ECE 569 Project B Report

Debasmit Das, Tongyang Liu

***Abstract*— This project aims to implement the verification of forward and generalised inverse Jacobian of a PUMA 560 robot through a Python code for OpenRave. Then the PUMA 560 is programmed to draw a locus - first a circle and then a line. The drawing should be visualized in the OpenRave GUI. The developed program is able to draw any locus given as an input to the robot. The end-effector speed is set to 0.1 m/s. Finally the performance of the robotic system is evaluated.**

I. INTRODUCTION

A. PUMA 560 Robot

PUMA 560 is serial manipulator with 6 degrees of freedom. It has 6 links and 6 revolute joints. Since the PUMA 560 has Joint four, Joint five and Joint six intersecting at a point, a closed form inverse kinematics solution exists [3]. The coordinate configurations for PUMA 560 robot is shown in figure 1. And the Denavit-Hartenberg parameters of PUMA 560 is shown in table I. Note that we set θ_i to be θ_i plus a fixed value in order to show the proper position of the manipulator.

B. Jacobian of PUMA 560

The PUMA has six revolute joints. So the joint angle vector can be represented by $\theta \in \mathbb{R}^6$. So, the joint velocity vector would be represented by $\dot{\theta} \in \mathbb{R}^6$. The task space velocities given by 3 linear velocities and 3 angular velocities can be clubbed together into a 6-dimensional velocity vector called as spatial velocity and is represented as $V \in \mathbb{R}^6$. The Jacobian is a linear mapping from the Joint Velocity vector to the Spatial Velocity. Since a Linear Mapping can be represented as a Matrix so is the Jacobian. The Jacobian is a function of the configuration i.e. $\theta \in \mathbb{R}^6$.

$$V = J(\theta)\dot{\theta} \tag{1}$$

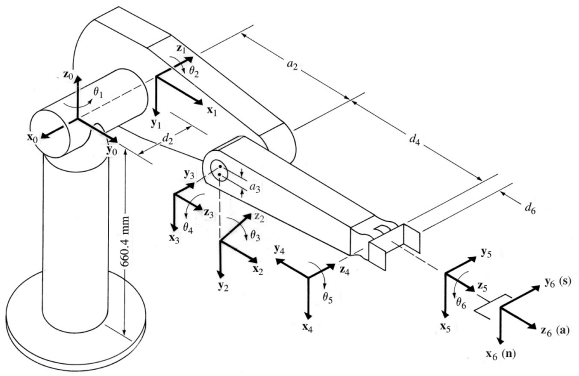


Fig. 1: PUMA 560 coordinate frame assignment.

TABLE I: PUMA robot arm link coordinate parameters

Joint i	θ_i	α_i	a_i	d_i	Joint range
1	θ_1+90	-90	0	0	-160 to +160
2	θ_2+0	0	431.8mm	149.09mm	-225 to 45
3	θ_3+90	90	-20.32mm	0	-45 to 225
4	θ_4+0	-90	0	433.07mm	-110 to 170
5	θ_5+0	90	0	0	-100 to 100
6	θ_6+0	0	0	0	-266 to 266

$J \in \mathbb{R}^{6\times6}$. Since the Jacobian is a square matrix the inverse exists. It is important to note that the inverse will not exist for singular configurations. Singular configurations for the PUMA 560 imply those $\theta \in \mathbb{R}^6$ such that $\mid J(\theta) \mid$ is zero. Therefore, it is essential to avoid singular configurations when tracking trajectories.As a matter of fact for a more general case the pseudo inverse of the Jacobian rather than inverse is used. Generally $J \in \mathbb{R}^{m\times n}$, $\theta \in \mathbb{R}^n$, $V \in \mathbb{R}^m$.In such a case the pseudo-inverse is given by the following forms.

$$J^{\dagger}(\theta) = \begin{cases} J^T(\theta)[J(\theta)J^T(\theta)]^{-1} & \text{if } m < n, \\ J^{-1}(\theta) & \text{if } m = n, \\ J^T(\theta)J(\theta)^{-1}J^T(\theta) & \text{if } m > n. \end{cases} \tag{2}$$

It can seen therefore that the second condition holds for the PUMA 560. So the inverse differential kinematics can be given by the following equations:

$$\dot{\theta} = J^{-1}(\theta)V \tag{3}$$

After this the task remains on how to find the Jacobian and Inverse and carry out the verification. The Jacobian for the two methods are verified with the equation in the text book [1].The Inverse Jacobian has been by multiplying the Jacobian and Inverse Jacobian and identity has been produced. The obvious proof that the Inverse Jacobian works is that we can draw the circle and line using the Inverse Jacobian Method.

II. MATH

A. Forward Jacobian

The Forward Jacobian of the PUMA 560 is formulated using 2 Methods. Firstly we will discuss the Vector Cross Product Method. For 6-link manipulators with revolute joints the Forward Jacobian is given by the following.

$$J(\theta) = \begin{bmatrix} \mathbf{z_0} \times {}^{\mathbf{0}}\mathbf{P_6} & \mathbf{z_1} \times {}^{\mathbf{1}}\mathbf{P_6} & \dots & \mathbf{z_5} \times {}^{\mathbf{5}}\mathbf{P_6} \\ \mathbf{z_0} & \mathbf{z_1} & \dots & \mathbf{z_5} \end{bmatrix} \tag{4}$$

All the co-ordinate representations are with respect to the base frame. In literature, this is called the space Jacobian and

will be represented by J_s. For our Puma robots the columns turn out to be of the following form.

$$z_i = \begin{cases} S_1({}^0A_i) & \text{if } i > 0, \\ S_1(I) & \text{if } i = 0, \end{cases} \tag{5}$$

S_1 is a sub matrix extraction operation that extracts the third column of a matrix till the third row. Similarly, we can formulate an equation for ${}^iP_n$.

$${}^iP_n = \begin{cases} S_2({}^0A_6 - {}^0A_i) & \text{if } i > 0, \\ S_2({}^0A_6) & \text{if } i = 0, \end{cases} \tag{6}$$

S_2 is a sub matrix extraction operation that extracts the 4th column of a matrix till the third row. Accordingly each column of the space Jacobian is calculated and then they are clubbed together to form the space Jacobian Matrix. Geometrically, z_i is the direction vector of the $(i+1)^{th}$ joint axis of the manipulator and ${}^iP_n$ is the position vector from the origin of the $(i+1)^{th}$ joint co-ordinate frame to the end-effector frame expressed in the base co-ordinates.

Now, we will list the alternative formula to obtain the Jacobian. This method is called the Differential Translation and Rotation Method. This method is derived from the fact that velocity can be thought of some differential rotation and translation over an infinitesimal period of time. All the co-ordinate representations are with respect end-effector frame. In literature, this is called the body Jacobian,J_b.

$$J_b(\theta) = \begin{bmatrix} \mathbf{J_1} & \mathbf{J_2} & ... & \mathbf{J_6} \end{bmatrix} \tag{7}$$

$$\mathbf{J_i} = \begin{bmatrix} (\mathbf{p} \times \mathbf{n})_z \\ (\mathbf{p} \times \mathbf{s})_z \\ (\mathbf{p} \times \mathbf{a})_z \\ \mathbf{n}_z \\ \mathbf{s}_z \\ \mathbf{a}_z \end{bmatrix} \tag{8}$$

$$U_i = {}^{i-1}A_6 = \begin{bmatrix} \mathbf{n} & \mathbf{s} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$i = 1, 2, ...6$$

Since the space Jacobian J_s considers the base reference frame and the body Jacobian J_b considers the end-effector reference frame. There is a relation between them that can be written as follows:

$$J_s = Ad_T(J_b) \tag{10}$$

In the above equation, $Ad_T()$ represents the adjoint mapping (include reference) with respect to T. T is the homogeneous transformation matrix relating the end-effector frame co-ordinates to the base frame co-ordinates. This is a method by which we can verify the correctness of J_b and J_s.

B. Inverse Jacobian

The Inverse Jacobian is used to calculate the joint velocities when the task space velocities are provided. The inverse Jacobian is important for tracking trajectories. Since the we have control authority on the joints of the robot it is important that we convert the variables from task space to the joint space. For the PUMA-560 manipulator, the task space is six-dimensional and the joint space is six-dimensional. Therefore the Jacobian is a square matrix and its inverse exists. Therefore, there is no need to implement a pseudo-inverse. The inverse of the Jacobian would not exist at singular points. So, we have to make sure that we choose such a task that does not force the PUMA-560 to be in a singular posture. It is important to note that majority of tasks i.e. drawing a circle or drawing a line are carried out with respect to the base reference frame. So, we will be using the inverse of the space Jacobian J_s in computing the joint velocities. So the following equation have to be followed.

$$\dot{\theta} = J_s^{-1}(\theta)V \tag{11}$$

III. SIMULATION

Our simulation was implemented by Python and was tested on OpenRave. Our task is to draw a line and a circle in a plane using two different methods,firstly the Inverse Kinematics Method and then the Inverse Jacobian Method. The two methods are then compared for performance. The Inverse Kinematics method is a position control method where the desired locus have to be sampled and the end-effector positioned at the sampled points. The Inverse Jacobian method is a velocity control method. Joint velocities are given as input to the robot manipulator and they are updated as the manipulator configuration changes. The velocity of moving along a locus is also given as input. In our case it is set as 0.1 m/s. The inverse kinematics method used is the Geometric Method since it is accurate and takes configuration indicators ARM,ELBOW,WRIST into consideration. Lets start off with discussing the Inverse Kinematics method.

A. Inverse Kinematics Method

Firstly, we will discuss the inverse kinematics method of drawing a circle. The ABOVE ELBOW and the LEFT ARM configuration has been used to plot the locus $y^2 + z^2 = r^2$ at a distance of x_0 from the robot. So in our program we sampled y co-ordinate at a rate of 0.01 m /iteration and varied it over the four quadrants till 2π to plot the circle. The GeomIK() subroutine for the Geometric Inverse Kinematics is called whenever a new sample point appears.The task space argument is sent as arguments to GeomIK().The task space or the pose vector looks as follows.

$$\mathbf{X} = \begin{bmatrix} x_0 \\ y \\ \pm\sqrt{r^2 - y^2} \\ 0 \\ \pi/2 \\ 0 \end{bmatrix} \tag{12}$$

Next we would look at the inverse kinematics method of drawing a circle. Here, the below elbow and the left arm configuration has been used to plot the locus $y = ax + b$ at a distance of z_0 from the robot.In our program we sampled the x co-ordinate at a rate of 0.01 m/iteration and varied it from a starting point x_{start} until the robot reaches the extreme i.e an external singular configuration. The GeomIK() subroutine for the Geometric Inverse Kinematics is called whenever a new sample point appears.The task space argument is sent as arguments to GeomIK().The task space or the pose vector looks as follows.

$$\mathbf{X} = \begin{bmatrix} x \\ ax + b \\ z_0 \\ 0 \\ \pi/2 \\ 0 \end{bmatrix} \tag{13}$$

After the joint angle vector is computed it is fed back into the joints of the PUMA 560. This methodology is very useful for implementing feedback position control. We set the incremental of joint angles in every step as: From Figure **??**, we can see that PUMA 560 robot moves when new joint angles are given.

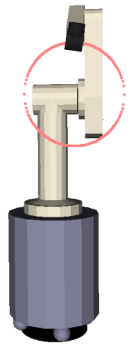
B. Inverse Jacobian Method

The inverse Jacobian method is also called the rate control method. The required task space velocities are transformed to the joint space velocities and they are then fed as input to the joints of the PUMA 560. But in the the OpenRave Environment the physics engine to takes the velocity inputs and cause rather too much disturbance to the robot [2]. So therefore we used a rather differential position control. We compute the the instantaneous joint velocities and then multiply with differential time to find the differential change in position. The change is summed up from the initial position to find the position at different times.

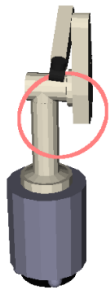
For Drawing a circle we used a parametric method namely the polar method. It has two advantages. Computing velocities is easier and representing the velocities is intuitive. It also saves lines of code as it is not required to write conditional equations for different quadrants of the circle. The disadvantage is that it is not that generalized we have to manually find the parametrized form of every locus before programming. The task space velocity for drawing the circle based on parametrized is the following

$$\dot{\mathbf{X}} = \begin{bmatrix} 0 \\ v cos(\theta) \\ -v sin(\theta) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

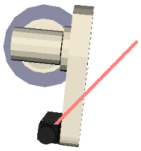
In the above equation, v is the desired speed of the robot arm. θ is the parameter that varies from 0 to 2π.
As for the line, we have to draw the locus as described in the inverse kinematics method. The Line to be followed is



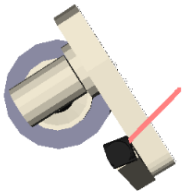
(a) Inverse Kinematics for Drawing a Circle



(b) Inverse Jacobian for Drawing a Circle



(c) Inverse Kinematics for Drawing a Line



(d) Inverse Jacobian for Drawing a Line

Fig. 2: Drawing Circle and Line with different approaches

$y = ax + b$. So accordingly the task space velocity looks like the equation below.

$$\dot{\mathbf{X}} = \begin{bmatrix} v cos(tan^{-1}(a))) \\ v sin(tan^{-1}(a))) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

In our program we allowed x as the variable. For the Inverse Jacobian Method we had to position the manipulator at the initial point of the locus using the Geometric Inverse Kinematics Method i.e. the GeomIK()

IV. Results & Discussions

We will discuss which method Inverse Kinematics(IK) or Inverse Jacobian(IJ) is suitable for drawing a line and a circle. The basic difference between a circle and a line is that circle is a closed figure and the line is not. For a closed figure, shape is very important. For a line, shape is not that vital but rather the the continuity of all the points on the line.

The important thing to note is that for shapes the position of the all the points is more important. So a position control (IK)method is apt for drawing a circle. On the other hand the IJ Method is a rate control method where a continuity of a locus is to maintained. Through simulation we have found out that the IJ method yields a circle of not accurate

shape. The main reason for this is inertia that since the IJ method involves velocity. As the arm moves with a certain velocity and as the configuration changes new velocities are computed. But since the robot was continuing with the previous velocity it continues to stay in that same velocity. So it diverts from the desired path and the shape of the circle is not retraced as accurately as the IK method does.Applications include drawing robots etc. For following a line, the robot end effector has to have a constant velocity and there is no desired turning from the locus. So the problem of inertia in drawing a line is not present. At the same time the position control method would yield a discontinuous locus of line which is not desirable. So the IJ Method is more suitable for drawing a line. Applications include car washing etc.

So, the bottom line is that the IK method is more suitable for drawing a circle and the IJ method is more suitable for following a line.

REFERENCES

[1] Fu, Gonzalez, and Lee, Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill series in CAD/CAM robotics and computer vision, 1987.

[2] Diankov, Rosen, Automated Construction of Robotic Manipulation Programs. Carnegie Mellon University, Robotics Institute, Aug 2010.

[3] D. L. Pieper. The kinematics of manipulators under computer control. PhD thesis, Stanford University, Department of Mechanical Engineering, 1968