Machines in Motion Lab Report

(Deadline: 2016-01-11)

Before the lab

A.

We wrote the following python class, called `mimLocator`

```python
#!/usr/bin/env python2.7

import numpy as np
import sys

class Locator:
    def __init__(self, p1, p2, p3):

        # initialize points
        self.p1 = np.array(p1)
        self.p2 = np.array(p2)
        self.p3 = np.array(p3)

        # set vectors form p1 to p2 and from p1 to p3
        self.v12 = self.p2 - self.p1
        self.v13 = self.p3 - self.p1

        self.v12 = self.v12 / np.linalg.norm(self.v12)
        # calculate the normal to v12 and v13
        self.n = np.cross(self.v12, self.v13)
        self.n = self.n / np.linalg.norm(self.n)

        # calculate corresponding axes of the plane, having p1 as base
        # axisX = v12 / |v12|
        self.axisX = self.v12 / np.linalg.norm(self.v12)
        self.axisY = np.cross(self.n, self.axisX)
        #self.axisY = self.axisY / np.linalg.norm(self.axisY)

        self.rot = np.matrix([np.array(self.axisX), np.array(self.axisY),
                              np.array(self.n)])

        self.q = [ 0,0,0,0 ]
        self.q[3] = np.sqrt(1 + self.rot[0,0] + self.rot[1,1] \
                + self.rot[2,2]) / 2
        self.q[0] = (self.rot[2,1] - self.rot[1,2]) / (4 * self.q[3])
        self.q[1] = (self.rot[0,2] - self.rot[2,0]) / (4 * self.q[3])
        self.q[2] = (self.rot[1,0] - self.rot[0,1]) / (4 * self.q[3])

        self.scale = np.linalg.norm(self.p2 - self.p1)

    def planeToCartesian(self, x, y):
        """Takes a point (x,y) on the plane the Locator was initialized for and
        transforms it into (x,y,z) coordinates located in cartesian space.
        """
        p = self.p1 + (x * self.scale * self.axisX) \
                + (y * self.scale * self.axisY)
        return p

def main():
    l = Locator([float(sys.argv[1]), float(sys.argv[2]),
        float(sys.argv[3])], [float(sys.argv[4]), float(sys.argv[5]),
        float(sys.argv[6])], [float(sys.argv[7]), float(sys.argv[8]),
        float(sys.argv[9])])
    print l.planeToCartesian(sys.argv[10], sys.argv[11])

if __name__ == '__main__':
    main()
```

In the `__init__` function, we initialise the class variables and calculate the normal vector, the rotation matrix, the quaternions, etc.. The `planeToCartesian` function takes a point in 2d-space and transforms it into the corresponding point on the plane. The `main` function is to make scripting for later tasks easier. It lets the script take three points on the plane as argument, from which the corresponding plane is calculated, and then for another 2d-point, that is handed over as commandline argument as well, returns its output if given to `planeToCartesian`.

C.

We amended the given code in the indicated area as follows:

```
#!/usr/bin/env python

# This script should return the x y z and orientation coordinates of the end effector of the left limb.
# PLEASE ADD YOUR CODE WHERE INDICATED
# Avoid modifying the rest of the code if not necessary
#
# Authors: Stefano Pietrosanti - s.pietrosanti@pgr.reading.ac.uk
```

```python
#           Guy Butcher

import rospy
import baxter_interface
import numpy
from geometry_msgs.msg import (
    PoseStamped,
    Pose,
    Point,
    Quaternion,
)

print("MIM tutorial: forward kinematics.")
# Initialising ROS node
rospy.init_node("SSE_forward_kinematics")

######################  INSERT YOUR CODE HERE
# Create a "Limb" instance called "left_arm" linked to Baxter's left limb

left_arm = baxter_interface.Limb('left')

# Create a "pose" variable which holds the output of endpoint_pose()

pose = left_arm.endpoint_pose()

######################



# Return pose
print("Endpoint coordinates:")
print("X: " + str(pose['position'].x))
print("Y: " + str(pose['position'].y))
print("Z: " + str(pose['position'].z))
```

During The Lab

A.

B.

C.

D.

E.

F.

G.

H.

I.

J.

K.

L.