Lab 5 Experiment Guide:
Robot Manipulator

Objectives

The goal of this lab is to control a 4 DOF robot manipulator in joint space and task space. The first task is direct teaching, which allows the robot to follow the user-defined waypoints. In this task, trajectory generation in joint space is used. The second task is teleoperation control of the end-effector position based on the keyboard input from the user. To this end, we compute Jacobian for the world linear velocity and use inverse kinematics. For both tasks, the target joint angles are finally defined and then the joint space P position controller is applied using velocity mode.

Problem statement

Problem 5A. Direct Teaching (week 1)

Complete the code that saves two user-defined waypoints for direct teaching. Generate a trajectory based on the waypoints and design the joint space P position controller to follow this trajectory.

Problem 5B. Teleoperation (week 2)

Complete the code that computes the Jacobian matrix for the world linear velocity. Based on the keyboard input from the user, control the robot's end-effector position using inverse kinematics.

Backgrounds

1. Manipulator

A manipulator is mechanically constructed from links that are connected by various types of joints. The links are usually modeled as rigid bodies. An end-effector may be attached to some link of the robot. Actuators deliver forces and torques to the joints, thereby causing motion of the robot.

2. Degrees of freedom (DOF)

In physics, the degrees of freedom (DOF) of a mechanical system is the number of independent parameters that define its configuration or state.

3. Task space

The task space is a space in which the robot's task can be naturally expressed. For example, if the robot's task is to plot with a pen on a piece of paper, the task space would be R^2.

4. Kinematics

Kinematics is studying geometrically the motion of the manipulator links related to the manipulation task in terms of position, velocity, and acceleration.

- Forward kinematics:

Forward kinematics refers to the process of obtaining the end effector's position and velocity, given the known joint angles and angular velocities. The joint velocity($\dot{q}$) can be mapped to the task velocity($\dot{x}$) using the Jacobian($J(q)$) using the following equation: $\dot{x} = J(q)\ \dot{q}$.

- Inverse kinematics:

Inverse kinematics refers to the process of obtaining joint angles and angular velocities, given the known position and velocity of the end-effector. The joint velocities can be obtained using the inverse of Jacobian.

When J(q) is not full rank, the joint angle values are called singularities. A manipulator is said to have redundancy if it has more degrees of freedom than is necessary to perform a given task.

5. Trajectory generation

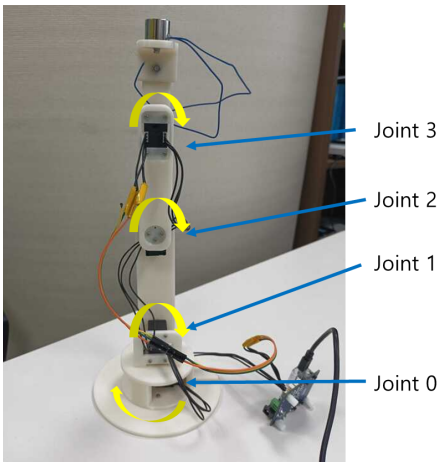
- Path: sequence of robot configuration in a particular order without regard for the timing of these configurations
- Trajectory: concerned about when each part of the path must be obtained thus specifying timing
- The goal of trajectory generation is to generate a smooth trajectory which connects a set of user-specified points.
- Polynomial interpolation
    - 1'st degree polynomial: 2 unknowns (initial and final position)
        - A straight-line path in joint space can be written $q(s) = q_{start} + s(q_{end} - q_{start}),\ s \in [0, 1]$

6. Task space motion control via inverse kinematics

Assuming the goal position is very close to the current position in the task space, the following equation holds: $q_d - q = J^{-1}(q)(x_d - x)$. Hence, by setting the target joint angle as $q_d = q + J^{-1}(q)(x_d - x)$, we can control the manipulator in task space with joint space controller.

Note) If $A \in R^{m \times n}\ (n > m)$ has full row rank, a right-inverse of A is defined as $A^{+} = A^{T}(AA^{T})^{-1}$

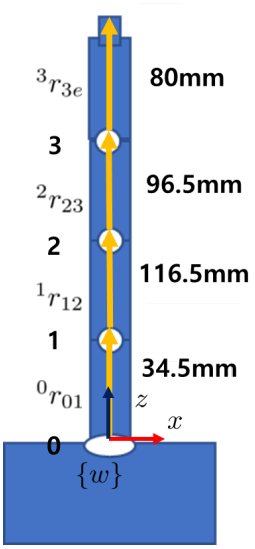
7. A 4 DOF manipulator



In Lab5, we will use a 4 DOF manipulator equipped with 4 revolute joints (joint 0 ~ joint 3). The axes of joints 1, 2, and 3 are parallel to each other, and the end-effector is the solenoid magnet that was used in Lab2. In Lab3, we controlled just one motor, so dof=1 and the dimensions of targetQ and targetQdot are 1(=dof) in the code. On the other hand, the manipulator used in Lab5 has 4 DOF. Hence, **dof=4,** and **the dimensions of targetQ and targetQdot are 4** in the code. You can check these variables in the skeleton code.

8. Computation of the Jacobian matrix

Let us consider a 4 DOF manipulator which will be used in Lab5. Then the linear velocity of the end-effector is a sum of the linear velocity contributions of 4 revolute joints. Hence, the world linear velocity of the end-effector can be represented as follows and the red rectangle denotes the Jacobian matrix for the world linear velocity. The dimension of this Jacobian matrix is **3 by 4**.



$$
\begin{aligned}
{}^{w}r_{3e} &= {}^{w}R_3\,{}^{3}r_{3e} \\
{}^{w}r_{2e} &= {}^{w}r_{3e} + {}^{w}R_2\,{}^{2}r_{23} \\
{}^{w}r_{1e} &= {}^{w}r_{2e} + {}^{w}R_1\,{}^{1}r_{12} \\
{}^{w}r_{0e} &= {}^{w}r_{1e} + {}^{w}R_0\,{}^{0}r_{01}
\end{aligned}
$$

$$
{}^{w}V_e = {}^{w}\Omega_0 \times {}^{w}r_{0e} + {}^{w}\Omega_1 \times {}^{w}r_{1e} + {}^{w}\Omega_2 \times {}^{w}r_{2e} + {}^{w}\Omega_3 \times {}^{w}r_{3e}
$$

$$
= \begin{bmatrix} {}^{w}P_0 \times {}^{w}r_{0e} & {}^{w}P_1 \times {}^{w}r_{1e} & {}^{w}P_2 \times {}^{w}r_{2e} & {}^{w}P_3 \times {}^{w}r_{3e} \end{bmatrix} \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}
$$

$$
= J(q)\dot{q}
$$

Here, ${}^{w}\Omega_i$ is the angular velocity of the i-th joint expressed in the world frame, ${}^{w}P_i$ is the joint axis of the i-th joint expressed in the world frame, ${}^{w}r_{ie}$ is a vector that points the end-effector from the i-th joint expressed in the world frame, and $\dot{q}_i$ is the joint velocity of the i-th joint (i=0, 1, 2, 3). Note that ${}^{w}\Omega_i = {}^{w}P_i\dot{q}_i$ holds.

9. Multi-thread

For teleoperation, the control loop of the manipulator should not stop while receiving the keyboard input from the user. To this end, we will use multi-thread. The following paragraph explains a process and thread.

A process can be thought of as an instance of a running program. Each process is an independent entity to which system resources are allocated and each process is executed in a separate address space. A thread is the unit of computation that runs in the context of a process. Multiple threads can exist within the same process, and they are scheduled by an OS. Multiple threads share the resources such as memory and global variable with the process.

The following example shows how to use multi-thread. In this example, the main function and the func1 function run different threads.

< Multi-thread example >

```
#include <iostream>
#include <thread> // thread library

void func1(){
   for(int i=0; i<3; i++){
      std::cout <<"thread 1 in progress \n";
   }
}

int main(){
   std::thread t1=std::thread(func1); // runs the func1 function in a new thread t1

   t1.join(); // allows the main function to wait until t1 completes its execution
   return 0;
}
```

Result.
thread 1 in progress
thread 1 in progress
thread 1 in progress

Preparation

List of components

- IBM PC with Linux
- Beaglebone set: Beaglebone embedded board, USB cable, 1 ethernet cable, 4(or More) GB microSD card (with debian Linux image in it)
- A 4 DOF manipulator
- DYNAMIXEL Starter Set
- Manipulator Control skeleton code

As the control of the manipulator is an extension of the motor control, please **review the Lab 3 material**, especially Development Environment Setup and the Articulated system class.

Lab Procedures

- See the documentation "EE405 Lab5 Procedure."

Final Report

Each student must write a final report for each experiment. Especially, if the content of the discussion is similar to that of other students, it will result in a deduction or a score of 0. The final report must include the following items:

- Purpose
- Experiment sequence (include **screenshots of the code** you implement)
- Experiment results
- Discussion
- References

The discussion questions for Lab5 experiment are as follows:

1. Compare revolute, prismatic, cylindrical, and spherical joints.
2. In Problem 5A, we utilized polynomial interpolation. Another useful trajectory generation method is the Bezier curve. Explain the advantages and disadvantages of the Bezier curve.
3. Since the DOF of the robot manipulator is four, there is one redundant DOF when controlling the position of the end-effector in Problem 5B. Give an example of the extra task that can be performed by utilizing this redundancy.
4. Choose your own discussion topic related to Lab 5 and provide an answer to it.

References

[1] Lynch, Kevin M., and Frank C. Park. *Modern robotics*. Cambridge University Press, 2017.

[2] C++ multithread, https://www.bogotobogo.com/cplusplus/multithreaded.php