University of Tennessee, Knoxville
Trace: Tennessee Research and Creative Exchange

University of Tennessee Honors Thesis Projects | University of Tennessee Honors Program

5-2011

Modeling of Human Hand Motion in a Maya Environment

Andrew M. Quinn
aquinn4@utk.edu

Aeshan Ali

Tessa Taylor

Quincy Beasley

Follow this and additional works at: http://trace.tennessee.edu/utk_chanhonoproj

Part of the Biomechanics and biotransport Commons, and the Other Biomedical Engineering and Bioengineering Commons

Recommended Citation
Quinn, Andrew M.; Ali, Aeshan; Taylor, Tessa; and Beasley, Quincy, "Modeling of Human Hand Motion in a Maya Environment" (2011). *University of Tennessee Honors Thesis Projects.*
http://trace.tennessee.edu/utk_chanhonoproj/1431

This Dissertation/Thesis is brought to you for free and open access by the University of Tennessee Honors Program at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in University of Tennessee Honors Thesis Projects by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

Modeling of Human Hand Motion in a Maya Environment

May 6, 2011

Prepared by

University of Tennessee Senior Biomedical Engineering Students:

Aeshan Ali

Quincy Beasley

Andrew Quinn

Tessa Taylor

MODELING OF HUMAN HAND MOTION IN A MAYA ENVIRONMENT

Aeshan Ali

Quincy Beasley

Andrew Quinn

Tessa Taylor

May 6, 2011

CONTENTS

Page

LIST OF FIGURES……...v

LIST OF TABLES……...vii

ABSTRACT…….……..1

1. BACKGROUND……..…….……...1

2. EQUIPMENT……...2

3. OBJECTIVES……...…....3

3.1 DESCRIBE ANATOMICAL FEATRUES AND MOVEMENT OF HAND AND ARM……...3

3.2 BUILDING A MAYA MODEL……..3

3.3 EXTRACTING ANGULAR OUTPUT ……..…3

3.4 MAYA INTERACTION…….4

4. METHODOLOGY……...5

4.1 RESEARCH…….5

4.2 HAND ANATOMY……..……5

4.3 MOTION CAPTURE, CYBERGLOVE…….6

4.4 MOTION CAPTURE, OPTOTRAK...…….6

4.5 MATLAB……..……..……..7

4.6 MAYA……...……...7

4.7 MOTION MODELING……...8

5. TIMELINE……9

5.1 FALL TIMELINE……9

5.2 SPRING TIMELINE……...…….9

5.3 TEAM ROLES……...10

6. FUTURE WORK……11

6.1 MAYA (REAL TIME) ……..……..11

7. BROADER IMPACT…….12

8. REFERENCES……..…….13

APPENDIX A……A-1

APPENDIX B……B-2

APPENDIX C……C-3

LIST OF FIGURES

Figure		Page
1	Coordinate system used to base rotations of hand motion……...5	
2	Hand motion in DCU……...……..……..6	
3	Fall and Spring Timeline……..……...9	

LIST OF TABLES

Table **Page**

1 Individual Roles……10

ABSTRACT

The purpose of this senior design project was to model human hand and arm motion using a combination of kinematics, CyberGlove II, Optotrak, and a Maya Virtual Reality Environment. Using MEL and C++ coding, coordinates from the Cyberglove II were extracted and implemented into the Maya program. Using the Optotrak system, coordinates from the forearm and hand were extracted and then transformed in Matlab. Both sets of coordinates were then used to model motion within the Maya environment. Future applications include using inverse kinematics to detail the forces within the hand and arm, which can then be applied to prosthetic design.

1. BACKGROUND

Human hand modeling is extremely important for the research, design, and implementation of prosthetics, robotic surgery, and ergonomics. The human hand consists of 27 bones interwoven with muscles, connective tissues, and tendons which receive input from axial neurons. The human hand is responsible for movements that are extremely precise as well as blunt movement.

Human hand motion is one of the most challenging features of a human to simulate. The human hand has 27 degrees of freedom (DOF). Each finger has 4 DOF, 3 for extension and flexion and one for abduction and adduction. The thumb on the other hand has 5 DOF, and the wrist has 6 DOF for rotation and translation.

The human hand is composed of 14 phalanges, 5 metacarpals, and 8 carpals for a total of 27 bones. Adding up the amount of bones in both hands would account for roughly one fourth of the amount of bones in an adult human. In addition the hand consists of about 20 muscles. The bones and muscles along with the amount of ligaments and tendons in the hand increase the complexity of capturing realistic hand motion.

In order to produce realistic motion, coordinates from the sensors within the Cyberglove will be extracted from the Cyberglove DCU and applied to the Maya program using MEL coding. Also, in order to have the hand anchored to the forearm, coordinates from sensors located on the forearm and hand will be extracted using the Optotrak system, and again applied to Maya using MEL coding.

2. EQUIPMENT

The equipment available to us is the Cyberglove II, which will give an output according to the motion of the hand being observed, Windows Visual, which will extract the angular output from the glove using C++ coding, and MAYA, which will use the data from Windows Visual to run a simulation model of the hand motion through a MEL code. For tracking the motion of the arm, making the glove motion relative to the arm, the Optotrak system was used, and the coordinates obtained from it were implemented in Maya with a MEL code.

3. OBJECTIVES

The primary objective of the MAYA design project was to use a number of cohesive factors to determine the extents and limitations of hand and arm motion. The reasoning behind this is that there are a variety of different biomedical applications that require the full study this motion. This study requires that individual portions of the hand and arm be modeled and detailed so that the MAYA environment may understand the linearity and attachment of individual finger and arm joints and their particular centers of gravity.

3.1 DESCRIBE ANATOMICAL FEATURES AND MOVEMENT OF HAND AND ARM

Before any computer involvement could be started it was important to first understand the reasoning behind the hand and arm's motion. The structure, function, and limitations are all essential for this part. With the aforementioned information the next stage of our developmental process can occur and the motion can be detailed. The hand will be studied by the following joint sections: wrist, metacarpophalangeal joint (MCP), proximal interphalangeal joint (PIP), distal interphalangeal joint (DIP). It is important to note that the thumb has no PIP or DIP joint, and are instead replaced by an interphalangeal joint (IP). The arm will be studied specifically to anchor the hand and to have the hand move relative to the arm. It will be represented solely by the radius and ulna.

3.2 BUILDING A MODEL IN MAYA

With the anatomical information of the hand and arm, a fully functioning model needs to be created. Using the LEX model provided and the links and joints tools in Maya, a functional hand and arm model was made. Joints ran through the radius/ulna, wrist, and through to the thumb and four fingers.

3.3 EXTRACTING OUTPUT

In order to move the model created in Maya, actual motion will have to be defined using angles and transformation matrices. While constructing these angles it is important to detail the degrees of freedom associated with each of the following joints listed above. The wrist will have two degrees of freedom, while all the joints in the fingers will have four degrees of freedom, and the thumb will have five degrees of freedom. These angles will first be interfaced with coding software, Windows Visual, to extract the angular data. The aforementioned processes will be used to determine the actual hand motion in the MAYA environment. Actual hand movement capture will be done using the Cyberglove II. Furthermore, wrist movement and anchoring the wrist to the arm will be done thru optical tracking using skin markers from the Optotrak.

3.4 MAYA INTERACTION

Once the hand and arm motion is captured and the hand motion angles are extracted using coding software, the output from the Cyber Glove II and Optotrak can then be input into MAYA to recreate hand motion and movement. The C++ coding was used to extract the angular output from the Cyberglove and put it into a text file. Matlab was used to transform the output from the Optotrak sensor located on the wrist to be relative to the sensor located on the forearm. Both sets of coordinates will then be read by a MEL code that will move the hand model within Maya.

4. METHODOLOGY

The motion of the human hand is very complex. With the numerous degrees of freedom making up hand movement, many different angles must be analyzed and computed in order to correctly model the motion. In order to accurately and correctly model this motion, data from the Cyberglove and Optotrak will be extracted using C++, analyzed with Matlab, and put into Maya using a MEL code which will move our model.

4.1 RESEARCH

The beginning of the work consisted of researching different aspects of the project. Topics researched included background on the CyberGlove II, Maya modeling, MEL code, Optotrak skin markers, matrix transformations and data extraction with the C++ coding, MatLab commands, and general hand and arm anatomy. The research gave the group a better understanding of how the motion will be captured, transformed, and displayed, and the necessary work that will have to go into those processes.

4.2 HAND ANATOMY

The extent of hand motion we would be analyzing was defined. We defined a three dimensional coordinate system within our links and joints model as shown in Figure 1. The wrist will have two degrees of freedom with the palm included, the fingers will have four degrees of freedom, and the thumb will have five degrees of freedom. Each segment will have its own coordinate system and will move relative to each other. However, all motion of the wrist will be made relative to the forearm through transformations performed in Matlab.

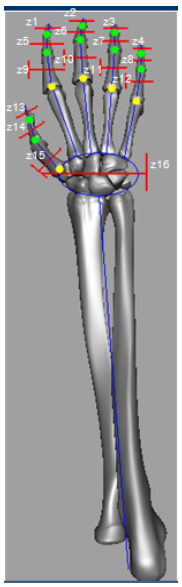

Figure 1. Coordinate system used to base rotations of hand motions

4.3 MOTION CAPTURE, CYBERGLOVE

The next step was to install the Cyberglove to begin extracting motion data. The process involved downloading the Software Development Kit (SDK) which contained the Device Manager and Device Configuration Utility (DCU). Further, Bluetooth software was installed to connect the glove wirelessly to the computer. Once installation was completed, the sensors within the glove were able to display motion on the DCU as shown in Figure 4.

Figure 2. Hand motion in DCU

Output from the glove was given as relative angular positions. As mentioned above, the angular output was extracted using the C++ coding and stored in a text file. The C++ code can be seen in Appendix B.

4.4 MOTION CAPTURE, OPTOTRAK

To capture the motion of the arm, the Optotrak system was used. Two sensors located on the forearm and hand were used to collect data on wrist motion. Furthermore, using the Matlab code mentioned next, the wrist and subsequent fingers were made to move relative to the forearm. The output from the Optotrak consisted of rotation angles and translation coordinates.

4.5 MATLAB

To better understand some of the many MatLab commands that will be required in translating the hand motion, a MatLab exercise supplied by our team mentor was completed. This exercise allowed us to become much more familiar with basic MatLab commands and different methods on applying those commands. The exercise consisted of creating for loops and if statements in order to count certain values in a 2 x 19446 supplied matrix.

Matlab was then used to transform data points obtained from Optotrak. As seen in the Matlab code in Appendix C, transformation matrices were created for the data collected from the sensors located on the hand and forearm. Then, calculations were made within the code to make the wrist movements relative to the forearm.

4.6 MAYA

Since all of the modeling of the hand and arm will be animated in Maya, exercises to familiarize ourselves with the program have been completed. Those exercises include creating objects in Maya, and setting the objects into motion to complete a task. While very simple, those exercises were helpful in understanding Maya better.

During the spring semester we developed a complete hand and arm model that includes the joint system in Maya. This joint system in Maya is based on the location of the sensors in the Cyberglove. A new MEL code was also made, and calls upon each joint to move based on data extracted from the Optotrak and Cyberglove tools. Using the MEL code found in Appendix A, we have been able to create accurate motion within this model.

4.7 MOTION MODELING

The first portion of creating an arm and hand model will be proper selection of which limbs will be utilized. A skeleton model will be loaded while only bones of the right arm and hand are retained. Once the bones are simulated in the MAYA environment it is important to create a joint chain. A joint chain will be created using the Joint Tool to create a joint from the elbow to the edge of the wrist. Further, joints will run from the wrist to each additional phalange. The joint chain will be placed over its correlating bone section and will be attached using the Apply Skin tool. A motion code to apply motion to the model is created using MEL coding language. The MEL code used can be seen in Appendix A. The code draws upon the actual model created, and a series of three dimensional test points to apply motion. The MEL code uses the angles output from the Cyberglove and the angles transformed from the Optotrak system and creates motion within the created hand /arm model.

5. TIMELINE

5.1 FALL TIMELINE

The MAYA- Cyberglove II project was assigned to us in the fall of 2010 when we were introduced to our group's mentor, Nicholas Battaglia, who is a graduate student. Shortly after the initial assignment, we met with Nick and discussed the overall objectives of the project, tools available to our group, and individual research the group needed to complete in order to be successful. Further, after completing our 'Project Plan' and 'Function and Requirements Document' we determined the objectives that should be completed before the end of fall semester. These objectives include obtaining motion capture information from the Cyberglove II bionic hand, inputting this information in information into a Matlab program, and finally interfacing the Matlab program with the MAYA program.

However, because of the prolonged arrival of the Cyberglove II, the direction of the project reversed itself. Instead, we have begun by familiarizing ourselves with MAYA, and modeling the motion of the arm. We are currently working on that aspect of the project. The previous timeline can be seen in Figure 5.

	Task Name	Duration	Start	Finish
1	Obtain and Analyze Output from Cyberglove	15 days?	Thu 9/30/10	Wed 10/2
2	Analyze the Cyberglove output in MatLab	8 days?	Thu 10/21/10	Mon 11/
3	Interface Matlab with Maya	14 days?	Tue 11/2/10	Fri 11/
4	Set-up VR Environment in Maya	43 days?	Wed 1/12/11	Fri 3/
5	Reality Simulation	18 days?	Tue 3/22/11	Thu 4/
6	Group/Project Assignments Distributed	1 day?	Tue 8/31/10	Tue 8/
7	F and R document and Project Plan	7 days?	Wed 9/22/10	Thu 9/
8	Midterm Peer Review	2 days?	Mon 10/4/10	Tue 10/
9	Midterm Design Rpeort	7 days?	Mon 10/4/10	Tue 10/
10	Final Oral Reports	7 days?	Mon 11/15/10	Tue 11/
11	Final Design Report	7 days?	Mon 11/15/10	Tue 11/
12	Design Notebook	61 days?	Tue 9/7/10	Tue 11/
13	Peer and Self Evaluations	4 days?	Thu 11/25/10	Tue 11/

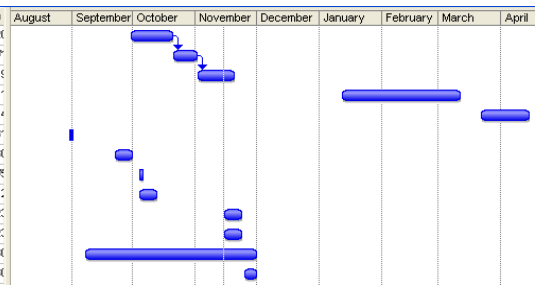

Figure 3. Previous Timeline of Fall and Spring Semesters

5.2 SPRING TIMELINE

As of the completion of our project, we have utilized the Optotrak system, the Cyberglove II, C++ coding, MEL coding, Matlab, and Maya. The Cyberglove has been installed and is fully functional. The C++ coding has been written to obtain angular output from the Cyberglove. A Matlab program has been made to perform transformations on the extracted Optotrak data. A MEL code has been written to input the transformed coordinates into the model created within Maya.

5.3 TEAM ROLES

Drew Quinn, our team leader, is responsible for compilation and completion of all assignments. Tessa Taylor, the organization chair, is responsible for the group binder. Ace Ali, our primary C++ programmer, has written code to pull angular coordinates from the Cyberglove. Ace has also served as a Maya assistant to Quincy. Quincy Beasley, our graphical interface chair (Maya), is the leader of the Maya program. He has worked to incorporate the extractions from Ace into the Maya environment by writing a MEL code for Maya to accept data and create motion accordingly. He has also created the arm/hand model within Maya, and written the Matlab code for transformations. Below is a table summarizing the title role of each individual within the group.

It was suggested that collaboration between the two hand project groups would be necessary and beneficial. Collaboration between groups would allow comparison of data which would result in more accurate data. Also, after the data was acquired from the Cyberglove it was easier to determine the relevance of the results, and was analyzed and interpreted more thoroughly. Further, combining the two groups experience with the unfamiliar software allowed problems to be solved quicker and more effectively.

Collaboration between the two hand groups has been developed. Drew, Tessa, and Krista worked to install the Cyberglove which required numerous interactions with the Cyberglove manufacturer. Ace and Jonathon worked to develop the C++ coding that pulled the Cyberglove data. Quincy and Galina developed a hand model within Maya, while Quincy also developed a MEL code that pulls the output data and inputs it into Maya.

Table 1. Individual Roles

Name	Team Role
Drew Quinn	Group Leader, Cyberglove
Quincy Beasley	Graphical Interface Chair
Ace Ali	Primary C++ Programmer
Tessa Taylor	Organization Chair, Cyberglove

6. FUTURE WORK

6.1 MAYA (REAL TIME)

If more work is done on this project, the glove can potentially move in real time with the Maya program. This will allow the simple analysis of any desired motion. The analysis with inverse kinematics could lead to improvements in orthopedic and prosthetic design by allowing more accurate data accessible very quickly.

7. BROADER IMPACT

The research done on the motion analysis of the human hand will be beneficial to many engineers and scientists in the biomechanics field. The results can be used to analyze the forces within the hand, and how different anomalies, such as a birth defect, injury, lifestyle, etc., can affect the motion of the human hand. Those results can then be applied to certain rehabilitation procedures, or even more so to prosthetic hand and arm design.

8. REFERENCES

1. ElKoura and Singh 2003 'Handrix: Animating the Human Hand' Eurographics/SIGGRAPH Symposium on Computer Animation).

2. Horace H S Ip and C S Chan 'Dynamic Simulation of Human Hand Motion Using an Anatomical Correct Hierarchical Approach' Image Computing Group, Department of Computer Science,City University of Hong Kong, 83, Tat Chee Avenue, Hong Kong.

3. Craig L Taylor, Ph.D. , Robert J. Schwarz, M.D 'The Anatomy and Mechanics of the Human Hand'

APPENDIX A.

MEL CODE FOR MOTION

APPENDIX A. MEL CODE FOR MOTION

```
{
      float $joint1Angle, $xVal, $yVal, $zVal;
      int $fileid;

      // need to load the robot (needs to be in current directory)
      file -o -f "C:/MayaPolaris/mayaRobotGoodPerspective.mb";
      refresh -f;

      // in order to test the robot, will read in points from a file
      $fileid = fopen("C:/Documents and
Settings/mkuhn/Desktop/mayaOptoFiles/robotTestPointsCircle.txt","r");
      if ($fileid == 0)
            print("ERROR: Problem reading in text file.\n");

      // iterate through text file, collecting each set of x,y,z coordinates
      // pass coordinates to Maya to show planned robot trajectory
      string $nextWord = fgetword($fileid," \n");
      while (size($nextWord) > 0) {
            // each iteration of this while loop corresponds to one robot
position
            $xVal = $nextWord;
            $yVal = fgetword($fileid," \n");
            $zVal = fgetword($fileid," \n");

            // this is all the code that is needed to get the robot in the
correct
            // position
            $joint1Angle = getJoint1Angle($xVal,$zVal);
            setAttr "joint1.rotateX" $joint1Angle;
            setAttr "ikHandle1.translateX" $xVal;
            setAttr "ikHandle1.translateY" $yVal;
            setAttr "ikHandle1.translateZ" $zVal;

            // delay by 1 second for visualization
            refresh -f;
            pause -sec 1;


            $nextWord = fgetword($fileid," \n");
      }

      return "done";
}

proc float getJoint1Angle(float $xVal, float $zVal)
{
      float $myAngle;
      $myAngle = atan2d($zVal,$xVal);
      if ($myAngle > 0)
            $myAngle = 180 - $myAngle;
      else
            $myAngle = -180 - $myAngle;

      return $myAngle;
}
```

APPENDIX B

C++ CODE FOR DATA EXTRACTION

APPENDIX B. C++ CODE FOR DATA EXTRACTION

```
/*****************************************************************
 AUTHOR: Chris Ullrich.
 Description: Base demonstration.
               -- COPYRIGHT VIRTUAL TECHNOLOGIES, INC. 1999 --
*****************************************************************/

#if defined( _WIN32 )
#define _WIN32_WINNT 0x0400                // avoid winsock problem
#include <iostream>
using std::cout;
#else
#include <unistd.h>
#include <iostream.h>
#endif
#include <iostream>
#include <fstream>
using namespace std;
#include <vhandtk/vhtBase.h>

// Turn off the following to use a tracker emulator instead of
// a real tracker
//#define USE_REAL_TRACKER

//
// Main function for the demo.
//
int main( int argc, char *argv[] )
{
      ofstream myfile;
      myfile.open ("C:\\example.txt",fstream::out|fstream::app);
      myfile <<"Writing this to a file.\n";


      vhtIOConn *gloveDict;
      vhtCyberGlove *glove;
      vhtIOConn *trackerDict;
      vhtTracker *tracker;
      try
      {
            // Connect to the glove.
            gloveDict = vhtIOConn::getDefault( vhtIOConn::glove );
            // Expand the CyberGlove connection to the CyberTouch
capabilities.
            glove = new vhtCyberGlove(gloveDict);

            // Connect to the tracker
#if defined( USE_REAL_TRACKER )
            trackerDict = vhtIOConn::getDefault( vhtIOConn::tracker );
            tracker = new vhtTracker( trackerDict );
#else
            tracker = new vhtTrackerEmulator();
#endif
      }
```

```
catch (vhtBaseException *e)

{
            printf("Error: %s\nPress <enter> to exit.\n", e->getMessage());
            getchar();
            return 0;


      }
      //
      // The demo loop: get the finger angles fro the glove.
      //
      vhtTransform3D trackerXForm;
      vhtVector3d position;
      vhtQuaternion  orientation;
      vhtVector3d axis;
      double baseT = glove->getLastUpdateTime();
            myfile << "BaseT: " << baseT << "\n";

      while(true) {
      // update data from the physical device
      glove->update();
      tracker->update();
      // Get update time delta
      myfile << "deltaT: " << glove->getLastUpdateTime() - baseT << " ";
      cout <<"deltaT:"<<glove->getLastUpdateTime() - baseT<<"\n";
      int numSensors = 24;
      for( int finger = 0; finger < /*GHM::nbrFingers*/numSensors; finger++ )
{
            if(finger!=7&&finger!=23){
                  myfile << glove->getData(finger)<< " ";
            }
      }
       myfile << "\n";
     myfile.close();
       myfile.open ("C:\\example.txt",fstream::out|fstream::app);
      // wait for 100ms
#if defined(_WIN32)
      Sleep(100);
#else
      usleep( 10000 );
#endif
      }
      return 0;
}
```

APPENDIX C.

MATLAB TRANSFORMATION CODE

APPENDIX C. MATLAB TRANSFORMATION CODE

```
clc

transAngles =[];
data = xlsread('ArmWrist.csv');
for i = 1:size(data,1)
    ARx = data(i,6);
    ARy = data(i,7);
    ARz = data(i,8);
    ATx = data(i,9);
    ATy = data(i,10);
    ATz = data(i,11);

    WRx = data(i,17);
    WRy = data(i,18);
    WRz = data(i,19);
    WTx = data(i,20);
    WTy = data(i,21);
    WTz = data(i,22);

    if ARx < 0
        ARx = -ARx;
    end
```

```
if WRx < 0
    WRx = -WRx;
end
if ARz < 0
    ARz = -ARz;
end

N_A(1,1) = cos(ARy)*cos(ARz);
N_A(1,2) = -cos(ARy)*sin(ARz);
N_A(1,3) = sin(ARy);
N_A(2,1) = cos(ARx)*sin(ARz) + cos(ARz)*sin(ARx)*sin(ARy);
N_A(2,2) = cos(ARx)*cos(ARz) - sin(ARx)*sin(ARy)*sin(ARz);
N_A(2,3) = -cos(ARy)*sin(ARx);
N_A(3,1) = sin(ARx)*sin(ARz) - cos(ARx)*cos(ARz)*sin(ARy);
N_A(3,2) = cos(ARx)*sin(ARy)*sin(ARz) + cos(ARz)*sin(ARx);
N_A(3,3) = cos(ARx)*cos(ARy);

W_A(1,1) = cos(WRy)*cos(WRz);
W_A(1,2) = -cos(WRy)*sin(WRz);
W_A(1,3) = sin(WRy);
W_A(2,1) = cos(WRx)*sin(WRz) + cos(WRz)*sin(WRx)*sin(WRy);
W_A(2,2) = cos(WRx)*cos(WRz) - sin(WRx)*sin(WRy)*sin(WRz);
W_A(2,3) = -cos(WRy)*sin(WRx);
W_A(3,1) = sin(WRx)*sin(WRz) - cos(WRx)*cos(WRz)*sin(WRy);
W_A(3,2) = cos(WRx)*sin(WRy)*sin(WRz) + cos(WRz)*sin(WRx);
```

```
W_A(3,3) = cos(WRx)*cos(WRy);


ATrans = [  N_A(1,1)  N_A(1,2) N_A(1,3) ATx; N_A(2,1) N_A(2,2) N_A(2,3) ATy; N_A(3,1)
N_A(3,2)  N_A(3,3) ATz; 0 0 0 1 ];


WTrans = [  -W_A(1,1)  W_A(1,2) -W_A(1,3) WTx; W_A(2,1) W_A(2,2) W_A(2,3) WTy; W_A(3,1)
W_A(3,2)  W_A(3,3) WTz; 0 0 0 1];


WA = WTrans*inv(ATrans);


theta2 = asin(WA(1,3));

theta1 = acos(WA(3,3)/(cos(theta2)));

theta3 = acos(WA(1,1)/(cos(theta2)));


%S1 = -N_A(2,3)/cos(asin(N_A(1,3)));

%C1 = N_A(3,3)/cos(asin(N_A(1,3)));

%XRotation(i) = atan2(S1,C1);
```