ROBOT CONCURRENT REMOTE OPERATION THROUGH INTERNET

Alberto Vale, Paulo Inácio, Nuno Antunes,
João Sequeira, Maria Isabel Ribeiro

Instituto Superior Técnico – Instituto de Sistemas e Robótica
Av. Rovisco Pais 1, 1049-001 Lisboa – Portugal
e-mail: {vale,jseq,mir}@isr.ist.utl.pt

Abstract: This paper describes an architecture for the remote operation of a mobile platform by multiple users using Internet. A client/server approach, developed using Java, provides the users with multiple control and sensing functionalities. Experimental results obtained under regular network traffic are presented.

Keywords: Remote Operation, Internet, Human Interface, and Protocol

1. INTRODUCTION

Economical, environmental and safety concerns are leading to an increase in the use of robots. Given the complexity of realistic environments, full autonomy in robotics is yet to be achieved and hence semi-autonomous robotics is being established as a major research line (IEEE Robotics & Automation, 1999). The use of Internet as a communication channel further extends the application examples, namely in hostile environments such as mining, nuclear waste disposal and surveillance. Depending on the kinematics of the robots and of the particular application considered it might be necessary to allow simultaneous access of multiple users to a single robot. A typical example may be a mobile manipulator with different users managing separately the mobile platform and the manipulator.

The Internet is known for the non-stationary characteristics of the traffic, which represents a harsh constraint in the application of standard control techniques. The common approach to this problem is to move the control problem from the class of pure remotely operated systems (which can be identified with tele-operation) to the class of semi-autonomous systems. Semi-autonomy considers the use of intelligent control techniques to minimize the practical effects of low bandwidth communication channels. The web browser's technology evolution simplifies the access to a worldwide scale communication channel and hence the use and design of remote operation tools. Moreover, current web browsers are almost hardware platform independent and hence allow the sharing of software components.

The training of users in the operation of a robot, using the expertise of different trained users in complex missions for safeguards or the establishment of cooperative interactions between multiple robots and users are just two examples where the concurrent operation is useful. Priority and arbitration mechanisms are used to manage multiple users in the access to a single robot.

This paper addresses three major issues in the remote operation of robots: the user-robots interface, the communication channel and the concurrent operation of a robot by multiple users. A single robot is considered in this paper. The robot to be remotely operated, shown in Figure 1 with a manipulator

mounted on top, is a cart-like mobile platform with 24 ultrasound sensors and an odometer system.

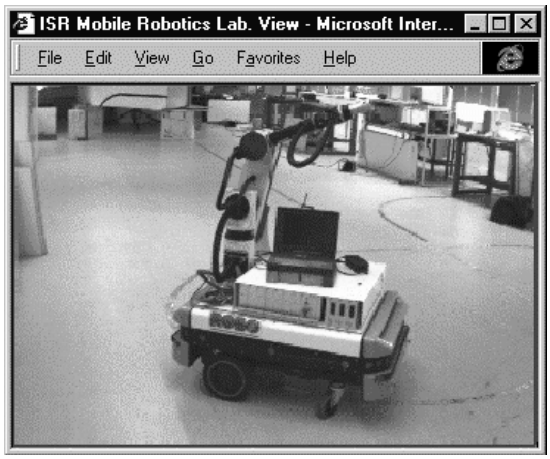

Fig. 1. Robuter image on video window

This paper is organized as follows. Section 2 presents an overview of the complete work presented in this paper. Sections 3 to 5 detail the communication protocol, security issues and user interfaces for the architecture modules. Experimental results are presented in Section 6. Section 7 concludes the paper and presents directions for further developments.

2. OVERVIEW

This work aims at developing an architecture for the concurrent remote operation of a mobile platform, through Internet, as illustrated in Figure 2. To ensure high portability between remote computational facilities, the software was required to be platform independent.

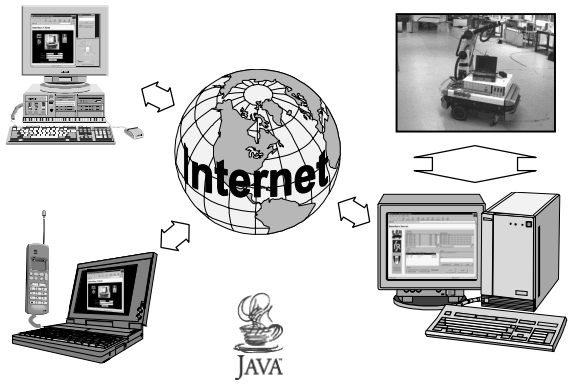

Fig. 2. Concurrent Remote Operation

Concurrency leads to a modular client-server approach. The server application manages all exchanges between users and the robot. The users have access to a number of tools with which they interact with the robot. Each of these tools is a client in the architecture. The strategy for the development of the system has been divided into three main steps described bellow:

- Specification of the basic architecture supported on Internet. The portability requirement led to the use of the Java language (Zukowski, 1998; Ammeraal, 1998). Moreover, Java's applet technology provides a secured way of exchanging small functionalities between clients and server and data encryption ensures security in communications.
- Definition of the basis tools through which the users interact with the robot. These tools include the client applications that receive and process sensor data and the applications through which the users command the robots.
- Design of two GUIs – Graphical User Interfaces to integrate clients and to access the server. Both interfaces were written in Java (Flynn, *et al.*, 1997; Stephen, 1998).

Beyond the automatic management, the server application still allows the manual management of users and full platform control by a system administrator.

3. BASIC ARCHITECTURE

The basic architecture is client-server type (Jeffery, 1996). Communication between server and clients uses a predefined protocol. The architecture is open in the sense that there are no constraints on the number of connectable clients.

Each tool available for the user to interact with the robot is a client. Users access clients through a web page after passing an identity authentication step. This web page constitutes the user interface.

The server main goal is to handle the messages received from the users, sending them, if appropriate, to the robot and returning any answer back to the users.

Each client has a priority assigned by the server. The order in which messages from/to the robot are routed from/to the users depends on the priorities assigned to the corresponding owners. The server has a database where the profile of every user is kept. This profile includes the users hierarchy and the list of tools that each user can access.

Encryption algorithms were applied on messages exchanged between users and server. The Blowfish and the R.S.A. (Rivest, Shamir e Adleman) algorithms (Ferreira, 1997) were chosen to encrypt the data exchanged between the users and the robot.

4. USER INTERFACE

To operate the platform, a remote user starts a session in the web page that establishes a connection

with the server program, after validation of user identity (see Figure 3).

Fig. 3. User interface web page

The user interface enables the user to access the tools through which it interacts with the robot. The main tool is the user interface itself. Once logged in, the user is informed of the available tools. Figure 4 shows a graphical arrangement for the standard set of tools.

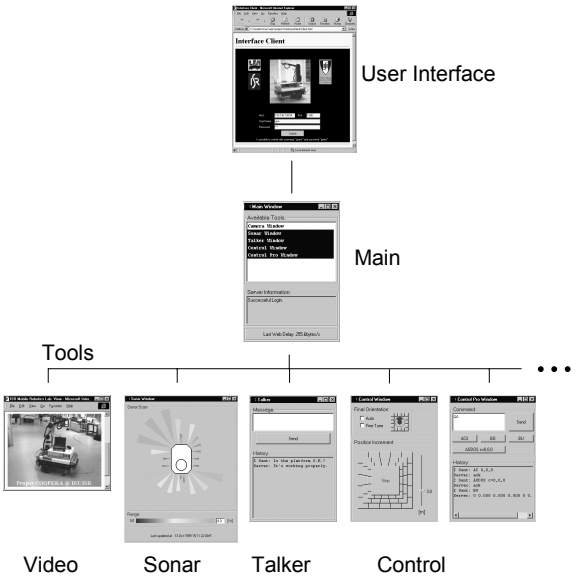

Fig. 4. Standard tools

Relevant messages from the server and the statistics on the communications channel are also displayed at the user interface.

Each tool interacts with the user through its own window. The main tool is a small floating window to simplify its display in the user screen area. Figures 3, 6 and 7 illustrate the graphical appearance of three tool windows. Each of these tools is independent of the others and is detailed ahead in the paper.

The Main Tool

The main tool provides the user with the choice of the tools to control the platform. The tool's structure (see Figure 5) was drawn in order to make use of the multi-threading processing. This tool manages the messages exchanged between the user and the server. To make this possible, the main thread receive all the server's commands and sends back to the server the answers to the received commands.

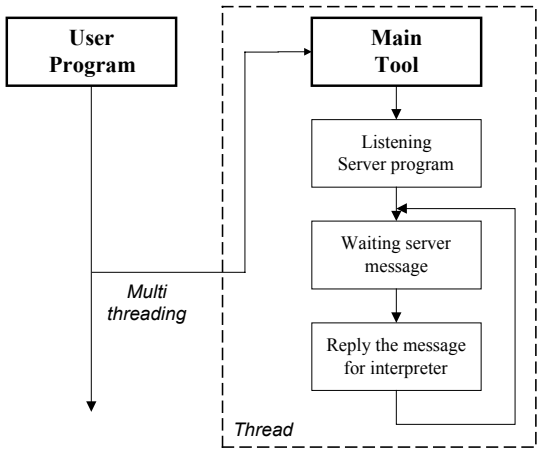

Fig. 5. Diagram of main tool

Video image tool

Any user can access this tool. It simply shows the video images captured through a web-cam, typically placed in the robot working area (see Figure 1 for a view of the test environment).

Ultrasonic sensor tool

This tool complements the video image tool. It displays the measurements of the ultrasonic sensors installed in a ring around the robot. The associated display window is divided in three areas: the visualization map, the range scale, and the most recent sensorial data. In the map center, the platform is represented with the front facing the top of the window (see Figure 6 for details). The occupied space around the robot, perceived by the ultrasound sensors, is represented by a set of cones. The height of each cone represents the distance at which an obstacle has been detected in the corresponding direction.

Talker tool

This tool establishes a communication channel between a user and the system administrator or among users. This tool allows the interaction among users geographically apart that are operating the same robot or a set of robots.

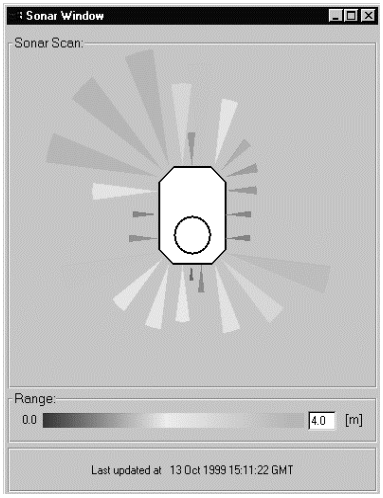

Fig. 6. Ultrasonic sensor tool window

Command/Control tool
The robot motion commands are generated with this tool. A "graphical joystick" enables the user to select a wide range of motion directions and wheel velocities.

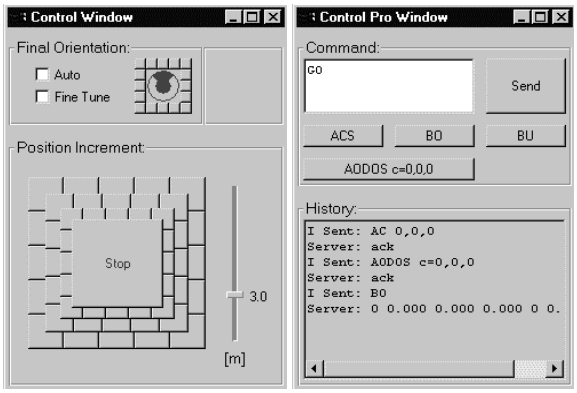

Fig. 7. Control tool and advanced control tool

Each time the user presses one of the buttons in the tool display, a motion command is sent to the server. In this tool two classes of motion buttons are available: one for platform rotation and the other for translation through the desired direction. All the motion and orientation commands in this tool are relative to the platform reference frame. In this way, the agreement between graphical display in the tool and the ultrasound data visualization tool is complete.

Advanced control tool
This tool implements a dispatcher of commands to the robot using the motion commands syntax directly. This requires a minimum knowledge of the robot kinematics and of the motion command language. This tool represents a remote terminal for the robot onboard computer and is especially useful in debugging any additional tool that needs to generate motion commands.

5. SERVER INTERFACE

The system administrator operates the server program. It is configured from two files, loaded at program startup. If these files do not exist the program immediately finishes by security reasons. A file with the logging of the events is created in case of an emergency stopping.

The graphical interface aims at simplifying the system administration by displaying all the information passing through the server (see Figure 8 where the interface server is displayed). The display area is divided in four areas. The top-left area displays, in real time, all the messages exchanged between the users and the server. On the top-right is displayed, also in real time, the commands sent to the platform and any reply messages are displayed also in real time. In the bottom-left area the user can exchange messages with the system administrator and with any other user logged in the system. The last area displays a list identifying the logged users and a summary of their current status.

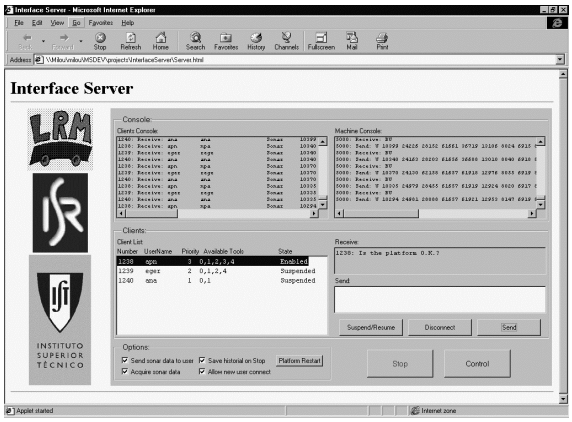

Fig. 8. Interface Server

The server structure is supported in two main threads as represented in Figure 9. One thread handles the robot whereas the other handles the new users. This second thread launches a new thread for each new user that logs in.

The messages received by the server are first parsed for validity according to the protocol displayed in Figure 10. When a message from a talker tool is received, it is displayed together with the user identification, in the related area of the interface. When a message from a control tool is received the server verifies the command type in the message content. If it is a stop command, the server sends it to the platform, waits for a reply and sends this reply to the control tool. If it is a motion command the server monitors the position error. Whenever this error is larger than a predefined value, the server automatically generates an additional motion

command to adjust the robot position. The final configuration is replied to the control tool window.

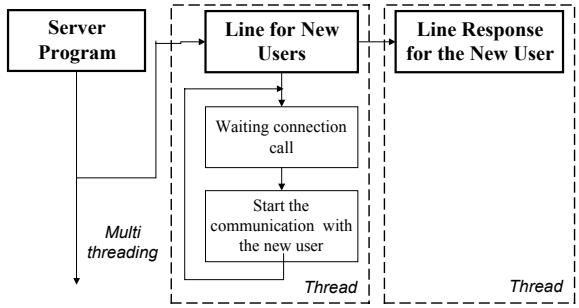

Fig. 9. Basis users management flowchart

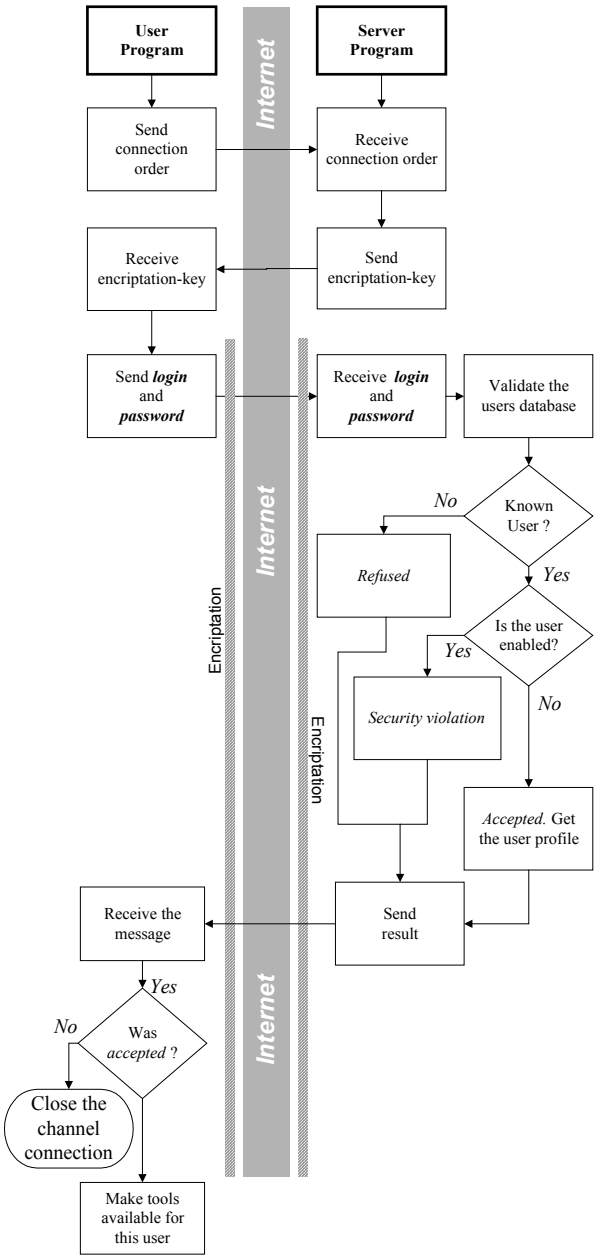

Fig. 10. Diagram of the validity protocol

When a message is originated in an advanced control tool, the server sends the content of the message directly to the robot and then waits for a reply. After receiving the reply the server program sends it to the user that originated the initial message.

The emergency stop is to be used in imminent danger situations to immobilize the robot. When an emergency stop is invoked, the server starts a procedure to send an emergency stop command to the robot. This procedure starts with the immediate suspension of all currently logged users having the privilege of sending motion commands. After the robot immobilization a history file is created, allowing the posterior inquiry on the emergency situation causes.

6. EXPERIMENTAL RESULTS

This section presents a set of experiments, carried out in such a way as to simulate real working conditions. The users could only interact with the platform through the available tools. The video image tool and the ultrasonic tool assumed a major role, because these are the only way to know the platform environment. The web-cam was located in a strategic point, allowing the visualization of all the workspace. The communication infrastructure that supported the test session was a LAN (local area network) of Instituto de Sistemas e Robótica (Lisbon pole).

Each test was running during one hour with ultrasound sampling rate of 4 seconds. All user tools were used to control the platform. To determine the minimum data transfer rate value, acceptable for remote operation, four tests were carried out in different hours of the day, which means different network traffic conditions. The communication speed of the messages exchanged between client and server oscillated among the 390 bytes/s and the 505 bytes/s with an average value of 450 bytes/s. These measurements were made with the following procedure: the server sent a message to the client, who replied with another message. As the length of these messages is known, the number of bytes exchanged was divided by the time spent in this transmission. In these measurements, the time spent in the information processing by a Pentium computer was discarded when compared with the communication speed. Some of these tests were made with different users, that were controlling the platform simultaneously. Each of these users had a different priority assigned by the server, which was verified by automatic management operating. Intentionally, an emergency situation was created, in which the server must immobilize the platform. This action was fulfilled through the emergency stopping of the server interface.

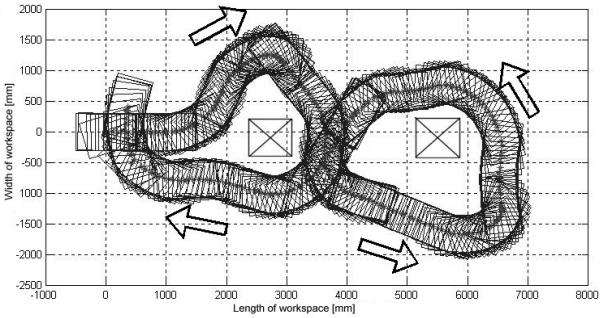

Fig. 11. Path with an "8" shape. Ultrasound sampling rate of 1s

In the first test the operator had to command the platform, using the available tools, along an obstacle contour, with an "8" shape path (see Figure 11). The second test was identical to the first one, but with an ultrasound sensor sampling interval of 1.25s (see Figure 12). The comparison between Figure 11 and 12 shows, in this last one, an increased number of situations where the user had to stop the platform to re-adjust its orientation. This is due to a lower refreshing rate of the environment in the platform's vicinity.

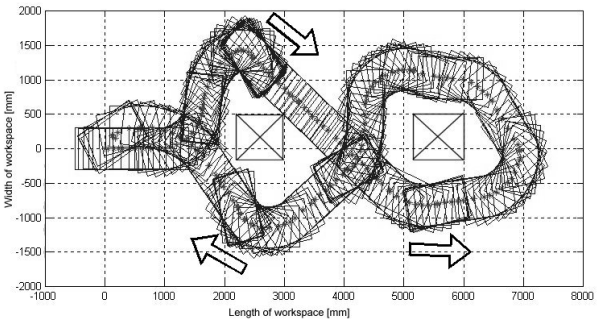

Fig. 12. Path with an "8" shape. Ultrasound sampling rate of 1.25s

In the third test, the user directs the robot along a path delimitated by barriers (see Figure 13). The resulting trajectory shows clearly three stop-and-reorient maneuvers due to odometer errors. The sampling interval of the first test was used. Figure 14 shows the results of ten trials of the third test where each "*" represents the origin of the robot reference frame. The density of "*" shows a good system repeatability.

It is worth noting that near the curve the web cam did not help the user to correctly perceive the orientation of the robot. The ultrasound sensor tool overcame this problem.

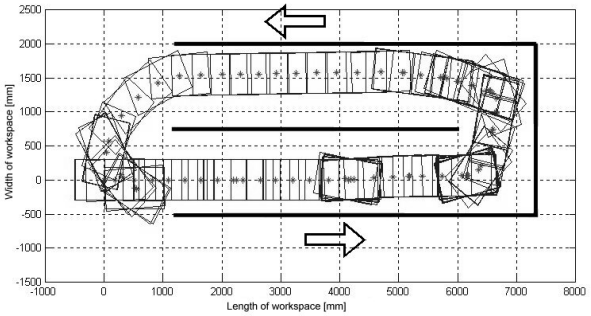

Fig. 13. Path with a "U" shape limited by barriers. Ultrasound sampling rate of 1s

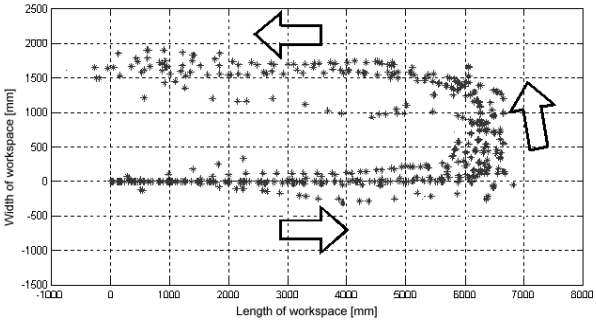

Fig. 14. Last experience repeated 10 times. Ultrasound sampling rate of 1s

7. CONCLUSIONS

From the aforementioned tests the feasibility of the remote operation through Internet is verified. Despite the simplicity of the test environment the architecture proved to have the necessary functionalities to fulfil the goal and it could be possible by concurrent users.

Still some future developments are to be considered, namely the inclusion of new interaction tools with the robot, and the use of a data compression algorithm to save some bandwidth.

8. REFERENCES

Ammeraal, L. *Computer Graphics for Java Programmers*. Sybex 1997.

Craig, J. *Introduction to Robotics, Mechanics and Control*. Addison-Wesley Publishing Company, 1991.

Ferreira, P. *Suporte de Aplicações Distribuídas*. L.E.I.C. – I.S.T. 1997.

Flynn, J. and Clark B. *Visual J++ Programando em Java*. Makron Books 1997.

IEEE Robotics & Automation, Vol. **6**, No 3, September 1999.

Jeffery. *Advanced Windows*. Microsoft Press, 1996.

Stephen, D. *Learn Java Now*. Microsoft Press, 1998.

Zukowski, J. *Mastering Java 1.2*. Sybex, 1998.