Assignment 3: Serial Manipulators
CS3630 Spring 2013

Due Beginning of Class Feb 12

February 6, 2013

Introduction

In this project, you will get a more comprehensive understanding of both forward and inverse kinematics by experimenting with it in *Processing*.

1 Getting Started (No Deliverable)

You should already have the *Processing* environment installed on your machine. If not, you can download it from http://processing.org/download/. Double click the 'Processing' application, and a simple editor-based environment is launched in which you can create, load, and run applets.

1.1 Unpacking PincherKinematics applet

Unzip the PincherKinematics.zip archive. Depending on which platform you are on, you might have to uncompress the file differently, e.g., on Linux do it with the command 'unzip PincherKinematics.zip'.

1.2 Running the PincherKinematics applet

You should now have a directory called PincherKinematics, which contains an applet source file named 'PincherKinematics.pde'. Load this file by choosing: 'File -> Skechbook -> Open ->', and run it by choosing: 'Sketch -> Run'.

1.3 Play with the Sliders

To start getting a feel for the kinematics structure of the Pincher robot, play with the sliders and observe the result.

2 Forward Kinematics

The Pincher robot from the applet is a 3D RRR manipulator, with the first joint (shoulder 1) having a vertical axis, and the other two (shoulder 2, elbow) horizontal. The forward kinematics are given by the following "products of exponentials" function:

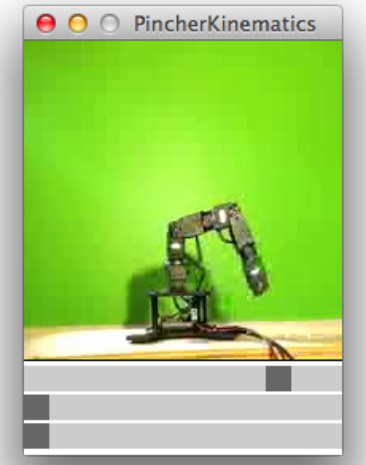
$$T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right)T_t^s(0)$$

where $T_t^s(0)$ is the pose of the tip of the last link, at rest.

Recall that for a *revolute* joint, the twist is given by $\bar{\xi} = (\bar{\omega}, p \times \bar{\omega})$, where $\bar{\omega}$ is a unit vector specifying the axis of rotation, and p is *any* point on the joint axis. In this case the exponential map simplifies to

$$\exp\left(\bar{\xi}\theta\right) = \begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\bar{\omega}\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix} \tag{1}$$

2.1 Define "Zero" Configuration



The formula for the rotation matrix $R(\bar{\omega}\theta)$ is easy whenever the rotation axis $\bar{\omega}$ is aligned with a coordinate axis. Hence, our first job is to make sure the "zero" configuration of the robot is convenient. Please rotate the robot to the position in the figure shown above, so that the robot is parallel to the image plane.

The imaging setup is pretty close to orthographic, so now we can read off coordinates straight from the image. You can do that by taking a screenshot and working in pixels.

2.2 Define Base Frame and Tool Frame

Next, we need to decide where the base frame S is going to be. Please choose the tool pose (tip of last link) in zero configuration as the base frame, in upright orientation (X pointing out to viewer, Y pointing right, Z pointing up).

Deliverable: what is the tool pose $T_t^s(0)$ at rest now? The answer requires you to decide on the tool orientation and picking one of the following three rotation matrices, for rotations around the X, Y, and Z-axis, respectively:

$$R([\theta,0,0]^T) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R([0,\theta,0]^T) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R([0,0,\theta]^T) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In your write-up, show the 4*4 matrix and how you obtained it.

2.3 Derive the Twist Exponential For Joint 1

Measure the position p of the first joint axis, and then, using Equation 1, derive the twist exponential $\exp\left(\bar{\xi}_1\theta_1\right)$ by substituting in p and picking one of the rotation matrices above.

In your write-up, show the 4*4 matrix and how you obtained it, i.e., we need something like

$$\begin{bmatrix} X & y \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\bar{\omega}\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix}$$

plus an explanation. If you get it wrong and there is no explanation, we cannot give half-credit...

2.4 Derive the Twist Exponential For Joint 2

Measure the position p of the second joint axis, and derive the twist exponential $\exp\left(\bar{\xi}_2\theta_2\right)$ in the same way, and write up the same way.

2.5 Derive the Twist Exponential For Joint 3

Measure the position p of the third joint axis, derive the twist exponential $\exp\left(\bar{\xi}_3\theta_3\right)$ in the same way, and write up the same way.

2.6 Put everything Together

Take the 4 last results and put everything (four 4×4 matrix formulas) together in one formula

$$T_t^s(q) = \exp\left(\bar{\xi}_1\theta_1\right)\exp\left(\bar{\xi}_2\theta_2\right)\exp\left(\bar{\xi}_3\theta_3\right)T_t^s(0) \tag{2}$$

Include the result in your write-up. No need to multiply out symbolically!

2.7 Extra Credit

Show what happens if you instead concatenate the matrix triplets

$$\begin{bmatrix} I & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(\bar{\omega}\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -p \\ 0 & 1 \end{bmatrix}$$

in Equation 2. What happens if you multiply neighboring translation matrices? Describe what they become, and how they relate to the "standard" way of doing things, Equation 7.2 in the **manipulators.pdf** notes.

3 Download SerialManipulator Applet (No Deliverable)

3.1 Unpacking SerialManipulator applet

Unzip the SerialManipulator.zip archive. Depending on which platform you are on, you might have to uncompress the file differently, e.g., on Linux do it with the command 'unzip PincherKinematics.zip'.

3.2 Running the SerialManipulator applet

You should now have a directory called SerialManipulator, which contains an applet source file named SerialManipulator.pde'. Load this file by choosing: 'File -> Skechbook -> Open ->', and run it by choosing: 'Sketch -> Run'.

3.3 Play with the Sliders

To start getting a feel for the kinematics structure of the Pincher robot, play with the sliders and observe the result.

4 Inverse Kinematics

When performing inverse kinematics, it's useful to know whether or not it's possible command a manipulator to a certain point. A useful concept for this is a manipulator's **workspace**. The workspace is defined as the entire space around the robot that is reachable by the manipulator tool. After messing around with the sliders for the Serial manipulator, create an image which outlines the workspace for the robot. Use an image manipulation program to draw out the boundary of the space which is reachable by the manipulator or write some code to draw wherever the tool is positioned.

After defining the workspace, write a couple sentences on the constraints which limit the workspace for this particular arm. What other types of manipulators would share the same limitations? What scenarios are well suited for this robot? Identify some scenarios where you might want to use a different style of manipulator and provide a reason why they would differ.

Deliverable

1. Provide an image of the Serial manipulator overlaid with a drawing of the boundary of the workspace.
2. Answer the questions asked regarding benefits/limitations of various manipulators and their reachable workspace.

5 Create a new Arm

Once you have messed around with the simulated arm, use it as the basis for creating your own custom manipulator. For rendering your new manipulator you can use the processing **createShape()** function. Give it a set of commands and animate it using what you've learned about kinematics. The crazier the better. Examples include snakes, torsos with arms (like Georgia Tech's Simon robot), walkers, etc. There will be extra credit for those who go above and beyond :)

Deliverable

1. A new processing sketch based on Serial Manipulator showing your new arm animated in some fashion.