机器人技术与实践

实验 3：ZJU-I 型机械臂 URDF 生成

姓名: 徐屹寒

时间: 2025/10

目录

实验目的

本实验旨在通过 SolidWorks 和 sw2urdf 插件完成 ZJU-I 型机械臂的 URDF 模型生成,并在 CoppeliaSim 中验证模型的正确性。主要目标包括:

1. 掌握 SolidWorks 装配体建模方法,完成机械臂各部件的装配
2. 理解机械臂零位姿态的定义,通过配合关系调整零位
3. 学习在 SolidWorks 中建立坐标系和参考轴的方法
4. 使用 sw2urdf 插件将装配体导出为 URDF 格式文件
5. 在 CoppeliaSim 中导入并验证 URDF 模型的运动性能
6. 编写测试脚本验证机械臂各关节的运动功能

实验内容

2.1. SolidWorks 装配体建立

2.1.1. 导入零件文件

首先将 ZJU-I 型机械臂的各个零件导入 SolidWorks 装配体环境。

Important

零件导入步骤:

1. 新建装配体

文件 → 新建 → 装配体，创建空白装配体文档

2. 插入零件

插入 → 零部件 → 现有零件/装配体，依次导入机械臂各部件

2.1.2. 建立配合关系

通过配合关系将各零件按照机械臂的运动链进行装配。

2.2. 零位姿态调整

机械臂的零位姿态定义了各关节角度为 0 时的配置,这是 URDF 模型的重要参考。本次实验通过配合关系在 solidworks 软件中确定零位姿态，后续再取消多余的配合关系。

Tip

注意事项:

零位姿态的定义会直接影响后续 URDF 模型在仿真软件中的初始姿态。

2.3. 坐标系与参考轴建立

使用 SolidWorks 的“参考几何体”功能为各零件建立坐标系和参考轴。

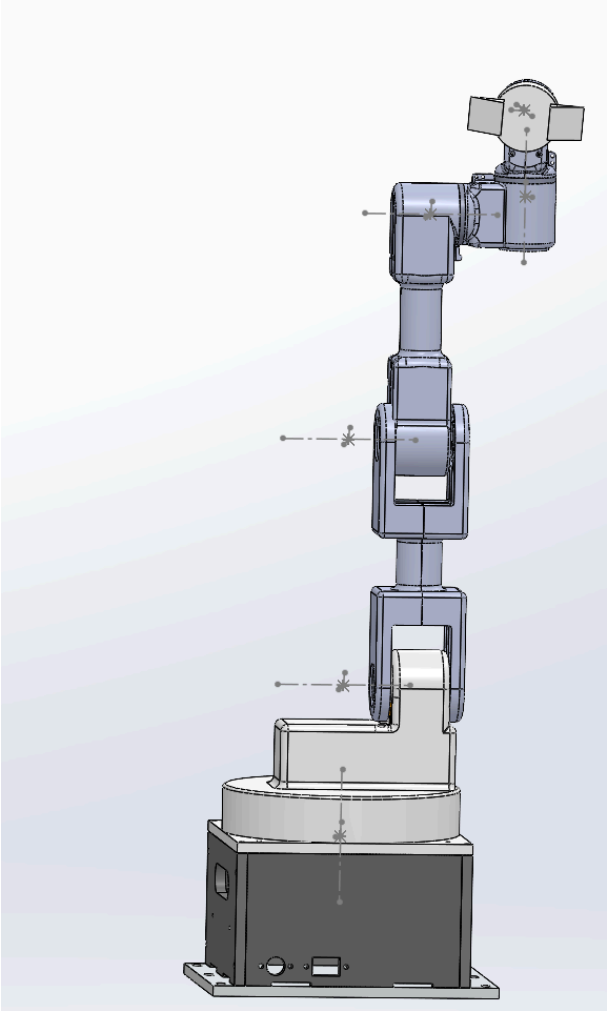

Figure 1: 装配体

2.4. 使用 sw2urdf 导出 URDF

安装并使用 sw2urdf 插件将装配体导出为 URDF 格式文件。

2.4.1. URDF 导出配置

Important

sw2urdf 导出步骤:

1. 启动导出工具

工具 → Tools → Export as URDF

2. 配置基座

选择基座零件作为“base_link”

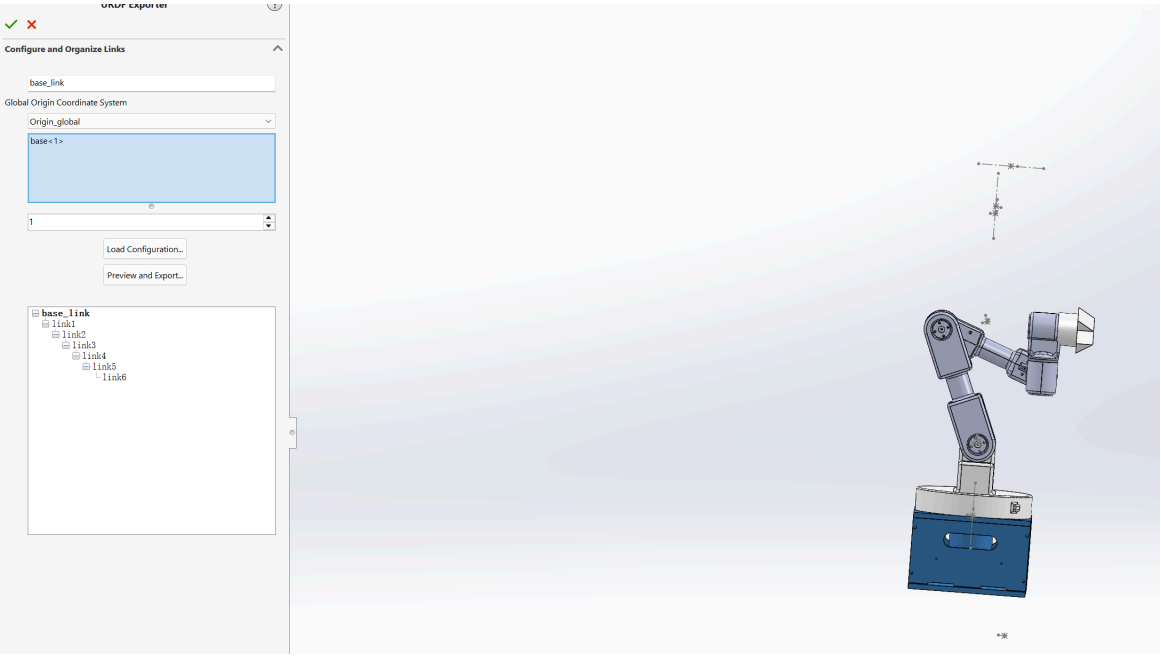

Figure 2: 配置基座

3. 定义关节和连杆

填入当前节点直接子节点的个数（除了末端执行器以外均为 1），接着配置下一个节点的各个属性（参考系、参考轴、关节类型、直接子节点个数）

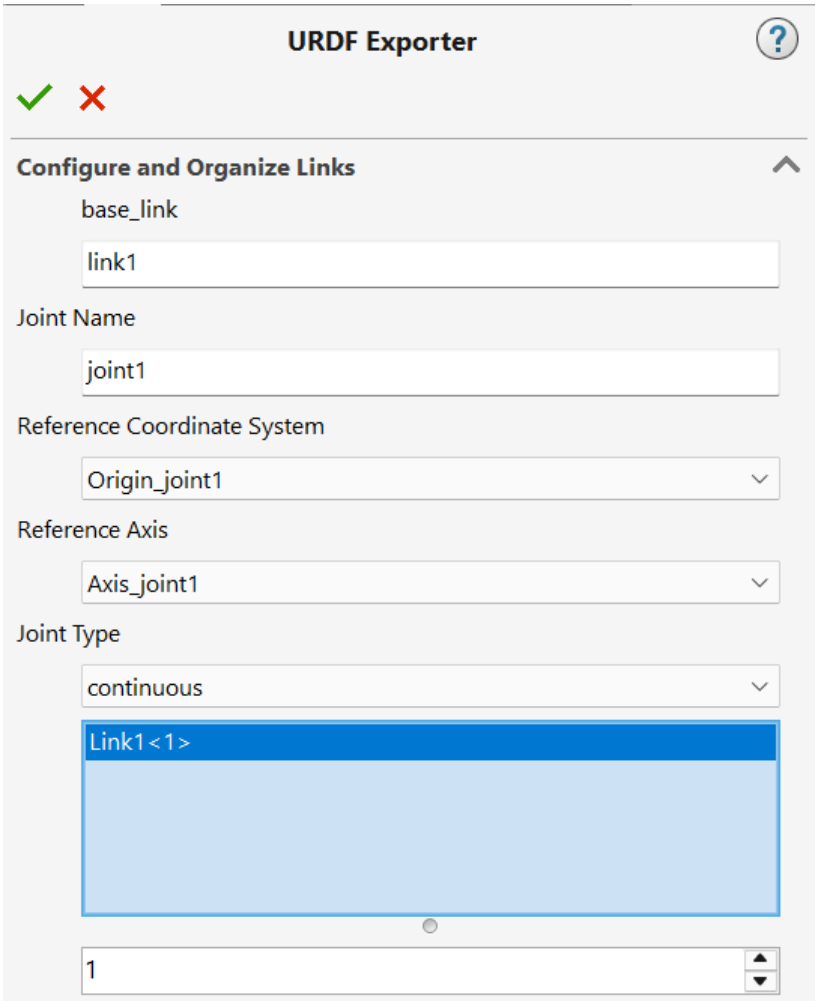

Figure 3: 配置节点属性

6. 预览和导出

点击 **Preview and Export** 按钮导出 URDF 文件和 mesh 文件

Note

URDF 文件检查:

导出后需要使用文本编辑器打开 URDF 文件,检查 mesh 文件路径是否正确

2.5. CoppeliaSim 导入 URDF

将生成的 URDF 文件导入 CoppeliaSim 进行验证。

2.5.1. URDF 导入步骤

Important

CoppeliaSim 导入 URDF:

1. 导入 **URDF** 文件

Modules → Importers → URDF importer

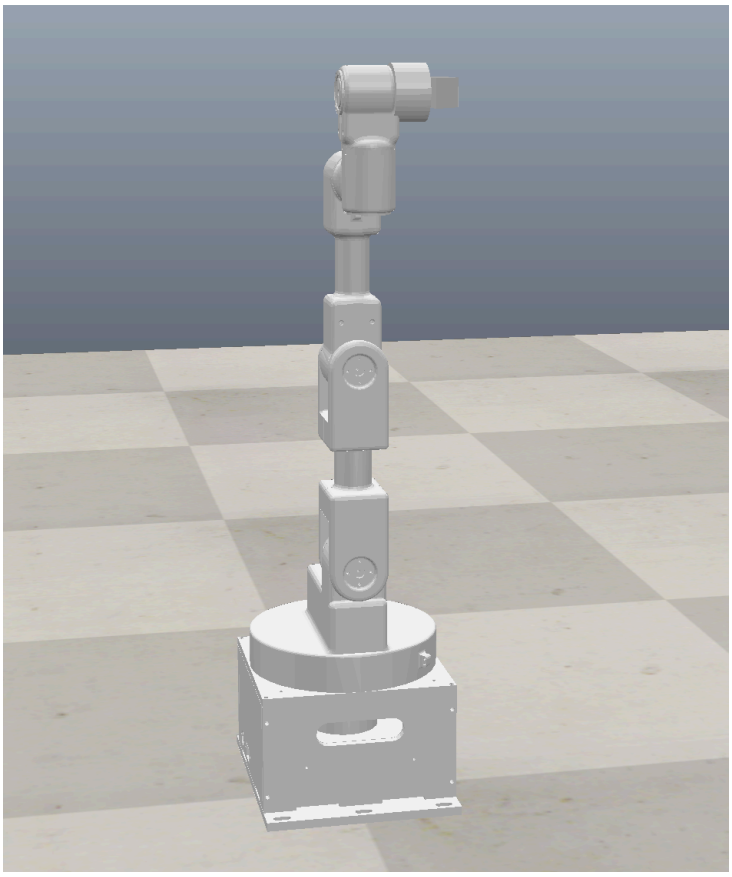

Figure 4: URDF 文件在 CoppeliaSim 中的显示

2.6. 编写测试脚本

编写 Python 脚本测试机械臂各关节是否可以正常运动。上交文件中附有视频。

附录: 测试脚本代码

以下为完整的机械臂运动测试脚本:

```
import math

def sysCall_init():
    """
    初始化函数 - 获取所有关节句柄
    """
    sim = require('sim')

    # 获取6个关节的对象句柄
    self.joint1 = sim.getObject('/joint1')
    self.joint2 = sim.getObject('/joint2')
    self.joint3 = sim.getObject('/joint3')
    self.joint4 = sim.getObject('/joint4')
```

```
    self.joint5 = sim.getObject('/joint5')
    self.joint6 = sim.getObject('/joint6')

def sysCall_actuation():
    """
    主控制循环 - 分段测试各关节运动
    """
    time = sim.getSimulationTime()

    # 关节1测试 (0-5秒)
    if time < 5:
        angle = math.pi * time / 5  # 0到π
        sim.setJointTargetPosition(self.joint1, angle)

    # 关节2测试 (5-10秒)
    elif time < 10:
        angle = math.pi/2 * (time - 5) / 5  # 0到π/2
        sim.setJointTargetPosition(self.joint2, angle)

    # 关节3测试 (10-15秒)
    elif time < 15:
        angle = -math.pi/2 + math.pi * (time - 10) / 5  # -π/2到π/2
        sim.setJointTargetPosition(self.joint3, angle)

    # 关节4测试 (15-20秒)
    elif time < 20:
        angle = math.pi * (time - 15) / 5  # 0到π
        sim.setJointTargetPosition(self.joint4, angle)

    # 关节5测试 (20-25秒)
    elif time < 25:
        angle = math.pi/3 * math.sin(2 * math.pi * (time - 20) / 5)  # 正弦运动
        sim.setJointTargetPosition(self.joint5, angle)

    # 关节6测试 (25-30秒)
    elif time < 30:
        angle = 2 * math.pi * (time - 25) / 5  # 0到2π,完整旋转
        sim.setJointTargetPosition(self.joint6, angle)

    # 复位 (30-35秒)
    elif time < 35:
        # 所有关节缓慢回到零位
        factor = 1 - (time - 30) / 5
        sim.setJointTargetPosition(self.joint1, math.pi * factor)
        sim.setJointTargetPosition(self.joint2, math.pi/2 * factor)
        sim.setJointTargetPosition(self.joint3, 0)
        sim.setJointTargetPosition(self.joint4, math.pi * factor)
        sim.setJointTargetPosition(self.joint5, 0)
        sim.setJointTargetPosition(self.joint6, 0)

def sysCall_sensing():
```

```
    pass

def sysCall_cleanup():
    pass
```