TELEROBOTICS EXPERIMENTS VIA INTERNET

Riko Šafarič, Matjaž Debevc, * Rob M. Parkin

Faculty of Electrical Engineering and Computer Science, University of Maribor,
Smetanova 17, 2000 Maribor, Slovenia,
Fax: ++386 (0)62 211 178, E-mail: riko.safaric@uni-mb.si

* Mechatronics Research Group, Department of Mechanical Engineering,
Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK,
Fax: +44 (0)1509 223934, E-mail: r.m.parkin@Lboro.ac.uk

Abstract

This paper describes design issues involved in providing remote users with internet access to laboratory or production line based hardware. Simulation tools for the robotic hardware were developed using C++ and VRML 97 to create a desktop virtual reality environment which improves the visualisation of the manipulator hardware and associated workspace. Communication between the remote user and project server via the internet, interface electronics and control software is also discussed. Off-line programming allows user to create and test user programmes inside a virtual robot cell. A collision detection software added to the robot virtual world enables production engineers to ensure that a robot can reach all the necessary points in a real world. Using of virtual reality simulator allows to reduce time for which the robot cell is out of production. Users using networked computers can access the on-line robot cell to perform a series of interactive experiments with real-world hardware including a real-time video supported Web controlled magnetic suspension system and a six degree-of-freedom MA2000 robot.

Keywords: robot, tele-operation, internet, collision detection, VRML 97
I. INTRODUCTION

The virtual reality robot cell simulator is suitable for companies that have large robotized production lines and a regular need to change robot programmes with minimal loss of downtime. It could be used also for engineers’ training in order to acquire valuable “hands on” experience. Engineers can use an animation of robot cell activity to conduct feasibility studies on their manufacturing processes while their product is still being designed. The experiences of the companies with robotized production lines show that they may save up to 85 % of downtime when the robot has to be reprogrammed and up to 50 % of programming time [18].

The other two experiments: Control of a DC-motor via internet [Rob- add the literature pointer] and the real-time video supported Web controlled magnetic suspension system [15] were part of the projects were we learned the tele-operating technology and are basis for the tele-robot application shown later in the paper.

One of the first successful World Wide Web (WWW) based robotic projects was the Mercury project [5]. This later evolved in the Telegarden project [9], which used a similar system of a SCARA manipulator to uncover objects buried within a defined workspace. Users were able to control the position of the robot arm and view the scene as a series of periodically updated static images. The university of Western Australia's Telerobot experiment [10] provides internet control of an industrial ASEA IRB-6 robot arm through the WWW. Users are required to manipulate and stack wooden blocks and, like the Mercury and Telegarden projects, the view of the work-cell is limited to a sequence of static images captured by cameras located around the workspace. On-line access to mobile robotics and active vision hardware has also been made available in the form of the Netrolab project [8], [14].

Problem with the static picture can be avoided by using video technology, which is becoming more and more popular in the Internet domain. Video is one of the most expressive
multimedia applications that provides a natural way of information presentation and has stronger impact than static text and figures [17]. The transfer of video is currently the hardest task to be performed while transmitting data over the Internet. Because transfer of fluent video demands high bandwidth capacity, different methods of video transmission are used. Lately streaming video [2] applications such as Real Video, QuickTime Movie and Microsoft Media Player, are commonly used in Internet. This application provides the ability of transmitting video recording over connections with small bandwidths such as telephone connections with V.90 standard. This kind of video transmission has been used for a more or less real-time video supported Web controlled magnetic suspension system described in section 2.

The previously mentioned projects [5], [9], [10] rely on cameras to locate and distribute the robot position and current environment to the user via the WWW. It is clear that such an approach needs a high speed network to achieve on-line control of the robot arm. Data transmission times across the world wide web depend heavily on the transient loading of the network, making direct tele-operation (the use of cameras to obtain robot arm position feedback) unsuitable for time critical interactions. Rather than allowing the users to interact with the laboratory resources directly, as in many of the previous examples, the reported approach in section 3 requires users to configure the experiments using a simulated representation (a virtual robot arm and its environment) of the real-world apparatus. This configuration data is then downloaded to the real work-cell, for verification and execution on the real device, before returning the results to the user once the experiment is complete. A virtual robot arm and environment model is used, instead of cameras, to minimise the data transmission time through the network so network speed is no longer a critical issue.
2 REAL-TIME VIDEO SUPPORTED WEB CONTROLLED MAGNETIC SUSPENSION SYSTEM

An introductory control experiment, including a real-time video supported Web controlled magnetic suspension system, was provided for basic instruction. Because of the increased usage of video technology in technical engineers' training process and to provide a better feeling to the users what is happening to the model of the experiment, which they can control, we have designed a prototype model of a real-time video supported web control. It represents web based supervision and handling of the industrial process in real time. The whole activity is viewed and can be manipulated by the user via World Wide Web and real-time video camera. The prototype presents a practical realisation of an exercise via World Wide Web for the area of on-line technical sciences. The whole system of distance experiment of a real-time magnetic suspension system is based upon the following components (Figure 1):

- Personal Computer:
  - Server (consists of a WWW server and RealVideo Server) and
  - Lab Computer Controlling Experiment with PHP support (HTML-embedded scripting language) and a system for real time control of the magnetic suspension system

- Experiment - Magnetic Suspension System [15]

- Lab Computer Controlling Video (consists of a RealVideo Encoder)

Figure 1 represents components involved in a remote laboratory connection together with a real-time video controlling. First part, which displays control of the experiment with a computer for control and a web server, is already used in different cases [16]. In most cases the data is recorded during the execution and shown to the student at the end of the experiment. To improve the classical model, shown in lower part of Figure 1, a prototype has been developed, which enables real-time observation of the experiment with help of a video camera.
Magnetic Suspension System [14] has been controlled on the personal computer (PC) extended by the controller board (A/D and D/A channels) used to measure input of the control system, namely: ball position, which is also the output of the optical position measurement system. The real-time sampling frequency of 1 kHz was attained.

A process is constantly running for control purposes on the server of the real-time magnetic suspension system. The process's ability is to receive and send signals from and to WWW server and scripts and to control the real-time system according to the input got from the user over the WWW interface.

The main problem here was to provide a real-time environment under Microsoft Windows NT. The system itself has no such capability. Provided solution requires a strong CPU on a system, which is totally devoted to the control system. This way we can assume a sampling rate of up to 10ms, which is in most cases just enough to sustain the control system in its position.

The whole system can be divided into:

1. A running process which controls the magnetic suspension system and accepts external input data (manipulation of the position of the ball)
2. A web interface allowing an remote user to change basic parameters of the control system
3. CGI scripts, which send the change of parameters to the controlling process and activate the results (graphical representation of the response of the magnetic suspension system to the changes given by the remote user)

The second system is completely separated and independent - Lab Computer Controlling Video. Its only purpose is to provide visual representation of the state of the magnetic suspension system with capturing the state with a video camera with usage of a video grabber card and then encoding the video data into appropriate form suitable for video transmitting.
Real Video server used on the computer enables the transfer of video recording to a distant user.

The only problem here was the time delay which occurs because of the time, the server needs to encode the data and the time needed on the client side for decode and buffering sufficient data for reproduction. To synchronise the procedure, a delay was added to the CGI scripts to provide a pseudo real-time representation of the whole process (that the result is consistent with the time base of the video representation). The delay is about 10 seconds in average and is compensated with a delay in the script.

In this electrical engineering example the users needs to choose the correct controller and its parameters of the control system holding the metal ball in a magnetic field. The main part of the window is dedicated to the live video picture of the experimental setting. Here the user can observe a large device in the Electro-magnetic laboratory (Figure 2).

After the student insert parameters and start the procedure he receives an on-line response of the magnetic ball model as a plot on graph in the bottom part of the window. At the same time the user can see the response of the system with help of an on-line video camera, which is positioned onto the ball.

User can watch the experiment or laboratory model, which is located at the lab with a help of a live video. User can use the web interface to choose and set system parameters on-line, and the response is sent back to him as numbers and actively generated plots.

Such examples can also be developed without the luxury of having live video picture of the experiment, but the realistic impression of the experiment was for the users to test the most important point of using a system. In such a tele-training application video observation can actually be a completely separate subsystem (Figure 1). It can be provided as a sort of an add-on to any remote laboratory application.
The virtual laboratory approach is based on the concept that it provides a working facility for programming and hands-on training whilst reducing the need for high cost actual devices. It is desirable that the robot simulation should be capable of being executed through any standard WWW browser application, e.g. Netscape Navigator etc [4]. Standard browsers for the VRML 97 language don't incorporate collision detection between shapes in the virtual world [13]. Because the adopted control strategy does not provide the remote user with immediate feedback from the actual work-cell, it is desirable that some kind of collision detection between the virtual robot and the virtual environment is created to prevent, or to predict, robot collisions in the real world. This problem may be solved by building JAVA oriented collision detection software or, as it was decided, to use libraries of the complete browser [7] and collision detection software [11] in the C++ language to decrease the execution time of the complete software or increase an animation speed.

The user must first download and install the complete MA2000 Robot Simulation application software (the teach pendant). Communication between the virtual robot model of the robotic manipulator, which is viewed by the remote user, and the control system which positions the joints of the actual laboratory based manipulator is achieved as follows:

- the user develops a robot task within the virtual environment,
- transmission of the completed robot task file from the remote user to the MuMaTE laboratory server,
- authentication, error checking and runtime scheduling of the received task file on the server,
- execution of the requested task within laboratory workcell, and finally
- collation and return the results to the remote user.
The laboratory test equipment includes:

- a WWW network server,
- a network layer,
- a robot workcell, and
- remote user personal computers.

The WWW network server is responsible for processing the requests for information by an external WWW browser, installed on the users remote personal computer, delivering on-line documents and providing access to the robotic and control hardware. The server is implemented currently on an Intel P166 Pentium based personal computer, running the Windows ‘95 operating system and a WWW server application program.

The robot work-cell, shown in Figure 3, allows Point to Point (PTP) motion of the robot. The robot data and environment are constant and are set in the VR software. The motion data is programmed by the user. The work-cell includes the MA2000 six axis educational robotic arm, manufactured by TecQuipment Ltd, which is supplied with its own software and teach pendant for developing tasks. It is designed to be driven from a host computer which then passes position and status information as a stream of parameters to a separate motor control system as each step within the robots programmed task is executed. The motor control system is based around an 8 bit microprocessor (Rockwell 6502) and is responsible for implementing the PID servo control for each joint, communication with the host to obtain updated position set points and control gains, acquisition of joint positions and current status of peripheral process devices within the work-cell. The robot controller achieves a constant sampling time for the position control loops, leaving the WWW server free to concentrate on the network interface and services user browser requests, thus overcoming the aperiodic sampling problems of the earlier servo experiment.
Within the university domain, network servers and local clients are connected to the internet via the campus 10Mbps Ethernet. Home user clients however utilise a much slower 14.4k or 28.8K modem connection to their local Internet Service Provider (ISP) using Point to Point Protocol (PPP). A copy of the virtual environment has to be installed on the home client’s computer. This configuration was chosen to allow various interfacing strategies to be investigated, whilst maintaining an open architecture for the future development of the project.

3.1 Software organisation

Following the success of the on-line servo experiment in the Mechatronics laboratory [2, 3], the development of an improved human computer interface, integrating the C++ language and VRML language within a non immersive desktop virtual reality environment, was undertaken in order to help improve the realism and sense of presence the user feels when programming the robot. This simulation tool allows the kinematic and dynamic behaviour of the system to be studied, and permits research into task planning, process synchronisation and the communication issues involved with the control of robotic manipulators. A robotic work-cell has also been constructed to enable the performance of these novel control paradigms to be studied within a real world environment. The additional processes which are executed by the MuMaTE server and clients remote PC are shown in Figure 4. Figure 5 illustrates the users view of the robot model and its associated 'virtual' teach pendant.

3.2 Interface to the server

The remote user posts a robot task file to the server by connecting to its WWW site and registering the job using an on-line form. The transmitted task file and user details are processed by a CGI program running on the server to determine user authentication, access control and job queue status.
If the work-cell is available for use then the task is sent to the robot controller, thus allowing the user to view the experiment via an online camera immediately. If the work-cell is currently in use then the user may decide to cancel their job and try again later, otherwise the file is placed in a queue for execution at a later date. In this case an acknowledgement will be returned to the user and the results stored in an on-line archive for retrieval at a later date.

The acquisition of data from within the work-cell takes the form of on-line video footage (at 3-4 frames per second using internet videoconferencing) captured by a digital camera located in the work-cell and numerical data (e.g. positional errors and timing information). Numerical data is collected via a data-logger and is returned at the end of the experiment which allow the various joint trajectories to be studied graphically off-line.

Real time manipulator control is achieved using a separate microprocessor based control system. However the existing controller requires that set points for limb movement, motor drive characteristics, process status, etc. are provided as a stream of parameters from a host computer, in this case the server, to the robot’s own control system, thus necessitating the development of a replacement command scheduling program (written in C++) to replace the manufacturer’s original software.

3.3 Robot task file

Robot task file transmission is run via the internet network using the hypertext transfer protocol. The ASCII text 'robot task file' (file transmit) adheres to the format shown in Figure 6. The 'hash' character is used to identify user comments within the program, whilst 'step0:' etc. indicates the beginning of a formatted block of data representing the next operation to be performed by the robot. The robot's overall task is broken down into a sequential list of
intermediate operations or 'steps', each of which is capable of changing the control parameters and pose of the manipulator using the numeric values contained in a data matrix.

The default data matrix takes the form described in Table 1. The upper row specifies work-cell and manipulator control settings whilst the lower row defines joint position data for the current operation. Variable 'rate' determines the speed at which the robot travels when performing the current step (1 is the slowest, 9 is the fastest). Variable 'mode' allows new function types to be incorporated within the data matrix (the default is 2). For example, setting the 'mode' value to '99' allows the data matrix to be used to modify the individual controller gains of the joint position servo. In this case the top line of the default data matrix will now change from: rate, input, output and wait to joint’s PID controller constants: joint, K_P, K_I and K_D, respectively. Variable 'input' interrogates the status up to four additional peripheral sensing devices within the robot work-cell (the default is 0 and means: ignore input devices). Variable 'output' actuates any four peripheral output devices connected within the work-cell (the default is 0 and means: ignore output devices). Variable 'wait' invokes a time delay before proceeding to the next step and value is assumed to be in seconds (the default is 0 and means: no delay). Variable 'jump' forces program to jump to a specified step number (the default is 0 and means: no jump). Variables 'waist', 'shoulder' and 'elbow' present waist, shoulder and elbow joint position and have values between 000-999 (position between 0 and 270 degrees). Variables 'pitch', 'yaw' and 'roll' present pitch, yaw and roll end effector joint position data and have values between 000-999 (position between 0 and 180 degrees). Variable 'grip' actuates pneumatic gripper (0-closed, 1-open).

3.4 Interface between VRML robot model and VRaniML browser

Figure 7 a shows a part of the VRML robot model written in the VRML 97 language. From the code we can see that JOINT2 shape (shoulder) is a child of JOINT1 shape (waist) so JOINT2 shape moves whenever JOINT1 shape moves. The VRaniML browser library [7],
written in C++, allows the user to display and move different entities described by the VRML robot model. Figure 7 b shows how effective the VRaniML browser library is in importing a VRML scene, identifying and changing the orientation of a given joint within this scene. The fourth line in Figure 7 b reads and displays the virtual robot mechanism and a robot environment described in Figure 8 on the computer screen. Line five finds the JOINT2 definition of the VRML robot and sets the pointer on mJOINT2. Line six is a function which calculates a rotation position of JOINT2, and the last line changes the angle in DEF JOINT2 Transform's command in the VRML robot model described in Figure 8.

3.5 VRaniML browser and V-collide software interface

V-Collide is a C++ library for interactive collision detection among arbitrary polygonal models undergoing rigid motions in VRML environments [11]. This library offers a practical toolkit for performing interactive and robust collision detection in VRML environments [13]. The VRaniML browser library, also written in C++, is used for the displaying of VRML models and movements of virtual bodies and shapes whilst the V-Collide library is responsible for preventing collision between virtual bodies. An interface between both libraries had to be written because of the significant differences which exist between them. The VRaniML library uses the grammatical rules of the VRML 97 language whilst V-Collide uses a homogeneous transformation matrix [3] for description of shapes in the virtual world.

Interpreting complex shapes with arbitrary polygonal models (triangles) is computationally expensive, therefore the following simplifications were made to the collision model. The collision regions of the first three robot joints (waist, shoulder, elbow) are modelled by coarse polygonal approximations to the VRML geometry of the robot model, this can be seen from comparing the exact second joint of the VRML robot model and a transparent box of the same joint made as collision detection model on Figure 8. The robot
workspace and end effector geometry (gripper, pitch, yaw and roll) however, adopt an exact collision model. In fact, most collisions will occur between the robot end effector and the work-cell environment. Clearly, the first three joints can collide too (for example joint 1 and joint 3 can collide between themselves and joint 3 can collide with the work-cell) but the same degree of precision in locating the exact point of collision is not required in these cases. This simplification to the collision model achieves significantly faster computations (hence screen refresh rate and animation speed) than would have been possible with an exact collision model applied to all joints. Figure 8 shows the exact robot model built within VRML and transparent V-Collide collision regions.

4 CONCLUSION

This paper introduced a WWW based virtual laboratory for robotics and control engineering students, which provides users with on-line access to real-world hardware for remote experimentation. The approach requires the user to develop tasks off-line, using their local computing resources, before submitting the experiment to the laboratory server for execution on the actual device.

A control experiment of a real-time video supported Web controlled magnetic suspension system has been implemented, users can specify the required operating parameters, invoke the experiment and observe the response of the system also via a real-time video transmission via internet.

A more sophisticated example involves the control of a robotic manipulator. A VRML based simulation model and teach pendant has been developed together with a distributed control methodology to eliminate the unpredictable network loading problems and variable transmission times faced by other direct tele-operated systems. The V-Collide library of C++ collision detection functions has also been integrated within the simulator to provide realistic detection of collisions between the virtual robot and its associated workspace.
References


Tables and Figures

<table>
<thead>
<tr>
<th>rate / joint</th>
<th>mode</th>
<th>input / $K_P$</th>
<th>output / $K_I$</th>
<th>wait / $K_D$</th>
<th>jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>waist</td>
<td>shoulder</td>
<td>elbow</td>
<td>pitch</td>
<td>yaw</td>
<td>roll</td>
</tr>
</tbody>
</table>

Table 1: Default Data Matrix

Figure 1: Remote laboratory connection with video

Figure 2: Interactive web page of a magnetic suspension system model

Figure 3: The Robot Workcell
Figure 4: The Robot Interface Between a Server and a Client
Figure 5: Virtual Teach Pendant and Robot Model

Figure 6: Robot Task File

Figure 8: VRML Model of MA2000 Robot and Robot Model With Collision Regions
DEF Robot Transform(
    translation 0 0 0 -2
    rotation 0 0 0 1 0 1.28
    children
    [  
      DEF BASE Inline(
        url "basebox.wrl"
      ),
      DEF JOINT1 Transform(
        rotation 0 0 0 1 0 0
        children[
          Inline(
            url "link1.wrl"
          ),
        ]
      ),
      DEF JOINT2 Transform(
        rotation 0 0 1 0 0 0 -0.47
        center 0 0 0 260.0
        children[
          Inline(
            url "link2.wrl"
          ),
        ]
      ),
      ...
    ]])]

a)

#include "vranim.h"

b->ReadURL("c:/robot_c/ma2000s.wrl");
vrTransform *mJOINT2 = 
  (vrTransform *)b->FindByName("JOINT2");
JOINT2rot=calculate_pos(JOINT2);
mJOINT2->SetRotation(SFRotation
(0, 1, 0, JOINT2rot));

b)

Figure 7: a) VRML Robot Model Program
and b) Rotating JOINT2 With Browser