A Matlab-based Remote Lab for Multi-Robot Experiments

Marco Casini * **Andrea Garulli** * **Antonio Giannitrapani** *
Antonio Vicino *

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

Abstract: Experimental validation is often crucial for establishing the effectiveness and robustness of algorithms for mobile robots. Unfortunately, it is generally difficult to let students use experimental setups, due to expensive hardware, complexity of usage and, not last, safety aspects. Often, experiments require the supervision of a tutor, and have to be carried out during the laboratory opening time. A possible way to overcome such limitations is to resort to remote labs. In this paper, a remote lab for controlling a team of mobile robots is presented. It is based on the Matlab environment and the mobile robots are built by using the LEGO Mindstorms NXT technology. Despite its low cost, thanks to its open architecture the proposed setup is versatile enough to be used in several kinds of experiments, ranging from single-robot to multi-robot systems, from centralized to decentralized control schemes. Users may connect to the remote lab and test their own algorithms in an easy way, by simply designing a Matlab function, without spending time on hardware aspects. A user interface allows them to observe the running experiment, and to download experimental data for offline analysis.

Keywords: remote labs, mobile robotics, LEGO Mindstorms, multi-vehicle systems

1. INTRODUCTION

Nowadays, the usage of remote labs for distance learning in the robotics and automation field is rapidly increasing (e.g., see Dormido [2004]). Differently from virtual labs, which provide software simulations of physical processes, remote labs allow users to remotely interact with real experiments, thus improving the realism and the appeal of analyzing the performance of different control laws. The importance of experimental testing is especially true in the mobile robotics field, where the validation on real data is crucial for assessing the effectiveness of a control technique. Unfortunately, performing this kind of experiments is often a difficult and expensive task, particularly when dealing with algorithms designed for teams of robots. This is confirmed also by the relatively few experimental results on multi-agent algorithms that can be found in the literature (Marshall et al. [2006], Ren and Sorensen [2008], Mastellone et al. [2008]). Therefore, allowing students to test multi-robot algorithms on real vehicles, while being highly instructive, may become a prohibitive task, especially in case of large classes.

This paper presents a remote lab for mobile robotics, based on the Matlab environment and the LEGO Mindstorms technology. By now, Matlab represents a standard tool in control education (Poindexter and Heck [1999]), thanks to its easy-to-learn and powerful language. The adoption of Matlab as main software tool has several advantages. It allows students to interact with the remote lab through a friendly environment, and to use a programming language they are likely familiar with. At the same time, this choice makes the interaction with the lab independent of the chosen robot technology and its low-level programming language, which is often a barrier to the experiencing from entry-level users. The mobile robots are built by using the LEGO Mindstorms NXT technology (The LEGO Group [2009]), which is widely used in control systems and robotics education, for its low-cost and simplicity. For instance, a LEGO-based experiment for teaching fundamental concepts of control and digital signal processing is reported in Heck et al. [2004], while a set of control experiences using LEGO are described in Gawthrop and McGookin [2006]. Regarding mobile robots built with LEGO, recent works can be found in Valera et al. [2008], where a distributed network-based control of mobile robots is described, and in Green et al. [2008], where a human can collaborate in a natural manner with a mobile robot.

The proposed remote lab is in its early stage, and work is still in progress. Although its main functionalities are ready, currently the lab is not available online yet. Once it is fully operative, the lab will be embedded into the Automatic Control Telelab (ACT), a remote laboratory developed at the University of Siena (Casini et al. [2004]) whose main goal is to allow students to put in practice their theoretical knowledge of control theory in an easy way and without restrictions due to laboratory opening time and process availability. Following the philosophy of the ACT, also the mobile robotics experiments will be freely accessible by everyone at the following web address: http://act.dii.unisi.it.

The paper is structured as follows. In Section 2 the proposed remote lab, along with its architecture, is described in detail. A typical user session is presented in Section 3. In

Section 4 possible experiments for teaching purposes are outlined, while in Section 5 some conclusions are drawn.

2. SETUP OVERVIEW

A first experimental setup, based on Matlab and LEGO, was developed at the University of Siena for the validation of cooperative control strategies for multi-vehicle systems (Benedettelli et al. [2009]). From that experience, it was clear that, while this architecture can be effectively used for ad-hoc teaching and research purposes, it needed significant improvements for its application to more general educational activities. In fact, the usage of this kind of setup requires a good knowledge of the overall system, and the presence of an instructor is often needed, thus limiting the usability from large, inexpert classes. To overcome these issues, it was planned to develop a remote lab to provide students with an easy and powerful tool to put in practice multi-robot control algorithms.

The chosen approach was to integrate the existing multi-robot system into the Automatic Control Telelab. In this way, by connecting to a server through a web page, a student uploads a Matlab function containing the control algorithm to be used during the experiment. A graphical interface allows the user to monitor the experiment and to download the recorded data for offline analysis. All these steps can be done without any knowledge of low-level hardware aspects. Moreover, the open architecture and the extreme versatility of the developed setup, makes it suitable for practicing with a number of different problems related to mobile robots (see Section 4 for an overview of possible teaching applications).

Besides the remote interaction, the setup described in this paper differs from that previously developed in Benedettelli et al. [2009] in at least two aspects. First, the mobile robots have been completely redesigned, in order to fully exploit the improvements provided the new LEGO NXT technology. Second, two wide-angle cameras are now used to detect the robots. This results in a larger experimental arena, at the price of more complex image processing.

In the following, the hardware and software architecture of the proposed remote lab are described in detail.

2.1 Hardware architecture

In Fig. 1 a sketch of the hardware architecture is depicted. Four main components can be identified: the team of mobile robots, a desktop PC acting as main server, two wide-angle cameras for robot detection, and a webcam used for the video streaming.

NXT mobile robots. The mobile robots are built by using the LEGO Mindstorms NXT technology. The vehicles are identical, and have been designed to be as compact as possible. The core of each vehicle is a NXT brick, which hosts a microprocessor providing computation capabilities, motor actuation, and Bluetooth communication. The robots feature a differential-drive kinematics, with two servomotors driving independently the right and left wheel. A steel ball transfer unit acts as third support, and contributes to lower the vehicle center of mass. This choice, together with a robust mechanical design, prevents the vehicles

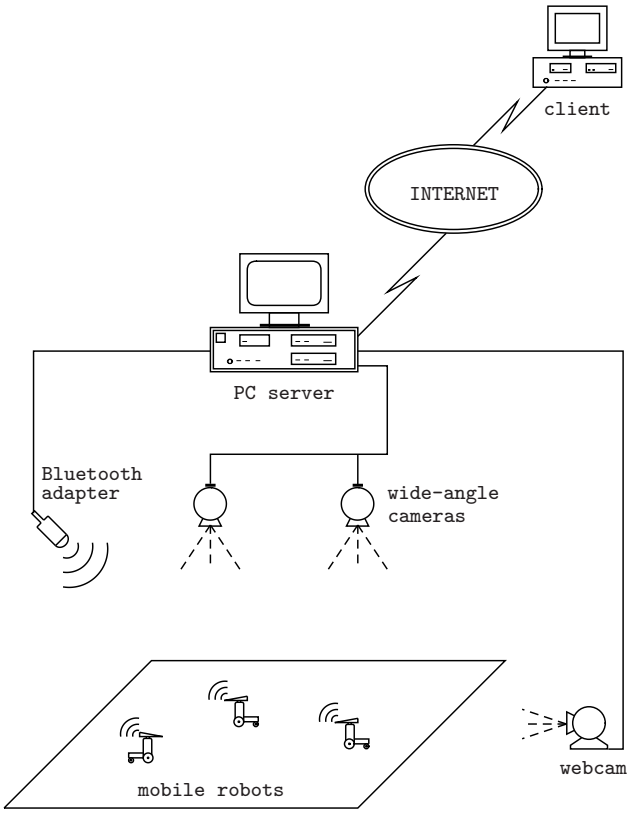


Fig. 1. Hardware architecture.

from overturn in case of collisions. A triangular marker (different in size for each robot) is placed on the top of each vehicle to allow visual detection through cameras. No sensors are present on the robots, except for incremental encoders used to control the wheel speed. Currently, the team is composed of three robots. A picture of a robot is reported in Fig. 2.

Fig. 2. The LEGO NXT mobile robot.

PC server. A desktop PC represents the core of the project, providing the necessary processing power and communication capabilities. An instance of Matlab runs

in background to perform all computations. The PC is connected to the Internet to accept connection to the web server it hosts. Through a Bluetooth adapter the PC communicates with the robots. Moreover, it is connected to two wide-angle cameras for vehicle detection, and to one webcam used for online video streaming.

Wide-angle cameras. Two USB wide-angle cameras are placed on the lab ceiling, pointing downward. They are used to detect the robots and to estimate their pose. These cameras are low-cost webcams featuring a field of view of about 71 degrees, and have been chosen to increase the robot workspace, while still maintaining a good image resolution. It is a fact that experimenting with multi-vehicle algorithms needs in general larger workspace. In the current configuration the experimental arena is about $3.5m \times 4.5m$.

Internet webcam. An additional webcam is connected to the server to provide online streaming video of the experiment. As in other remote labs, the online video, in addition to experiment signals, helps increasing the sense of presence inside the lab.

It is worth remarking that the remote lab has been developed having in mind, among other criteria, a good trade-off between costs and performance. For this reason, it was decided to use only standard hardware (a desktop PC, USB webcams, LEGO), rather than more technologically advanced and expensive solutions (like, e.g., professional wide-angle cameras connected to the PC through a frame grabber).

2.2 Software architecture

In this subsection, some details concerning the software implementation are illustrated.

A web server, accepting incoming connections from users, runs on the PC server. The PHP extension and a webcam server are also installed to dynamically generate web pages and to provide online video, respectively. A Matlab script supervises all the following tasks, to coordinate and monitor an experiment.

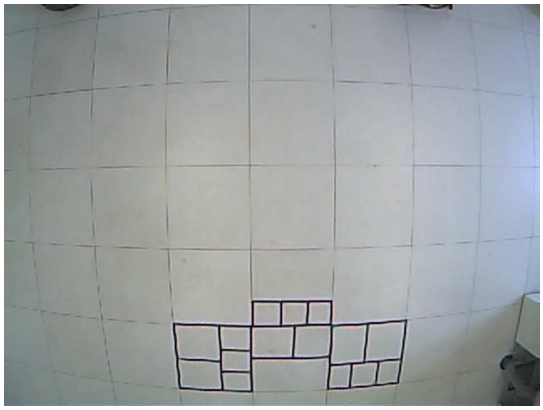
- *High-level robot control.* Due to the limited processing capacity of NXT, it has been chosen to perform all the computations regarding the algorithms to be tested on the PC server. This solution has several advantages. In fact, the usage of Matlab for implementing control algorithms allows one to use a wide set of powerful functions already provided in Matlab and its toolboxes. Moreover, the high-level Matlab language reduces the time needed for developing algorithms with respect to other languages. The user-defined Matlab function is used to compute the references of robot linear and angular speed according to an easy syntax (see Section 3 for details). The sample time used for updating robot speed references has been set to 0.5 seconds.
- *Robot detection.* By means of the two top cameras, the server is able to detect the position and the orientation of each robot. During the experiment, the images of the cameras are captured through the Matlab *Image Acquisition Toolbox* (The Mathworks, Inc. [2009a]). Next, the images are filtered to extract the triangular markers corresponding to each robot. This step is accomplished by exploiting functions for feature extraction provided by the Matlab *Image Processing Toolbox* (The Mathworks, Inc. [2009b]). Since the acquired images are affected by a significant barrel distortion, the estimation of the robot pose may be inaccurate. For this reason, an offline calibration procedure has been previously performed, leading to an estimate of the internal parameters of the cameras, by means of the Matlab *Camera Calibration Toolbox* (Bouguet [2008]). Finally, the robot poses extracted from the two images are merged together through a registration process, in order to transform local coordinates into the global reference frame. To this purpose, the homogeneous transformation relating the frame of the two cameras has been previously estimated offline (Hartley and Zisserman [2003]). In Fig. 3, the result of the distortion correction and image registration are shown. Figs. 3(a) and 3(b) show the raw captured images, while in Fig. 3(c) the final corrected and registered image is depicted.
- *Robot communication.* Through the wireless Bluetooth adapter the server can communicate with NXTs. To this purpose the *RWTH - Mindstorms NXT Toolbox* for Matlab has been used (Institute of Imaging & Computer Vision - RWTH Aachen University [2009]). This connection is used for sending the desired linear and angular speed to the vehicles, and for receiving control signals like the battery levels of each robot.
- *Data exchange.* Through a TCP connection, users can send the start/stop experiment signals, while during the experiment the server communicate to users the vehicle positions.
- *Safety system.* To prevent robots from getting lost, whenever a robot is about to going outside the workspace, the experiment is stopped and vehicles come back to their initial position.

The speed control of the vehicles is demanded to a program running on the NXT brick of each robot.

- *Low-level robot control.* Speed control is in charge of a program written in *Not eXactly C (NXC)* running on the microprocessor of each vehicle (Hansen [2007]). It is composed of two parallel tasks. The first one is devoted to check whether there is an incoming message from the Bluetooth connection. Once a message is received, it is parsed and the linear and angular speed references are passed to the second task. Such a task implements the actual low-level control of the right and left wheel speed. Although the NXT is already equipped with a built-in PID control for motor control, an ad-hoc incremental PI controller has been implemented in order to obtain improved performance.

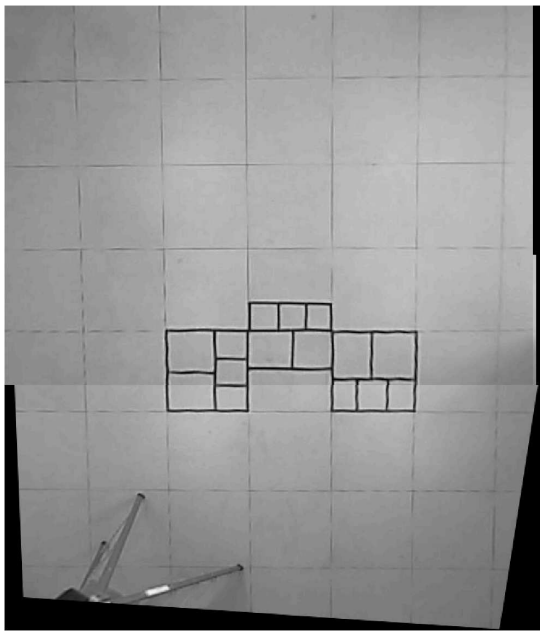
The last component of the software architecture is a remote client which is used by students to interact with the lab.

- *Client.* A user connects to the remote lab by an Internet browser. Web pages are HTML pages generated by PHP scripts, while the graphical interface is a



(a)

(b)



(c)

Fig. 3. Camera calibration and image registration: raw images (a) and (b), and final image (c).

Java applet which ensures platform independence. In Fig. 4, a snapshot of the user interface during an experiment is reported. By means of the buttons placed on the top-right side, users can start and stop

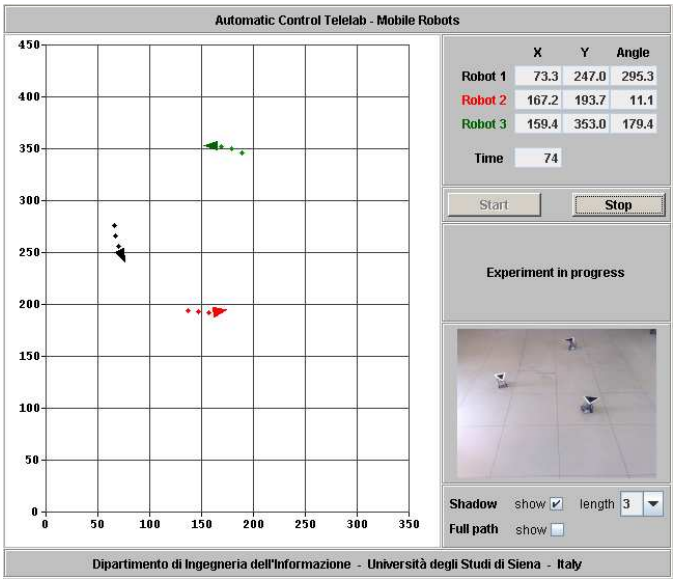


Fig. 4. The user interface.

the experiment. The workspace along with the robot positions are shown in the left-hand pane and are updated every second. Finally, the video streaming is placed on the bottom-right side.

3. SESSION DESCRIPTION

This section illustrates an overview of a typical user session.

After connecting to the ACT through a common web browser, the user is offered a set of different physical processes on which perform control and system identification experiments. After selecting the "Team of mobile robots" experiment, the user is requested to upload a Matlab function which will be used during the experiment to control the robot team. In fact, user interaction takes place through this function, which returns the control inputs for each vehicle, based on the current state of the team. During the experiment, this function is invoked by the supervisor script at each sampling time, in order to update the speed reference signals to be sent to the NXTs controlling the vehicles.

Before describing the input and output parameters of such a function, it is worth recalling the kinematic model of the built vehicles. Let $p_i(t) = [x_i(t)\ y_i(t)\ \theta_i(t)]'$ be the position and orientation of the i-th robot at time t (see Fig. 5), then the robot pose evolves according to the unicycle model

$$\begin{aligned} \dot{x}_i(t) &= v_i(t)\cos(\theta_i(t)) \\ \dot{y}_i(t) &= v_i(t)\sin(\theta_i(t)) \qquad i = 1, \ldots, N \\ \dot{\theta}_i(t) &= \omega_i(t) \end{aligned}$$

where $v_i(t)$ and $\omega_i(t)$ denotes the linear and angular speed of the vehicle and play the role of control inputs. The user function must be designed according to the following syntax

```
function [v, w] = Robot_Control(t, P)
```

Concerning the input parameters, `t` is the elapsed time since the beginning of the experiment (in seconds), and `P` is a $3 \times N$ matrix, whose i-th column contains the current pose $p_i(t)$ of the i-th robot (in meters and radians). The output parameters `v` and `w` must be $1 \times N$ vectors containing the desired linear and angular speed of each robot (expressed in m/s and rad/s). Currently, the team is composed of $N = 3$ robots, but work is in progress to add further vehicles. In order to increase the versatility of the whole setup, no collision avoidance is performed by the supervisor script, rather, it is left to the user to check and prevent any robot collision within the `Robot_Control` function. Anyway, as previously mentioned, thanks to robust robot design, collisions are tolerated by the system.

Since the `Robot_Control` function is provided by the user, it may contain syntax errors or it may drive the robots outside the workspace. In the former case, errors can be easily detected by Matlab and consequently the experiment is not performed. In the latter case, the experiment is stopped whenever one robot is going outside the workspace.

Once the Matlab function has been uploaded, the main user interface shows up (see Fig. 4). From there, the user can start and stop the experiment, as well as monitoring the evolution of the team over time. At the end, it is possible to download the data collected during the experiment (as a MAT-file), for later analysis. At each sampling time, the robot poses and the control inputs are recorded.

The results of a simple experiment involving three robots are summarized in Fig. 6. Here, *Robot #1* must reach a desired position while at the same time avoiding collisions with *Robots #2* and *#3*. The latter two vehicles move on a square and a circle, respectively, at a constant linear speed of 0.03 m/s, while *Robot #1* is faster (0.06 m/s) to improve its maneuverability.

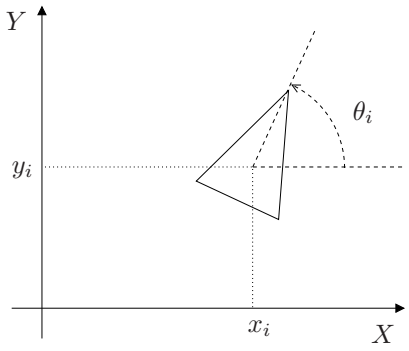


Fig. 5. Robot pose.

4. EDUCATIONAL APPLICATIONS

Once the project is online, it is planned to use it for educational and research purposes. It can be effectively used by students in robotics courses for experimenting a number of control algorithms. At present, it is quite difficult to let students apply theoretical knowledge of mobile robotics on real testbeds, especially for courses with several students. So, it is common to ask students to perform simulations instead of testing algorithms on

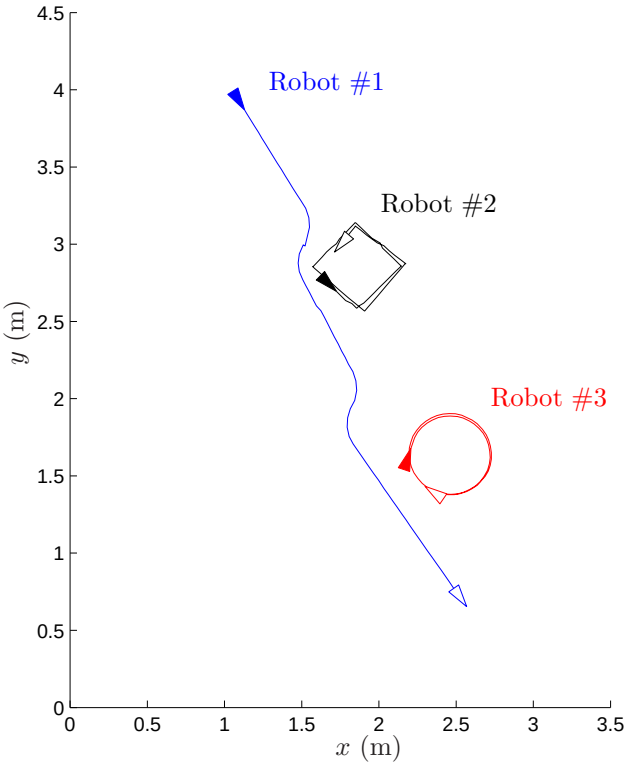


Fig. 6. Robot paths during an experiment: filled triangles denote the initial robot poses, empty triangles denote the final robot poses.

real devices. However, it is well known how important is to validate such algorithms on an experimental setup. By using this remote lab, it is possible to assign students different tasks to be performed on real vehicles in an easy way and from any computer connected to the Internet.

It is worth remarking the versatility of the chosen architecture. Since at each time instant the state of the whole team is available, it is possible to implement both centralized and decentralized control schemes (e.g., using only a subset of the available information in the computation of control inputs). The mobile robots are not equipped with any kind of exteroceptive sensors. Nonetheless, advanced users can implement sort of *virtual sensors*, i.e. simulate via software the presence of onboard sensors from the current state of the team (e.g., range and bearing measurements among vehicles). Different kinds of control laws can be compared. For instance, it is pretty straightforward to implement static feedback controllers, since the state of the system is always available. At the same time, it is also possible to test dynamic controllers, by exploiting persistent variables provided by the Matlab environment.

In the following, a brief description of some experiments which can be performed with this lab is reported.

- *Single-robot experiments.* A student can experiment with single-robot control algorithms by simply setting to zero the speeds of the other vehicles and ignoring their poses. In this way, it is possible to test algorithms on path planning, obstacle avoidance, robot maneuvers, etc. Of course, it is up to the user to define the needed objects inside the `Robot_Control`

function. For instance, for an obstacle avoidance algorithm, the shapes and locations of obstacles must be defined in this function. Moreover, this function has to compute distances from obstacles and check collisions.

- *Multi-robot experiments.* Algorithms similar to that reported in the previous point may be tested in a multi-robot scenario. In this case, for instance, one can design a controller to avoid robot collisions while performing a given task.
- *Cooperative control experiments.* It is possible to use the team of vehicles in a cooperative way to test algorithms of formation control, motion coordination, coverage problems, etc.
- *Competitive control experiments.* Robots may compete each other in order to perform a given task. For instance, a typical example of competitive experiment consists in the pursuit-evader problem.

It is worth remarking that the task to be executed is chosen by the user through the function `Robot_Control` described in Section 3. Such a function can be easily used to simulate the experiment behavior through a provided Matlab simulator. So, students are encouraged to simulate their algorithms before testing them on the actual experimental setup, and comparing the obtained results.

5. CONCLUSIONS

In this paper, a remote lab for controlling a team of mobile robots has been presented. Vehicles are built by using the LEGO Mindstorms NXT technology. Through an Internet connection, a user can upload a Matlab function implementing the control law to be tested. A graphical interface, along with a video streaming, shows the evolution of the experiment, and allows the user to download the experimental data for offline analysis. Thanks to the open architecture of the lab, it is possible to test algorithms for several mobile robotics problems. The proposed low-cost setup features high versatility, and is suitable for practicing with single-robot and multi-vehicle scenarios, as well as with both centralized and decentralized control schemes.

To make the lab available 24 hours a day for remote experiments, current work is being done for equipping each robot with two metal plates for automatic battery recharge inside a *recharge box*, similarly to what done in Carusi et al. [2004]. The usage of the LEGO lithium rechargeable battery will simplify this task. At the same time, together with the enlargement of the experimental area, it is expected to increase the number of robots, in order to test algorithms involving larger team. In the future, it is planned to enrich the user interface with additional features, like the definition of virtual sensors or virtual obstacles.

ACKNOWLEDGEMENTS

The authors would like to thank Roberto Zingone for his contribution to the development of the vision system.

REFERENCES

D. Benedettelli, M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A LEGO Mindstorms experimental setup for multi-agent systems. In *Proc. of Int. Conf. on Control Applications (CCA)*, pages 1230–1235, Saint Petersburg, Russia, July 2009.

J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, 2008. [online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/.

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

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.

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

P.J. Gawthrop and E. McGookin. Using LEGO in control education. In *Preprints of 7th IFAC Symposium on Advances in Control Education*, Madrid, Spain, June 2006.

S. A. Green, X. Chen, M. Billinghurst, and J. G. Chase. Collaborating with a mobile robot: An augmented reality multimodal interface. In *Proc. of 17th IFAC World Congress*, pages 15595–15600, Seoul, South Korea, July 2008.

J. Hansen. Not eXactly C (NXC) - Programmer's Guide, 2007. [Online]. Available: bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

B. S. Heck, N. S. Clements, and A. A. Ferri. A LEGO experiment for embedded control system design. *IEEE Control Systems Magazine*, 24(5):61–64, 2004.

Institute of Imaging & Computer Vision - RWTH Aachen University. RWTH - Mindstorms NXT Toolbox, 2009. [Online]. Available: http://www.mindstorms.rwth-aachen.de.

J. A. Marshall, T. Fung, M. E. Broucke, G. M. T. D'Eleuterio, and B. A. Francis. Experiments in multirobot coordination. *Robotics and Autonomous Systems*, 54(3):265–275, 2006.

S. Mastellone, D.M. Stipanovic, C.R. Graunke, K.A. Intlekofer, and M.W. Spong. Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments. *The International Journal of Robotics Research*, 27(1):107–126, 2008.

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.

W. Ren and N. Sorensen. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems*, 56(4):324–333, 2008.

The LEGO Group. The Lego Mindstorms home page, 2009. [Online]. Available: http://mindstorms.lego.com.

The Mathworks, Inc. Image Acquisition Toolbox 3 - User's guide, 2009a.

The Mathworks, Inc. Image Processing Toolbox 6 - User's guide, 2009b.

A. Valera, M. Vallés, P. Albertos, R. Simarro, I. Benítez, and C. Llácer. Embedded implementation of mobile robots control. In *Proc. of 17th IFAC World Congress*, pages 6821–6826, Seoul, South Korea, July 2008.