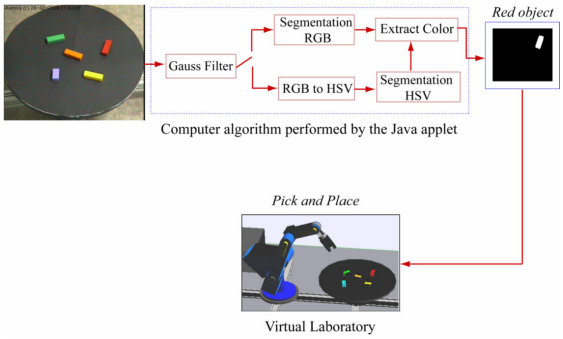

Figure 14. Computer vision algorithm to recognize real objects

ACKNOWLEDGMENT

Authors of this document would like to thank Pr. Sebatián Dormido from UNED University, the main manager of the AutoMatLab project, Dr. Francisco Esquembre from University of Murcia, the creator of EJS, and the "Ministerio de Educación y Ciencia" of the Spanish Government for its financial support.

REFERENCES

[1] Augmented Reality Interface for Telerobot Application Via Internet, http://lsc.cemif.univ-every.fr:8080/Projects/ARITI .

[2] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz, "A multimodal interface to control a robot arm via the web: A case study on remote programming," *IEEE Transactions on Industrial Electronics*, vol. 52, pp. 1506-1520, Dec. 2005. (doi:10.1109/TIE.2005.858733)

[3] F. A., Candelas, C.A. Jara and F. Torres, "Flexible virtual and remote laboratory for teaching Robotics", *4th Int. Conf. on Multimedia and Information & Communication Technologies in Education*, vol. 3, pp. 1959-1963, Nov. 2006.

[4] K. Kosuge, J. Kikuchi and K. Takeo, VISIT: A teleoperation system via the computer network, *Beyond Webcams: an introduction to online robots*, MIT Press, Cambridge, 2002.

[5] F. Esquembre, "Easy Java Simulations: a software tool to create scientific simulations in Java", *Computer Physics Communications*, vol. 156, pp. 199-204, Jan. 2004. (doi:10.1016/S0010-4655(03)00440-5)

[6] D. Gillet, N. Anh, and Y. Rekik, "Collaborative web-based experimentation in flexible engineering education". *IEEE Transactions on Education*, 48, 696-704, Nov. 2005. (doi:10.1109/TE.2005.852592)

AUTHORS

Carlos A. Jara is with the Department of Physics, System Engineering and Signal Theory, University of Alicante, Carretera de San Vicente del Raspeig, s/n, 03690 San Vicente del Raspeig, Alicante, Spain. Email: cajbdfists.ua.es.

Francisco A. Candelas is with the Department of Physics, System Engineering and Signal Theory, University of Alicante, Carretera de San Vicente del Raspeig, s/n, 03690 San Vicente del Raspeig, Alicante, Spain. Email: francisco.candelas@ua.es.

Fernando Torres is with the Department of Physics, System Engineering and Signal Theory, University of Alicante, Carretera de San Vicente del Raspeig, s/n, 03690 San Vicente del Raspeig, Alicante Spain. Email: fernando.torres@ua.es.

This work was supported in part by the research project DPI2005-0622 and the FPI grant BES-2006-12856.

Manuscript received 19 June 2008. Published as submitted by the authors.