EE209

Robotics: Design, Manufacture, and Control

Lab 1 report

Dong Ma (104561968)

10-07-2016

Division of labor: everything done by Dong Ma alone

Abstract

Inverse kinematics and forward kinematics area widely used in numerous fields of robotic design and computer graphics. This technique has found a good utilization especially as a tool for animation of articulated structures. The insight of forward and inverse kinematics is based on the fundamental knowledge in kinematics generally. This report presents an overview of the method used to find the inverse kinematics and forward kinematics of a simple mechanical linkage. The introduction of the differences between the forward and inverse kinematics are elaborated at the beginning of the reports. The basic analysis of the model chosen are explained in the background section. Then the methods used to solve the problem and the results obtained are included in the method section. Finally, the conclusions of this lab are discussed.

1. Introduction

Kinematics in robot system is mainly divided into two categories – *forward kinematic* and *inverse kinematics*. In the forward kinematics problem, the transformation describing the position and orientation of a tool, or end effector, is determined by known joint variables. Joint variables are associated with a particular axis, or joint, and are denoted by qi. The two common types of joints used in robot manipulators are revolute and prismatic joints. For a revolute joint, qi is the angle of rotation, while for a prismatic joint, qi is the joint displacement (di).

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired position of the end-effector. Specification of the movement of a robot so that its end-effector achieves a desired task is known as motion planning.

In this lab, we are asked to find a mechanical linkage and implement the inverse kinematics and use the implementation to simulate a sample mission of the robot.

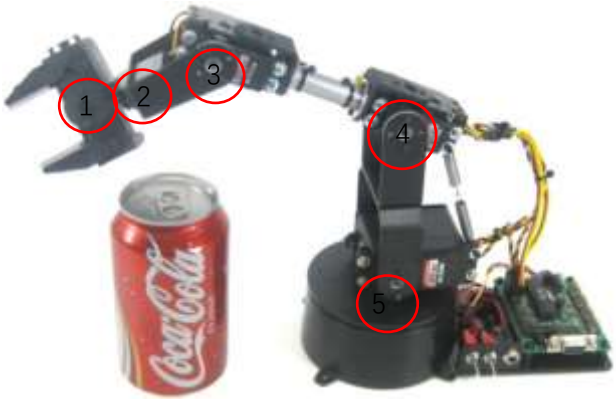

Figure 1

I choose a robotic arm (Figure 1) with 4 DOF as the sample of my research. This robotic arm has a total of 5joints as labeled and an end effector located at end that enable the arm to grab small objects such as a Coke can. All four joints of this arm except joint 5 are revolute joints which allow corresponding sub-arm to rotate. According to the design of this robot, each joint has its own range of rotation which is important in calculating the kinematics. The detailed behavior and restrictions of each joint will be addressed in the background session of this report.

2. Background

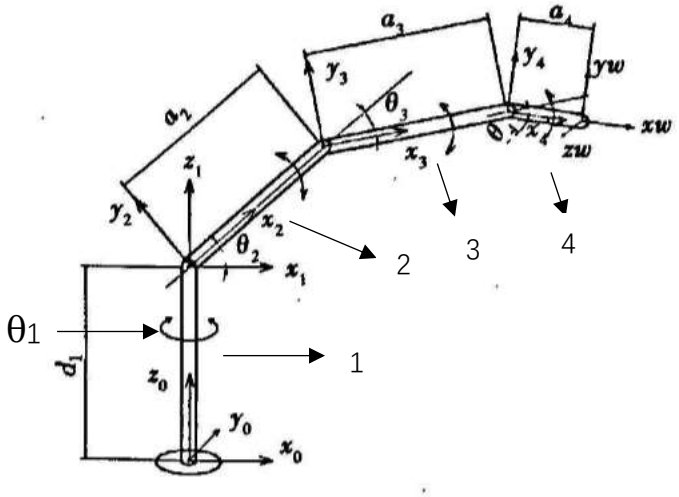

Figure 2

Figure 2 shows the schematic diagram of the robotic arm we picked. To simplify the analysis, the effect of the gripper is ignored making the robotic arm a four degree of freedom, a four-joint spatial manipulator. Therefore, joint 1 and joint 2 are viewed as a single joint. Joint 5 and joint 4 are connected through a bar of distance d1. Joint 4 and joint 3 are connected through a bar of distance a2. In the same manner, joint 3 and joint 2 are connected by a bar of distance a3. Joint 1 is not considered as a factor to influence our system because we set it coincide with joint 2. Frame 0 (x0, y0, z0) is chosen as the base frame of the system and bar 1 is aligned with the z-axis of the frame 0. Frame 1's z-axis (z1) is aligned with the z-axis of Frame 1 and bar 2 is aligned with the x-axis of Frame 1. Finally, bar 3 is aligned with the x-axis (x3) from Frame 3 and bar 4 is aligned with the x-axis (x4) from Frame 4.

As mentioned before, all joints are revolute joints but not all of them are able to rotate in 360 degrees without restriction. As labeled on Figure 2, we have θ1, θ2, θ3 and θ4 to indicate the rotation angle of each joint. Because joint 5 is located on the bottom of the arm, it should be able to rotate in 360 degrees to control the turning direction of the robotic arm. According to the datasheet of this arm and some logical conjectures, θ2 should be in the range of 15~75 degrees, θ3 should be in the range 30~120 degrees and θ4 should be in range of 40~120 degrees.

3. Methods

In this section, we will compute the forward kinematics and inverse kinematics of our model. First, we need to determine the Denavit-Hartenberg parameters for each linkage. The Denavit-Hartenberg convention (DH convention) is commonly used to define reference frames for robotic serial arms. The general rule of deciding DH parameters is shown on Figure 3.

Link i	α_i	a_i	d_i	Θ_i
1	Angle between z_0 and z_1 measured about x_1	Distance from z_0 to z_1 measured along x_1	Distance from x_0 to x_1 measured along z_0	Angle between x_0 and x_1 measured about z_0
2	Angle between z_1 and z_2 measured about x_2	Distance from z_1 to z_2 measured along x_2	Distance from x_1 to x_2 measured along z_1	Angle between x_1 and x_2 measured about z_1
n	Angle between z_n and z_{n+1} measured about x_{n+1}	Distance from z_n to z_{n+1} measured along x_{n+1}	Distance from x_n to x_{n+1} measured along z_n	Angle between x_n and x_{n+1} measured about z_n

Figure 3

Link	ai	αi	di	θi
1	0	90	d1	0
2	a2	0	0	0
3	a3	0	0	0
4	a4	0	0	0

Figure 4

According to the frame systems we defined, the DH-table is constructed as figure 4 shown. Since we got the DH table of our robot system, we can then derive the individual transformation matrices for link 1,2,3,4.

$$ {}^{i-1}Ti = \begin{bmatrix} cos\theta i & -sin\theta i\ cos\alpha i & sin\theta i\ sin\alpha i & \alpha i\ cos\theta i \\ sin\theta i & cos\theta i\ cos\alpha i & -cos\theta i\ sin\alpha i & \alpha i\ sin\theta i \\ 0 & sin\alpha i & cos\alpha i & di \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Matrix 1

The general transformation matrix is shown as Matrix 1. Then the individual functions for i=1,2,3,4 can be derived. Therefore, ${}^{0}T_4={}^{0}T_1*{}^{1}T_2*{}^{2}T_3*{}^{3}T_4$. We can then define a point in our coordinate system as a starting point and define a target position, and derive the homogeneous function of the transformation function and then plot it on Matlab.

The inverse kinematics are much harder to calculate compare to forward kinematics because it includes some complicated Jacobian transformation and an iteration loop is required.
We can first set a desired destination desired=[dx, dy, dz], and then define a variable *diff* to let us know the distance between the current position and the desired position. Through an iteration loop with the monitor of *diff*, we can keep track the movement each time. If the current matrices can't achieve to the desired matrices after *n* rounds, we jump out of the loop and mark this transition as invalid. During the iteration process, we should transfer the matrix to the Jacobian form because $dp=J*dq$ (*q* is current *Jocobian* matrix, *J* is the analytical *Jacobian* matrix, *p* is the difference between the target matrix and the current matrix). Since $dq=J^{-1}*dq$, we also need to find out the inverse of the inverse of J. The formula of finding the inverse of J is shown on Figure 5.

```
J11 = -
sin(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(t
heta2)+a1);
J12 = -
cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(t
heta2));
J13 = -cos(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3));
J14 = -cos(theta1)*a4*sin(theta2+theta3+theta4);
J21 =
cos(theta1)*(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(t
heta2)+a1);
J22 = -
sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3)+a2*sin(t
heta2));
J23 = -sin(theta1)*(a4*sin(theta2+theta3+theta4)+a3*sin(theta2+theta3));
J24 = -sin(theta1)*a4*sin(theta2+theta3+theta4);
J31 = 0;
J32 =
(a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3)+a2*cos(theta2)+a1);
J33 = a4*cos(theta2+theta3+theta4)+a3*cos(theta2+theta3);
J34 = a4*cos(theta2+theta3+theta4);
J41 = 0;
J42 = 1;
J43 = 1;
J44 = 1;
J = [J11 J12 J13 J14;J21 J22 J23 J24; J31 J32 J33 J34; J41 J42 J43 J44];
det(J);
J1 = inv(J);
```

Figure 5

Finally, we can generate the pseudo code for finding the inverse kinematics.

```
/*************************************************************/
count=0;
While (count<n&&diff>0.0001) {
    // find out the distance from desired position to the current position
    Δx =desired position-current position
    // find out the inverse of Jocabian and calculate dq
    dq=J-1*dq
    // update the current position in Jocabian form
    q=q+dq
    // modify the range of angle to make sure it doesn't exceed the range
    if(..){}   if(...){}
    //get the current position
    x=f(q)
    count++;
}
Return 1 if it is possible to reach;
Return 0 if it is impossible to reach;
/*************************************************************/
```

4. Results

Forward Kinematics：

As the procedures described in the Method part, we can use Matlab to plot the result on a 3d frame as shown on Figure 6. Figure 7 shows the attainable operational state space.

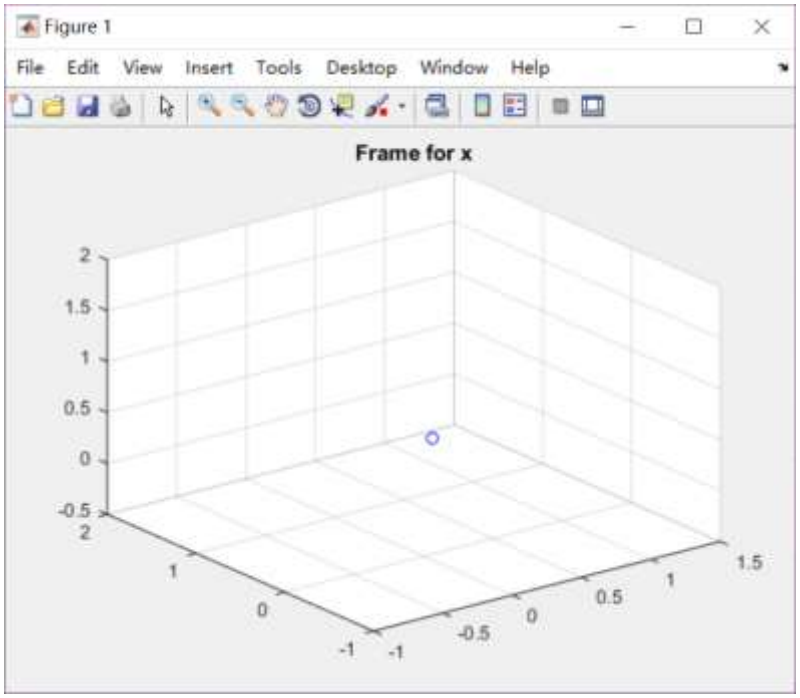

Figure 6

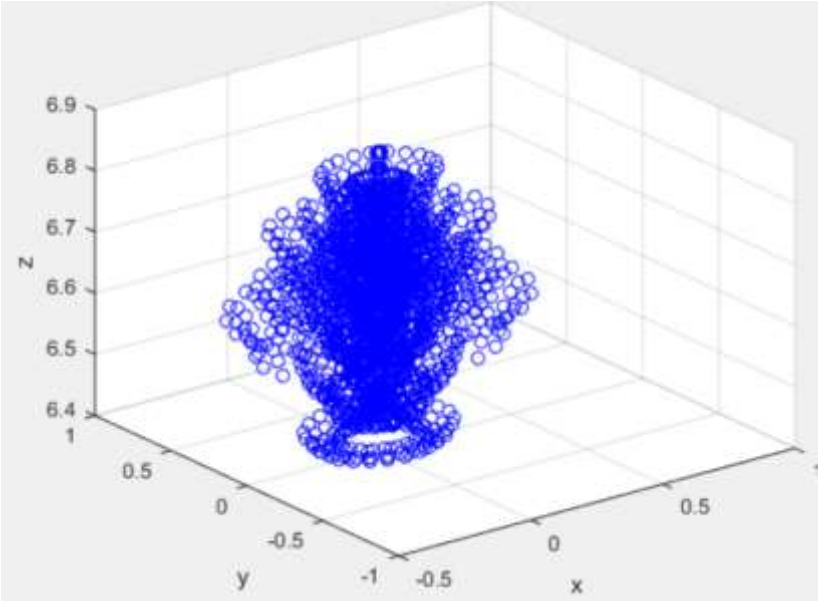

Figure 7

Inverse Kinematics：

As discussed before, the inverse kinematics of the robotic arm needed to be reached through an iteration loop. For the first case, we set the destination position to [0.7,0.4.0.2] and the x position from each iteration is shown as Figure 8.

```
    0.0174
    0.0026
    0.1990

   -0.1251
   -0.2638
    0.2163

    0.1278
   -0.0593
    0.1416

   -0.0933
   -0.1790
    0.4277

    0.1401
    0.0152
    0.1416

     0
```

Figure 8

Figure 7 shows part of x positions, the script finally returns 0 because it is unable to reach to the target positions after 500 moves as shown on Figure 9

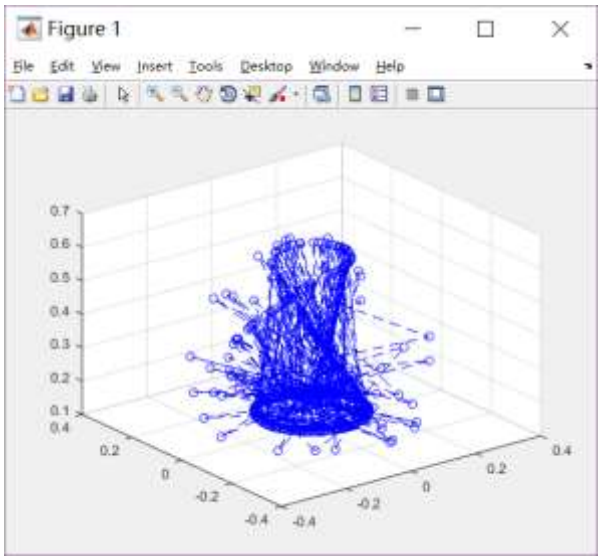

Figure 89

Next we change the target position to [0.1,0.1,0.3]

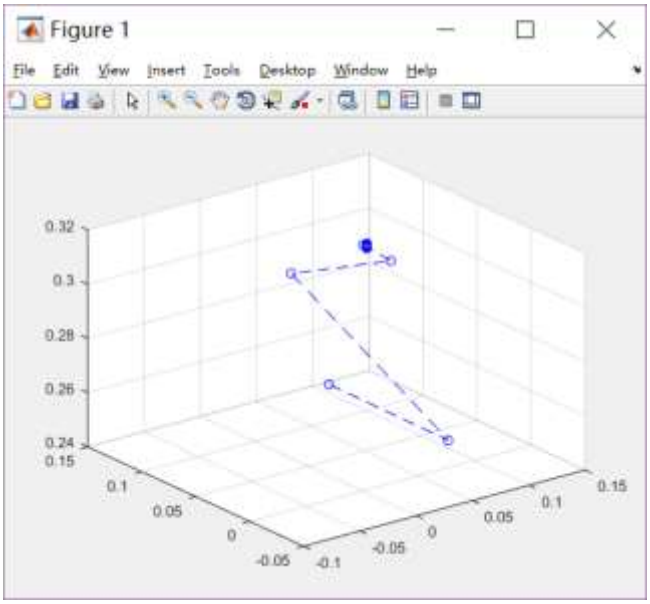

Figure 10

As shown on Figure 10, it is possible to reach to the target position from its start position. Figure 11 indicates that after 8 iteration process, the target position is reached so it returns 1.

```
0.0999
0.0999
0.3001

0.0999
0.0999
0.3001

0.0999
0.0999
0.3001
                0.0999
                0.0999
0.0999          0.3001
0.0999
0.3001          0.0999
                0.0999
0.0999          0.3000
0.0999
0.3001          0.0999
                0.0999
0.0999          0.3000
0.0999
0.3001          1
```

Figure 11

5. Conclusions

This lab demonstrates the basic principles of direct kinematics and inverse kinematics. Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters. The inverse kinematics problem, however, consists of the determination of the joint variables corresponding to a given end-effector position and orientation. Moreover, the inverse kinematics problem is much more complex than forward kinematics problem because of the multiplicity and the infinity of the solution.

This lab is definitely not an easy one and unfortunately my partner dropped the class so I have to do everything by myself. Overall, the kinematics problems of robotic system are challenging and this lab took me 4 days to accomplish. The forward kinematics problem is easier to solve because there are numerous clear examples from textbook, on the other hand, the inverse kinematics problems required strong math background to solved it successfully on Matlab and there are very few resources on Internet to learn from.

Overall, it is an unforgettable lab experience and I am glad that I learn a lot from this lab.

Reference

[1] Siciliano, B. (2009). Robotics: Modelling, planning and control.
[2] lampMohammed, A. A., & Sunar, M. (2015). Kinematics modeling of a 4-DOF robotic arm. 2015 International Conference on Control, Automation and Robotics.
[3] Hertz, R. B., & Hughes, P. C. (1993). Forward Kinematics of a 3-DOF Variable-Geometry-Truss Manipulator. Solid Mechanics and Its Applications
[4] Tsai, L. (1996). Kinematics of A Three-Dof Platform with Three Extensible Limbs.