ME 446 Laboratory #1
Kinematic Transformations

Report is due at the beginning of your lab time the week of February 17th. Partners must submit their own report. Lab sessions will be held the weeks of January 27th, February 3rd, February 10th.

Objectives

- Introduce the TMS320F28335 DSP controller for the CRS robot arm and using Code Composer Studio IDE to program the TMS320F28335 DSP with C.
- Perform forward kinematics analysis for CRS robot arm following the Denavit-Hartenberg (DH) convention
- Practice calculating transformation matrix with Robotica
- Derive inverse kinematic equations for the CRS robot arm
- Use the provided Code Composer Studio project to verify your solution

NOTE: CRS robot arm has five motors/joints, in this class we are only using/controlling the first three.

Part 1: forward kinematics

In this part you will derive solutions to the forward kinematics problem for the CRS robot arm. First find parameters of the CRS robot arm following D-H convention, then verify the theoretical result using a given C program in Code Composer Studio.

Physical Dimension

Below is the physical dimension of the CRS robot arm used in this lab. Use this when deriving D-H parameters.

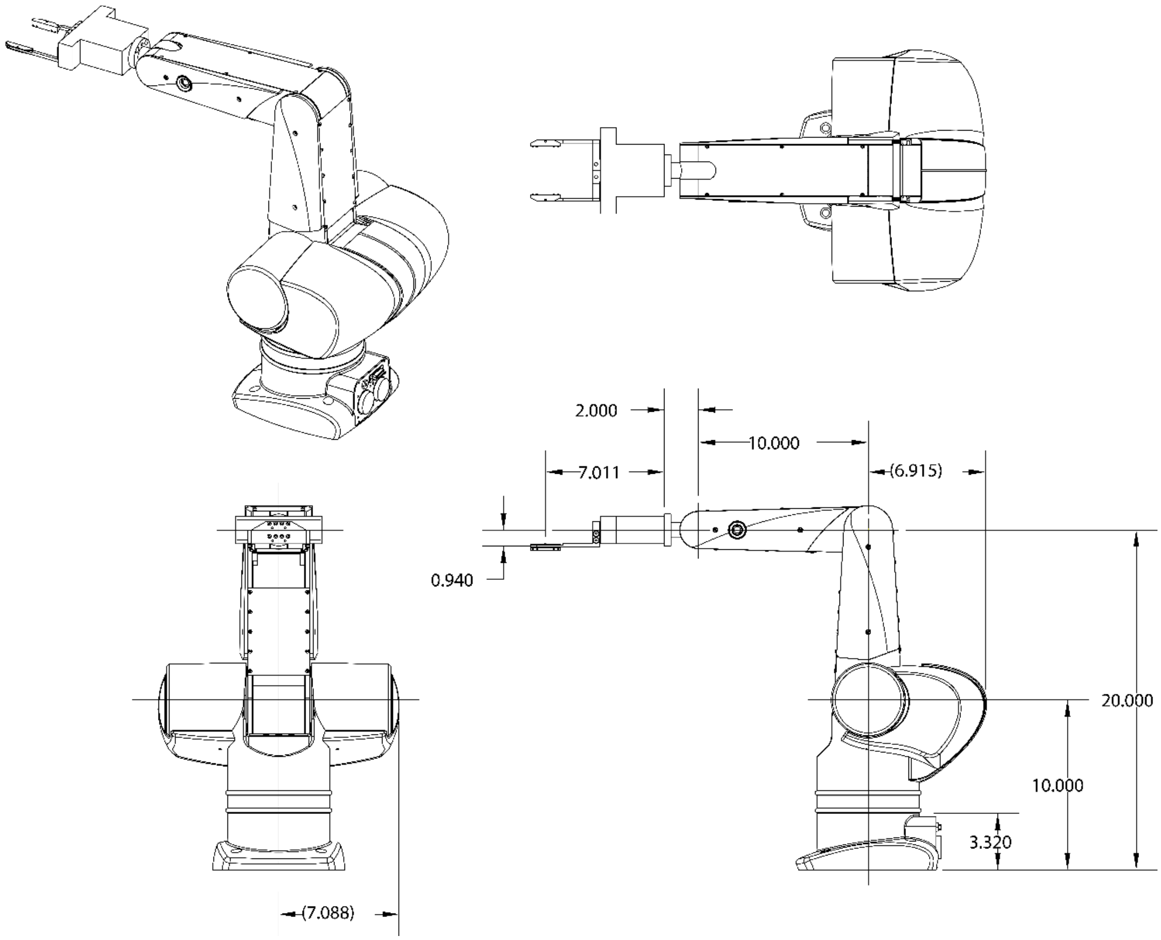

Figure 1.1 Physical Dimensions of CRS robot arm

Theoretical Solution

Find the forward kinematic equations for the CRS robot. In particular, we are interested only in the position of the end effector. We will use Robotica to find expressions for each of the three components of d_3^0.

Verification

For any provided set of joint angles $\{\theta_1, \theta_2, \theta_3\}$, we want to compare the calculated position of the end effector to the actual position of the end effector. Our C code will print the calculated position and with rulers we will roughly measure the actual positon for the comparison.

Procedure

1.1 SVN Check-out

For your lab assignments, all the given code is stored in a SVN repository.

1. Create a folder under C:\ (using your NetID or whatever), then inside it create two folders named “workspace” and “repo”

2. Right click on "repo" and select SVN checkout. In the check-out window type in your individual class repository: file:///N:/labs/ME446/ME446Repository and select "OK".

1.2 Code Composer Studio (CCS)

In this portion of the lab, you will be modifying a given Code Composer Studio project to read joint angles of your CRS robot arm.

1. Open Code Composer Studio, select the "workspace" folder created in the last step.
2. Click project → Import New CCS Project, click "Browse" then select C:\your folder\repo\trunk, choose the project named "CRSRobot" and click "finish".
3. Run the project and power the robot arm according to the procedure found in trunk/Procedure for Running Robot.docx
4. After running the CCS project, your TA will show you how use an application called Tera Term to display joint angles from your robot arm (or whatever else you would like to print for debug purposes). Your TA will also show you how to define offsets in your code such that the natural resting position of the robot arm is home but not motor zero angle positions.
   NOTE: output of optical encoders will be reset to 0 every time the CCS project is downloaded and run on the DSP (or the code restarted). Make sure your robot arm is at the home position before running your C program.
5. Additionally, we have created some MATLAB functions that can communicate data back and forth between MATLAB and the DSP controller through serial port COM1. If you look at the top of lab.c you will see two global variables, *whattoprint* and *theta1array*. Before these variables there is a #pragma statement that tells the linker to locate these variables in a special memory section that the MATLAB functions can look in and find all the variables MATLAB can either write to or read from. For these functions to work you MUST HAVE THE DSP RUNNING and you MUST CHANGE THE CURRENT DIRECTORY of MATLAB to c:\<yourcreateddirectory>\<yourrepositorydirectory>\trunk\CRSRobot\matlab. With MATLAB in this directory it can locate the project files it needs to discover the location of these special variables. The three functions you will be using are ME446_serial_ListVars, ME446_serialwrite, ME446_serialread. At MATLAB's command change directory to this "CRSRobot\matlab" directory in your repository. Check out help on each of these functions by typing "help ME446_serialread" for example. Also before you try these functions make sure COM1 is connected to the serial port cable labeled MATLAB. With guidance for your TA, experiment with each of these functions by first playing with the two given variables *whattoprint* and *theta1array*. Then add one more float array to save 100 theta2 values and another float variable that performs something different on the DSP controller when it is changed from 0 to a positive number.
6. In addition there is a Simulink interface that allows you to upload 4 32 bit integers from the DSP and download 7 16 bit integers to the DSP every 5ms. Probably the most useful use of this Simulink interface will be the saving of response data through the uploading of the 4 32 bit integers. To upload floating point numbers we multiply the floating point number by 10000 and upload it as an integer and then when it is received in Simulink it is divided by 10000 to give a float number but with

at the most four decimal places of precision. Feel free to change this 10000 factor if you need more precision but keep in mind the maximum number a 32 bit integer can store is 2^31 – 1, 2147483647. Notice at the top of lab.c there are four float variables, Simulink_PlotVar1, Simulink_PlotVar2, Simulink_PlotVar3, Simulink_PlotVar4. To upload data to Simulink you simply have to write values to these variables every time the lab() function is called and automatically the RS 232 serial interrupt function will transfer these values to Simulink when requested. Note, due to the speed of the serial port, Simulink will only request the values every 5 ms. The lab() function is called every 1 ms., so only every 5th number written to these variables will be uploaded to Simulink. Since Simulink is performing the request and the lab() function does not know which sample the request will occur, we write to the variables every millisecond so the latest data is always ready to spend. Go to the bottom of the lab() function and you will see that three of the Simulink variables are being assigned the three motor angles. Also search in user_CRSRobot.c for the four Simulink variables and you will see that they are being multiplied by 10000 before being sent over the serial port. Test the Simulink upload by in Matlab changing directory to "CRSRobot\matlab". Then launch the Simulink file "simulink5ms_plotAndGains.slx". Make sure that the robot controller is running your DSP code and then start Simulink. The first time you run the Simulink file it will need to build itself so be patient until it starts. Once it starts you should see the motor angles being plotted. Move the robot joints to see the three data streams changing. Also have your TA show you how you can have Simulink save the uploaded data to Matlab variables. If you would like you can also play around with downloading values to the DSP. Your TA can show you the variables that are updated when you download values.

7. For one more initial exercise with the robot arm, determine the positive direction of the robot joint motor's angle and the positive torque direction of each joint motor. Produce a stick figure picture showing the positive direction of the three joint angles and torques. To do this you will slightly modify the code in the function "lab" in your lab.c file. The function *lab(float theta1motor,float theta2motor,float theta3motor,float *tau1,float *tau2,float *tau3, int error)* is called by the supporting code once every millisecond. It is passed the radian value of each of the three motor joint angles and references to the three torque commands for the joint motors. Acceptable values for tau1 through tau3 are a real number between -5 and 5. The unit of this -5 to 5 number is directly proportional to torque. For this exercise, one at a time, change the line of code that assigns an open loop torque to each of the motors and run the robot while watching the theta angles displayed in Tera Term. For example change the line of code *tau1 = 0.0; to *tau1= 1.0;. Debug your code and perform the required steps to run the robot arm and you should see joint one turn in its positive direction until it goes outside of the safety region and then will stop. Repeat for joint two and three zeroing the other joints so only one joint moves at a time.

1.3 Physical Implementation

1.2.1 Before practicing D-H convention, proper coordinate frames have to be defined. Figure 1.2 gives a sketch of the CRS robot arm with all the z axis of joints fixed. Label the rotation directions of all joints according to the positive direction you found in 1.2 above.

1.2.2 Use right hand rule to determine the direction of all z axis. Then, on figure 1.2, complete the D-H frame by properly adding x axes.

1.2.3 Finish the D-H table below. Refer to Figure 1.1 for physical dimensions of CRS robot arm.

Joint	a_i	α_i	d_i	θ_i
1				
2				
3				

1.2.4 **Show TA your D-H table.**

1.4 Theoretical Solution

1.3.1 Work with Robotica in Mathematica to find the forward kinematics of the CRS robot arm. First, during lab, perform all the steps of the Robotica tutorial to give you an introduction to using Robotica. Appendix A has the link to this tutorial along with a link to a more detailed document that may help you in future labs.

1.3.2 Modify the robotica example file with the DH parameters for the CRS robot. Use "Shift Enter" to reevaluate each cell and properly display the forward kinematic equation for the CRS. Find $T_3^0, Jacobian\ and\ J^T$. Note the Jacobian is the variable J created by the FKin[] function.

1.3.3 These equations will be a function of the thetas defined by the D-H method. The angles of each of the three motors of the robot link will not necessarily be the same angles as define by the D-H method but we do know that the D-H thetas can be calculated by a linear combination of motor angles. The easiest way to find the relationship between theta1, theta2 and theta3 and motor1_theta, motor2_theta, motor3_theta is to move the linkage to different 90 degree joint positions and record in a table the motor theta values and the values of the D-H thetas. To find the linear combinations for motor thetas create 3 unknowns that can be solved for using the different 90 degree points. So for example using some constants c1, c2, c3 set theta3 = c1*motor_theta2 + c2*motor_theta3 + c3 create 3 or 4 equations using your above table values of thetas and motor_thetas and solve for the c1, c2 and c3 constants. You can do this using Matlab or it may be possible for you to just look at three or four equations and figure out the values of c1, c2 and c3.

1.3.4 Now that you have equations for the D-H thetas in terms of the motor thetas, substitute into your robotica file these equations for theta1, theta2 and theta3 given motor_theta1 2 and 3. Rerun the robotica commands and then you will have the forward kinematic equations and Jacobian for the robot using the motor angles.

1.3.5 **Show TA your equations.**

1.5 Verification

1.4.1 For demonstration your TA will ask you to move your robot joints to different sets of joint angles $\{\theta_1, \theta_2, \theta_3\}$. Your C program must print to "Tera Term" these joint angles along with the calculated x,y,z position of the end effector's joint.

1.4.2 Using a ruler, measure and compare the position of the end effector to the x,y,z position printed in the serial port terminal.

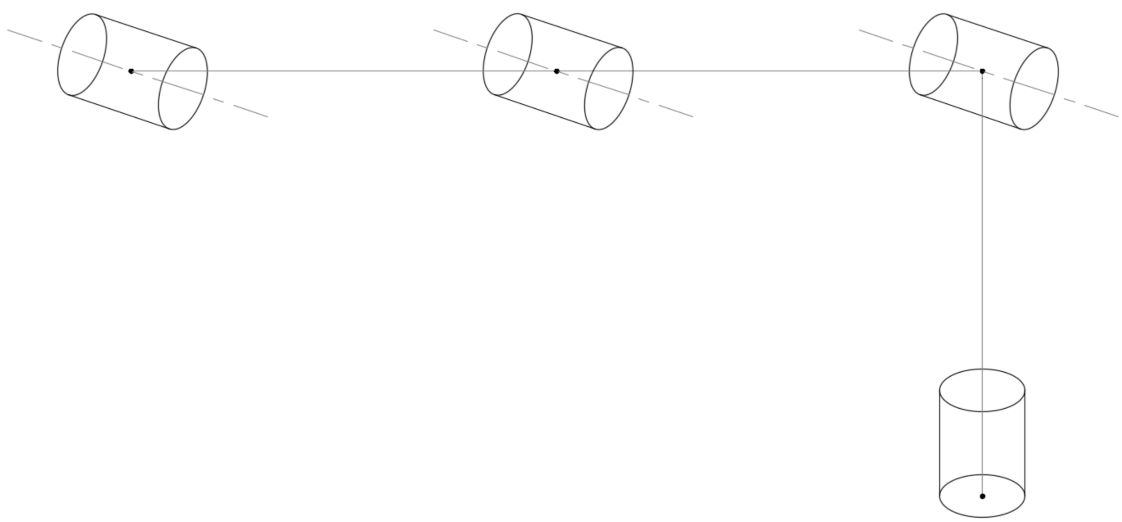

Figure 1.2 CRS robot (DH frames to be assigned)

Part 2: inverse kinematics

Geometric Approach

Given a desired point in space (*x, y, z*), write three mathematical expressions that yield values for each of the joint angles. For the CRS robot, there are (in general) two solutions to the inverse kinematics problem. We will implement only the *elbow-up* solution.

Verification

For any provided point in space (*x, y, z*), we want to compare the joint angles of the robot arm indicated by your inverse kinematics equations, to what your CCS program gives after manually moving the end effector to the specified point.

Procedure

2.1 Theoretical Solution

2.1.1 Establish the world coordinate frame (frame *w*) centered at the center of the CRS's base. The x_w and y_w plane should correspond to the surface of the table, with the x_w axis straight ahead of the robot arm with θ_1 equal to zero. Axis z_w should be normal to the table surface, with up being the positive z_w direction and the surface of the table corresponding to $z_w = 0$.

2.1.2 Given a set of (x, y, z) coordinate, solve for the corresponding joint variables $\{\theta_1, \theta_2 . \theta_3\}$ in geometric approach. Write down the three mathematical expressions:

$$\theta_1(x_w, \mathrm{y}_w, z_w) = ?$$

$$\theta_2(x_w, \mathrm{y}_w, z_w) = ?$$

$$\theta_3(x_w, \mathrm{y}_w, z_w) = ?$$

2.1.3 You now have inverse kinematic equations to give you your defined D-H θ's. But in the control of the robot arm we will be controlling the individual motors of the robot arm. Use the equations you found in the forward kinematic calculations that equate the D-H θ's to the motor θ positions to rewrite the equations to solve for θ_{M1}, θ_{M2} and θ_{M3} given x,y,z.

2.2 Verification

For demonstration, your TA will ask you to move the robot arm to multiple x, y, z positions. Your C code should first use the forward kinematic equations from Part 1 to find the x, y, z position of the end effector given the measured motor angles. Then to verify that your inverse kinematic equations are correct, take this calculate x, y, z position and use it in your inverse kinematic equations to find calculated motor angles. These angles should match the robot's current measured motor angles. Your C code should print to "Tera Term" the motor's measured angles, the x, y, z position calculated by forward kinematics, DH angles and the motor angles calculated with your inverse kinematic equations.

Report:

Produce a lab report that details all the math used to calculate forward and inverse kinematics for the first three joints of the CRS robot arm. This can be hand written BUT must be readable. When explaining the forward kinematic derivation, make sure to explain how each DH parameter was found even if the parameter happens to be zero. Also detail steps taken to find the relationships between motor angles and D-H angles and note the positive torque direction of the joint motors. Pictures and sketches are a very important part in explaining your work and analysis. Hand sketches will be accepted BUT make sure they are neat and readable. Also add to your report the COMMENTED code you wrote in Lab1.c (or whatever you call your file). This report should have enough detail in it that a new student in this class would be taught how to perform the different tasks in this lab assignment.

Appendix A:

Small Robotica Manual/Tutorial

Full Robotica Manual