HW 4

Wednesday, September 28, 2022 12:39 PM

MeEn 537 Homework #4

1. Download the provided Python files from Learning Suite and put them in the same homework folder as before. Unlike our previous homeworks which used Jupyter notebooks directly, this python file should be easier to debug, while still having the ability to run individual cells (like the Jupyter notebooks). Implement the changes to your "transforms.py" file as described in the provided "hw04_test_FK.py". Also put "kinematics.py" in the same folder.

   (a) Using, "hw04_transforms_SO3.py," copy these functions into your "transforms.py" file, and implement all of the code marked with "TODO" so it can convert any rotation matrix to RPY, axis/angle, or a quaternion, and back. Check your answers in the examples given in "hw_04_test_FK_notebook.py". Each one below counts as its own problem (except for vii, which is just informational).

      i. implement "R2rpy" function
      ii. implement "R2axis" function
      iii. implement "axis2R" function
      iv. implement "R2q" function
      v. implement "q2R" function
      vi. implement "euler2R" function
      vii. Generating code to convert Euler angles to rotation matrices is pretty straightforward (euler2R), but going the opposite way (converting any arbitary R matrix to Euler angles) is nontrivial. We will provide this function in the solution, but you are not expected to implement it yourself.

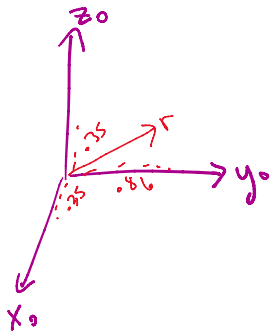
   (b) In the "kinematics.py" file, we will next implement some functions and complete a class that will make modeling a full robot arm much easier. Make sure to read the instructions in the code, including the hints.

      i. Complete the two "TODO" sections in the "dh2AFunc" function.
      ii. Complete the "TODO" in the "__init__" function for the SerialArm class. This generates a function that will calculate A(q) for each set of DH parameters in our DH parameter table.
      iii. Complete the "TODO" in the "fk" function to enable modeling the forward kinematics of a robot.
      iv. At the bottom of the "kinematics.py" file is an example of how this class can be used. Because this file has a "main" argument at the bottom of the file, if the file is run using python, it will attempt to execute that code. Alternatively, you could import "kinematics" and do the same example in a new file (similar to how we use "transforms.py". If you can run the code, visualize the two coordinate frames, and they all make sense, then you get full points.

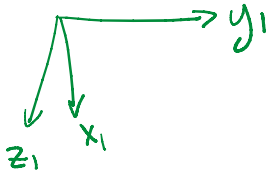
2. Let R represent a rotation of 90° about y_0 followed by a rotation of 45° about z_1.

   (a) Find the equivalent axis/angle to represent R. Sketch the initial and final frames and the equivalent axis vector *r*.

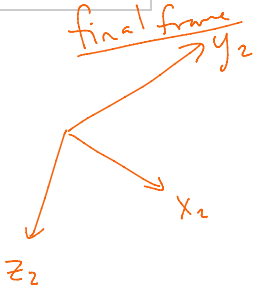
$[1.7178 \quad 0.3574 \quad 0.8629 \quad 0.3574]$



$R_{y_0}(90°)$



$R_{z_1}(45°)$



(b) Also find the equivalent quaternion representation. [0.6533 0.2706 0.6533 0.2706]

3. Create three symbolic variables in sympy to represent roll, pitch and yaw angles, (ψ, θ, ϕ) and do the following (assuming a rotation about z, y, x for current axis convention):

   (a) compute a rotation matrix (you may need to re-use code from the solution for HW 02 for symbolic versions of rotx, roty, and rotz). See code

   (b) Use this to transform a unit vector in the z-direction. See code

   (c) Looking at the elements of the rotation matrix devise an algorithm to determine the roll, pitch and yaw angles. Hint – find the pitch angle first.

$$\begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi) & \sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi) \\ \sin(\psi)\cos(\theta) & \sin(\phi)\sin(\psi)\sin(\theta) + \cos(\phi)\cos(\psi) & -\sin(\phi)\cos(\psi) + \sin(\psi)\sin(\theta)\cos(\phi) \\ -\sin(\theta) \; R31 & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

To find θ:

$$\theta = \arcsin(-R31)$$

To find ϕ:

$$\phi = \arccos(\cos(\theta))$$

To find ψ:

$$\psi = \arcsin(\cos\theta)$$