Ergonomic and Efficient Software Alternatives for High Cost Manipulators - Direct, Wireless and Networked Control Techniques

RAUL MIHALI
School of Engineering and Design
169 University Avenue, Bridgeport, CT 06601,
U.S.A. Phone: (203) 576-4116, Fax: (203) 576-4766

TAREK SOBH
School of Engineering and Design
169 University Avenue, Bridgeport, CT 06601,
U.S.A. Phone: (203) 576-4116, Fax: (203) 576-4766

Abstract The process of deciding-on and purchasing the right manipulator(s) for a predetermined task can often turn to be very frustrating, especially when budget and purchase timing are essential factors. The market tends to get larger and variety driven and there is a choice for almost any given price range, however, the price / size ratio seems to remain constant. Larger scale manipulators do not show the price amortization enjoyed by the majority of computerized consumer hardware over the past few years. In addition, the manufacturers for many of these manipulators do not provide adequate pre-sales supporting technical material (whether a result of lack of standardized specifications or pure negligence), nor effective warranties and service.
Primarily affected are higher level educational institutions, where manipulators are likely to be exposed to student projects that demand constant diversity and various controlling software and hardware technique. These manipulators are likely to become victims of abusive usage and, in addition, the institutions need to offer some of the highest standards of safety for the students.
This paper presents a software simulation and control package applied on a specific manipulator. The package presents a significant tool in solving problems such as the above mentioned ones. In addition, the software offers a variety of implementation examples that can be directly derived from the simulation package.

1. INTRODUCTION

The paper starts by presenting some of the aspects of the manipulator used, then describes a fully functional simulation and control software specifically designed to address the problems mentioned in the *abstract* section. The software package could be used for example from student residences as a "virtual" manipulator so that students can write their own simulation and control software (project, homework assignment, etc) that could be then tested "live" on the actual robot next class. Such usage can also significantly reduce the safety risks involved with freshmen students attempting to control the robot. The package can also be “worked-on” by the students, for example adding vision processing or any project specific duties, once the controlling and simulation parts are no longer of interest to develop. The software could also act as a remote manipulation tool from anywhere on the web, by having it connect to another copy of the tool that resides as a net server on a machine that is connected to the manipulator serially. The model can be extended to an unlimited set of simulation packages that are all interconnected through TCP/IP and ultimately connected to the actual robot. Again, these are only some of the immediate applications that although common, are problematic issues in most engineering schools.

2. THE MANIPULATOR

We use a manipulator manufactured by Mitsubishi, model RV-M1 (Movemaster EX) (figure 1).

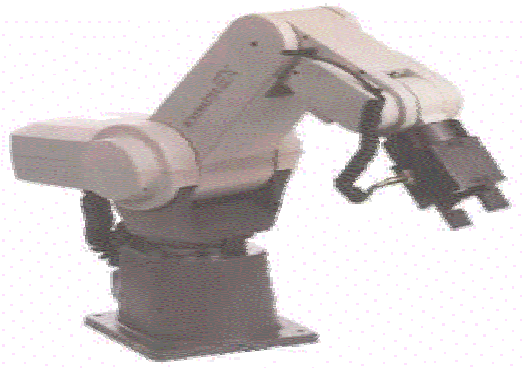

Figure 1. Mitsubishi RV-M1 (Movemaster EX)

The manipulator is a general-purpose commercial robot used in industrial applications (for example, pharmaceutical / chemical industry to manipulate substances in a grid).
The arm offers 5 degrees of freedom (not including the gripper). We will detail specifications as needed through the paper, the reader can consult a distributor for a detailed brochure (www.rixan.com for example). The robot comes with all the necessary information to program it from a serial port equipped computer or from its "teaching" pad and has a software package (mainly editor) that allows writing short program sets using the robot's language set and have them stored in its RAM/EPROM. The controller of the robot accepts direct

and inverse kinematics commands directly through the serial port.

3. THE SIMULATOR

3.1. Overview

The simulator was designed from its inception with the student as a main beneficiary in mind. One of the goals was to be able to reproduce as much as possible the actual robot and its characteristics through the software package, in such a way so that the software itself could act as a "virtual" manipulator, almost replacing the need for the actual manipulator. Such an approach should allow the students to familiarize themselves much better with the respective manipulator, and to face no surprises when later connecting to the actual robot.

3.2. DESIGN CONSIDERATION

3.2. A. Interface, GUI

One of the commonly encountered problems in the majority of the simulators available nowadays is their graphic user interface layer. Students tend to get excited by installing a certain simulator, but very often loose their determination when they see a briefly sketched set of links, lack of an intuitive GUIs or instant overloads of variables and input coordinates. While certainly having a remarkably vast level of theoretical complications and combinations, our simulator maintains a very presentable visual and physical implementation that tends to be the actual final product.

We opted for using OpenGL and rendering the manipulator closely to the actual model, so that it cannot be confused with any other manipulator (figure 2).

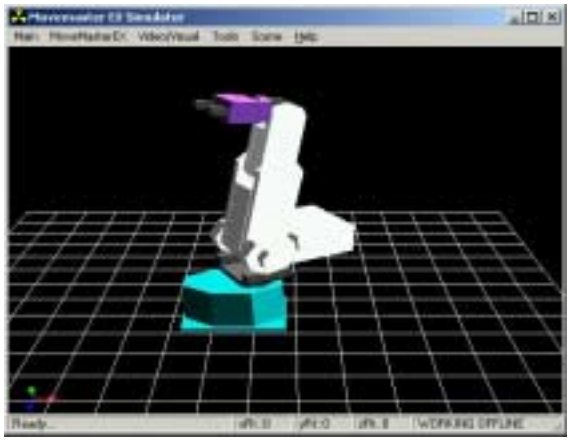

Figure 2. Simulated manipulator

3.2. B. Programming Language

A second aspect to consider is the programming language to choose. Many students or engineers do not always have the right aspiration towards advanced programming techniques, and probably this should not be a showstopper for a robotics enthusiast.

As a result, we opted for using Visual Basic and making the code as simple (though robust) as possible. With the help of publicly available software tools [1], the OpenGL power was integrated in Visual Basic. Having these two entities as a starting base, the simulator code proves to be simple; minor changes in the code can derive to custom requirements. Visual Basic is also a great medium for describing robotics equations such as inverse kinematics / dynamics, trajectory calculations, as debugging of these tends to be much simpler when compared to most of the other languages. Of course there is a limitation drawback that boosts C++ as a preferred choice at the professional levels (fast synchronizations, advanced hardware control at assembler level, etc), although the differences tend to be diminished lately by technologies like Active X.

The following sections present each distinctive part of the simulator

3.3. Front End (GUI)

When the simulator is being activated, the user has the view from Figure 2, and dragging of the mouse over the graphics scene will rotate the point of view around the manipulator. The user has the choice to perform many different view related operations through the *Scene* tab: set the mouse to perform desired rotations, translations, change the point of view or lock onto views such as “top”, “side”, etc. The coordinates of the viewing point are dynamically updated on the status bar of this view. The orientation of the axes is also displayed in the lower left corner, and their coloration is being used consistently throughout the simulation package to represent distinctly each of the axes. In general, any of the options that are being used have a direct effect on the CAD manipulator displayed and even on the actual robot, if a connection is active.

3.4. Kinematics

Although the MoveMaster EX manipulator accepts as controlling parameters both direct and inverse kinematics by design (thetas or X, Y, Z, roll, pitch, yaw), in the simulator we decided to implement the direct and inverse kinematics as well, giving the user the option to see direct/inverse kinematics action on the CAD model itself, without the need to be connected to the manipulator. The inverse kinematics equations were solved through direct geometric / trigonometric approaches [2], although similar equations would have been reached through the usage of more traditional DH (Denavit – Hartenberg) tables [3]. A step-by-step demonstration of the equations used are available online at www.bridgeport.edu/~risc,

and [4,5,6] show previous similar simulation work that was successfully implemented.
The kinematics control module is available through the "MoveMasterEX" tab (Figure 3).

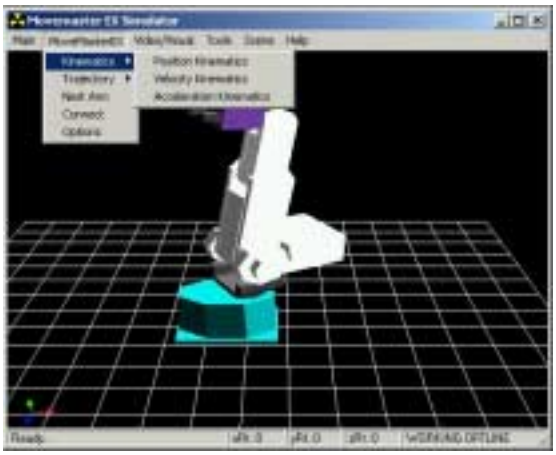

Figure 3. MoveMaster EX tab

Notice that although the Velocity Kinematics and Acceleration Kinematics are provided as sub options under the Kinematics options, they are not implemented as the MoveMasterEX manipulator does not support them (the manipulator only has a limited velocity control, a choice of 5 or 6 preset values [7]). If it is desired to adjust the software tool for a different manipulator, then the developer will implement these as needed, following closely the implementation of the existing kinematics model.
The Position Kinematics interface (Figure 4) allows the direct and inverse kinematics control of the robot.

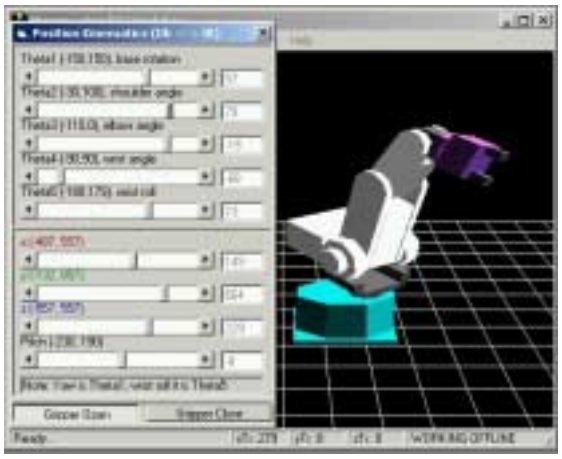

Figure 4. Position Kinematics Interface

The activation of any of the direct kinematics scroll bars will instantly update the inverse kinematics ones and vice versa. The CAD manipulator itself moves accordingly too. If the simulation package is connected to the actual manipulator or a server version of the simulator, these will move too (these features are described through the following sections). If by adjusting any of the inverse kinematics scrollbars the manipulator would risk a singularity or an out-of-workspace position (solution), the user will be warned and both the CAD robot and the actual one (if connected) will not be updated until a new correct (possible) position is reached.
Such an implementation allows a very safe control / strategy for the existing manipulator and allows the user to easily observe the actual workspace and its limitations.
The marginal values used for *thetas* and the inverse kinematics were matched from the robot's technical manual [7]. Minor discrepancies were noticed, which are typical and ignorable for this particular class of manipulator.

3.5. Trajectories

Trajectory control / plotting is an essential step in any moderated robot control project. The simulator encapsulates a robust trajectory generation module, and through the easy to use source code, the user should be able to observe and modify as needed the implementations. The trajectory curves implemented in this package are Lagrange, COONS, Hermite, B-Spline, Bezier and Ferguson [8], which should be more than sufficient for most of the applications (Figure 5).

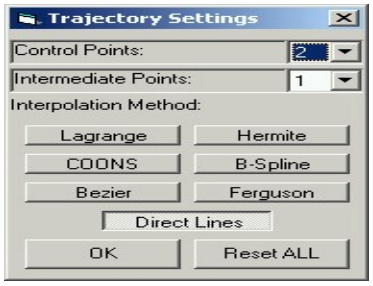

Figure 5. Trajectory Settings

The user can set up and adjust trajectories without too much experience with the simulator. A set of control points needs to be defined (2 to 50), then a number of intermediate points for the interpolations and the trajectory will be dynamically adjusted in the scene and can be applied to the actual manipulator through the *Apply* option. Figures 6, 7, 8 show a few examples of designed trajectories (Lagrange, Hermite and Bezier respectively).

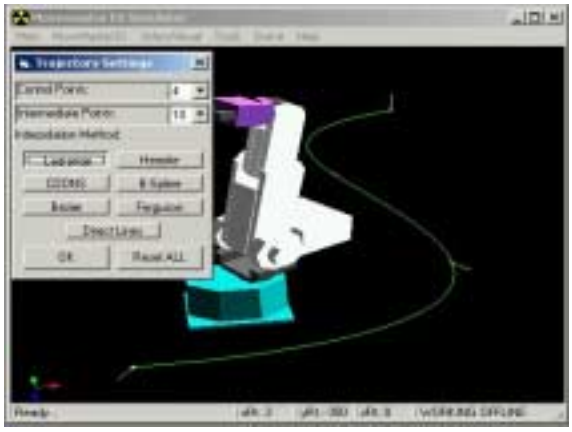

Figure 6. Four points Lagrange trajectory curve

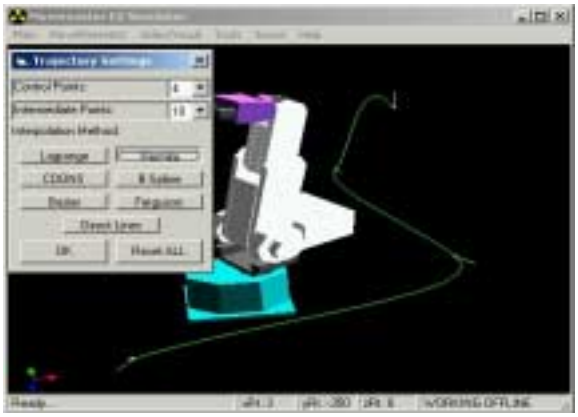

Figure 7. Four points Hermite trajectory curve

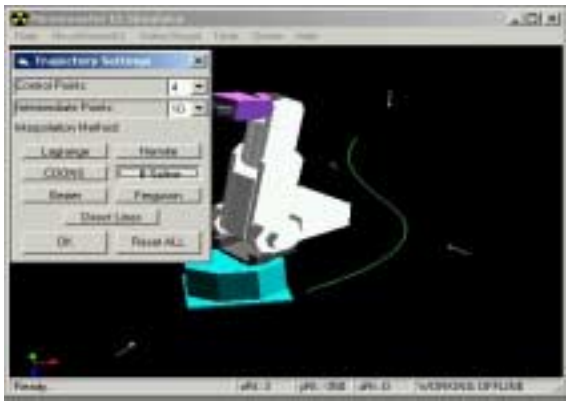

Figure 8. Four points Bezier trajectory curve

For choosing and adjusting the actual control points and their exact order, the user will combine the Position Kinematics panel described above with the *Trajectory Point Set* option (Figure 9).

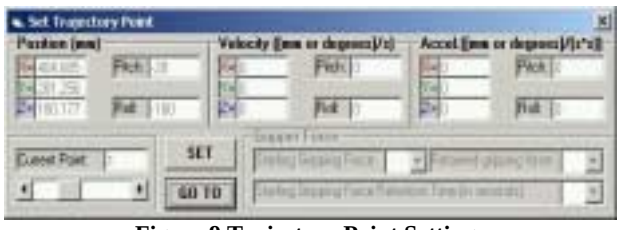

Figure 9 Trajectory Point Settings

Once the arm is moved to the desired location and with the desired pitch/yaw (using the Position Kinematics Pad), the user can *set* this point, *set* the gripping forces and navigate from a set point to another (using the Trajectory Point Settings panel). Once a trajectory is being set to the desired parameters, it can be saved as a file and reused. Although figure 9 displays options for speed and acceleration at the respective point as well, they are not implemented due to the limitations of the robot. If the simulator package is to be adjusted for a different manipulator, the developer will be able to utilize the existing interface ands will probably opt for a similar backend implementation as the one here. Notice that the actual robot will move synchronously with the users operations if it is connected to the simulator, and throughout the steps necessary to set up a trajectory the CAD model presents continuous feedback to the user.

3.6. Other MoveMaster EX Settings

For optimal results, the simulator allows for fine adjustment of some of the manipulator's simulation parameters (Fig 10), as follows:

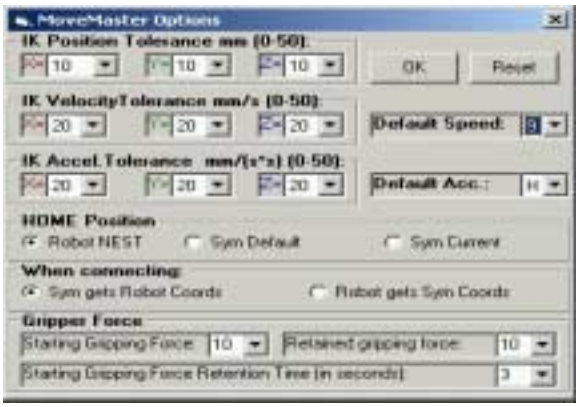

Figure 10. MoveMaster Settings

IK tolerance

Allows the user to adjust the values that determine the precision of the inverse kinematics module. The user needs to be careful with the values as they could cause inverse kinematics replies that are not within the actual workspace of the manipulator. The IK Velocity and IK Acceleration Tolerances are not implemented due to the limitations of this robot, although are present if a different robot will be used.

Home Position

The user can redefine the nesting position of the arm, so when the simulator is connected to the actual robot, the robot will move to the newly set nest position. *Robot Nest*, will use the factory (or teaching pad) set nesting position. *Sym Default*, will use the simulator's default nest position, which in this case coincides with the factory default one. *Sym Current*, the current simulator/CAD position will be used as nest position.

When Connecting
The user can also control the way in which the position synchronization between the CAD model and actual robot is done when the connection is made. *Sym Gets Robot Coords*, will leave the manipulator at its current position and adjust the CAD model coordinates to match that, while *Robot Gets Sym Coords* will cause the manipulator to move to the position current in the simulator/CAD model.

3.7. Vision Features

To ease the development of vision processing algorithms, the simulation tool allows the user to connect a camera to the package and have frames or sequences of frames available for processing. We have tested the simulator with a USB camera model DVC323 by Kodak under Microsoft Windows 2000, although any camera with a valid VFW (Video for Windows) driver will work as well. There are plenty of choices for controlling a digital camera or camcorder from within a Visual Basic application. For the visual support, we have arbitrarily picked a publicly available OCX control (Xvideo2 by www.cbcsolutions.com).
A more distinct feature in the simulation package is the ability to have the package run in server mode and have a client session connect to it and retrieve for processing a bitmap image of the actual "virtual" scene. For example a user could decide to add a few objects to the scene (Figure 11) of the server application, then have this bitmap transmitted to the client session for actual vision processing.

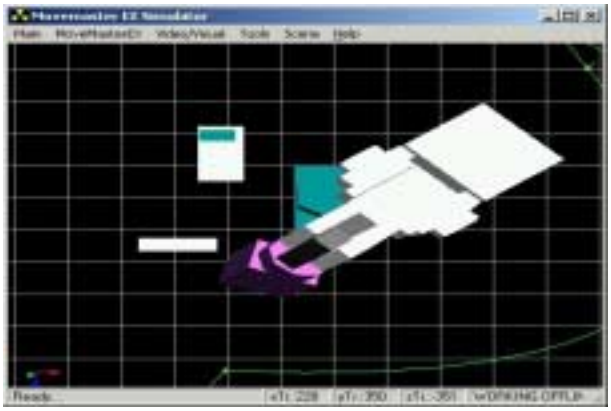

Figure 11. Objects added to the scene

The client tool would normally not have these objects in the scene, as it would be used as a simulator /control tool on the images that arrive from the server, which in fact acts as the virtual "real" manipulator. The goal could be to grab the virtual objects (for example), and the user could also request different views of the server scene for easier processing. Notice that this would not be possible through an actual (physical) camera, as cameras cannot be dynamically re-positioned unless a second or more manipulators exist. The simulation tool can also be used to send to any client level application the actual video camera images that are grabbed as described in the previous paragraphs.
As an example, a student could build a simulation and control package with vision processing that would actually perform on this simulator and not on an actual robot, there would be no need to buy the manipulator, nor the camera. If either of them or both are present, the simulator can be used as well.

3.8. Connecting to the Actual Robot

The connection to this robot needs to be done through a serial port. The process has been simplified and the typical failures of adjusting the port settings have been eliminated (Figure 12).

Figure 12. MoveMaster Connection Settings

Although the default settings should only require the change to the connected COM port, any other serial port option can be adjusted and tested. Once the test is successful the connection can be made and the actual robot will be in synchronization with the CAD model. The TEST choice will ensure that the robot is properly connected and does feedback properly on the selected port coordinates.
A second connection choice is available too, namely TCP/IP. If another copy of this simulation tool is running and is active as a server, its IP and Port need to be specified and the connection can now be made to the second simulator, which consequently can be connected to the actual robot (please see next section for more details), or to a chain of other servers and then to the actual robot.

3.9. Networking the Simulator

The simulator can be switched into *Server mode*, which will allow a client session of the simulator, usually located elsewhere geographically, to connect through TCP/IP and control the server side CAD model. The connected client communicates with the server through direct kinematics (*thetas*), although the TCP/IP port protocol implementation is made easy enough to allow any sort of communication, even direct passing of robot specific commands to the serial port of the robot. Notice that the server could be simultaneously connected to the actual robot or be a client to another instance of the tool, thus, a chain of simulation packages is possible (Figure 13).

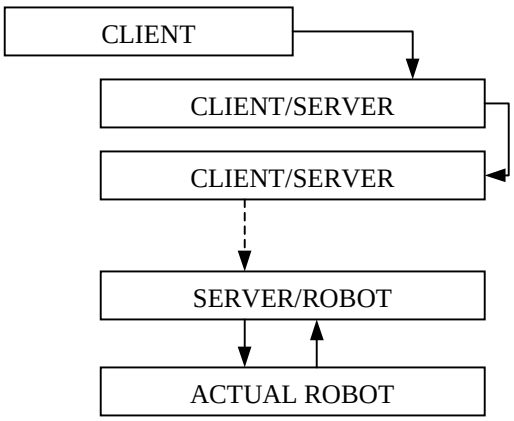

Figure 13 Networking Model

Such a connection model would be very appropriate for a class simulation for example, where each student's workstation could be set to display the exact settings of the client simulator (professor's or project presenter's workstation) and this one further chained to the actual robot as well. The TCP/IP networking also allows for easier development of any other simulation software, by simply running the server and the "to be designed" client on the same machine as in Figure 14.

Any of the chained workstations can be controlled as well through the position kinematics interface, in this case overriding all the consequent workstations down on the chain until the actual robot (if connected). The workstations could also be just left to display the position kinematics interface, which would adjust the scroll bars automatically when a connected client would send *thetas*.

As an extension to the networked model and its server side implementation, we have extended the package for wireless control through a cell phone, a wireless (HDML - Handheld Device Markup Language) server that allows basic control of the manipulator.

A logging window allows the visualization of the protocol messages, while the CAD model and (if connected) the actual manipulator will move to the various cell phone sent commands.

On the cell phone, once the web browser is pointed to the IP/port address mentioned on the top of the server window, the user is sent a small HDML page that allows him / her to activate any of the joints of the manipulator by pressing a key from 0 to 9 (0,1: base angle increase / decrease, 2-3: elbow angle, etc). Once the user selects one of the options, the server intercepts the choice, moves the robot joint and presents the same controlling page for further movements. Various websites such as [11] proved useful in building the necessary HDM. The cell phone server features of the simulator were tested with various cell phone models and various service providers.

4. CONCLUSIONS

In this paper, we present a software model designed to alleviate the need to have access to high cost manipulators. The presented software package offers a variety of usage possibilities, from a standalone simulation package, a networked simulation package, to a complete "virtual" manipulator package. The availability of similar simulation tools for the majority of high cost manipulators would solve the majority of the problems involved with the acquisition of these manipulators. The simulator could be used in educational institutions, by allowing the students to perform a large number of projects involving a certain manipulator without actually purchasing one, or purchasing a single or s limited number of robots for final demonstration purposes only. The simulator could, for example, be given to the student for an extended project and have him / her continue the work without needing access to the actual manipulator. The tool can evidently be used very well as a remote automation system, or as a distance learning method, especially by setting up a networked chain for all the students in a distance learning class, with one student or instructor demonstrating on the actual robot and the rest following the scenes closely on their workstations.

Currently we have successfully tested and used the features detailed through this paper. The simulation package can be found at www.bridgeport.edu/~risc. The future work on this package will not be detailed here, as the package itself was designed as only a basis for various future applications.

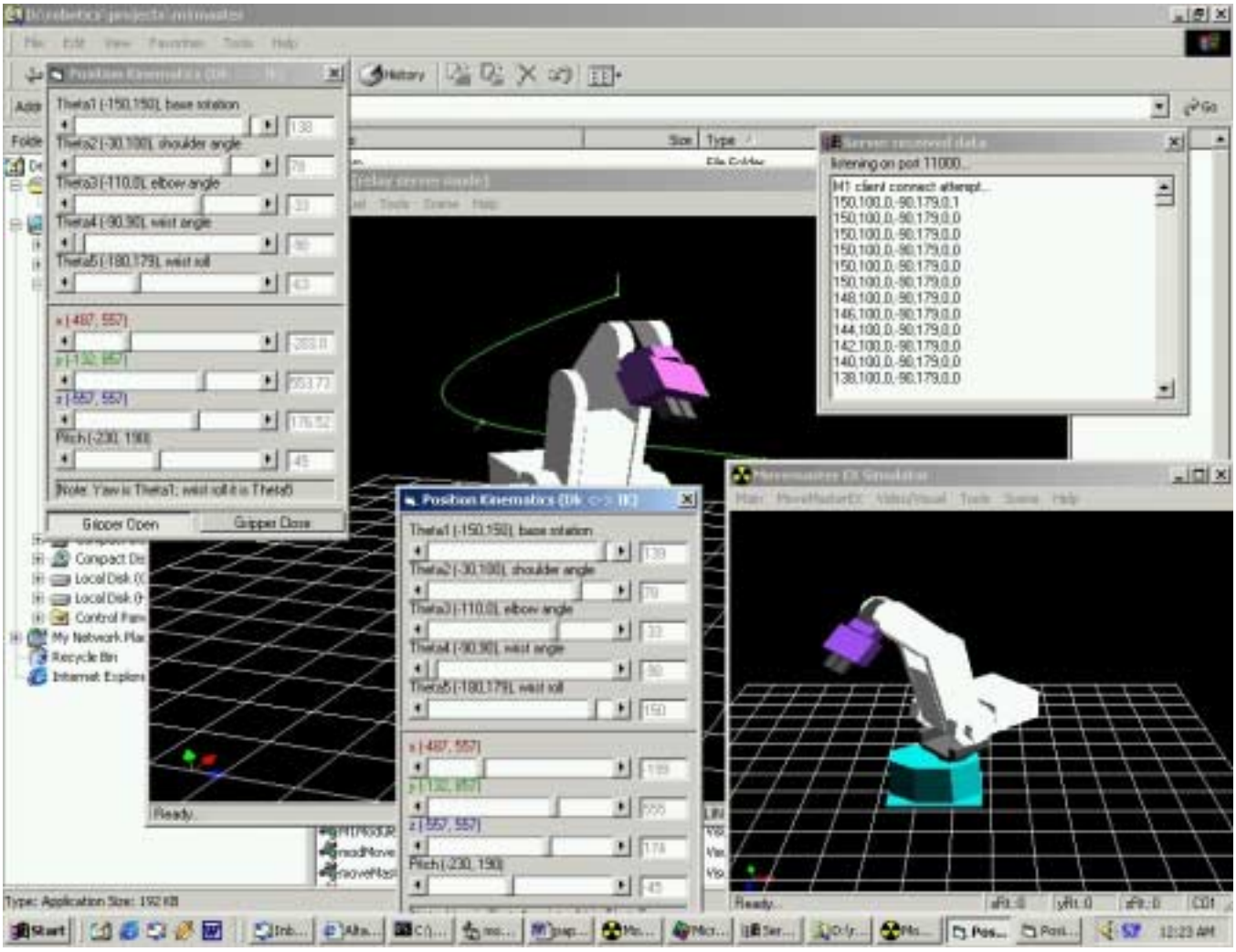

Figure 14. Server and Client applications running on the same workstation

References

[1] http://home.pacific.net.hk/~edx/

[2] Benedetti, R., and Risler, J. J. *"In Real Algebraic and Semi-algebraic Sets*" (1990), Hermann, pp. 8-19

[3] McKerrow, Phillip John, "*Introduction to Robotics"*, Addison Wesley, 1991

[4] *Journal of Intelligent and Robotic Systems - Theory and Applications (Incorporating Mechatronic Systems Engineering)* / Kluwer Academic Publishing, Mihali, Raul C., Sobh, Tarek M., *The Formula One Tire Changing Robot (F1-T.C.R.)*, accepted for publication in the Journal of Intelligent and Robotic Systems, April 1999

[5] *Journal of Intelligent and Robotic Systems - Theory and Applications (Incorporating Mechatronic Systems Engineering)* / Kluwer Academic Publishing, Mihali, Raul C., Mher Grigorian, Sobh, Tarek M., An Application of Robotic Optimization: Design for a Tire Changing Robot, 28-36, 1999

[6] *Journal of Intelligent and Robotic Systems - Theory and Applications (Incorporating Mechatronic Systems Engineering)* / Kluwer Academic Publishing, Tarek M. Sobh, Abdelshakour A. Abuzneid and Raul Mihali, *A PC-Based Simulator/Controller/Monitor software for manipulators and Electromechanical Systems, 2000*

[7] *Mitsubishi Industrial Micro-Robot System Model RV-M1 MoveMaster EX Technical Manual*, Mitsubishi Electric Corporation, Japan

[8] http://www.ccwap.com/hdml.htm

[9] *Assisted Graphics - FORTRAN programs for Geometrical Representations, Volumes I and II*, A. Tanasescu, R. Constantinescu, I.D.Marinescu, L. Busuioc, Editura TEHNICA, Bucharest, 1989