Effective Simulation and Control Techniques for Alleviating the Access to High-Cost Manipulators. Wireless Control Perspectives

RAUL MIHALI, TAREK SOBH
Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, CT 06601, USA
May 2001

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, time or minimal losses appear as essential factors. The market does tend to get larger and variety driven and there is a choice for almost any given price, however, the price / size ratio seems to remain constant. Larger scale manipulators do not show the price amortization enjoyed by the majority of the 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, various controlling software and hardware technique; they are likely to become victims of abusive usage and in addition to these, the institutions need to offer some of the highest standards of safety for the students.

This paper presents a flavor of a software simulation and control package applied on a specific manipulator that is a significant tool in solving problems such as the above mentioned ones. In addition, the package offers a variety of implementation examples that can be directly (immediately) derived from the simulation package, as well as a completely functional implementation of a cell-phone control example of the CAD model or the actual robot.

1. INTRODUCTION

The paper goes from presenting some of the aspects of the manipulator used, to presenting a fully functional simulation and control software specifically designed to address the problems mentioned in the *abstract* section and more. The software package could for example be used from student residences for acting as a "virtual" manipulator so they can write their own simulation and control software (project, homework assignment) that could be then tested "live" on the actual robot next class. Such usage can significantly reduce the safety risks involved with freshmen students attempting to control the robot. The package can also be “worked-on” by the student, such as adding vision processing or any project specific duties, given that 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 the machine that is hooked to the manipulator serially. On top of that, the software package contains a quick full implementation example of controlling the manipulator via a regular web enabled cell-phone. Again, these are only some of the immediate applications that although not unusual, are actual issues in most of the schools.

2. THE MANIPULATOR

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

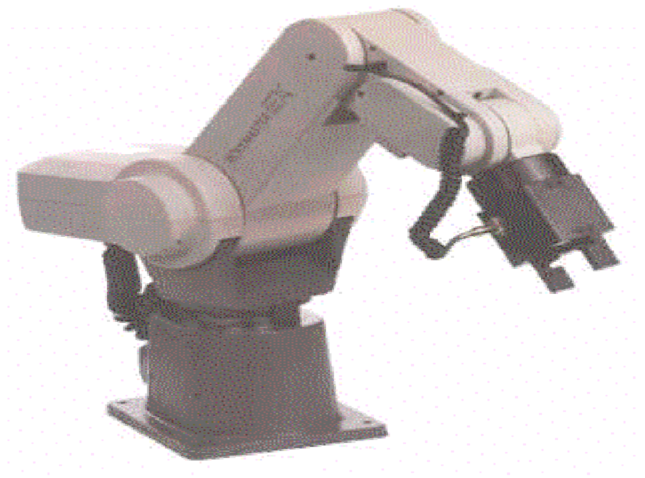


Figure 1. Mitsubishi RV-M1 (Movemaster EX)

The model is known as a general-purpose commercial manipulator used in industrial applications (for example used in pharmaceutical / chemical industry to manipulate substances in a grid).

Briefly describing, the arm offers 5 degrees of freedom (not including the gripper), DC servo motors drive. We will detail specifications as needed through the paper, but please consult a distributor for a detailed brochure (www.rixan.com for example). The robot does come 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 with the robot's language set and have them stored in its RAM/EPROM.

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 from the actual robot and its characteristics through the software, 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 student to familiarize him (her) self much better with the respective manipulator, and to have no surprises at all 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 of 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. The robotics field needs to be presented in its full color, while certainly having one of the vastest levels of theoretical complications and combinations, it maintains a very presentable visual or physical implementation that tends to be the final product.

This was one of the aspects to consider, and we opted for using OpenGL and rendering the manipulator closely to the actual model, such as it cannot be confused with any other manipulator (figure 2).

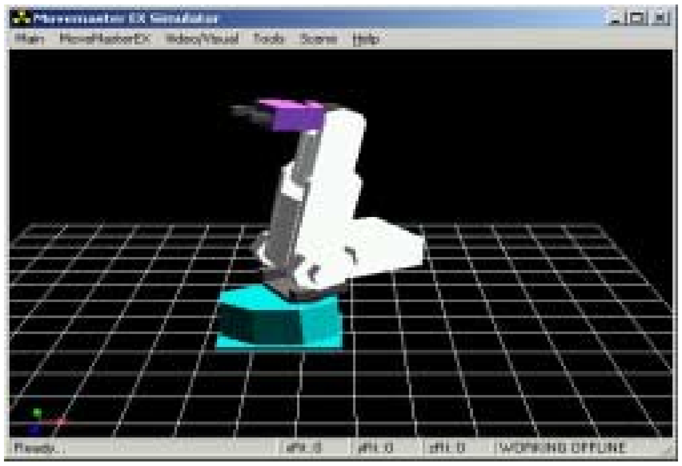


Figure 2. Simulated manipulator

3.2. B. Programming Language

A second aspect to consider, in fact immediately derived from the first one mentioned before, 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 at all a showstopper factor for a robotics enthusiast.

As a result, we opted for using Visual Basic and make 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; changes of few minutes 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 in 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 any dragging of the mouse over the graphics scene will rotate the point of view for a better observation. 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) the simulator solves the inverse kinematics as well, giving the user the option to see 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 – Hartemberg) 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 were successfully achieved.

The kinematics control module is available through the “MoveMasterEX” tab (Figure 3).

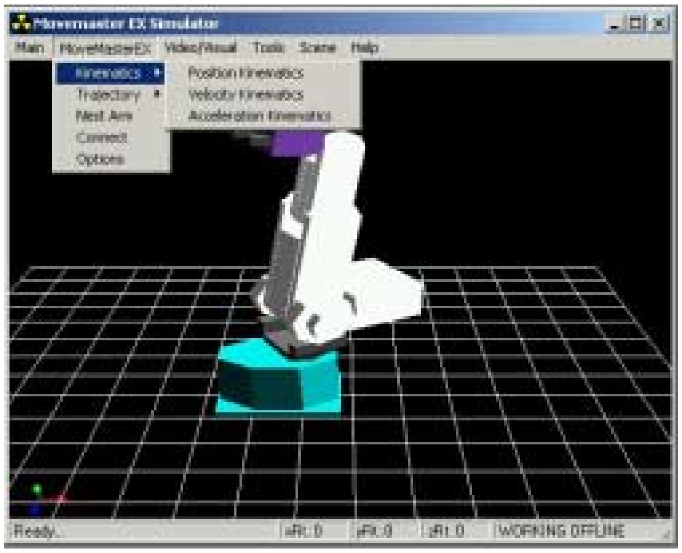
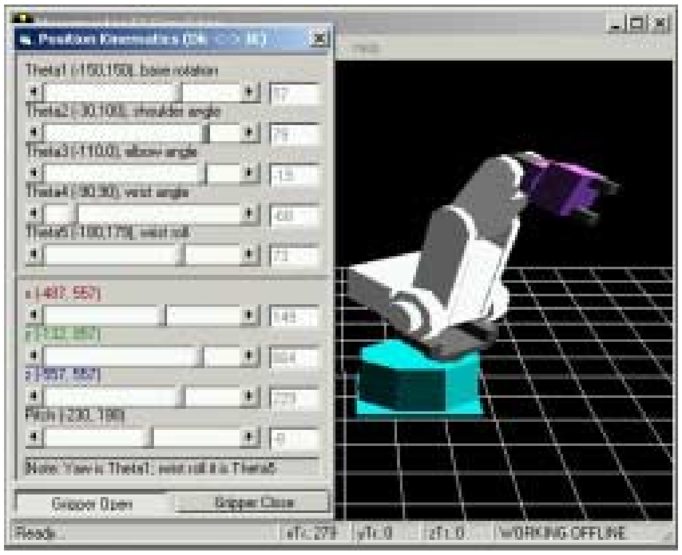


Figure 3. MoveMaster EX tab

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 5-6 preset values [7]). If it is desired to adjust the software tool for a different manipulator, then the developer will implement these as needed.

The Position Kinematics interface (Figure 4) allows the direct and inverse kinematics control of the robot.



Position Kinematics (DK <-> IK)

Theta1 (-150,150), base rotation: 17
Theta2 (-30,100), shoulder angle: 79
Theta3 (-110,0), elbow angle: -15
Theta4 (-90,90), wrist angle: -60
Theta5 (-180,179), wrist roll: 73
x (-487, 557): 145
y (-132, 887): 504
z (-557, 557): 229
Pitch (-230, 198): -9

Note: Yaw is Theta1; wrist roll it is Theta5

Figure 4. Position Kinematics Interface

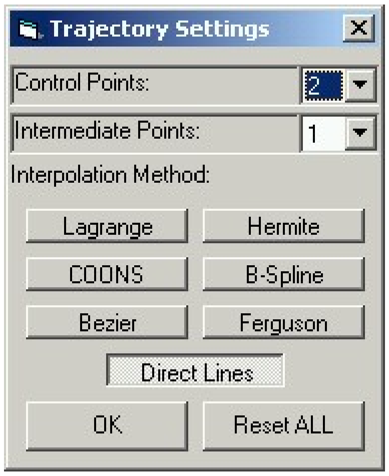
The activation of any of the direct kinematics scroll bars will instantly update the inverse kinematics ones and vice versa. The CAD manipulator itself moves accordingly too and if the simulation package is connected to the actual manipulator or a server version of the simulator, they will of course move too (these features are described through the following sections). If by controlling any of the inverse kinematics scrollbars the manipulator would risk 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 position is reached.

Such an implementation allows a very safe control over 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).



Trajectory Settings

Control Points: 2
Intermediate Points: 1
Interpolation Method:

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).

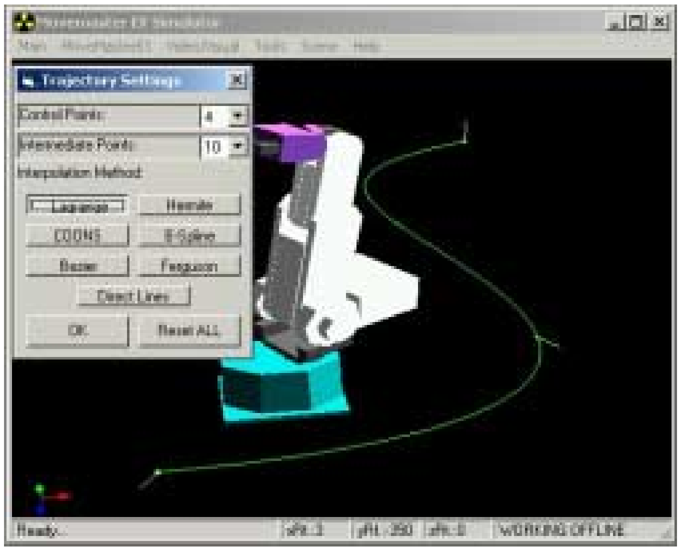

Control Points: 4
Intermediate Points: 10
Interpolation Method: Lagrange

Figure 6. Four points Lagrange trajectory curve

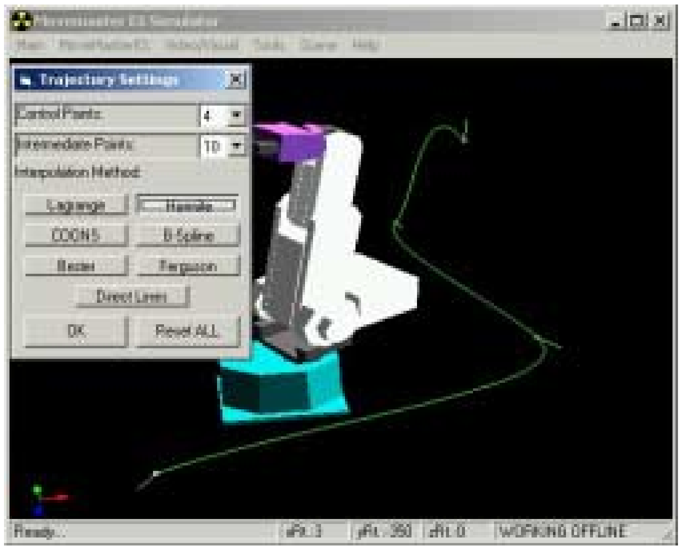

Control Points: 4
Intermediate Points: 10
Interpolation Method: Hermite

Figure 7. Four points Hermite trajectory curve

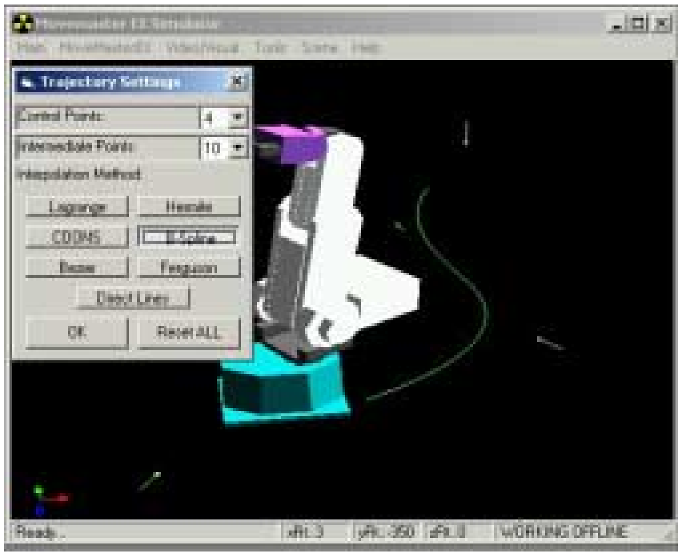

Control Points: 4
Intermediate Points: 10
Interpolation Method: B-Spline

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).

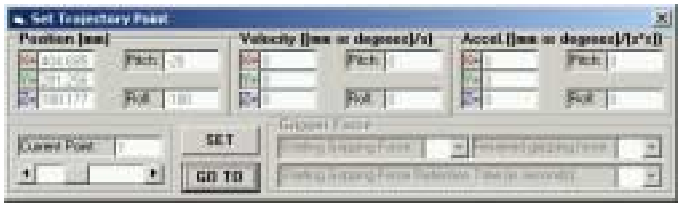
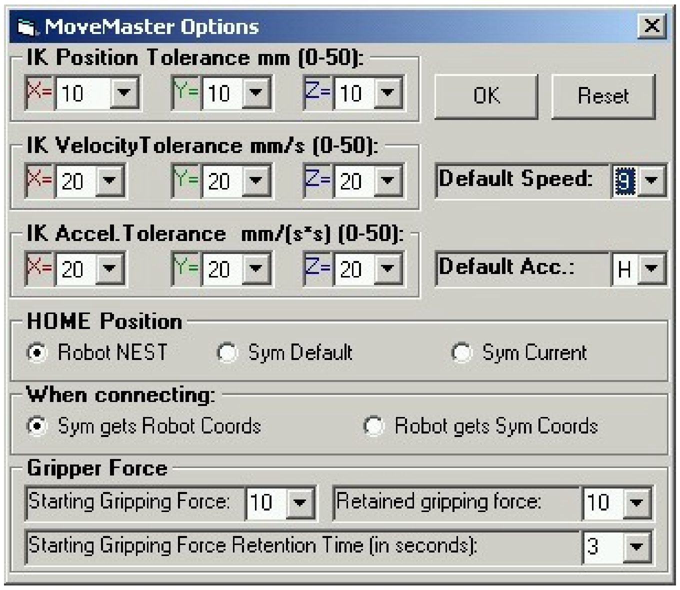


Figure 9 Trajectory Point Settings

Once the arm is moved to the desired location and with the desired pitch/yaw, the user can set this point, set the gripping forces and navigate from a set point to another. Once a trajectory is being set to the desired parameters, it can be saved as a file and reused with other occasions. Although the figure 9 displays options for speed and accelerations at the respective point as well, they are not implemented due to the limitations of the robot. 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 such as IK tolerance, redefining the nesting position of the arm, the way in which the position synchronization between the CAD model and actual robot is done when the connection is made etc (Figure 10).



MoveMaster Options

IK Position Tolerance mm (0-50): X= 10 Y= 10 Z= 10

OK Reset

IK VelocityTolerance mm/s (0-50): X= 20 Y= 20 Z= 20

Default Speed: 9

IK Accel.Tolerance mm/(s*s) (0-50): X= 20 Y= 20 Z= 20

Default Acc.: H

HOME Position: [✓ Robot NEST] Sym Default Sym Current

When connecting: [✓ Sym gets Robot Coords] Robot gets Sym Coords

Gripper Force

Starting Gripping Force: 10 Retained gripping force: 10

Starting Gripping Force Retention Time (in seconds): 3

Figure 10. MoveMaster Settings

3.7. Vision Features

To ease the development of vision processing algorithms, the simulations 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 (Vide for Windows) driver will work as well. For the visual support, we have picked a publicly available OCX control (Xvideo2 by www.cbcsolutions.com), although there are plenty of choices for controlling a camera from within a Visual Basic application.

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.

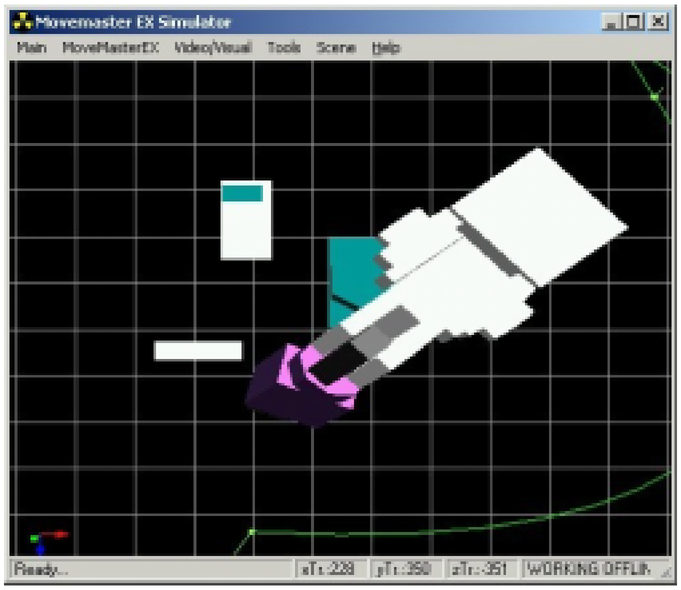


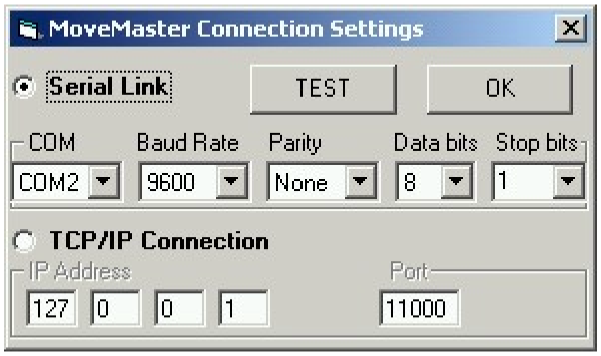
Figure 11. Objects added to the scene

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 acts as the virtual "real" manipulator. The goal could be to grab the virtual objects (for example), and for a better aid the user could also request different views of the server scene for an easier processing. Notice that this would not be easily possible through an actual camera, as cameras can not be dynamically re-positioned unless with the aid of a second or more manipulators. 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 an actual robot.

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).



MoveMaster Connection Settings

(●) Serial Link

COM	Baud Rate	Parity	Data bits	Stop bits
COM2	9600	None	8	1

( ) TCP/IP Connection

IP Address: 127 0 0 1

Port: 11000

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 and once the test is successful the connection can be made and the actual robot will be in synchronization with the CAD model.

A second connection choice is available too, which is TCPIP. 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 is now made to the second simulator, which consequently can be connected to the actual robot (please see next section for more details).

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, a chain of simulation packages being possible (Figure 13).

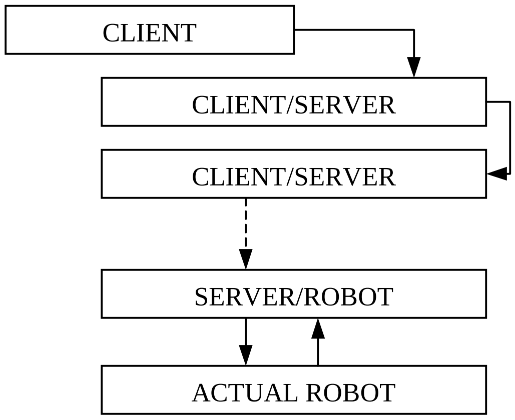
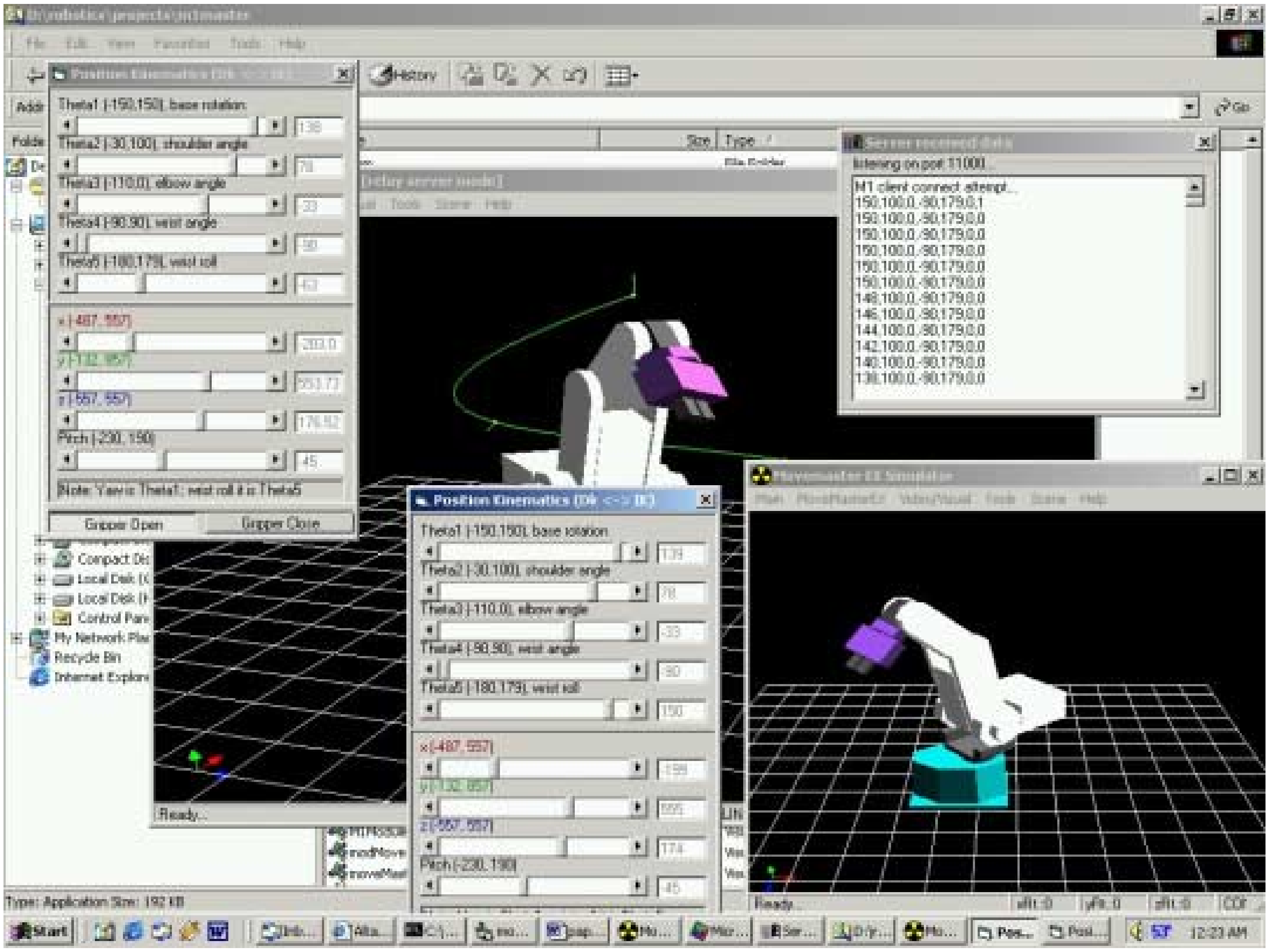


Figure 13 Networking Model

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.



Position Kinematics

Theta1 [-150,150], base rotation 138
Theta2 [-30,100], shoulder angle 78
Theta3 [-110,0], elbow angle -33
Theta4 [-90,90], wrist angle -90
Theta5 [-180,179], wrist roll 43
x [-487, 557] -203.0
y [-132, 957] 553.73
z [-557, 557] 176.52
Pitch [-230, 190] 45
Note: Yaw is Theta1; wrist roll it is Theta5
Gripper Open Gripper Close

Server received data

```
listening on port 11000...
M1 client connect attempt...
150,100,0,-90,179,0,1
150,100,0,-90,179,0,0
150,100,0,-90,179,0,0
150,100,0,-90,179,0,0
150,100,0,-90,179,0,0
150,100,0,-90,179,0,0
148,100,0,-90,179,0,0
146,100,0,-90,179,0,0
144,100,0,-90,179,0,0
142,100,0,-90,179,0,0
140,100,0,-90,179,0,0
138,100,0,-90,179,0,0
```

Position Kinematics (Dk <-> IK)

Theta1 [-150,150], base rotation 139
Theta2 [-30,100], shoulder angle 78
Theta3 [-110,0], elbow angle -33
Theta4 [-90,90], wrist angle -90
Theta5 [-180,179], wrist roll 150
x [-487, 557] -159
y [-132, 957] 555
z [-557, 557] 174
Pitch [-230, 190] 45

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

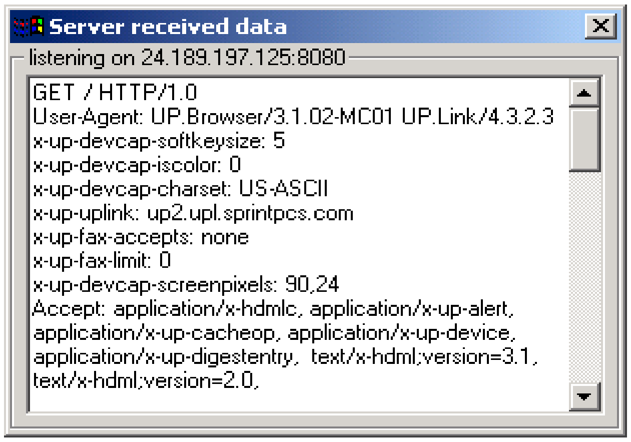
Any of the chained workstations can be controlled as well through the position kinematics interface, in this case overriding all the consequent workstations until the actual robot (if connected). The workstations could also be just left to display the position kinematics interface, which would adjust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

3.10. Controlling the Manipulator Via a Wireless device (Cell Phone)

Wireless control, remote manipulation and distance learning are easily considered among the top interest technological problems with mass appliances. To demonstrate the easiness of turning the control of the manipulator through this software package into a wireless solution, a full cell-phone based control interface was added. The simulator can be turned into the *Web / Cellular Server mode*, which will turn it into a wireless (HDML) server that allows basic control of the manipulator.

For this step, the package was enriched with a basic implementation of a wireless HTTP server that allows the connection from any web enabled cell phone.

Figure 15 shows the server window that allows the visualization of the protocol messages, while the CAD model and (if connected) the actual manipulator will move to various cell phone sent commands.



```
GET / HTTP/1.0
User-Agent: UP.Browser/3.1.02-MC01 UP.Link/4.3.2.3
x-up-devcap-softkeysize: 5
x-up-devcap-iscolor: 0
x-up-devcap-charset: US-ASCII
x-up-uplink: up2.upl.sprintpcs.com
x-up-fax-accepts: none
x-up-fax-limit: 0
x-up-devcap-screenpixels: 90,24
Accept: application/x-hdmlc, application/x-up-alert,
application/x-up-cacheop, application/x-up-device,
application/x-up-digestentry, text/x-hdml;version=3.1,
text/x-hdml;version=2.0,
```

Figure 15. Cell Phone control server interface

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 being sent a small HDML page that allows him 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): The HDML page is simple:

```
<HDML VERSION=3.0 PUBLIC=TRUE TTL=60
MARKABLE=TRUE>
<DISPLAY TITLE="M1 Cell Pad">
Use 1-9 to control the arm<BR>
<A TASK=GO DEST=0.hdml LABEL=0 ACCESSKEY=0>
<A TASK=GO DEST=1.hdml LABEL=1 ACCESSKEY=1>
<A TASK=GO DEST=2.hdml LABEL=2 ACCESSKEY=2>
<A TASK=GO DEST=3.hdml LABEL=3 ACCESSKEY=3>
<A TASK=GO DEST=4.hdml LABEL=4 ACCESSKEY=4>
<A TASK=GO DEST=5.hdml LABEL=5 ACCESSKEY=5>
<A TASK=GO DEST=6.hdml LABEL=6 ACCESSKEY=6>
<A TASK=GO DEST=7.hdml LABEL=7 ACCESSKEY=7>
<A TASK=GO DEST=8.hdml LABEL=8 ACCESSKEY=8>
<A TASK=GO DEST=9.hdml LABEL=9 ACCESSKEY=9>
</DISPLAY></HDML>
```

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. Note that for the implementation of this wireless control feature, a small TCP/IP listener server was implemented as a side application and the communication with a cell phone was tweaked such as the reply from the software package is acceptable. Various websites such as [9] proved useful in building the necessary HDML although for an advanced application there are various books available.

4. Conclusions

In this paper, we present a software model designed to alleviate the 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 also proves unique utilization in high end education institution, by allowing the students to perform a large number of projects involving a certain manipulator without actually purchasing, or purchasing a single or limited number for final demonstration purposes. The simulator could for example be given to by the student through a vacation 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 a distance learning method, especially by setting up a networked chain for all the students in a distance learning class, with one student 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, as the package itself was designed as only a basis for various future applications.

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