A Multimodal Interface to Control a Robot Arm via Web: A Case Study on Remote Programming

R. Marín, *Member*, *IEEE*, P. J. Sanz, *Member*, *IEEE,* P. Nebot, and R. Wirz
Department of Computer Science & Engineering. University of Jaume I
E-12006 Castelló - SPAIN
{rmarin, sanzp, pnebot}@uji.es

Abstract

In this paper, we present the user interface and system architecture of an Internet-Based Tele-Laboratory, which allows researchers and students to control and program remotely two educational online robots. In fact, the challenge has been to demonstrate that remote programming combined with an advanced multimedia user interface for remote control is very suitable, flexible and profitable for the design of a Tele-Laboratory. The user interface has been designed by using techniques based on augmented reality and non-immersive virtual reality, which enhance the way operators get/put information from/to the robotic scenario. Moreover, the user interface provides the possibility of letting the operator manipulate the remote environment by using multiple ways of interaction (i.e. from the simplification of the natural language to low level remote programming). In fact, the paper focuses on the lowest level of interaction between the operator and the robot, which is the remote programming. As explained in the paper, the system architecture permits any external program (i.e. remote experiment, speech recognition module, etc.) to have access to almost every feature of the Tele-Laboratory (e.g. cameras, object recognition, robot control, etc.). The system validation has been performed by letting 40 Ph.D students within the "EURON Summer School on Internet and Online Robots for Telemanipulation" workshop (Benicàssim, Spain, 2003) to program several telemanipulation experiments with the Tele-Laboratory. Some of these experiments are shown and explained in detail. Finally, the paper focuses on the analysis of the network performance for the proposed architecture (i.e. time delay). In fact several configurations are tested through various networking protocols (i.e. RMI, TCP/IP, UDP/IP). Results show the real possibilities offered by these remote programming techniques, in order to design experiments that are intended to be performed from both, home and the campus itself.

Keywords: Robotics Tele-Lab, Remote Programming, Education & Training, Distributed Systems

1 Introduction

Online robots enable multiple users to have control over a remote robotic system in a teleoperated manner. These kinds of systems are very useful for teaching robotics, due to the fact that they enable the general public (e.g. students) to gain experience of robotics technology directly [Marin et al., 2002a]. In fact, the next step is giving more freedom to the operator, and not only to enable remote control of the robotic system, but also remote programming. To accomplish this, the user interface design is fundamental and techniques like virtual and augmented reality provides a good help to improve the way students and scientist program remotely the Tele-Laboratory.

Enabling remote programming of the robot system permits the operator to develop external programs that take control over the whole set of robotic functions. Thus, for example, we could design an experiment in Java for performing a visual servoing manipulation, or we could even use this interface for designing a voice-operated robot. In fact, this is what we have done for the Telelab.

The interest for the design of Internet-based Tele-Laboratories is increasing enormously, and this technique is still very new. A very good example of already existing experiments in this area can be found in [McKee, 2002].

1.1 System overview

In our case, the robot scenario is appreciated in Figure 1, where three cameras are presented: one taking images from the top of the scene, a second camera from the side, and a third camera from the front. The first camera is calibrated, and used as input to the automatic object recognition module and 3D-model construction. The other two cameras give different points of view to the user when a teleoperation mode is necessary in order to accomplish a difficult task (e.g. manipulating overlapped objects).

Once the user gets an online connection to the robot, the manipulator goes to the initial position, so the whole scene information is accessible from the top camera. At that moment, the computer vision module calculates a set of mathematical features that identify every object on the board [Marin et al., 2002b]. Afterwards, the contour information is used in order to determine every stable grasp associated with each object [Sanz et al., 1998], which is necessary for vision-based autonomous manipulation. This process is a must whether high level task specification from a user is required, and also in order to diminish network latency.

Meanwhile, we shall use the output from these visual algorithms in order to construct a complete 3D model of the robot scenario (i.e. the objects and the robot arm). Thus, users will be able to interact with the model in a predictive manner, and then, once the operator confirms the task, the real robot will execute the action.

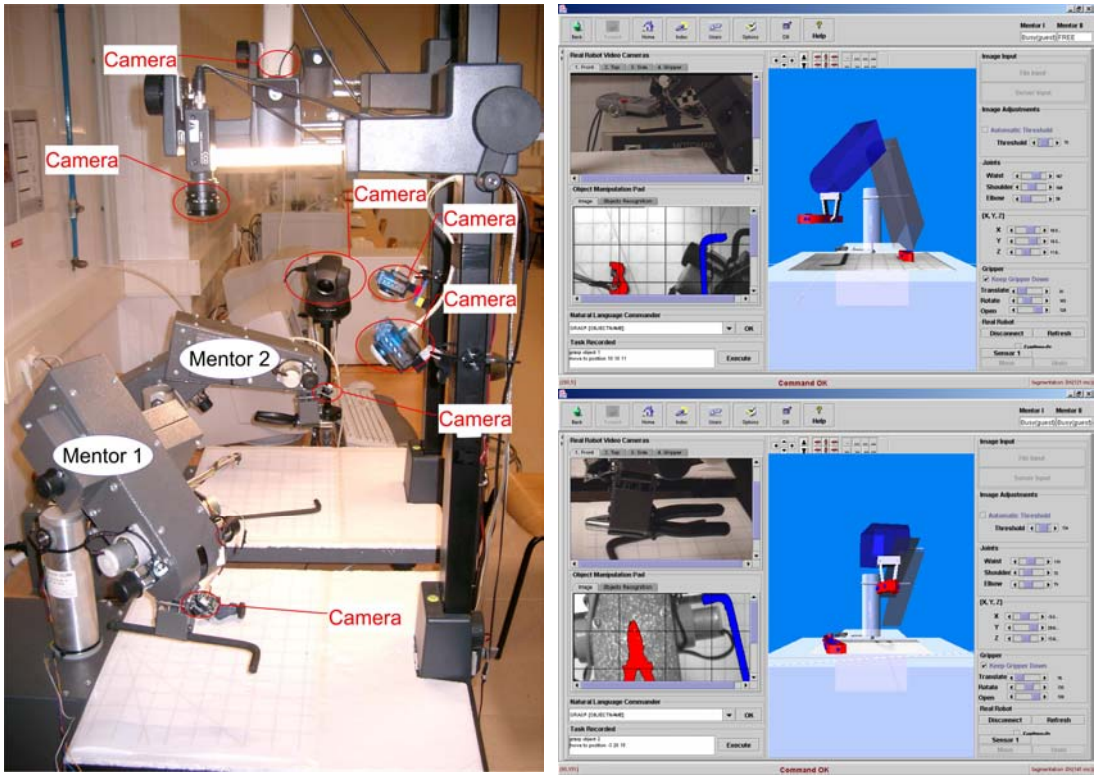


Figure 1. Tele-Laboratory experimental setup (left); and its user interface associated (right).

2 The Telerobotic Interface: A Comparative Study

One of the essential points in the design of a web based telerobotic system is the definition of the Human-Robot interaction. In most telerobotic systems, user interaction is still very computer-oriented, since input to the robot is accomplished by filling in forms or selecting commands from a panel. Very little attention has been paid to more natural ways of communication such as natural language, or gestures.

Another essential point is making the robot able to learn from the operator and from practice. It means the system's robustness and effectiveness increases as time goes by as the process is performed again and again.

2.1 The UJI Robotics Telelab

The multiple interaction channels between the user and the robot system can be appreciated in Fig 2. In this figure a pyramidal representation indicates that the complexity degree in the communication with the robot is decreasing from the bottom to the top. It means that at the bottom the complexity is at a maximum for the user and on the top (i.e. the ideal case in which the natural language is available) is at a minimum.

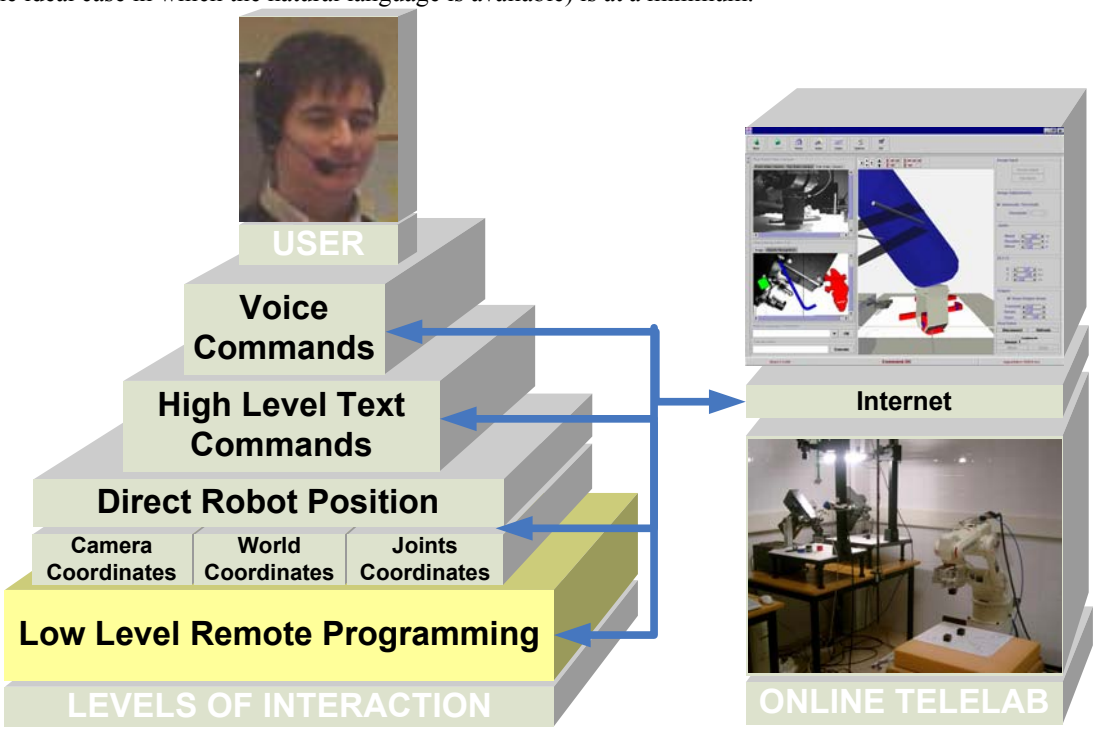


Figure 2. Different interaction channels enable to users selecting the more suitable interaction with the robot.

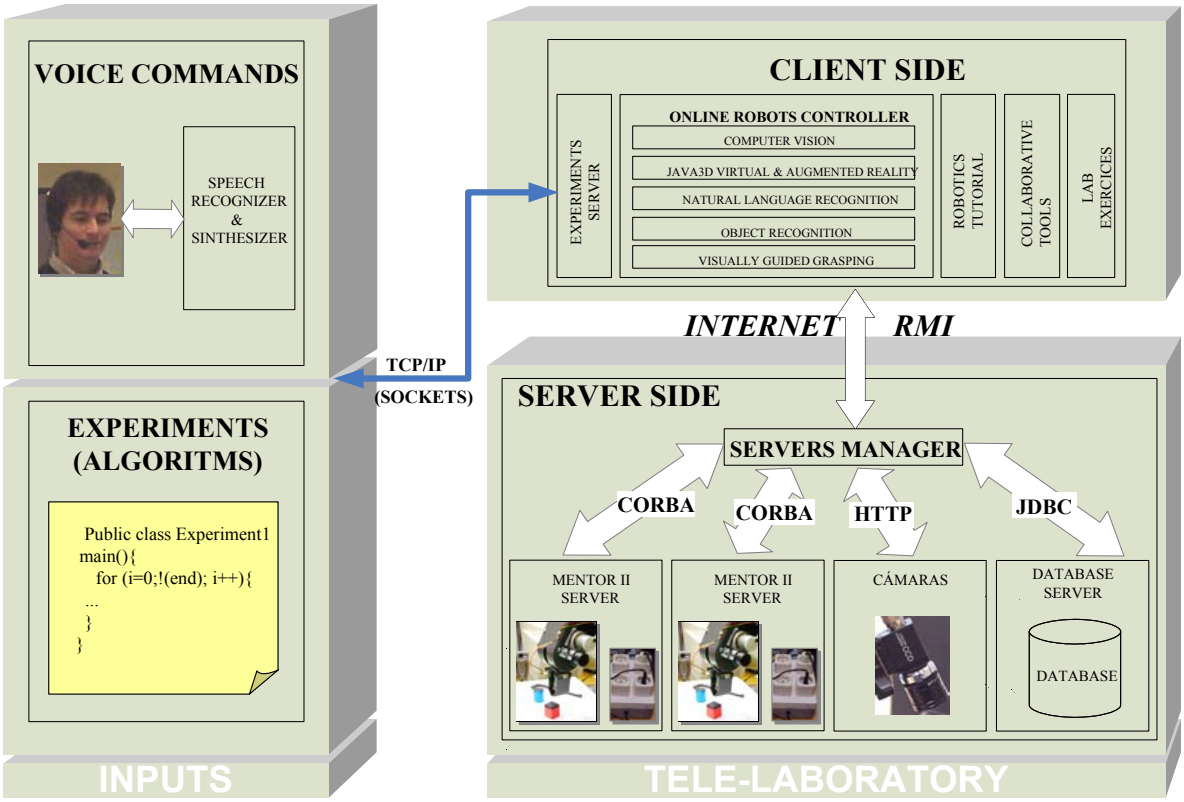


Figure 3. The UJI Robotics Tele-Laboratory architecture

In particular, as we can see in Figure 3, the Tele-Lab accepts experiments (i.e. algorithms) as inputs using any programming language capable of managing TCP/IP sockets. We already provide a Java library for using the robot in a simple manner. Future efforts will be oriented to facilitate this library not only in Java but also with other programming languages (i.e. Matlab, C, etc.). The outputs of the experiments are returned to the user by means of the Tele-Laboratory "Client Side" user interface, which permits the operator to see the results of their programmed actions directly on a multimedia Java3D user interface.

2.2 Some Issues on Telerobotic Interfaces

As it is well known, a Telerobotic System consists of 3 main parts: (1) the remote robot, (2) the communication network, and (3) the User Interface. In our situation, in order to compare the UJI Tele-Lab with other online robots, we are going to focus on the user interface, which is our main interest area. Note the remote robot depends on the current application, and the communication network is assumed will be always the same (i.e. the Internet).

First of all we are going to enumerate the parameters that, in our opinion, will make clear the elaboration of the posterior comparative analysis:

- Internet based versus Web based: Although every Telerobotic system included in the comparative analysis uses the Internet as the connection medium, the ones that are really interesting for us are the online robots, which means their remote control can be accomplished using the World Wide Web. For a list of the advantages of using the Web refer to [Goldberg & Siegwart, 2001].

- Predictive display: The use of predictive displays helps to diminish the Internet latency and delay effects. Besides this, it can help with the implementation of virtual task specification environments [Marin et al., 2002c].

- Augmented reality: By using Augmented reality techniques the user information on the client side is enhanced enormously. It obviously helps the robot programming and monitoring [Marin et al., 2002a].

- Virtual reality (3D model): By using 3D-model construction from camera input the user is able to interact with the system from any point of view. Besides this, it is a very useful technique for implementing the predictive display functionality and the task specification environment [Marin et al., 2002c].

- Automatic Object Recognition: In order to specify tasks in a more natural manner, the automatic object recognition module is very helpful in those situations were the environment is structured enough to perform the operation in a reasonable time. Moreover, by defining robotic tasks by using object names instead of object numbers implies the operations are going to be invariant to the initial position, which enhances a lot the system performance and robustness [Marin et al., 2002b].

- Incremental Learning: In our opinion a system should have the ability to learn from experience. It means the system is becoming more intelligent as time goes by. This learning feature should at least enhance the Automatic Object Recognition system performance [ICRA02b].

- Autonomous Grasping: One of the key modules in order to enable high level interaction is the Autonomous Grasping. By default, the Telerobotic system should calculate the best grasping alternatives for every object in the scene [Sanz et al., 1998].

- Very high level commands: In order to avoid the "cognitive fatigue" of the user when manipulating a scenario remotely, the idea is to allow the specification of very high level commands. It means the user is able to operate the robot in a more supervised manner, e.g. "grasp the pencil" [Marin et al., 2003].

- Voice Recognition: In our opinion it would be very interesting if the system allows the user to make these high level commands using a microphone. This means that the user is able to interact with the remote robot at a distance using the voice as the only input [Marin et al., 2002d].

- Off-line Programming: Another important situation to take into account when implementing an online robot is what to do when someone else is using the real manipulator. The Off-line programming technique allows the user to program the robot in a simulated environment, without needing to have physical control of the manipulator. It is very important in order to let people (e.g. students) work with the robot as long as possible [Marin et al., 2003].

- Tasks Specification: In our opinion it would be interesting to have a way of specifying tasks that have previously been recorded from another interaction. This capability could be used for increasing the robot possibilities (e.g. task planning).

- Multiple Users Management: Of course, an online robot should implement some criteria in order to allow multiple users to share the robot control. Initially, the FIFO (First In First Out) alternative could be a way of doing it, but is always interesting the possibility of associating priorities depending on the context.

- Remote Programming: This feature is in our opinion very interesting because means the system can be programmed remotely by using any programming language (i.e. Java, C. etc.). The advantages of having so are evident, and they mean the students and scientist can use the system to validate their algorithms in a particular area (e.g. computer vision, object recognition, visual servoing, control, etc.).

2.3 A Comparative Study

Bearing in mind all these properties, a comparative study among several web telerobotics systems focused on manipulation is presented in Table I. Please, note that only those systems available on-line and enabling the remote control of a robot arm will be considered. Nevertheless an exhaustive comparative, including other manipulation systems or other kind of robots (e.g. mobile robots, etc.), is out of the scope of this work. Moreover, it is worth remarking that some of the presented information has been acquired by asking the authors directly (i.e. K. Goldberg, etc.). Others have been obtained by revising proceedings, books, research magazines and by using them directly via Web. In the following a short description of each system showed in the comparative is presented.

(1) The Mercury Project is known as the first telerobotic system on the web. In August 1994, Ken Goldberg's team mounted a digital camera and air jet device on a robot arm so that anyone on the Internet could view and excavate for artifacts in a sandbox located in the Robotics laboratory at the University of Southern California

[Goldberg et al., 1995]. The primary goal was creating a very reliable system capable of operating 24 hours a day and surviving sabotage attempts. Moreover, the system had to be low in cost, because of budget limitations. The secondary goal was creating an attractive web site that encourages users to repeat their visits [Goldberg et al., 2001]. To facilitate use by a wide audience of non-specialists, all robot controls were made available via Mosaic, the only browser available at that time. The 2D interface matched the workspace of the robot, so users could move by clicking on the screen with a few buttons for out-of-plane effects. The Mercury Project was online for seven months starting in late August 1994. During this time it received over 87.700 user connections. An off-center air nozzle caused sand to accumulate in one side of the box, so the sand had to be periodically re-groomed. Other than a few random downtimes, the robot was online twenty-four hours a day.

(2) Telegarden. This telerobotic installation allows WWW users to view and interact with a remote garden filled with living plants. Members can plant, water, and monitor the progress of seedlings via the movements of an industrial robot arm. The Telegarden was developed at the University of Southern California and went online in June 1995, and it has been online since in August 1995 [Goldberg et al., 1996]. In its first year at USC, over 9000 members helped cultivate. In September 1996, the Telegarden was moved to the new Ars Electronica Center in Austria. This project uses an Adept-1 robot and a multitasking control structure so that many operators can be accommodated simultaneously [Telegarden-Website]. Users are presented with a simple interface that displays the garden from a top view, the garden from a global composite view and a navigation and information view in the form of a robot schematic. By clicking on any of the images one commands the robot to move to a new absolute location or one relative to where they just were. Whereas the Mercury Project required operators to wait up to 20 minutes on a queue for a 5-minute turn to control the robot, the Telegarden applies multi-task to allow "simultaneous" access. After X-Y coordinates are received from a client, a command is sent to move the robot arm, a color CCD camera takes an image, the image is compressed based on user options such as color resolution, lighting, and size, and then returned to the client.

(3) Australia's Telerobot on the Web. The University of Western Australia's Telerobot went online in late Sept 1994, enabling users to manipulate different kind of objects over a board, following a "Pick & Place" strategy [Australia's-Website]. The challenge with the interface design is to provide enough information to make interpretation easy while minimizing transmitted data and maintaining a simple layout. Ways of doing this include restricting the number of degrees of freedom that an operator controls as well as making use of Java and JavaScript technology [Taylor et al., 2000]. Remotely operate a 6-DOF ASEA manipulator located at the University of Western Australia. Believed to be the first remotely operated industrial robot on the Web.

(4) The ARITI (i.e. Augmented Reality Interface for Telerobotic applications via Internet) system presents a display interface enabling any person to remotely control a robot via the Web [Arity-Website]. A mixed perceptual concept based on a Virtual Reality (VR) and Augmented Reality (AR) technologies is part of the control user interface, allowing one to easily perform a task and describe the desired environment transformation that the robot has to perform.

(5) A very interesting system is the one reported by [Lloyd et al, 1997] at the University of British Columbia. A central problem in model-based telerobotic technology is obtaining and maintaining accurate models of the remote site. The system facilitates this using a fast gray-scale vision system at the remote site, which can recognize objects of known type and return their spatial positions to the operator. Basically, it takes a 2D image and finds edges for every object on the scene. Computing geometrical properties of those edges the system is able to recognize an object as belonging to a particular class. This work focuses on 3D model construction from vision, enabling the previous simulated interaction through an Internet connection (not web based). The UJI Online Robot offers a model 3D construction from vision too, which has the special feature of allowing the manipulation of not only known, but also unknown objects, thanks to the inclusion of a grasp determination and execution algorithm. Besides this, it allows many other advantages such as interacting by using voice as input.

(6) Another project to be taken into account is VISIT, which uses the predictive approach in order to communicate user interface and robot with existing time delay [Kosuge et al., 2001]. The system is designed based on advanced robotic technologies, such as computer vision, advanced motion control and so forth, so that the operator can execute a task in a remote site by utilizing a simple mouse. The computer vision module extracts object characteristics from a camera input, in order to incorporate them to the predictive model. No object recognition is presented.

Table I. A comparative study among different telerobotics systems currently available on the web.

	Mercury Project	Telegarden	Australia's Telerobot	ARITY	British Columbia	VISIT	UJI Robotics Telelab
Internet based	•	•	•	•	•	•	•
Web based	•	•	•	•			•
Predictive Display					•	•	•
Augmented Reality			•	•			•
Virtual reality (3D model)				•	•		•
Automatic OR					•		•
OR under occlusions					•		
Incremental Learning							•
Autonomous Grasping							•
Very high level commands							•
Voice Recognition							•
Off-line Programming				•			•
Tasks Specification				•		•	•
Multiple users management	•	•	•	•			•
Remote Programming							•

3 Remote Programming: The "Experiments" Java Library

In this section we are going to explain the low level details of the remote programming capability.

As seen in Figure 3, the "Experiments Server" module provides a TCP/IP interface to remote applications (i.e. experiments) that can take control over most of the Tele-Laboratory features (i.e. cameras, object recognition, robot movements, etc.). The "Experiments Server" maintains an open TCP/IP socket over a well-known port (i.e. 7745). It is at this socket where the external applications must be connected to be able to communicate with the Tele-Laboratory. Moreover, the "Experiments Server" is responsible for receiving the commands and the data from the applications (e.g. students' experiments), as well as sending back to the clients the results and evaluation of the execution of these actions on the Tele-Laboratory. Just note this TCP/IP interface can be used by any programming language that provides a standard socket library. In fact, a simple "telnet" command could be applied over this port in order to test and validate the "Experiments Server" functionality.

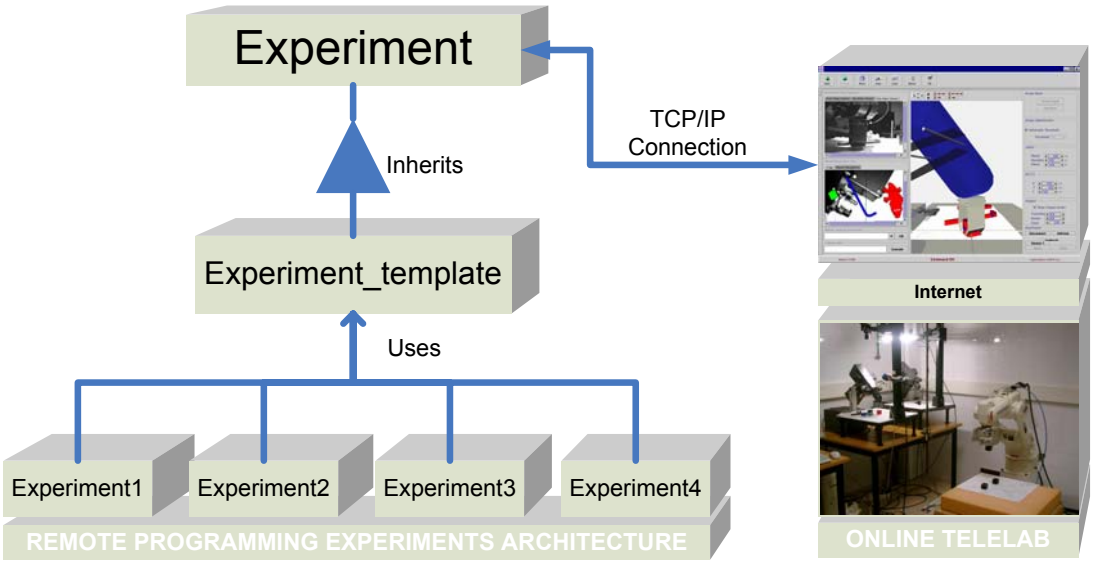


Figure 4. The "Experiment" Java Library Software Architecture

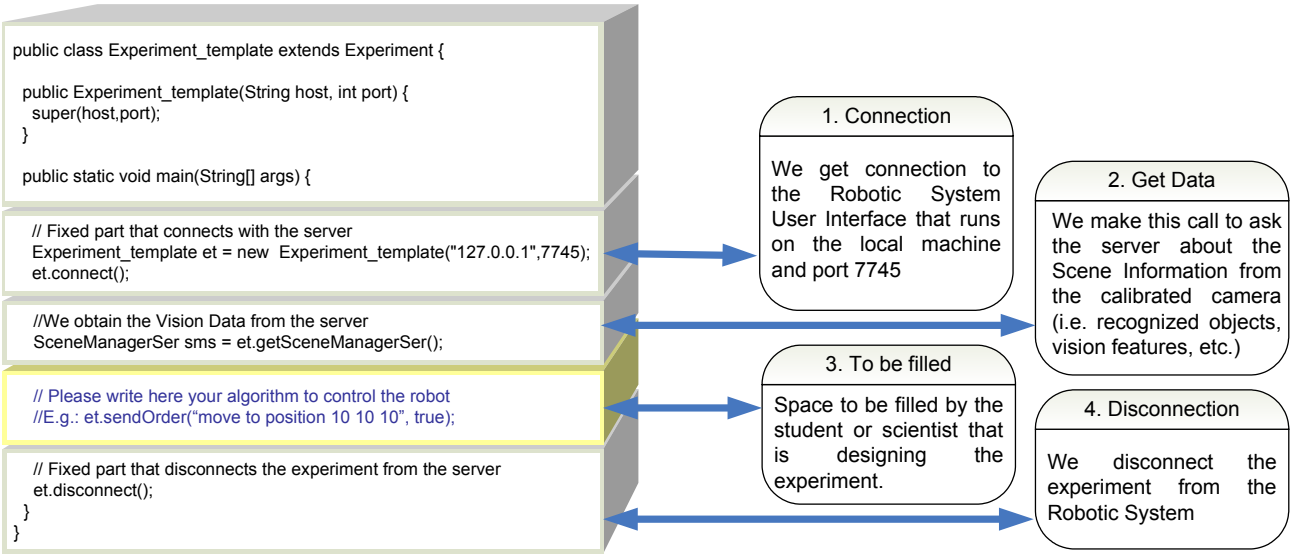


Figure 5. Template offered to the scientist/students as basis to perform a remote programming experiment with the robotic system.

In order to help students to use this feature, an "Experiment" Java Library has been implemented that takes care of all the TCP/IP communication details between the users' programs and the Tele-Lab Experiment Server. The "Experiment" Java Library has the benefit of giving the user the possibility of having a copy of the object instances that are running in the Tele-Laboratory itself. To do that, it is necessary that these Java objects be "serialized" (i.e. the object can be transformed into a chain of bytes). To allow this, Java has the "serializable" class, as a way of not only send strings through a socket, but also object's instances. Again, if the object to be transferred by the socket contains attributes, every one of them must be serializable too. If they are not, the object could not be transferred.

The "Experiment" Java Library consists of a class/library and a skeleton for the accomplishment of experiments in the Telelab. The experiments (i.e. user algorithms) need to have information about the real robot scenario (i.e. camera inputs, object recognition, etc.). The whole set of data referring to robot vision are all stored in a class called "SceneManager". Therefore, the transfer of this object to the experiments will be necessary. To do that, the "Experiments" Java Library includes a method called "getSceneManagerSer()", which provides the user with the capability of getting a copy of the SceneManager object that has the information of the real robot environment.

As explained in Figure 5, once a student or a scientist wants to program an experiment by using the "Experiments" Library, he/she must create a Java class (e.g. Experiment1) that extends from the "Experiment" class. By doing this every aspect related to sockets programming and object serialization are properly encapsulated.

To facilitate even more the experiment implementation, an "Experiment_template" is provided that already inherits from the "Experiment" class and presents the structure of a typical Remote Programming experiment, which is: (1) extending the "Experiment" class, (2) creating an instance of the experiment, (3) calling the "getSceneManagerSer" to obtain the serialized objects from the TeleLab, (4) executing the corresponding actions on the Telelab, and (5) closing the connection (see Figure 6).

3.1 Available Remote Programming Commands

The "Experiments" Java Library provides the user with the following methods:

1. "***connect()***": It connects the experiment with the Tele-Laboratory through the TCP/IP interface.
2. "***disconnect()***": It disconnects the experiment from the Tele-Laboratory.
3. "***getSceneManagerSer()***": It retrieves the current computer vision data from the Tele-Lab. Information about the existing objects on the scene, their geometrical information as well as the result from the object recognition module are provided.
4. "***sendOrder(String command, Boolean prediction)***": It sends a text command to the Tele-Laboratory (e.g. "grasp the allen", "move to position 10 10 5", and "move up"). For a description of the available commands

see Table I. Moreover, if the "*prediction*" variable is true the command will only be executed by the predictive display provided with the Tele-Lab 3D virtual interface. Otherwise, the command will be executed for both, the predictive interface and the real robot too.

Table II. Robot commands available.

Command	Description
Grasp the {object name}	It executes the most stable grasping alternative on the object {object name}
Grasp the object {number}	It executes the most stable grasping alternative on object {n} numbered from top to down and from left to right in the top camera.
Grasp using grip {Grip number} {object name}	It executes the grip numbered as {Grip number} on the object {object name}
Grasp using grip {Grip number} object {n}	It executes the grip numbered as {Grip number} on the object {n} numbered from top to down and from left to right in the top camera.
Add Grasp P=({x1}, {y1}, {x2}, {y2}) to {object name}	It adds a new grip to the object {object name} by following the straight line defined by the intersection of the 2D image points {x1, y1} and {x2, y2} in camera coordinates (top camera).
Add Grasp P=({x1}, {y1}, {x2}, {y2}) to object {n}	It adds a new grip to the object {n} by following the straight line defined by the intersection of the 2D image points {x1, y1} and {x2, y2} in camera coordinates (top camera).
Add Grasp C=({c1}, {c2}) to object {object name}	It adds a new grip to the object {object name} by following the straight line defined by the intersection of the object's contour points {c1, c2}.
Add Grasp C=({c1}, {c2}) to object {n}	It adds a new grip to the object {n} by following the straight line defined by the intersection of the object's contour points {c1, c2}.
Ungrasp	Having an object grasped in the gripper, it places it at the current robot position.
Ungrasp at position {x} {y}	Having an object grasped in the gripper, it places it at position {x, y} in image coordinates
Ungrasp over the {object name}	Having an object grasped in the gripper, it places it on top of the {object name}
Ungrasp over the object {number}	Having an object grasped in the gripper, it places it on top of object {number}
Move ahead	It moves the TCP ahead 1 cm
Move down	It moves the TCP down 1 cm
Move left	It moves the TCP to the left 1 cm, taking the manipulation pad orientation as the reference
Move right	It moves the TCP to the right 1 cm, taking the manipulation pad orientation as the reference
Move up	It moves the TCP up 1 cm
Move to position {x} {y} {z}	It moves the TCP to the world coordinates position [x, y, z]
Move to the sector {number}	The scenario is divided into 16 sectors, which are numbered from 1 to 16. The command moves the robot to the {number} sector.
Rotate gripper to {± number}	It rotates the gripper {± number}%
Open gripper to {number}	It opens the gripper a {number}% of its capacity
Get Position	It returns the world coordinates position of the robot [x, y, z].
Get Joints	It returns the joints coordinates position of the robot [j1, j2, j3, j4, j5, j6, j7].
Pick the {object name} up	It grasps the object labelled as the {object name}, and then elevates it 5 cm over the board
Pick the object {number} up	It grasps object {number} and then elevates it over the board 5 cm (by default)
Place it at position {x} {y}	Having an object grasped, it places it at position {x, y} in image coordinates
Place it over the {object name}	Having an object grasped, it places it on top of the {object name}
Place it over the object {number}	Having an object grasped, it places it on top of object {number}
Refresh	It brings the robot to the initial position, captures the scene image, recognizes the objects, and constructs the 3D model.

4 Examples of Experiments: Education & Training Validation

In order to validate and evaluate the interest of remote programming for students and scientist, we proposed several exercises to 40 international Ph.D students that were participating at the "2003 EURON International

Summer School on Internet and Online Robots for Telemanipulation" (Benicàssim, Spain, 2003). At this section we are presenting some of the exercises that were implemented by this group of researchers.

4.1 Objects Attributes Experiment

The first example consists of designing a remote program that simply uses the "Experiment" library in order to obtain information about the objects that are currently on the robotics scenario (i.e. number of objects, area, etc.). In Figure 6 we can see in yellow the two single steps that should be performed to accomplish the experiment. First of all we obtain from the Tele-Lab the computer vision data related to the robot scenario. For example we can print the dimensions of the calibrated camera image. The second step gets the information for every object in the scene and shows their area. In Figure 7 can be seen the result of the execution of the experiment.

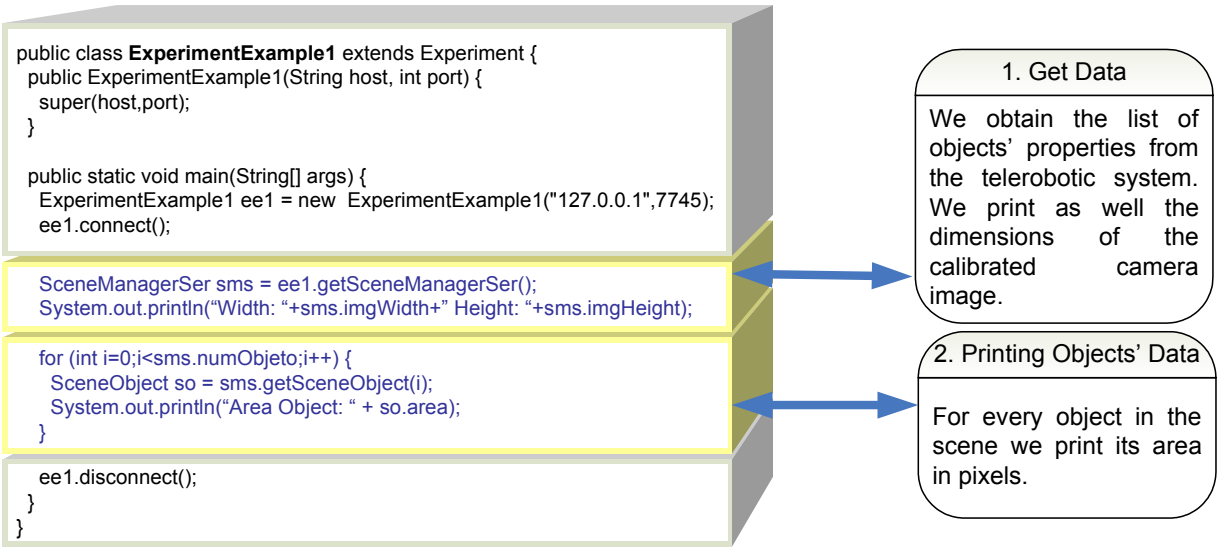


Figure 6. Remote Programming Algorithm for the "Objects Attributes" experiment.

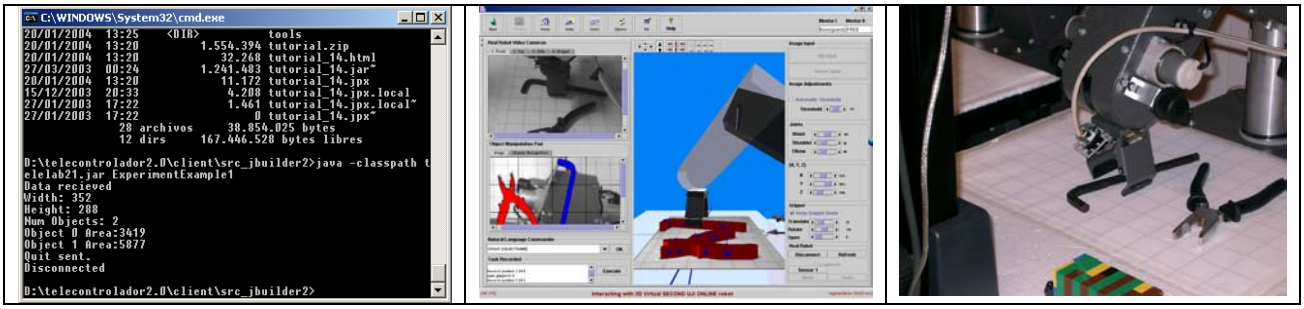


Figure 7. Execution of the "Objects Attributes" experiment in a single step: (left) remote experiment execution console, (center) Tele-Lab user Interface status, and (right) the real robot state.

4.2 Path Planning Experiment

In this second experiment the student uses the remote programming feature to bring the robot from a point to another (i.e. from [x1, y1, z1] to [x2, y2, z2]) by following a straight line. See Figures 7 and 8 for the algorithm and the execution output respectively.

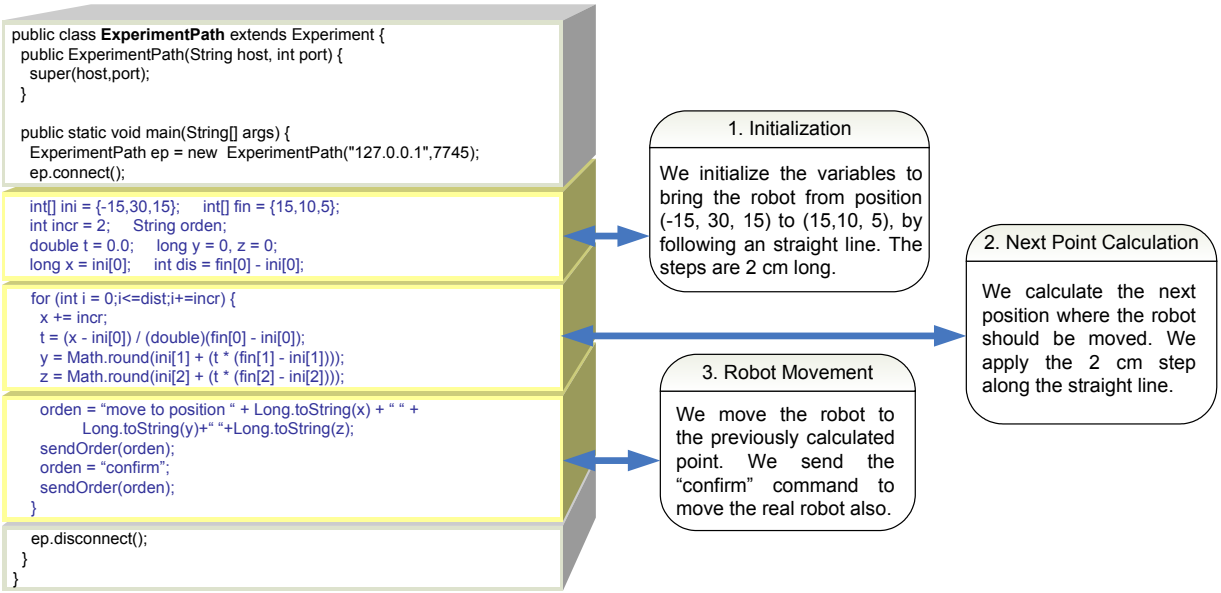


Figure 8. Remote Programming Algorithm for the "Path Planning" experiment.

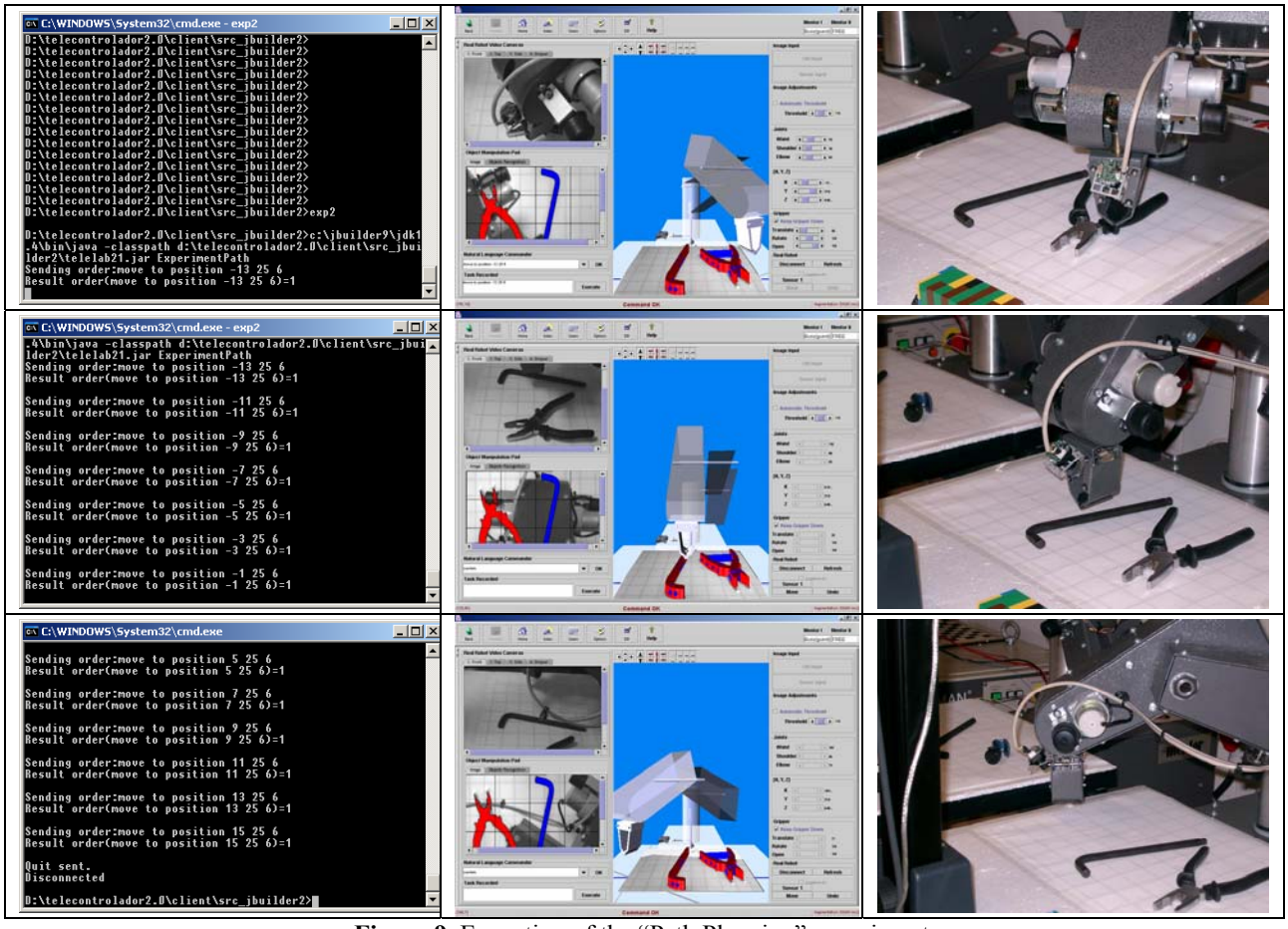


Figure 9. Execution of the "Path Planning" experiment.

4.3 Pick and Place Experiment I

The third experiment consists of initiating a grasping action over one of the objects in the scene, by using the most stable grasping points. In particular, we execute the action "grasp object 1" on the TeleLab, then we move the gripper to the quadrant 8 of the robot scenario, and finally we ungrasp the object by executing the command

"ungrasp". As can be seen in the following algorithm the parameter that we pass to the "*sendOrder"* command is *false*, which means we deactivate the prediction feature and the action is performed on the real robot.

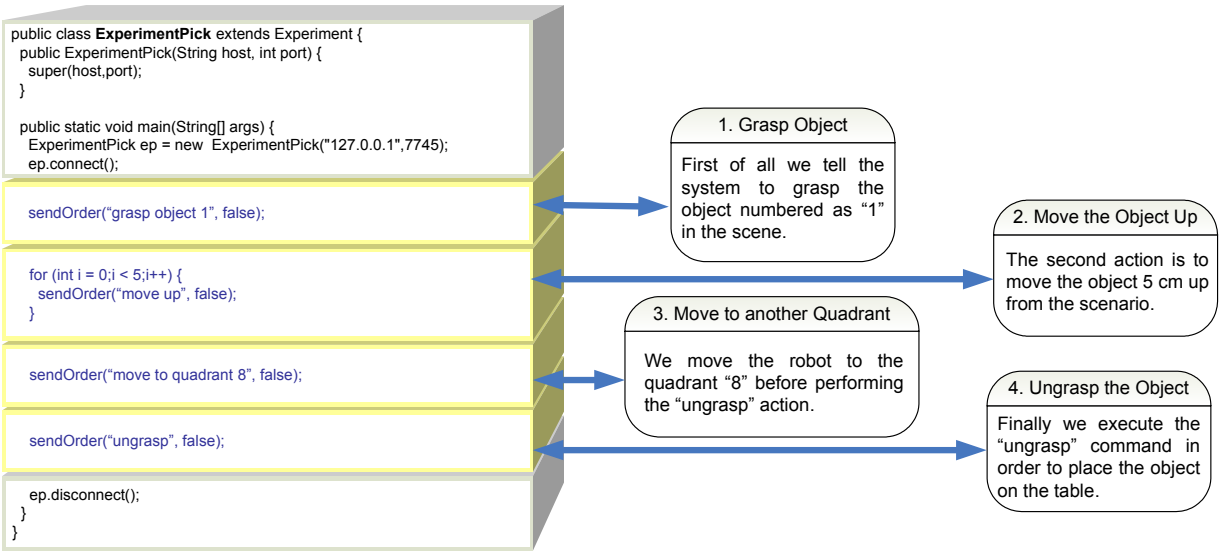
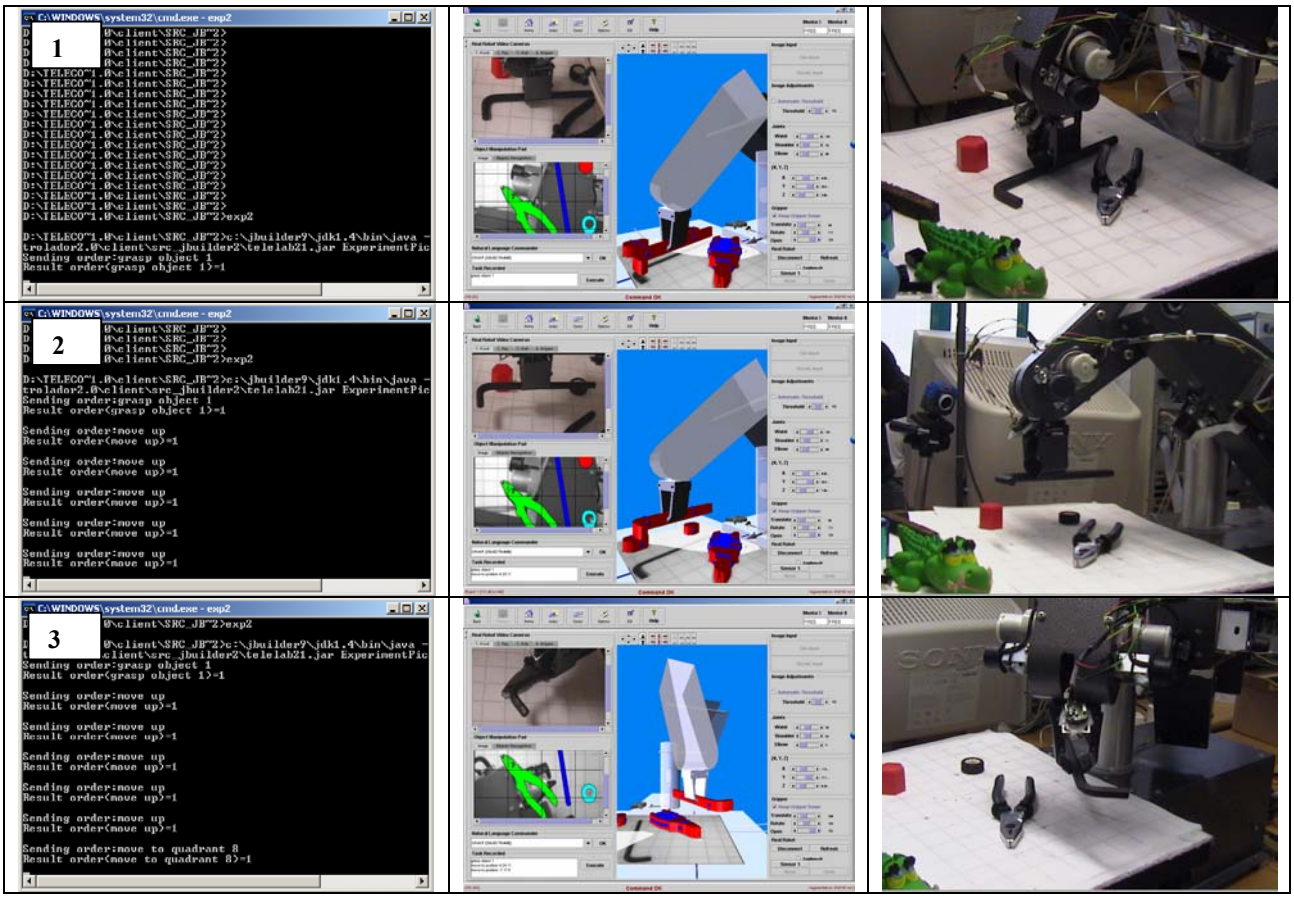


Figure 10. Remote Programming Algorithm for the "Pick and Place Experiment I".

In Figure 11 we can appreciate the states of the robot during the execution the Pick and Place Experiment I



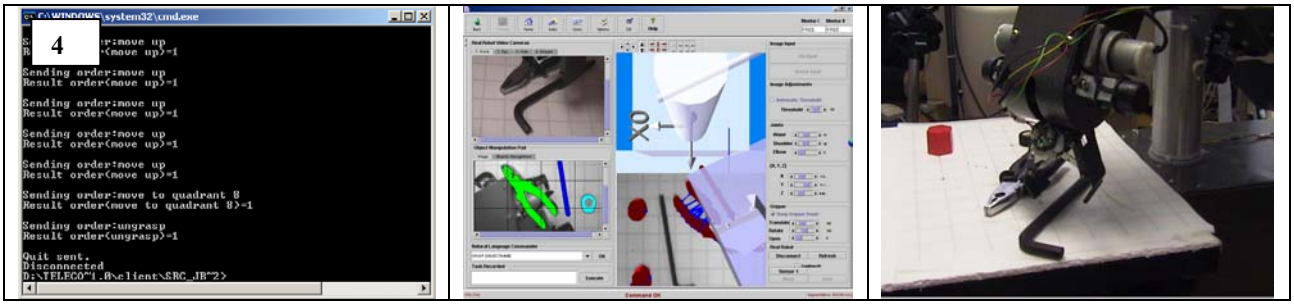


Figure 11. Snapshots of the educational robot when executing the Pick and Place Experiment I.

4.4 Pick and Place Experiment II

This second Pick and Place Experiment enables the user to execute a grasping action over two particular points of the object contour. For this example the two selected points are defined by the intersection of the Object's Maximum Inertia Axis and its contour. For a broad range of applications this alternative is sufficient.

As can be seen in the algorithm, in this situation we add a grasping to the object by using the object's attributes "p1" and "p2" that represents the contour points that intersect with the Maximum Inertia Axis. After that, we execute the grasping command on that object by using the already created grasping.

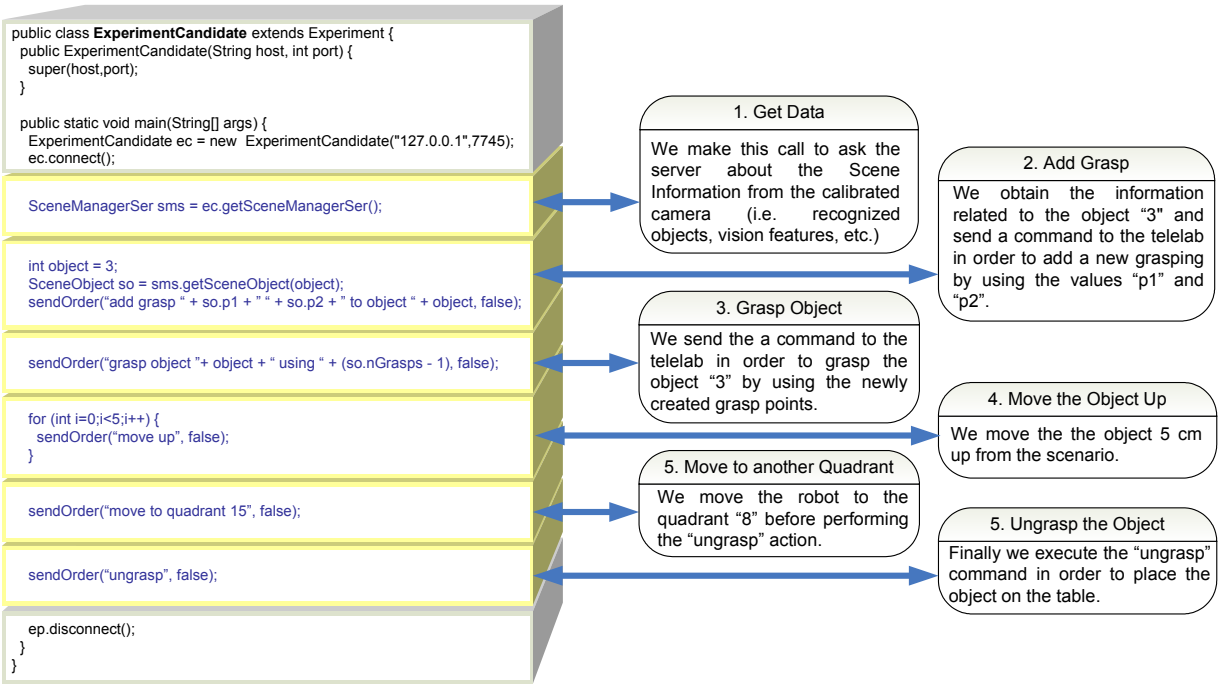


Figure 12. Remote Programming Algorithm for the "Pick and Place Experiment II".

4.5 Autonomous Robot Experiment

In order to validate the Telelab user interface and the system architecture itself we proposed to the students that they design a remote experiment for the control of a robot in a more autonomous manner. The exercise proposed was to leave the system waiting for a specific kind of object to appear in the robotic scenario, and once this occurs grasp the object and classify it into a particular position on the table. The autonomous control is especially related to the kind of industrial applications that classify objects coming along a conveyor belt using camera vision techniques. The solution proposed for this experiment can be seen in Figure 13.

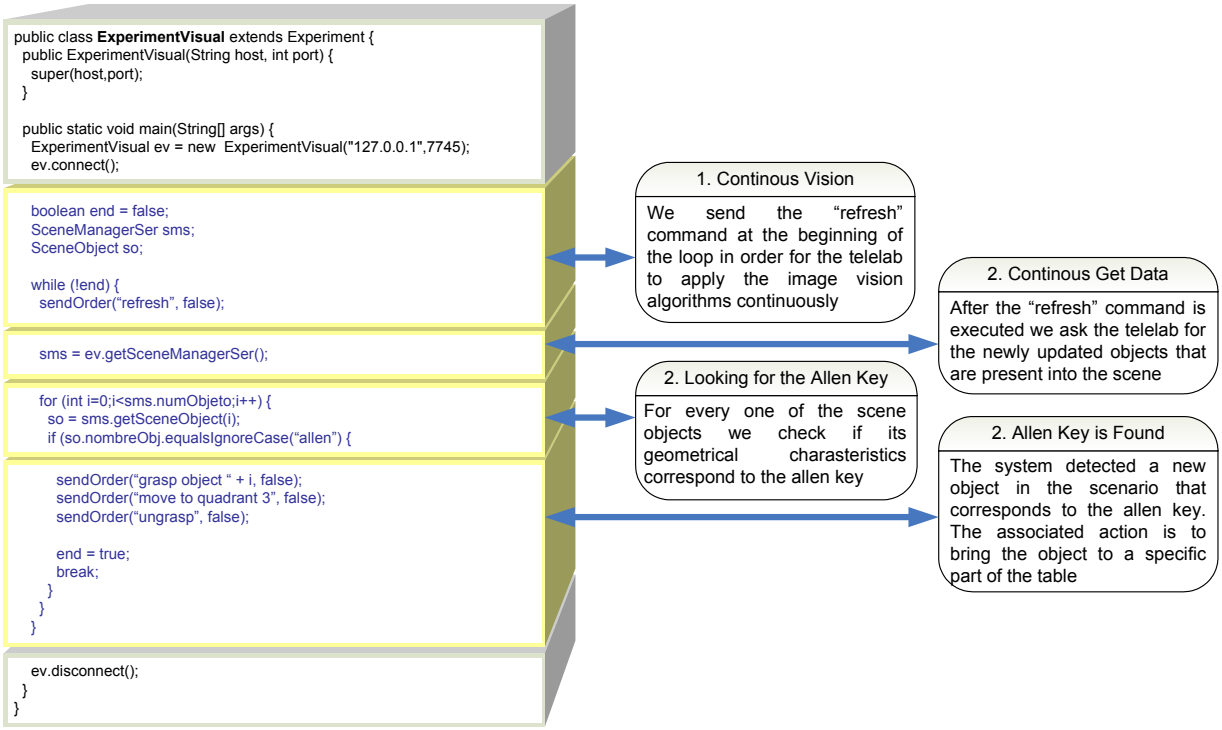


Figure 13. Remote Programming Algorithm for the "Autonomous Robot" experiment.

5 Results

In order to know how convenient the use of the Remote Programming Tele-Laboratory technique is, different configurations of the architecture have been tested. For every one of these configurations some experiments have been performed consisting of 400 remote movements on the robot following a square path. These experiments have been done by using TCP, UDP and RMI protocols. Performance details obtained of these experiments are given below.

The first configuration that has been tested is a configuration with a Unique Client/Server IP (UCS_IP). This configuration can be seen in Figure 14. Two different experiments have been implemented with this configuration, one using TCP protocol and the other one using UDP protocol.

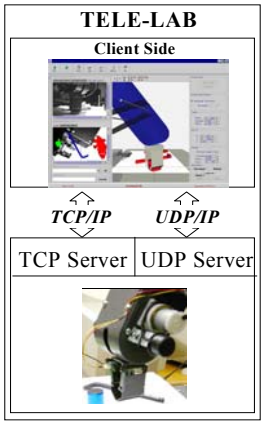


Figure 14. Unique Client/Server IP (UCS_IP) Software Architecture

In the Figure 15 it can be appreciated that the latency **on campus** for both, TCP and UDP protocols, is convenient enough to perform teleoperation experiments. Although there are some robot movements that have invested almost 14 msec in communicating with the server side, the average is less than 1 msec. Moreover, by looking at Figure 15 it can be seen that by teleoperating the robot from home (same city) using the TCP/IP approach, the operation is rapid enough for our purposes (less than 90 msecs of average).

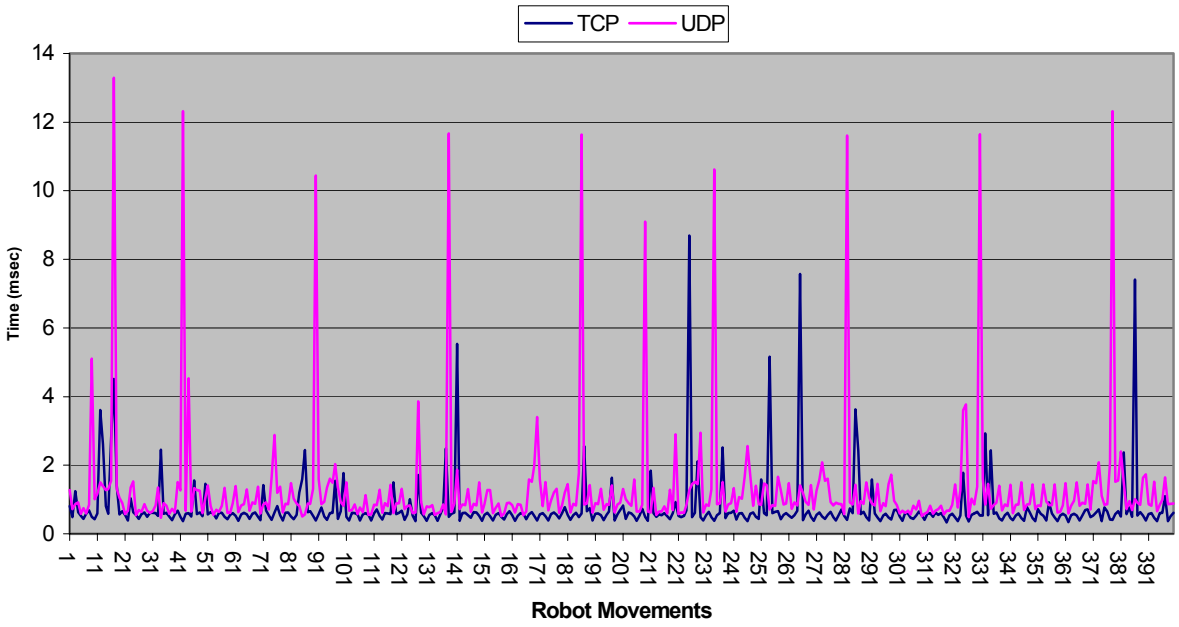


Figure 15. Graphical representation of the latency for the configuration Unique Client/Server IP (UCS_IP) **on Campus**, using both, the TCP and the UDP protocols

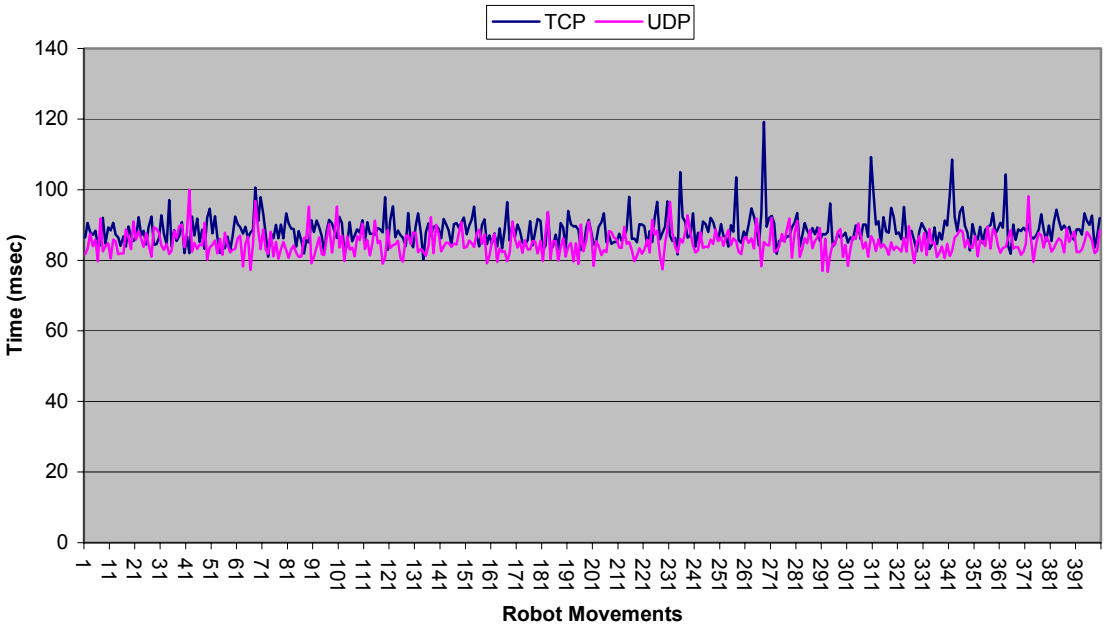


Figure 16. Graphical representation of the latency for the configuration Unique Client/Server IP (UCS_IP) **in the same City**, using both, the TCP and the UDP protocols

The next configuration, TeleLab Experiment RMI (TE_RMI), consists of using the TeleLab features in order for the students and scientists to program their own algorithms remotely. This means the experiment will connect first via TCP/IP on the local machine with the TeleLab user interface (Client Side), and then, once the operation is confirmed, the action is executed on the real robot via a RMI connection with the Servers Manager. Again, the Servers Manager connects via CORBA with the remote robot, as can be seen in Figure 17.

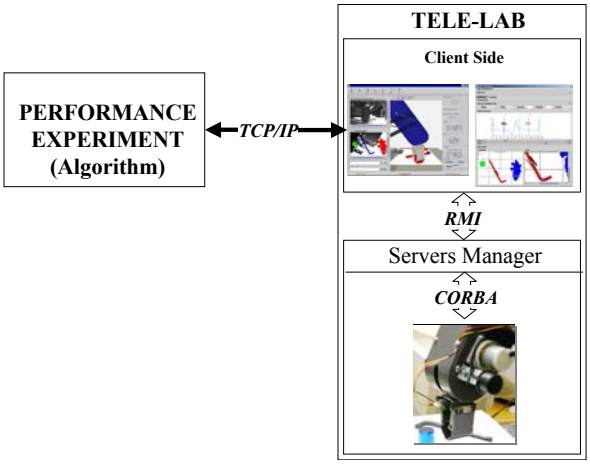


Figure 17. TeleLab Experiment RMI (TE_RMI) Software Architecture

It must be taken into account that once an operation is sent to the Client Side, this operation is executed virtually over the predictive interface, and then, once the action is confirmed by sending the command "confirm", this operation goes through the RMI and CORBA interfaces to move the real robot. It means this "predictive" configuration is very useful for students that are working off-line or just want to check the next robot movement in the virtual environment. And of course, it supposes the time latency is increased a lot. In fact, the average for the experiments executed on campus is 2770 msecs, and for those working from home is 3975 msecs.

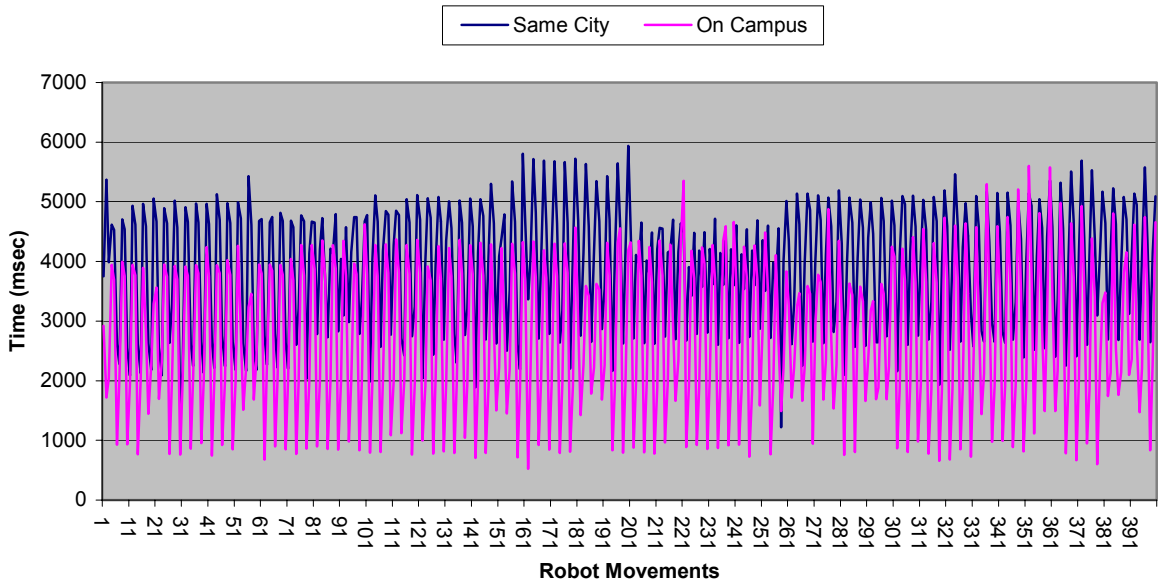


Figure 18. Graphical representation of the latency for the configuration TeleLab Experiment RMI (TL_RMI), using both on Campus and in the Same City configurations

As can be seen, the first configuration (UCS_IP) is the fastest one, and is the most convenient for traditional telerobotic applications. In fact the latencies shown in Table III justify this.

In summary, we have presented a Telelab Multirobot Architecture (RE_RMI) that permits any student or scientist not only to exercise robot control from a user interface, but also to control the manipulator from a Java program (remote programming). This second configuration is much more expensive because it requires a predictive display feature (i.e. requires confirmation of each command) and also it uses not only a TCP/IP protocol but three (i.e. TCP; RMI and CORBA).

Table III. Summary of the Time Latencies for every configuration experimented

Time Latency in msecs	UCS_IP_TCP On Campus	UCS_IP_UDP On Campus	UCS_IP_TCP Same City	UCS_IP_UDP Same City	TE_RMI On Campus	TE_RMI Same City
Average	0,56	0,97	88,16	86,68	2770	3975
Variance	0,74	2,98	17,66	11,17	1721131,97	1098603,41
Standard Deviation	0,86	1,73	4,20	3,34	1311,92	1048,14

6 Conclusions

One of the major difficulties associated with the design of online robots is the Internet latency and time delay. When using the Internet as a connectivity medium, it cannot be assured that a certain bandwidth is going to be available at a given moment. It means that once the information is sent from the server to the client (and vice versa), there are no guarantees that this data will reach its destiny in a minimum period of time. The delay associated with the Internet communications is unpredictable, so that the system design must take care of this situation, by means of giving more intelligence to the server side and using “predictive display” techniques.

Hence, the user interface has been designed to allow the operator to "predict" the robot movements, before sending the programmed commands to the real robot ("Predictive system"). This kind of interface has the special feature of saving network bandwidth and even being used as a whole task specification, off-line, programming, interface. By using a predictive virtual environment and giving more intelligence to the robot means a higher level of interaction, which avoids the “cognitive fatigue” associated with many teleoperated systems, and helps enormously to diminish the Internet latency and time delay effects.

Such a complex and friendly user interface would not be possible without the help of virtual and augmented reality techniques. With these two technologies the user is able to control almost every viewpoint of the real robot scenario, and then simulate the programmed tasks before sending it to the real manipulator. Moreover, automatic object recognition and autonomous grasping execution are crucial in order to provide such a level of user interaction.

The present paper presents an extension of the UJI Online Robot architecture that enables not only control a robot by interacting with an advanced user interface, but also letting the scientist and students program their own experiments by using the Java programming language. This has been possible thanks to the definition of the “Experiments” library, that allows in a encapsulated way to provide the Telelab clients with the serialized objects necessary to perform these action.

The researchers that have been using the Tele-Lab until now have found it very useful, due to the fact that it enables the operator to control the whole robotic system (cameras, object recognition, predictive interface, robot control, etc.) from a single library. It can be considered as a very good alternative for researchers performing a rapid prototyping of their algorithms by using a real robotic system accessible from any computer. Moreover, a pilot group of students have been using the “Experiments” library to design their own experiments. They became very enthusiastic and were motivated by the experience.

Future efforts will focus on the design of more advanced experiments for remote manipulation (e.g. remote visual-servoing, etc.), as well as the study of automatic methods for the evaluation of the experiments.

Acknowledgments

This is to acknowledge sources of financial support for this research, that was provided in part by the Spanish Ministry of Science and Technology (CICYT) under project CICYT-DPI2001-3801.

References

[Ariti-Website] Augmented Reality Interface for Telerobot Application via Internet (http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html).

[Australia's-Website] Australian Telerobot webpage (http://telerobot.mech.uwa.edu.au/).

[Goldberg et al., 1995] K. Goldberg, M. Mascha, S. Gentner, J. Rossman, N. Rothenberg, C. Sutter, and J. Widgley, "Beyond the Web: Manipulating the Real World", Computer Networks and ISDN Systems Journal 28, no 1 (December 1995).

[Goldberg et al., 1996] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, C. Sutter, and J. Wiegley. "A Telerobotic Garden on the World Wide Web". SPIE Robotics and Machine Perception Newsletter 5(1). Reprinted in SPIE OE Reports 150, June 1996.

[Goldberg & Siegwart, 2001] K. Goldberg, Roland Siegwart (ed.), "Beyond Web Cams: An introduction to Online Robots", MIT Press, Massachusetts, 2001.

[Kosuge et al., 2001] K. Kosuge, J. Kikuchi, and K. Takeo, "VISIT: A Teleoperation System via the Computer Network" Beyond webcams, and introduction to online robots, MIT Press, Massachussetts, 2001.

[Lloyd et al., 1997] J. E. Lloyd, J. S. Beis, D. K. Pai, D. G. Lowe. "Model-based Telerobotics with Vision". In proceedings of IEEE International Conference on Robotics and Automation (ICRA), p. 1297-1304, April 1997.

[Marin et al., 2002a] R. Marín, P.J. Sanz., J.S. Sanchez, A Very High Level Interface to Teleoperate a Robot via Web including Augmented Reality. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Washington, May, 2002.

[Marin et al., 2002b] R. Marín, J.S. Sanchez, P.J. Sanz. "Object Recognition and Incremental Learning Algorithms for a Web-based Telerobotic System". In Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA). Washington, May 2002.

[Marín et al., 2002c] R. Marín, P.J. Sanz., J.S. Sanchez, "A Predictive Interface Based on Virtual and Augmented Reality for Task Specification in a Web Telerobotic System". In Proceedings of the IEEE International Conference on Intelligent Robots and Systems. Lausanne, October, 2002.

[Marin et al., 2002d] R. Marín, P. Vila, P.J. Sanz, A. Marzal. "Automatic Speech Recognition to Teleoperate a Robot via Web". In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS). Lausanne, October, 2002.

[Marín et al., 2003] R. Marín, P.J. Sanz, A. P. del Pobil. "The UJI Online Robot: An Education and Training Experience". Autonomous Robots 15, pp. 283-297, Kluwer Ac Pub., the Netherlands, 2003.

[McKee, 2002] G. T. McKee, The Development of Internet-Based Laboratory Environments For Teaching Robotics and Aritificial Inteligence. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA). Washington, May, 2002.

[Sanz et al., 1998] P.J. Sanz, A. P. del Pobil, J. M. Iñesta, G. Recatalá. "Vision-Guided Grasping of Unknown Objects for Service Robots". In Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 3018-3025, Leuven, Belgium. May 1998.

[Sherindan, 1992] T. Sherindan, Telerobotics, Automation, and Human Supervisory Control. Cambridge: MIT Press, 1992.

[Taylor et al., 2000] K. Taylor and B. Dalton, "Internet robots: a new robotics niche", *IEEE Robotics & Automation Magazine*, vol. 7(1), 2000.

[Telegarden-Website] Telegarden webpage (http://www.usc.edu/dept/garden/).