浙江大学实验报告

成员: 刘祥洲、张柯尧、张宇嘉、江小小
组号: 第六组
日期: 2022 年 12 月 31 日
地点: 石虎山

课程:	双足机器人	指导老师:	周春琳	分数:	
实验名称:	太空机械臂设计	实验类型:	实物实验	组号:	第六组

1 实验目标

1. 根据网络公开资料，设计一款中国空间站的机械臂 3D 模型，机械臂的关节和连杆可以简化为圆柱体，自定义机械臂的尺寸，要求臂展不超过 80cm。
2. 设计开发机械臂的运动学控制程序，在仿真环境中展示太空机械臂的行走运动，如视频所示，至少展示行走 2 步，即从初始位置移动到位置 1，再移动到位置 2，移动姿态和速度在合理范围内自定义。
3. 根据机械臂的尺寸，自定义基座位置，但两个基座需要满足轴线正交关系，如图所示（假设机械臂末端于基座之间接触后可以通过电磁铁紧密吸合）。

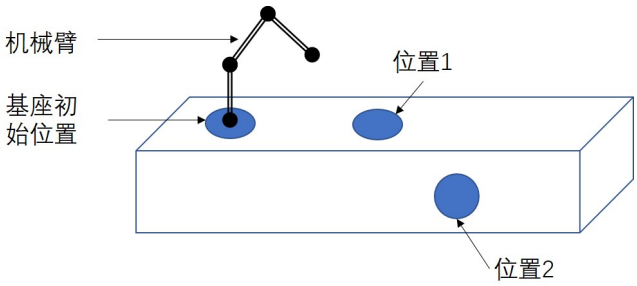
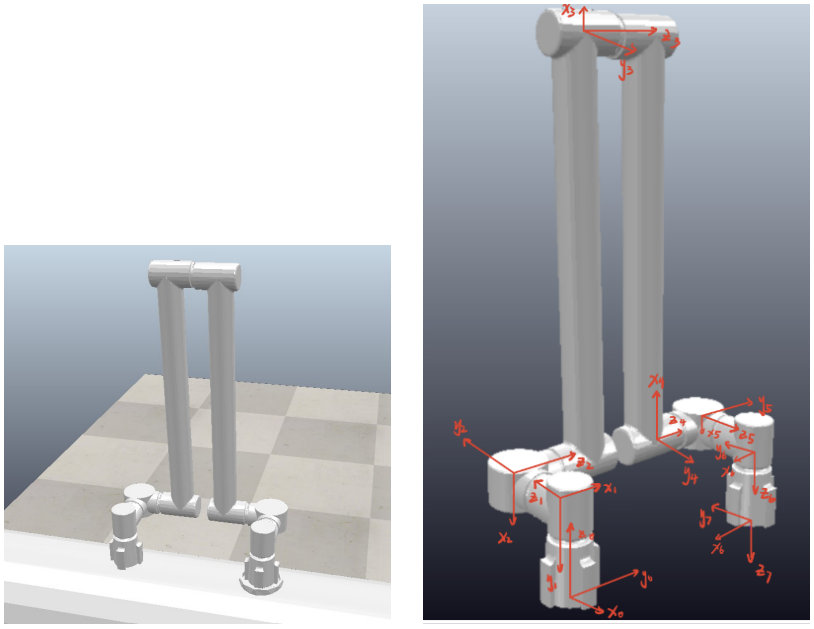


图 1: 题目示意图

2 实验内容

2.1 模型搭建与 DH 参数



(a) 模型 (b) 坐标系

图 2: 太空机械臂模型

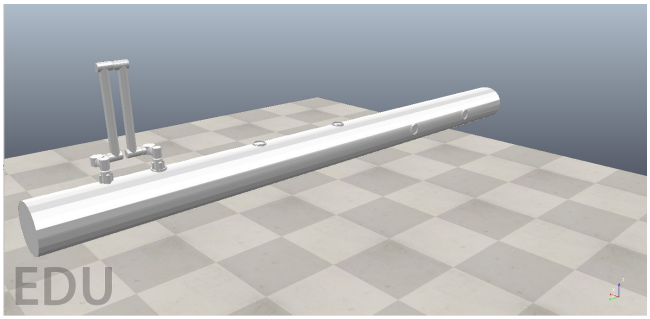


图 3: 模型与场景展示

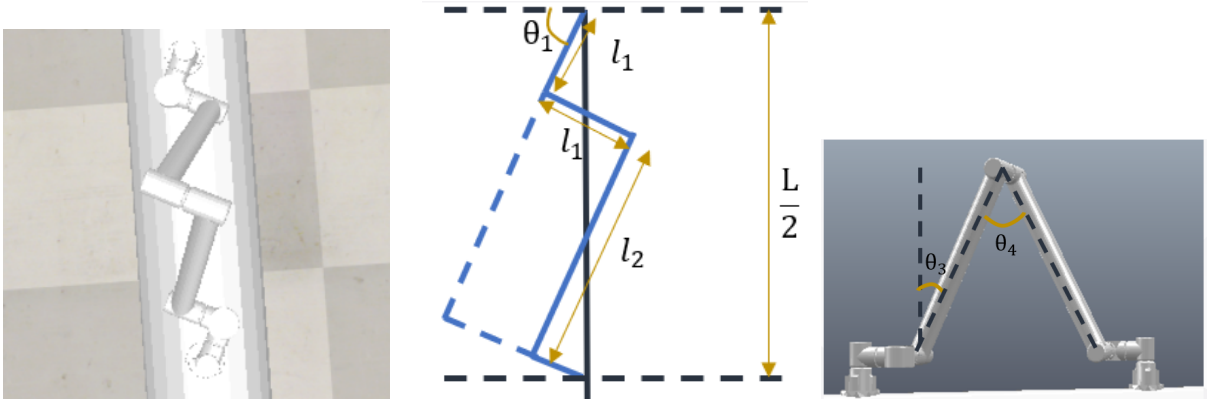
表 1: DH 参数表

	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	$-\frac{\pi}{2}$	100	$\theta_1+\pi$
2	0	$\frac{\pi}{2}$	100	$\theta_2+\frac{\pi}{2}$
3	500	0	100	$-\theta_3-\pi$
4	500	0	100	$-\theta_4$
5	0	$\frac{\pi}{2}$	100	$-\theta_5+\pi$
6	0	$-\frac{\pi}{2}$	100	$-\theta_6-\frac{\pi}{2}$
7	0	0	100	θ_7

DH 参数通过距离角度等变量，构建出父子坐标系，揭示各个关节相互之间运动关系，方便我们未来处理各个关节之间相互运动关系。

2.2 平面移动的几何关系

通过计算我们发现如下平面移动几何关系：



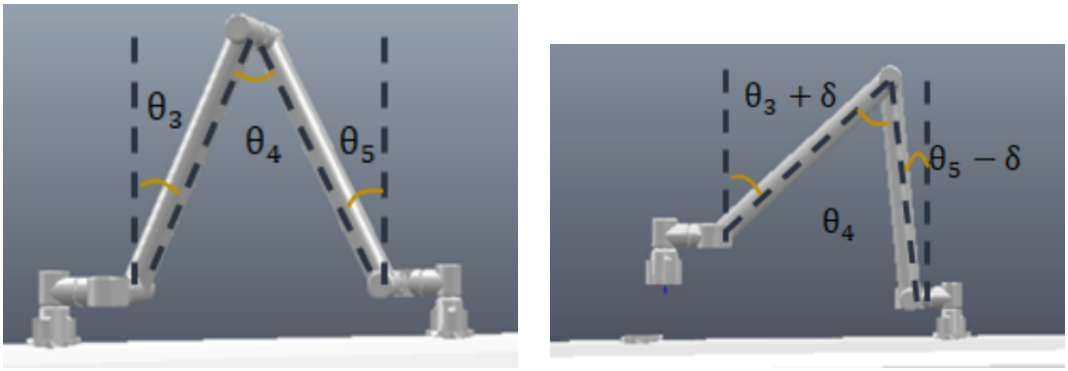
(a) 俯视 (b) 俯视（几何抽象） (c) 侧视

图 4: 平面运动几何关系示意图

假设我们俯视观察机械臂，并且只观察前半段，设水平方向上跨 L，并且两个大臂之间夹角为 θ_4，大臂与竖直垂线之间夹角为 θ_3，θ_2 是第二个关节的转角，θ_1 如上图所示：

- 显然有 $\theta_2=0, \theta_4=2\theta_3$
- 考虑到我们的模型参数有 $l_1=100, l_2=500sin\theta_3$
- 根据几何关系有 $\frac{L}{2}cos\theta_1=\frac{3}{2}l_1=150$ 即 $\theta_1=arccos(\frac{300}{L})$
- 又有 $\frac{L}{2}sin\theta_1=l_1+l_2=100+500sin\theta_3$ 即 $\theta 3=arcsin(\frac{\frac{L}{2}sin\theta_1-100}{500})$

综上，当前进的步长 L 已知时，我们可以由几何关系得到 θ_1,θ_2,θ_3,θ_4；且有 θ_5,θ_6,θ_7 与 θ_3,θ_2,θ_1 对称。
有了以上的几何关系，我们可以考虑机械臂在平面上的移动过程：抬腿，旋转 θ_1，落腿（抬腿的逆过程）只用详细说明抬腿即可：由下图可知，只要在之前计算出的姿态基础上，改变 θ_3，θ_4



(a) 抬腿前 (b) 抬腿后

图 5: 抬腿过程示意图

2.3 逆运动学

由 python 编程计算。由于机械臂存在一个自由度的冗余，将 θ_7 固定为 0。首先我们由如下公式得到 ${}^{i-1}T_i$

$$
{}^{i-1}T_i = \begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\alpha)\sin(\theta) & a\cos(\theta) \\ \sin(\theta) & \cos(\alpha)\cos(\theta) & -\sin(\alpha)\cos(\theta) & a\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

编写逆运动学求解函数。根据两种 ${}^{3}T_5$ 对应相等:

$$
{}^{3}T_5 = {}^{3}T_4 {}^{4}T_5
$$

$$
{}^{3}T_5 = {}^{0}T_3{}^{-1} {}^{0}T_7 {}^{5}T_7{}^{-1}
$$

根据（3，3）（3，4 ）对应相等可以求得

$$
o_x c_1 c_2 + o_y s_1 c_2 + o_z s_2 = 0 \tag{1}
$$

$$
\begin{aligned}
&100a_x c_1 c_2 + 100a_y s_1 c_2 + 100a_z s_2 \\
&- 100o_x c_1 c_2 - 100o_y s_1 c_2 - 100o_z s_2 \\
&- p_x c_1 c_2 - p_y s_1 c_2 - p_z s_2 + 100s_2 - 100 = 200
\end{aligned} \tag{2}
$$

计算求得 θ_1 数值解，$\theta_2 = atan(-\frac{o_x c_1 + o_y s_1}{o_z})$

编写逆运动学求解函数。根据两种 ${}^{1}T_3$ 对应相等:

$$
{}^{1}T_3 = {}^{1}T_2 {}^{2}T_3
$$

$$
{}^{1}T_3 = {}^{0}T_1{}^{-1} {}^{0}T_7 {}^{3}T_7{}^{-1}
$$

根据（3，3）对应相等可以求得

$$
-(a_x s_1 - a_y c_1)s_6 - (n_x s_1 - n_y c_1)c_6 = 0 \tag{3}
$$

求解得到 $\theta_6 = atan(-\frac{n_x s_1 - n_y c_1}{a_x s_1 - a_y c_1})$

编写逆运动学求解函数。根据两种 ${}^{1}T_5$ 对应相等:

$$
{}^{1}T_5 = {}^{1}T_2 {}^{2}T_3 {}^{3}T_4 {}^{4}T_5
$$

$$
{}^{1}T_5 = {}^{0}T_1{}^{-1} {}^{0}T_7 {}^{5}T_7{}^{-1}
$$

根据（1，4）（3，4）对应相等可以求得

$$\begin{aligned}&500s_2c_3 - 500s_2c_{34} + 300c_2\\&= 100a_xc_1 + 100a_ys_1 - 100o_xc_1 - 100o_ys_1 - p_xc_1 - p_ys_1\end{aligned} \tag{4}$$

进行求解

$$M = -100a_xs_1 + 100a_yc_1 + 100o_xs_1 - 100o_yc_1 + p_xs_1 - p_yc_1 - 100$$

$$N = \frac{100a_xc_1 + 100a_ys_1 - 100o_xc_1 - 100o_ys_1 - p_xc_1 - p_ys_1 - 300c_2}{s_2}$$

$$A = 500$$

$$B = 500$$

计算得到 $\theta_4 = \pm acos(\frac{M^2+N^2-A^2-B^2}{-2AB})$

$$a = -Bs_4$$

$$b = A - Bc_4$$

$$c = M$$

计算得到 $\theta_3 = atan2(c, \pm\sqrt{a^2+b^2-c^2}) - atan2(a, b)$

编写逆运动学求解函数。根据两种 ${}^4T_6$ 对应相等：

$${}^4T_6 = {}^4T_5{}^5T_6$$

$${}^4T_6 = {}^0T_4{}^{-1}{}^0T_7{}^6T_7{}^{-1}$$

根据（2，4）对应相等可以求得

$$\begin{aligned}&100c_5 = 100a_xs_1c_{34} + 100a_xs_2s_{34}c_1\\&+ 100a_ys_1s_2s_{34} - 100a_yc_1c_{34} - 100a_zs_{34}c_2\\&- p_xs_1c_{34} - p_xs_2s_{34}c_1 - p_ys_1s_2s_{34} + p_yc_1c_{34}\\&+ p_zs_{34}c_2 - 500s_4 - 100s_{34}c_2 + 100c_{34}\end{aligned} \tag{5}$$

计算得到 $\theta_5 = \pm acos((100a_xs_1c_{34}+100a_xs_2s_{34}c_1+100a_ys_1s_2s_{34}-100a_yc_1c_{34}-100a_zs_{34}c_2-p_xs_1c_{34}-p_xs_2s_{34}c_1-p_ys_1s_2s_{34}+p_yc_1c_{34}+p_zs_{34}c_2-500s_4-100s_{34}c_2+100c_{34})/100)$

代码已经在之前的实验中实现过，详见 inv.py，这里只需调用如下接口：

```
q = myiks_solver(x, y, z, rx, ry, rz)
```

2.4 轨迹规划

尝试了两点间三次多项式轨迹规划，但效果不好，最终采用五次多项式，**五次多项式的好处在于能够控制两点处的速度和加速度，可以在保证各段速度连续的情况下同时保证速度光滑**。

假设机械臂的末端点的运动轨迹方程为：

$$q(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 + a_6t^6 + a_7t^7 \tag{6}$$

我们已知的条件为如下等式

$$q(0) = startPosition[i] \tag{7}$$

$$q(t) = endPosition[i] \tag{8}$$

$$\dot{q}(0) = startVel[i] \tag{9}$$

$$\dot{q}(t) = endVel[i] \tag{10}$$

$$\ddot{q}(0) = startAcc[i] \tag{11}$$

$$\ddot{q}(t) = endAcc[i] \tag{12}$$

最终我们可以得到如下矩阵解法

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ t^5 & t^4 & t^3 & t^2 & t & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5t^4 & 4t^3 & 3t^2 & 2t & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20t^3 & 12t^2 & 6t & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} startPosition[i] \\ endPosition[i] \\ startVel[i] \\ endVel[i] \\ startAcc[i] \\ endAcc[i] \end{bmatrix} \tag{13}$$

2.5 代码实现细节

2.5.1 支撑腿的改变

在实现中，我们使用如下代码获得各个 joint 和 link 的 handler，并将其分别存放在两个 list 中。

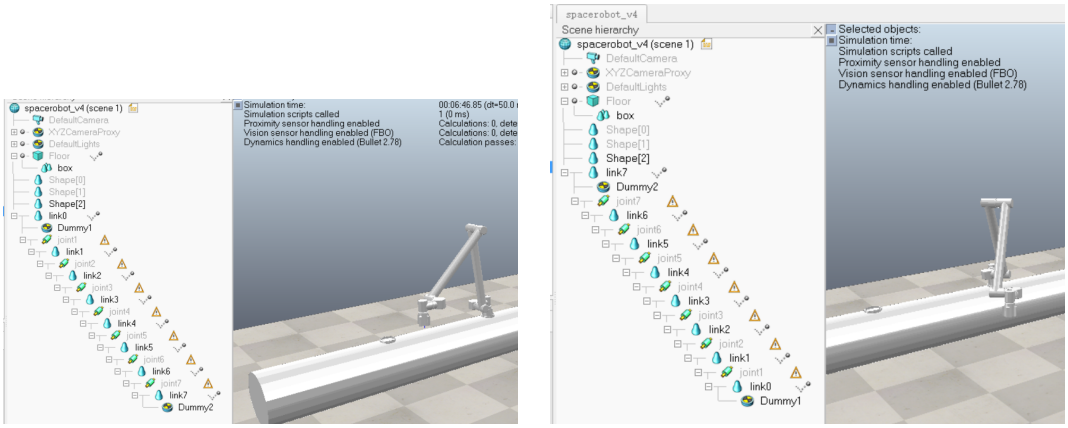
```
for i in range(7):
        jointName = "joint" + str(i+1)
        ret, joint = sim.simxGetObjectHandle(clientID, jointName, sim.simx_opmode_blocking)
        if ret != sim.simx_return_ok:
            print("Can't find joint %d" % (i+1))
            exit()
        jointHandle.append(joint)

    for i in range(8):
        linkName = "link" + str(i)
        ret, link = sim.simxGetObjectHandle(clientID, linkName, sim.simx_opmode_blocking)
        if ret != sim.simx_return_ok:
            print("Can't find link %d" % (i))
            exit()
        linkHandle.append(link)
```

每当要改变机械臂的支撑腿的时候，就使用如下函数：

```
def ChangeCoordinate(jointHandle, LinkHandle):
    global real_Parent
    jointHandle.reverse()
    LinkHandle.reverse()
    sim.simxSetObjectParent(clientID, LinkHandle[0], -1, True, sim.simx_opmode_streaming)
    for i in range(7):
        sim.simxSetObjectParent(clientID, jointHandle[i], LinkHandle[i], True, sim.
                                                    simx_opmode_streaming)
        sim.simxSetObjectParent(clientID, LinkHandle[i+1], jointHandle[i], True, sim.
                                                    simx_opmode_streaming)
    real_Parent = not real_Parent
    state.revrese()
```

该函数是对 sim.simxSetObjectParent 函数的一个包装，其意义是将机械臂各 joint 和 link 的父子结构倒转；除此之外，该函数还将 jointHandle 和 linkHandle 两个 list 翻转，使得个 joint 和 link 依旧按照距离支撑点的距离从小到大放置在 list 中。

除此之外，我们还在代码中用一个 state 列表记录各个关节角在仿真中的转角，我们同样在这里将其翻转。



(a) 改变前 (b) 改变后

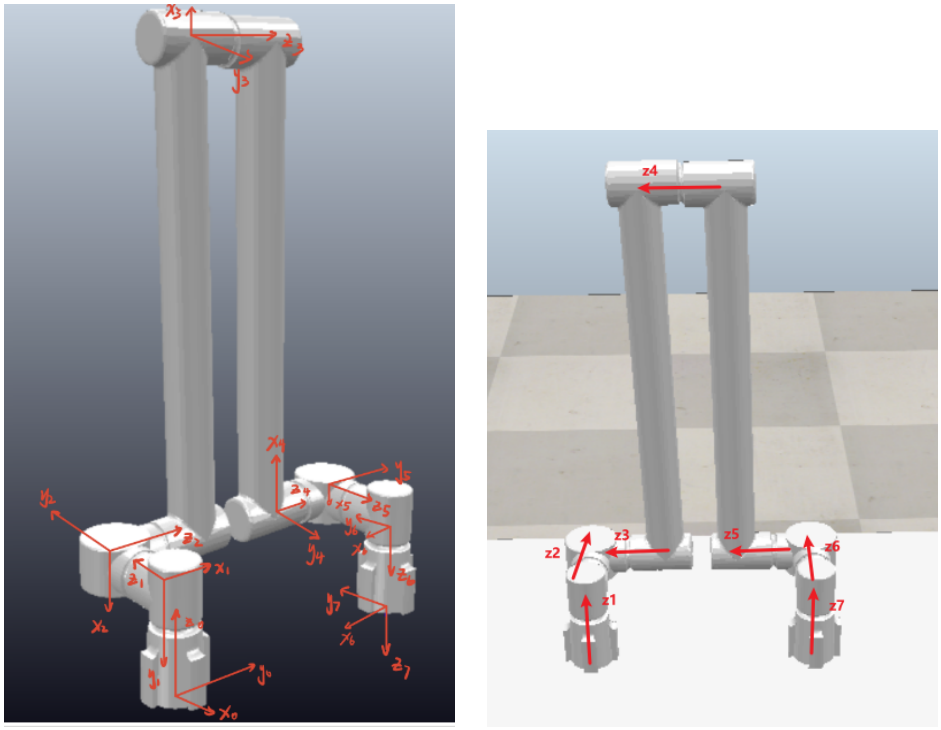
图 6: 支撑腿改变前后父子关系展示图

2.5.2 从逆运动学解到仿真中的关节角

逆运动学中解出的关节角的角度制是相对我们在列 DH 参数表的时候建立的坐标系的，而仿真真中的关节角则是该 joint 相对原始状态旋转的角度。由于建系的方向与仿真中转轴的方向不一致，支撑腿的改变等因素，这两者并不是保持一致的，因此，我们需要一个将逆运动学解出的解转化为当前仿真中真实的角度的函数：

```
    def IKSolution2RealTheta(thetas):
    global real_Parent
    if real_Parent == False: #好处理  [0, 0, -np.pi/2, 0, -np.pi, 0 , 0]
        real_theta1, real_theta2, real_theta3, real_theta4, real_theta5 = thetas[0:5]
        real_theta6 = -thetas[5]
        real_theta7 = -thetas[6]
    else: # [0, np.pi, -np.pi/2, 0, np.pi/2, np.pi, 0]
        real_theta1 = thetas[0]
        real_theta2 = np.pi + thetas[1]
        real_theta3 = -np.pi/2 - thetas[2]
        real_theta4 = -thetas[3]
        real_theta5 = np.pi/2 - thetas[4]
        real_theta6 = np.pi - thetas[5]
        real_theta7 = - thetas[6]
    return real_theta1, real_theta2, real_theta3, real_theta4, real_theta5, real_theta6, real_theta7
```

思路是，通过 real_Parent 参数判断仿真中的父子关系是否是初始时的父子关系（与 DH 坐标系的 0 到 7 一致）；若是，则仅用调整仿真转轴与 DH 坐标系中 z 轴方向不一样的角度的正负号；若不是，则先加上一组在该父子关系下能使仿真初态转到 DH 初态的角度值，再调整 DH 坐标系中 z 轴方向不一样的角度的正负号



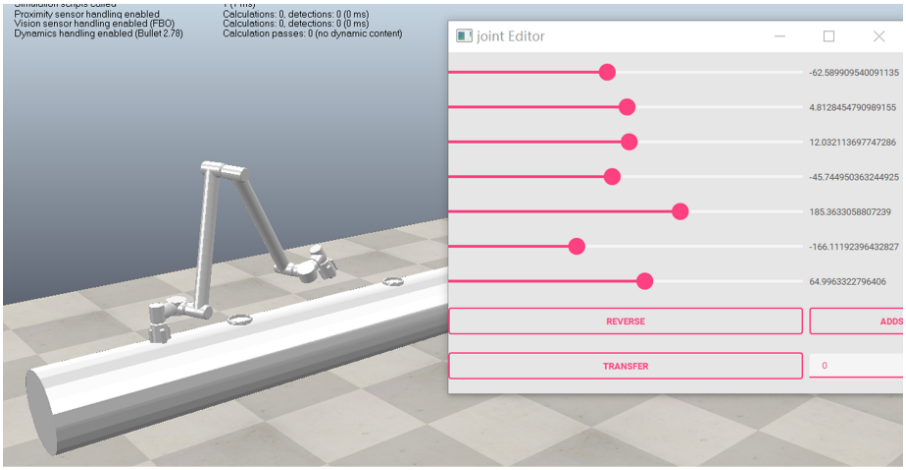
(a) DH 参数表使用坐标系 (b) 仿真中转轴方向

图 7: DH 坐标系与仿真中转轴方向的对比

2.6 效果展示

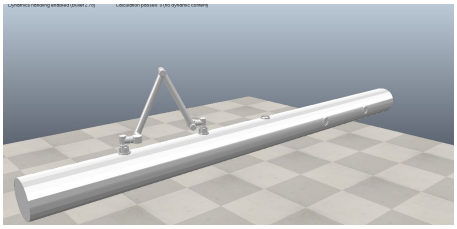
2.6.1 辅助 UI

我们使用 QT 搭建了一个可以实时控制仿真环境中机械臂行为的 UI 界面，可以直接拖拽控制条控制各个关节角的角度；
我们使用这一工具验证逆运动学解的结果，并微调以适应装配时产生的误差。

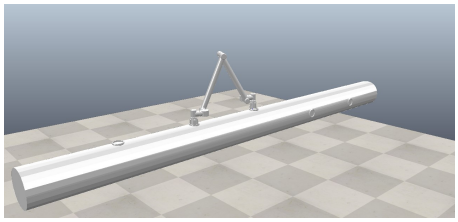


2.6.2

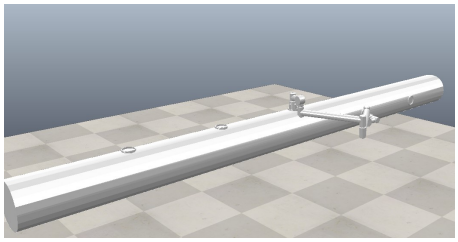
更详细的效果见附录视频



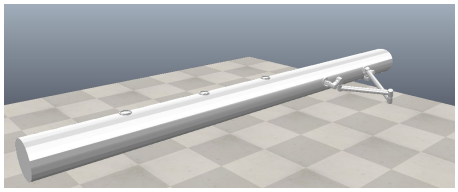
(a) 步骤 1



(b) 步骤 2



(c) 步骤 3



(d) 步骤 4

图 8: 运行效果展示图