A remote lab for multi-robot experiments with virtual obstacles

Marco Casini, Andrea Garulli, Antonio Giannitrapani, Antonio Vicino

Dipartimento di Ingegneria dell'Informazione
Centro per lo Studio dei Sistemi Complessi
Università di Siena
Via Roma, 56, 53100 Siena, Italy
E-mail: {casini,garulli,giannitrapani,vicino}@ing.unisi.it

Abstract: Remote labs are increasingly being used as an effective tool for letting students practicing with real experiments, with limited effort and cost. A multi-robot setup, based on the LEGO Mindstorms technology and the Matlab environment, allowing students to experiment with real robots through the Internet, has been developed. In this paper, a novel feature of this setup is presented. The concept of *virtual obstacles* is introduced, and some obstacle avoidance experiments are reported in order to illustrate the main features of the proposed experimental environment.

Keywords: remote labs, mobile robots, multi-robot systems, obstacle avoidance

1. INTRODUCTION

The potential benefits of interactive classes, where students actively take part in the pedagogical process by doing things while acquiring skills, are widely acknowledged (the so-called "learning by doing" paradigm). The importance of practical activities is especially true in engineering curricula, featuring technology-oriented courses like control systems or robotics. Laboratory sessions with hands-on experiments are currently scheduled in many such courses. Unfortunately, letting students practicing with real physical processes can be a very expensive task, in terms of both time and cost. Things get worse in case of large classes. When the number of students grows, laboratory experiences, although highly instructive, quickly become unfeasible. In these cases, the progress of information technology and the diffusion of the Internet can be of help. As a matter of fact, remote labs are playing an increasingly important role in technical education (e.g., see Dormido [2004] for a thorough discussion on technology enhanced learning in the automatic control field). Enjoying all the advantages of a real laboratory, remote labs make available to students a number of physical processes to remotely experiment with. Typically, users run their own experiments and collect data by simply connecting to the lab through the Internet with a common web browser, thus allowing a vast audience of learners to gain access to a variety of (possibly expensive) educational experiments. Nowadays, remote laboratories represent a successful trade-off between the need of practical experiences and the shortage of available resources.

Over the last decade, a remote lab focused on robotics and control systems curricula has been developed at the University of Siena (Casini et al. [2004]). The Automatic Control Telelab (ACT) puts a number of different teaching experiences at users' disposal, ranging from basic electric motor identification and control experiments to more advanced control problems of nonlinear mechanical systems (e.g., magnetic levitator or helicopter). Recently, a new teaching experience has been added to the ACT, which allows students to work with teams of mobile robots (Casini et al. [2009, 2011]). The setup consists of four vehicles and a supervisor which monitors the system and manages the user interaction. Robots are built with the LEGO Mindstorms technology, which is widely used for teaching purposes (Filippov and Fradkov [2009], Valera et al. [2011]) and research activities (Benedettelli et al. [2010]), allowing for the rapid prototyping of small footprint vehicles with limited cost. A web page allows users to interact with the lab and a graphical interface provides the online visualization of the selected experiment. Several predefined activities with the multi-robot team are available, but it is also possible to run a user-defined experiment by uploading a suitable Matlab function which implements the robot control laws.

In this paper, a novel feature of the multi-robot system embedded in the ACT is presented. In order to enable more complex experiments, while at the same time preserving the easiness of use, the concept of *virtual obstacles* is introduced. Besides specifying the robot controllers, now students may define polygonal regions which model obstacles present in the robot workspace. At each time step, in addition to the robot poses, the supervisor computes the distance of each robot to the obstacles. With this information, a user can embed in his/her controller also an obstacle avoidance module to prevent vehicles to collide with the virtual obstacles or among each other. The choice of having virtual obstacles rather than real objects improves the versatility of the system and reduces the risk of dangerous physical impacts. Some experiments carried

out within the proposed framework are described, in order to illustrate the relevant features of the developed setup.

The paper is structured as follows. Section 2 provides a brief overview of the multi-robot system developed within the ACT remote lab. The concept of virtual obstacles and their implementation are described in Section 3. A collection of possible experiments concerning obstacle avoidance algorithms is presented in Section 4. Some concluding remarks and future extensions are sketched in Section 5.

2. MULTI-ROBOT REMOTE LAB

The developed multi-robot remote lab consists of four identical mobile robots built with the LEGO Mindstorms technology (see Figure 1). The vehicles feature a differential drive with two independent motors directly coupled to the left and right wheel. A third support guaranteeing the stability of the robot is provided by a passive ball transfer unit. Motors are commanded by a NXT brick which runs a PID controller in charge of tracking the reference signals of the wheel speed. The kinematics of the i-th vehicle can be well approximated by that of a unicycle

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

where $[x_i(t)\ y_i(t)]^T$ is the robot position, $\theta_i(t)$ its orientation, and $v_i(t)$ and $\omega_i(t)$ denote the linear and angular speed of the vehicle.

Fig. 1. LEGO NXT mobile robot.

Two wide-angle webcams, placed on the lab ceiling and pointing downwards, cover the whole experimental area (approximately 4.5×3 meters). Their purpose is to detect position and orientation of each robot, by extracting from the acquired images the circular markers placed on the top of each robot (see Figure 1). Each NXT microcontroller communicates through a Bluetooth link with the Centralized Supervision System (CSS), which manages the overall multi-robot system. Among other tasks, the CSS is responsible for the image processing, the generation of the speed reference signals to be sent to each robot and the management of the user interaction. A user willing to run a multi-robot experiment can connect to the system through a dedicated web server. From the starting page it is possible to choose whether to run a predefined experiment or proceed with a user-defined one. In the latter case, the user uploads a Matlab function which implements the desired control law. Such a function, which will be invoked by the CSS during the experiment at each sampling time, must return the linear and angular speed reference signals to be sent to the robot. A graphical user interface, implemented as a Java applet, allows the user to monitor the experiment through a graphical panel showing online the robot positions and possible virtual obstacles placed in the workspace (see Figure 2). An additional feedback is provided by a live streaming of the actual robots moving in the lab (right bottom picture in Figure 2). At the end

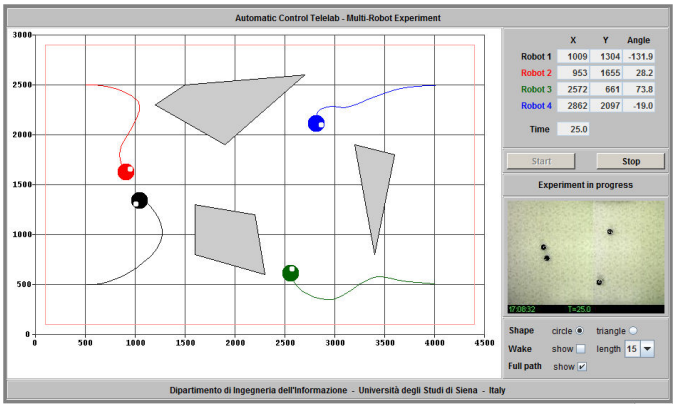


Fig. 2. Graphical user interface.

of an experiment the user can download all the data for offline analysis (robot poses and control inputs at each sampling time). To reproduce a graphical animation of a real experiment from the downloaded data, a Matlab script is also provided *(Experiment Player)*. Analogously, to test a controller before uploading it for a real experiment, a Matlab simulator of the multi-robot team is available *(Experiment Simulator)*. To make the lab available 24 hours a day, an automatic recharge system has been developed. When the battery voltage of a robot goes below an alert threshold, the CSS takes control of the vehicle and drives it to the charging station. The reader interested in technical details on the implementation of the multi-robot experimental setup is referred to Casini et al. [2009]. The Automatic Control Telelab is freely accessible through `http://act.dii.unisi.it`.

3. VIRTUAL OBSTACLES

In this section, it is described how users can define virtual obstacles as well as which information useful to design the controller is returned by the CSS. As mentioned in Section 2, users have to upload a Matlab function in order to perform a personalized experiment. The syntax of this function is

```
function [U,N_robot,Pose_init,Obstacle]=
                user_function(Time,Pose,Ts,Map)
```

At each time step, `Time` denotes the experiment time, `Pose` is a matrix containing the poses (x_i, y_i, θ_i) of each robot, `Ts` is the sampling time, and `Map` is a structure regarding virtual obstacles which will be described in details later. Regarding output variables, at each time t, `U` contains the forward and angular velocity reference for each vehicle. Variables `N_robot`, `Pose_init` and `Obstacle` denote the

number of robots used in the experiment (up to four), their initial poses, and the definition of virtual obstacles in the environment; such variables are not changed during the experiment but are useful for experiment initialization.

To define virtual obstacles, one has to create a structure `Obstacle` with the following syntax:

```
Obstacle(i).vertex=V
```

where `i` denotes the progressive number of each obstacle and `V` is a matrix $v \times 2$ whose rows contain the $x - y$ coordinates of the v vertices of the polygon. There is no limitation on the number of obstacles as well as on the number of vertices of each obstacle. Since each vertex is connected to the adjacent one, the resulting polygon can be both convex and nonconvex. If the variable `Obstacle` is not set inside the user-defined function, the environment is assumed without obstacles, unless differently set by the user in the web page of the experiment (e.g., by choosing a predefined or a random environment).

During the experiment, at each time step, information regarding the environment are stored in the structure `Map`. Such a structure contains the following fields:

- `Obstacle`. It is a copy of the previously defined homonymous structure, useful when the environment is not defined inside the user function (i.e., for random and predefined environments).
- `Obstacle_Grid`. It is a matrix 300×450 containing the occupancy grid of the environment with the resolution of 1 *cm*. The value of each entry denotes if the corresponding location is free or if it is taken by an obstacle. Thus, information contained in such a matrix can be easily used inside the user-defined algorithm.
- `Obstacle_Distance`. It is a matrix whose rows and columns denote the robot identity and the obstacle identity, respectively. Each entry is a structure which contains the minimum distance of the selected robot from the chosen obstacle (field `min`) and the corresponding point of the obstacle with minimum distance (field `p_min`). For instance,
  `Map.Obstacle_Distance(2,3).min`
  denotes the minimum distance between Robot#2 and Obstacle#3, while
  `Map.Obstacle_Distance(2,3).p_min`
  contains the nearest point of the obstacle to the robot. In Fig. 3, an example of computation of such distances is reported.
- `Robot_Grid`. It is a matrix similar to `Obstacle_Grid` but referred to other robots instead of obstacles. Since the matrix dimensions are still 300×450, it is possible to obtain the occupancy grid for both obstacles and robots by just combining the information in matrices `Obstacle_Grid` and `Robot_Grid`.
- `Robot_Distance`. Like `Obstacle_Distance` but referred to robots.

Note that while information stored in `Obstacle_Distance` (and in `Robot_Distance`) represent the essential knowledge for developing a path planning algorithm, the occupancy grids are useful for the implementation of algorithms based on a discrete representation of the environment (see, e.g. Barraquand and Latombe [1991]). It is worthwhile to note that all the distance and occupancy grid information could be computed within the user-defined function on the basis of the actual robot poses and the obstacles definition. However, in principle, such computations can be quite involved and they may not be suitable for educational purposes.

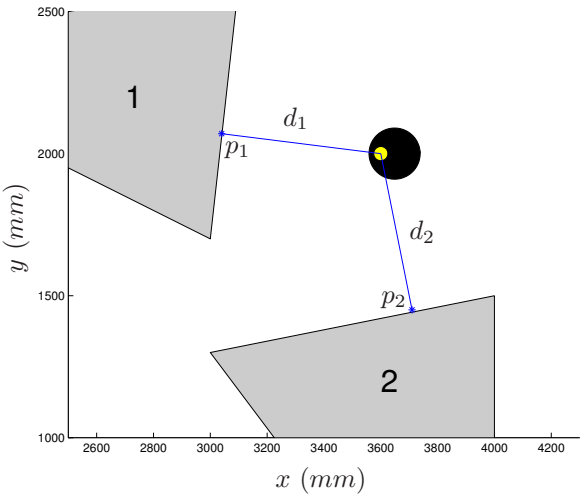


Fig. 3. Distance computation. For each robot, the minimum euclidean distance from each obstacle ($d_i,\ i = 1, 2$) and the point of minimum distance ($p_i,\ i = 1, 2$) are computed and stored into the `Map` structure.

4. EXPERIMENTAL TESTS

In this section, some experiments carried out with real robots are presented, in order to illustrate some of the possible teaching experiences which can be worked out within the remote lab described so far. The availability of virtual obstacles naturally leads to consider motion planning and collision avoidance algorithms.

One of the most basic issues to be addressed in autonomous navigation is the obstacle avoidance problem. An essential condition for the successful operation of robots moving in non-empty environments is the safety requirement, i.e. the ability of avoiding collisions which could cause damages to things or people. As a matter of fact, several collision avoidance algorithms have been investigated since long time, for different kinds of robots and environments (see the book Latombe [1991] for a comprehensive treatment of robot motion planning techniques). One of the first and most intuitive obstacle avoidance algorithms is based on the concept of *artificial potential field*, originally proposed in Khatib [1986]. In this framework, the robot is assimilated to a particle immersed in a potential field, and, as such, it moves according to a force which is the negative gradient of the potential function. By suitably shaping the artificial potential it is possible to drive the robot towards the target, while at the same time avoiding collisions with the obstacles. In the analogy with the gravitational field, the robot is like a ball free to move in an uneven terrain with mountains (the obstacles) and valleys whose lowest point is at the target. Like the ball runs through the valleys and is repulsed by the mountains due to the gravitational force, similarly the robot is driven to the target and pushed faraway from the obstacles.

Let $q \in \mathbb{R}^2$ denote a point of the environment and $p_t \in \mathbb{R}^2$ be the target location. Then, the artificial potential function $U(\cdot)$ evaluated at q is the sum of an attractive term $U_a(q)$ and a repulsive term $U_r(q)$, i.e.

$$U(q) = U_a(q) + U_r(q), \tag{2}$$

where, in turn, the total repulsive potential can be computed as the sum of the potential $U_{r_i}(q)$ associated to each obstacle

$$U_r(q) = \sum_{i=1}^{n_o} U_{r_i}(q), \tag{3}$$

with n_o being the total number of obstacles. Depending on the shape of $U_a(q)$ and $U_r(q)$, different robot motions are obtained. The most common choice for the attractive term is a parabolic well with its minimum at the target

$$U_a(q) = \frac{1}{2} k_a ||q - p_t||^2, \tag{4}$$

where k_a is a positive scaling factor. The repulsive term has to be designed by taking into consideration a couple of objectives. The repulsive force has to be stronger and stronger as the robot approaches to an obstacle, and faraway obstacles should have little or no influence. A possible function satisfying such requirements is

$$U_{r_i}(q) = \begin{cases} \frac{1}{2} k_{r_i} \left(\frac{1}{\rho_i(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho_i(q) \leq \rho_0 \\ 0 & \text{if } \rho_i(q) > \rho_0 \end{cases}, \tag{5}$$

where k_{r_i} is a positive scaling factor. The function $\rho_i(q)$ denotes the distance from q to the i-th obstacle, whereas the positive constant ρ_0 determines the region of influence of an obstacle. Notice that $U_{r_i}(q)$ tends to infinity as q gets closer to the obstacle and is null when the robot is farther than ρ_0 from the obstacle. Given the artificial potential field $U(q)$ defined through (2)-(5), the corresponding virtual force at position q can be computed as

$$F(q) = -\nabla U(q). \tag{6}$$

At each time step t, the obstacle avoidance algorithm based on the artificial potential field method computes the desired velocity vector $\dot{p}(t)$ of a robot placed in the position $p(t)$ proportional to the resulting force $F(p(t))$ in (6), i.e. $\dot{p}(t) = k\, F(p(t))$.

This algorithm has been implemented as a Matlab function and tested in the developed remote lab. Since the actual robots have a unicycle-like kinematics as described in (1), their velocity vector cannot be arbitrarily set due to the nonholonomic constraints. Hence, a further step is necessary to generate the references of the linear speed v and angular speed ω, given the nominal velocity $\dot{p}(t)$ provided by the planning algorithm. For this purpose, a proportional controller has been implemented as follows

$$v(t) = k_v ||\dot{p}(t)||, \tag{7}$$

$$\omega(t) = k_\omega (\theta_r(t) - \theta(t)), \tag{8}$$

where $\theta_r(t) = \text{atan2}(\dot{p}_y(t), \dot{p}_x(t))$ represent the nominal direction of motion computed by the obstacle avoidance module, and k_v and k_ω are positive gains. Figure 4 shows the path of an experiment involving one robot and four virtual polygonal obstacles (Experiment A). The robot starts at position $p_i = [4.28\ 2.39]^T$ (the default initial position of the first robot of the team) and has to reach the target placed at $p_t = [0.45\ 0.45]^T$ (all coordinates are expressed in meters). It can be observed that in 90 s the robot succeeds in reaching the desired target while at the same time avoiding the virtual obstacles. The effects of the discrete-time implementation of the algorithm (with a sampling time $T_s = 1\,s$ prescribed by the vision system), as well as of the nonholonomic kinematics of the vehicle, are clearly visible in the second half of the robot path. The unnatural oscillation between the last two obstacles is due to impossibility for the robot to follow exactly the direction of fastest descent of the potential function, because of kinematic constraints and discrete-time velocity updates. The final robot position (denoted by a small light point inside the black circle in Figure 4) is slightly different from the desired one, since the task is considered accomplished when the robot is close enough to the target (in this experiment, the threshold is set to 1 cm). The plots of robot coordinates against time are reported in Fig. 5.

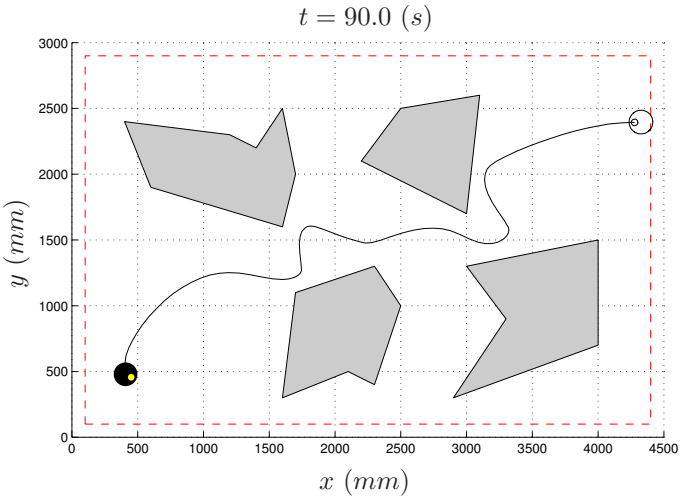


Fig. 4. Experiment A. Single-robot and four obstacles (shaded regions): initial robot pose (empty circle), final robot pose (filled circle), robot path (solid line).

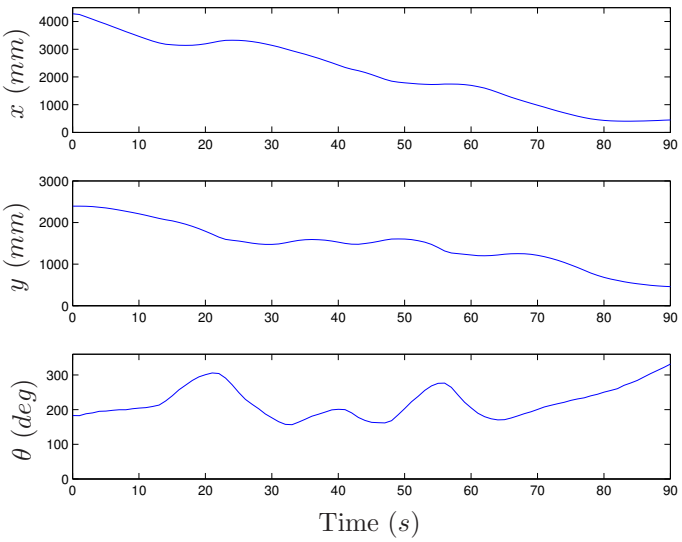


Fig. 5. Experiment A. Time plots of robot coordinates.

Although the artificial potential field method is very appealing for its simplicity and efficiency, it suffers from a major drawback. Analogously to the gradient methods developed for numerical optimization, a robot moving according to the force induced by the potential function can get stuck in a local minimum. Wherever the attractive

force balances the repulsive force, the potential function $U(q)$ presents a local minimum in which the vehicle can get trapped. This especially happens in the presence of non-convex obstacles or complex environments giving rise to symmetric attractive and repulsive forces. Notice that the attractive potential depends on the target position. Hence, a robot moving in a given workspace can end up in a local minimum or not depending on the target location. An example of such phenomenon is observed in Figure 6, which shows the results of a second experiment (Experiment B) carried out with one robot in the same virtual environment of Experiment A. The robot initial position is the same as before, but this time the target is placed at $p_t = [3.0\ 0.8]^T$. In this case, a local minimum arises somewhere between the two rightmost obstacles which catch the vehicle, thus hampering the reaching of the target. Again, due to the discretization of the controller and the kinematics of the vehicle, rather than stopping at the local minimum the robot keeps moving around it. It is worth remarking that since the seminal paper Khatib [1986] several improvements to this method have been proposed (see, e.g., Khatib and Chatila [1995]), mainly aiming at mitigating the problem of local minima through the selection of suitable artificial potential functions. The interested reader is referred to the book Latombe [1991], Ch. 7, and references therein for further information.

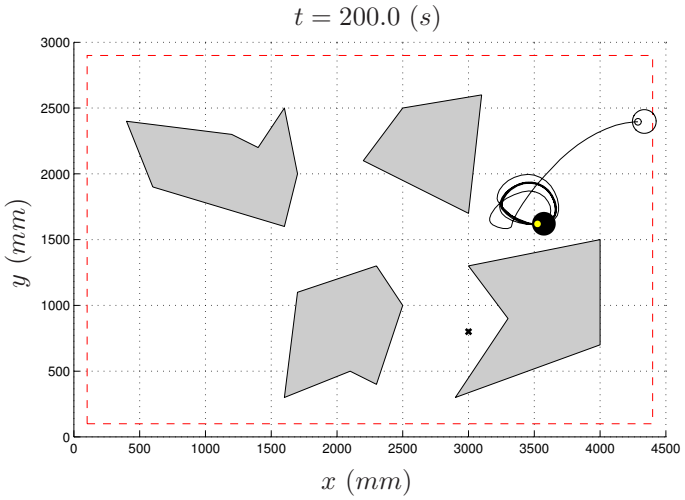


Fig. 6. Experiment B. Single-robot and four obstacles (shaded regions): initial robot pose (empty circle), final robot pose (filled circle), robot path (solid line), desired target (black cross).

A nice feature of the potential field method is the possibility of computing beforehand the repulsive potential at each point of the environment, thus speeding up the planning when dealing with complex environments. Moreover, this collision avoidance technique can naturally address the case of moving obstacles, as long as the robot is equipped with sensors measuring the distance from the obstacles and their direction. The versatility of the developed remote lab allows users to test their obstacle avoidance algorithms also in presence of non static obstacles. Figures 7(a)-7(c) show three snapshots taken during an experiment carried out with two robots in an environment containing one virtual rectangular obstacle (Experiment C). In this case, each agent must reach a target position while avoiding both the static obstacle and the other robot, which plays the role of a moving obstacle. Recall that among the input parameters passed to the user-defined Matlab function there is also the distance and the direction of a vehicle with respect to all the others. Hence, the potential field method described so far needs just minor modifications to prevent robots from colliding each other. As a matter of fact, it suffices to add a new term like (5) in the repulsive potential (3) to account for the moving obstacle represented by the other vehicle. With such modified motion planning algorithm, both robots have safely reached the desired target without collision. Notice that after about 27 s the robots face each other nearly colliding. When they get too close, the repulsive force prevails and the vehicles swerve, thus avoiding the impact.

Several other trials have been run, with different number of robots and virtual obstacles. Modified versions of the techniques illustrated in this section have also been tested, with various results, ranging from safe but slow paths to more aggressive trajectories eventually leading to collisions. From an educational viewpoint, this is a key feature of the proposed setup. As it happens with virtual robot simulators (e.g., see Guzmán et al. [2008]), the possibility of playing safely with the parameters of the algorithms under investigation, without risking dangerous and expensive crashes, greatly improves the understanding of the role of each parameter.

5. CONCLUSIONS

In this paper, a multi-robot setup including virtual obstacles has been presented. The developed architecture, embedded in the ACT remote lab, allows students to test motion planning algorithms for single- or multi-robot systems, moving in environments with static and/or dynamic obstacles.

An additional feature currently under investigation is the development of *virtual sensors*, which will complete the process of building an augmented reality multi-robot setup, where real robots equipped with virtual sensors move in real environments populated with virtual obstacles. Moreover, given the encouraging results obtained in the past (Casini et al. [2005]), such experimental setup will be exploited to organize student competitions on relevant multi-robot problems.

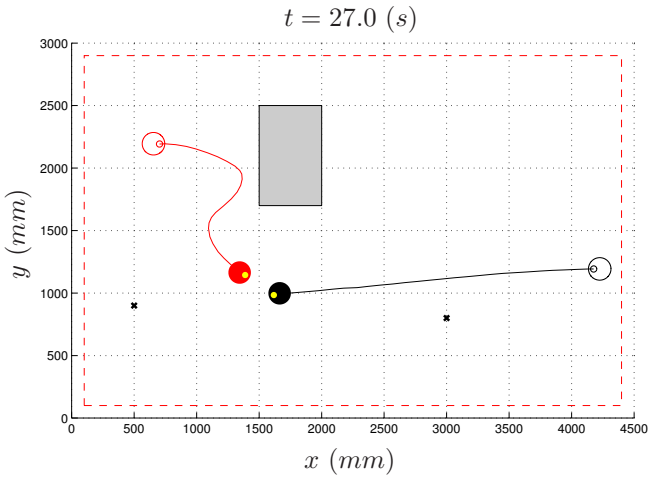
REFERENCES

J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 1991.

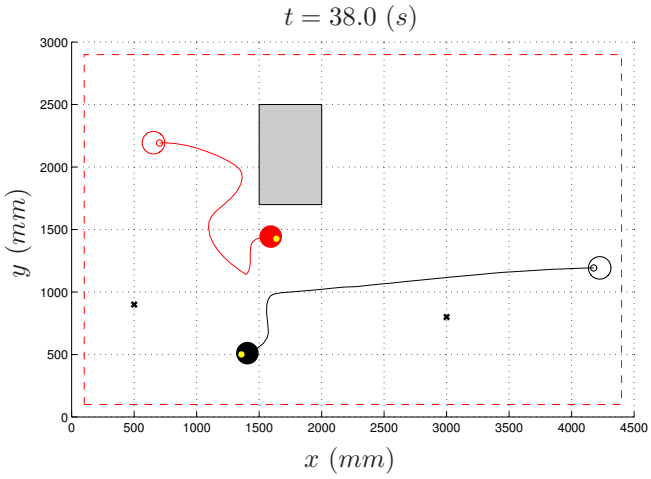
D. Benedettelli, N. Ceccarelli, A. Garulli, and A. Giannitrapani. Experimental validation of collective circular motion for nonholonomic multi-vehicle systems. *Robotics and Autonomous Systems*, 58(8):1028–1036, 2010.

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.

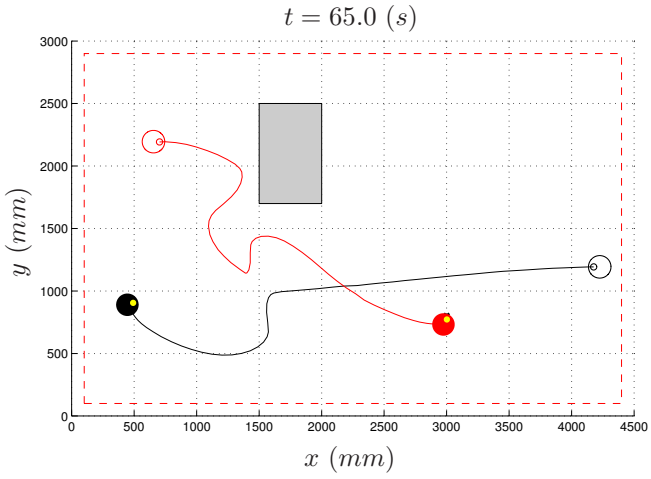
M. Casini, D. Prattichizzo, and A. Vicino. A student control competition through a remote robotics lab. *IEEE Control Systems Magazine*, 25(1):56–59, 2005.



(a)



(b)



(c)

Fig. 7. Experiment C. Three snapshots of a multi-robot experiment with one obstacle (shaded region): initial robot pose (empty circle), final robot pose (filled circle), robot paths (solid line), desired targets (crosses).

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

M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A LEGO Mindstorms multi-robot setup in the Automatic Control Telelab. In *Proc. of the 18th IFAC World Congress*, pages 9812–9817, Milan, Italy, August 2011.

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 *Proc. of the IEEE Control Applications & Intelligent Control*, pages 1236–1241. IEEE, 2009.

J. L. Guzmán, M. Berenguel, F. Rodríguez, and S. Dormido. Interactive tool for mobile robot motion planning. *Robotics and Autonomous Systems*, 56(5): 396–409, 2008.

M. Khatib and R. Chatila. An extended potential field approach for mobile robot sensor-based motions. In *Proc. of the International Conference on Intelligent Autonomous Systems*, 1995.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90, 1986.

J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

A. Valera, M. Valles, L. Marín, A. Soriano, A. Cervera, and A. Giret. Application and evaluation of Lego NXT tool for mobile robot control. In *Proc. of the 2011 IFAC World Congress*, pages 9805–9811, Milan, Italy, August 2011.