

VIRTUAL RAPID ROBOT PROTOTYPING

Mehmet Ismet Can Dede, and Sabri Tosunoglu
Florida International University
Department of Mechanical Engineering
10555 West Flagler Street
Miami, Florida 33174

ABSTRACT

Competition in marketplace requires manufacturers to develop their products in shorter turn-around times. This encourages engineers and researchers to devise methodologies to respond to this market-driven requirement. Rapid prototyping is one solution to produce a design in minimum amount of time. In this paper we focus on rapid prototyping of robots. For this purpose, development of the 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, 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 work is developed by integrating the use of SolidWorks®, Matlab® and a number of their modules, and demonstrated on an RP manipulator. 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.

INTRODUCTION

Computer aided design (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 test the performance of the robots 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. This work addresses the integration of two software packages; SolidWorks® and Matlab®, to design robotic systems.

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 its own 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. In this work, the capability of the software is 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. There are other kinematics solution packages that can be used in mechanism design and one example is SAM® software [8].

Another helpful function of the software is also addressed by transferring the robot's data into the Matlab® Virtual Reality environment as VRML files. This enables the designer to also design the robot's controller using the CAD files in addition to the structural design of the robot. The present work aims to demonstrate the integrated use of these software packages on parallel as well as serial robot systems.

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® recently released a new translator to translate SolidWorks® model information into Matlab® as Simmechanics blocks [4]. The blocks created are briefly explained in this paper.

Final sections of the paper are dedicated to describe the integration of Virtual Reality model with Simmechanics model. Robot prototyping examples utilizing SolidWorks® and Matlab® are also provided.

SOLIDWORKS® MODELING AND ANIMATION

Today's CAD tools cannot be considered simple software tools for two- or three-dimensional sketching anymore since their capabilities allow them to be utilized as analyzers as well. Finite element analysis is probably the most-commonly used tool of many analysis tools that are available with these CAD tools. In this paper, we focus on mechanism design, its animation, and eventually its kinematic and dynamic simulations with the intent to develop a controller for the designed system. To accomplish this goal, the initial aim is to develop a robot using one specific CAD tool; SolidWorks®, develop the joints and then actuate them in the CAD environment.

Although most of the CAD tools have mechanism design and animation capabilities, each of them has

different procedures. The procedure described in this paper is for SolidWorks[®] CosmosMotion mechanism design tool.

Currently, many 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 [1]. Another example is the use of these tools by NASCAR race teams as Teague and Lang reports [2].

Some researchers develop their humanoid robots and their gaits using the SolidWorks[®] CosmosMotion environment. One example is Tokyo-based Speecys Corp. Omura reports: “Demonstrated through a prototype of the SPC-003 robot, the software uses a combination of SolidWorks[®], 3D CAD and COSMOSMotion, and a mechanical analysis simulator. With the software platform, a design engineer can verify robotic motion before prototyping a robot.” [3]. The figure below shows the actual humanoid and the SolidWorks[®]/CosmosMotion windows utilized to develop gaits for the humanoid.

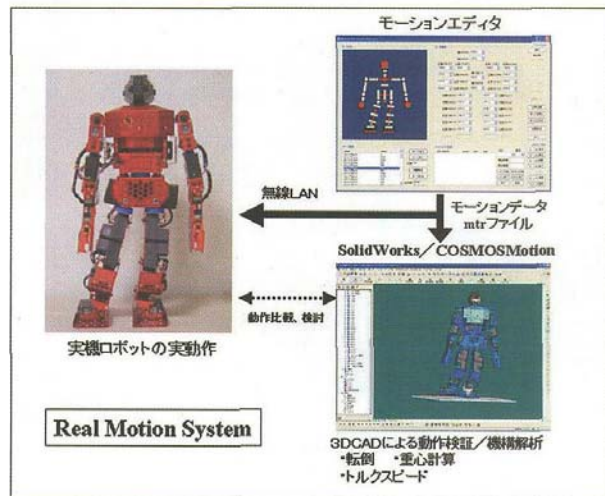


Figure 1. Humanoid gait development using SolidWorks[®]/CosmosMotion

Part creation and assembly development are considered quite straightforward for today’s designers. The challenge is now on quickly simulating and animating these mechanisms. The next sub-section describes the creation of a mechanism animation in SolidWorks[®].

SolidWorks[®] Mechanism Creation and Analysis

Assembly creation process is very important if the mechanism is to be animated. The mating process should be accomplished carefully. Parts that are screwed, welded or somehow fixed to each other should not have a motion relative to each other. If there is any kind of joint placed between two parts, it is a definite sign that the parts are not fixed relative to each other. On the other hand, when creating a joint, two parts to form the joint should have a motion, either translational or rotational with respect to each other.

CosmosMotion lets the user to input motion to the joint; thus, it can create an animation of the mechanism. The sample mechanism or robot arm we are using to demonstrate the procedure in this paper is an RP manipulator. If the mating process is accomplished correctly, the mechanism should have one revolute (R) and one prismatic (P) joint. Figure 2 shows the motion-building window of CosmosMotion. As it can be observed from the figure, definitions of the mating conditions result in the creation of one revolute and one prismatic joint.

It can also be observed from Figure 2 that the motion for the “Revolute” joint is described as a constant speed of 360 deg/sec or in other words 60 rpm. There are different ways of describing the joint motions. Probably the most efficient way is to calculate the joint trajectories as a result of the inverse kinematics solutions for the given end-effector trajectory and then to input the joint motion as a function of time for each joint.

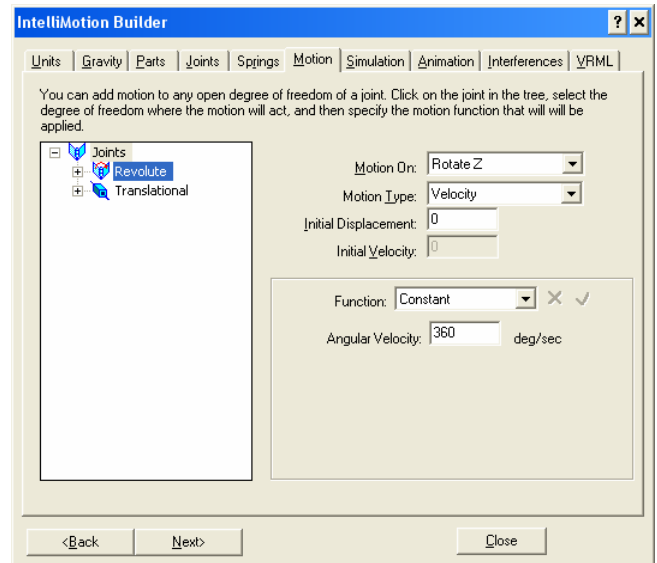


Figure 2. SolidWorks[®]/CosmosMotion motion builder

After simulating the mechanism for the pre-described joint motions, the animation created can also be saved as a movie file. Figure 3 shows the window to save the animation movie. “Create Animation” button on the window saves the animation to the designated location on the hard disk.

Following the animation creation, various analyses on the system can be carried out. One such analysis includes path following of any point on the mechanism and its trajectory on each world axis. Figure 4 shows the main window of SolidWorks[®] with the path drawn for a certain point of the mechanism and the trajectory of that point along x, y, and z-axis. The red line on the trajectory plot example shown in Figure 5 can be moved forward and backward synchronized with the mechanisms posture at that time interval.

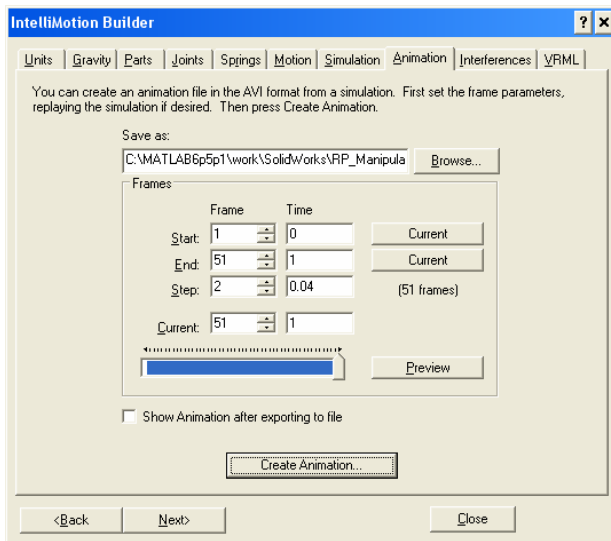


Figure 3. SolidWorks[®]/CosmosMotion animation window

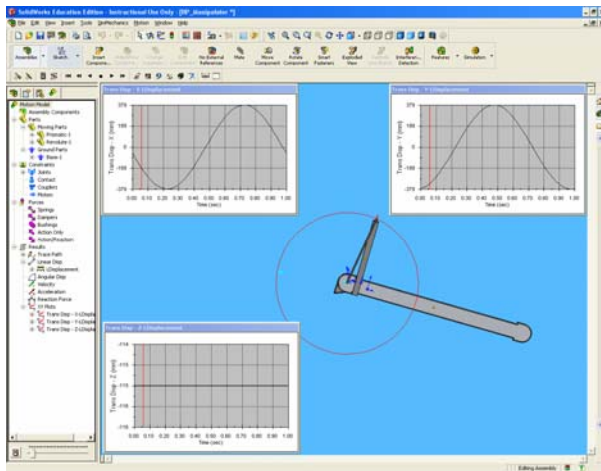


Figure 4. SolidWorks[®] main window with analysis results

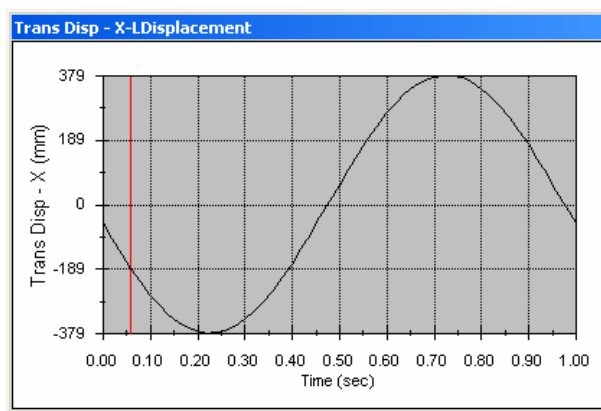


Figure 5. Trajectory of a point on the mechanism along x-axis

MATLAB[®] MODELING USING SIMULINK BLOCKS

Matlab[®] models of physical systems can be created in two different environments that Matlab[®] provides. A matrix based programming language called M-files is

one Matlab[®] environment that the programmer can create the model by line-by-line M-file code writing [5]. Simulink is the other programming platform that Matlab[®] offers [6]. 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. The new branch of blocks that the robotics researchers are interested in is collected in Simmechanics blockset. This blockset offers link, joint, actuator and sensor blocks to develop the simulation model of any physical system. Some of these blocks are introduced in Table 1.

Table 1. Matlab[®] Simmechanics blocks

	“Ground” block grounds one side of a joint block to a fixed location in the World coordinate system
	“Joint Initial Condition” block sets the initial linear/angular position and velocity of some or all of the primitives in a joint block.
	“Joint Actuator” block actuates a joint block primitive with generalized force/torque or linear/angular position, velocity, and acceleration motion signals.
	“Joint Sensor” block measures linear/angular position, velocity, acceleration, computed force/torque and/or reaction force/torque of a joint primitive.
	“Revolute” joint block represents one rotational degree of freedom. It can be driven by the “Joint Actuator” block and its motion can be measured by the “Joint Sensor” block if the blocks are attached to this block.
	“Prismatic” joint block represents one translational degree of freedom. It can be driven by the “Joint Actuator” block and its motion can be measured by the “Joint Sensor” block if the blocks are attached to this block.
	“Body” block represents a user-defined rigid body. “Body” block is defined by mass, inertia tensor and coordinate origins.
	“Body Sensor” block measures linear/angular position, velocity, and/or acceleration of a “Body” block with respect to a specified coordinate system.

The simulation solver type and the step size can be adjusted for customized use. Gravity is also modeled in the environment to represent the outer world accurately. A programmer can also use the control algorithm developed using the Simulink blocks to control the real-time system. This is another feature that Matlab® provides and it creates C codes from the simulation to run in real-time to operate the hardware. This feature also saves time for the control design engineers in real-time tests.

Before running the simulation, the simulation parameters have to be set. Figure 6 shows the simulation parameter window. The duration of the simulation, ordinary differential equation solver type and step size type and amount can be set using this window.

Another tab that can be observed from Figure 6 is the Real-Time Workshop tab. Using this tab, the simulation created can be translated into C code to run the code in real-time.

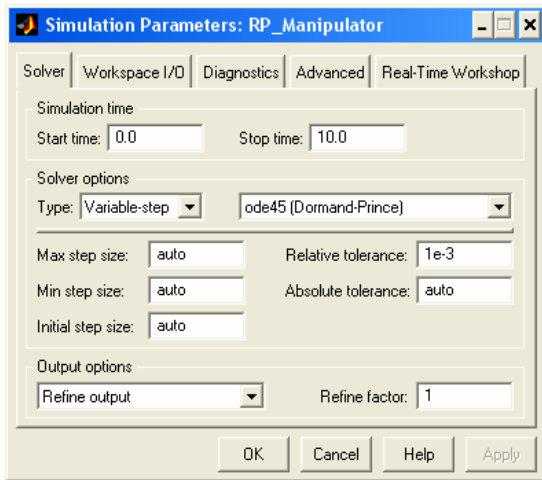


Figure 6. Simulation parameters window

CREATING VIRTUAL REALITY MODEL FROM SOLIDWORKS® MODEL

Probably most of the robotics engineers would like to see their robots in motion even during the initial design stage and especially in simulation phases. 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. Figure 7 shows the visual representation of a 6-DOF robot manipulator drawn by using the visualization tool. 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.

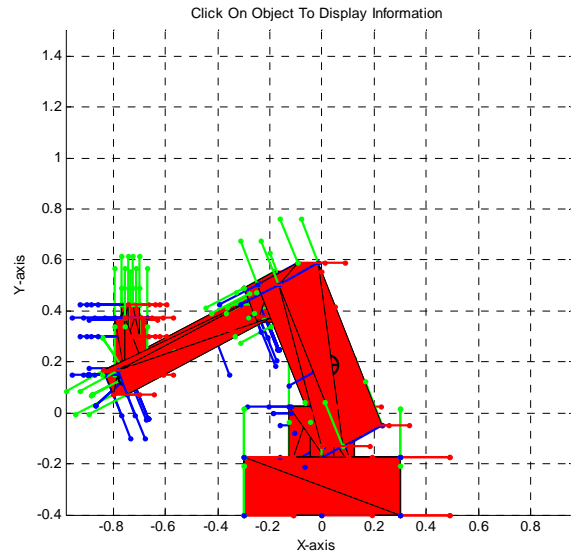


Figure 7. Visual representation by visualization tool

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 smoother visually in Matlab® than in the SolidWorks® animation.

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.

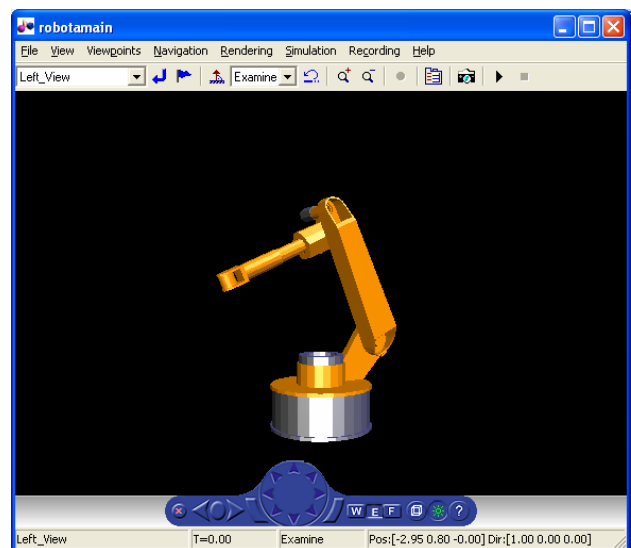


Figure 8. Visual representation by VR screen

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 [7]. 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 9 shows the automatically-created Simmechanics model of an RP robot manipulator.

The Simmechanics blocks that are created as a result of SolidWorks® to Simmechanics translation are illustrated in the second window of Figure 9. The base part, “Base,” of the manipulator is fixed to the base, which is described as a zero-DOF with respect to the base (ground). The base is then connected to the revolute link, “Revolute,” via a revolute joint, which is described as one rotational DOF in between the base and the revolute link. The revolute link is then connected to the end-effector, “Prismatic,” via a prismatic joint, which is described by a translational motion between the revolute link and the end-effector.

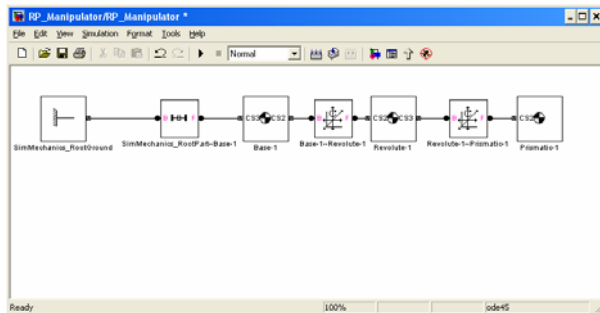


Figure 9. Simmechanics model of an RP manipulator translated from SolidWorks®

Figure 10 shows the “Body” block for the revolute link where the mass, inertia and length information is translated from SolidWorks®. Figure 11 shows the translated revolute joint and the rotation axis information. “Sensor/Actuator” port is to be increased to connect sensors and actuators for simulation purposes in later stages.

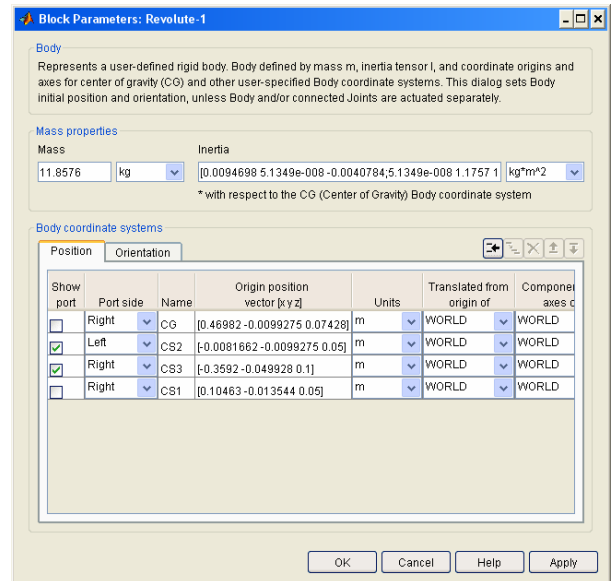


Figure 10. Revolute body block parameters translated from SolidWorks® model

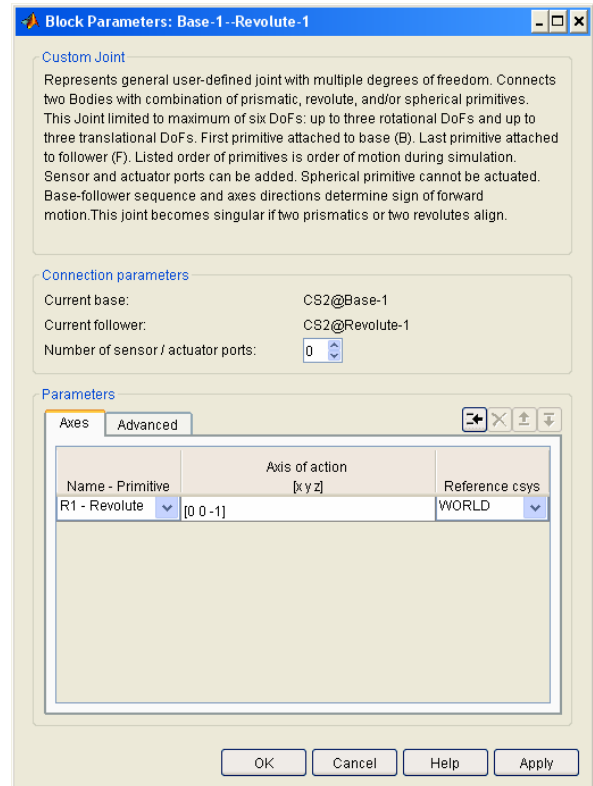


Figure 11. Revolute joint block parameters translated from SolidWorks® model

INTEGRATION OF VIRTUAL REALITY MODEL WITH SIMMECHANICS MODEL

Virtual Reality 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 (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.

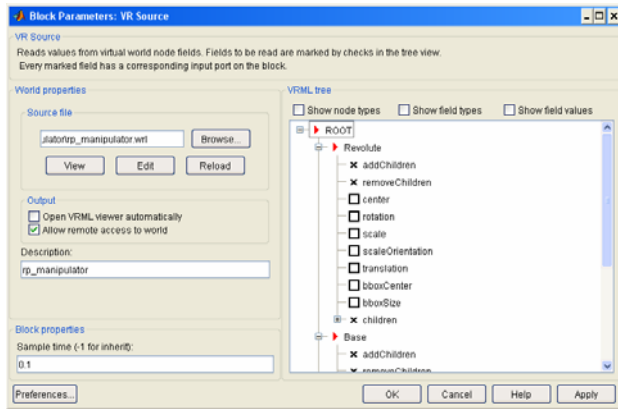


Figure 12. VR sink block parameter window

Figure 12 shows the “VR Sink” block parameter window when the RP manipulator VRML is loaded. As seen in this figure, there are boxes next to some of the parameters that are blank (unchecked). The value of these parameters can be provided from the Simmechanics blocks continuously during the simulation. For example, the center of rotation and the rotation amount about any axis boxes can be checked, and; therefore, controlled during the simulation to rotate the part about that center in the VR screen.

After all the boxes necessary to represent the motion that the Simmechanics blocks are performing checked, “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.

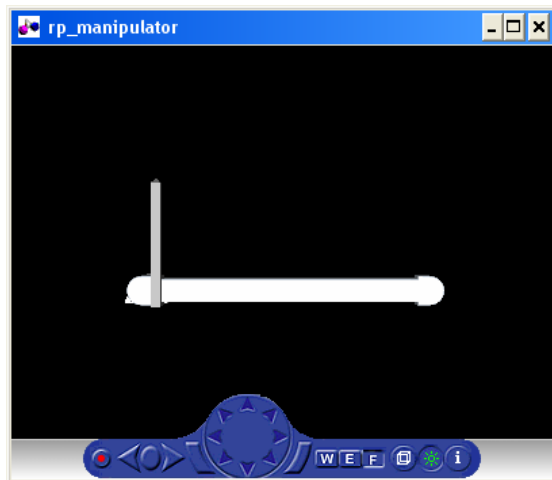


Figure 13. Virtual reality representation of RP manipulator translated from SolidWorks[®]

SIMULATION RESULTS

An example simulation is carried out for the serial RP robot manipulator that is introduced previously. The sample RP manipulator is developed as a planar robot. The simulation is carried out in two different phases. The first phase is the kinematics simulations. In kinematics simulations, dynamics of the system is disregarded. The reason to start with kinematics simulations is to verify the inverse kinematics solutions as well as the task created. The sample task requires the end-effector to follow a straight-line trajectory.

Figure 14 shows the main window of the simulation. In this window integration of the Simmechanics outputs with the VR screen can be observed from the connections made to “VR Sink”. The end-effectors trajectory is also followed by another scope that appears as the “X-Y Graph” on the bottom right side of the window. Simmechanics blocks are hidden inside the block named “RP_Manipulator” on the top right corner of this window.

The Simmechanics blocks created as a result of the translation from SolidWorks[®] were presented previously. New Simulink blocks are added to these blocks to actuate the mechanism as shown in Figure 15.

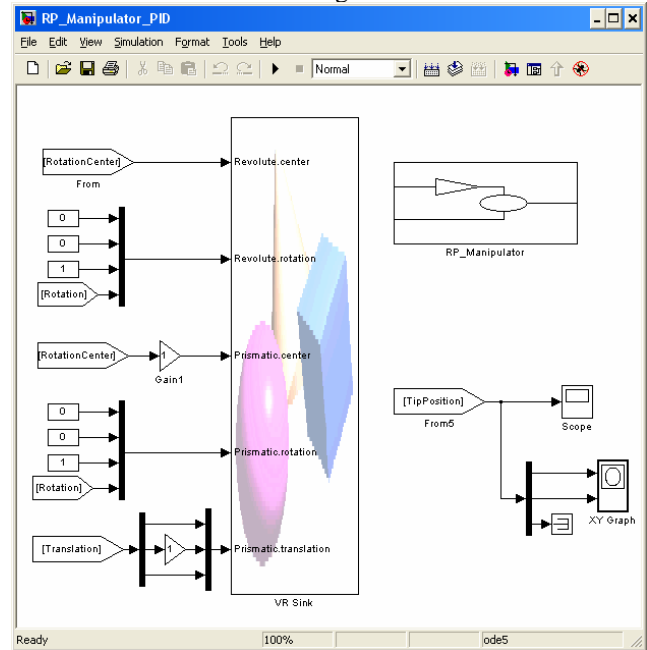


Figure 14. Simulation main window

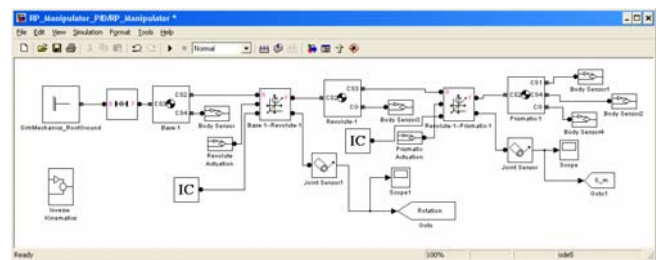


Figure 15. Modified Simmechanics blocks for RP Manipulator

One new block added is the inverse kinematics solution. This block is formed using simple Simulink blocks to solve for joint motion for a given end-effector motion. Figure 16 shows the inverse kinematics calculations for this manipulator.

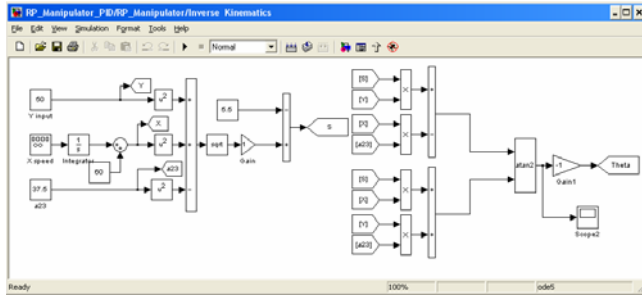


Figure 16. Inverse kinematics calculations

Joints actuation for the kinematics solution and dynamics solution differ. In kinematics solution joints are actuated with the provided joint trajectories as a function of position, velocity and acceleration. Dynamic effects are neglected for this solution. Figure 17 shows a sample joint actuation for the kinematics solution.

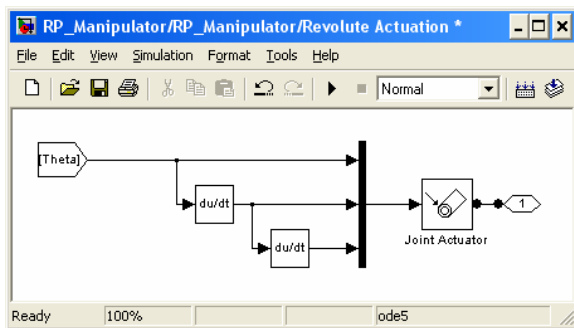


Figure 17. Joint actuation for kinematics solution

As it can be observed there is no control law over the motion fed into the actuator. Therefore in the kinematics solution the dynamic effects do not have influence over the motion and the result comes out to be a perfect straight-line path followed by the end-effector as shown in Figure 18.

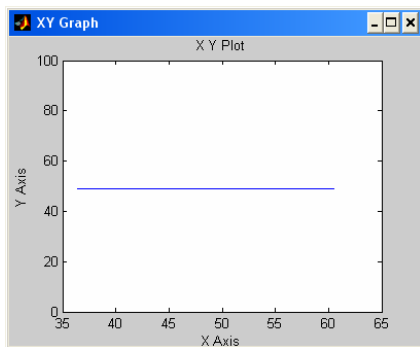


Figure 18. End-effector's path for the kinematics simulation

The joint actuation blocks are modified for the simulations taking into account the dynamics of the mechanism. As it can be observed from Figure 19, a PID controller is used as the control law for the revolute joint actuation. In addition, compensation for gravitational effects is added to the joint torque calculation. Also "Initial Condition" blocks are used to start the joint actuation close to the initial position. This way transition state overshoots are minimized for each joint.

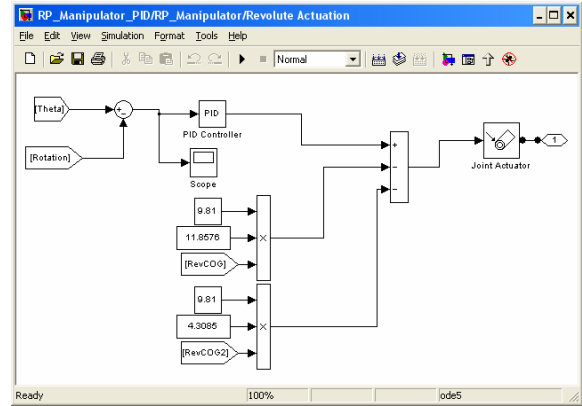


Figure 19. Joint actuation for dynamics solution

A transition state is expected for every mechanism at the start of manipulation. This transition state can be clearly observed from the path of the end-effector shown in Figure 20.

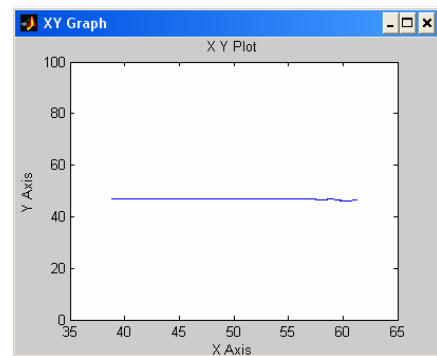


Figure 20. End-effector's path for the dynamics simulation

The transition state can also be observed from the error signal created to be fed into the PID controller for each joint. Figure 21 shows the error signal plot for the revolute joint. Looking at these graphs during the manipulation, the designer can set the control parameters or change the control algorithm. The specific task was performed at a relatively low speed. Therefore, the effect of the centrifugal and Coriolis force terms may be neglected in the control design. For faster manipulation speeds, these effects cannot be neglected. Thus, non-linearity cancellation can be incorporated by employing computed-torque control.

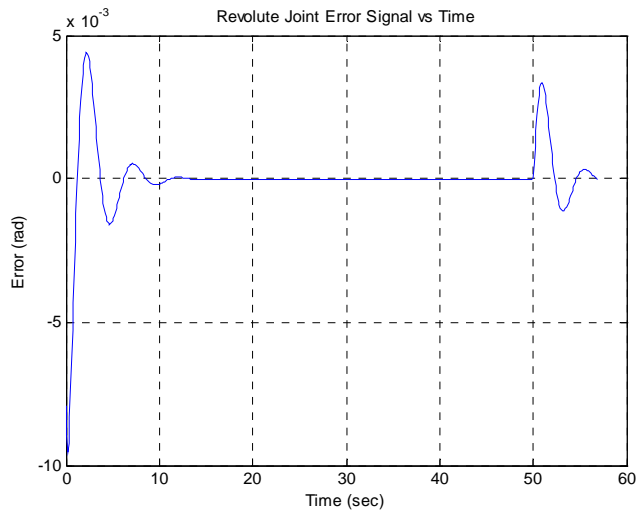


Figure 21. Revolute joint error signal

CONCLUSIONS

Today most of the manufacturers utilize CAD models to design and build their systems. Pressure on manufacturers for quick turn-around times encourages new methods for rapid prototyping techniques. Hence, integration of CAD models into other professional packages facilitates the design process.

To streamline the design process for robot design, this work addressed the use of SolidWorks[®] and Matlab[®] environments. Such a rapid prototyping approach allows design engineers to simulate the robot motions, check its workspace, design a suitable controller, test its performance, and modify the design at any stage of the design process before even building a prototype. Smooth and accurate information flow between various modules is seen as a crucial step to save time as well as prevent errors.

The rapid prototyping approach is seen especially beneficial for robots, which enjoy a high number of

actuated axes; therefore, a high number of actuators need to be controlled in unison. The process is demonstrated in terms of a serial RP manipulator although the methodology is also valid for parallel mechanisms.

Future work will address more robust integration between the software modules and apply the process to parallel as well as serial mechanisms.

REFERENCES

- [1] J-S. Pap, W. Xu, and J. Bronlund. "A robotic human masticatory system: kinematics simulations," *International Journal of Intelligent Systems Technologies and Applications*, 1.1/2, July 5, 2005.
- [2] P. E. Teague, and M. M. Lang, "Fast-track design: this NASCAR Team improves performance by analyzing vendors' models--not just the parts," *Design News*, Volume 59, Issue 7, May 17, 2004.
- [3] Y. Omura, "Robotic software simulator: aids robotic development (Real motion system software)," *Design News*, Volume 61, Issue 4, March 20, 2006.
- [4] Matlab[®] CAD Translator
<http://www.mathworks.com/access/helpdesk/help/toolbox/phymod/mech/f3-8118.html> accessed August 2006.
- [5] Matlab[®] M-file programming
http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_prog/f10-60352.html accessed August 2006.
- [6] Matlab[®] Simulink
<http://www.mathworks.com/products/simulink/?BB=1> accessed August 2006.
- [7] Matlab[®] Simmechanics Block Library
<http://www.mathworks.com/access/helpdesk/help/toolbox/phymod/mech/f5-7409.html> accessed August 2006.
- [8] SAM software
http://www.artas.nl/Xmain_us.htm accessed August 2006.