2008

An Integrated Simulation Design With Three-Dimensional Motions and a Hydraulic Stewart Simulator

Cheng Lin
Old Dominion University, clin@odu.edu

Gene Hou
Old Dominion University, ghou@odu.edu

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Hector Garcia
Old Dominion University, hgarcia@odu.edu

Original Publication Citation

Lin, C., Hou, G., Shen, Y., & Garcia, H. (2008). *An integrated simulation design with three-dimensional motions and a hydraulic Stewart simulator.* Paper presented at the 2008 ASEE Annual Conference & Exposition, Pittsburgh, Pennsylvania.

AC 2008-458: AN INTEGRATED SIMULATION DESIGN WITH THREE-DIMENSIONAL MOTIONS AND A HYDRAULIC STEWART SIMULATOR

Cheng Lin, Old Dominion University

Gene Hou, Old Dominion University

Yuzhong Shen, Old Dominion University

Hector Garcia, Old Dominion University

Redesign of the Stewart Flight Simulator Platform with Real-Time Sensing and Actuation

Abstract

This paper presents an integrated design process and tests of a Stewart simulator with a virtual visualization tool, which uses Virtools to create and generate three-dimensional motions. An inverse kinematic algorithm is written to convert each visualized motion to the displacements of six cylinders in a Stewart motion simulator. Information of the displacements is then transferred through the User Datagram Protocol (UDP) to a personal computer which has the LabVIEW software. An NI USB-6251 data acquisition device is applied to interact with the LabVIEW program and the Stewart hydraulic simulator. The approach presented in this paper to function an old Stewart hydraulic simulator can also be applied to other simulators.

1. Introduction

The major objective of this project is to develop a prototype system which can simulate the motion of a water craft when it is driven through different waves and obstacles. This physical simulator will also facilitate a virtual and interactive environment to support for the future design and studies of water craft dynamics, sea keeping and human-craft interaction. Figure 1 shows the flow chart of the project. The main components include a wave model, a simulated water craft with an interactive control tool, a joystick, a Stewart platform, and a data acquisition device. The three-dimensional virtual visualization environment is created by using the software of Virtools[1] and a joystick is applied to maneuver the simulated water craft. Information of the craft motion is then transferred to a Stewart Stewart Platform[2] manipulator through an inverse kinematics, a LabVIEW[3] program, and a data acquisition device, which is an NI USB 6251[4]. Communication between the Virtools PC and the LabVIEW PC is achieved by using the software of User Datagram Protocol (UDP) [5]. Development of this project was demonstrated in one of the labs for the class of MET 415, Introduction to Robotics, at the Department of Engineering Technology at Old Dominion University. This project includes software programming, inverse kinematics, feedback-signal sensing, solenoid activation, electrical-circuit connections, and data communications. Descriptions of the main components are described in the following sessions.

2. Virtools

Virtools is a 3D application/simulation development environment that bundles together models and code into files that can be played by the Virtools Web Player. The simulation can be built inside the Virtools development environment, pulling together code, data, and models into a playable simulation. It's a robust and complete 3D engine (web-playable or standalone) with built-in physics, Artificial Intelligence (AI), and server communications[1].

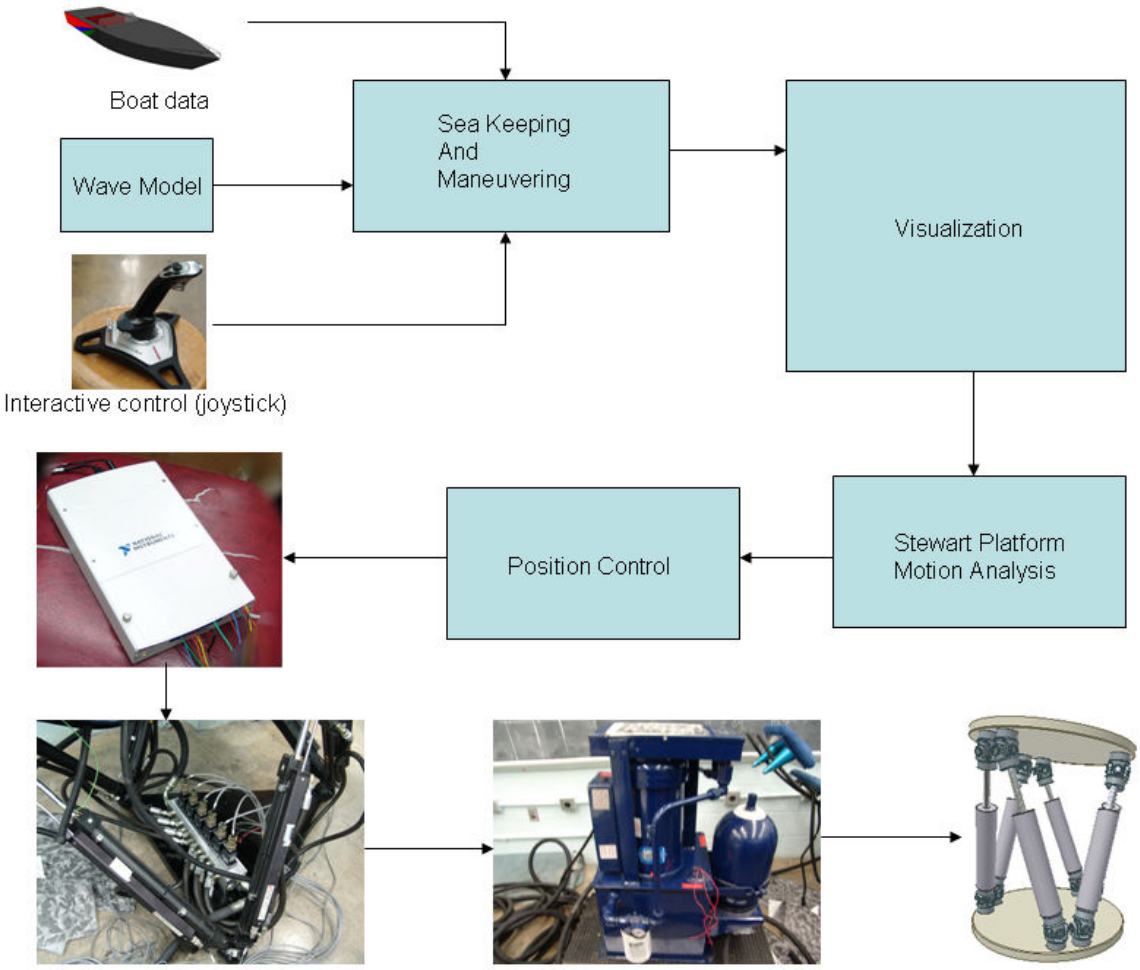

Figure 1: Flow Chart of the Project

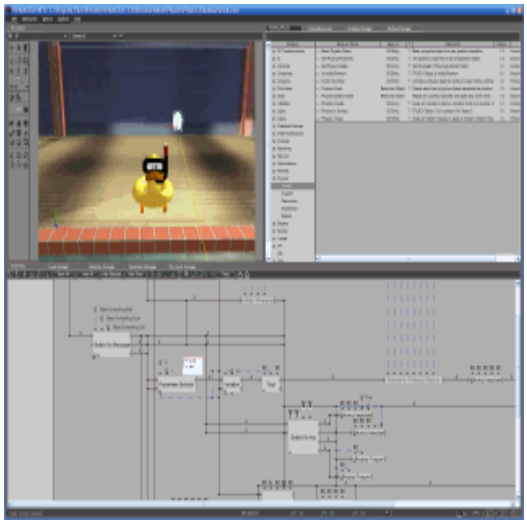

Figure 2: Example of Virtools application.

Currently the Wave implementation in Virtools interacting with the boat object is achieved using existing Virtools Building Blocks (BB) [6] applied to a deformable mesh which represents the water surface. First a Bezier Progresion BB[6] (Figure3) is applied to the mesh. This block interpolates a float according to a 2D Bezier curve in the [Min,Max] range, in a given number of milliseconds. This float value is then used to displace the vertices of the mesh after the values are processed by another Virtools BB called the noise BB[6].

The interpolation of the float value in the Progression BB is calculated over a 10 second interval which loops continuously during the simulation. The lower end of the Bezier curve is 0 and the higher end is 500 units. The progression of the float values in the curve occurs between 0% and 100% which defines the progression of the process. Start=0%, middle time=50%, end=100%. The float from the Progression BB is then sent to the Noise BB (Figure4) and generates a random starting point from the number sent. Then an axis vector is set to define the strength of the noise effect along each three axes (x,y,z) and displaces the vertices of the mesh with a random vector. These 2 BB combined generate the wave motion in the Virtools environment. The mesh is then declared as a movable floor object so that it can interact with the boat.

The interaction between the Wave and the boat (Figure 5) is currently produced by using the Object Keep[1] on Floor BB. This BB forces an object to stay on the declared floor. It uses the object's bounding box to test the object's position relative to the floor. The position of the boat object then is sent to the UDP_Sender [7] BB to calculate the leg lengths of the motion base simulator.

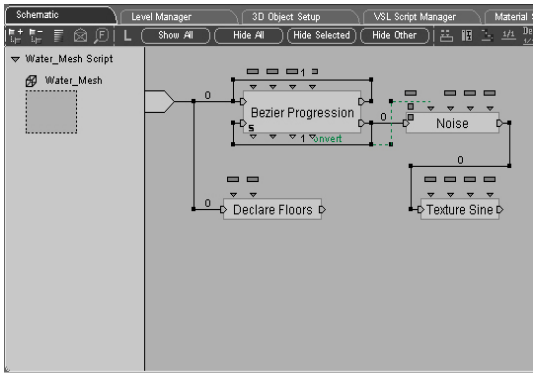

Figure 3: Bezier and Noise BB

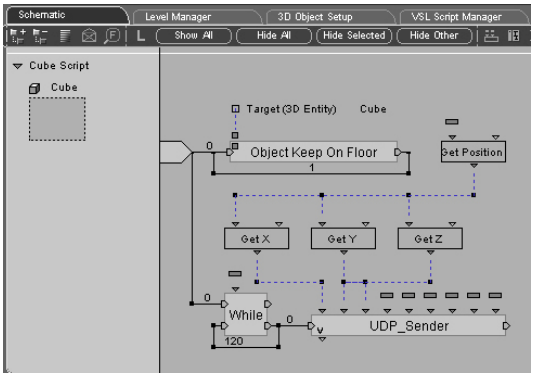

Figure 4: Keep on Floor BB

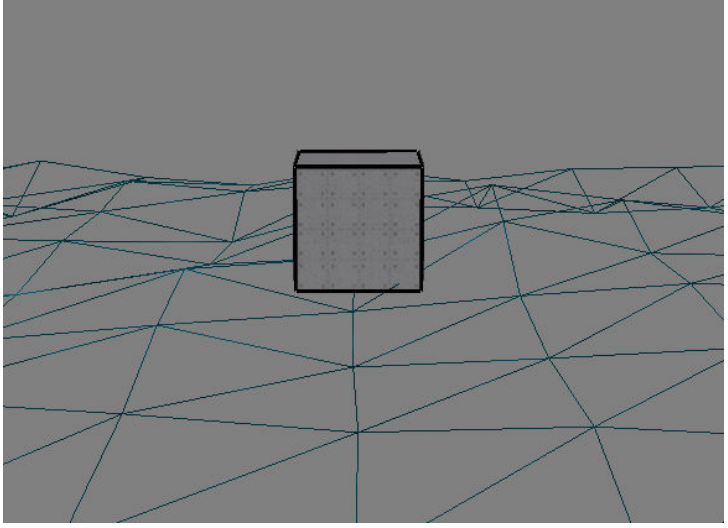

Figure 5: The Wave Pattern and the Object.

3. Inverse Kinematics Algorithm

This section derives the inversed kinematics equations for position analysis of the wave-riding simulator which is basically a Stewart platform. Let the centers of the base and the movable platforms of the Stewart platform be fixed at Points *O* and *C*, respectively. Also let subscripts, *i* and *j,* be denoted as the joints at the base and the movable platforms, respectively. The scheme of the system is shown in Figure 1.

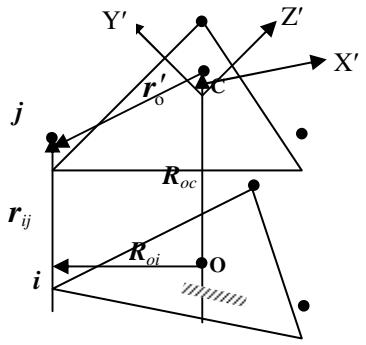

Figure 6: The coordinate axes at the center of the 6-DOF Simulator.

The center of the base, *C*, is considered as the center of the wave rider whose position and orientation will be issued by Virtool in our system at evenly spaced time instances. Our task here is to find the length of the cylinder, $\boldsymbol{r}_{ij}$, joining *i* and *j* for a given motion trajectory, in terms of the global coordinate system, *x*-*y*-*z*. Specifically, the goal here is to find the new position of the platform at t_i, for the given changes in the translational and angular displacements of the water craft, defined from t_{i-1} to t_i by $\Delta \boldsymbol{s}$ and $\Delta\theta$ as,

$$\Delta \boldsymbol{s} = \begin{Bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{Bmatrix} \text{ and } \Delta\theta = \begin{Bmatrix} \Delta\theta_x \\ \Delta\theta_y \\ \Delta\theta_z \end{Bmatrix} \qquad (1)$$

In other words, assuming that the length of the cylinder l_{ij}^{i-1} is known at t_{i-1}, the task is to find the new length at t_i, l_{ij}^{i}, in terms of $\Delta \boldsymbol{s}$ and $\Delta\theta$. These changes in the positions, $\Delta \boldsymbol{s}$, and in orientation, $\Delta\theta$, are output by Virtool at t_i expressed in terms of the global coordinate system. Particularly, it should be noted that $\Delta\theta$ is not the changes in Euler angles as presented in the literature[8,9,10]. In the following derivation, the superscript prime indicates that the associated quantity is defined in terms of the body-fixed coordinate system.

The equation used to compute the length of each cylinder is given by

$$l_{ij}^{i} = \sqrt{\boldsymbol{r}_{ij}^{T}\boldsymbol{r}_{ij}} \tag{2}$$

The superscript, *i*, indicates the time instant. Unless it is necessary for clarity, the superscript will be dropped from the rest of derivation.

The position vector from joint *i* to *j*, $\boldsymbol{r}_{ij}$, at time equal to t_i can be given by

$$\begin{aligned} \boldsymbol{r}_{ij} &= -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc} + \boldsymbol{r}_{0} \\ &= -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc} + A^{i}\boldsymbol{r}_{0}' \end{aligned} \tag{3}$$

where A^i is the transformation matrix evaluated at t_i, the vector, $\boldsymbol{R}_{oi}$, denotes the position vector from Point *O* to Point *i*, and similarly, $\boldsymbol{R}_{oc}$, denotes the vector from Point *O* to Point *C* at the previous time instant, t_i. The last term in Equation(2), $\boldsymbol{r}_0'$, is the position vector measured from Point *C* to Point *j*, whose value is fixed in terms of the local coordinate system.

At *t=0*, the body-fixed coordinate system of the movable platform is coincided with that of the base platform. Therefore, the transformation matrix A^0 is equal to the identity matrix. In this case, the corresponding Euler parameters[11] are $e_0 = 1$ and $e_1 = e_2 = e_3 = 0$. Thus, $\boldsymbol{p}^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T$. At time t_i, the change of the angular displacement of the movable platform, $\Delta\boldsymbol{\theta}$, is known. One can then find the updated $\boldsymbol{p}^i$ at t_i as

$$\boldsymbol{p}^{i} = \boldsymbol{p}^{0} + \Delta\boldsymbol{p} \tag{4}$$

where the changes in the Euler parameters are approximated by[9]

$$\Delta\boldsymbol{p} \approx \frac{1}{2}\left(G^{0}\right)^{T}\Delta\boldsymbol{\theta} \tag{5}$$

where G^0 is given as $G^0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (6)

With the new $\boldsymbol{p}^i$, one can then construct the new *E* and *G* matrices[11] at t_i and obtain the transformation matrix $A^{i-1,i}$ which accounts for the change of the coordinate system from t_{i-1} to t_i as $A^{i-1,i}=EG^T$, where *E* and *G* are given as

$$E = \begin{bmatrix} -e_1 & e_0 & -e_3 & e_2 \\ -e_2 & e_3 & e_0 & -e_1 \\ -e_3 & -e_2 & e_1 & e_0 \end{bmatrix} \tag{7}$$

and

$$G = \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix} \tag{8}$$

The accumulated transformation matrix at t_i, is then given by

$$A^i = A^{i-1} A^{i-1,i} \tag{9}$$

The value of $\boldsymbol{R}_{oc}$ is obtained by the following recursive relation,

$$\boldsymbol{R}_{oc} = \boldsymbol{R}_{oc}^{i\text{-}1} + \Delta \boldsymbol{s} \tag{10}$$

where $\Delta \boldsymbol{s}$ is known. Substituting Equation(10) into Equation(3), one obtains

$$\begin{aligned} \boldsymbol{r}_{ij} &= -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc} + \boldsymbol{r}_0 \\ &= -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc} + A^i \boldsymbol{r}_0' \\ &= -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc}^{i\text{-}1} + \Delta \boldsymbol{s} + A^i \boldsymbol{r}_0' \end{aligned} \tag{11}$$

In summary, Equation (1) can be rewritten as

$$\boldsymbol{r}_{ij} = -\boldsymbol{R}_{oi} + \boldsymbol{R}_{oc}^{i\text{-}1} + \Delta \boldsymbol{s} + A^i \boldsymbol{r}_0' \tag{12}$$

The change of the length of each of cylinder can be approximated from the velocity of the hydraulic cylinder in the time interval between t_{i-1} and t_i. This process results in the following relation,

$$\Delta \boldsymbol{r}_{ij} = \Delta \boldsymbol{s} - \tilde{\boldsymbol{r}}_0 \Delta \theta \tag{13}$$

where the skew matrix, $\tilde{\boldsymbol{r}}_0 = \begin{bmatrix} 0 & -r_{0,z} & r_{0,y} \\ r_{0,z} & 0 & -r_{0,x} \\ -r_{0,y} & r_{0,x} & 0 \end{bmatrix}$. The change, $\Delta \boldsymbol{r}_{ij}$, produces the change in the length of the leg as

$$\Delta l_{ij} = \frac{\boldsymbol{r}_{ij}^{i\text{-}1T} \Delta \boldsymbol{r}_{ij}}{l_{ij}^{i\text{-}1}} = \begin{bmatrix} \dfrac{\boldsymbol{r}_{ij}^{i\text{-}1T} I}{l_{ij}^{i\text{-}1}} & -\dfrac{\boldsymbol{r}_{ij}^{i\text{-}1T} \tilde{\boldsymbol{r}}_0}{l_{ij}^{i\text{-}1}} \end{bmatrix}_{1\times 6} \begin{Bmatrix} \Delta \boldsymbol{s} \\ \Delta \theta \end{Bmatrix}_{6\times 1} \tag{14}$$

Note that $\boldsymbol{r}_{ij}^{i\text{-}1}$ and l_{ij}^{i-1} in Equation(14) should be those evaluated at t_{i-1}. This can be observed from the following derivation:

$$
\begin{aligned}
l_{ij}^{i} &= \sqrt{\boldsymbol{r}_{ij}^{T}\boldsymbol{r}_{ij}} \\
&= \sqrt{\left(\boldsymbol{r}_{ij}^{i-1} + \Delta\boldsymbol{r}_{ij}\right)^{T}\left(\boldsymbol{r}_{ij}^{i-1} + \Delta\boldsymbol{r}_{ij}\right)} \\
&\approx l_{ij}^{i-1} + \frac{\left(\boldsymbol{r}_{ij}^{i-1}\right)^{T}\Delta\boldsymbol{r}_{ij}}{l_{ij}^{i-1}}
\end{aligned}
\tag{15}
$$

The collection of the lengths of six cylinders gives a matrix equation,

$$
\Delta l_{ij} = B\Delta\boldsymbol{q} \tag{16}
$$

where B is a 1×6 matrix whose row is equal to $\left[\frac{\boldsymbol{r}_{ij}^{i\text{-}1T} I}{l_{ij}^{i\text{-}1}} \quad -\frac{\boldsymbol{r}_{ij}^{i\text{-}1T}\tilde{\boldsymbol{r}}_{0}}{l_{ij}^{i\text{-}1}}\right]_{1\times 6}$ and $\Delta\boldsymbol{q} = \begin{Bmatrix}\Delta\boldsymbol{s}\\ \Delta\theta\end{Bmatrix}_{6\times 1}$.

4. User Datagram Protocol (UDP)

In Virtools, behaviors describe how elements act in an environment. Virtools provides a large collection of reusable behaviors in the form of behavior BBs. Users can create complex scenes using behavior building blocks through a simple graphical user interface. Users can also create their own building blocks using the Virtools software development kit (SDK) by developing dynamically linked libraries (DLLs) that implement the behaviors. In this project, the software that drives the motion base simulator and the Virtools visualization program can reside in different computers or the same computer, increasing configuration flexibility to facilitate fast computations. The communication between the Virtools program and the motion based simulator utilizes network-based User Datagram Protocol (UDP). A Virtools behavior building block UDP_Sender was developed to implement the UDP network communications. The building block (BB) UDP_Sender[7] is shown in 6.

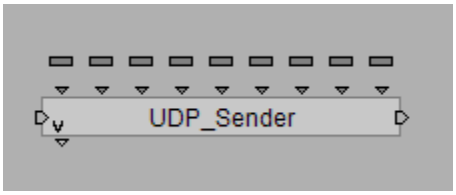

Figure 7: UDP_Sender Building Block

UDP does not use a three-way handshake to establish connection as does Transmission Control Protocol (TCP)[5]. It also does not check every packet it receives. Thus UDP provides fast communications and is selected in this project. The data included in the UDP socket includes water craft position (x, y, z) and orientation (roll, pitch, yaw). These six parameters are then converted to the leg lengths of the motion base simulator using inverse kinematics applied to the Stewart Platform.

5. Stewart Manipulator

5.1 Introduction of the Simulator

A Stewart Platform manipulator is applied to simulate the motion created by the Virtools. The Stewart Platform manipulator has the characteristic of great rigidity, high-force-weight ratio, and with the mobility of six degree of freedom. The 6-DOF Stewart Platform shown in Figure 8 was controlled by an old and unavailable DOS programs. . Figure 9 shows the main components of the simulator. The six 5/3 servo directional control valves, which are activated by solenoids, are used to control the motion of the hydraulic fluid. Figure 10 shows the schematic motion control of each cylinder. Depending on which side of the solenoid is activated, the fluid will move the cylinder either in the upward or downward direction. To activate the solenoid, a 5V DC with 50 mini-amps is needed. When both solenoids are not activated, the fluid supply will be cut off and the cylinder will remain at its current position. Also a potentiometer is attached to each cylinder to provide a feedback signal of the current position for each cylinder. Therefore, to program this platform, a data acquisition device with six analog inputs (for potentiometers) and twelve $\pm$ 5V digital outputs (for the control of the six servo valves) is needed. In addition, a 5V power supply for the six potentiometers is also required. To receive the feedbacks from the potentiometers and to activate the servo valves simultaneously, the Flat Sequence Structure[8] in the LabVIEW 8.2 is selected. Based on the hardware needs of this control, an NI USB-6251[4] is selected. The device (Figure 11) can supply thirty-two digital I/Os, sixteen analog inputs, and two analog outputs. The procedures to program the simulator using LabVIEW are listed in the following paragraphs.

Figure 8: The 6-DOF Hydraulic Stewart Platform.

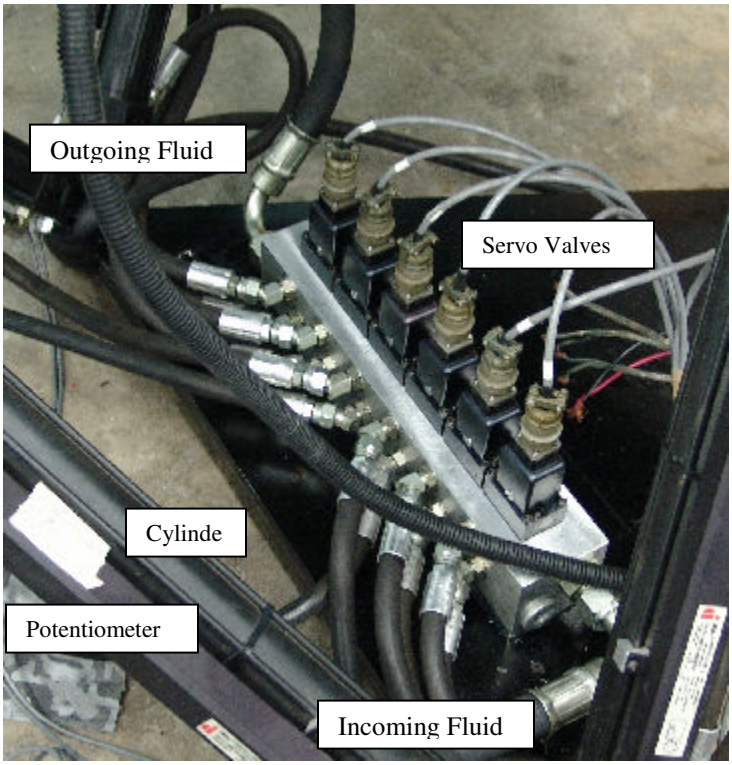

Figure 9: Main components in this Stewart simulator.

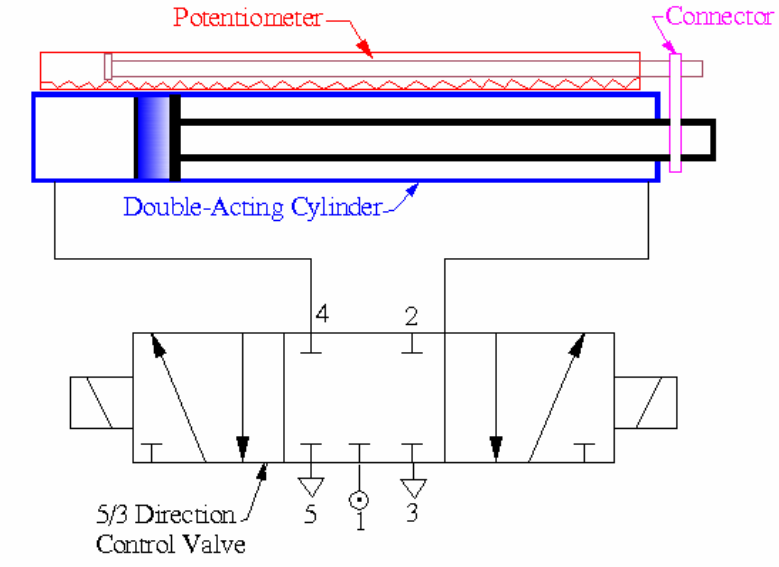

Figure 10: Schematic motion control for each cylinder.

Figure 11: NI USB 6251[4].

5.2 Testing Procedures for Potentiometers

Each potentiometer was taken out of the simulator for this testing purpose. The steps are listed as following:

(1) Power the potentiometer with a five Volt in D.C.
(2) Move the connector (Figure 10) on the potentiometer to a certain displacement and measure the output voltage. Result shows that the displacement is proportional to the output voltage.
(3) Interface the feedback from the potentiometer to one of the analog input ports at the NI USB 6251 and check the readout, which is compared to the output voltage in Step 2. These two readings must be the same. To read the signal, the "Measurement & Automation Explorer" of the NI USB 3251 must be activated. Figure 12 shows the analog input ports available in this device and Figure 13 shows the pin connections. The feedback from the potentiometer can be connected to any one of the ports listed on the screen.
(4) Repeat the same test from Steps 1 to 3 for the other five potentiometers.
(5) Wire the connection between the potentiometers and the NI USB 3251. Figure 14 shows the design drawing.

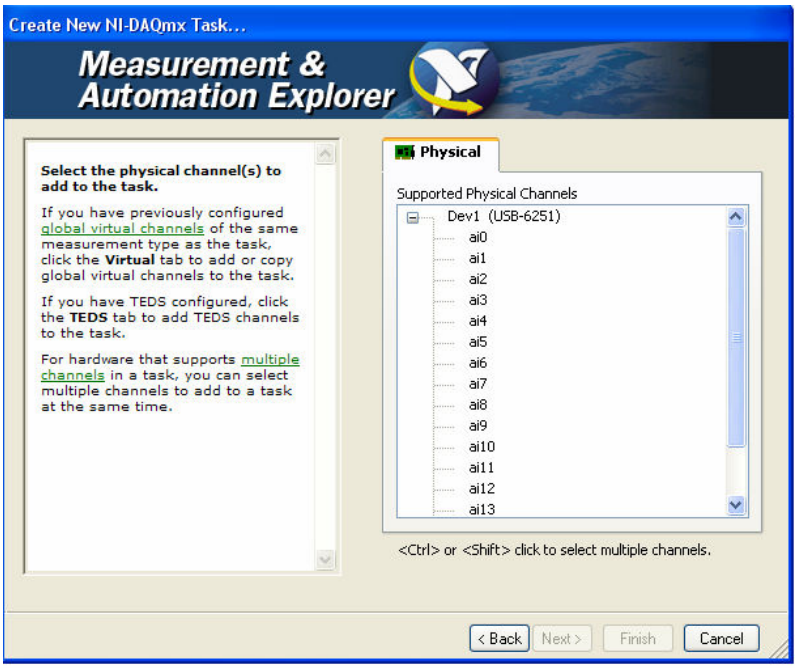

Figure 12: Measurement & Automation Explorer.

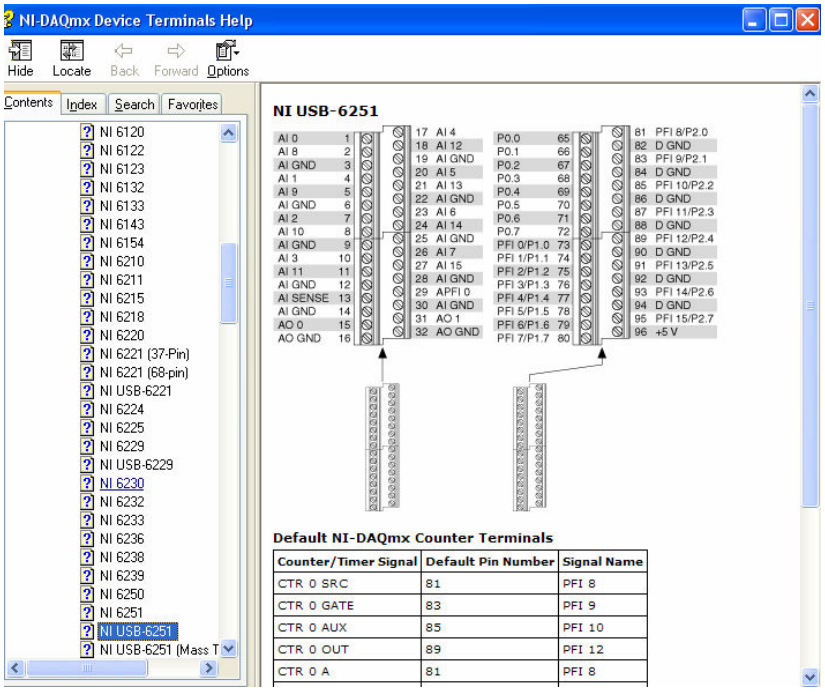

Figure 13: USB 6251 Pin-connection diagram.

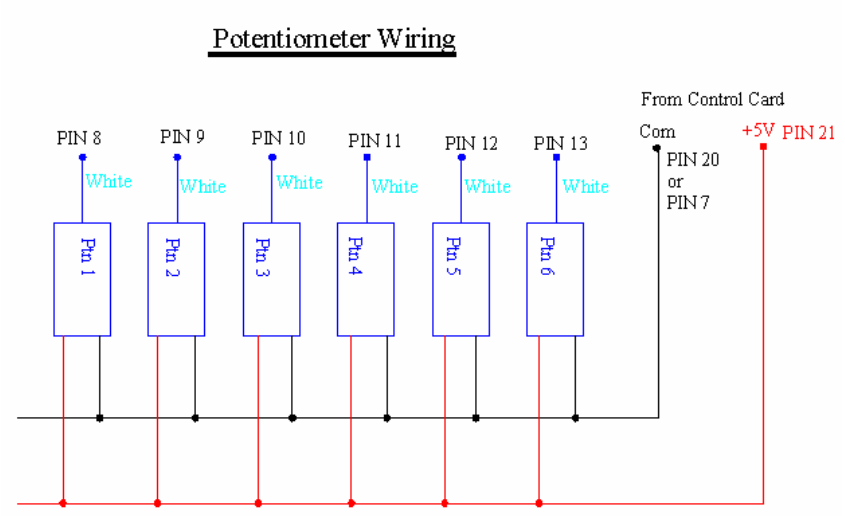

Figure 14: Electrical wiring for the potentiometers.

5.3 Testing Procedures for Servo-Control Valves

The following procedures are used to test the servo-control valves:

1. Power the hydraulic-fluid pump.
2. Power one of the solenoids on a servo-control vale and see if the cylinder is moving. If it is, then power off the solenoid and power on the other solenoid and see if the cylinder is moving in the opposite direction.
3. Cut the power off on both solenoids and see if the cylinder remains at the same position.
4. Design the pin connection between the servo-control valves and the controller device. Figure 15 shows the drawing.

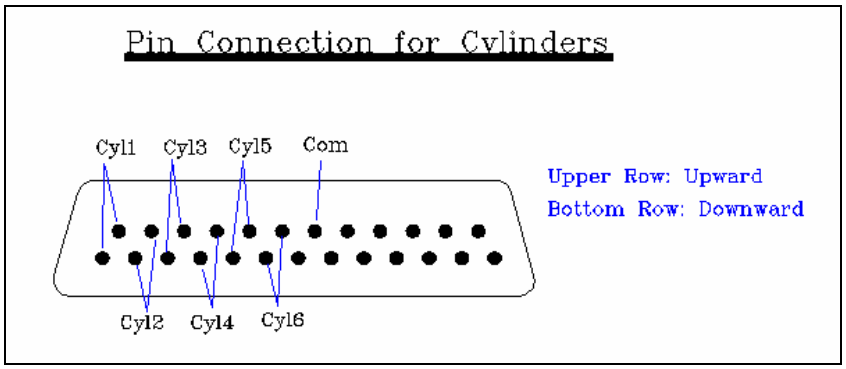

Figure 15: Pin-connection diagram for the servo-control valves.

5.4 LabVIEW Programming for Potentiometers

Figure 16 shows the LabVIEW program for sensing the feedback signals from the six potentiometers. The blue lines represent the feedback signals of the voltages detected from each potentiometer. The following equation is used to display the current position of each cylinder[12]:

$$Curpos = \frac{FBD}{ACTV} * Cylen \tag{17}$$

Where *Curpos, FBD, ACTV*, and *Cylen* represent current position, feedback signals, activating voltage, and cylinder length respectively. In this case, *ACTV = 5 V* and *Cylen = 12* inches.

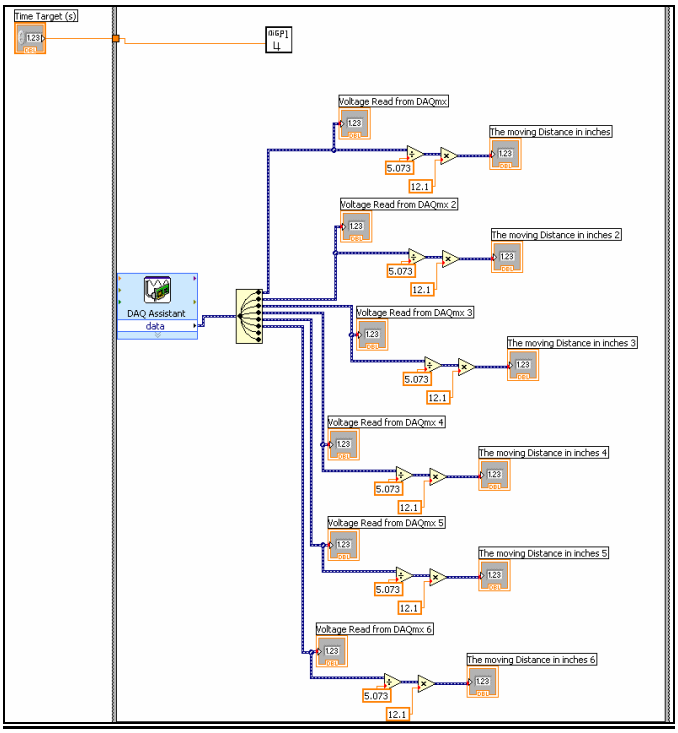

Figure 16: The LabVIEW program for six potentiometers.

5.5 LabVIEW Programming to Activate Solenoids

Figure 17 shows a program to command the motion of a cylinder. Since the flow rate of the hydraulic fluid is a constant, the output from the program is the unit of *ms*. The following equation is applied to calculate the traveling time for each cylinder:

$$TravTime = \frac{(ComDisp/12*5) - Vpotentiometer}{5} * 1.6 \qquad (18)$$

Where *TravTime, ComDisp, Vpotentiometer* represent traveling time, voltage read from a potentiometer, and commanded displacement for the cylinder. In this case, each cylinder needs approximate 1.6 seconds to travel twelve inches. If the value of the numerator in Equation 18 is positive, then only the left solenoid in Figure 10 will be activated. If the value is negative, then only the right solenoid will be activated. Equation 18 was implemented in the LabVIEW program shown in Figure 17.

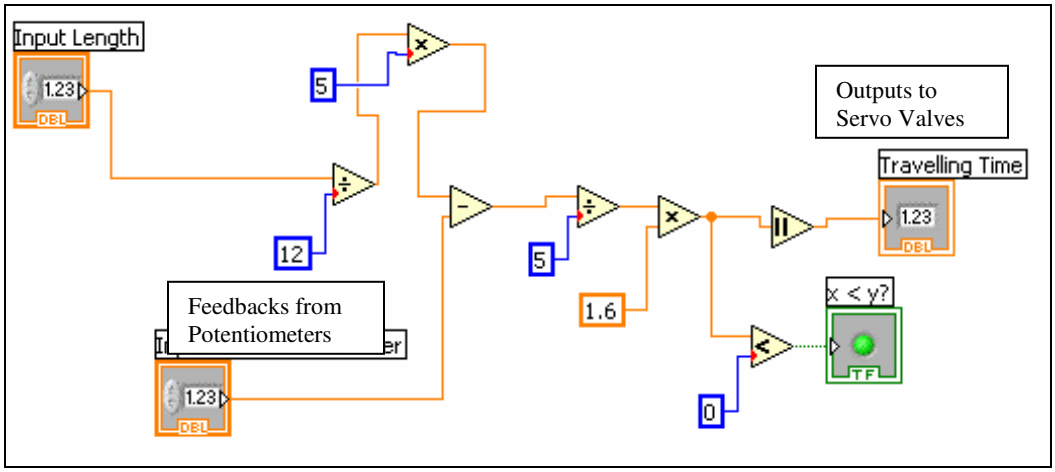

Figure 17: The LabVIEW program to activate a cylinder.

5.6 The Combined LabVIEW Program

Figure 18 shows the flow chart of the combined LabVIEW. The UDP port in the LabVIEW will be opened to read six character strings every time. The "String to Number" function which is also available in the LabVIEW converts the string to six real numbers representing six cylinder displacements. The combined program was developed by using the subprograms developed in Session 5.4 and 5.5. Because of the size of the program, only partial program to control Cylinder 1 is shown. To activate the cylinders simultaneously, the flat sequence structure of the LabVIEW was applied.

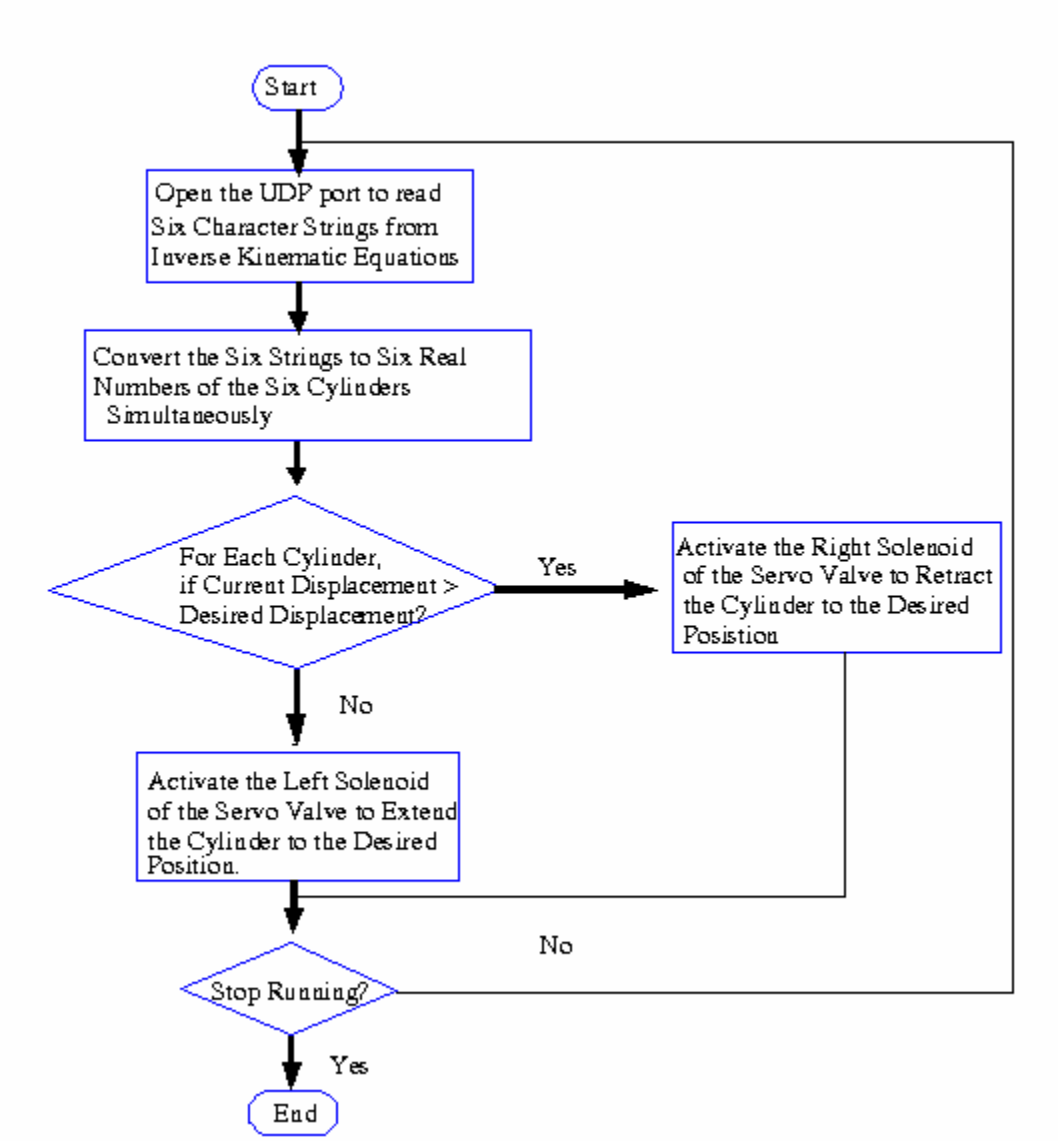

Figure 18: Flow chart of the combined program.

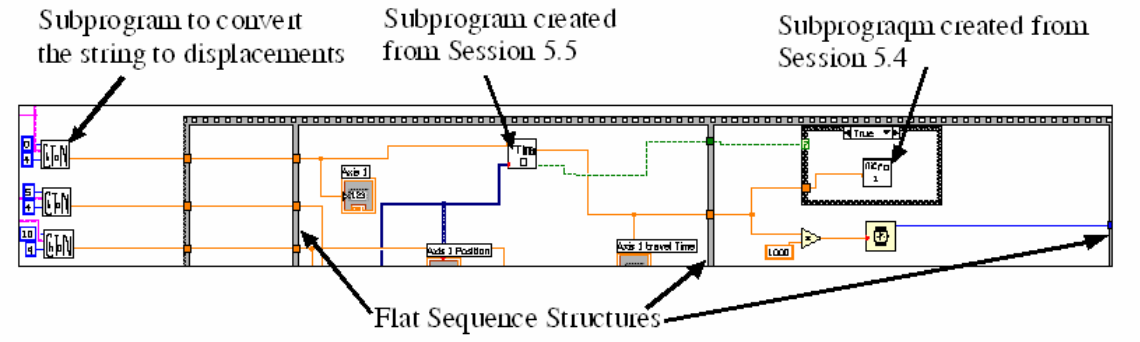

Figure 19: Partial LabVIEW Program to Control Cylinder 1.

6. Summary

From this design project, engineering students can achieve the following objectives:

(1) Generate three-dimensional motions using VirTools.
(2) Understand the kinematics of a Steward manipulator.
(3) Data Transfer between computers.
(4) Create a control software program using current programming tools.
(5) Use data acquisition device to interact with sensing devices and actuators.
(6) Check and wire electrical circuits.

The approach presented in this paper to function a Stewart hydraulic motion base simulator can also be applied to other simulators which have analog/digital position sensors and servo control valves. The only concern in this motion control is that the hydraulic fluid rate is always constant and therefore the speed is fixed. A flow control valve is to be added in the future so that the speed can be adjusted.

References:

1. http://www.Virtools.com
2. H. Guo, "Dynamic analysis and simulation of a six degree of freedom Stewart platform manipulator", *Proceedings of the Institution of Mechanical Engineers, Part C (Journal of Mechanical Engineering Science)*, v 220, n C1, Jan. 2006, p 61-72
.3. LabVIEW programming, http://www.ni.com/labview/whatis/?metc=mtfew9
4. NI USB-6251, http://sine.ni.com/nips/cds/view/p/lang/en/nid/202597
5. W. Stallings, "Data and Computer Communications, Sixth Edition", Prentice Hall, , 2000.
6. Virtools 4.0 User Guide, 2006.
7. Virtools 4.0 SDK programmers guide, 2006.
8. *Application Notes for the Hexad 3000H Six Degree-of-Freedom Motion Platform*, Arnicola Simulation Systems, Inc. Conklin, NY. Rev. 7, 2005.
9. Liu, K, Fitzgerald, J. M. and Lewis, F. L., "Kinematic Analysis of a Stewart Platform Manipulator", *IEEE Transactions on Industrial Electronics*, Vol. 40, No.2, 1993, pp. 282-293.
10. Wang, Z., Wang, Z., Liu, W., Lei, Y, " A Study on Workspace, Boundary Workspace Analysis and Workpiece Positioning for Parallel Machine Tools", *Mechanism and Machine Theory*, Vol. 36, 2001, pp. 605-622
11. Haug, E. J., *Intermediate Dynamics*, Prentics-Hall, Inc., Englewood Cliffs, NJ, 1992
12. J. Rehg, "Introduction to Robotics in CIM Systems", Prentice Hall, 2003.