3_haptic_ball

November 6, 2024

1 Lab 3: a haptic ball

This final lab is very short and quite fun. We are going to generate haptic feedback on the robot.

We will model this a follows:

Select a position (x,y) on the board where a hypothetic deformable ball should be centered. Select a radius r for the ball, and write a code that does the following:

Whevener the end effector comes inside the circle, generate a repulsive force such that: - the direction of the force is given by the segment from (x,y) to the current position of the effector - The magnitude of the force is linearly decreasing from 0 at the border of the circle to 1 N at the center

You can achieve this simply using the contact jacobian term that you used in the software lab to input a force.

Once this is working you can try different kinds of interpolation to simulate different materials.

[ ]:
```
                                #        x,y
#System size                    #        /\
l1 = 0.06                       #       /  \
l2 = 0.165                      # l2-> /    \<-r2
r1 = 0.060                      # l1 ->\_dd_/<-r1
r2 = 0.163                      #   |   q1  q2
d  = 0.150 / 2                  # Y|
                                #   |____>
                                #     X
```

[ ]:
```
# set zero positions on each motor
from motor_control.AROMotorControl import AROMotorControl
mc = AROMotorControl()
mc.setZero(1)
mc.setZero(2)
```

For completing this lab, you will require two specific methods:

1. **Forward Geometry** from Lab 2.
2. **Contact Jacobian** from the software labs.

If you encounter difficulties in implementing these methods on your own, we have provided them in the `utils` file. You can easily import these methods using the following Python code:

from utils import *

[ ]:
```python
def fg(q1, q2):
    # TODO: implement this or use your implementation from lab 2
    pass

def J(q1, q2):
    # TODO: Implement this
    pass
```

Test your implementation

[ ]:
```python
from utils import *
import numpy as np
import time
dt = 0.001
N=300000 #30 seconds
t = time.perf_counter()
i=0
p0 = np.array([0,0.15]) # Adjust this accordingly
R = 0.02 # Adjust this accordingly
for i in range(N):
    # read positions and convert them into radians
    q1 = mc.readPosition(1) * np.pi / 180
    q2 = mc.readPosition(2) * np.pi / 180
    p = fg(q1, q2)
    d = np.linalg.norm(p-p0)
    f = np.array([0.,0.])
    if (d<R):
        f = (p-p0)/d
    tau = J(q1, q2).T @ f
    current_1 = tau[0] * 300
    current_2 = tau[1] * 300
    mc.applyCurrentToMotor(1, current_1)
    mc.applyCurrentToMotor(2, current_2)
    #wait for next control cycle
    t +=dt
    while(time.perf_counter()-t<dt):
        pass
        time.sleep(0.0001)
    if (i%100==0):
        #print(f"q={q}")
        #print(f"p=fk_delta(q)={p}")
        print(f"f={f}, tau={tau}")
```