VIRTUAL PROTOTYPING FOR ROBOT CONTROLLERS

Mehmet İsmet Can DEDE, *cdede002@fiu.edu* İzmir Institute of Technology, 35430, İzmir, Turkey

ABSTRACT

Production of a new mechanism involves design, manufacturing and testing phases. In order to achieve shorter turn-around times, researchers have studied methods of shortening these phases. Today, structural tests and simulations conducted in virtual environments prior to manufacturing are a part of most of the standard production processes. However, in robot production not only the structural aspect of the mechanism but also its controller is required to be tested. This paper summarizes a versatile method to rapid prototype the robots in virtual environments to conduct controller tests. The verified controllers are then employed in actual robot prototype. The procedure is implemented in a gimbal-based joystick production process.

Keywords: Robot, Virtual Rapid Prototype, Hardware in the Loop Simulation

1. INTRODUCTION

Robot design starts with a Computer-Aided Design (CAD) model, system assembly, and simulation of the system motion. It then progresses to the development of kinematic and dynamic model simulations, structural design, and is completed by designing a controller for the robot. This streamlined approach allows easy reiteration of the design process at any stage; thus, it allows the designer to optimize system parameters as much as possible. The rapid prototyping environment presented in this paper is developed by integrating SolidWorks©, Matlab© and their modules to create a detailed system model for use in simulations and controller development. Although the process is applicable to the design of any mechanical system, robots with their high degrees of freedom are especially suitable for rapid prototyping in virtual environments.

CAD software is a tool to design any physical system either in two- or in three-dimensional space in a virtual environment. This tool is especially useful for designers in the design of multi-degree-of-freedom (DOF) robots to evaluate their performance even before manufacturing them. As professional software packages evolve, more functions are made

available to the design engineers. However, to design a robotic system, usually several software tools are needed.

SolidWorks© is a powerful CAD tool to design system parts and assemblies, and animate the system motion utilizing its animation tool CosmosMotion. For certain types of parallel mechanisms, the software performs kinematic and dynamic analysis while the robot is in motion to calculate various physical quantities of the system such as the forces exerted on joints, positions, velocities, accelerations, and so on. The capability of the software can be extended to robots that use serial architectures by carrying out the kinematic analysis of the robot externally and then importing the data to SolidWorks© for animation purposes.

Another helpful function of the software is also addressed by transferring the robot's data into the Matlab©. The visual representation of the robot is transferred to Matlab© Virtual Reality environment as VRML files. This enables the designer to view the robot in action while the simulation is running.

In order to create simulations, Matlab© introduces the Simulink environment and Simmechanics blocks which can accomplish forward kinematics and dynamics modeling. Control algorithm of the robot can also be developed in this environment. Matlab© also released a new translator to translate SolidWorks© forward kinematics and dynamics information into Matlab© as Simmechanics blocks [1]. The blocks created are explained in the following sections.

After the simulation environment is created, controllers are developed and evaluated for predefined tasks. As data is collected as a result of the simulations virtual representation of the robot in motion provides a better understanding of the robot operation. This completes the stage as called Virtual Rapid Robot Prototyping as referred by Dede and Tosunoglu [2]. The next phase is to integration of the real-time devices to the simulation. This enables to run real-time tests with the actual robots and their controllers.

In the next section a brief introduction is given for animation and analysis environments used in this paper. These tools are then used to design a gimbal-based joystick to be used in a force-reflecting teleoperation application. Created simulation and the results of this simulation are presented as the controllers are tested. Later, integration stage is explained for real-time devices. Final sample test results are then given for the fully operational real-time system.

2. ANIMATION AND ANALYSIS BACKGROUND

Today's CAD tools cannot be considered simple software tools for two- or three-dimensional sketching since their capabilities allow them to be utilized as analyzers as well. The finite

element analysis is probably the most-commonly used tool of many analysis tools that are available for these CAD tools. In this work, the focus is on robot design, its animation, and eventually its kinematic and dynamic simulations with the intent to develop and implement controllers for the designed system. To accomplish this goal, the initial aim is to develop a robot using one specific CAD tool; SolidWorks©, and then animate the robot in the same CAD environment.

SolidWorks© is one of the capable CAD tools currently available besides Pro-Engineer©, AutoCad©, Unigraphics©, I-Deas© and others. Most of these programs have their own animation tools and they can also output VRML files to develop Virtual Reality environments. The significance of SolidWorks is that Matlab© recently introduced a tool to work with SolidWorks© so that physical properties are extracted from the SolidWorks© model and used automatically to create Matlab© Simulink models. The physical properties extracted include mass, inertia, center of gravity, position and orientation, link length and other geometric parameters.

Although most of the CAD tools have mechanism design and animation capabilities, each has different procedures. The procedure described in this section is for SolidWorks© CosmosMotion mechanism design tool.

SolidWorks© is utilized for product design purposes by many companies that range from aerospace and defense to automotive industries. For instance, Alliance Spacesystems, Inc. used SolidWorks© to develop robotic arms for NASA's Mars Exploration Rover (MER) mission [3]. After using the stress and thermal analysis tools to optimize the design, the system is further analyzed by animating the mechanism in the animation module.

The U.S. Army Research Laboratory also develops their designs for military applications in SolidWorks©, and later uses ANSYS and CircuitWorks in the analysis stage [4]. In the automotive industry, Commuter Cars Corporation [5] manufactures world's fastest urban car Tango after they have configured their designs in SolidWorks©.

Currently, many robotics researchers use these CAD tools in their simulation works. For instance, Pap, Xu and Bronlund study kinematic simulations of a robotic human masticatory system using CosmosMotion [6]. Some researchers at Tokyo-based Speecys Corp. developed their humanoid robots and their gaits using the SolidWorks© CosmosMotion environment [7].

Robotics researchers at Florida International University (FIU) also chose to create their gaits using SolidWorks© and CosmosMotion [8]. After creating the parts and assemblies for the robot, they also configured it as a mechanism to animate it for testing new gaits. While

testing the gaits before building the robot, they also analyzed the design for optimization purposes.

In another humanoid study at FIU, researchers developed a kinematic simulation environment for humanoid studies [9]. The walking gaits of the humanoid are first created and then simulated in the program. Later, the source code to be embedded in the microprocessor is created automatically by this program. The main interface window of the program is shown in Figure 1.

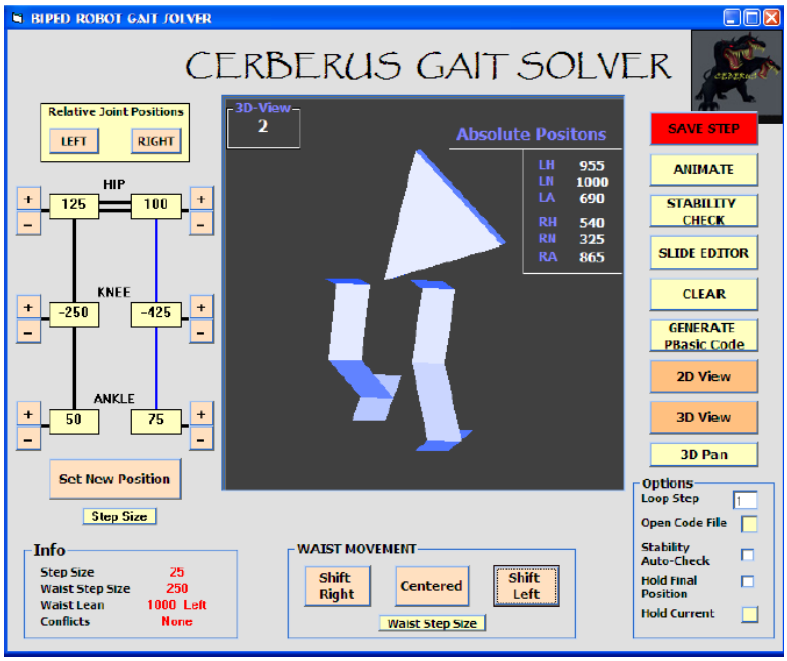

Figure 1. Main interface window of the Cerberus Gait Solver [9]

Nowadays, a number of dynamics and kinematics simulation tools are commercially available such as Adams©, Labview© and Matlab©. Among these, Matlab© is probably the most used tool for robotics researchers as it offers two separate platforms to create simulations. The matrix-based programming language of Matlab©, M-file, makes the software a popular choice for robotics engineers that perform matrix-based solutions for robot kinematics and dynamics. Matlab© also offers a large cluster of subroutines in its block-based simulation environment, Simulink. These subroutines are called blocks and they are wired together to form simulations. Special purpose blocks are collected together at smaller clusters. One of these clusters is the Simmechanics toolbox, where Matlab© offers blocks to configure mechanisms and robotic devices. Simmechanics blocks include a wide range of robot components such as actuators, sensors, bodies and joints.

Today, majority of the robotics simulation studies presented in research articles are created in the Matlab© Simulink environment [1]. Similarly, companies introducing cutting-edge technologies also heavily rely on Matlab© in creating simulations to test their algorithms and products. For instance, Boeing engineers used Matlab© Simulink environment to create a system model and simulation-test the flight control laws for their X-40A Space Maneuver Vehicle (SMV) [10]. Daimler Corp. also designed, tested and implemented the cruise controller for the Mercedes Benz trucks using the toolboxes offered by Matlab© [11]. Similarly, Lockheed Martin Space Systems utilized Matlab© to configure their control designs and automate the development of accurate, real-time spacecraft simulations [12].

3. VIRTUAL PROTOTYPING STAGE

This stage is initiated as all the design processes, with a design objective. Following this, conceptual designs are created and evaluated. Final design concept is selected as a result of this evaluation. The next step is to refine the final design concept for its functionality and ruggedness. In order to fine-tune the design prior to manufacturing a prototype, a CAD model is required. Later, tests are performed on this model to further evaluate the design. This completes the usual path of the design procedure. In contrast, the design of the robot controller starts just at this point. At this stage the model information is required to be transferred to an environment where the controller can be developed and simulation tests can be conducted. Matlab© Simulink is chosen as the development platform for both of these. This is the last phase of the virtual prototyping stage in a robot design. Following subsections explain these phases as applied on a gimbal-based joystick design.

CAD Modeling Phase

The system is designated to be used as the master system in a force-reflecting teleoperation application. The concept is defined as a system that is able to read operator commands in two Degrees-of-Freedom (DOF) and to reflect forces along the same directions. Therefore, servomotors are bedded on each axis. Their encoders are used to read operator demands while they reflect forces that are requested by the slave side of the teleoperation. Ultimately, the operator is to feel controlled resistive forces as he/she works with the joystick. As another feature designated for this system, the joystick is required to have fault tolerance. This is established in joint level by embedding each gimbal between two servomotors. Thus, if one servomotor fails during the operation the other motor takes over the operation.

The detailed design of the joystick is explained in detail in [13]. The final design is modeled in SolidWorks© following the standard part modeling, assembly developing and mechanism configuring procedures as explained in [2].

At this stage of the design, different from the standard machine element structural analysis, mechanism analysis tool of CAD programs can be used. In this design, this tool helped to calculate motion limitation of the joystick along each axis.

Simulation Model Creation Phase

Matlab© models of physical systems can be created in two different environments. A matrix-based programming language called M-files is one Matlab© environment that the programmer can create the model by writing the code line-by-line [1]. Simulink is the other programming platform that Matlab© offers [1]. This platform utilizes a drag-and-drop graphical-user-interface; hence, it is a more user-friendly option for model creation and simulation purposes for the designer.

Simulink coding utilizes previously-defined blocks instead of developing the program line-by-line. As Simulink environment became a major simulation environment for most of the researchers, the platform also evolved in time providing a variety of blocks for modeling purposes. Realatively newer blocks that the robotics researchers are interested in are collected in Simmechanics blockset. This blockset offers link, joint, actuator, and sensor blocks to develop the simulation model of any physical system that uses these components.

Creating Simmechanics Model from SolidWorks© Model: Simmechanics model of a physical system (such as a robot manipulator) can be created from scratch by using the blocks explained previously [1]. The mass, inertia, orientation, and center of gravity information for the links and joints can be input manually when creating the system from scratch. Although this is possible by getting this information from SolidWorks© manually, it is time consuming for complex systems with higher degrees-of-freedom, and prone to introduce errors as link/system parameters are changed during the design phase.

Mathworks recently released a translator to translate SolidWorks© models into the Simmechanics environment. The end result of this translation is a forward kinematics and dynamics model of the system created in SolidWorks©. The model is created using Simmechanics blocks and all the necessary information (mass, inertia, orientation, etc.) is automatically transferred to these blocks. Figure 2 shows the automatically-created Simmechanics model of the joystick as well as. Actuation blocks are also integrated to the model by the designer denoted as “Revolute X Actuation” and “Revolute Y Actuation.”

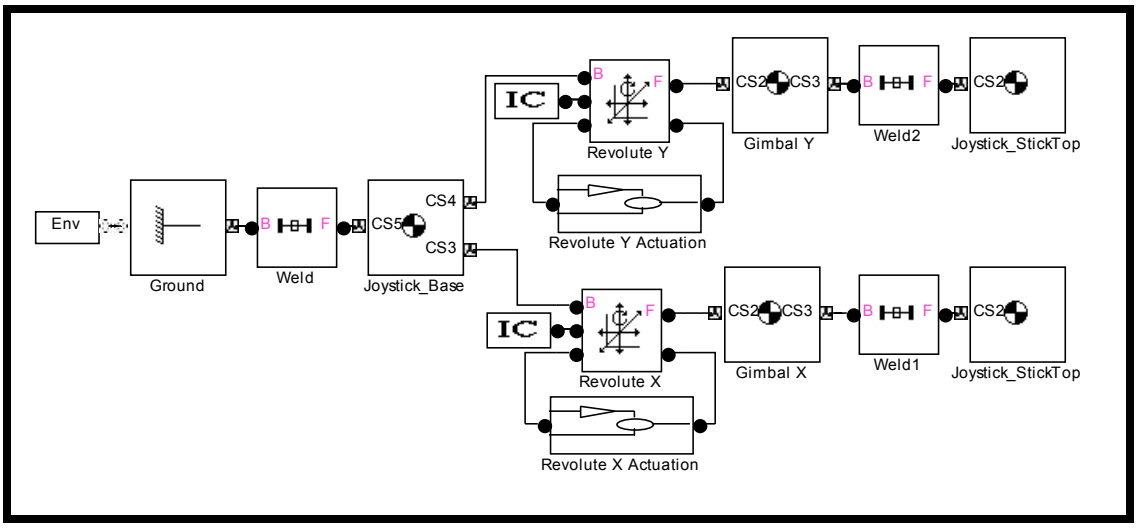

Figure 2. Simmechanics model of the gimbal-based joystcik translated from SolidWorks©

The custom actuator blocks contain actuators and sensors to drive each axis. The controller that uses the sensory information to drive these actuators is also present inside these custom built blocks. As these blocks are completed, the system model is ready to be simulation tested. The actuator and sensor blocks and the formation of the controllers are detailed in [2].

Creating Virtual Reality Models from SolidWorks© Models: Even in the early design stages, it is advantageous to the design engineers to observe the motions of a robotic system. This would provide them a better visual test of the system before manufacturing the robot, and allow them to redesign some of the parts if necessary after inspecting the animations. Matlab© provides this opportunity in two different ways. One is the built-in visualization tool that develops the visual representation of the model automatically. This tool basically draws straight lines from one node to another for each link. It also shows the axis system and Center of Gravity (COG) for each link and joint if these options are activated. All the links and joints are synchronized with the model, which means that as the simulation runs, the visual representation of the model updates itself accordingly.

Another option to create the visual representation of the Simulink model is to use the Virtual Reality Toolbox. This toolbox enables the user to import the 3D CAD models into the Virtual Reality (VR) screen. The motion of the links and joints are then coordinated using V-Realm Builder and “VR Sink” block of Simulink. Once the coordination is complete, the animation of the 3D model is much faster and relatively smoother visually in Matlab© than in the SolidWorks© animation. Figure 3 shows the VR representation of the joystick in the Matlab© VR screen.

Trajectory creation and joint motion creation in Matlab© simulations are also easier with respect to SolidWorks© animation creation. In SolidWorks© animation, an external software module should be used to solve for inverse kinematics of the mechanism to calculate each joint motion. Whereas in Matlab©, inverse kinematics solution can also be carried out within the same simulation.

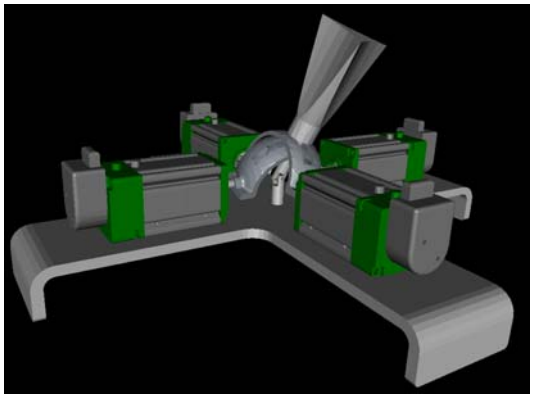

Figure 3. Visual representation of the joystick in VR screen

Integration of Virtual Reality Model with Simmechanics Model: VR model and the Simmechanics model are both created using the SolidWorks© model. These two models are required to be integrated and synchronized using Simulink blocks. As described previously, "VR Sink" block is utilized for this purpose. The inputs to this block are to be created from the Simmechanics block for synchronization purposes.

The rotation centers are created as the origin (0,0,0) in the world axis system. This is not true for the links translated from the SolidWorks© model to VRML. The centers of rotation should be corrected by using the information in the "Body" blocks of the Simmechanics model.

Figure 4 shows the "VR Sink" block parameter window when a two degree-of-freedom manipulator VRML is loaded. As seen in this figure, there are boxes next to some of the parameters that are left blank (unchecked). The value of these parameters can be provided from the Simmechanics blocks continuously during the simulation. For example, the boxes for the center of rotation and the rotation amount about any axis can be checked, and; therefore, controlled during the simulation to rotate the part about that center in the VR screen.

After all the boxes are checked to represent the motion that the Simmechanics blocks are performing, the "VR Sink" block opens ports to interact with the simulation environment. Connecting the necessary inputs from the Simmechanics blocks to these ports, the

integration of visual representation of the mechanism with the Simmechanics blocks is completed.

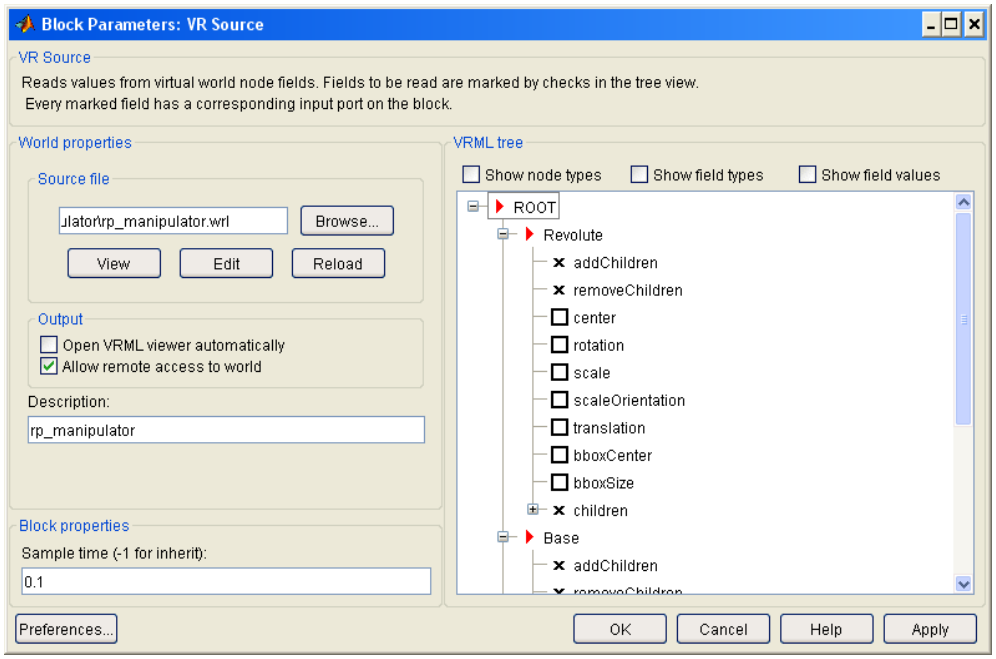

Figure 4. VR sink block parameter window

As a last step before running simulations, simulation solver type and the time-step size need to be specified for customized use. Figure 5 shows the simulation parameter window. Gravity is also modeled in the environment to represent the outer world accurately and it can be switched on or off as required by the simulation scenario.

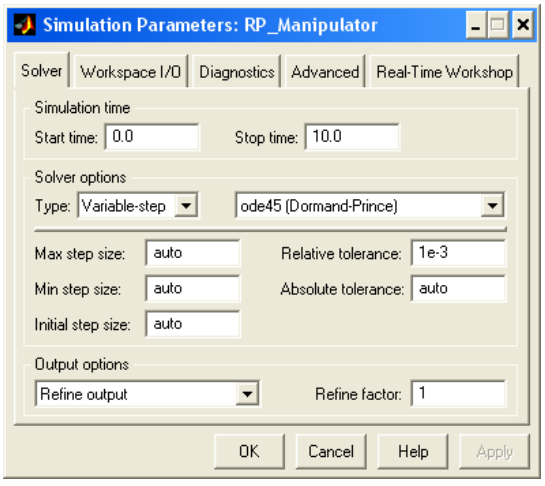

Figure 5. Simulation parameters window

4. HARDWARE IN THE LOOP TESTS

Simulation studies can only provide knowledge of the controller performance up to a level. The modeling uncertainties and errors play a role in divergence from the actual system's performance. The next step before building a prototype is the next phase of tests that integrate real-time hardware in to the simulation. These types of tests are called hardware in the loop tests.

The convenience of working with Matlab© during simulation phases is that this programming platform also allows hardware to be integrated in the simulation. In order to achieve this, the simulation is required to have communication with the hardware. Although Matlab© provides drivers for the most common data acquisition and motion controller cards, some cards need to have a driver to be configure to work with simulations. Nowadays, almost all the cards have their own driver written in C or its variations. The interface or the driver can be created for Matlab© using the driver files written in C programming language. The process of writing a driver is explained for the joystick in [13]. The end result is a Simulink block as shown in Figure 6. The number of inputs and outputs can be adjusted according to the needs as the interface is written.

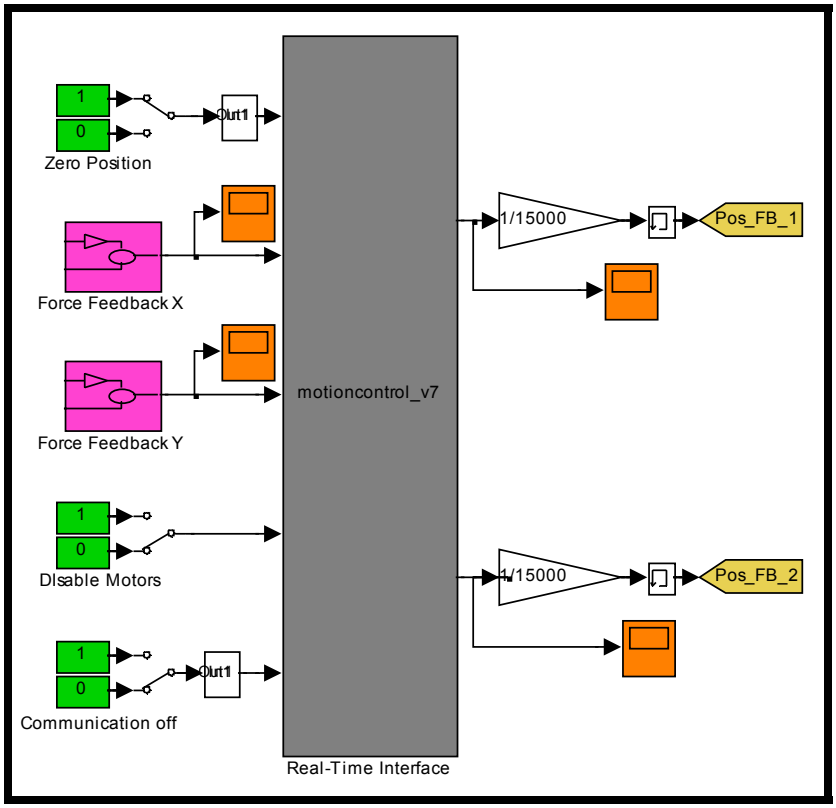

Figure 6. Interface window for the gimbal-based joystick

In this application, the interface is used to read encoder positions from the joystick servomotors in both axes. The first input is spared to reset the encoders to the null position at the start of the manipulation. The second and the third inputs are to send torque commands to the servomotors. Using these inputs motion controller card generates necessary signals to drive the motors. The fourth input is used to disable motors in case of an emergency. The last input is to activate and deactivate communication between the motion controller card and the simulation. As it is observed from Figure 6, the interface is a new Simulink custom block.

The next step is to synchronize the simulation time with the real-time clock. This can be achieved by either using a custom delayer to slow down the simulation process at a constant frequency. This method is not very useful if a smaller step size is required for calculations. At this point, Matlab© offers Real-Time Workshop option where the Simulink blocks are translated into a C code and runs real-time at a given sampling rate (has speed limitations due to the machine it uses). This rate is usually 1 kHz for ordinary robotics applications. The created can easily work at that sampling rate either in the Real-Time Windows Target or the xPC Target that act as targets of Real-Time Workshop. The Real-Time Windows Target creates the code to run in a PC that has Windows based operating system. Whereas, xPC Target does not require an operation system to run its code. The main computer is connected to the other PC that runs xPC Code and the interaction with the hardware in the loop simulation is achieved by this way.

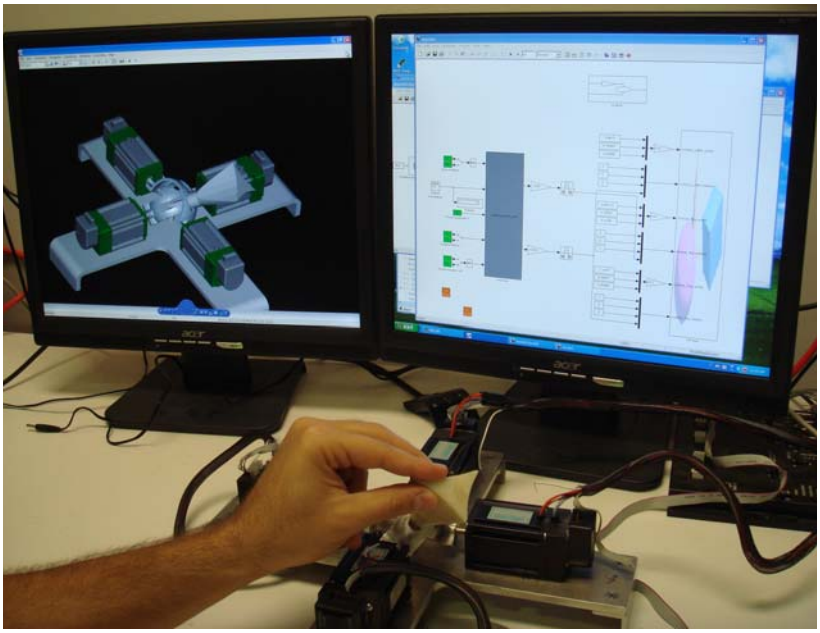

Figure 7. Identical master-slave teleoperation hardware in the loop test setup

In the application that is considered in this paper, Real-Time Windows Target is used to create the real-time code. After manufacturing a prototype of the gimbal-based joystick it is used in the identical limited-workspace teleoperation tests. The virtual replica of the joystick is used as the slave manipulator in these tests. The experimental setup is presented in Figure 7.

Therefore, teleoperation controllers are evaluated in the hardware in the loop simulation tests. Results of the teleoperation controller test at a constant time delay is presented in the following figures [14]. Two variations of a teleoperation controller called wave variable technique are evaluated on the same system. The position tracking performances are investigated in the Figures 8 and 9.

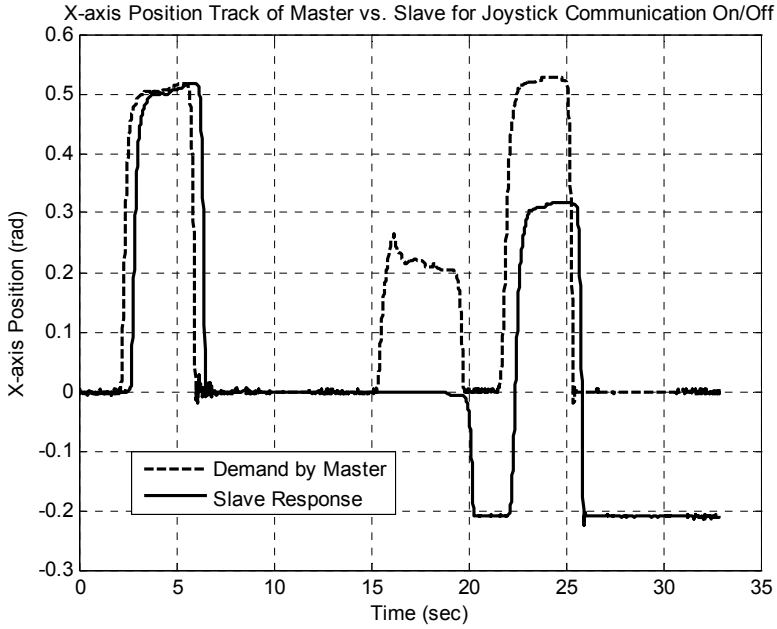

Figure 8. Position tracking performance of the identical master-slave teleoperation using the customary wave variable technique

The hardware in the loop simulation test procedure explained in this section can also be applied to custom simulator creation [15]. Figure 10 shows another teleoperation system where at the instant only the master system is integrated to the system and the mobile robot (WiRobot DRK8000) is running in a virtual environment.

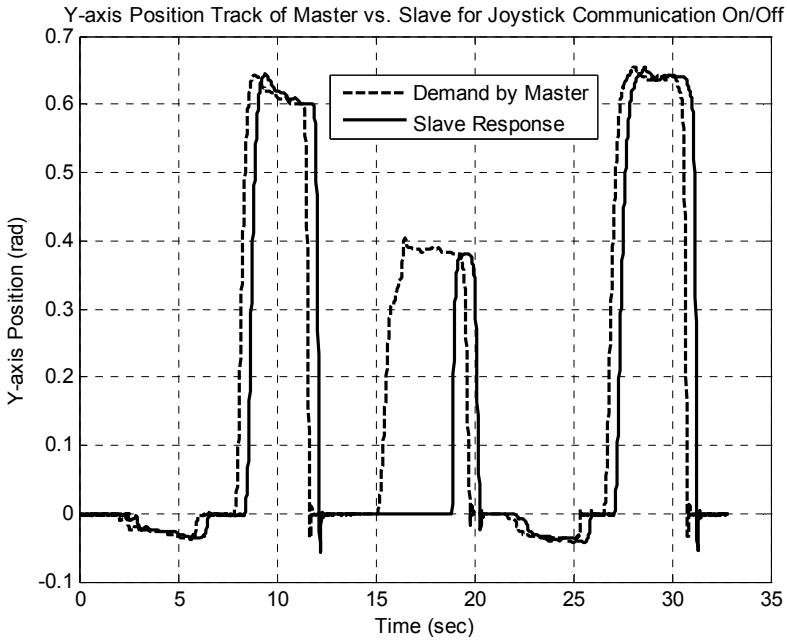

Figure 9. Position tracking performance of the identical master-slave teleoperation using the wave variable technique with position feedforward component

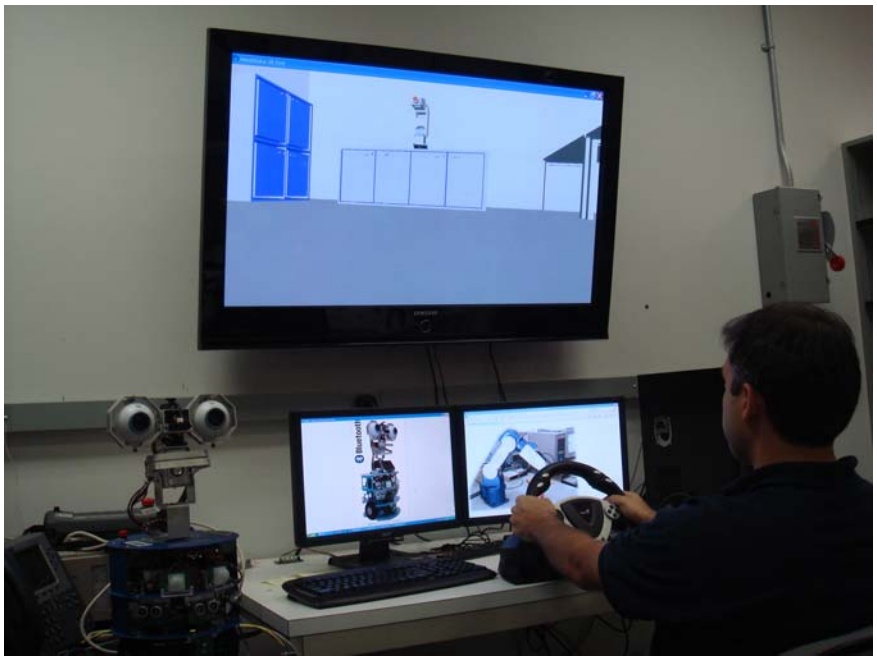

Figure 10. Operator driving WiRobot DRK8000 in VR screen of the Robotics and Automation Laboratory at Florida International University

The next step is to integrate the mobile robot to the system. After the integration process, the production code is generated by Matlab© and this code is embedded to a controller card that works with data acquisition and motion controller cards.

5. CONCLUSION

Robot production is a tedious process that involves a controller design as well as mechanical part and mechanism designs. Numerical simulation techniques are widely used in the industry while prototyping a new product. This procedure saves time because tests are conducted even before building a prototype. In this paper, the same idea for designing parts is applied to building controllers for robots or any mechatronics application.

First, a modeling phase is explained to build a virtual prototype of a robot in CAD environment. Then the virtual prototype is transferred to a simulation environment, Matlab©, as the system model. As a result of this transfer not only the mechanical and mechanism properties but also the virtual information of the robot is translated into Matlab© Simulink programming platform. After the system model is created in this platform, controllers are also built and simulation tests are conducted to evaluate them.

The next step is defined as the hardware in the loop tests. In these tests the hardware is integrated in the simulation to further evaluate the controller performances. An example system that was created using this technique is introduced. The benefit of using this simulation platform, Matlab©, is that the simulation is not modified but only an interface is developed for hardware integration. Thus, the testing process is accelerated.

As the simulations are completed and the controller performances are evaluated, the final step is creating a production code to embed in the controller card of the robot. This is also generated by Matlab© from the controller developed as a result of the simulation tests. Ultimately, the procedure described in this paper saves time and money in developing new robot controllers and robots. The range of savings as a result of using this method with respect to the customary methods of robot controller design is a future research area for this study.

REFERENCES

1. **Matlab©, The MathWorks, Inc.** Retrieved September 26, 2007 from http://www.mathworks.com.

2. **Dede, M.I.C., and Tosunoglu, S. (2006**), “Virtual Rapid Robot Prototyping,” ASME Early Career Technical Journal, Vol.5, No.1, pp 7.1-7.8.

3. **Alliance Spacesystems, LLC.** Retrieved September 26, 2007 from http://www.alliancespacesystems.com.

4. **SolidWorks Corporation.** Retrieved September 26, 2007 from http://www.solidworks.com.

5. **Commuter Cars Cororation.** Retrieved September 26, 2007 from http://www.commutercars.com.

6. **Pap, J-S., Xu, W., and Bronlund, J. (2005),** "A robotic human masticatory system: kinematics simulations," International Journal of Intelligent Systems Technologies and Applications, Vol.1, No.1/2, pp 3-17.

7. **Omura, Y. (2006**), "Robotic software simulator: aids robotic development (Real motion system software)," Design News, Vol. 61, No.4, pp 56-58.

8. **Madadi, V., Dede, M.I.C., and Tosunoglu, S. (2007),** "Gait Development for the Tyrol Biped Robot," ASME Early Career Technical Journal, Vol.6, No.1, pp 7.1-7.8.

9. **Dede, M.I.C., Nasser, S., Ye, S., and Tosunoglu, S. (2006),** "Cerberus: Development of a Humanoid Robot," ASME Early Career Technical Journal, Vol.5, No.1, pp 8.1-8.8.

10. **Boeing.** Retrieved September 26, 2007 from http://www.boeing.com.

11. **Daimler AG.** Retrieved September 26, 2007 from http://www.daimler.com.

12. **Lockheed Martin Space Systems Company**. Retrieved September 26, 2007 from http://www.lockheedmartin.com/ssc.

13. **Dede, M.I.C., and Tosunoglu, S. (2006),** "Development of a Real-Time Force-Reflecting Teleoperation System Based on Matlab© Simulations," Proceedings of the 19th Florida Conference on Recent Advances in Robotics.

14. **Dede M.I.C., and Tosunoglu, S. (2007),** "Modification of the Wave Variable Technique for Teleoperation Systems Experiencing Communication Loss," Proceedings of the 7th IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp 380-385.

15. **Dede, M.I.C., and Tosunoglu, S. (2007),** "Development of a Virtual Haptic Laboratory," ASME Early Career Technical Journal, Vol.6, No.1, pp 2.1-2.8.