2_FG_IG

October 15, 2024

1 Lab 2: forward and inverse geometry: writing with a pen

In this second lab, you are going to accomplish two different tasks: first, you are going to write a sentence by manually controlling the robot end effector, and if your implementation of forward geometry is correct, you will be able to reproduce what you wrote on your computer.

Then, you will use inverse kinematics to command the robot to draw a circle

warning : you will need the PD controller written in the previous lab to continue with this one.

1.1 Setting zero positions

First of all, we will need to properly initialise the robot. Follow the steps below to set the zero position for each motor. For more information, have a look at the description of setZero in ARO-MotorControlAPI.md file

- Step 1: Rotate the motors to the desired zero position, and put the aligner on the links to lock them in place.
- Step 2: call setZero(motor_id) on each motor.
- Step 3: Turn off the power supply, and wait for at least 5 seconds.
- Step 4: Turn the power supply back on, and wait for at least 5 seconds.
- Step 5: Remove the aligner

This zero position will be saved in motors' RAMs, and you will not have to recalibrate until you keep the power supply on.

[10]:
```
from motor_control.AROMotorControl import AROMotorControl
mc = AROMotorControl()
mc.setZero(1)
mc.setZero(2)
```

```
motor 1 has been reset. offset: 4294607111. restart required.
motor 2 has been reset. offset: 4294940986. restart required.
```

[10]: True

1.2 Forward Geometry

The robot is a delta arm with the following characteristics:

[11]:
```
import numpy as np
                        #         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
```

Two of the joints are controlled by the motors, while the other two joints are unactuated. However, solving the forward geometry analytically is doable with a bit of trigonometry. Finding the position of the passive joints is straightforward, and from these two positions you can find the position of the effector.

[35]:
```
import numpy as np
from math import sqrt, cos, sin

def fg(q1, q2,positive=True):
    # TODO: Implement this
    return np.zeros(2)
```

To test your forward geometry, write a 10-second long sequence that will read the encoders, apply forward geometry to store the positions of the effectors, and then plot them.

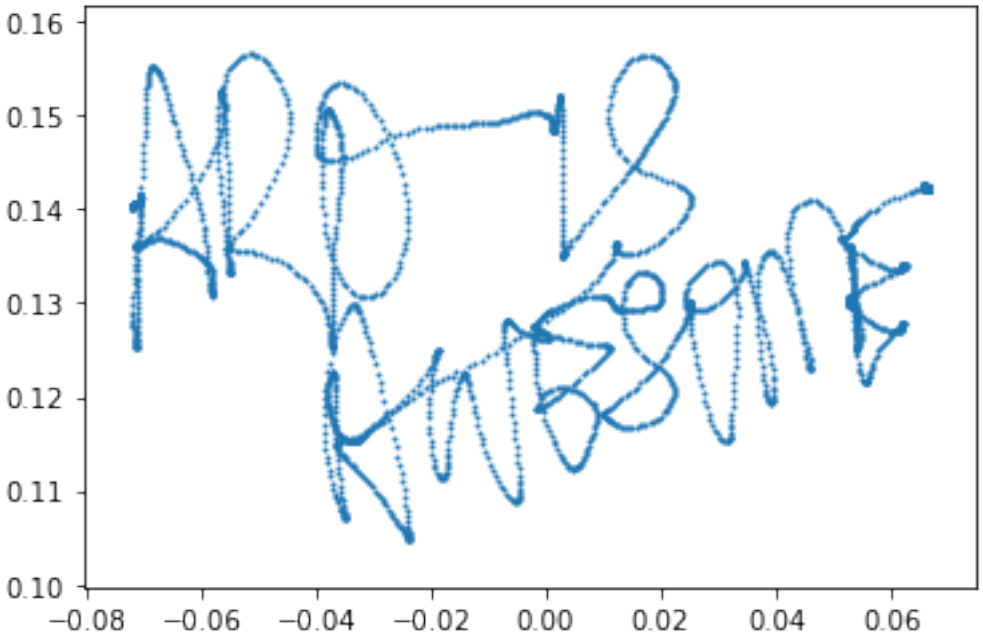
[22]:
```
import matplotlib.pyplot as plt
import time

def read_fg_loop(mc, T=10e3):
    xs, ys = [], []
    t = time.perf_counter()
    N=int(T)
    dt = 1. / 1e3
    wait = 1. / 1e4
    for i in range(N):
        t +=dt
        # TODO: Please note that angles q1 and q2 are in degrees, and
        # need to be converted to radians before being used with trigonometric
        # functions that expect radian arguments.
        q1,q2 = mc.readPosition(1), mc.readPosition(2)
        xy = fg(q1, q2)
        xs.append(xy[0])
        ys.append(xy[1])
        while(time.perf_counter()-t<dt):
            pass
            time.sleep(wait)
    plt.scatter(xs, ys, s=1)
```

```
    plt.show()
```

Test your implementation within a try-catch block, as you would normally do.

[33]:
```
from motor_control.AROMotorControl import AROMotorControl
try:
    mc = AROMotorControl()
    read_fg_loop(mc, T=3000)
except KeyboardInterrupt:
    print("KeyboardInterrupt received, stopping motors...")
except Exception as e:
    print(f"an error occurred: {e}")
finally:
    mc.applyCurrentToMotor(1, 0)
    mc.applyCurrentToMotor(2, 0)
    print("motors stopped!")
```




```
motors stopped!
```

1.3 Inverse geometry

Now write the IG function that will compute the inverse geometry of the robot. Use it to command the robot to draw a circle on the board

[6]:
```
def ig(xy):
    '''Computes forward geometry'''
    return np.zeros(2)
```