ROBOT ARM Simulation

Initial bleh:

**Co-ordinate systems \*\***

Whenever we describe the position or movement of an object, it is useful to have a point of reference from which the object is displaced. This point can be the origin of an orthogonal cartesian co-ordinate system and any displacements can be described as a deviation about the system along it’s x,y or z axis. This distance must be measured to a chosen point on the object, which will act as the origin of the objects own coordinate system. Thus, the displacement of object B can be referenced as its distance from object A as a vector t = [X Y Z]’.

\*insert photo of blob A to blob B distance\*

Additionally, the objects coordinate frame B may have also rotated with respect to the current coordinate frame A. These rotations could be defined as rotations about the x,y and z axis which will be referred to as phi, theta and psi respectively.

It can be shown that any arbitrary rotation can be succinctly expressed as a product of elemental rotations about the x,y and z axis.

These elemental rotations can be expressed by a skew symmetric rotation matrix R = [matrix for x] , R = [matrix for y] and R = [matrix for z].

As shown by [REF] , these rotations can be generated by R = matrix exponential (skew (angle))

There is an issue however as these rotations are not commutative. A rotation of XYZ is not the same as ZYX.

The ZYX SO3 convention will be used.

This will be denoted by R\_AB = RotZ\*Roty\*Rotx , which denotes the rotation of B with respect to A.

This can also be decomposed into its elementary rotations by rot2euler.m **(explain math)**

Thus, the position X Y Z and orientation phi theta psi of the object B, which is known as the pose of the object, can be expressed with respect to coordinate system A as:

p = [X Y Z phi theta psi]

which can be expressed as a homogenous transformation matrix

T\_AB = [ R\_AB t\_AB

0 1 ]

**Homogenous transformations**

If there were an additional object C and we knew the transformation matrix between A and B T\_AB and B and C T\_BC , we can determine the transformation T\_AC by multiplication:

T\_AC = T\_AB \* T\_BC

which gives us the pose of object C with respect to A

**Kinematics \*\***

A robotic arm is a series of joints and links which come together to form an open serial kinematic chain. The last joint of the chain is known as the end effector and the goal of kinematics is to determine the pose of the end effector with respect to the base of the robot.

This necessitates determining the transformation matrices between each joint of the robot. This method is error prone and cumbersome as it involves defining and computing transformation matrixes. Therefore, the denavit hartenberg convention is used instead which standardizes the placement and spatial relations between coordinates in mechanisms.

**Denavit Hartenberg convention**

This is (**plz explain)**

And thus, using purely revolute joints, the pose of joint 1 with respect to joint 0 is given by

p\_1 = [R\_01 t\_01

0 1]

**Forward Kinematics**

Thus, using the denavit hartenberg convention the following forward kinematic model was developed :

\*\*image of drawing\*\*

With the following DH parameters

\*\*table of params\*\*

And the transformation matrices between the base and each joint were:

T01

T12

T23

T34

T45

T56

The pose of the end effector could be determined by finding the transformation from the base to the final joint:

T06 = T01\*T12\*T23\*T34\*T45\*T56

And thus the relation p = T\_06 (q) where q = [theta1 theta2 theta3 theta4 theta5 theta6]’ are the joint angles of the robot.

**Inverse Kinematics / Differential kinematics**

Solving the Forward kinematic problem to find the joint angles required for a certain pose is useful. However, tasks are typically prescribed in cartesian space rather than joint angles. Thus, a method is required to prescribe an end effector pose in the task space which will return the joint angles required to achieve that pose. This problem is known as inverse kinematics.

This problem can be solved analytically for simpler robot configurations or when the task can be broken down (e.g. spherical wrist). However, for more complex cases it requires a numerical solution.