Online Robotic Experiments for Tele-Education at the University of Pisa*

A. Bicchi*† A. Caiti* L. Pallottino *† G. Tonietti*†

† Interdepartmental Research Centre "E. Piaggio", University of Pisa
* Department of Electrical Systems and Automation, University of Pisa

Abstract

In this paper we describe work being done at our Department to make the Robotics laboratory accessible to students and colleagues, to execute and watch real-time experiments at any time and from anywhere. We describe few different installations, and highlight the underlying philosophy, which is aimed at enlarging the lab in all the dimensions of space, time, and available resources, through the use of Internet technologies. In particular four experimental set-ups with respectively hardware and software architecture description are presented: the DC motor, the magnetic levitator, the Non-Holomonic Motion Planner (NHMP), and the Graphic Environment Tool.

1 Introduction

In the recent past, Internet and WWW technologies have been impacting robotics rather profoundly under many regards. In the first place, and most basically, their immense popularity attracted the interest of common people to advanced technologies, which in part explains the renewed vigour of Robotics (widely felt as a companion technology to computers and the internet), and the new expectations and challenges posed by the society to Robotics research.

More directly, many applications have been developed to date that interface robots and the web. The Mercury project[1] (probably the earliest example of robots on the web), and the PumaPaint[2] project are examples of installations that explored various ways of social and artistic interactions with robots on the web. Technically, they consist in interfaces allowing to specify certain tasks to be accomplished by using a limited class of specific functions. Starting with the work of Taylor,[3] various sites have developed interfaces to program a robot arm or vehicle (see e.g., 4–8) through the net. Purposes vary from executing traditional robotic tasks, such as picking and placing parts, to wandering in museums and interact with people. A web site for interactive design of benchmark problems in planning nonholonomic vehicles motions, and for collecting from the web different proposed solutions, was presented in 9.

Among the many advantages offered by internet robotics, there is the possibility of offering students wider access to University laboratories in order to exercise on an experimental setup notions they have learned theoretically in the classroom or on books. In the domain of robotics,

*Work done with partial support of FAI ROBOT and TIGER projects, CNR - MIUR. E–mail: bicchi@ing.unipi.it

the amount and quality of lab practice is often limited by limitations on the number of robot workstations available; on the freedom in exercising, due to security constraints; on time and hours, due to staff schedule; and on the variety of robotic devices.

The development of internet-based experimental set-ups accessible by students has been going on since several years at the Faculty of Engineering of the University of Pisa,[9–12] mainly driven by the demographic pressure to which our Faculty was and still is being exposed. In particular, the increasing number of students enrolling in engineering courses, together with difficulties in enlarging existing didactic laboratory facilities, prevent each student to have access to a proper number of laboratory experiences. This ultimately risks to jeopardize the reaching of basic and advanced engineering skills. The work being done at our University to virtually expand the Robotics Lab, and make available to students and colleagues a larger number of experiments, at ease from their home and at their preferred working hours, is described in this paper. In particular, we will present in detail four installations. The first two are remote experimental facilities on a basic DC motor control problem and a Levitator Control Systems. The user can access the experiments via a web–based queuing mechanism, choose among various types of experiments, make her/his choice about the type of controller, the sampling time and other parameters, then execute the experiment, watch it through a webcam and retrieve experimental data from the real plant. The third gives the possibility to apply path planning algorithms to a unicycle–like experimental hardware. Through a camera it is possible to see the motion of the car between obstacles. Finally, the fourth installation is a more complex environment for robot programming, simulation, and execution on different hardware platforms. The program provides a high-level interface to multiple robots, allowing easy programming, debugging, simulation, and finally interpretation in the proprietary language of the chosen target robot. The user can check execution of the program by visualizing the robot operation through the browser. The program is intended primarily for teaching robot programming to students. Possible extensions towards a device-independent, graphical programming language for robot programming in automated factories can be envisioned.

Installations, described in this paper, can be found at the following web page:
http://www.piaggio.ccii.unipi.it/robotics/telelab-eng.html

The realization of the above mentioned experimental set-ups has led more recently to the participation of the Pisa team to nationally funded projects focused on the realization of e-learning training courses in robotics. To reach this goal, a suite of support tools has been designed and is currently being implemented so that the existing laboratory experiences can be upgraded to become proper learning objects. In this paper we describe our current effort in producing self-assessment capabilities, tutoring procedures, and training evaluation procedures, which are considered among the tools that have a major impact on the students learning process and on the incremental design and evolution of the system.

The paper is organized as follows: the next three sections are devoted to the description of the experimental set-ups and of their hardware and software architecture; section two focuses on the basic experiments (DC motor and magnetic levitator), section three on the Non-Holomonic Motion Planner (NHMP), and section four on the robot programming platform. Section five describes the upgrade of the experimental set-ups toward learning objects, which is currently in progress. Finally, some conclusions are given.

2 DC Motor and Levitator Control systems

The first two online experiments that have been developed are the DC Motor and the Levitator Control Systems. The primary aim of these experiments is to allow students to gain practice in

Figure 1: The Magnetic Levitator experimental setup developed by *"Laboratorio Didattico Sperimentale"* (LADISPE), Politecnico di Torino

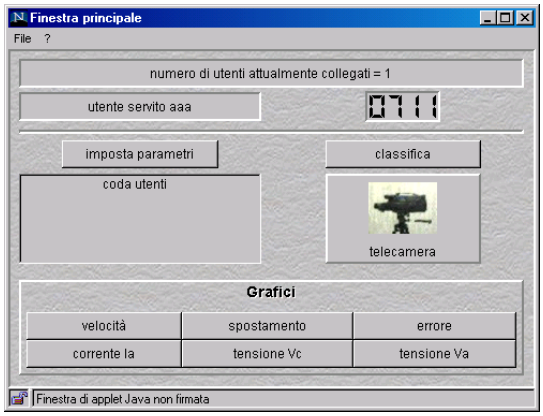

Figure 2: JAVA interface to the online experiments

the design of controllers for open–loop stable and unstable plants respectively.

The hardware of the DC Motor control system is quite essential and consists in an inexpensive brushed DC motor, a PWM power amplifier, and an incremental encoder. The hardware of the Levitator Control System consists in a Levitator setup (see fig.1) realized by $LADISPE$, Politecnico di Torino. Dedicated PCs, operating under FreeBSD and RT–Linux respectively, support both the digital controller implementation and the Internet communication. The latter consists in a videoconference broadcasting tool (VIC v2.8), and in a downloadable JAVA applet. The digital controller is implemented as a routine in the OS kernel, realizing a digital linear filter between past inputs (encoder measurements) and outputs (control signals sent to the D/A port), of up to the sixth order. The Graphic User Interface (GUI) of both online experiments are quite similar. Connecting to the main page of the remote lab, the user can download a JAVA graphic interface to control the experiments, appearing as in fig.2. By clicking the *Camera* button, a videoconference

Figure 3: Live image of the experimental hardware of the DC Motor

session is started, and live images of the experimental setup are shown as in fig.3 (for users that do not have a videoconference tool installed, a free software is made available). Users can access the experiments by registering a nickname into a queue of users. The queue, which at present has a capacity of 10 users, is served FIFO, and is implemented by a JAVA class provided by the JDK JAVA developer toolkit. When the queue ahead has been served, the user is given access to a mask for setting various choices, such as: reference signal type (step, ramp or sinusoidal) and amplitude; type of controller class (PID, transfer function); type of analog-to-digital conversion (forward and backward Euler, Tustin); sampling time, etc.. Upon clicking the *OK* button, parameters are passed to the PC controller, and the chosen experiment is executed, while the user watches the effects of her/his choices in real time. At the end of the experiment, several data about the hardware (e.g. shaft position and velocity, tracking errors, motor currents and voltages of the DC Motor) can be plotted with such convenient utilities as interactive zooming and grids (see examples in fig.4 and fig.5). Furthermore, data can be saved in local files by the user for later examination and analysis. The remote experiments can be used by the instructor or by the student himself to grade the student's preparation. To this purpose, various useful performance indexes (ISE: *Integral Square Error*, ITAE: *Integral Time–Absolute Error*, ITSE: *Integral Time–Square Error*, and IAE: *Integral Absolute Error*) are evaluated automatically at the end of the experiment, and a list of "top" users is updated (see fig.6). We have found that this video–game inspired feature helped much in motivating the student's attention.

3 The Non Holonomic Motion Planner

Another online tool is based on the Non Holonomic Motion Planner Benchmark. Such benchmark, is an interactive environment available on the World Wide Web intended to allow fair and thorough comparison of different techniques to solve a basic problem in nonholonomic motion planning. By connecting to one of the server sites, users, potentially unaware of the technical subtleties of the planning problem, but well conscious of his application needs, can design the benchmark problem that is most significant to his purposes. Users can then obtain different solutions from

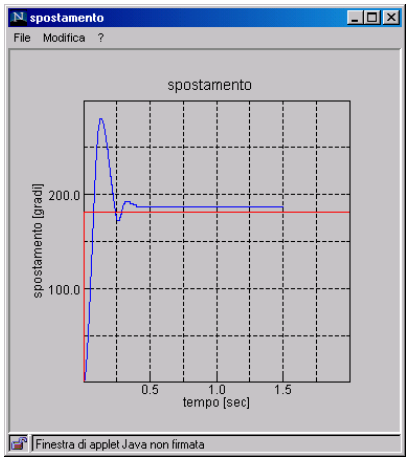

Figure 4: Plot of positions of the DC motor in a typical experiment

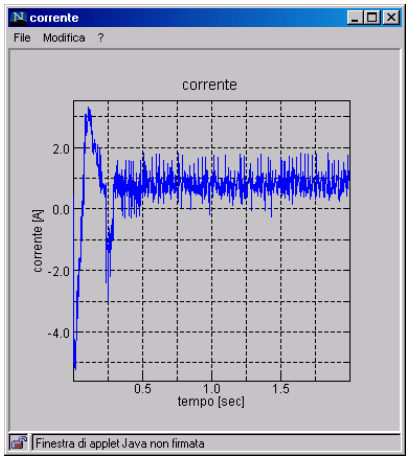

Figure 5: DC motor current in the DC Motor experiment

several algorithm providers, and compare them both qualitatively (by graphic display), and quantitatively. Providers implement their own algorithms at their sites, with wide freedom of choice in programming language, computational architecture, etc., while complying with few simple protocol conventions. It is believed that similar usage of the Web, easily extendable to domains other than NHMP, can usefully contribute to the fair comparison of result among researchers, as well as to the diffusion of advanced research results towards application–oriented users.

The GUI of the NHMP has been conceived to allow non–expert users to simply interact and practices with the online experiment. On the right–side of the GUI, the user is allowed to choose from different path–planning algorithms, and also can test their own by download it on the server of the application (see fig.7). After the setting is completed, the button *See Motion* allows the user to watch the simulation results of the chosen path planning algorithm as in fig.8.

The NHMP has been recently updated with the possibility to apply path planning algorithms to an experimental hardware developed in our lab. The experimental hardware, depicted in fig.9, consists on a webcam rigidly connected to a platform. Two wheels, which both are actuated by

Figure 6: Performance indices are evaluated, and exercises are ranked automatically by the system.

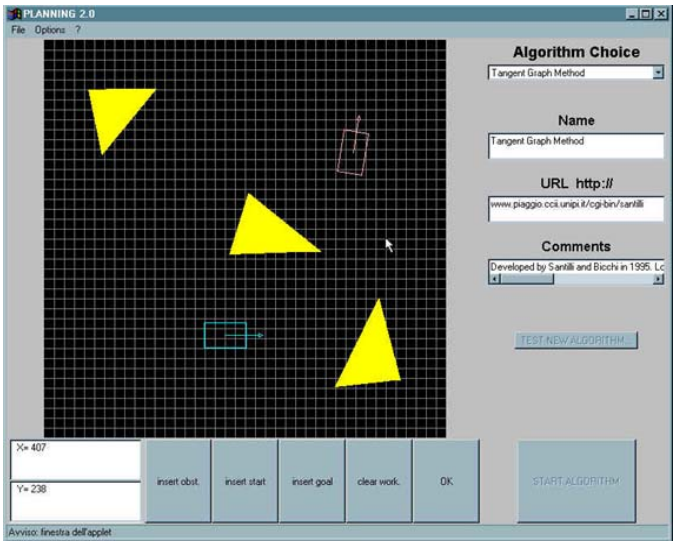

Figure 7: Java interface of the Non Holonomic Motion Planner. Users can insert in the grid both start and goal positions of the non–holonomic vehicle (*blue* and *red*, respectively), and create obstacles (*yellows*), to test the path planning algorithms

Stepper Motors, are responsible of the movement of the platform in planar coordinates (x, y, θ). In a *unicycle–like* fashion, the movement of the platform is *Forward (FW)/Backward (BW)* if both wheels rotate *Clockwise (CW)/Counter–Clockwise (C–CW)* at the same velocities, otherwise the platform rotates *CW/C–CW*. The camera, during the motion, continuously watches regions of a figure depicting a "unicycle" (a yellow square) at the center of a green square, and whose position is fixed. This implies the motion of the camera is perceived by the online user as a movement of the unicycle in the square region (see fig.10).

The Internet communication, and the visualization of virtual obstacles representing those chosen by the user, are provided by a free Videoconference Broadcasting Tool we programmed for this experiment.

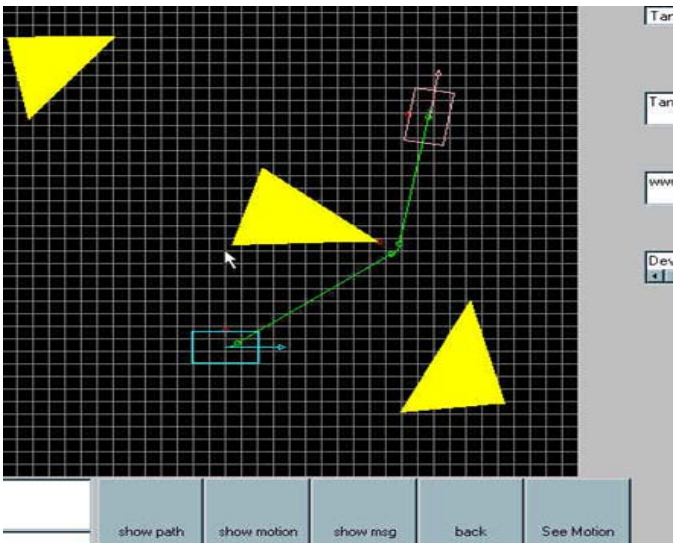

Figure 8: Highlight of the NHMP interface showing the simulation results of the path planning algorithm.

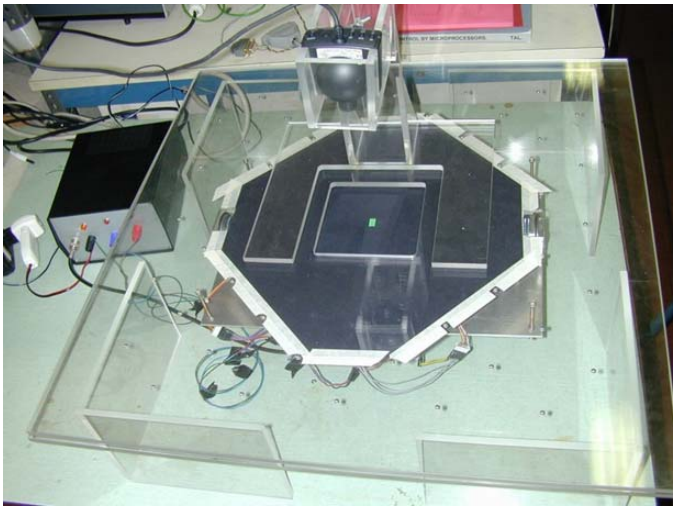

Figure 9: Experimental setup of the NHMP

4 Graphic Environment Tools for Remote Robot Programming

The development of this site (dating back to 1997, see 11) was aimed at providing students at their first robot programming course, with means for building, checking and simulating an abstract robotic program, compiling it for a few different robot arms, and watch the actual execution of the program on one of the arms.

The rationale behind this installation is that robot programming languages are still very often of robot manufacturers property, and are sometimes specific for a given robot model. The coexistence of many different programming languages for robots is clearly inconvenient for industries, as it makes it necessary to train programming personnel specifically for each device, and severely

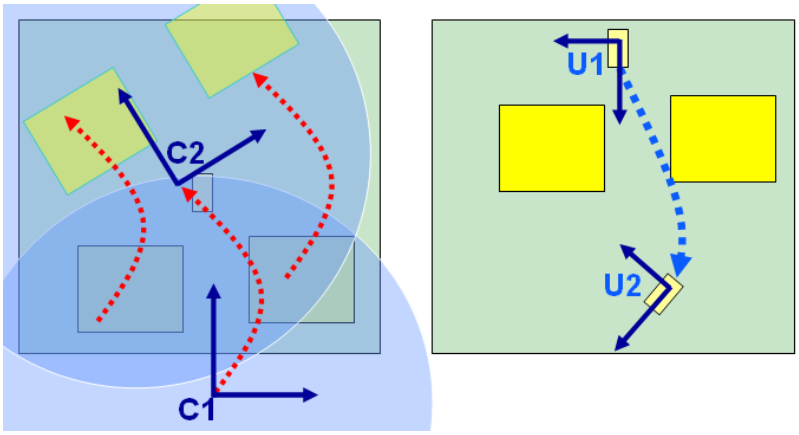

Figure 10: How the NHMP works. The movement of camera reference from $C1$ to $C2$ (*left*), is perceived by the online user as a movement of the unicycle reference from $U1$ to $U2$ (*right*). Note that virtual obstacles are rigidly moved with the camera

limits reuse of software. The number of subtle differences in languages also makes teaching robot programming at the earliest educational levels more difficult than it should. Naturally, specificity of languages is to some extent dictated by specificity of robot hardware (number of joints, joint limits, etc.). However, at the highest levels of the language abstraction hierarchy (i.e., at the object and task levels), it is natural to recognize that all robot arms share a device-independent set of primitives, which could be used as a natural basis for a common language for robots.

The quest for a general-purpose language, to uniformly program different robotic devices, motivated several researchers to adapt universal programming languages such as Pascal or C with specific libraries for robots. Among the earliest and most successful projects in this direction is the RCCL package developed at the McGill Research Centre for Intelligent Machines.[13] Several researchers are also experimenting with object–oriented robot programming languages, to accommodate for uniform and transparent treatment of different components of the robotic system, such as sensorial sources (see e.g. 14).

Our project's aim was to provide a programming language of a rather wide generality, which also integrates modern graphic user interface tools to achieve an intuitive programming interface easy to teach to an undergraduate, or even high-school, level. Inspiration to our work came in part from the Onika Project at Carnegie Mellon University,[15] consisting in a multilevel human-machine interface for reconfigurable real-time control systems, programmed through icons, to interact with a real-time operating system in the context of a reconfigurable software framework to create reusable code. Onika presents appropriate work environments for both engineers and end-users applications. In particular, it verifies that all jobs are complete and syntactically correct. Onika has been fully integrated with the Chimera real-time operating system in order to control several different robotic system in the A.M.L. at C.M.U..

The Graphic Environment Tools (GeT) for remote robot programming is an integrated teaching tool, which allows users to program at their ease, through an Internet connection, one of several exercise tasks that an instructor may prepare. The philosophy of using GeT is to dispose of a hardware installation comprised of a host server, running the Web interface and servicing the application, and several target servers, each managing directly a robot arm (see fig.11) via a CGI script. The core of the installation is a JAVA applet providing a graphic robot language with a basic instruction set, and an interpreter for different robot controller languages. The user connects to

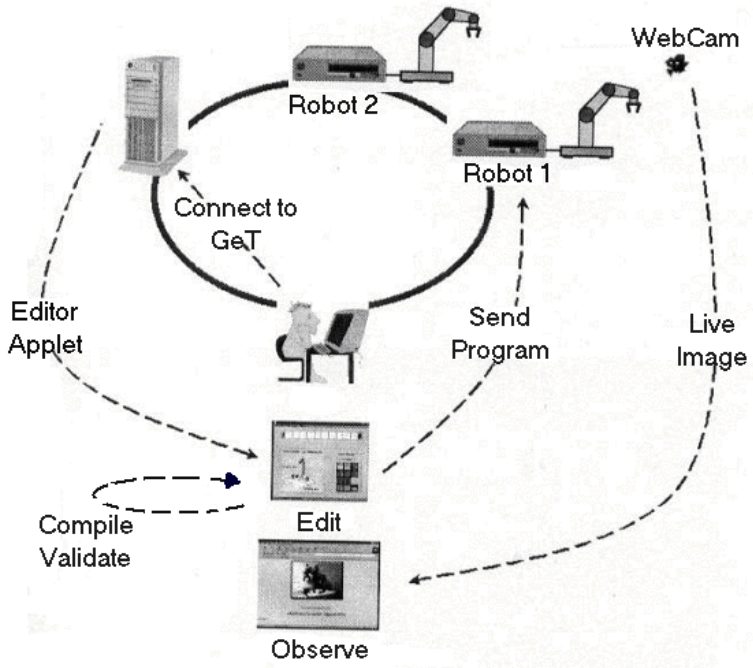

Figure 11: An example of a hardware architecture using GeT

the host server via any browser that supports Java Virtual Machines (JVM). Once connected, it is possible to start the videoconference broadcast (see fig.12) by clicking on the SCORBOT CAMERA button. Furthermore, by pressing the START button an instance of the Editor class is created in a new window on the user's computer. The structure of the GeT editing applet, illustrated in fig.13, is comprised of two main parts: a Program Score, initially blank (top); a Workspace Description for the robot and task (bottom left); and a Command Icon repository area (bottom right).

The Workspace Description area includes a graphic representation of the robot arm and its environment, including sensors, for the programming task to be executed. Information such as initial positions of objects, and naming of sensor–related variables are reported in this area (see fig.14).

The Command Icon Repository collects icons for available commands, divided in two sets. By clicking on the Movement tab, a set of primitives for controlling robot movements are displayed, including such commands as HOME, MOVE TO, etc. (see fig.15 on the left).

By clicking on the Flow tab, a set of conditional and looping constructs are offered, including IF, LOOP, RESET, etc. (see fig.15 on the right).

In general, a large amount of icons can be used to program a task. In this case, the readability and the comprehension of the program can be increased by the development of a *Macro Repository area* that is currently under study. This area should allow students, and teachers, to create new *program* icons simply starting by a GeT program written with Movement and Flow icons. These icons will represent what is called *procedure* in C language (see fig.16). The user builds her/his own program for the specified task by simply dragging and dropping commands from the Command Icon to the Program Score. The score can be scrolled back and forth, and altered by moving, exchanging, and deleting cells. Command icons become active when in the Program Score, and a cell editing command is activated by clicking on the cell (see fig.17). Arguments to commands can be set by choosing the Data Set option in the cell edit menu. For instance, the parameter mask

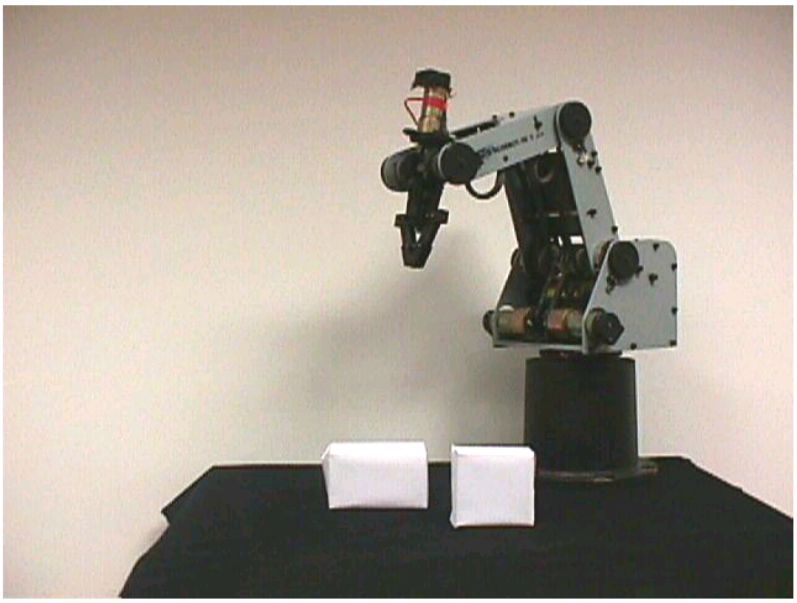

Figure 12: A webcam view of the target robot arm.

for a MOVE command are shown in fig.18. Only once the user has completed editing the program, this can be validated for the robot arm. By choosing the Compile option of the Project menu, the graphic program score is parsed and checked for possible syntax errors and logical inconsistencies. Finally, the program is validated with respect to kinematic and hardware constraints, such as e.g. workspace limitations and joint velocity saturations.

Upon validation, the user can simulate the program via a simple tool included with the applet. The simulation window (see fig.19) contains a graphical panel showing the robot arm simulating the execution of the program. A command console with buttons to change the viewpoint, activate a step by step or a continuous simulation, or exit the simulation is also contained. After a successful validation phase, the program is translated in the programming language for the target robot arm. By clicking on the Send tab of the Compile menu, the program is sent to the target controller, and executed by the target robot arm as soon as this becomes available. Although the videoconference is broadcasted (users can simultaneously view what the robot is doing), access to the GeT applet by multiple remote users are regulated in mutual exclusion by a queuing mechanism implemented at the host server level.

4.1 Implementation Details

In our present implementation, the host server is an Intel Pentium II 350Mhz with 64MB di RAM and a 10GB hard disk, operated under Windows XP and communicating with the robot via a RS232 link. The web camera is connected via a parallel port and is operated under the Microsoft NetMeeting software. A robot Scorbot-er V plus, produced by Eshed Robotec BV, is presently connected to the interface. This is an anthropomorphic 5-axis robot arm, whose controller uses a proprietary language.

Recently, the hardware of the GeT has been upgraded with a sensorized platform useful for pick–and–place experiments. The platform has a cylindrical shape (see fig.20), and it is sensorized

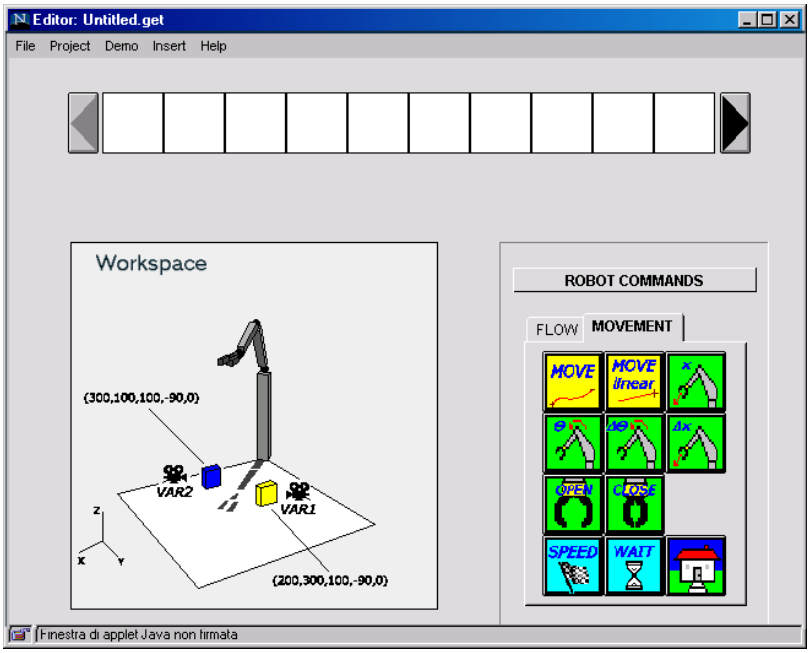

Figure 13: The GeT applet appearance.

by three ON/OFF touch sensors disposed at the vertices of a triangle inscribed to the base circumference (see fig.21). The map between end-effector positions X_E and sensors values ($S1$, $S2$, $S3$) is as follows

$$\begin{cases} X_E \in (S1, G, S2) & \rightarrow & (1,1,0) \\ X_E \in (S2, G, S3) & \rightarrow & (0,1,1) \\ X_E \in (S3, G, S1) & \rightarrow & (1,0,1) \\ X_E \in (S1, G, S1) & \rightarrow & (1,0,0) \\ X_E \in (S2, G, S2) & \rightarrow & (0,1,0) \\ X_E \in (S3, G, S3) & \rightarrow & (0,0,1) \end{cases}$$

where (a, b, c) represents the sector of the circle with center b between the two points a and c belonging to the circumference.

Such map can be adopted easily by users so as to identify the region of the platform in which the robot end–effector belongs to. By this fact, users can develop algorithms, and GeT programs, in wich the position of the end–effector is recursively adjusted to find the goal position (placed at the center of the platform).

5 From laboratory experiments to learning process

The current evolution of the experimental systems toward learning experiences, or "learning objects", in the e-learning community jargon, is now described. The ultimate goal of this development is that of constituting a network of remotely accessed laboratories for web-based education (e-learning) in robotics and automation. In order to reach the objective, several actions are being taken. Some of these actions are related to the general network structure (sharing of metadata descriptions and of authoring and knowledge repository tools, definition and implementation of

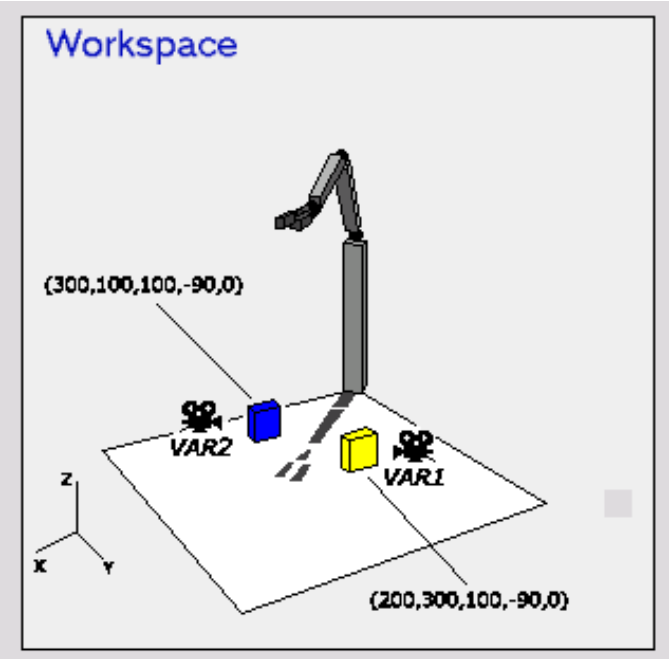

Figure 14: Graphic visualization of the robot workspace.

a common administrative management facility, etc.). Other actions require the tailoring to the robotics experiments presented in the previous section of general guidelines to be followed in the network. In particular, the general guidelines require that each experiment is complemented by the following objects:

- authentication and traceability of the students;
- instructional material on the specific topics addressed by the experiments, in the form of an hypertext document;
- a set of tests proposed to the students, with automatic evaluation of the results, to provide the students of a self-assessment capability;
- tutor interaction tools, in the form of a bulletin board and a FAQ list (asynchronous interaction)
- procedures for the evaluation of the learning object

A system of access control through authentication is being established. The access control will limit the availability of the additional learning tools to registered students; however, the possibility to interact with the experiment from unregistered students will be kept.

Self-assessment tools are being designed in the form of "figure of merits" on the performance of the student proposed solution on given test cases, in comparison with a standard proposed solution and with the solution obtained by the other students in the same course. This approach has already been implemented on the two basic experiments (DC motor and magnetic levitator)through standard performance measures well-established in the control and automation literature. For the two robotic set-ups, however, the definition of figure of merits is not that straightforward. To our

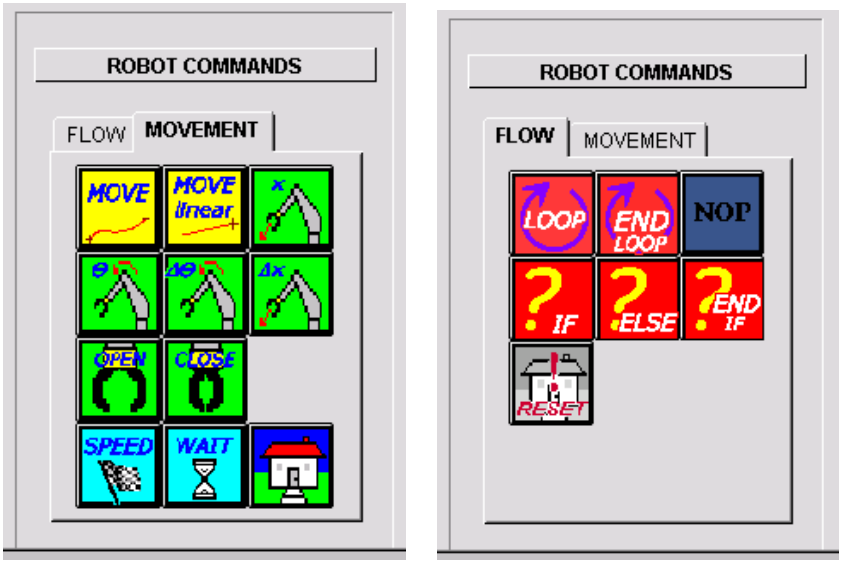

Figure 15: Left: The set of robot movement primitives, Right: The set of conditional and looping commands

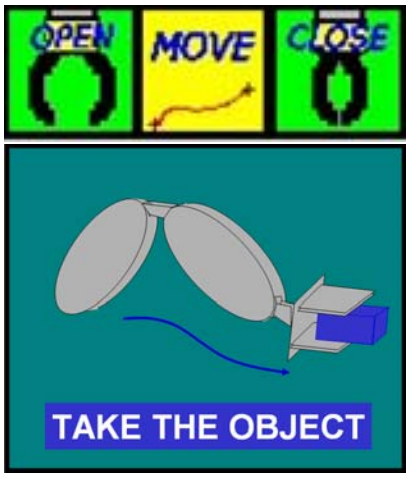

Figure 16: An example of an icon that can belong to the Macro Repository area: the TAKE-THE-OBJECT icon. Using Movement and Flow icons, user must insert a minimum of three icon in the Program Score (*up*): OPEN, MOVE TO and CLOSE. Using a Macro icon of this program (*bottom*), user will add only an icon to the Program Score, and will specify the (relative or absolute) position of the object.

knowledge, there does not exist a single standard quantitative measure of performance of planning algorithms; there are indeed several different choices, in terms of failure/success on test cases, time to accomplish the planned task, computational time of the planning algorithm, complexity of the planning algorithm, complexity of the planned manoeuvre, etc. Indeed, the peculiarities of the tasks to be planned may require different performance measures on a case-by-case basis. Starting from this consideration, two different significative measures of performance have been enucleated. For the NHMP, a cost function is defined in terms of a weighted sum of length and mean curvature of the

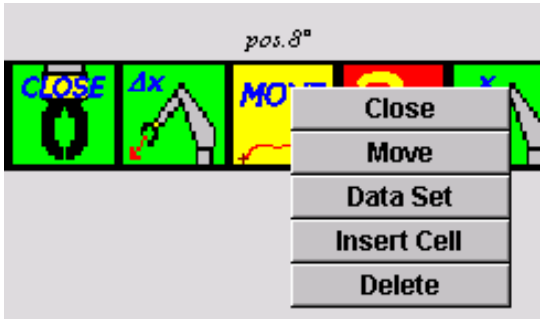

Figure 17: Editing menu for a command cell

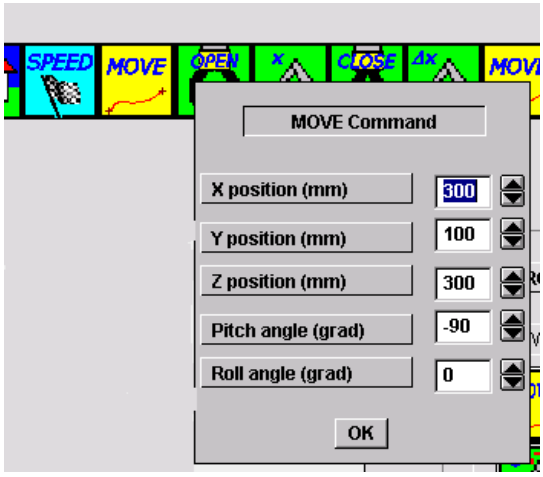

Figure 18: The parameter mask for a MOVE command

planned path; in this way the shortest and smoothest paths are favoured. The students implement their own path planning algorithm in a similar fashion as that of the algorithm providers described in the NHMP section, and are able to compare the performance of their implemented solution with that of the existing algorithms and of their fellow students over a set of pre-defined problems. In the GeT experiment, the standard exercise is that of a pick-and-place task in presence of set-up generated disturbances; the planner has availability of a quantized feedback information from the sensorized platform previously described. The students have to design through the GeT interface a program able to execute the task. The figure of merit of each proposed solution is given by a weighted sum of the number of instructions (building blocks of the language) and the number of elementary operations in the manoeuvre execution. The rationale underlying the choices of the figures of merit is the following: in the NMHP case, it is evaluated the performance of the planning solution, irrespectively of the algorithm complexity and of execution time; on the contrary, the GeT programming evaluation focuses on the computational efficiency of the proposed solution.

A fundamental component of a learning object, whose presence is critical for continuous improvements of the overall system and/or of some of its specific components, is a procedure for training evaluation. Assuming that most of the participants in the web-based courses will be university students, the training evaluation is performed accordingly to the first two levels of the Kirkpatrick model:[16] evaluation of reaction and learning. An experimental follow-up procedure for the stu-

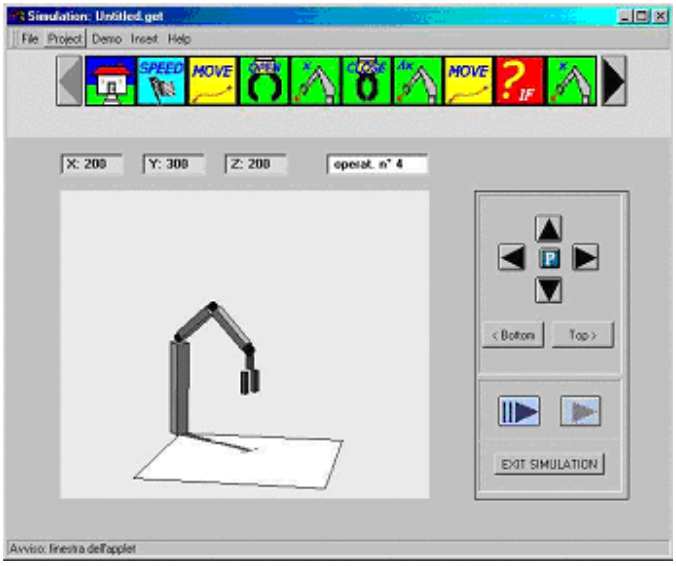

Figure 19: The simulation window of GeT

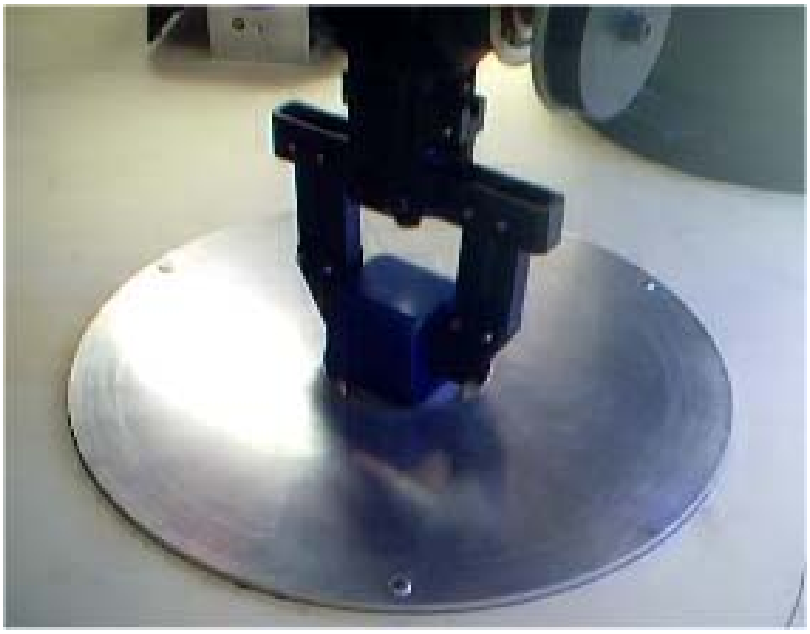

Figure 20: The sensorized platform of GeT

dents enrolling in the e-course for professional update to evaluate the learning transfer (the third level of the model) is also in preparation. At the reaction level, the student perception of the learning object is assessed; this is accomplished through a questionnaire in which each student has to indicate her/his subjective evaluation (on a one-to-five scale) regarding difficulties of learning; completeness of the instructional material; difficulties in access and use of the experimental set-up; availability of the experimental set-up; efficacy of the telepresence feedback (in all of our cases, of the videoconferencing system). As for learning evaluation, the results obtained by the students in

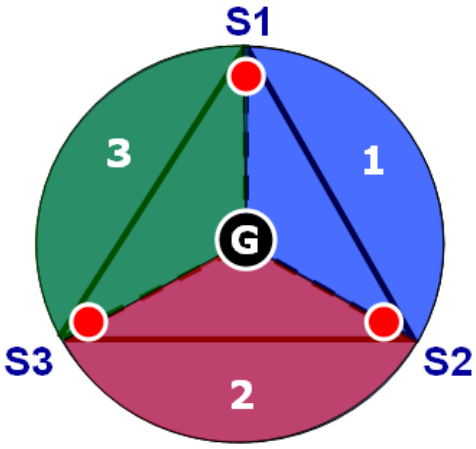

Figure 21: Disposition of the binary sensors of Get platform

the final e-learning course examinations are considered, and histograms are built from the results. The transfer evaluation will consist in a six-months follow-up of those students that have enrolled in the course not as part of their university program, but for professional update. After six months from the end of the course, they will be required to fill a questionnaire in which they have to give a quantitative evaluation (on a one-to-five scale) of: change in job mansions after the course; use of the new knowledge gathered in the course; interaction in the working environment with topics addressed in the course; how much of the topics learned in the course are still maintained. At the present stage, the implementation of a protocol for the evaluation of the fourth level of the Kirkpatrick model (impact and cost/benefit) is considered premature.

6 Conclusions and future developments

In this paper we have described several WWW installations intended for support to teaching elementary robotics courses. The DC motor and Levitator installations are rather simple but (probably because of such simplicity) very effective and robust. The Non Holonomic Motion Planner is adopted as a benchmark for path planning algorithms for non–holonomic vehicles. In the future, our aim is to share NHMP with other Labs, in the sense that it may be used as path–planner by several online non–holonomic plants.

The robot programming installation GeT represents an ambitious project, that has two ultimate objectives: one goal, more research-oriented, is the development of an intuitive, GUI-supported, object–oriented multitarget robot programming language for distributed programming. To this end, much work remains to be done, and our planned next steps involve increasing the number of target devices to which GeT programs can be downloaded, and to incorporate new sensors (such as force/torque sensors) and/or actuators in the system. The second goal, e-learning oriented, is to provide students with the basic capabilities in task planning experimentation.

All the set-ups presented are being complemented with tools for e-learning support, with the final aim of making our lab one of the nodes of a network of virtual laboratories. This network may eventually provide students with wider possibilities and experiences in the field of mechatronics.

7 Acknowledgements

Work done with partial support of FAI ROBOT and TIGER projects, CNR - MIUR. Authors gladly acknowledge co-authors of[10] A. Coppelli, F. Quarto, L. Rizzo, F. Turchi, and A. Balestrino. Work of student Torquato Cecchini and Ing. Andrea Pisani has been useful for setting–up the GeT experiment.

References

[1] K. Goldberg, M. Mascha, S. Gettner, and N. Rothenberg: "Desktop teleoperation via the world wide web", in *Proc. 1995 IEEE Int. Conf. Robotics and Automation*, 1995.

[2] M.R. Stein: "Painting on the World Wide Web: The Puma Paint Project" http://yugo.mme.wilkes.edu/ villanov/

[3] B. Dalton and K. Taylor: "A framework for internet robotics", in *Proc. Int. Conf. Intelligent Robots and systems (IROS): Workshop on web robots*, 1998.

[4] E. Paulos and J. Canny: "Delivering real reality to the world wide web via telerobotics", in *Proc. IEEE Int. Conf. Robotics and Automation*, 1996.

[5] R. Simmons: "Xavier: an autonomous mobile robot on the web," in *Proc. Int. Conf. Intelligent Robots and systems (IROS): Workshop on web robots*, 1998.

[6] P. Saucy and F. Mondada: "Khep-on-the-web: One year of acces to a mobile robot through the Internet", in *Proc. Int. Conf. Intelligent Robots and systems (IROS): Workshop on web robots*, 1998.

[7] R. Siegward, C. Wannaz, P. Garcia, and R. Blank: "Guiding mobile robots through the web", *Proc. Int. Conf. Intelligent Robots and systems (IROS): Workshop on web robots*, 1998.

[8] W. Burgard, A. B. Cremers, D. fox, D. Hähnel, G. Lakemeyer, D. Schultz, W. Steiner, and S. Thrun: "The interactive museum tour guide robot", in *Proc. 15th Nat. Conf. Artificial Intelligence*, 1998.

[9] S. Piccinocchi, M. Ceccarelli, F. Piloni, A. Bicchi: "Interactive Benchmark for Planning Algorithms on the Web", *Proc. IEEE Int. Conf. on Robotics and Automation*, 1997.

[10] A. Bicchi, A. Coppelli, F. Quarto, L. Rizzo, F. Turchi, and A. Balestrino: "Breaking the Lab's Walls: Tele-Laboratories at the University of Pisa". In Proc. IEEE Int. Conf. on Robotics and Automation, Seoul, Korea, pages 1903–1908, 2001.

[11] F. Giannini, F. Turchi: "Strumenti per la programmazione remota di robot con tecnologia Java", Thesis for the Laurea degree, University of Pisa, 1997.

[12] F. Quarto: "Implementazione hardware e Software di un sistema di controllo digitale a distanza tramite World-Wide Web e Java", Thesis for the Laurea degree, University of di Pisa, 1998.

[13] J.Lloyd, V. Haryward: "Multi-RCCL User's Guide", McGill Reserch Center for Intelligent Machines, McGill University, 1992.

[14] C. Zielinski: "Object-oriented Robot Programming" Robotica (1997) volume 15, pp41-48, Cambrige University Press

[15] M.W.Gertz, P.K. Khosla: "Onika: A Multilevel Human-Machine Interface for Reconfigurable Real-Time Control Systems" http://www.cs.cmu.edu/ aml/onika.html

[16] D.L. Kirkpatrick, *Evaluating training programs: the four levels*, Berret-Koehler, San Francisco, 1998.