Robotica (1999) volume 17, pp. 49–57. Printed in the United Kingdom © 1999 Cambridge University Press

Web-based telerobotics

Ken Taylor, Barney Dalton and James Trevelyan

University of Western Australia, Nedlands 6009, W. Australia (Australia)
http://telerobot.mech.uwa.edu.au/

SUMMARY
A six axis robot has been connected to the World Wide Web at http://telerobot.mech.uwa.edu.au/. The robot was made available on the World Wide Web in September 1994 and has been through many revisions since that time. The remote user receives static images of the robot's work space and can manipulate wooden blocks.

The implementation and application of this interface to telerobotics is discussed. Useability and operator behaviour are investigated through analysis of movement records. Questions addressed are: What is the level of interest in this technique and is it a practical interface for telerobotics.

KEYWORDS: Telerobotics; World Wide Web; Useability; Operator behaviour.

1. A TELEOPERATED INDUSTRIAL ROBOT ON THE WORLD WIDE WEB

The "web" was designed as a hyper-text distributed information storage system for technical documentation. Web data is stored in "pages" which can contain plain text, formatted text, images, and "fields" which the reader can fill in to request further information. We use this facility known as "forms" to extend the concept to the control of a physical device. It has recently become possible with Java Applets to do some processing at the remote computer which we currently use to display a three dimensional model of the robot's current pose and which offers great possibilities for increasing the sophistication of the interface.

In September 1994, we connected an ASEA IRb-6 robot to the Internet through a Web server so that anyone with Web access could control the robot. This was replaced in August 1996 by an ABB IRB 1400 robot and the software rewritten to work with the new robot and to incorporate knowledge gained from the first version. It is now in use 71% of the available time by an average of 87 operators a day.

As far as we are aware, this is the first industrial robot allowing full 6 degree of freedom movement to be used in this way. Live Internet cameras are now commonplace and there are several teleoperable devices including a few industrial robots,[1] operable through web browsers.

The first robot on the web appeared 4 weeks prior to our robot. Ken Goldberg and Michael Mascha at the University of Southern California connected a two-link SCARA robot in a similar manner.[2] It could be positioned very simply with a single click on the workspace map or on an image taken by a camera mounted on the arm. The interface was specifically designed to have only two degrees of freedom to be readily understandable to inexperienced users.

1.1 Robot server

Our robot server consists of a personal computer running a Web server program under a Windows operating system (Figure 1).

Four monochrome CCD cameras are connected to a frame grabber, which collects images of the robot work space. A serial link provides access to the robot controller. When an operator using a Web client program such as Netscape requests information, the server sends a page containing four images of the robot (from different angles), an optional three dimensional model, explanatory text, and a number of information fields. When a movement request is received, the server interprets the data in the fields and sends an appropriate movement command to the robot controller. On completion of the move, the server collects new images from the cameras and sends a new page with these images so that the user can see the new state of the robot.

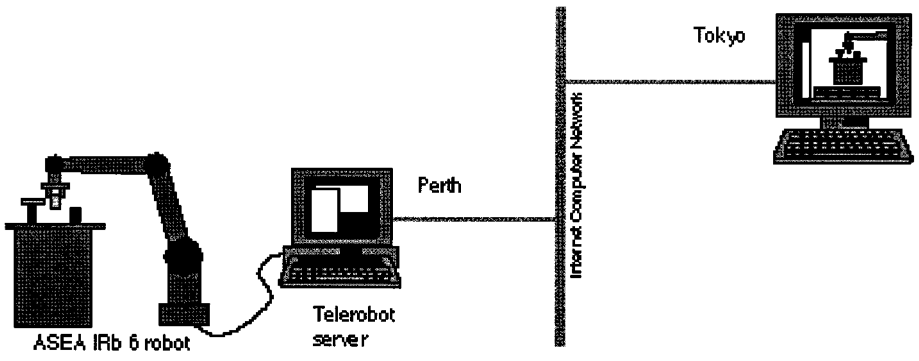

Fig. 1. The Telerobot system. A computer in Perth interfaces between the robot and a remote machine on the Internet.

1.2 Operator's controls

The robot control page is intended to provide the user with enough information about the robot workspace to decide on the next movement. We allow the user to control the image quality, specifying size and number of shades of grey or switching an image off. This provides a trade off between information content and downloading time. Two of the images are calibrated allowing control of magnification (zoom). These images automatically follow the robot gripper so that the tool centre point will be as close to the middle of the image as the camera's field of view will allow.

When moving to a requested pose the robot performs vertical movements and horizontal movements sequentially. If the final position is above the starting point, vertical movement will be made first. If it is below, the horizontal movement will be made first.

The user can choose absolute or relative coordinates, and can also request that the gripper be opened or closed. The display is intended to be self-explanatory although help pages are provided.

The work space table has a grid of thick black lines with 100 mm spacing so that users can judge horizontal coordinate values from the camera images.

1.3 Tasks performed

Some operators just explore the interface and try a few movements. The more creative operators can manipulate the blocks on the table to produce towers; see Figure 2 for an example. Those with a destructive bent push the blocks on to the floor, which unfortunately means they are not then available to future operators, until a volunteer passes the lab and replaces the blocks.

Initially, the task was complicated by conflicts between users. Many different operators could give different movement commands at the same time without being aware of each other. Thus, the environment could be changed before an operator had time to decide on his or her next move. An allocation system was devised by March 1995 to give a single user exclusive access. Other users (observers) could watch progress, but could not gain control until the current operator had been idle for three minutes.

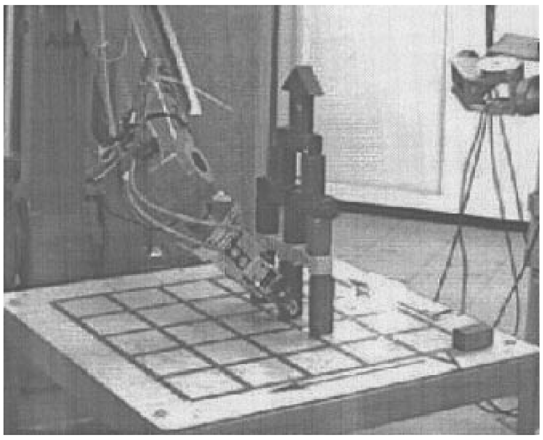
Fig. 2. Tower of blocks created by John Willoughby, construction time was five hours.

Interestingly Goldberg's first robot had an allocation system but their current Telegarden[1] project allows access by multiple users to increase the system's availability. They avoid confusion by allowing users to alter the environment in such a way that it does not affect tasks performed by other users.

2. IMPLEMENTATION

This project started with a question: "Why are there so few robots?" In the late 70's and early 80's many people forecast a boom in robotics with exponential growth in sales. This has not happened. Instead of the 10's of millions of robots forecast, the world robot population is still less than 1,000,000 robots, according to 1995 IFR statistics. It is well known that most robots work in the automotive industry.

2.1 An ambitious idea

We started to think about the obstacles which robotics must overcome before the technology can be widely applied. Several themes emerged:

- Robots are expensive machines, typically costing between US$30,000 and $80,000.
- Most robots are complex to install and operate. Intensive training is essential and connection to external devices is a technically involved task requiring trained installation engineers.
- Most robots are still programmed by leading them through the required motion with a teach pendant. This is time consuming and accuracy is limited. Off-line programming requires skills, which are not common among users, who are normally trained plant technicians. It is not easy to apply in practice because most robots have large absolute positioning errors despite excellent repeatability. It is notable that much of robotics research assumes the use of off-line programming which is still seldom applied in industry 20 years after it became technically feasible.
- Even if off-line programming is mastered, it is extremely difficult to program robots to perform tasks which humans find trivial such as recognising and selecting objects.
- Robots could be programmed for a wider range of tasks using computing skills which many people now have (point and click + limited keyboard use), removing another obstacle.
- Robotics research has produced a mountain of complexity, with thousands of research papers published each year. Simple approaches help people to understand the technical issues, and are more likely to be adopted.

Robotics applications have traditionally required fully autonomous control which has necessitated solving all of the problems required to achieve automatic control before a task can be accomplished. This can be the downfall, as problems of perception and control are frequently overwhelming. A supervisory control regime offers the prospect of solving control problems incrementally. It falls between the extremes of manual and automatic control. Supervisory control is defined in Sheridan,[3] "in the strictest sense, supervisory control means that one or more human

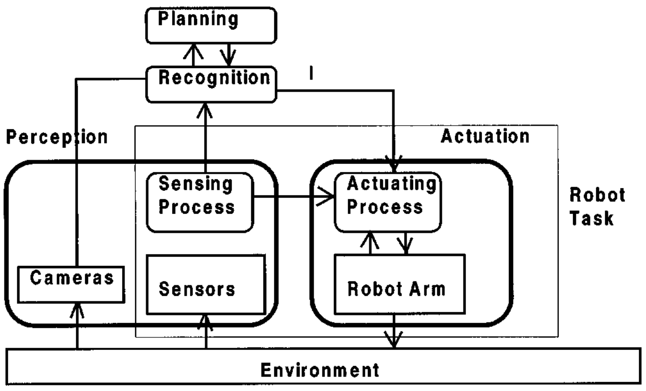

Fig. 3. Outline of the supervisory control scheme.

operators are intermittently programming and continually receiving information from a computer that itself closes an autonomous control loop through artificial effectors to the controlled process or task environment". It has found negligible application in industrial robotics and is discussed by Sheridan[4] under the heading "The Distinction Between Teleoperators and Industrial Robots". He states "There has been surprisingly little interest in human supervisory control for industrial robots and the availability of a nearby human operator has been taken for granted".

While we were researching these issues, a startling idea occurred to us: to control a robot using the Web and open our laboratory to remote population of robot users. In doing so, we could explore an entirely different approach to robot programming using telerobotics and supervisory control, implementing the regime in Figure 3.

It was an ambitious objective. At the time there were no physical devices controlled from the Web, yet we had a good robot that was utilised for perhaps 1% of the time at most, and hence potentially available to other people for the other 99%.

An important stimulus was the famous Cambridge Coffee Pot[5] which could be observed remotely via the Web. We had the necessary hardware to "publish" images of our robot on the Web. Yet linking our DOS-based image collection software and robot control software with the Web seemed a daunting task, given that we had no understanding of Web software and the TCP/IP protocol by which signals are transmitted along the Internet.

We wanted to use components as generally available and simple as possible and consequently decided to use a single PC for image capture, as a robot controller and as a web server.

2.2 *Software*

Much of the software we have adapted to this project has been found to contain errors that would occasionally cause problems. All of these, which had been tolerable for other applications have had to be removed to achieve a reliable system.

Web data is normally static. Therefore to save data transmission and time, Web browser programs store each Web page retrieved by a user on the user's computer. Thus, if the user wants to return to that page, it is now stored locally and does not need to be requested again from the server. An image used for robot control or for monitoring a changing scene is dynamic. A fresh version needs to be retrieved from the server each time it is accessed. Therefore we include a random number in the name of each image generated to force the browser program into bringing a new image each time.

The server software we selected was Robert Denny's[6] Win-httpd. On request from a browser a WWW server launches a CGI script which can be a batch file or executable which must output an HTML formatted document or point to an existing document on disk. The POST method is used to launch a windows C++ CGI script. CGI is the protocol for passing data between the WWW server and the application. A server is uninterested in what the application does, only looking for an HTML formatted document as output. In the first version of our software, the WWW server launched a DOS batch file that started two DOS executables: the first to control the robot, and the second to take the pictures. Data was passed between applications by writing to disk from the first and reading back the data from the second. The batch file launched one of two alternative applications for generating the HTML document. This decision depending on whether the request had come from a person classified as an operator or observer, being made by the executable controlling the robot and passed to the batch file via the exit code. The drawbacks in this approach were several. Applications running in DOS under Windows 3.11 receive a fixed proportion of the processor time regardless of their need to run more slowly than they otherwise would. Each time a request is made the full executable has to be loaded from disk. As the system became more sophisticated, the amount of data that had to be passed between executables became unwieldy and some of the functionality had to be repeated in both executables. With the new robot, three cooperating Windows applications are used to achieve the desired result. A core CGI script communicates with an image server that generates the images, and a robot server that communicates with the robot. The robot server and image server run continuously and respond to messages from the controlling CGI script. This greatly reduces the overhead for running the script as robot and image initialisation is done at boot time only.

The DOS version of the software was adapted from software used for various other purposes, a large proportion of which was then used for the Windows version. The applications on which it was based worked well but surprisingly several problems soon became apparent when they were combined. First, the robot control software needed several improvements to avoid an indefinite "hangup" in the serial communication link with the robot. The communications software had used polling for serial communications. This was modified to be interrupt driven as accurate timing was not possible with the multitasking of the Windows operating system. Secondly, we had to extend our partial implementation of the robot's proprietary communication protocol. Lastly, we discovered a number of mistakes in our kinematics calculations as remote users attempted more ambitious tasks.

This reveals a major problem with tolerobotic applications and that is the difficulty of making them reliable. This has been a problem for several telerobotic installations that have appeared briefly on the Web and is considered important by Goldberg (private correspondence dated 12 Nov., 1996) who believes "design of the interface for reliability is central to WWW Telerobotics".

Due to our current hardware set-up we are still running under Windows 3.11, and consequently have to deal with the occasional system crash. Running a short program to toggle two relays (from the array originally installed to switch alternative cameras to the frame grabber) every thirty seconds solved this. A PLC monitors this array and re-boots the computer if the relays stop oscillating, commonly known as a watchdog timer.

The web server launches the main controller program with a file containing all the user details. Using a hidden field in the form the submission is examined to see if it is from an operator, an observer, or someone who can now become the operator. In the case of an observer, the response is simple: the most recent pictures along with the current operator details are written to an HTML file. If the user is a new operator then new pictures are taken and the new controller's details recorded. If the user is the controller then the robot is first moved to the required location. This required location may be specified by values within the form, or from clicked image point. If it is the latter then a stereo point must be calculated from the two image points. Once the robot has finished moving, pictures are taken centered on the tool end point and an operator page returned.

The robot server program receives requests in the form of Cartesian coordinates, spin and tilt. The spin and tilt are converted to a quaternion representation and are passed to the robot controller over the network. The program on the robot controller is then initialised with this new target pose. The robot server program waits for the robot to finish and indicates to the calling process whether the move was successful. Similarly the image server receives requests for images of a certain size and sub-rectangle of the image plane; the image is grabbed and saved in GIF format. Figure 4 shows a schematic diagram of the hardware and software components.

2.3 *Robot*

The ABB robot consists of a 6-axis manipulator with a controller which can accept remote commands through a serial communication link (RS232, 9600 baud) using the SLIP protocol. A proprietary windows DLL allows high level communication with the robot. Among the available commands are:

(i) Read current status and robot position in Cartesian workspace coordinates
(ii) Control robot operating mode (standby – operate)
(iii) Start/stop programs stored in the controller memory.

To make a move, a target position variable is changed in the controller, a program is then invoked with this target position. The controller program first checks to see if the gripper is changing state, and if so changes a digital I/O line connected to the pneumatic valves of the gripper system. The robot move is then split into two parts. A vertical move with no reorientation, and a horizontal move with reorientation. If the required position is above the current one, then the vertical move is performed first; otherwise, the horizontal move is executed first. This helps to reduce collisions with other blocks, and reduces the number of moves an operator has to specify to achieve a given objective – a fundamental goal of this research.

Our robot control software incorporates automatic compensation for errors in the kinematic model of the manipulator. Model data is read from a data file. The program calculates the joint angles, which are required to attain the specified pose, using the calibrated kinematic model. Then it calculates a Cartesian position and quaternion which, when sent to the robot's controller, will result in those joint angles being attained, and hence the required pose.

The program restricts robot movements to the space above the table with the blocks placed on it. Movements below the table surface are not allowed. The orientation of

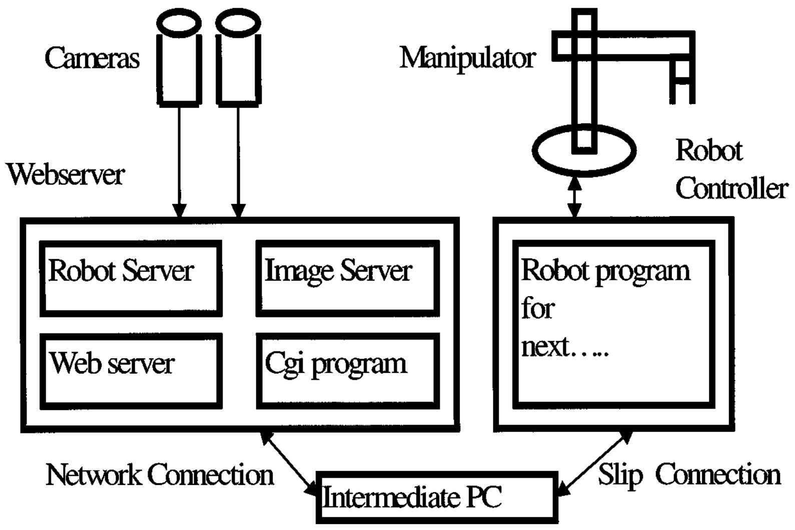

Fig. 4. Details of the Telerobot system. For each web request a CGI script is launched which then communicates with the robot and camera servers.

Table I. Operator preference in move requests.

Move	Number	Percentage
Relative	3217	11
Absolute	24711	83
Clicking Image	1874	6
Total	29802	100

the tool is also restricted within an inverted cone with a 45 degree base angle.

2.4 Operator interface

The design and layout of the interface has been revised many times in response to observation of the difficulties users are having.

As an example, the robot could initially be positioned with only relative coordinates. After introducing the choice of relative or absolute coordinates, it immediately became apparent that most users preferred specifying absolute movements so that was made the default. This preference can be seen in Table I, which shows operators' movement requests during the period when three methods of moving the robot were available and the default was move absolute.

During May 95 an image was added showing six wire frame models of the robot from different viewing angles. This contained a lot of information though the images were only 4.5 kilobytes in size. A click in the wire frame image would cause the robot to move to the position clicked. A single click will provide only two parameters where three are required to define a position in space and three more to additionally specify the orientation. Therefore, orientation and the third component were maintained. For example x and y may be altered but z will remain constant. Surprisingly this was not a popular method of controlling the robot as Table I shows, only 6% of movement requests were made this way. It was so disliked that in 42% of operator requests the option to switch off this image was selected (Table II).

This feature was not included when the software was rewritten. The facility is now provided by a Java applet which may be rotated to any viewing angle and which requires only the joint angles to be sent across the network after each move.

We also added software to allow users to leave comments, a couple being:

Man this thing is incredible!!! After 5 minutes I felt trembles 10 minutes later perspiration dripped from me oooops . . . 3 hours later and I've got no chance of finishing that assignment!!!!! But the funny thing is I don't care, moving these bricks, dropping them on the floor, picking them up, wooooooooooo I feel like a two year old all over again!!! GREAT STUFF! – Randall Fletcher (Fletch) 20 March 1995

Crude but has vast possibilities. – Chris Layne bclayne@erols.com 28 August 1996

This was really interesting. I'm going to let my kids try it next time! Much better than watching t.v. for them. Thank You! – Pam woods@galstar.com 30 November 1996

Table II.

	Primary Image	Secondary Image	Wire Frame
Primary Image	0.2	0.1	0.5
Secondary Image		1	3
Wire Frame			39

In December '94 we allowed the use of Euler angles (roll, pitch and yaw) for specifying wrist orientations. This proved to be very confusing, users had little experience of these terms and found them hard to visualise. In September 1996, we changed this to spin and tilt (Figure 5). Spin is defined as rotation about the Z-Axis of the table and Tilt the angle that the gripper makes with the table's Z-Axis. Spin is used to get the jaws of the gripper parallel with the length of a block.

This is much simpler to understand and constrains the orientation of the gripper jaws to two degrees of freedom. The effect of this is to ensure that a line drawn through the gripper end points is always in the same plane as the surface of the table, which is ideal for block manipulation. This aspect of the interface no longer draws criticism.

2.5 Imaging

Initially there was a single camera to provide feedback and users had great difficulties with perceiving depth. This was improved by arranging lighting to throw shadows from the blocks on the grid. A second camera was added in January '95 mounted orthogonally to the first and this provided a much better view of the workspace. However, it requires some thought to interpret a scene viewed this way. Currently four cameras are positioned as shown in Figure 6 and we are investigating user preferences.

A camera mounted on the sixth axis has also been tried, but was removed due to the highly restrictive effect on the robot's workspace. Placing cameras on the robot is a trade

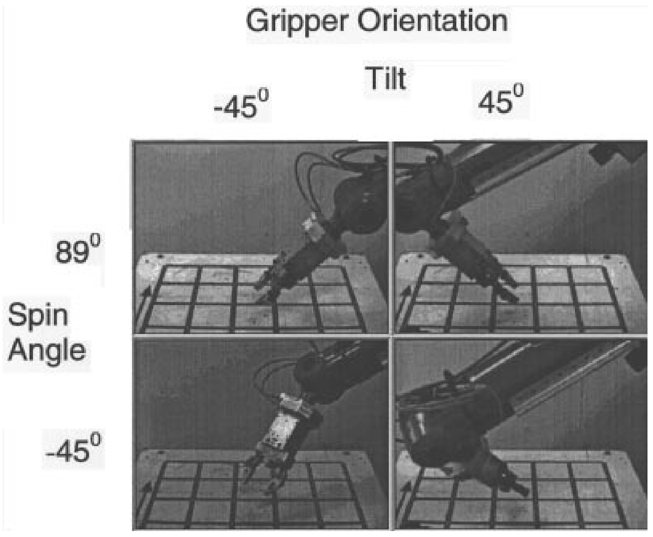

Fig. 5. Images showing gripper orientations for different values of Spin and Tilt. Note that the jaws of the gripper are always parallel to the plane of the table.

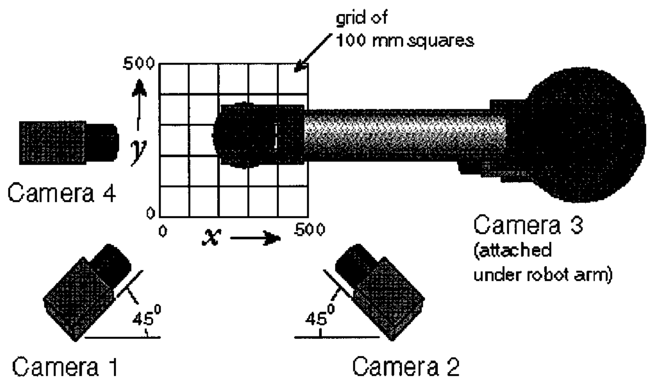

Fig. 6. Camera and table layout.

off between being close to the end effector and restricting the workspace. The correct placement of cameras has proved to be a contentious issue, most camera positions seem to elicit both praise and criticism from users.

The two fixed cameras are calibrated with respect to the robot's workspace so that image coordinates can be related to robot coordinates. This enables some intelligent processing on the images, such as; knowing the robot tool position a software pan-tilt and zoom can be performed to give the user a close up view of the end effector. This is of particular importance as it is essential to keep the image size as small as possible while still supplying the information the user needs. Calibration of the camera also enables 3-D points to be specified by the user clicking a point in each of the images, allowing point and click movement of the robot.

The cameras feed images to a Data Translation Quickcapture frame grabber. The video signal from different cameras is switched to the frame grabber with reed relays controlled from a PC brand, model PC-36 digital I/O board. The grabber grabs an image of 768×512 pixels, depending on user options some sub rectable (software pan and tilt see above) of this is extracted and resized before being saved in GIF format. For the uncalibrated images, a central 512×512 rectangle is used. We have used several versions of GIF coding software and currently use a version by Huseby.[7]

4. ROBOT USAGE

Data collected on robot usage allows us to determine how the robot is being used. It has driven the interface development as it can be quickly seen whether the robot is being used as anticipated and what troubles users are having. It offers considerable advantages over most telerobotic studies, which are based on very small data sets. However there is no control over the tasks users perform hence it is difficult to study task competencies.

4.1 Data acquisition

All visitors since 21 October 1994 to the robot are recorded. Currently records are lost only rarely but earlier records are much less complete, with perhaps 50% lost in the early months. The details recorded are:

- Internet Address of Visitor.
- Date and Time of request.
- Item requested.

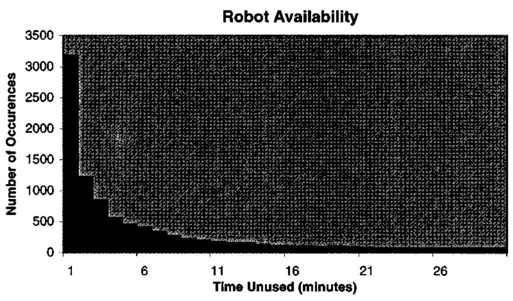

Fig. 7. Graph showing usage of the robot, idle time is usually less than 20 minutes.

- Commands to robot.
- Image Specifications.

Initially the web server logs were relied on to record this information but for increased flexibility and ease of data analysis logging is now done within the telerobot application. Visitors are identified by Internet address. Some Internet addresses are allocated dynamically so the same person visiting on separate occasions can show up as more than one person.

4.2 Numbers of users

In the 95 days between 11 September 1996 and 15 December 1996, 8260 people operated the robot. This excludes those who could not gain control because others were using it. It was in use 71% of the available time and the periods for which it remained unused rarely exceeded 10 minutes (Figure 7).

4.3 Requests in a session

Defining a session to be at least two requests to the robot from a single site less than 4 minutes apart; a comparison of 34,000 sessions for the first telerobot and 5,650 for the new telerobot, Figure 8 reveals a greater interest by operators in the newer telerobot. This excludes operators who gained control of the robot but did not make any further requests. To complete a simple block stacking task requires at least 10 requests to the robot. Previously only 2.5% of operators made more than 10 requests to the robot in a session and this has now increased 10 times to 24% of all operators.

4.4 Response times

The operator controls the robot by providing a goal and receiving a response. The robot receives a goal, attempts to achieve the goal and reports back to the operator on the result. This methodology is necessary to conform to the client server model of the World Wide Web but is also consistent with the supervisory control model. It eliminates control instabilities, which would otherwise be associated with long and variable delays in the control loop, by performing all the time critical control locally. Regardless of the sophistication of the control process the test task of stacking blocks can never be performed by the robot at the same rate as a human due to it's limited dexterity. Within this constraint the speed at which task can be performed depends on the:

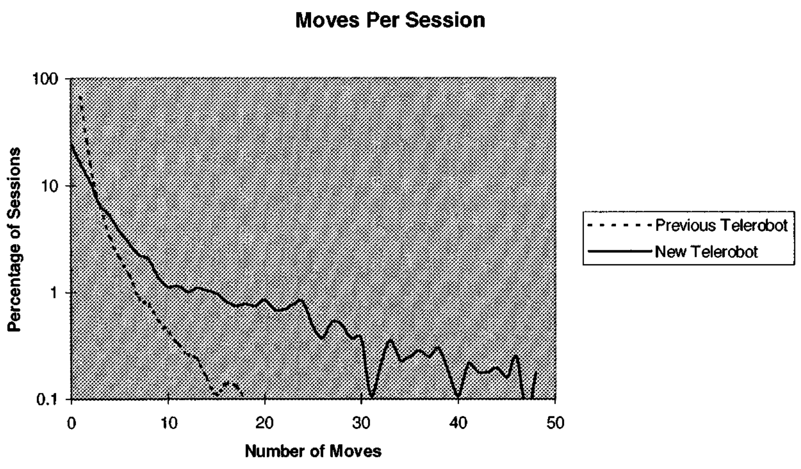

Fig. 8. Graph showing number of moves made in a session for different users.

- Sophistication of the goal that is provided to the robot.
- Local processing time.
- Movement time.
- Transmission time.
- Operator decision making time.

As sophistication of the goal given to the robot is increased, the number of actions performed between each feedback step increases, eliminating two components; feedback time and operator decision making time, leading to a more efficient system.

Based on 53,000 requests to the old telerobot and 58,000 requests to the new telerobot the distribution of intervals between requests is shown in Figure 9. Across a local ethernet connection, the minimum time per robot request was measured at 17 seconds with the old telerobot and at 13 seconds with the new telerobot. This would account for four of the 7 seconds in the difference in median time between moves.

Robot movement speed is limited to a slow 100 millimetres a second. This is to limit the inertia of the arm and eliminate the risk of damage when the gripper is brought down on a solid object. This does mean movement time can be up to 6 seconds, for a movement from one side of the table to the other.

4.5 Imaging preferences

To modify image properties the old telerobot required checking a button on the robot control page labelled “Change Image”. After submission, the page returned allows the user to select three aspects of the image quality; size, resolution and number of shades of grey. The current interface layout includes the image controls in a drop down combo box at the base of the main controller page.

The defaults were; 16 shades of grey, 256×256 pixels, full resolution. The resolution control is no longer offered. Image properties can now be controlled for each image and the defaults are currently 100 by 100 pixels and 16 shades of grey. A sample of 30,800 requests revealed 94% accepted all of the image defaults for the old telerobot. However, looking at the greyscale selections in Figure 10 based on a sample size of 267,000 and 30,800 for the old telerobot it can be seen there is currently a far greater enthusiasm for alternative settings.

4.6 Useability and operator behaviour

The popularity of the telerobot, shown by the 71% usage statistic, and enthusiastic operator comments provides conclusive evidence of the high level of interest in World Wide Web telerobotics. However, useability is more prob-

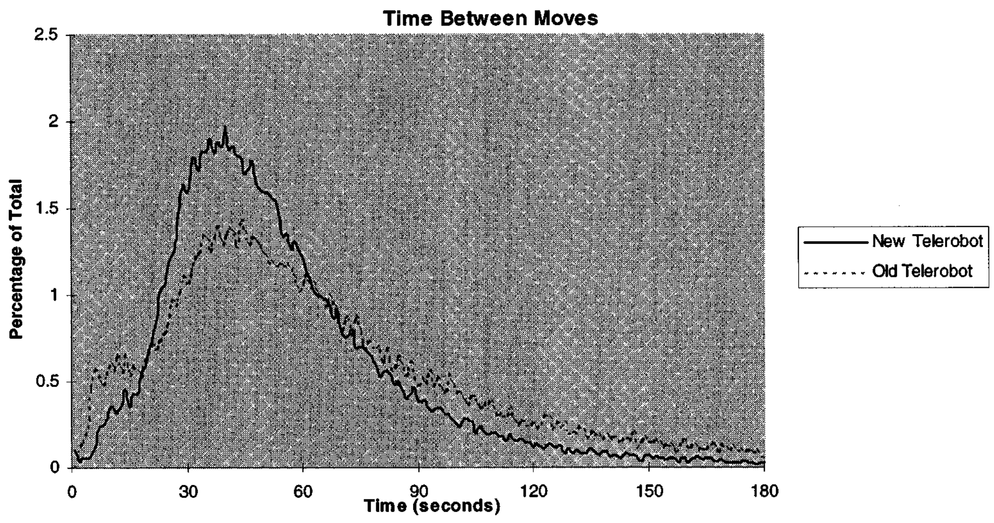

Fig. 9. Graph of time betwen robot move requests for both the old and new telerobot.

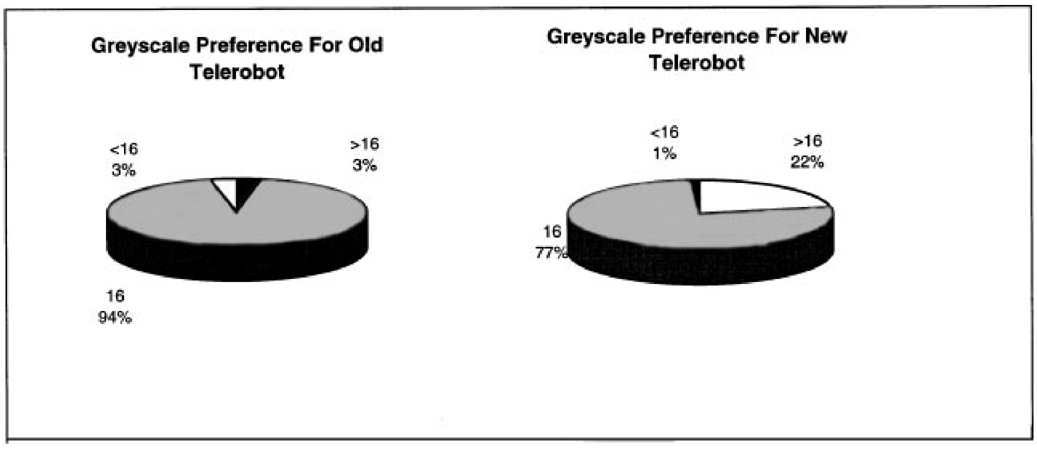

Fig. 10. Percentage of users changing greyscale preferences for the old and new systems.

lematic and difficult to quantify. Many of the statistics presented indicate the importance of the interface design, the importance of providing the correct feedback, the strong influence that interface design has on operator behaviour and the difficulty in predicting the effect of any particular change on that behaviour.

The new telerobot performs the same tasks as the previous one. From an operator's perspective the differences are that of response time (up to 4 seconds faster) and that the interface has changed in a number of ways as previously described. These changes have led to a dramatic increase in the number of moves per session, which appears to be mainly due to the operator's preference for the current interface layout and feedback supplied. The influence of interface layout on operator behaviour can be seen in the willingness to vary the greyscale settings which has changed from 6% of all requests to 24% of all requests. While the developments in the interface design have been overwhelmingly positive it is very much a trial and error process that has seen some failures. The introduction of the wire frame model and the ability to move the robot by clicking on it was an attempt to provide a more visual interface. Contrary to expectations it did not appeal to operators (nearly a third of all operators switched the feature off entirely).

5. APPLICATIONS

Telerobotics has, until now, been an expensive technology. Therefore, it has only been used where environmental conditions make human operations extremely hazardous, such as nuclear reactors, underwater, and space applications. Normal economic consideration seldom apply in these instances.

As far as we are aware, "remote" telerobotic experiments have used high bandwidth communication links, typically with real-time video, and high bandwidth control and sensing links. In early 1993, the space robot technology experiment ROTEX flew on the space shuttle Columbia. The project is described in a paper by Hirzinger *et al.*[8] which also mentions a number of similar projects. Those being; a space station mobile servicing centre being built by the Canadian space agency, NASA's Flight Telerobotic Servicer project (significantly cut back due to cost overruns), Japan's space station Remote Manipulator System (JEM-RMS) and the Japanese ETS-VII project which is another experimental space telerobotic servicer.

Our research has demonstrated that telerobotics is feasible with low and variable communication bandwidth, using cheap additions to a rapidly expanding network infrastructure. The difference being that this relies on supervisory control concepts, as opposed to continuous feedback. The robot must have a local control loop with some degree of autonomy, which executes user commands. The degree of local control can depend on the application, and there is no reason why many of the techniques developed for fully automatic robots could not be used with limited autonomy in this context. For example, direct vision feedback from a gripper-mounted camera could be used to guide the robot in the final phase of grasping an object, without the need for user intervention.

Having accepted certain bandwidth restrictions, in order to reduce the cost and allow the user to be further removed from the robot, we can propose new classes of applications for telerobotics where the motivation is economic or even entertainment.

5.1 *Robotics training*

Many research and educational institutions cannot afford to purchase industrial robots, partly because they are idle for much of the time. Therefore, they rely increasingly on simulation packages. When we demonstrated this technology to a class of students in Toronto, several of the students thought that the image they saw was virtual reality and not the real robot! This technique may provide more effective access to a real robot for students from remote sites.

5.2 *Space*

The ROTEX experiment demonstrated the potential utility of robots in space for construction and repair operations but at considerable cost, particularly in communication bandwidth and ground support facilities. Space facilities may one day require numerous robots performing construction or repair tasks. This technology shows how such robots could be controlled using modest ground and communication facilities.

Rather than a single robot being controlled from a lavishly equipped ground station, and occupying all the available communication bandwidth for stereovision feedback, we propose a distributed network of operator terminals located at the contractors responsible for the many different components of a space project. Each terminal can be configured to control one or more robots operating at slow speed requiring, perhaps, only intermittent monitoring. This may be a more economic alternative for many projects.

5.3 Entertainment

We have no doubt that the major incentive for most of our users has been curiosity and entertainment. The first robot demonstrated on the Web, just a few days before our own, was based entirely on entertainment. Using a model robot, users were invited to "blast" foam balls with a compressed air jet to discover clues and solve a mystery.

Therefore, entertainment providers may find this a profitable alternative technology. Imagine, for example, exploring part of a mid-ocean trench, all from the comfort of your home as a diversion on a cold winter evening. Exploration of a remote environment is the goal of Mechanical Gaze[9] which allows users to interact with museum exhibits. There is currently a telescope available on the Web – another entertainment possibility.

5.4 Remote manufacturing

Remote manufacturing is an exciting possibility that has been recently taken up by a team from University of California, Berkeley who refer to their project as Cybercut.[10] A designer could prepare a design on a computer and without leaving the screen control the equipment to manufacture it. The product is then couriered to the customer. The technique is suitable for sharing machinery where the cost of ownership may make it's use impractical for a single person or organisation but where the cost is not prohibitive when spread across a large user base.

6. CONCLUSION

We consider the demonstration of the concept a great success and continue to refine and explore the limits of it's capabilities. These are rapidly expanding as the facilities afforded by the World Wide Web expand. The rate at which they occur is astounding. Two years ago browsers were quite crude, they are now sophisticated and allow manipulation of three dimensional images and support local processing through Java applets. The use of local processing provides opportunities for more sophisticated interfaces based on simulation which we are exploring. While we do not offer the capabilities of the most expensive telerobotic installations associated with space applications, we have achieved a lot of the functionality with a fraction of the resources. Remotely operated robots may one day outnumber the more conventional robots we see today, because we have shown that our existing communication networks can make a real robot accessible to a large number of users.

The interest generated in both our installation and Goldberg's[2] has been remarkable. There is no doubt that curiosity and entertainment has provided a strong incentive for tens of thousands of users to access our system. This suggests that robotics could become a popular entertainment medium in the future. As well, the possibility of making expensive machine tools available to artisans at low cost presents considerable opportunities.

Most, if not all the technology is available to pursue these opportunities now.

Acknowledgements

Thanks are due to Peter Murphy for adapting our image collection software to generate GIF images used by the Web. Hing Wong wrote many elements of the robot control software. Troy Phillips made many vital improvements to our robot control software. He made use of interrupts so that serial communication routines would work in the context of the Windows operating system. He also improved the accuracy and robustness of transformations between homogeneous coordinate frames and the quaternion scheme used by the ASEA robot controller. Bradley Saracik implemented the allocation scheme to guarantee a single user control without requiring a user to "sign off" before others could have a turn. Shalini Cooray and Dan Macey had the misfortune to be thrown into windows programming writing the first port to Windows 3.11.

References

1. Internet robots can be seen at http://www.usc.edu/dept/garden/ and http://vive.cs.berkeley.edu/capek/
2. K. Goldberg, M. Mascha, S. Genter, N. Rothenberg, C. Sutter and J. Wiegley, "Desktop Teleoperation via the World Wide Web" *Proc. IEEE International Conference on Robotics and Automation*, Nagoya, JAPAN (May 19–26, 1995) pp. 654–659.
3. T.B. Sheridan, "Human Supervisory Control of Robot Systems" Proc. IEEE Conference, International Conference of Robotics Automation, San Francisco (Apr. 7–10, 1986) pp. 808–812.
4. T.B. Sheridan, *Telerobotics, Automation and Human Supervisory Control* (MIT Press, Cambridge, Mass., 1992).
5. The Cambridge Coffee Pot located at web address: http://www.cl.cam.ac.uk/
6. R. Denny, (1995). Robert Denny's Win-httpd World Wide Web Server at web address: http://www.city.net/win-httpd/
7. Sverre H. Huseby, "C" source code obtained from Internet. sverrehu@ifi.uio.no, Bjoelsengt. 17, N-0468 Oslo, Norway.
8. G. Hirzinger, B. Brunner, J. Dietrich and J. Heindl, "Sensor based space robotics- ROTEX and its telerobotic features" *IEEE Transactions on Robotics and Automation* **9**, No. 5, 649–663 (1993).
9. Eric Paulos and John Canny "A World Wide Web Telerobotic Remote Environment Browser" *Fourth International World Wide Web Conference* Boston, Massachusetts, USA (Dec. 11–14, 1995): Mechanical Gaze - http://vive.cs.berkeley.edu/capek/
10. Cybercut – Remote Manufacturing at http://kingkong.me.berkeley.edu/cybercut/