A LEGO Mindstorms multi-robot setup in the Automatic Control Telelab

Marco Casini, Andrea Garulli, Antonio Giannitrapani, Antonio Vicino

Dipartimento di Ingegneria dell'Informazione
Via Roma, 56 - 53100 Siena - ITALY
Email: {casini,garulli,giannitrapani,vicino}@ing.unisi.it

Abstract: This paper presents an experimental setup for multi-robot systems based on the LEGO Mindstorms NXT technology. The team of mobile robots is supervised by a vision system, which allows one to simulate different types of sensors and communication architectures. The whole setup is embedded in the Automatic Control Telelab, a remote lab featuring several educational experiences in control. Remote users can design control laws for the multi-agent system in the Matlab environment and test them by performing real experiments in the proposed setup. The paper presents several experiments showing how this setup can stimulate students' interest in mobile robotics.

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

1. INTRODUCTION

Education in the automatic control field has undergone significant changes in the last decade. The diffusion of new technologies at affordable price has enriched the way subjects like control systems or robotics are being taught in engineering curricula. Low-cost devices, highly customizable and easy to program, are frequently adopted for setting up hands-on experiments which complement traditional theoretical classes. In this respect, the LEGO technology has a leading position, thanks to its low-cost and simplicity (e.g., see Gawthrop and McGookin [2004], Fradkov and Albertos [2009]). Originally designed as an educational toy for children, by now LEGO Mindstorms kits have been used to build a variety of experimental setups falling in very different areas ranging from signal processing (Heck et al. [2004]), to mechatronics (Valera et al. [2009]), cyber-physical laboratories (Filippov and Fradkov [2009]), to name just a few.

At the same time, impressive innovations in the educational field have been brought in by the Internet and the consequent emergence of a number of on-line tools, allowing teachers to put in practice new pedagogical patterns for enhancing the learning of technical disciplines. Concerning the robotics and automation field, a major role is played by remote laboratories whose employment for distance learning is becoming increasingly widespread (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 stimulating students' interest in implementing, testing and comparing different control laws.

The importance of experimental validation is especially true in robotics. For instance, testing on real data algorithms designed for mobile robots, supposedly operating in real-world environments, is crucial for a correct evaluation of the performance. Unfortunately, carrying out real experiments can be a difficult and expensive task, especially when dealing with teams of robots. This is confirmed also by the relatively few experimental results on multi-agent algorithms that can be found in the literature (Ren and Sorensen [2008], Mastellone et al. [2008], Benedettelli et al. [2010]). As a consequence, allowing students to test multi-robot algorithms on real vehicles, while being highly instructive, may become a daunting task.

This paper presents a remote lab for mobile robotics able to enjoy all the benefits of practicing with real robots, while at the same time minimizing the amount of work required to build a fully functional setup for multi-agent systems. The proposed lab is based on the Matlab environment and the LEGO Mindstorms technology. A graphical interface, accessible by standard web browsers, allows users to select one out of a set of predefined experiments, or to test ad-hoc controllers by uploading a Matlab function. During the experiment, visual feedback is ensured by video streaming, and a live reconstruction of the robot paths is depicted in a dedicated window. At the end, all the relevant data can be downloaded for off-line analysis. An automatic recharge station (built on purpose) makes the system available 24 hours per day, without the need of any human operator. A beta version of this setup is currently embedded into the *Automatic Control Telelab* (ACT), a remote laboratory formerly developed at the University of Siena and targeted at students of control systems willing to put in practice their theoretical knowledge on several physical processes (Casini et al. [2004]).

The paper is structured as follows. In Section 2 an overview of the proposed remote lab is reported, while the main features are described in details in Section 3. In Section 4 a set of experiments performed within the developed setup is reported, while in Section 5 some conclusions are drawn.

2. SETUP DESCRIPTION

The experimental setup consists of four mobile robots which can move in a workspace of approximately 4.5×3 meters. Robots are built with LEGO NXT components and they are equipped with two servomotors able to independently drive the left and right wheel, and a steel ball transfer unit acting as third support. A *hat* with special markers has been placed on the top of each vehicle for detecting the robot pose (see Figure 1). The position and orientation of the robots are extracted from images taken by two-wide angle cameras connected to a desktop PC (the *PC server*). While high-level control laws are computed by the PC server and the desired linear and angular speed are sent to robots through a Bluetooth connection, low-level speed control for each robot wheel is managed by a controller coded in NXC (Hansen [2007]) and embedded into the NXTs.

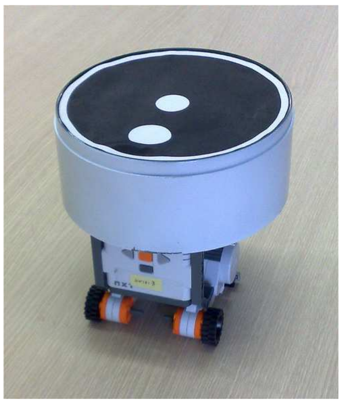


Fig. 1. The LEGO NXT mobile robot.

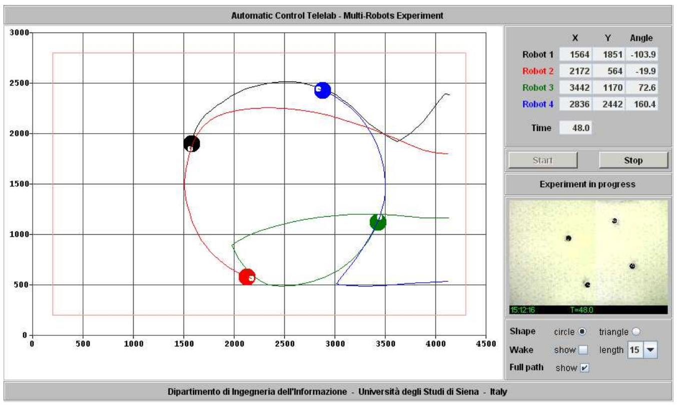
Interaction with the remote user is provided through web pages and a GUI implemented as a Java applet, which allows one to start/stop the experiment as well as to view the experiment behavior by means of a graphical panel and an on-line camera (Figure 2). User-defined controllers consisting in a Matlab function can be uploaded and run by using a specific interface (see Section 3).

Since the system must be available 24 hours a day, a crucial aspect regards the automatic recharge of robots batteries. To this purpose, once an experiment is over, a procedure drives the robots to a "box" able to recharge their lithium batteries without any human intervention, thanks to two metal plates placed on the bottom of each robot. A more detailed description of the system setup can be found in Casini et al. [2009].

3. FEATURES OVERVIEW

In this section, an overview of some features of the multi-robot remote lab is reported.

Although this facility can be used in research contexts (see Section 4 for research driven experiments), the main aim of this project is educational; to this purpose, it is essential to provide easy-to-use tools to users, in order to allow them to concentrate on a given task rather than spending time in understanding how to use the system. So, efforts have been directed to simplify user interaction, by allowing students to write controllers as a Matlab function and to run them



Automatic Control Telelab - Multi-Robots Experiment

	X	Y	Angle
Robot 1	1564	1851	-103.9
Robot 2	2172	564	-19.9
Robot 3	3442	1170	72.6
Robot 4	2836	2442	160.4

Time 48.0

Experiment in progress

Shape: circle [✓] triangle
Wake: show ☐ length 15
Full path: show ☑

Dipartimento di Ingegneria dell'Informazione - Università degli Studi di Siena - Italy

Fig. 2. The graphical user interface allowing interaction with the experiment.

through a very simple web interface. In addition, some useful tools (simulator and player) can be downloaded to speed up controller synthesis and performance analysis.

Before describing how users can perform experiments, it is useful to report the kinematic model of the robots. Let $p_i(t) = [x_i(t)\ y_i(t)\ \theta_i(t)]'$ be the position (m) and orientation (rad) of the i-th robot at time t (see Figure 3), then the robot pose evolves according to the unicycle model

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

where $v_i(t)$ and $\omega_i(t)$ denote the linear (m/s) and angular speed (rad/s) of the vehicle. Users who want to design their own controllers have to write a Matlab function which returns the values $v_i(t)$ and $\omega_i(t)$ of each robot, as it will be explained in the following.

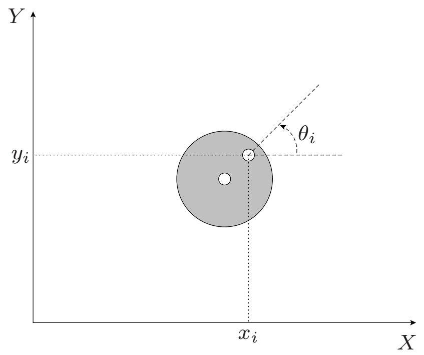
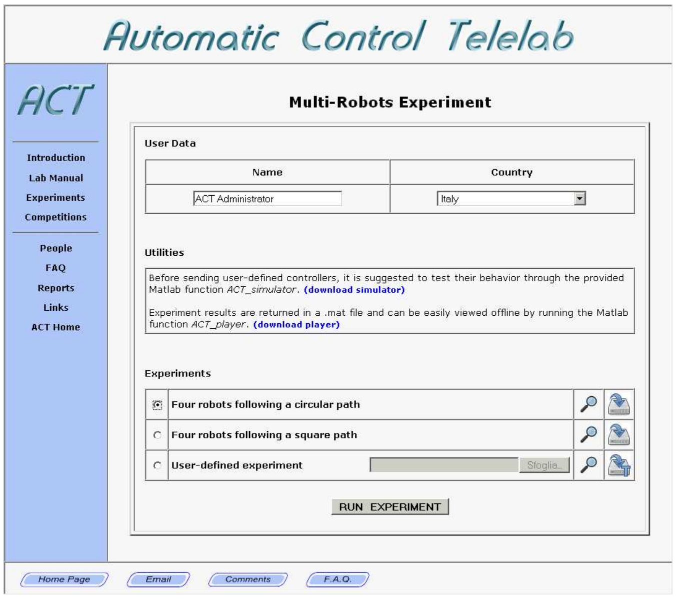


Fig. 3. Sketch of robot pose. The shape of the robot reflects the top view of a real vehicle.

User-defined experiments. Through the *Control Interface* (Figure 4), users may select the experiment to perform by choosing between a set of predefined experiences or by designing their own experiment. Predefined experiments have been provided both for showing the capabilities of the proposed remote lab and for helping users design their own tests. To perform a custom experiment, users have to download and modify a template file (Figure 5). This file is a Matlab function which is invoked at each sampling time and play the role of controlling robots. Here, the input parameter `Time` denotes the time instant when the routine is called, `Ts` is the sampling time, and `Pose` is a matrix whose columns contain the pose (x, y, θ) of each vehicle. The output arguments are the number `N_robot` of robots involved in the experiment and their initial pose `Pose_init` (used only during the experiment initialization), and a matrix `U` containing the desired linear and angular speed of each robot.



Multi-Robots Experiment

User Data

Name	Country
ACT Administrator	Italy

Utilities

Before sending user-defined controllers, it is suggested to test their behavior through the provided Matlab function *ACT_simulator*. **(download simulator)**

Experiment results are returned in a .mat file and can be easily viewed offline by running the Matlab function *ACT_player*. **(download player)**

Experiments

[✓ Four robots following a circular path]
Four robots following a square path
User-defined experiment   Sfoglia...

RUN EXPERIMENT

Fig. 4. The *Control Interface*.

```
function [U N_robot Pose_init]
               =ACT_MR_template(Time,Pose,Ts)

  N_robot=4; % number of robots

  %-- initial position (0=default position)
  Pose_init=[3.92 3.92 3.92 3.92
             2.43 1.83 1.20 0.57
               pi   pi   pi   pi];

  U=zeros(2,N_robot);  % robot commands
end
```

Fig. 5. Matlab code of template function for designing a user-defined controller.

A simple example of user-defined function is reported in Figure 6 to demonstrate the simplicity of this tool. Here, after setting an appropriate starting pose, four vehicles have to follow a circular trajectory of a given radius in a certain time, without using position feedback. To do this, the same linear and angular speed is given at each robot (the two rows of matrix U denotes the linear and angular speed, respectively, while columns denote robots identities).

Range of possible experiments. Since the user-defined Matlab function described above is completely open, users are able to perform several kinds of experiments, ranging from single-robot to multi-robot, from centralized to decentralized control. Although robots are not equipped with exteroceptive sensors, users may easily implement different types of virtual sensors starting from the global knowledge of the robot poses. For instance, it is possible to simulate range and bearing measurements among robots as in the case of on-board sensors like e.g., sonars and/or laser range finders.

```
function [U N_robot Pose_init]
                       =circle(Time,Pose,Ts)

  N_robot=4;  % number of robot used

  cx=2.25; % center of circle X-coord (m)
  cy=1.5;  % center of circle Y-coord (m)
  r=0.8;   % radius of circle (m)
  p=100;   % period (s)

  % initial pose
  Pose_init=[cx+r     cx   cx-r     cx
               cy   cy+r     cy   cy-r
             pi/2     pi  -pi/2      0];

  w=2*pi/p;   % angular speed (rad/s)
  v=2*pi*r/p; % linear speed (m/s)

  % robot speed commands
  U=[v v v v   % linear speed (m/s)
     w w w w]; % angular speed (rad/s)
end
```

Fig. 6. Example of user-defined function.

Simulator. A Matlab function simulating the robots kinematics is available for download from the *Control Interface* (Figure 4). Here, the continuous-time model (1) has been discretized and implemented. This function is particularly useful to have preliminary information about the behavior of a user-defined controller before uploading and testing it on the real setup. Since this simulator takes as input a Matlab function with exactly the same structure as the one to be uploaded, users do not need to make any change in their code to perform simulations. A graphical animation of the robot motion is also provided. At the end of the simulation, relevant data is returned to perform a posteriori analysis.

Experiment player. When a remote experiment is over, data can be downloaded as a Matlab workspace file (.mat). For each time step, significant data like robot poses and input commands are provided. To help the user evaluate the behavior of a performed experiment, an *experiment player* is provided as a Matlab function able to graphically reproduce the real behavior from the downloaded data. In addition to reproduction, it is also possible to use it for creating an animation of the experiment. All the experiment representations depicted in the next section have been created by this tool.

4. ILLUSTRATIVE EXAMPLES

In order to illustrate the potential of the proposed setup for teaching and research purposes, in this section we

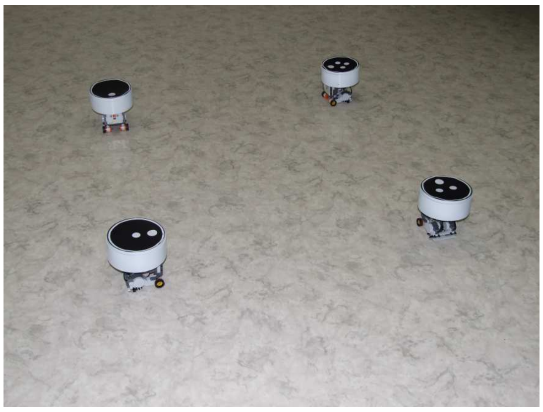
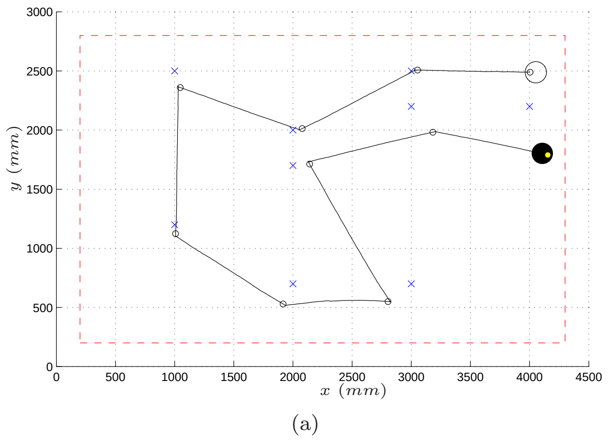


Fig. 7. The team of LEGO mobile robots.



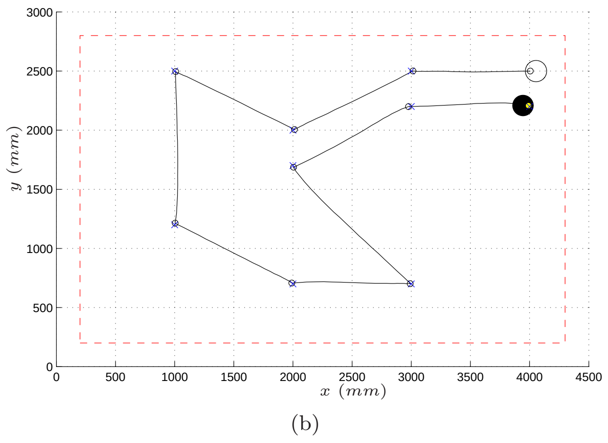


Fig. 8. Experiment A: open-loop control law (a), closed-loop control law (b). Initial (empty disc) and final (filled disc) robot position; desired way points ($\times$) and corresponding robot positions ($\circ$); actual robot path (solid line).

will present a selection of real experiments ranging from simple single-robot control to more complex decentralized architectures for the collective motion of a multi-agent team (see Figure 7). All the figures in this section have been created from data gathered at the end of each experiment.

4.1 Experiment A: Single-robot control

The first experiment is a very basic control problem suitable for an entry-level robotics course, involving only one robot. The user is given a sequence of way points (marked with a cross in Figures 8(a)-8(b)), and the objective is to design a control law driving the robot to pass through them. This simple problem is well suited for showing the superiority of feedback control laws over open-loop ones. For instance, Figure 8(a) shows the results of a naive turn-and-go strategy, precomputed beforehand. The robot motion is decomposed into a sequence of two basic actions, namely: i) turning at constant rate, and ii) going forward at constant speed. Then, given the desired way points, and the chosen forward and angular speed, the appropriate action is selected for each sampling time. Obviously, this open-loop controller exhibits a very poor performance, with a tracking error growing unboundedly over time. The system behavior can be significantly improved by resorting to a feedback control scheme, which takes into account the error between the next way point to reach and the current robot pose, as provided by the vision system. A control law, consisting of two independent proportional controller for the forward and angular speed, has been implemented and tested. Now, the effects of a number of disturbances like wheel slippage, uneven terrain, actuator nonlinearities, are well compensated for by the closed-loop system, as it is clearly visible in Figure 8(b).

4.2 Experiment B: Multi-robot centralized control

The second experiment is more complex than the previous one, requiring the control of all the four vehicles of the team. Starting from an arbitrary initial configuration, each robot must first reach a specified vertex of a square. Then, the vehicles must simultaneously move to the adjacent vertex (in counter-clockwise direction), like baseball players on the diamond. In this case, a number of issues arising from the interaction of multiple agents must be tackled, like e.g. collision avoidance. This scenario can be effectively adopted for practicing with high-level strategy of mission planning and motion coordination. The results summarized in Figures 9(a)-9(c) refer to an experiment performed with a centralized controller, which is responsible for the overall team coordination, as well as for the motion control of each vehicle. Given the state of the system, i.e. position and orientation of the robots, the controller takes care of planning the next point to be reached by the agents, synchronizes the motion of the vehicles, and finally computes the velocity commands to be sent to the robots. It is worth remarking that all these tasks can be accomplished just by uploading a single Matlab function.

4.3 Experiment C: Cyclic pursuit

The developed setup can be exploited also for more advanced multi-robot experiments, like those required for testing ongoing research work. As an example, the third experiment presented in this section concerns the well-known cyclic pursuit problem, where each robot has to follow the next one (modulo n). Differently from what happened in Experiment B, this time the team coordination must be achieved in a decentralized fashion, without

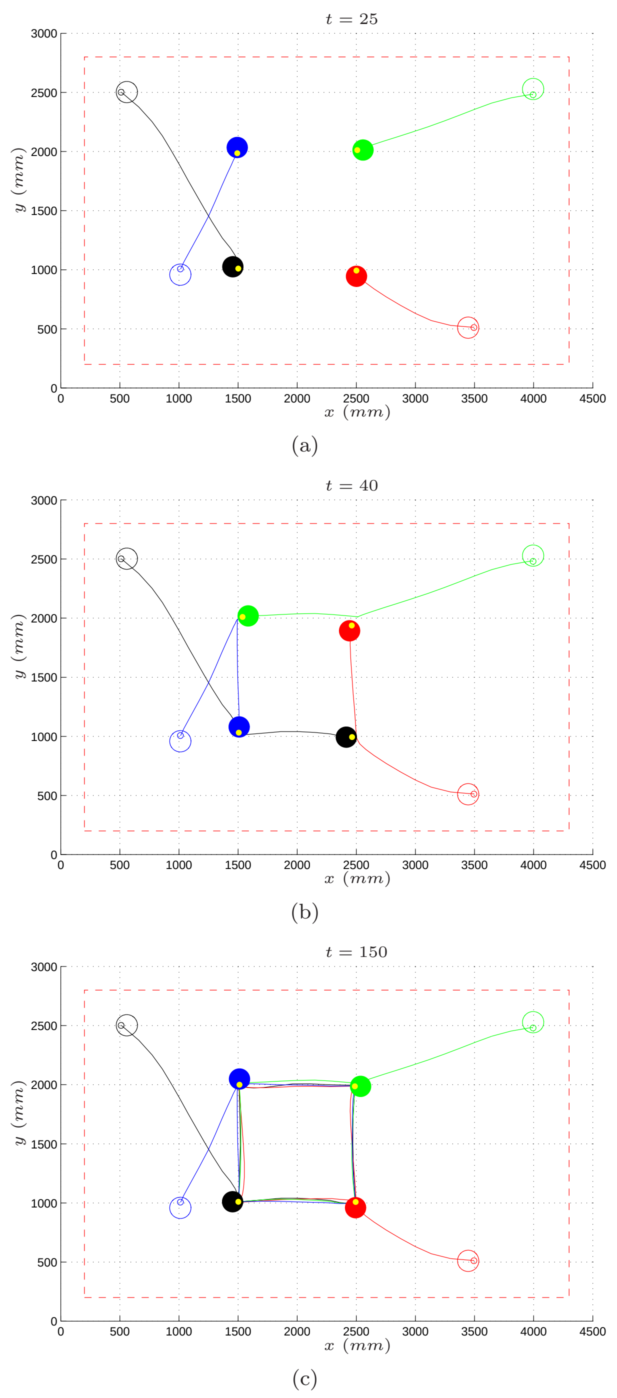


Fig. 9. Experiment B: three snapshots taken at time $t = 25$ s (a), $t = 40$ s (b) and $t = 150$ s (c). Initial (empty disc) and current (filled disc) robot position; actual robot path (solid line).

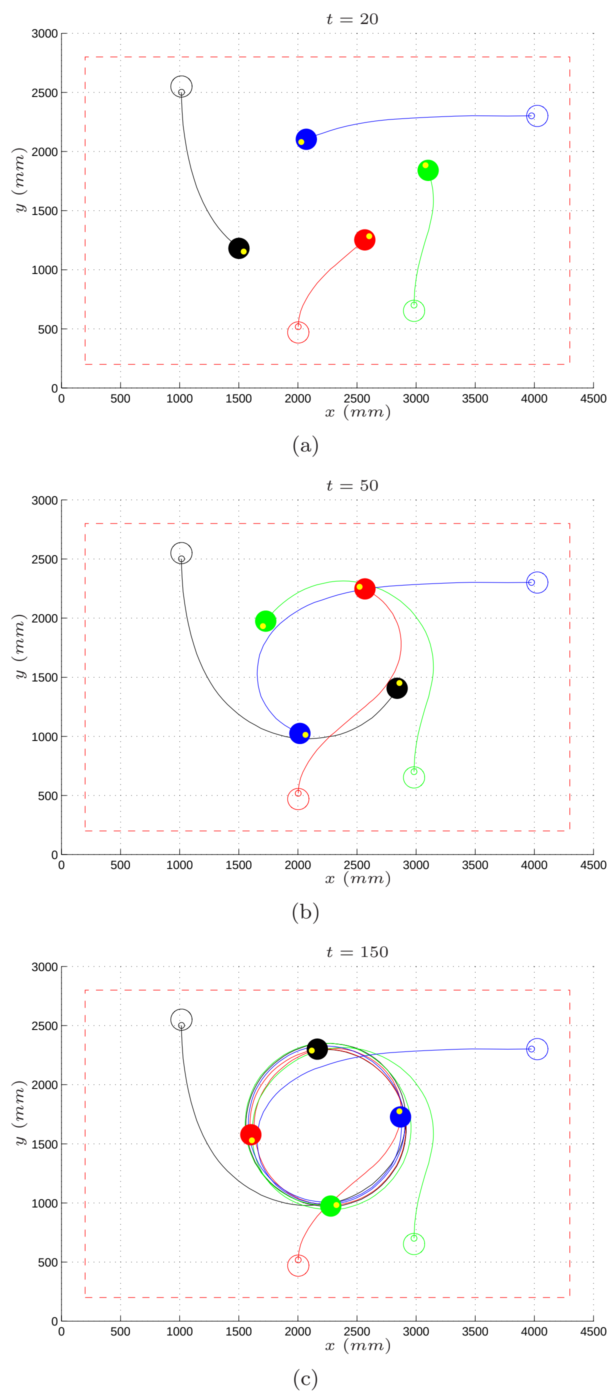


Fig. 10. Experiment C: three snapshots taken at time $t = 20$ s (a), $t = 50$ s (b) and $t = 150$ s (c). Initial (empty disc) and current (filled disc) robot position; actual robot path (solid line).

the aid of a global controller that knows the full state of the system. Figures 10(a)-10(c) report the results of an experiment where the cyclic pursuit algorithm studied in Marshall et al. [2006] has been implemented. Roughly speaking, the forward and angular speed of a vehicle are proportional to the distance and direction of its preceding neighbor (the prey), respectively. By properly selecting the ratio of gains of the two controllers, it is possible to achieve a team configuration where the vehicles rotate equally spaced on a circle. Notice that the tested algorithm assumes that a robot is equipped with a suitable sensor providing range and bearing measurements with respect to its prey. Although the LEGO vehicles do not carry any such exteroceptive sensor, the experimental setup is flexible enough to allow one to simulate the presence of on-board "virtual sensors". In this case, from the pose of the vehicles provided by the vision system, the required measurements are synthetically generated via software (possibly modeling different levels of sensor accuracy), and

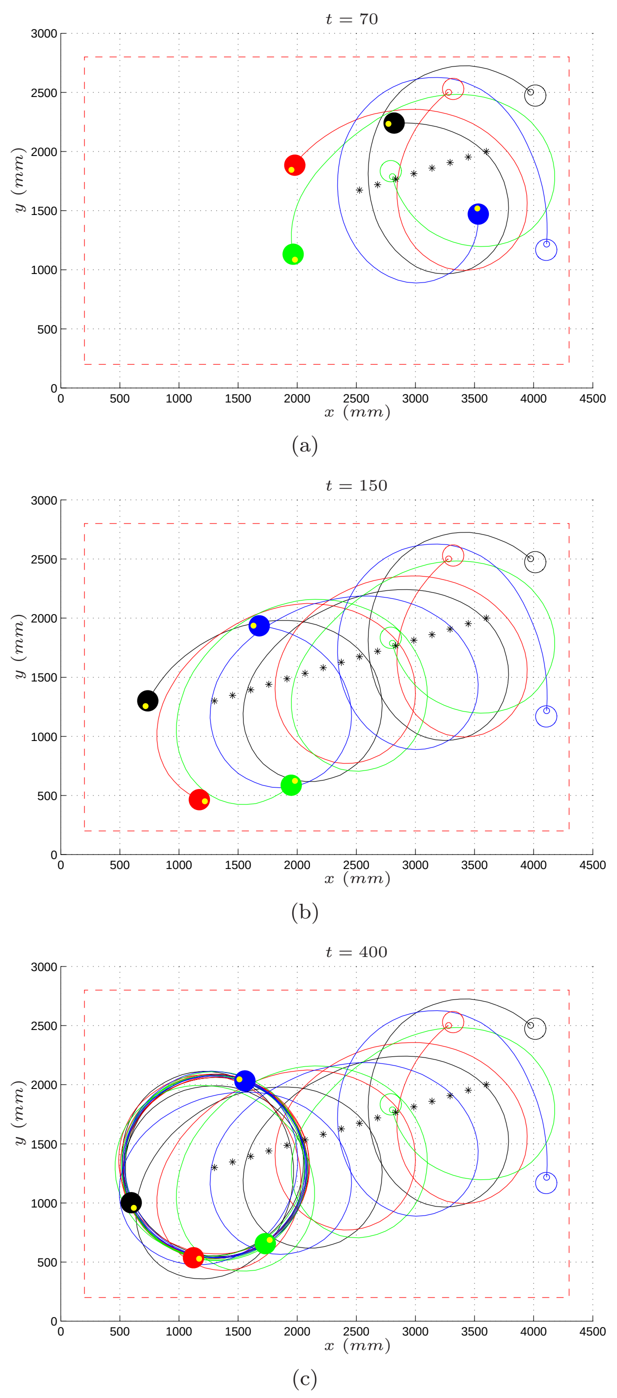


Fig. 11. Experiment D: three snapshots taken at time $t =$ 70 s (a), $t = 150$ s (b) and $t = 400$ s (c). Initial (empty disc) and current (filled disc) robot position; actual robot path (solid line); reference beacon (asterisk).

then the control law of each robot is computed preserving the decentralization of the algorithm.

4.4 Experiment D: Collective circular motion

The fourth experiment shows how the proposed setup can be used to validate multi-robot controllers which take into account constraints commonly found in a real-world scenario, like the one proposed in Ceccarelli et al. [2008]. Specifically, the problem considered there was that of collective circular motion. The objective was to design a decentralized control law able to put the vehicle in rotation about a virtual reference beacon, while at the same time keeping a minimum distance to the next rotating vehicle and avoiding collisions among members of the team. The problem becomes harder if one explicitly considers a number of constraints naturally arising in practice from technological limitations. It is supposed that robots are indistinguishable, and that each agent can only take range and bearing measurements with respect to “close enough” neighbors, i.e. vehicles falling inside the field of view of its sensors. The solution proposed in Ceccarelli et al. [2008] has proved to have nice theoretical properties in the case of static reference beacon, while only simulation results where available in case of a moving target. Figures 11(a)-11(c) refer to the validation of that control law for tracking a slowly moving reference beacon (marked with an asterisk). It can be seen how the vehicles proceed spiraling about the moving beacon and finally settle on a circle as the reference stops.

5. CONCLUSIONS

An experimental setup for multi-robot systems, embedded in the remote control lab ACT, has been developed and tested. The experiments presented in Section 4 are just a few examples of possible applications of the proposed setup. Their very different nature (single- or multi-robot, centralized or decentralized architecture) and purpose (teaching or research) demonstrate the versatility of the developed tool. At the same time, user interaction is extremely easy, requiring just basic skills of Matlab programming. The interested reader can play with a beta version at `http://act.dii.unisi.it/mr`. Movies and animations of real experiments, including those presented in this paper, are available at `http://act.dii.unisi.it/mr_video`.

Several additional features are currently under development. The possibility of including virtual obstacles, or even more complex artificial environments, will allow students and researchers to deal with fundamental tasks in mobile robotics, such as path planning, mapping and SLAM. Students’ interest will be stimulated by setting up a set of competitions on specific topics, such as virtual target tracking or cooperative pursuit-evader games, by employing the student competition environment of the ACT (Casini et al. [2005]).

REFERENCES

D. Benedettelli, A. Garulli, and A. Giannitrapani. Experimental validation of collective circular motion for nonholonomic multi-vehicle systems. *Robotics and Autonomous Systems*, 58:1028–1036, 2010.

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.

M. Casini, D. Prattichizzo, and A. Vicino. A student control competition through a remote robotics lab. *IEEE Control Systems Magazine*, 25(1):56–59, 2005.

M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A Matlab-based remote lab for multi-robot experiments. In *Proceedings of the 8th IFAC Symposium on Advances in Control Education*, 2009.

N. Ceccarelli, M. Di Marco, A. Garulli, and A. Giannitrapani. Collective circular motion of multi-vehicle systems. *Automatica*, 44(12):3025–3035, 2008.

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

S.A. Filippov and A.L. Fradkov. Cyber-physical laboratory based on LEGO Mindstorms NXT-first steps. In *Proceedings of the IEEE Control Applications & Intelligent Control*, pages 1236–1241, 2009.

A.L. Fradkov and P. Albertos, organizers. Invited session on "LEGO based control education and prototyping in robotics, mechatronics and embedded systems", *IEEE Control Applications & Intelligent Control*, 2009.

P.J. Gawthrop and E. McGookin. A LEGO-based control experiment. *IEEE Control Systems Magazine*, 24(5):43–56, 2004.

J. Hansen. Not eXactly C (NXC) - Programmer's Guide. http://bricxcc.sourceforge.net/nbc/nxcdoc, 2007.

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.

J. Marshall, M. Broucke, and B. Francis. Pursuit formations of unicycles. *Automatica*, 42(1):3–12, 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.

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

A. Valera, M. Valles, A. Fernandez, and P. Albertos. Platform for the development of mechatronic practical works based on LEGO Mindstorms NXT robots. In *Proceedings of the IEEE Control Applications & Intelligent Control*, pages 1224–1229. IEEE, 2009.