Teleoperating a Mobile Robot*

A Solution Based on JAVA Language

F. Monteiro, P. Rocha, P. Menezes, A. Silva, J. Dias
Departamento de Engenharia Electrotécnica
Instituto de Sistemas e Robótica
Pinhal de Marrocos, 3030 Coimbra, Portugal
{fmanuel, pmrocha, paulo, arlindo, jorge}@isr.uc.pt

Abstract - This article describes a computational framework, which can run almost on every computer connected to a IP based network, to study teleoperation techniques.
The paper discusses the teleoperation of a mobile platform from a remote computer, both connected by the Internet communication network. To obtain a cheap and portable solution, the user interface and communication management was developed using Java language. The development based on this language gave us, a cheap, portable and easily to improve software package that is useful to test new control techniques with large time delays. Since the real time control over Internet can suffer from unpredictable delays and bandwidth constraints, we use the RTP protocol to overcome some of these problems.

I. INTRODUCTION

Several research groups have recently turned their attention to the development of teleoperated systems due to the every day appearance of new potential applications that can benefit from the use of this kind of solutions. The recent evolution of microprocessors and communication systems, and the wide availability of simple and cheap solutions using these systems, have also attracted many people to an area that traditionally was only accessible to restricted groups with strong financial support.

Among the applications of teleoperated systems we find those related with the exploration and manipulation in hazardous environments, like nuclear power plants [9, 10]. The research activities in this area are quite diverse, going from the development of communication links, to the search for an optimum user interface for a certain application. Here we can cite some of the major works that have been developed in the recent years:

- Space applications: NASA JPL labs are developing techniques since the beginning of space exploration. These techniques are being used on the development of a mobile vehicle that will integrate the Mars Mission[5]. This vehicle will be controlled from Earth and will transmit images of the Mars surface, being the major problems the small bandwidth available and the enormous delays that raise from the large distances involved.
- Volcano exploration: These extremely adverse environments have already caused the death to hundreds of researchers. This has motivated the development of mobile robots capable of performing such exploration. Among these we can cite another NASA project called DANTE[4]. This is based on a 8 leg robot that can be teleoperated safely miles away from the volcano.
- Tele-surgery: This recent technique is under development, it's purpose is to enhance surgeon ability in dealing with traditional endoscopic devices. Teleoperation systems, specifically BFR (Bilateral Force Reflection), improves the feedback received by the surgeon. Under development are also systems devoted to remote surgery, where the surgeon can be miles away from the patient.

Teleoperation and remote surveillance related techniques have been exploited recently at the Institute of Systems and Robotics - Coimbra resulting in the development of projects like :

- A surveillance system using artificial vision [1].
- Remote surveillance using a mobile robot equipped with an active vision head [2]. Both the mobile robot and the camera head were controlled remotely by a 6 degrees of freedom joystick.
- Teleoperation of a Puma manipulator [3]. In this project the operator receives the necessary feedback from a three dimensional representation of the manipulator and from a camera mounted on the robot tool.

The challenge in the development of the project that this paper describes results from the use of a platform independent language and the Internet Protocol as the communications support. The advantages of using this kind of approach arise from both the wide availability of the communication media and the development of an

*This work was partially financed by NATO Scientific Affairs Division on the framework of the NATO Science for Stability Program, PO-ROBOT project.

application that is expected to run in a wide variety of platforms.

The aim of this project is to perceive the difficulties felt by an operator in controlling our robots, when the communication link presents small bandwidth and/or large delays and to create techniques to overcome these problems.

II. MAIN DIFFICULTIES

Most of the difficulties still present in teleoperation issues are related with communications problems. Economical, technical and physical reasons, all contribute to make the dealing with a low bandwidth, high latency, communications link the core of teleoperation. When we try to long distance teleoperate any kind of robot, delay and bandwidth problems also become of great concern to us.

Delay is unavoidable and, obviously, increases with distance. Studies[7] have shown that delays of 1/4 of second are noticed by the operator and for higher values his performance rapidly becomes degraded. As a result of the delayed feedback to his actions the operator becomes more and more stressed as time elapses. Human mind performs much better if there is a continuous flow of action-response sequences. However, communications delay is not the only stress source that will affect the human operator:

- Stress and fatigue can also be caused by the operator having to control every aspect of the robot's performance: this imposes a high level of concentration which will be hard to sustain over time, causing errors to become more frequent and overall efficiency to be reduced. This leads to the important question of the robot's level of autonomy. While teleoperation may not seem the right place to address this question, we think it is essential to give the robot the right level of autonomy, so that the operator will be spared of the executing repetitive, low-level tasks, which will not only stress him but also increase the communications overload. Robot autonomy can also be the answer to one of the most serious drawbacks of teleoperation: what will the robot do if he steps into a dangerous situation and the operator can not intervene, e.g. due to communications delay or simply by being distracted? Its level of autonomy will dictate the outcome...

- Not directly related to communications, but still a main difficulty in teleoperation, is the development of an interface easily understood by the user. When new data arrives it must be represented in such a comprehensive and logical way that the robot status will be immediately apprehend by the operator. Only this way we can expect a prompt and correct response from the operator to any unexpected situation.

We believe that, to make a teleoperation environment useful and operator-friendly, these three issues, i.e. communications, robot's autonomy and interface, must be seriously taken into account.

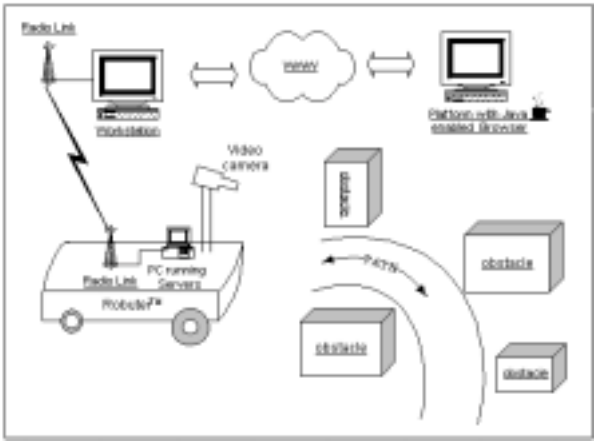

Fig. 1: Communication system.

III. SYSTEM ARCHITECTURE

As can be seen in figure 1, this system takes advantage of existing components and facilities.

The user interface is provided via a JAVA enabled WWW browser on a platform with Internet connection. The browser is responsible to provide the runtime platform independent support (the JAVA Machine), making possible to run the user interface application in any machine without requiring any modification or recompilation. The distribution of this application is made using the same browser application from a Web Page. This application sends commands and receives sensorial data through the Internet to a computer that acts as a gateway between the Internet and the robot. This gateway is responsible for keeping the conversation with the remote client and for converting the high-level commands in the robot low level commands.

A. COMMUNICATIONS

The use of the Internet to support the communications between the operator and the robot is quite attractive due to its worldwide availability, which makes possible to control the robot from virtually everywhere in the world. The disadvantages come from the fact that the Internet Protocol does not guarantee neither constant bandwidth nor reliable transport of messages. At first sight, part of the problem could be solved using the Transport Control Protocol, because it guarantees the ordered delivery of messages, but, as can be seen from the following example, it poses other problems that makes it unsuitable for this kind of application. If, for instance, the first of a sequence of five messages is lost, this protocol attempts to recover it by forcing its retransmition, until it is correctly received or the link is considered as dead. The TCP protocol delays any received messages until they can be delivered by the same order they were sent. This kind of behavior is not desired for a real-time application where in most cases an out of date data does not worth anything. In other words, if a message can't be delivered within a predefined amount of time it should be discarded instead of

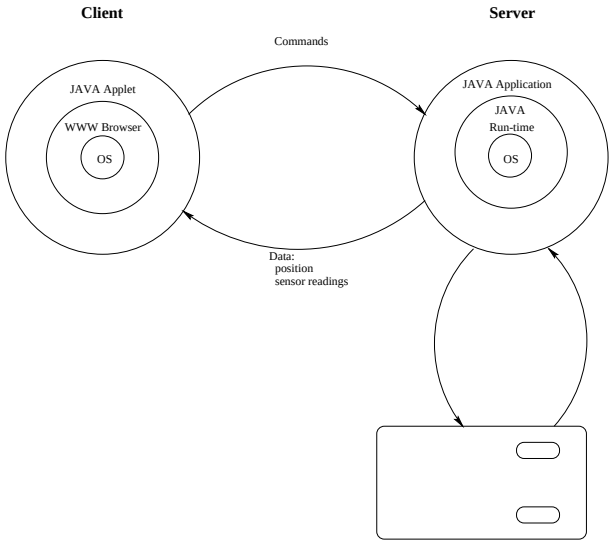

Fig. 2: Software Architecture.

delaying other messages that follow.

From the above stated, one can easily see that only datagram based protocol can be used because each message is independent and its own delivery does not depend on the correct reception of the previous ones. A question may now be posed: "If there is no guarantee of the delivery of messages, how can this communication system be used on the control of a device like a robot?" As will be referred later, this problem can only be solved by adding some degree of autonomy to the robot allowing it to avoid dangerous or unwanted situations.

Instead of using the simple Internet datagrams we have chosen a protocol that was proposed for real-time applications, the Real-Time Transport Protocol (RTP) [6].

In addition to provide the (unreliable) transport of messages between the peers, this protocol provides additional informations that are useful in this application.

- **Peer Identification** This information is very important because as the robot can not safely used by more than one operator.
- **Round-trip delay** Low-level commands can not be used if the communications delay is above a certain level, so this parameter can be used as a reliability meter.
- **Interarrival Jitter** Can be used as a quality measure of the link.
- **Percentage of packets lost** Same as the above.

B. USER INTERFACE

Since we are using the Internet as the communications medium, it seems obvious the develop all software modules in a language both Internet oriented and with multi-platform support. JAVA is the language which best satisfies these needs.

JAVA allows us to abstract from system architecture, since an applet written in this language will run in any platform with a World Wide Web (WWW) browser having the necessary runtime support. These browsers can be found for almost every general purpose platforms available in the market. A potential operator only has to correctly specify the Uniform Resource Locator (URL) address of the robot's control web page, to be allowed to operate it remotely. Security schemes can always be implemented, but broad access can always be useful, e.g. for educative purposes.

When an operator using a JAVA enabled browser retrieves the robot's control web page, he will be presented with a JAVA application: an applet. This applet is the client side of our system and interfacing with it is the only contact the operator will have with the system. This applet handles all operator-system interface by accepting operator given robot commands and displaying all information needed by the operator to correctly choose such commands.

It is by now obvious that this applet is the core of what we call the User Interface (see fig.3). Other applets can be deployed to perform secondary tasks (e.g. see the joystick-like applet in fig. 4). The main applet presents to the operator a model of the robot's environment as well as two representations of the robot itself. One of this (the lower one in fig.3) represents what we call the **confirmed** robot's configuration, which is the last one the client received information about. With the robots used in our experiments range information obtained from ultrasonic sensors readings is also presented. If communications delay is meaningful, and usually is, the **confirmed** configuration is already obsolete when arriving at the client. This fact causes two main problems to the operator: in one side it leads him to make control decisions based on obsolete information, and, on the other side, the delayed feedback to the operator's commands will, as already mentioned, stress him in the long term. To try to overcome both this problems a second configuration is presented, the **expected** configuration. This is the configuration we expect the robot to occupy at present time, estimated with base in three information sources:

1. Previously received data, i.e. the history of the configurations known to have been occupied by the robot until the present moment.
2. A transmission delay report, such as the one given by the RTP communications protocol.
3. The commands given by the operator since the last information about the robot's configuration was received.

If the robot stops for a long enough period, the **confirmed** configuration should get nearer the **expected** one. The operator can now see immediately the results of his commands by following the behavior of the **expected** configuration, and this also gives him an approximation of the robots actual configuration. By watching the **confirmed** configuration path he can make sure the robot behaves as predicted.

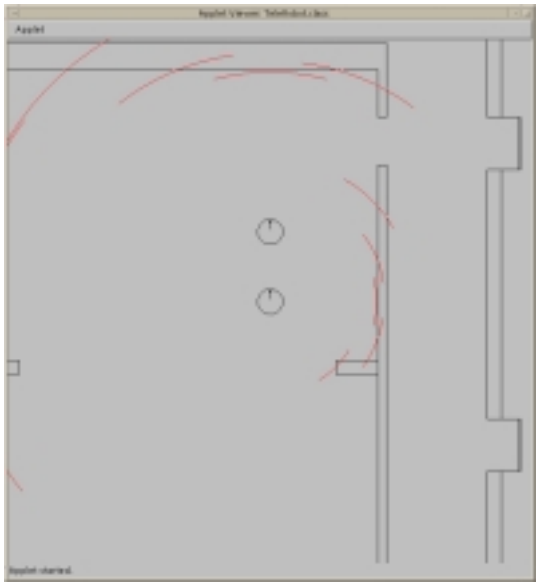

Fig. 3: Teleoperation interface, showing the expected robot (first from top) and the confirmed one. The arcs are the sensor readings.

Fig. 4: Joystick-like control applet.

To actually send commands to the robot two distinct modes are available to the operator, a high-level mode and a low-level mode:

- **Low-level mode -** In this mode, a joystick-like applet is available to the operator (see fig. 4). Simple commands like *forward, backward, left, right* or *stop* can be sent to the robot pressing the correspondent buttons on the applet. This is the most straightforward way for an operator to control the robot.
- **High-level mode -** In high-level control mode, which is still under development, the operator will be able to give goals to the robot by clicking in the environment's model, as well as some way-points, if desired.

Both this modes imply a certain degree of autonomy from the robot, but, while in the low-level control mode simple "virtual bumpers" to avoid collisions are enough, in high-level mode more intelligent behaviors, such as path-planning, must be implemented.

C. THE SERVER

The server plays two different roles in our system. The simplest of this role is the receiving, analyzing and sending to the robot of the commands arrived from the interface client, as well as sending to the client the current robot's configuration and sensor readings. The more complex role involves the implementation of the robot's autonomy abilities. The purpose of this abilities is not only to make the system fault-tolerant, by correcting or ignoring dangerous commands from the operator, but also to automate repetitive and low-level tasks in order to reduce the operator's stress, by decreasing its workload, as initially purposed. Two levels of autonomy, related to the two control modes, are present in the system:

- In low-level control mode the robot's autonomy is reduced to a sort of "virtual bumpers" which stops the robot when he gets to near to an obstacle, either due to an operator's command or to lack of one. This last situation can arise both by an operator's error or communications breakdown.
- Higher level autonomy is needed when the high-level control mode is in action. If the operator chooses a new goal to the robot, it must be able of planning and following a path to that goal, avoiding known or even unexpected obstacles. We propose to implement this step by integrating in our system path planning and obstacle avoidance modules already developed at the ISR.

Both autonomy levels, but specially the second one, contribute effectively to reduce the workload of the operator, decreasing the stress and fatigue caused by the constant monitoring of the robot. Robot autonomy also protects him from problems resulting from communication breakdowns, which increases the system fault-tolerance ability.

IV. PRELIMINARY RESULTS

Although this project is still in a early development stage, some tests have already been made using two different mobile robots: The first, a Robuter (see fig. 5) is a cart like robot, while the second, a Nomad (see fig. 6) has a circular base without holonomic restrictions. Both platforms are equipped with a ring of ultrasonic Polaroid sensors and odometry systems.

We have successfully controlled both robots, guiding them easily along a corridor, through the use of the interface developed. The use of different platforms has helped us to identify the kind of control functions that the operator should be presented with in order to obtain a better performance.

V. FUTURE WORK

Much work is still to be done in our system, which includes:

- Improving the estimation of the "expected configuration" by integrating the measures received from the robot through the use of filters.
- Improving the high-level control mode to augment the autonomy level of the robot, allowing it to overcome unexpected obstacles by itself whenever possible [11].

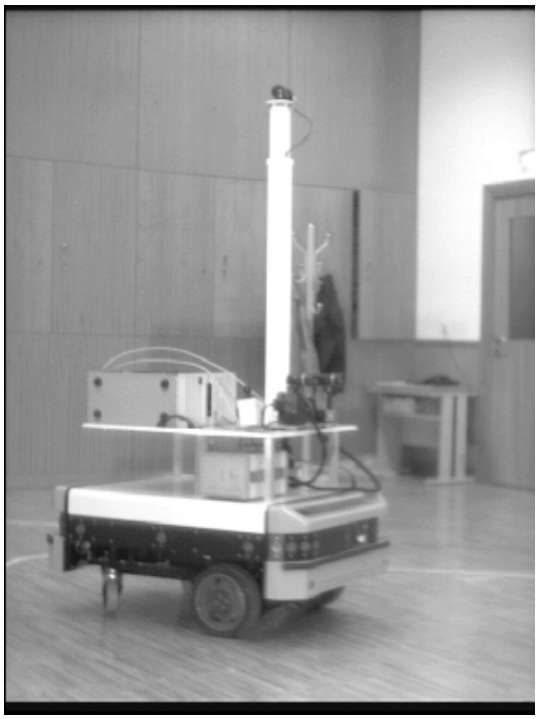

Fig. 5: Robuter platform

Fig. 6: Nomad platform

- Integrating a module to provide visual feedback, which is currently under development. In this module the video stream obtained from a robot mounted camera is subject to lossy compression before being transmitted.

References

[1] S. Figueiredo, N. Martins, J. Dias, A. de Almeida, "Surveillance System Using Artificial Vision", RECPAD'95 - Associação Portuguesa de Reconhecimento de Padrões, Universidade de Aveiro, Aveiro - Portugal, 23-24 March 1995.

[2] F. Almeida, R. Almeida, R. Carvalho, "Sistema de Televigilância", ISR Project Report, 1995

[3] Cortesão R., Araújo R., Nunes U., de Almeida A. T., "Buildin a Teleoperation Environment", IFAC, AATC'97, Vilamoura, Algarve, Portugal, April 1997.

[4] http://maas-neotek.arc.nasa.gov/Dante/dante.html

[5] http://mpfwww.jpl.nasa.gov/default.html

[6] Schulzrinne H., Casner L., Frederick R., Jacobson V., "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, GDM Fokus, Precept Software, Xerox Palo Alto Research Center, Lawrence Berkeley National Laboratory, January 1996.

[7] Antal K. Bejczy, Won S. Kim, Steven C. Venema. "The Phantom Robot: Predictive Displays for Teleoperation with Time Delay", Proceedings IEEE International Conference on Robotics and Automation,pages 546-551, Cincinnati, Ohio, May 1990.

[8] Polly K. Pook, Dana H. Ballard. "Remote Teleassistance". IEEE International Conference on Robotics and Automation,pages 944-949, 1995.

[9] Fogle R, F, "The Use Of Teleoperators In Hostile Environment Applications", *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 61-66, Nice, France May 12-14, 1992.

[10] Stone H. W., Edmonds G., " Hazbot: Hazardous Materials Emergency Responce Mobile Robot", *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 67-73, Nice, France May 12-14, 1992.

[11] Bhatia P., Uchiyama M., "Shared Intelligence for Telerobots with Time Delay: Theory and Human Interface with Local Intelligence", *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1441-1448, San Diego, California May 8-13, 1994