Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

RACT: a Remote Lab for Robotics Experiments

Marco Casini * **Francesco Chinello** * **Domenico Prattichizzo** *
Antonio Vicino *

* *Dipartimento di Ingegneria dell'Informazione, Università di Siena,*
Via Roma 56, 53100 Siena, Italy
e-mail: {casini, chinello, prattichizzo, vicino}@ing.unisi.it

Abstract: The "Robotics & Automatic Control Telelab" (RACT) is a remote laboratory on robotics developed at University of Siena, which extends the field of application of the "Automatic Control Telelab" (ACT). This extension consists of adding experiments on a remote robot manipulator. RACT is mainly intended for educational use, and its Matlab-based architecture allows students to easily put in practice their theoretical knowledge on robotics. The first implementation of RACT consists of a remote experiment on inverse kinematics and of an experiment on visual servoing. Experiments on visual servoing represent the most advanced feature of the remote lab and work is in progress to add more experiments of this type.

Keywords: Distance Learning, Remote Labs, Robotics, Inverse Kinematics, Visual Servoing.

1. INTRODUCTION

Information technologies influenced the way of teaching many disciplines among which education in robotics and automation. Citing Dormido's excellent review, Dormido (2004), *"as educators we must have an open attitude and that we should sensibly incorporate technological development, because otherwise we may risk teaching the students of today how to solve the problems of tomorrow with the tools from yesterday."* Many distance learning paradigms have been developed in recent years and among these, laboratories accessible through the Internet are certainly of the most effective. Regarding the web technologies used in robotics and automation courses, comprehensive contributions have been provided by Poindexter and Heck (1999) and Dormido (2004), who describe the use of virtual and remote labs in control teaching. This paper will deal with remote labs, whose distinguished feature with respect to virtual labs is that users can remotely interact with real experiments instead of software simulations.

Nowadays, numerous remote labs are available all over the world. Remote processes may be of various nature, ranging from mechanical to electronic, from hydraulic to thermal to chemical, etc., or a mixture of them, see, e.g., Overstreet and Tzes (1999); Choy et al. (2000); Ramakrishnan et al. (2000) and Safarič et al. (2001). Several remote labs are world-wide available also for robotics experiments. One of the first examples of remotely driven industrial robot arm has been presented in Goldberg et al. (1995). Another successful project is the telegarden project presented in Goldberg (2000). In Bicchi et al. (2001) and Saucy and Mondada (2000), an expensive resource, such as a mobile robot, is shared on the web. Recently, in Safarič et al. (2005), authors presented a work-cell with a 6 axes robot for remote experiments. In Carusi et al. (2004) a LEGO mobile robot is used as device for remote experiments on autonomous navigation.

In this paper, the "Robotics & Automatic Control Telelab" (RACT) is presented. The main architecture is shared with the "Automatic Control Telelab" (ACT), the remote laboratory of the University of Siena, that is continuously running since 1999, see Casini et al. (2003, 2004). In the following, the main features of RACT are briefly summarized.

1. RACT is based on the Matlab environment. In general, a remote laboratory can either use a well-known software, such as LabVIEW (see Ramakrishnan et al. (2000)) and Matlab/Simulink (Junge and Schmid (2000); Apkarian and Dawes (2000); Hahn and Spong (2000)), or use some special purpose software, as in Bicchi et al. (2001) and Choy et al. (2000). It is the authors' opinion that well-known software environments are better suited to spread out the remote laboratory practice. Usually, users do not want to learn ad-hoc control languages which are tailored for the particular remote lab and they like to take advantage of control functions developed in other well-known contexts. On the other side, integrating a well-known software in a remote lab architecture may present some difficulties and makes the overall remote lab design more complex.
2. At the present stage, remote robotics experiments in RACT are built around a *Unimate PUMA 560* robot. However, in the future it is the authors' intent to use a latest robot which provides better performances.
3. RACT allow students to perform a certain number of robotics experiments. The first two experiments that have been designed for remote execution deal with a classical subject in robotics like inverse kinematics (Sciavicco and Siciliano (2000); Spong et al. (2006)) and a more advanced one like visual servoing (Chaumette and Hutchinson (2006)). In robotics education, it is very important that students put inverse kinematics into practice. The experiment consists in finding the joint variables in terms

of the given end-effector's position and orientation and then moving the robot manipulator according to the resulting joint variables values. Visual servoing is a more recent subject in robotics. It consists in driving the robot manipulator through images taken from a camera that can be mounted on the robot (camera-in-hand) or fixed with respect to the world frame. The experiment that has been designed for RACT refers to the camera-in-hand visual servoing scheme: a desired image is given and the remote user is asked to design a control law based on the current and desired images to reach the final position where the desired image has been grabbed.

The paper is organized as follows. In Section 2, a general description of RACT including motivations and hardware/software architecture is reported. Section 3 deals with a detailed description of the available and future experiments, while in Section 4 some conclusions are drawn.

2. THE "ROBOTICS & AUTOMATIC CONTROL TELELAB"

The "Robotics & Automatic Control Telelab" is an extension of the "Automatic Control Telelab", a remote lab developed at University of Siena for controlling remote processes, see Casini et al. (2003, 2004). The aim of this extension is to allow remote experiments on robotics. Although this project is still under development, some experiments are already working and some other will be added in the near future, see Section 3 for further details.

2.1 Motivations

The main motivation to realize such a remote lab is of educational nature. In fact, the positive feedback received by student about the ACT lead us to extend the present configuration to take into account also robotics, with a particular emphasis on robot manipulators.

Since robot manipulators allow one to design a wide range of experiments, and since such experiments are conceptually different from those already available in the ACT (typical experiments for designing and testing feedback control laws in automation), the need for developing a new remote lab become unavoidable.

In the first implementation of the lab, the experiments have been divided in two classes.

- *Basic experiments.* These experiments relate to some basic concepts on robot manipulators, like, e.g., forward and inverse kinematics. By using RACT, students can easily put in practice their theoretical notions on real manipulators without being physically present in the lab.
- *Visual servoing experiments.* These experiments concern the so-called image-based visual servoing, i.e., the control of robotic manipulators by means of images taken by one (or more) cameras, see e.g., Hutchinson et al. (1996). This framework allows one to design a wide range of experiments, ranging from easy to very difficult ones. The former should be used for educational purposes, while the latter should be used in research contexts to test advanced visual servoing algorithms.

2.2 General architecture

The robot used in this lab is a *Unimate PUMA 560*, an anthropomorphic manipulator with 6 degrees-of-freedom (Fig. 1).

Fig. 1. The "Unimate PUMA 560" manipulator.

Like the ACT, also the RACT server is based on the Matlab environment, allowing a powerful and user-friendly interaction with the robot. In fact, an essential aspect of remote labs for distance learning is the ease of use (see, e.g., Poindexter and Heck (1999)), to allow students to concentrate themselves to the requested task rather than to the usage of the lab. This aspect is guaranteed by Matlab, since it is a standard tool in the control and robotics community, and it is already known by students. Students will be requested to design some Matlab functions to perform the given experiment (see Section 3 for further details), and to test it through a graphical user interface. The robot communicates with the server through a serial port. To allow the communication between Matlab and the robot, the "Puma Toolbox" has been used (see next subsection). To allow the interaction with the user, the *Apache* web server with PHP extensions (see The Apache Software Foundation (2007)) has been installed on the server. The *Webcam32* software has been used for interfacing with cameras allowing on-line video and picture capture, see Kolban (2006). At this stage, the operating system used is Microsoft Windows XP, and implementation under Linux is in progress.

On the client side, users may connect to the remote lab through web pages, where they can find all the useful information about the available experiments. Once an experiment is chosen, they need to upload a file containing the designed Matlab function, after that they can interact with the robot through a Java applet. The use of Java applets allows an interaction which is platform independent and does not need to install any specific software on the client side. In fact, a web browser with a Java Virtual Machine is usually installed in any computer. The communication between the applet and the server is made through the TCP protocol. Note that in this lab timing aspects are not critical, since the controller resides in the server side. In fact, a delay in the transmission between client and server does not affect the system stability and performance, but it will only conduct to experiments in which the output is given back delayed.

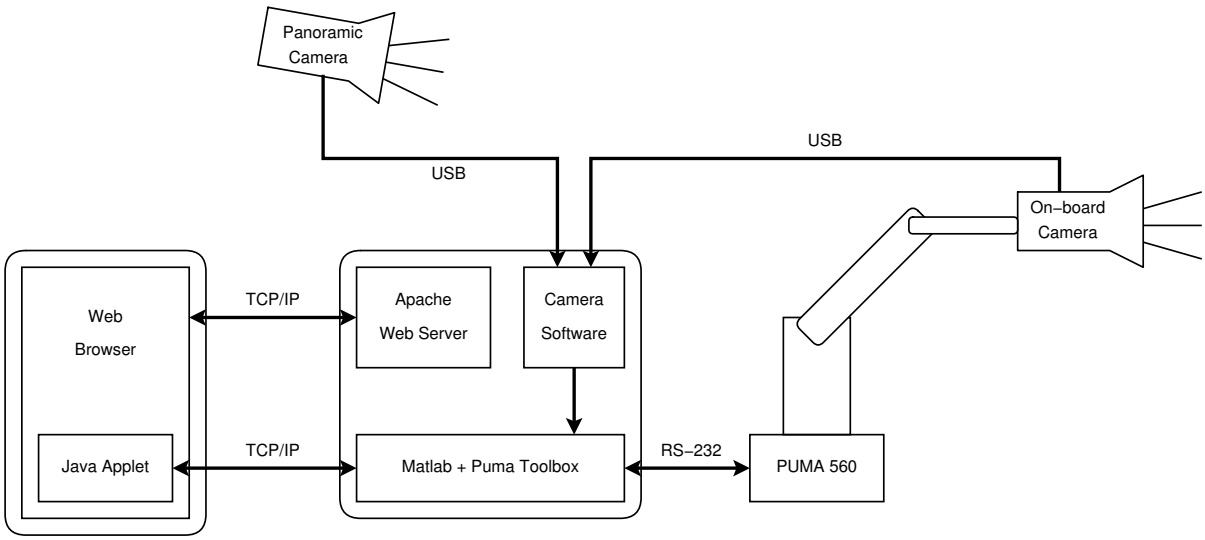

Fig. 2. The Robotics & Automatic Control Telelab architecture.

It is worthwhile to notice that Matlab just runs on the server side, and it is not needed on the client. In fact, the client application is only required to send a control function written in Matlab code, which simply consists in a text file. However, the presence of Matlab on the client machine should help users to debug and test their functions before sending them to the server. Moreover, Matlab can be a useful tool for off-line analyzing the experiment data. In fact, once a working session is finished, users may download a file containing all the data regarding the experiments; such data can be downloaded in Matlab format (.mat) or as ASCII text file.

To prevent that a user could monopolize the robot, for each working session, a maximum number of experiments has been fixed, after that the user is automatically disconnected. In addition, a time-out has been also implemented for the same reason. The state of the robot (ready or busy) is shown by an appropriate notice. At the present stage, any user may perform experiments whenever the robot is ready, and no access queue is provided. However, such a facility may be easily implemented in the future if needed.

Like other remote labs, safety aspects have been taken into account. To this purpose, to prevent damages to people or to itself, the robot has been placed in an off-limit site and suitable hardware security systems allows it to move inside a safe region. Moreover, some software safety mechanisms have been provided, like, e.g., executing the function designed by a user with low privileges to avoid the execution of commands potentially dangerous for the system.

A sketch of hardware/software architecture is reported in Fig. 2.

2.3 The "Puma Toolbox"

As previously pointed out, the interaction between the PUMA 560 and Matlab is possible thanks to the "Puma Toolbox". Such a toolbox has been developed at University of Siena (for details, see Chinello (2007)) and allows to drive the robot by a suitable set of Matlab functions. The Puma Toolbox requires a PUMA series robot using the VAL II system (Unimation Robotics (1983)). The toolbox improves several VAL instructions and commands to calibrate, move and return the position of the robot. Through these functions, it is possible to show angle plots or 3D animation of the robot, to use Simulink for moving each joint following a specific trajectory or to convert positions in various coordinate system. In fact, the VAL language is not able to automatically move the robot in different positions if these are changing in time by an external device like a camera, while this becomes possibile by using the toolbox functions. However, if users need to program the robot by means of the VAL language, a suitable function is available. All the programs and variables are saved in the controller memory so the robot can use them. Normally, users have to save and read the memory manually, while by using the toolbox one can store, read and clear the memory directly from Matlab, by using functions or scripts, and saving all the variables of interest in the Matlab workspace.

The toolbox is designed to use the PUMA for research applications, allowing the usage of the robot without a TTY emulator or similar. The functions provided by the Puma Toolbox are more than 50. In Table 1, the functions used in RACT are reported.

3. EXPERIMENTS DESCRIPTION

In this section, a description of the available experiments as well as those to be added in a future is reported.

3.1 Inverse kinematics experiment

A first experiment regards the inverse kinematics topic, i.e., evaluating robot joint angles in order to reach a given position and orientation of the end-effector. Students are asked to design a function in Matlab to perform this task. Of course, all the useful data regarding the geometry of the manipulator (e.g., links length and coordinate system orientation) are available to students in the web page associated to the experiment.

The Matlab function designed by students must take as inputs the six coordinates associated to the target position and orientation of the end-effector. All the coordinates are referred to a prescribed reference system. As output, this function must return the six angles in the Denavit-Hartenberg representation, see Spong et al. (2006). So, the function declaration looks like the following:

$$\texttt{function}\ [\mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3, \mathtt{d}_4, \mathtt{d}_5, \mathtt{d}_6] = \mathtt{IK}(\mathtt{x}, \mathtt{y}, \mathtt{z}, \hat{\mathtt{x}}, \hat{\mathtt{y}}, \hat{\mathtt{z}});$$

where IK denotes the function name, $\mathtt{d}_i$, $i = 1, \ldots, 6$, are the six angles in the Denavit-Hartenberg convention, $\mathtt{x}, \mathtt{y}, \mathtt{z}$ are the spatial coordinates and $\hat{\mathtt{x}}, \hat{\mathtt{y}}, \hat{\mathtt{z}}$ are the corresponding angles.

Once students have designed the function, they may upload it to the server through a web page. At this step, they can interact with the real robot by the user interface reported in Fig. 3. As previously reported, to allow platform independence, such an interface has been implemented as a Java applet.

To increase the sense of presence in the lab, two cameras have been provided. The former (*on-board camera*) is placed on the end-effector of the manipulator, while the latter (*panoramic camera*) is located outside the robot workspace to take a panoramic view. In the next subsection, it will be shown how the on-board camera will be used to perform visual servoing experiments.

Users are asked to fill in the *reference* fields, i.e., the target position of the robot, and to start the experiment. The joint angles are computed by the designed function, and the corresponding fields are automatically filled. At this point, the *pumatbx_movejt* function let the joint angles assume the computed values. After moving, the *pumatbx_where* function returns the position and orientation of the end-effector computed by the robot internal routines. Such values are then written in the *output* fields, and the positioning error, i.e., the difference between output and reference, are computed.

pumatbx_start	Starts a serial connection to the robot and returns its handler.
pumatbx_cal	Calibrates the joints of the robot at session start.
pumatbx_where	Returns the robot position (in Eulero angles) and precision point (encoder angles).
pumatbx_move	Moves the manipulator to a position using Eulero angles and a translation vector.
pumatbx_movejt	Moves each joint to a given encoder angle.
pumatbx_draw	Moves the robot along a given coordinate vector.
pumatbx_d_h2jt	Converts the joint angles in the Denavit-Hartemberg convention to a *precision point*.
pumatbx_jt2d_h	Converts the current *precision point* to the joint angles in the Denavit-Hartemberg convention.
cin_dir_puma	Computes the matrix of forward kinematics.
cin_inv_puma	Computes the angles of inverse kinematics, through the Denavit-Hartemberg convention.

Table 1. Description of the "Puma Toolbox" functions used in RACT.

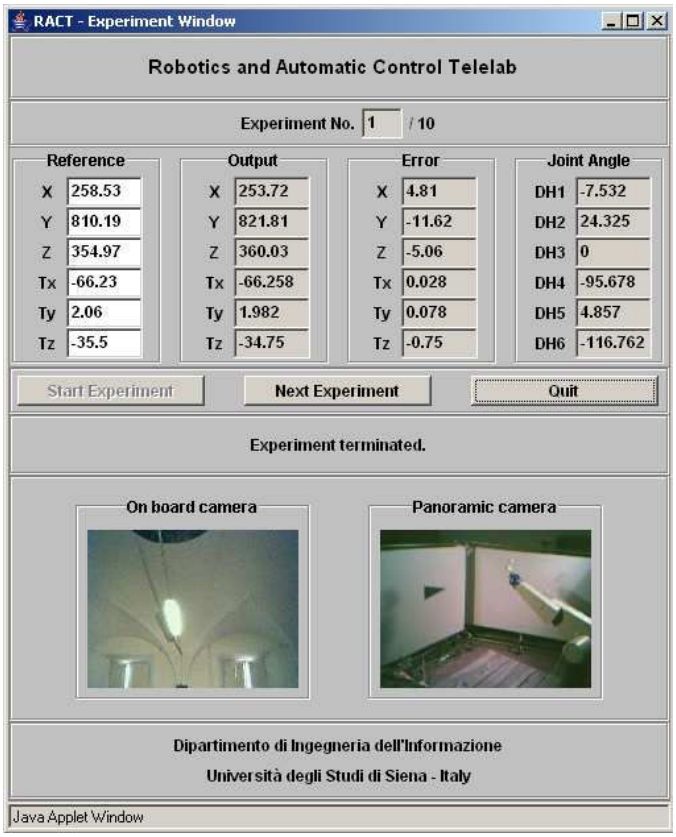

Fig. 3. The user interface for performing the inverse kinematics experiment. Position data are in *mm* while angles are in degrees.

3.2 Visual servoing experiment

This experiment deals with the control of the robot based on images acquired by a camera. In particolar, the task consists in moving the robot from a known initial position to a final (target) one. The only information available about the target position is a picture taken by the camera placed on the end-effector. Thus, the objective is to move the robot in a position such that the current image is similar to the target one.

In general, finding an algorithm to solve this kind of problem is a complex task, especially if the robot has several degrees-of-freedom and it is immersed in an unknown environment. Since the main purpose of this project is of educational nature, to simplify the experiment, the robot has been placed in front of a white panel on which a black triangle has been placed. Moreover, the on-board camera has been located orthogonally to the panel and the robot is allowed to move maintaining such a camera orientation. Since the end-effector orientation is given, it is possible to reach a given position using only translations, thus reducing the complexity of the experiment.

So, students are asked to design a function which takes as input the current and the target images and returns the vector denoting the relative robot movement along the three axes:

$$\texttt{function}\ [\mathtt{m}_\mathtt{x}, \mathtt{m}_\mathtt{y}, \mathtt{m}_\mathtt{z}] = \mathtt{VS}(\mathtt{current\_img}, \mathtt{target\_img});$$

where VS is the function name and $\mathtt{m}_\mathtt{x}, \mathtt{m}_\mathtt{y}, \mathtt{m}_\mathtt{z}$ denote the relative motions along each direction. This function can

be designed by students assuming to know the camera calibration data or not. Of course, in the last case, the function design will result more involved. Note that the usage of Matlab allows students to design their functions starting from a large number of routines already provided in Matlab toolboxes, like, e.g., the "Image Processing Toolbox" or the "Robotics Toolbox" (see Corke (1996)), allowing the design of powerful algorithms.

Once uploaded the designed function, users can interact with the robot by the interface reported in Fig. 4. Here, students can set the starting and target position of the robot by filling the appropriate fields. These reference positions can also be set randomly within a prescribed range. Moreover, since the requested task (i.e., moving the robot close to the target position) requires in general several iterations, it is possible to stop the execution at every step (manual mode) or to run it until the maximum number of steps is reached (auto mode). Three camera windows are provided. The first two windows show images taken by the on-board camera in the current and target position, respectively. The third one contains the picture taken by the panoramic camera.

After each step, i.e., each robot movement based on the designed function, the x, y, z coordinates of the current position are reported along with four graphic charts representing the positioning errors along the three axes and the euclidean distance between the current and the target position. The objective is therefore to reduce such an error in the smallest number of steps.

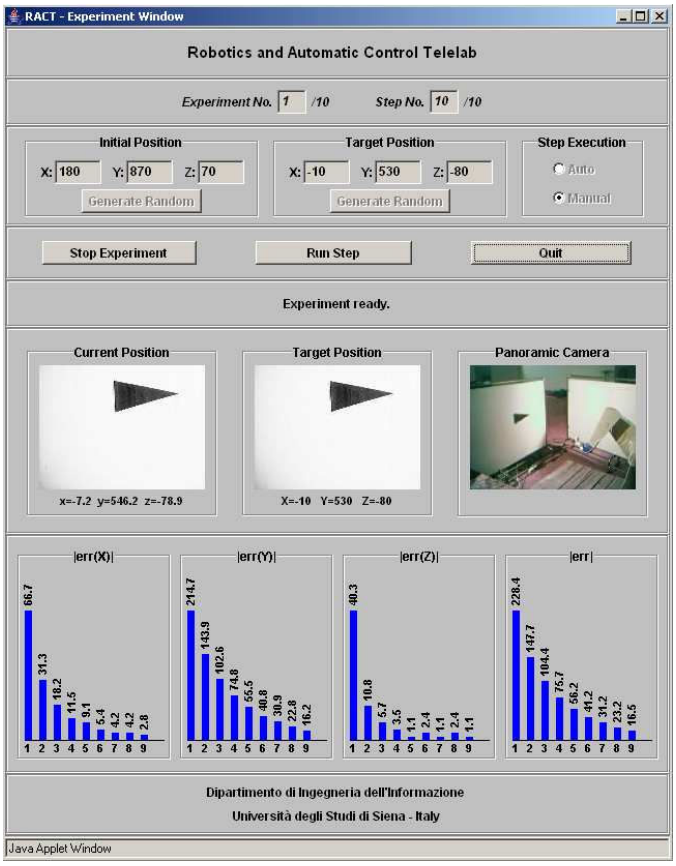

Fig. 4. The user interface for performing the visual servoing experiment. Data are in *mm*.

It is worthwhile to note that, although the target coordinates are reported in the interface (and eventually set by the user), they are unknown to the designed function, which takes as input just the current and target image.

Also in this case, a limit on the maximum number of experiments a user can execute in one session is fixed. In addition, the maximum number of executable steps for each experiment is also fixed. At the end, users can choose to download only the numerical values (in Matlab or ASCII format) or also the pictures acquired by the on-board camera for each step. Downloading the images can be useful to off-line test and calibrate the designed function.

3.3 Future experiments

Although the previously reported experiments are already fully working, the RACT is still under development. The following actions could be taken in future development of RACT:

- Other than inverse kinematics, an easier experiment regarding forward kinematics will be provided. In this case, given the six angles in the Denavit-Hartenberg convention, students will be asked to design a function which computes the corresponding end-effector position.
- In addition to translation, rotations along the axis orthogonal to the image plane should be provided. To provide more complex experiments, rotations along the three axes should be taken into account.
- Placing several panels in the robot workspace, it will be possible to set up various experiments, depending on the features reported on each panel. For example, one panel may contain a black triangle (as in Section 3.2), another one may contain some lines, another one may contain colored features, etc. In this way, it will be possibile to perform a high number of visual servoing experiments involving minor changes.
- Using new robotic manipulators to increase the capabilities of the system, in order to offer a wider range of experiments which cannot be provided by the Puma.

4. CONCLUSIONS

In this paper, a remote laboratory on robotics has been presented. The first two experiments implemented in the lab regard inverse kinematics and visual servoing. Other than educational purposes, the long-term goal of this lab is to provide remote tools for research aims. A remote lab on visual servoing will certainly attract the attention of many researchers in our community and for this reason a strong effort will be made to enrich the range of visual servoing experiments.

REFERENCES

J. Apkarian and A. Dawes. Interactive control education with virtual presence on the web. In *Proc. of IEEE American Control Conference*, pages 3985–3990, Chicago, IL, June 2000.

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, pages 1903–1908, Seoul, Korea, 2001.

F. Carusi, M. Casini, D. Prattichizzo, and A. Vicino. Distance learning in robotics and automation by remote control of LEGO mobile robots. In *Proc. Int. Conf. on Robotics and Automation*, pages 1820–1825, New Orleans, USA, Aprile 2004.

M. Casini, D. Prattichizzo, and A. Vicino. The Automatic Control Telelab: a user-friendly interface for distance learning. *IEEE Transactions on Education*, 46(2):252–257, 2003.

M. Casini, D. Prattichizzo, and A. Vicino. The Automatic Control Telelab: A web-based technology for distance learning. *IEEE Control Systems Magazine*, 24(3):36–44, 2004.

F. Chaumette and S. Hutchinson. Visual servo control. I. Basic approaches. *Robotics & Automation Magazine, IEEE*, 13(4):82–90, 2006.

F. Chinello. Simulazione e controllo per il manipolatore robotico PUMA 560. Master's thesis, University of Siena, June 2007. (in Italian).

G. Choy, D.R. Parker, J.N. d'Amour, and J.L. Spencer. Remote experimentation: a web-operable two phase flow experimnet. In *Proc. of IEEE American Control Conference*, pages 2939–2943, Chicago, IL, June 2000.

P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996.

S. Dormido. Control learning: present and future. *Annual Reviews in Control*, 28:115–136, 2004.

K. Goldberg. *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. Mit Press, 2000.

K. Goldberg, M Masha, S. Gettner, and N. Rothenberg. Desktop teleoperation via the world wide web. In *Proc. of the* 1995 *IEEE Int. Conference on Robotics and Automation*, 1995.

H.H. Hahn and M.W. Spong. Remote laboratories for control education. In *Proc. of* 39*th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation,*, 12(5):651–670, 1996.

T.F. Junge and C. Schmid. Web-based remote experimentation using a laboratory-scale optical tracker. In *Proc. of IEEE American Control Conference*, pages 2951–2954, Chicago, June 2000.

N. Kolban. Webcam32 - the ultimate webcam software, 2006. [online]: http://surveyorcorp.com/webcam32.

J.W. Overstreet and A. Tzes. An internet-based real-time control engineering laboratory. *IEEE Control Systems Magazine*, 19(5):19–34, October 1999.

S.E. Poindexter and B.S. Heck. Using the web in your courses: What can you do? what should you do? *IEEE Control Systems Magazine*, pages 83–92, Febraury 1999.

V. Ramakrishnan, Y. Zhuang, S.Y. Hu, J.P. Chen, C.C. Ko, Ben M. Chen, and K.C. Tan. Development of a web-based control experiment for a coupled tank apparatus. In *Proc. of IEEE American Control Conference*, pages 4409–4413, Chicago, IL, June 2000.

R. Safarič, M. Debevc, R.M. Parkin, and S. Uran. Telerobotics experiments via internet. *IEEE Transactions on Industrial Electronics*, 48(2):424–431, 2001.

R. Safarič, M. Truntič, D. Hercog, and G. Pačnik. Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering (iJOE)*, 1(1), 2005.

P. Saucy and F. Mondada. Khepontheweb: open access to a mobile robot on the internet. *IEEE Robotics and Automation Magazine*, 7(1):41–47, March 2000.

L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer, 2000.

M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons, Inc., 2006.

The Apache Software Foundation. Apache HTTP Server Project, 2007. [online]: http://httpd.apache.org/.

Unimation Robotics. PUMA MARK II robot 500 series: equipment and programming manual, February 1983.