LAB 1: Kinematic Linkage

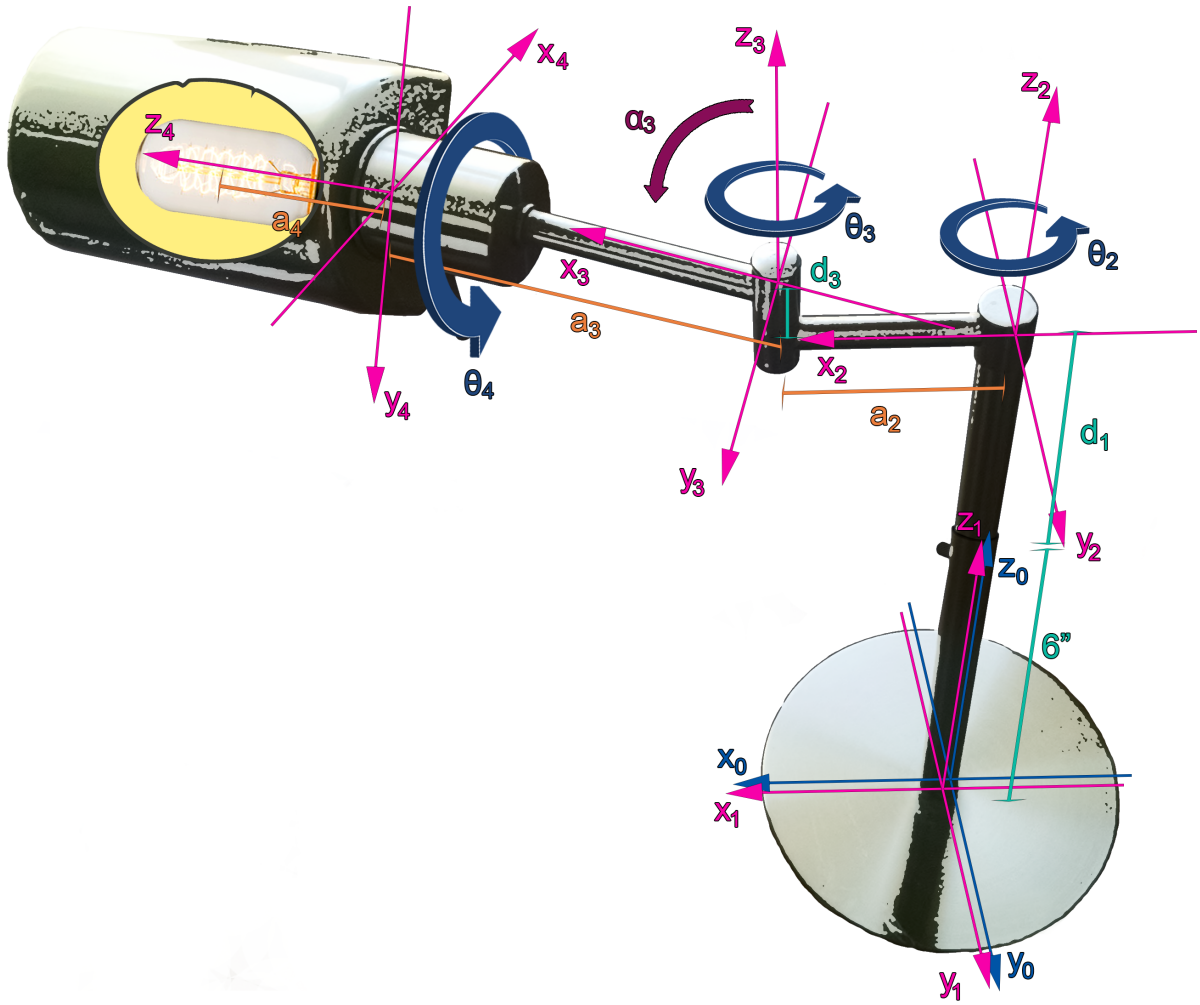
EE183DA 01.25.2018

Jonathan Chang
104853981

Introduction

While there are many models of reality, it is useful to specify robotic designs using kinematic linkages, or parameters specifying degree of freedom movements centered at the joints, and limited to one degree of freedom around conventional axes per linkage, either revolute or prismatic. This simplifies calculations to a great extent. Robots often have a control base and a remote part that is meant to interact with objects in the environment, the latter of which we call the end effector. We must manipulate the intermediate parts to affect the position of the end effector in a desirable way. Therefore, there are two directions in which this problem is defined: forward kinematics, using the configuration space parameters of the intermediate linkage to specify the position of the end effector; and inverse kinematics, using the position of the end effector to specify parameters of the linkage.

In this lab, we use the Modified Denavit-Hartenberg convention to analyze the position of a lamp. First, we define the operational space.



Schematic showing axes and operations.

As we see from the schematic, the lamp fixture body is designed so that the lamp is capable of being positioned at multiple points within a radius. An initial prismatic joint adjusts the height of the lamp, and a series of revolute joints allow the lamp to be positioned flexibly, such that it isn't confined to a fixed radius around the base. The reflector causes light to be directed toward one side, but usually down. We will analyze a linear motion of the lamp swinging its secondary arm outwards.

Methods

The Modified Denavit-Hartenberg (D-H) method of computing kinematics uses a number of parameters:

α_i specifies the angle from z_i to z_{i+1} .

a_i specifies the distance from the *i*th joint to the subsequent joint along x_i .

d_i specifies the distance from the antecedent joint to the *i*th joint along z_i .

θ_i specifies the angle from x_{i-1} to x_i .

Additionally, T_i^{i-1} is the transformation matrix, which is a function $f(\alpha_{i-1}, a_{i-1}, d_i, \theta_i)$.

Also, $T = \begin{bmatrix} R & \vec{r} \\ 0 & 1 \end{bmatrix}$, such that R is a 3x3 rotation matrix constrained to three degrees of freedom, and $\vec{r}$ is the 3-vector specifying x , y , and z .

In accordance with these variables, the D-H parameters are as such:

i	α_{i-1} (degrees)	a_{i-1} (inches)	d_i (inches)	θ_i (degrees)
0 (base)	0	0	0	0
1	0	0	d_1	0
2	0	0	0	θ_2
3	0	3"	1"	θ_3
4	90°	6"	0	θ_4
5 (end effector)	0	3"	0	0

The following mechanical constraints also apple:

$$1" \leq d_1 \leq 6"$$
$$-180° \leq \theta_2 \leq 180°$$
$$-90° \leq \theta_3 \leq 90°$$
$$-180° \leq \theta_4 \leq 180°$$

The code implementation of forward and inverse kinematics was written in Python 3, and can be found on https://github.com/jachang820/KinematicLinkages

Results

The initial position of the lamp upon setting up the modeling is:

$$[x, y, z] = [10.79", -4.5", 11"]$$

The operation space is:

$$\begin{bmatrix} 0.866 & 0 & -0.5 & 10.79 \\ -0.5 & 0 & -0.866 & -4.5 \\ 0 & 1 & 0 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

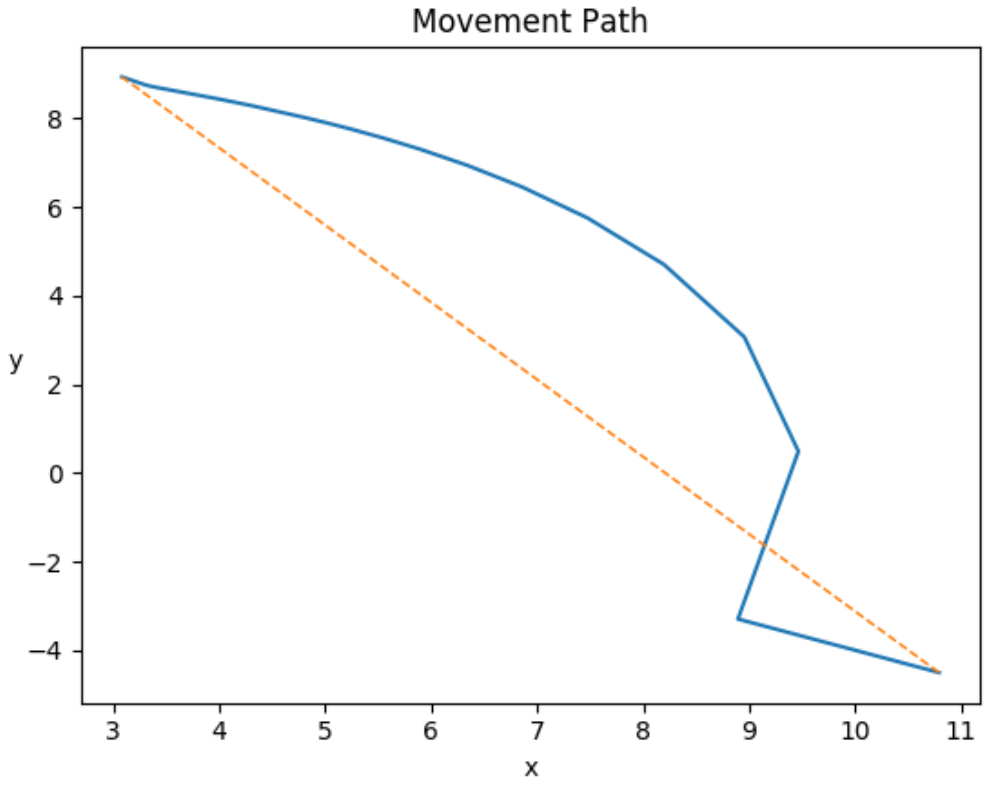
Using inverse kinematics, we specify a position such that the secondary arm swings out. The position specified is:

$$[x, y, z] = [3", 9", 11"]$$

The resulting operational space calculated is:

$$\begin{bmatrix} 0.601 & -0.094 & 0.793 & 3.08 \\ 0.784 & -0.123 & -0.609 & 8.94 \\ 0.155 & 0.988 & 0 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The trajectory between the two end points is compared to a straight line motion, shown by the following graph.



[x, y] plot between trajectory path and interpolated straight line path

The trajectory appears unstable due to only being constrained at the end points. In order to cause it to travel closer to a straight line, constraints along the intermediate points of the path are used. This can be done by interpolating *n* points between the end points of the end effector positions. Higher resolution would be straighter, but more computationally expensive.

The hardest part of the lab is the instability when dealing with inverse kinematic calculations, which fail unexpectedly. Much fidgeting with the parameters was necessary, making it a frustrating exercise. Out of the roughly 40 hours doing this lab, about 35 was spent debugging inverse kinematics code, and 2 more on making pretty graphics in Photoshop (perhaps unnecessarily). The easiest part and also the most fun was the forward kinematics code.