ACE 2009

8th IFAC Symposium on Advances in Control Education

October 21-23, 2009, Kumamoto, JAPAN

ACE2009

8TH IFAC SYMPOSIUM ON ADVANCES IN CONTROL EDUCATION

Organized by:

Kumamoto University

Kumamoto National College of Technology

In co-operation with:

Society of Instrument and Control Engineers (SICE)

The Institute of Electrical Engineers of Japan (IEEJ)

The Japan Society of Mechanical Engineering (JSME)

The Japan e-Learning Association (JeLA)

Research for Intelligent System Technology in Kumamoto (RIST)

An educational tool for mobile robots remote interaction

Luis Payá, Oscar Reinoso, Alejandro Sánchez, Arturo Gil, Lorenzo Fernández

Departamento de Ingeniería de Sistemas Industriales
Miguel Hernández University. Avda. de la Universidad s/n. 03202, Elche (Alicante) Spain.
(e-mail: lpaya@umh.es, o.reinoso@umh.es).

Abstract: This work presents a tool that provides an easy and intuitive remote interaction with a team of mobile robots through Internet. It has been designed with the aim that students of Robotics carry out training with mobile robots from their own home, independently of the operative system they use and with a minimum installation in their computer. This tool facilitates the understanding of several algorithms and control strategies in the field of mobile robotics and computer vision. In this paper we detail the architecture of the platform and the model of practical sessions we propose.

1. INTRODUCTION

Hand-on experiments are a fundamental part of engineering studies so that students can acquire the ability to solve real problems using real equipment. Internet has become a powerful tool in this field of education, and some kinds of remote laboratories have been developed to provide access to laboratory equipments through Internet. However, there are some challenges we must address in order to improve the effectiveness of the remote laboratories, as a friendly user interface, reliability, robustness and transparency, interactivity and feedback to the student.

In this work, we present a remote laboratory that allows the operation of several mobile robots. It can be applicable to subjects about computer vision and robotics and permits accessing the mobile robots available in our laboratory, to monitor and control their evolution. We have tried to centre the focus of the experimentation in the goals of the subject. It has been built with the aim that the student does not need any knowledge about the internal architecture of the robot to develop the experimentation.

The information and communication technologies have been put into practice in recent years in the field of control systems teaching, with successful results, as (Dormido *et al.*, 2004) show. These technologies allow us to implement both virtual laboratories that simulate the behaviour of some devices, and remote laboratories that permit us to use the real equipments available in the laboratory in a remote way. As an example, the work of (Candelas *et al.*, 2003) presents a platform that is capable to simulate a robot arm and also, students can teleoperate the equivalent real robot. Also, (Jimenez *et al.*, 2005) present a tool to carry out control systems training in a remote way, using Matlab/Simulink to test different kinds of controllers.

(Masár *et al.*, 2004) present a framework to develop virtual laboratories for use in distance education for control engineers including a virtual laboratory for inverted pendulum and a virtual laboratory for mobile robot control through a web browser. (Guzmán *et al.*, 2009) present an interactive tool for mobile robot motion planning to enhance full understanding of these algorithms by the students. (Borgolte, 2008) exposes some considerations that must be taken into account for remote operation of a laboratory with a mobile robot, such as the collision avoidance, network time-outs and battery charging. (Khamis *et al.*, 2002) address a survey on the design and implementation of remote experiments on motor control of mobile robots, emphasizing on user interface enhancement through the use of virtual reality and in (Khamis *et al.*, 2003) these concepts are specified in the 'Developer' architecture to teleoperate and control the B21r robot. New control platforms are being developed, as (Marin *et al*, 2009), that introduces the control of several robots by using a PDA or mobile phone as input and presents some performance results.

The platform we have built permits the access to the robots available in the laboratory, in a transparent way for the user, from their own home, in a free timetable and with no limitations of time. The students can access the platform from any computer through Internet, independently of their operative system and with a very simple installation. There is a common graphical interface to access all the robots, and the students can monitor and control them in an intuitive way, without needing a deep knowledge of the architecture of the robot they are using. This way, the training is centred in the objectives of the subject and the students take more profit of the educative process.

The remainder of the paper is structured as follows. Section 2 presents a general description of the system. In section 3 the software architecture and communications between the components of the system are detailed. Section 4 presents the graphical interface and the possibilities it offers to monitor and control the robots. In section 5 we outline the protection measures we have implemented to avoid malfunction and, at last, in section 6, we present the practical experiments the students can carry out with the platform and the conclusions of the work.

2. SYSTEM DESCRIPTION

In this section, the principles of the physical architecture of the system are presented. Each working environment is composed by a WifiBot robot that has a wireless connection to the access point, and an Axis 207W camera, that provides us with a general view of the working environment. Nowadays, there are three different working environments available that allow the simultaneous access of three students to carry out the practices. The structure of environments is shown on fig. 1.

The robots can move freely in a 200x300 cm rectangular area, bounded by walls, and with some obstacles inside it. The web cam that monitors the environment is situated in the ceiling, at a distance of 300 cm from the floor.

The platform has been developed with the Wifibot 4G robot. It is a flexible and low-performance robot, that offers a wide variety of expansion possibilities at different levels, what make it suitable to the development of practices. It is equipped with a pan-tilt camera onboard that takes images from the environment, from the point of view of the robot with a maximum resolution of 640x480 pixeles. Also, each wheel has an encoder to measure each speed individually. These encoders allow us to estimate the position of the robot (after an odometry process). At last, two frontal infrared sensors are situated in the front of the robot, with a detection range between 20 and 120 cm. These sensors can be used both for tasks of obstacle avoidance, as a safety measure and in tasks that imply following a mobile agent.

The second component of the working environment is the IP Camera, which is situated in the ceiling of the laboratory. It provides up to 30 frames per second and it has a wireless interface to access the visual information captured. Thanks to this camera, we can have a bird's eye view of the environment where the robots evolve.

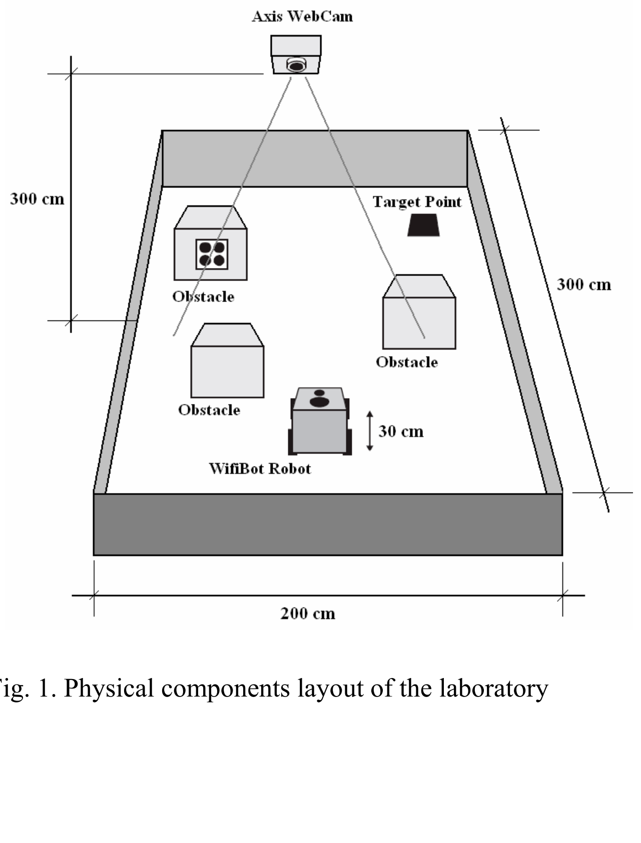


Fig. 1. Physical components layout of the laboratory

Apart from these hardware components, some other visual artificial landmarks have been situated along the environment with different purposes. The first one is a square mark situated on the floor, which can be viewed from the ceiling camera. It will constitute the target point where the robot has to tend in some tasks. The second mark consists in two circles with different size, situated at the top of the robot. Thanks to it, we can draw the trajectory and orientation of the robot from a set of frames captured from the ceiling camera. The last mark consists in four small circles situated at the vertexes of a square. It can be viewed from the robot camera. The students have to detect it in a practical session with a computer vision process and situate the robot at a relative position from it.

Besides, there is a PC that is connected to the network of the laboratory where the next basic services run:

- *Web server*. In this server, the client application is available so that the students can download and install it in their computers.

- *Identity server*. It is the base of the communications system we have implemented.

- *SQL server*. It is used to implement several safety mechanisms such as the restriction to the access and a record of all the actions each user has carried out with each robot.

3. SYSTEM ARCHITECTURE

3.1 Software architecture

The communication between the different members of the system has been implemented using the CORBA (Control Object Request Broker Architecture) reference model. The robustness and efficiency of this standard for the implementation of distributed applications has been demonstrated in previous works (Payá *et al*., 2006), (Payá *et al*., 2007). The main components of the communication network, shown on fig. 2, are:

Onboard Servers. Each robot has been equipped with an onboard PC where the necessary serves to access the sensors and actuators of the robots run. These services have been implemented using the following CORBA interfaces:

- *RobotBasics Interface*. It implements the basic movement actions the robot can carry out (starting movement, setting up an advance/steering speed and stopping the robot). It also gives access to the value of the infrared sensors and the odometer data of the robot.

- *SingleCamera Interface*. It allows obtaining snapshots from the camera each robot has on it and from the ceiling camera.

- *LaserServer Interface*. It offers data from a laser distance server. It is available only in the robots that have this kind of sensor.

Identity Server. It runs in a PC that is connected to the laboratory network. Its IP address must be known by the rest of components of the system. This is the only piece of information we must configure to establish the communication between all the components. This server has two functions. The first one consists in keeping a list of all the robots and the cameras of the environment that are active at each moment in the system, and the list of services they offer (camera, infrared sensors, sonar, etc.). The second function is to carry out an access control from the client applications to the robots.

Client application. It can run on any computer, independently of its architecture and operative system. This application permits the access to the robot environments to carry out the practical sessions, independently of the location it runs from. The student can access from the computers in the laboratory, but also from any computer that has Internet access. This client application has been entirely developed using the Java programming language, because it provides independency both from architecture and from operative system. Also, Java supports CORBA in a native way, without necessity of using other foreign libraries. This feature permits the interaction of the client application with the CORBA services developed using the C++ language, both those offered by the robots and those offered by the environment cameras and the identity server.

The implementation of the client application has been carried out through Java Web Start. It is the reference implementation model of the JNLP (Java Networking Launching Protocol) specification. With this technology, it is possible to start desktop Java applications that reside in a web server, checking first if the version of the application the client has installed is the latest one. If not, it downloads the newest version and the application will run in local mode. The starting of such applications can be carried out from a web page or through a link in the client desktop. Thanks to this technology, we can be sure that an application is always distributed with its most up to date version.

The information about the location of the application, the version and other details can be found in a configuration file whose extension is '.jnlp'. This way, it is necessary to put at the disposal of the students the '.jnlp' file in a web page so that they can download it and run it after. Also, any person who wants to make use of the application must install in his computer the last distribution of JRE (Java Runtime Environment). We must take into account that currently, Java Web Start is included in the JRE.

3.2 Communications

Two different communication protocols have been used to satisfy the needs of the laboratory: CORBA and SSH communications.

CORBA communications. The basic communications framework has been developed with this reference model. The sequence of operation is the following one; at first, it is necessary that the Identity server is running. After this, the onboard servers of the robots must be activated. When one of these servers starts, it publishes the name of the available robot and the interfaces it implements in the identity server. When a client application wants to access the services a robot offers, it first makes a request to the identity server, indicating the symbolic name of the robot and the interfaces it wants to access. If the robot is available at that moment, the Identity Server provides the client application with the references to the interfaces the robot implements. From that moment, the communication is carried out in a bidirectional way between the client application and the onboard server of the robot. In a typical mode of operation, the client application sends some requests of data capture from the sensors and movement commands to the motors.

SSH communications. This level of communication is used with the objective that the students can test the algorithms they have implemented to control the robot. The main goal is to allow the student to send the program he has created to the robot he is operating, to compile and to run it.

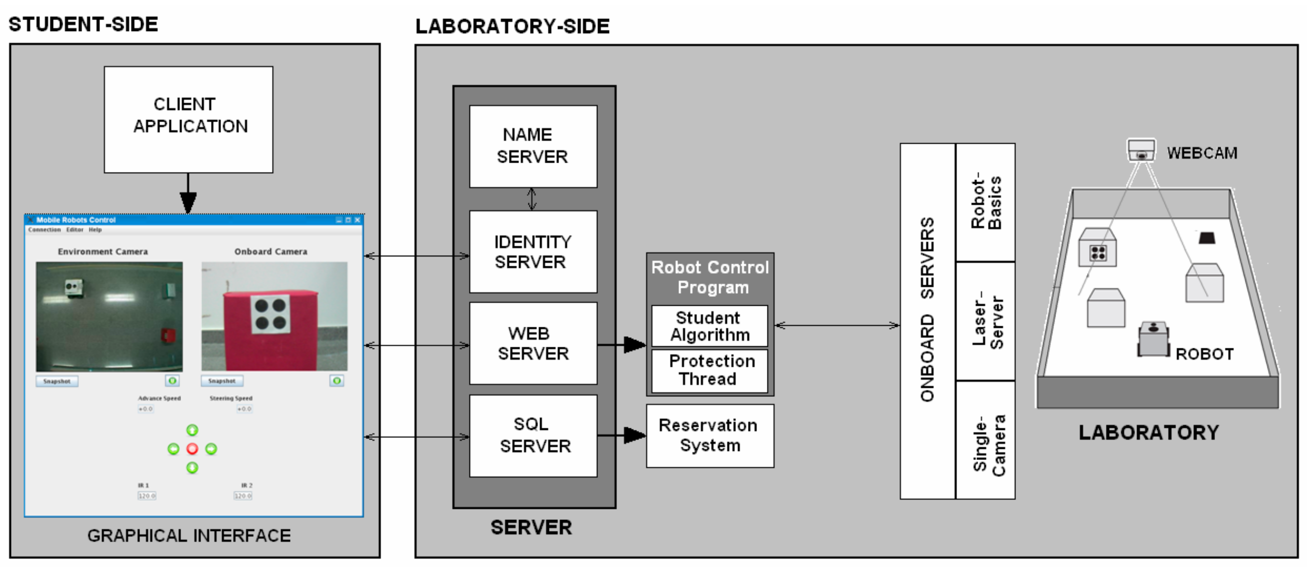


Fig. 2. Software architecture of the platform developed to interact with the remote laboratory

The implementation of this communication flow has been made with JSch, a Java library that allows us to integrate the functionality of the SSH2 protocol in our program. SSH gives support for a reliable remote access, file transfer and TCP/IP readdressing. It can encrypt, authenticate and compress the data transmitted automatically. The platform makes use of the following JSch functions:

- *Scpto*: This function is used to send the file that contains the source code from the user's computer to the computer in the laboratory where it will be compiled and run.

- *Exec*: It permits running commands in a remote way. In this platform, it has been used to compile the source code, using the command 'makefile' in the remote machine and to run it later. Besides, this function permits showing on the user's screen the standard output ('stdout') and the standard error output ('stderr'). If the standard error does not activate, it means that the compilation has been carried out successfully. Otherwise, the student should correct the source code and send it again to the remote machine.

The advantage of this method is that the student does not need to install any library in his computer. However, the depuration of the possible errors during the compilation becomes a slow process, because the source code must be sent several times until all the possible errors have been corrected.

4. PLATFORM DEVELOPED

Fig. 3 shows the appearance of the client application. From the main window of this application, the student can continuously see the images that both the camera of the ceiling and the camera of the robot capture.

Both video panels have a button associated so that the student can capture and save some snapshots during the development of the task, in JPEG format. Also, the amount of frames per second is configurable to avoid the saturation of the network.

This window also shows the advance and the steering speeds of the robot. In addition, using the arrow buttons, the student can operate the robot with basic movements. Using the top menus, the user can decide which data he wants to save while the application is running and the name of the files to save these data, and which data he wants to be visualized in real time, while the robot is performing the task. These charts are shown on independent windows (fig. 3).

The first step when the student runs this application consists in getting a working environment (if any available). This option is accessible from the menu Connection, that will show the window on fig. 4. Each student is assigned a login and a password to access the system. If these data are correct, when clicking on the button 'Find Robots', the system looks for any available environment and if any existing, the robots are shown in the 'Available Robots' textbox. The student will select one of them, and, after clicking 'Connect', he will access the basics navigation window (fig. 3).

When a student accesses one working environment to carry out the practices, he has the exclusive rights to operate with the robot. None of the other students can access this environment until the current user leaves it. Besides, to avoid that only a student can monopolize the environment during a long period of tine, a reservation system has been established. Through it, each student can make a reservation during a time slot in a timetable (the maximum period of time is limited to one hour in the current implementation). During this time slot, only the student who has made the reservation can access the working environment.

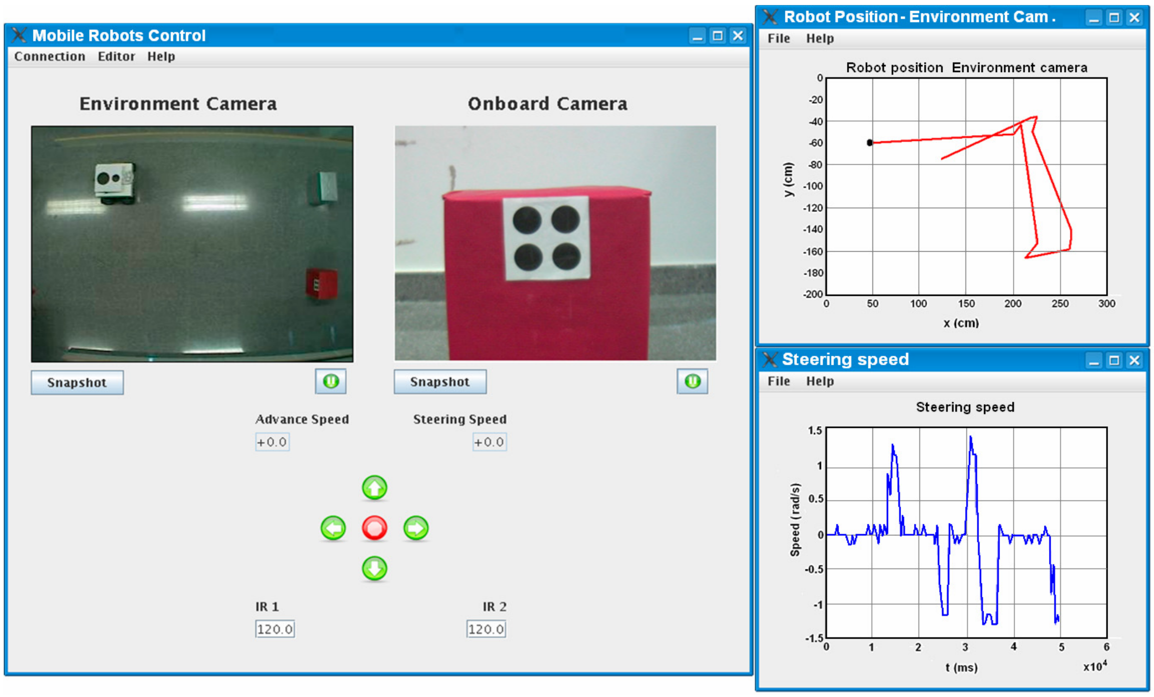
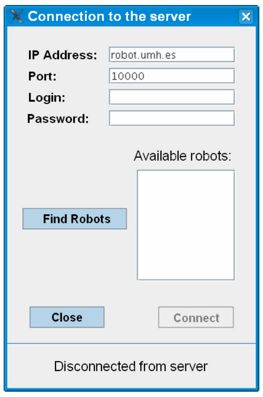


Fig. 3. Appearance of the graphical interface of the client application

The identity server is in charge of managing the users' queue.

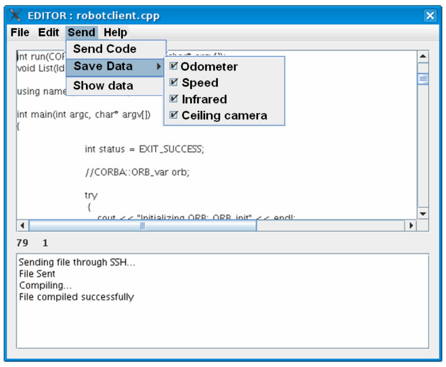


```
IP Address: robot.umh.es
Port: 10000
Disconnected from server
```

Fig. 4. 'Connection' window

The graphical interface puts at the disposal of the student the basic control buttons and the data from the sensors. However, if the student wants to solve a more complex task, including both computer vision and robot control, he has to develop a control algorithm in a high level language (C language in this case) and he has to compile and run it in the remote environment, and monitor the task.

To start this process, the student has to access to the 'Editor' window (accessible from the 'Editor' menu in the main window), shown on fig. 5, where the student can write, send and compile the program. This compilation is carried out in the server. If the compilation fails, the errors or warnings are shown on the bottom text box of the Editor window.



```
int run(COF
void List(Id

using name

int main(int argc, char* argv[])
{

            int status = EXIT_SUCCESS;

            //CORBA::ORB_var orb;

            try
             {
               cout << "Initializing ORB: ORB_init" << endl;
```

```
79    1
```

```
Sending file through SSH...
File Sent
Compiling...
File compiled successfully
```

Fig. 5. 'Editor' window

Once the program is correct, before running the task, the student can select the data to be recorded and stored in some text files while the robot is running the control algorithm, and the data he wants to visualize in real time. These options are accessible from the 'Send' menu (fig. 5).

5. SYSTEM PROTECTION MEASURES

Several protection measures have been implemented to avoid malfunction of the system. First, to preserve the application from unknown users, a SQL database has been implemented. It restricts the access to the application to those users that have been previously included in the database, through a login and a password. This verification is carried out before checking which robots are available in that moment. Secondly, this database also registers all the actions the users have carried out in the system, from the verification of the student until he leaves the system and the environment is released.

Apart from this safety measure, another prevention mechanism has been implemented to avoid malfunction of the robots. To do it, an independent thread has been programmed. This thread runs concurrently with the user program and it is in charge of monitoring the advance and steering speeds of the robot, the infrared sensors and the batteries level. In case that any speed is over a threshold, or the robot has a wall or obstacle very near, this thread can stop the robot and reinitialize the algorithm. If the battery level goes under a threshold, the robot is stopped and the student is assigned a different environment to test the algorithm.

6. PRACTICAL EXPERIMENTATION

Several practical sessions are proposed to the students to be carried out through this platform.

6.1 Basic control of the robot

The student must create a program so that the robot moves forward, describing a straight line. The robot must avoid the obstacles it finds in its way, and it must stop when it has travelled a distance of three meters. The student has to make use of the infrared sensors (to detect the presence of obstacles), of the odometer data (to compute the position of the robot and the distance it has travelled) and of the images captured by the ceiling camera (to know the position of the robot at each moment and compare it with the results computed using the odometer data).

In general, the robot has to move straight, and it has to describe a 90º arc when it finds an obstacle. The robot must turn to the side where the obstacle is farthest from, and it must stop when the distance it has travelled is higher than three meters. To compute this distance, the student is provided with a function that integrates the odometer data and that returns the coordinates of the robot with respect to the initial position. The problem must be solved using a competitive coordination of two behaviours, 'Go straight' and 'Turn'. The students are advised to design a flow chart of the task before writing the program.

Once the robot has fulfilled the task, the student has to make a graphical representation of the path the robot has followed, using the odometer data. After this, the student must draw the same trajectory, but computed from the information provided by the external camera. A comparison and a critical analysis of the results are demanded.

As the main objective of these practices is to empower the knowledge in robotics control, we provide the students with some templates to develop the algorithm, and with some methods they can use in their programs. There is a specific function that computes the position of the robot using the information of the ceiling camera.

6.2 *Going to a target point*

Taking as a basis the previous exercise, the student must make the necessary changes in the program so that the robot moves from its initial position to a target point, trying to avoid the obstacles it finds. An artificial landmark on the target point indicates its situation. This landmark can be viewed from the ceiling camera so, analysing the images of this camera, its position can be inferred. As in the previous exercise, the robot must move straight to the target point, turning when it finds an obstacle. Once the obstacle has been surpassed, the robot must orientate again to the target point and move straight to reach it.

The problem has to be solved through the implementation of two behaviours: 'Go to destination' and 'Avoid obstacle'. At each moment, only one of these behaviours must be active, depending on the reading of the infrared sensors (presence of an obstacle).

6.3 *Visual control of the robot*

The goal of this session is to introduce the concepts of visual control of a robot. The student has to make use of the *OpenCV* library, that includes interesting functions for image manipulation.

In the working environments where the robots move, there are some visual marks that are visible from the camera each robot carries on it. Firstly, the robot must move with a random movement, trying to avoid the obstacles, until one of these marks appears in its visual field. From this moment, the robot has to try to situate at a determined distance from the mark, and it must stop when this goal is achieved.

This problem must be solved using competitive coordination of the behaviours 'Random movement' and 'Go to mark'. Only one of these behaviours must be active at the same time depending on the presence of the mark in the visual field of the robot.

7. CONCLUSION

This work has presented an interactive tool we have implemented so that students can monitor and control the evolution of a team of mobile robots through Internet. With this platform, they can deepen in the knowledge of basic concepts of mobile robots, computer vision and their relationship.

Several servers have been implemented so that the students can operate the robots in a transparent way, independently of the internal architecture of the robot. Also, these concepts could be extended to other kinds of robots so, we could add new environments or change one or more robots without necessity of modifying neither the Identity Server nor the Client Application.

Future work will include the implementation of communication between the robots, so that the remote user could test some algorithms that imply collaboration between several robots to solve the task.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish 'Ministerio de Educación y Ciencia' through the project DPI2007-61197, 'Sistemas de percepción visual móvil y cooperativo como soporte para la realización de tareas con redes de robots'.

REFERENCES

Borgolte, U. (2008). Considerations on a remotely operable mobile robot laboratory. *Proceedings of Virtual University 2008,* Bratislava, Slovakia.

Candelas, F.A., S.R. Puente, F. Torres, F. Ortiz, P. Gil, J. Pomares (2003). A virtual laboratory for teaching robotics. *International Journal of Engineering Education,* **no. 19(3)**, pp. 363-370.

Dormido, S. (2004). Control learning: present and future. *Annual Reviews in Control,* **no. 28**, pp. 115-136.

Guzmán, J.L., M. Berenguel, F. Rodríguez and S. Dormido (2008). An interactive tool for mobile robot motion planning. *Robotics and Autonomous Systems,* **no. 50**, pp. 396-409.

Jiménez, L.M., R. Puerto, O. Reinoso, C. Fernández, R. Ñeco (2005). Recolab: laboratorio remoto de control utilizando matlab y simulink. *RIAI, Revista Iberoamericana de Automática e Informática Industrial,* **no. 2(2)**, pp. 67-72.

Khamis, A., M. Pérez-Vernet and K. Schilling (2002). A remote experiment on motor control of mobile robots. *Proceedings of 10th Mediterranean Conf. on Control and Automation,* Lisbon, Portugal.

Khamis, A., D.M. Rivero, F. Rodríguez and M. Salichs (2003). Pattern-based architecture for building mobile robotics remote laboratories. *Proceedings of IEEE Int. Conf. on Robotics and Automation,* Taipei, Taiwan.

Marín, R., L. Nomdedeu, R. Wirz and P.J. Sanz (2009). Robotics Internet Tele-Lab: programming using mobile devices. *Proceedings of Int. Conf. on Multimedia and ICT in Education,* Lisbon, Portugal.

Masár, I., A. Bischoff and M. Gerkes (2004). Remote experimentation in distance education for control engineers. *Proceedings of Virtual University 2004,* Bratislava, Slovakia.

Payá, L., M. Juliá, O. Reinoso, A. Gil, L.M. Jiménez (2006). Behaviour-based multi-robot formations using computer vision. *Proceedings of 6th IASTED Int. Conf. on Visualization, Imaging and Image Processing,* Palma de Mallorca, Spain.

Payá, L., O. Reinoso, A. Gil, L.M. Jiménez (2007). Plataforma distribuida para la realización de prácticas de robótica móvil a través de Internet. *Información Tecnológica,* **no. 18(6)**, pp. 27-38.