班　级：　　机工 2001

学　号：　　2020030190

北京化工大学

毕业设计(论文)

题　目　协作六自由度机械臂 ROS 仿真研究

专　　业　机器人工程

学　　生　李兆镇

指导教师　魏彬（副教授）

2024 年 **05** 月 **14** 日

诚信申明

本人申明：

本人所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

本人签名：

2024年05月31日

毕业设计（论文）任务书

设计(论文)题目： 协作六自由度机械臂 ROS 仿真研究

学院： 机电工程学院 **专业：** 机器人工程 **班级：** 机工 2001

学生： 李兆镇 **指导教师：** 魏彬 **专业负责人：** 陈国华

1.设计（论文）的主要任务及目标

(1)完成六自由度机械臂理论分析；

(2)完成机械臂的三维模型建立；

(3)在 ROS 系统中完成对机械臂的仿真控制；

(4)在仿真界面完成常见规划算法的比较分析。

2.设计（论文）的主要内容

(1)在 Solidworks 中完成机械臂建模及到 ROS 系统 URDF 文件的转换；

(2)在 ROS 系统的 Rviz 三维可视化界面完成对虚拟机械臂的仿真控制；

(3)分别使用 RRT 算法和 RRTStar 算法对虚拟机械臂进行轨迹规划。

3.设计（论文）的主要要求

(1)三维模型必须具有六个自由度；

(2)虚拟仿真控制机械臂尽可能真实可靠；

(3)轨迹规划可完成主动避障功能。

4. 主要参考文献

[1] 田佳唯. 六自由度机械臂路径规划算法改进及强化学习融合方法[D].北京化工大学,2024.DOI:10.26939/d.cnki.gbhgu.2023.001359.

[2] 程聪.六自由度机械臂半实物仿真系统研究[D].北京理工大学,2016.

[3] 金翰扬. 基于 ROS 六自由度机械臂控制系统及路径规划[D].南京航空航天大学,2022.DOI:10.27239/d.cnki.gnhhu.2022.000816.

[4] 彭君. 改进 RRT 算法在移动机器人路径规划中的应用研究[D].南京邮电大学,2023.DOI:10.27251/d.cnki.gnjdc.2022.001282.

5.进度安排

	设计（论文）各阶段名称	起止日期
1	文献查阅、综述撰写、开题报告及答辩	2023.11.10——2023.12.31
2	完成六自由度机械臂理论分析	2024.01.01——2024.01.31
3	完成模型建立和转换	2024.02.01——2024.03.15
4	完成 MoveIt!机械臂控制和控制算法分析	2024.03.16——2024.04.20
5	完成论文撰写、准备答辩	2024.04.21——2024.06.01

协作六自由度机械臂 ROS 仿真研究

摘　要

在“中国智造新纪元”的宏伟框架下，我国制造业迎来了以智能机器人技术为主导的研发热潮。六自由度机械臂凭借其卓越的稳定性和广泛的适用性，在制造业的广阔天地中得到了深入且广泛的应用。本研究以高通量催化剂化工操作台所使用的六自由度机械臂为研究对象，分别对其进行理论分析，建模仿真和路径规划算法进行研究。

本文的主要研究内容如下：首先完成六自由度机械臂的理论分析，包括对该机械臂 D-H 参数表的建立，机械臂正逆运动学分析和该机械臂的拉格朗日动力学模型建立。其次在 Solidworks 中完成了六自由度机械臂的建模和装配，在 Rviz 中完成该模型的三维可视化。然后在 ROS 系统中完成了 MoveIt!核心架构的配置工作，对其中的运动学规划器、运动学求解器、可视化配置过程作了详细阐述并展示了通讯节点的计算图。在配置过程中对机械臂做了五种位姿的运动规划，并对四次规划的结果进行了说明。应用 RRT 算法和 RRTStar 算法完成对机械臂的轨迹规划。通过对其规划的轨迹做关节空间角度和速度随时间的变化分析，分析说明使用 RRTStar 算法完成本机械臂的轨迹规划和避障更加高效合理。

关键词：机械臂，MoveIt!，RRT

A ROVOTIC ARM WITH SIX DEGREES OF FREEDOM WAS SUBJECTED TO A ROS SIMULATION STUDY

ABSTRACT

In the context of the "New Era of China's Intelligent Manufacturing," the Chinese manufacturing industry has witnessed a surge in research and development activities, spearheaded by intelligent robotics. The 6-DOF robotic arm, renowned for its exceptional stability and broad applicability, has emerged as a pivotal technology in the vast realm of manufacturing. In this study, the 6-DOF robotic arm used in the high-throughput catalyst chemical operating table is taken as the research object, and its theoretical analysis, modeling simulation, and path planning algorithm are investigated, respectively.

The primary focus of this paper is to present a comprehensive theoretical analysis of the 6-DOF robotic arm. This analysis encompasses the construction of the D-H parameter table for the robotic arm, a positive and negative kinematic analysis of the robotic arm, and the establishment of the Lagrangian dynamics model for the robotic arm. Secondly, the 6-DOF robotic arm was modelled and assembled in Solidworks, and the 3D visualisation of this model was completed in Rviz. Then the configuration of the MoveIt! core architecture was completed in ROS, in which the kinematic planner, kinematic solver, and visualization configuration process were elaborated in detail, and the computational diagrams of the communication nodes were shown. The kinematic planning process was conducted for the robotic arm in five positions during the configuration process. The results of four planning sessions are presented in the following sections. The RRT algorithm and RRTStar algorithm are employed to complete the trajectory planning of the robotic arm. The analysis of joint space angle and speed change with time on the planned

trajectory indicates that the RRTStar algorithm is a more efficient and reasonable approach for trajectory planning and obstacle avoidance of this robotic arm.

KEY WORDS: Robotic arm, MoveIt!, RRT

符号说明

符号	符号名称	单位	说明
a_{i-1}	连杆长度	mm	z_{i-1} 沿着 x_{i-1} 到 z_i 的距离
α_{i-1}	连杆扭角	°	z_{i-1} 绕 x_{i-1} 到 z_i 的转角
d_i	关节偏置	mm	x_{i-1} 沿着 z_i 到 x_i 的距离
θ_i	关节转角	°	x_{i-1} 绕 z_i 到 x_i 的转角
$\boldsymbol{\theta_{iM}}$	最大转角	°	关节 i 的最大转角
$\boldsymbol{\theta_{im}}$	最小转角	°	关节 i 的最小转角
L	拉格朗日函数	J	动能与势能之差
T	动能函数	J	机械臂关节的动能
V	势能函数	J	机械臂关节的势能
$\boldsymbol{M(\theta)}$	质量项系数矩阵	/	机械臂质量项系数矩阵
$\boldsymbol{C(\theta,\dot{\theta})}$	离心力和科氏力项系数矩阵	/	机械臂离心力和科氏力项系数矩阵
$\boldsymbol{G(\theta)}$	重力项系数	/	机械臂重力项系数矩阵
$\boldsymbol{\tau}$	关节输出力(力矩)	N/mm	机械臂关节的输出力(力矩)

缩略词说明

缩略词	英文全称	中文释义
ROS	Robotic Operation System	机器人操作系统
RRT	Rapidly-Exploring Random Trees	快速扩展随机树
URDF	Unified Robot Description Format	机械臂描述格式
PRM	Probabilistic Road Map	概率路线图
ACM	Allowed Collision Matrix	免检冲突矩阵

目 录

前 言

机器人产业是我国国民经济的重要组成部分。为大力发展机器人产业，中央近年来先后出台了《“十四五”机器人产业发展规划》[1]、《“机器人+”应用行动实施方案》[2]等相关政策。

机器人系统比较复杂，包含机械、驱动、控制和感知等子系统，涉及机械、电子、人机交互、自动控制、软件工程和人工智能等多个领域，给机器人开发者和研究者带来了很大的挑战。为提高开发效率，避免“重复造轮子”的困境，ROS(Robot Operating System)机器人操作系统应运而生，为机器人开发提供了一套标准框架[3]。

ROS 可应用于机械臂、无人机、无人车及人形机器人等不同种类的机器人上。越来越多的机器人开发者、实验室和企业选择 ROS 作为机器人开发框架，用于机器人仿真和实物的研发测试。ROS 给用户提供了真实操作系统所应有的服务，包括硬件抽象，常用函数的实现，进程间消息传递，底层设备的控制，以及功能包管理[4]。它也提供用于获取、编译、编写和跨计算机运行代码所需的工具和库函数。

在现阶段，催化剂配方选型过程通常需要大量实验耗材以及大量的工时投入，这种方法效率低下，操作误差等问题难以避免。伴随着机器人技术和虚拟仿真水平的发展与提高，带有机械臂的虚拟实验平台正在被越来越多地研发，这将使虚拟实验探寻催化剂最佳配方成为现实。这类平台不仅可以避免上述弊端，而且可以加快实验研发进程，有效提高催化反应实验的工作效率，缩短催化剂开发周期。

从技术角度观察，ROS 系统是为提高机器人开发效率而建立的，而催化剂配方选型过程的全自动化实现离不开机器人技术。若能将二者结合起来，将在很大程度上提高相关机械臂的开发效率和配方实验的完成效率。

为了避免研发实验过程中的损失，新的技术和方案在应用到实验装置前，需经过仿真验证。由此看来，ROS 系统的机械臂仿真分析在催化剂配方的研发与优化设计方面也具有十分重要的意义。

1 绪论

1.1 选题背景和意义

为了筛选与优化催化剂配方，需要进行大量的实验验证，传统实验验证需要经过循环实验，逐步优化催化剂的组成和合成条件，以获得对某个特定反应具有最佳性能的催化剂配方。在现阶段，催化剂配方选型过程通常需要大量实验耗材以及大量的工时投入，这种方法效率低下，操作误差等问题难以避免。

伴随着虚拟技术和化工软件水平的发展与提高，虚拟实验平台正在被越来越多地研发并投入使用，这使虚拟实验探寻催化剂最佳配方成为现实。不仅可以避免上述弊端，而且可以加快实验研发进程，有效提高催化反应实验的工作效率，缩短催化剂开发周期。

借助 ROS 机器人系统搭建虚拟化工实验平台的机械臂，可以实现对实验平台的机械臂运行的优化和评估。这种虚拟机械臂的建立为研究人员提供了一个强大的工具，可以通过模拟和分析不同机械臂操作的耗时，以及它们在不同工作路径下的路径轨迹、速度和加速度等，从而提高探寻最佳的催化剂配方的开发效率。

由此看来，借助 ROS 机器人系统搭建虚拟化工实验平台的机械臂对探寻催化实验中催化剂的最佳配方具有重要意义，对于促进可持续发展、降低成本、加速研发过程都具有积极作用。通过虚拟化工实验平台机械臂的应用，我们可以更加高效地进行催化剂设计和优化，为实现环保和高效的化学工程技术提供重要支持。另外，该虚拟化工实验平台的机械臂能参与多种化学实验过程，应用场景广泛，在化学实验中有着较好的发展前景。

1.2 国内外研究现状

1.2.1 虚拟环境搭建

虚拟实验是科学实验领域中较为年轻的分支，其发展已经进入到虚拟现实阶段[5]。国内目前的虚拟实验基本以虚拟现实技术为基础，通过在仿真环境模拟人与周围真实环境交互方式来建立观察界面从而进行实验的过程。

针对无人生物安全实验室的复杂操作需求，马琳娜等人聚焦于多自由度高可靠性工业级机械臂的运动规划和实时避障技术[6]。她们基于生物安全实验室的设计规范，结合无人智能系统的设计理念，提出了移动无人生物安全实验室的设计方案。在此方案中，她们深入研究了基于 D-H 模型的机械臂运动学正解，以及与之相适应的生物安全实验内部仪器布局优化方法。

为了应对复杂环境中的碰撞风险，该研究团队建立了无人生物安全实验室机械臂和仪器设备的包围盒树包络模型(如图 1-1)，并基于此模型进行了详尽的碰撞分析。基于这一模型，他们提出了一个高效的碰撞判定算法，显著提高了机械臂在生物安全实验环境中的操作安全性。

为了实现机械臂的高效运动规划，该研究团队提出了一个约束机械手姿态的随机概率路图（CMO-PRM）机械臂运动规划算法。该算法不仅考虑了机械臂的运动学约束，还融入了实时避障策略，确保机械臂在复杂环境中能够安全、快速地完成任务。基于这一算法，她们编写了相应的机械臂运动路线规划程序，为无人生物安全实验室的自动化操作提供了有力支持。提出有限退避快速扩展随机树(RL-RRT)机械臂在线避障算法。

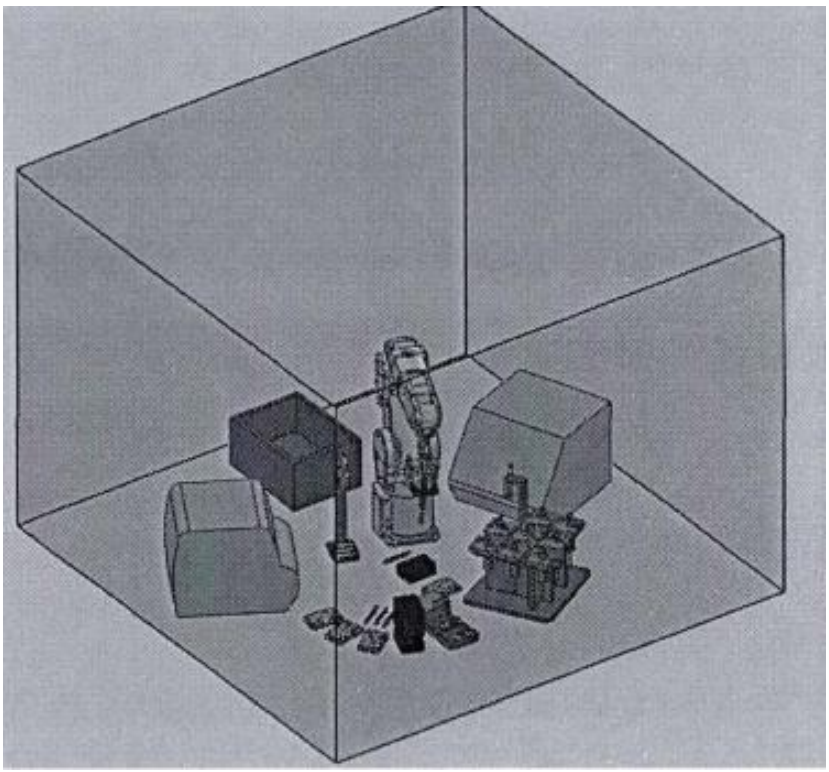


图 1-1 浙江大学马琳娜生物实验室示意图

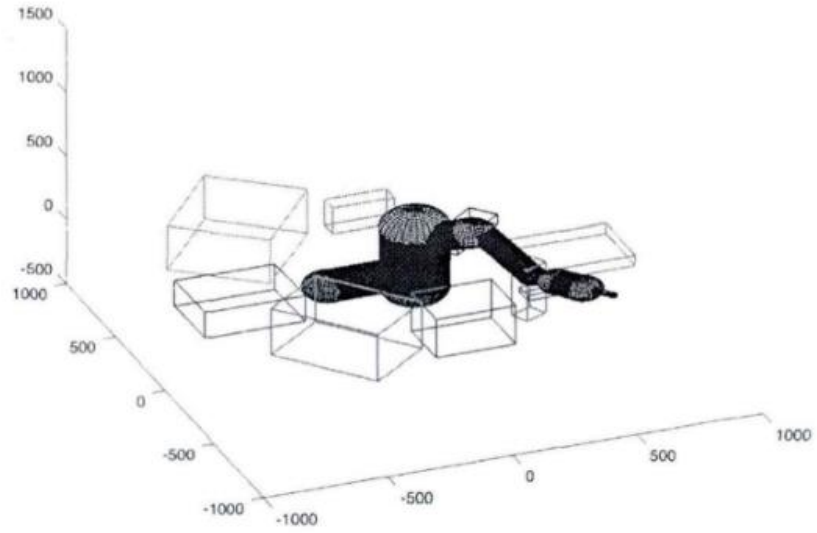


图 1-2 机械臂和仪器设备的包围盒树包络模型

在探索虚拟环境建模策略的过程中，哈尔滨工业大学的李廷睿在 ROS 系统架构内，巧妙地运用了 Gridmap 代码库，成功模拟了月球表面的参数化地形，并详细研究了不同参数对模拟效果的影响[7]。他的研究将月球表面细分为三大区域：月面岩石、月面陨石坑以及月面基础地形，如图 1-3 所示。为了准确模拟这些地形的复杂特性，他分别为月面岩石和陨石坑建立了详尽的数学模型。此外，他还对分形布朗运动模型进行了简要研究，旨在捕捉月球表面地形的随机性与复杂性。

在创建月面参数化地形时，李廷睿巧妙地运用了钻石-四边形算法，该算法以其高效和精确的特性，成功模拟出了月球表面的多样地形。完成建模后，他采用了 Rviz 作为可视化工具，将模拟的月球表面地形以生动的方式呈现出来。为了方便其他机器人仿真软件如 Gazebo 的使用，他借助 Terrain Generator 代码库将模型转换为 SDF 格式，这是一种广泛使用的机器人仿真文件格式。

针对三维地图构建，李廷睿对 LOAM 算法进行了深入探索。他全面分析了 LOAM 算法在构建三维地图时的优势与局限，并详细解读了其中的激光里程计算法与激光建图算法及其流程。他特别关注了激光里程计算法中的特征点提取、特征点匹配以及运动估计模块，并进行了详尽的公式推导。最终，在 ROS 系统环境下，他成功实现了 LOAM 算法，并通过在模拟的月球表面地形上进行三维点云地图的构建，验证了该算法的实用性。

不仅如此，李廷睿还对传统的 D 算法和自适应蒙特卡洛定位法进行了原理分析与验证。他探讨了 D 算法在模拟月球表面地形中的适用性，并通过在 ROS 系统下结

合 gmapping 代码库与模拟的月球表面地形，验证了这两种算法在模拟环境中的有效性。这些研究不仅为月球探测任务中的机器人定位、地图构建和路径规划提供了重要参考，也为相关领域的研究人员提供了宝贵的经验借鉴。

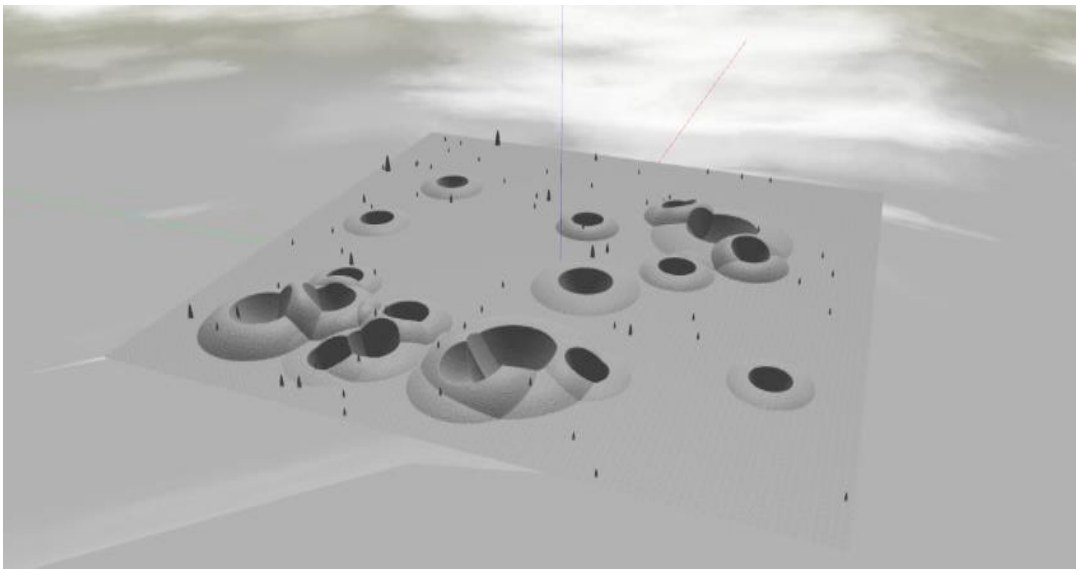


图 1-3 月面模拟地形图

国外虚拟实验起步较早当前国外有很多组织机构都在开展虚拟实验系统的研究和建设工作特别是在一些著名的大学和实验室[8]。从技术的核心视角来看，其基本理念是统一的，即通过软件手段来模拟和重现硬件设备的各种功能。这种模拟的实现方法多种多样，涵盖了基于 VRML 的虚拟现实技术来创建仿真实验[9]、利用 Active 技术开发的仿真实验、借助交互式 Flash 平台搭建的仿真实验[10]、以及基于 Java 技术和 QuickTime VR 技术的仿真实验等。在这些方法中，尤以基于 VRML 和 Flash 技术的仿真实验应用最为广泛。

1.2.2 虚拟仿真器

在化学实验仿真方面，国内由于相对缺乏相应的工业软件，大多数仿真化学实验集中在虚拟化学实验教学方面，将常规化学实验教学中无法开展的前沿、高难度实验内容与虚拟仿真技术相融合[11]，突破了化学实验教学条件限制和时空限制，实现了资源的共建共享(如图 1-4)。虚拟模拟体验作为一种颠覆性的教学模式[12][13]，展现了其无可比拟的简捷性、高度安全性和真实情境的精准模拟能力。这种教学模式能够模拟药物反应、毒性测试以及污染现象等实际化学品的特性，有效填补了传统化学实验教学中的实践空白，为学生提供了一个既安全又高效的学习平台。

图 1-4 虚拟化学实验室

国外在虚拟仿真器应用方面也颇有建树，韩国电子技术研究所提出了一个使用 IEC 61131-3 IDE 和开源 ROS 集成 Gazebo 模拟器的过程开发验证系统[14]。他们将机器人驱动的与物体操作相关的命令和代码与 Gazebo-ROS 系统通信，以验证自动化程序。仿真程序提供了一个可以测试自动化程序的虚拟环境(如图 1-5)。该虚拟环境允许开发人员分析其程序对整个系统的影响，从而简化了软件验证和确认的过程。该研究所为了实现无人机虚拟环境和学习环境[15]，使用了 Gazebo、ROS 和 Open-AI Gym。并利用对该虚拟环境下无人机的飞行数据进行强化学习，验证了强化学习算法的准确性。

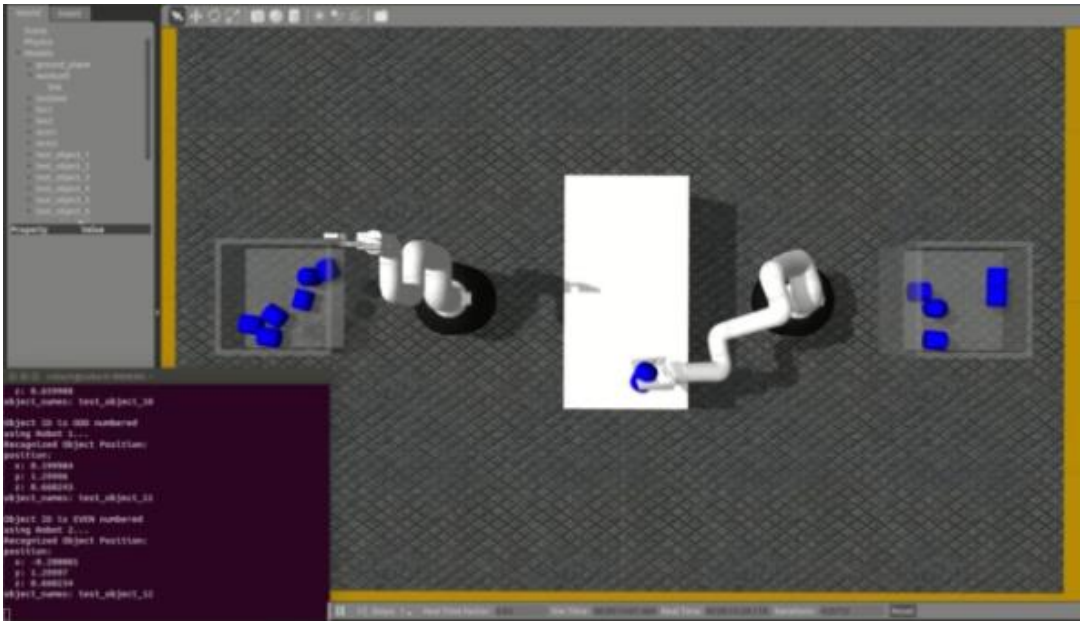


图 1-5 Gazebo-ROS 仿真实验环境

1.2.3 机械臂虚拟仿真

机械臂虚拟仿真的研究早已成为了机器人科研领域的一个重要分支[16]。机械臂的虚拟仿真主要是对机械臂进行正逆运动学特性及工作空间进行分析，对机械臂的动力学特性进行分析研究，对机械臂的运动轨迹进行规划并在此基础上研究各个关节的速度、加速度、力矩等物理学特性，完成对机械臂系统的综合评价，为机械臂实物的研发做前期准备和模拟。

西安电子科技大学结合虚拟现实技术的软件平台 VRML 语言对机器人三维仿真模型进行了分析和建模[17]，此平台结合 JAVA 语言完成系统的二维交互界面和系统在内的所有工程计算。哈尔滨工程大学张勇等人的机械臂模型由 Pro/E 三维软件画出，然后导出为虚拟现实中的模型，再通过接口技术，在 Simulink 中完成对虚拟机械臂的可视化操作[18](如图 1-6)。

东北大学的刘鹏团队巧妙地运用了虚拟现实技术，对上肢康复机器人的控制与仿真机制进行了深入探究[19]，并辅以图 1-7 的详细展示。在他们的研究中，团队充分考虑了痉挛扰动对系统可能产生的潜在影响，进而分别对康复机器人系统实施了传统控制、模糊控制以及基于模糊控制策略的仿真分析。经过一系列详尽且周密的仿真测试，结果显著地揭示出，模糊控制方法在面对康复机器人系统中存在的不确定性干扰时，展现出了极为有效的抑制能力。这一发现为康复机器人的精准控制提供了有力的技术支撑，并有望在未来的医疗康复领域发挥重要作用。

与张勇等人的研究路径相似，刘鹏团队也基于 MATLAB 的虚拟现实建模工具，精心构建了机器人的三维仿真模型。结合 MATLAB/Simulink 强大的计算仿真功能，他们设计并实现了一个具备五自由度的上肢康复机器人仿真系统。这一系统的成功搭建，不仅为康复机器人的后续研究与发展提供了有力的技术支撑，也为相关领域的研究人员提供了宝贵的实验平台。

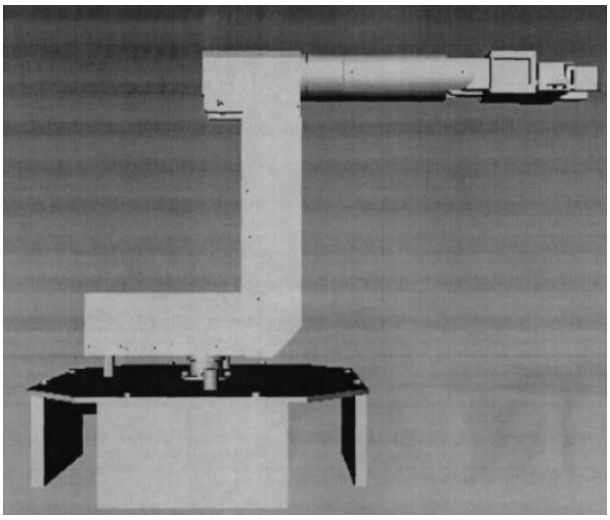


图 1-6 哈工程张勇虚拟机械臂模型

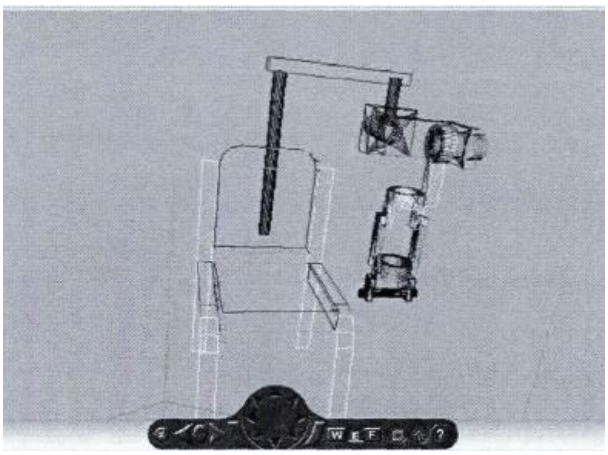


图 1-7 上肢康复机器人 VR-Sink 效果图

威奇托州立大学 Mousa Moradi 等人基于 IMU-Vision 的实时混合控制算法用于与 6-DOF Kinova 虚拟机械臂进行交互[20]。这篇文献提出的人机交互（HRI）控制方案利用了来自 Myo 手势控制臂带惯性测量单元的嵌入式陀螺仪传感器和来自 Microsoft HD 相机的 800*600 像素分辨率。该算法利用数值离散时间积分器和均值滤波器的数学特性来处理陀螺仪的原始角速度数据。处理后的数据进一步在用户沿 x、y 和 z 轴的顺时针或逆时针动作期间向机械臂的末端执行器提供角位移。这也促进了末端执行器(夹持器)的运动，该运动也通过算法中的阈值比较由滚动动作同时控制。使用计算机视觉工具箱和斑点分析技术设计了一个基于视觉的反馈系统，以使系统更可靠，并在到达所需物体时控制末端执行器的距离。结果表明，使用陀螺仪信息和用户输入对六自由度虚拟机器人肢体进行了显著控制。虚拟机械臂在如预期的那样到达距离所需物体 320 毫米后停止移动。对于三个不同的物体，圆柱形物体的实际距离和测量距离之间的最大误差计算为 15.3cm。由于其流畅地控制了手臂手势控制器，这

项技术有可能在不久的将来帮助身体残疾或神经系统疾病的人使用辅助机械臂进行日常生活活动(如图 1-8)。

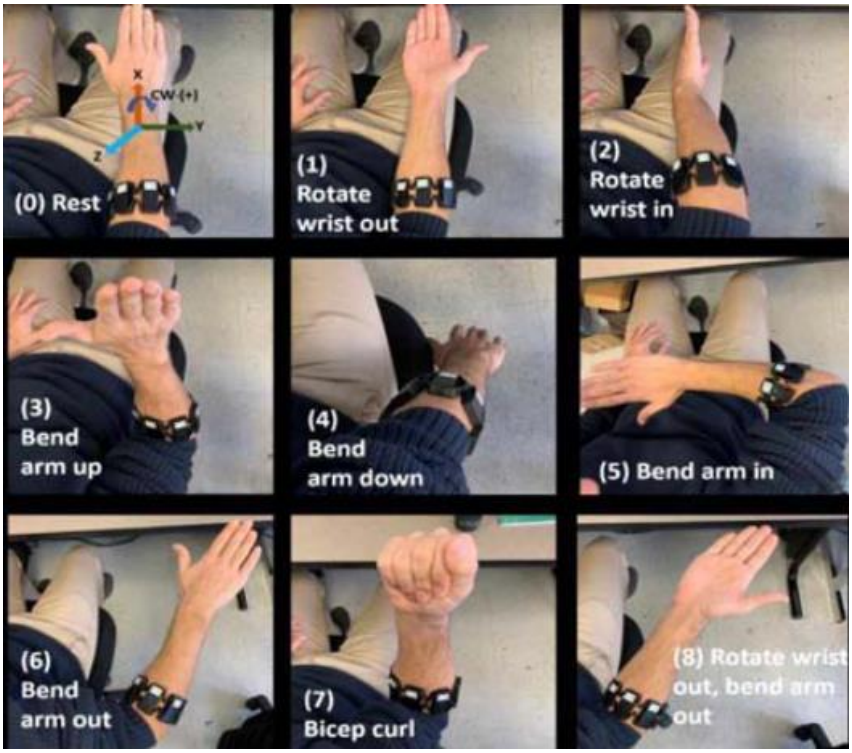


图 1-8 采集 IMU 数据的九种手势图

早稻田大学的 Daichi Watanabe 团队开发了一款可穿戴机器人附属装置，该装置在增强现实(AR)环境中为用户提供了触觉反馈。这款系统包含一个佩戴在用户前臂上的四自由度机器人和一个方形末端执行器(如图 1-9)，能够模拟各种触觉感受。通过这一系统，用户可以在 AR 环境中获得更加真实和沉浸式的交互体验。文献中展示了该系统在 AR 中提供 UI 触觉体验的潜在应用，并进行了初步评估，得到了用户的积极反馈。这一创新技术为未来的 AR 交互设计提供了新的思路。

图 1-9 可穿戴机器人附属装置图

1.2.4 机械臂轨迹规划算法

中北大学潘振钊从机械臂轨迹优化和机械臂轨迹跟踪这两方面展开了研究，使其在稳定运行的同时提高作业精度保证在各种工业作业中的应用[22]。他基于改进 P-RRT*算法对其机械臂进行了机械臂运动规划。通过结合目标偏置策略和基于目标检测的避障措施，在 RRT(快速扩展随机树)算法的基础上，潘振钊等人进行了精心的优化，以确保随机树能够更高效地定向生长至目标点。一旦机械臂巧妙地避开了障碍物，经过优化的随机树会立即且精准地锁定目标位置，并继续其生长过程。在 MATLAB 搭建的二维环境中，他们对三种 RRT 算法进行了实验对比，结果显示，这种改进后的算法显著提升了机械臂的避障效率，并且生成的路径更为短捷，有效地证明了其高效性和实用性。他深入探讨了机械臂中 P-RRT*算法的改进方案，并特别关注了这些改进在机械臂关节空间内的应用。为此，他进行了一系列关于直线插值、三次插值、五次插值等多项式插值的仿真和实验验证(如图 1-10 所示)。这些验证不仅证明了改进策略的有效性，还显著提升了机械臂轨迹规划的精确度和适应性。

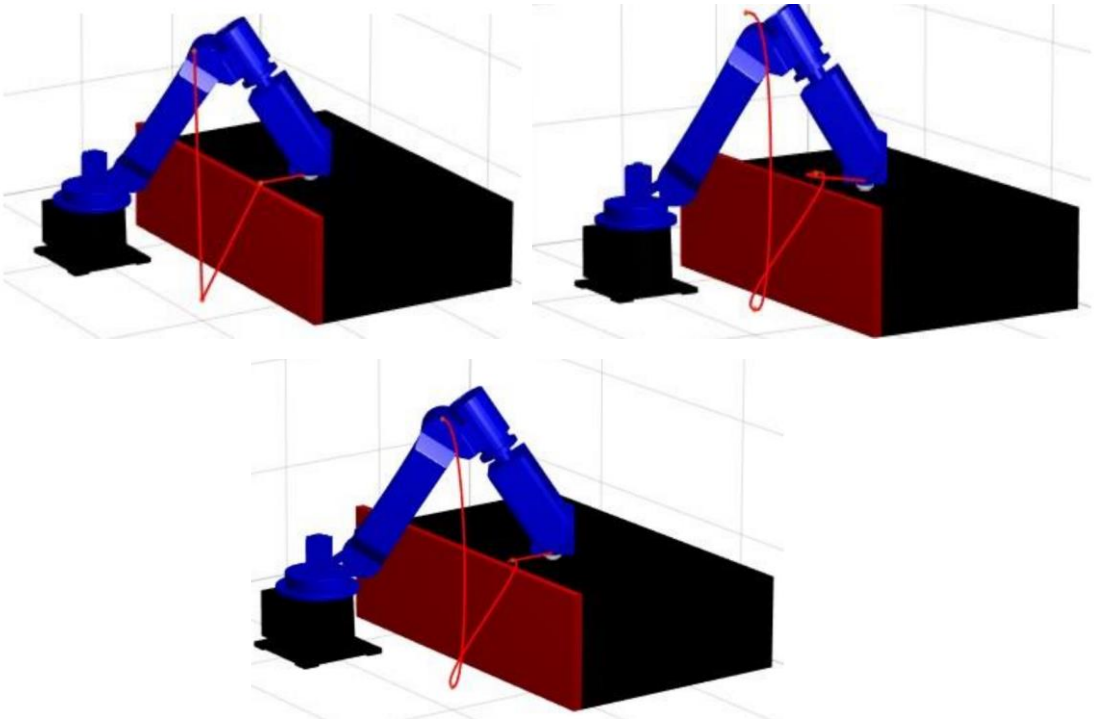


图 1-10 改进 P-RRT*算法直线插值、三次、五次等多项式插值仿真图

在机器人避障规划问题中，当障碍物影响机器人的末端运动时，机器人的避障运动和末端路径跟踪运动相互竞争。塞尔维亚贝尔格莱德大学 Aleksandar Rodic 和 Petar B. Petrovic 等人针对这一问题，提出了一种基于主从任务变换的避障算法[23]。为了减少计算负载，使用矢量投影方法对可能与障碍物碰撞的机械手单元进行了初

步筛选和消除，这也统一和简化了障碍物环境。在避障问题的初级解决阶段，对机械臂使用传统无主从任务变换避障算法(如图 1-11)。在避障问题的最终解决方案中，使用了主从转换变量的避障算法(如图 1-12)。响应相较于实时最小距离的变化，实现了机械臂末端的轨迹跟踪运动和防撞运动之间的平滑以及连续的优先级转换。为了减少终端轨迹误差，创建了误差调整系数以实时向跟踪功能报告类似终端避障的实际轨迹。根据仿真结果，当障碍物意外到达其预期任务轨迹时，以及当障碍物在距离其约 0.02 米的范围内时，末端执行器可以可靠地避开障碍物。这确保了机械臂能够精确到达目标位置。

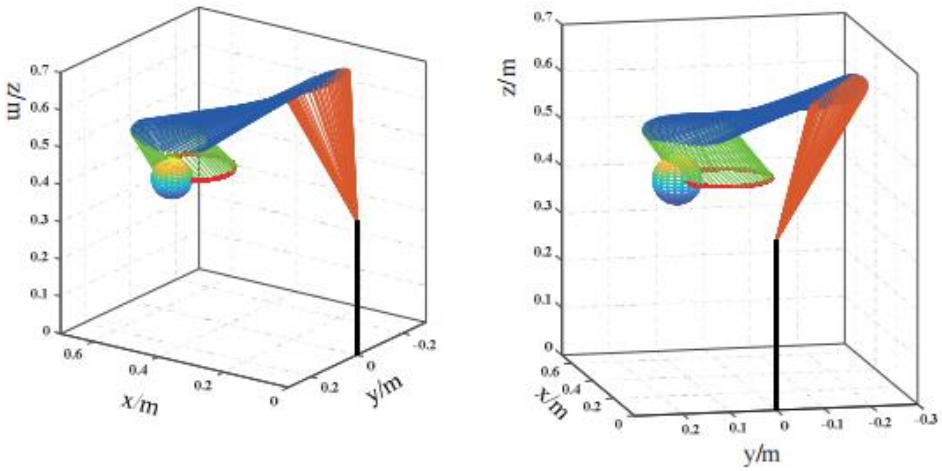


图 1-11 传统无主从任务变换避障结果图

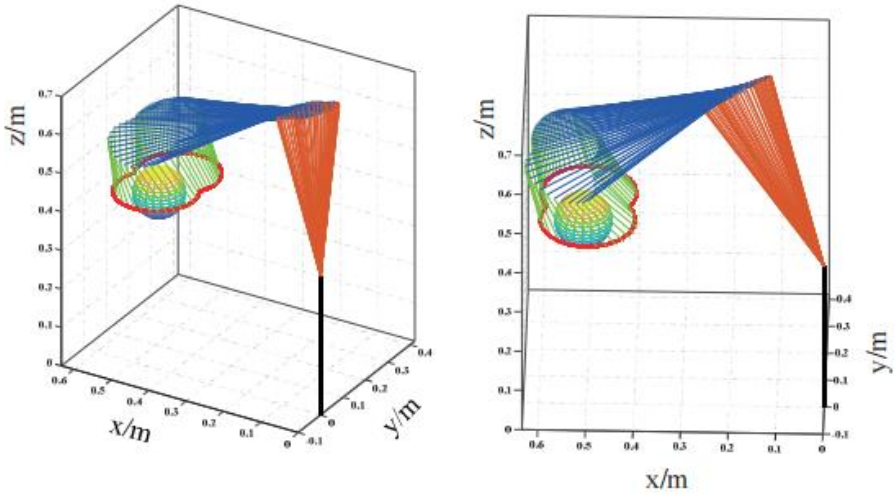


图 1-12 基于主从任务变换的避障算法结果图

1.3 主要研究内容

本文在第一章完成对选题背景和意义的探讨，展现了国内外在虚拟环境搭建、六自由度机械臂虚拟仿真研究、机械臂轨迹规划方面的进展。在第二章完成了对六自由度机械臂的理论研究，为在仿真环境控制机械臂做理论准备。在第三章完成在 Solidworks 中完成机械臂建模，并完成从 Solidworks 到 ROS 系统 URDF 模型文件的转换。在第四章完成 ROS 系统的 Rviz 三维可视化界面完成对虚拟机械臂的仿真控制并展现了本课题所完成的 ROS 系统的通讯计算图；在第五章，本文分别使用 RRT 算法和 RRTStar 算法对虚拟机械臂进行轨迹规划，并对两种算法所规划的路径做了关节空间角度和速度随时间变化的分析，最后使用 RRTStar 算法展现了避障过程。本文的主体结构如图 1-5 所示。

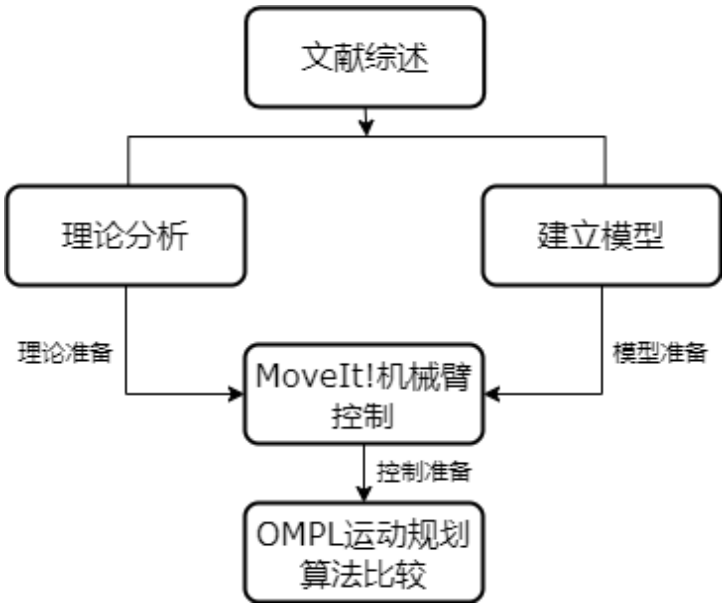


图 1-5 全文结构图

2 六自由度机械臂理论分析

2.1 六自由度机械臂运动学

六自由度机械臂运动学包含正运动学和逆运动学[24]。在给定机械臂各关节角度的条件下，机械臂的正运动学研究的是如何计算其末端执行器的位置和姿态。这一过程实际上是将机械臂的关节空间参数映射到笛卡尔空间，以便更直观地理解机械臂的运动状态。而机械臂的逆运动学则与之相反，它关注的是在已知机械臂末端执行器位置和姿态的条件下，如何反推出相应的关节角度。这一过程实现了从笛卡尔空间到关节空间的映射，对于实现机械臂的精确控制和规划至关重要。

2.1.1D-H 描述法

Denavit-Hartenberg（D-H）描述法是一种对机械臂连杆和关节进行建模的非常简单的方法[25]，可用于绝大多数机械臂模型，也可用于表示机械臂任何坐标的变换(如图 2-1)。

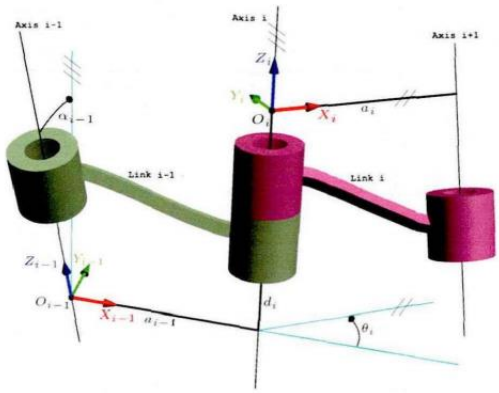


图 2-1 D-H **描述法**

D-H 描述法以其简洁性著称，它仅需四个关键参数即可描述原本需要六个自由度来表达的关节之间的变换。这四个参数分别是：连杆长度 a_{i-1}、连杆扭转角 α_{i-1}、连杆偏移 d_i 以及关节转角 θ_i。通过严格遵循 D-H 描述法的坐标系标定规则，我们可以为六自由度机械臂的每个关节和连杆精准地定义其坐标系，进而得到如图 2-2 所示的坐标系配置。这一方法不仅简化了机械臂的运动学建模过程，也为后续的运动规划和控制提供了便利。

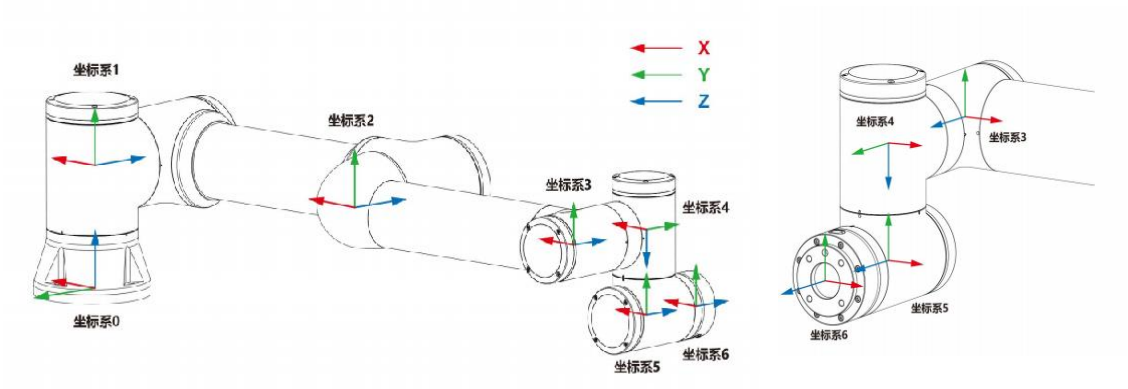


图 2-2 机械臂坐标系图

根据上图所建立的连杆坐标系确定该机械臂的 D-H 参数 a_{i-1}、α_{i-1}、d_i、θ_i。a_{i-1} 为从 Z_{i-1} 到 Z_i 沿 X_{i-1} 测量的距离，α_{i-1} 为从 Z_{i-1} 到 Z_i 绕 X_{i-1} 旋转的角度，d_i 为从 X_{i-1} 到 X_i 沿 Z_i 测量的距离，θ_i 为从 X_{i-1} 到 X_i 绕 Z_i 旋转的角度。依此规则建立 D-H 参数表，如表 2-1。

表 2-1 改进 D-H 参数表

i	a_{i-1}/mm	α_{i-1}/°	d_i/mm	θ_i/°
1	0	90	152	θ_1(0)
2	-425	0	0	θ_2(0)
3	-395	0	0	θ_3(0)
4	0	90	102	θ_4(0)
5	0	-90	102	θ_5(0)
6	0	0	100	θ_6(0)

在 MATLAB 中完成相应的代码，得到机械臂的 D-H 模型图，如图 2-3。

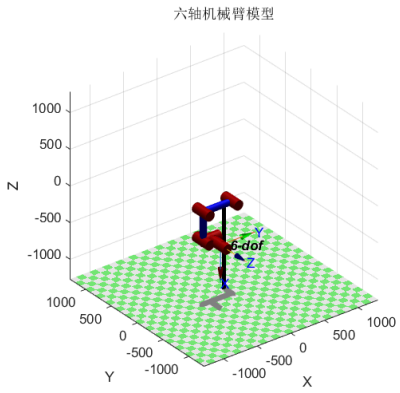


图 2-3 D-H 模型图

2.1.2 六自由度机械臂正运动学

机械臂正运动学是在已知机械臂关节角度的情况下计算末端的位置和姿态，即从关节空间映射到笛卡尔空间[26]。根据 D-H 描述法，确定每两个相邻连杆所需的四个参数后，就能够按照平移旋转公式建立相邻连杆的相对位姿。

$$ {}^{i-1}_{i}T = Rot\left(z,\ \theta_i\right)Trans(0,0,d_i)Trans(a_{i-1},0,0)Rot\left(x,\ \alpha_{i-1}\right) \tag{2-1} $$

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

其中 c 和 s 分别为 cos 和 sin 的缩写。

分别计算各相邻连杆的齐次变换矩阵后，可得到末端连杆 n 固连坐标系{N}相对于基坐标系{0}的齐次变换矩阵。

$$ {}^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \cdots {}^{N-1}_{N}T \tag{2-2} $$

根据 D-H 参数表依次列写每两个相邻连杆的齐次变换矩阵。

$$ \mathbf{{}^{1}_{0}T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -152 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{{}^{2}_{1}T} = \begin{bmatrix} 1 & 0 & 0 & -425 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \mathbf{{}^{3}_{2}T} = \begin{bmatrix} 1 & 0 & 0 & -395 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{{}^{4}_{3}T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -102 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-3} $$

$$ \mathbf{{}^{5}_{4}T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 102 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{{}^{6}_{5}T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

将上述六个齐次变换矩阵按顺序依次相乘，即可得到机械臂末端法兰坐标系相对于基坐标系的相对齐次变换矩阵$\mathbf{{}^{6}_{0}T}$（如式 2-4）。

$$ \mathbf{{}^{6}_{0}T} = \mathbf{{}^{1}_{0}T\,{}^{2}_{1}T\,{}^{3}_{2}T\,{}^{4}_{3}T\,{}^{5}_{4}T\,{}^{6}_{5}T} = \begin{bmatrix} 1 & 0 & 0 & -820 \\ 0 & 0 & -1 & -354 \\ 0 & 1 & 0 & -102 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-4} $$

2.1.3 六自由度机械臂逆运动学

将机器人目标位置与当前位置的差定义为向量函数$f(X)$：

$$f(X)=\begin{bmatrix}x_{target}-x\\ y_{target}-y\\ z_{target}-z\end{bmatrix} \tag{2-5}$$

建立关节角度组为X：

$$X=[\theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]^T \tag{2-6}$$

建立目标函数$F(X)$：

$$F(x)=\frac{1}{2}f^T(X)f(X) \tag{2-7}$$

这样就将机器人运动学求逆问题转化为求min F(X)问题。

F(X)的一阶泰勒展开式为式 2-7。

$$F(X+k\vec{h})=F(X)+F'(X)\cdot(k\vec{h})+o(k\vec{h}) \tag{2-7}$$

其中，$\vec{h}$为二维向量，表示X偏移的方向，k为$\vec{h}$的系数。

于是有式 2-8。

$$\lim_{k=0}\frac{F(X)-F(X+k\vec{h})}{k\|\vec{h}\|}=-\frac{F'(X)\vec{h}}{\|\vec{h}\|}=-F'(X)\cos\alpha \tag{2-8}$$

其中，α是 h 和F′(X)的夹角。

关节位移量为式 2-9。

$$\begin{bmatrix}\Delta\theta\\ \Delta r\end{bmatrix}=-kF'(X) \tag{2-9}$$

2.1.4 六自由度机械臂工作空间分析

利用蒙特卡洛法对该机械臂进行工作空间分析[27]，在关节空间均匀选取随机角度组合，再按机械臂的正运动学分析机械臂末端可能到达的点位置集合，按式 2-10 计算所到达的工作点，构成工作空间。

$$\theta_i=(\theta_{iM}-\theta_{im})rand+\theta_{im}，\ i=1\sim6 \tag{2-10}$$

其中θ_i为所需的随机角度，θ_{iM}、θ_{im}为关节i可到达的最大和最小角度，$rand$为0-1 的随机值。

计算工作空间所需的关节限制参数如表 2-2 所示。

表 2-2 机械臂关节限制参数

关节编号 i	转动范围	转速范围	关节名称
1	(-179°,179°)	15-150°/s	肩偏航
2	(-269°,89°)	15-150°/s	肩俯仰
3	(-162°,162°)	15-150°/s	肘俯仰
4	(-269°,89°)	15-180°/s	腕俯仰
5	(-179°,179°)	15-180°/s	腕偏航
6	(-179°,179°)	15-180°/s	腕滚动

随机选取 100000 个关节角度数组时，机械臂工作空间如图 2-4 所示。

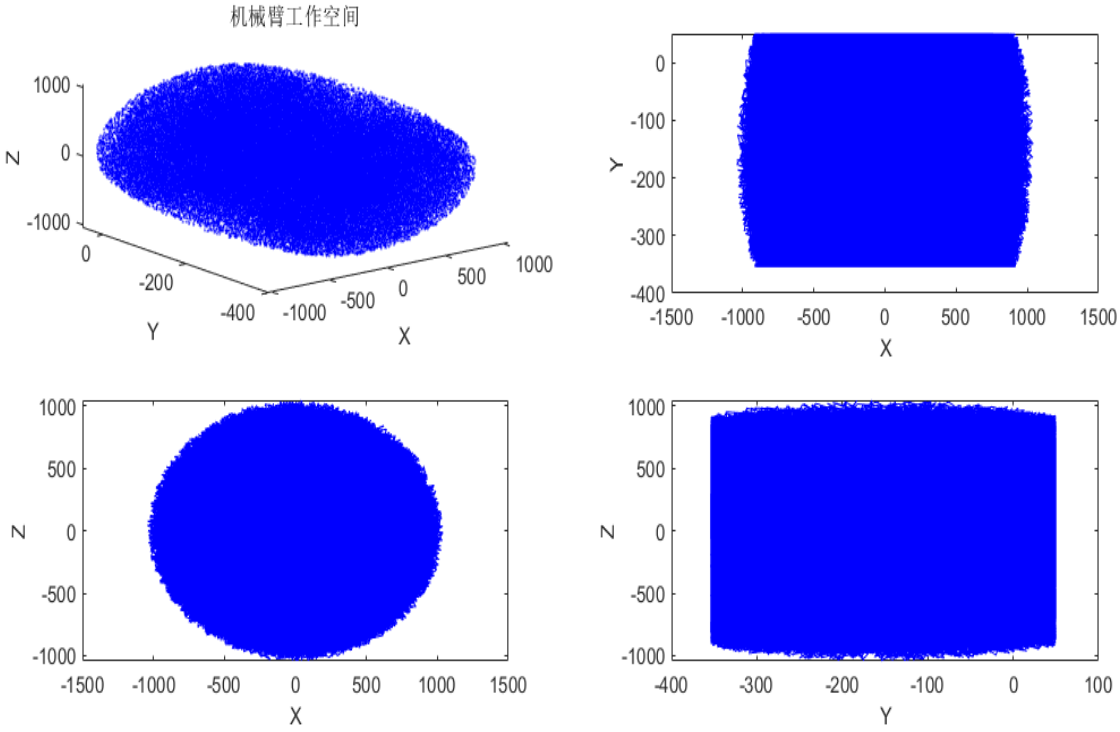


图 2-4 机械臂工作空间

2.2 六自由度机械臂动力学

2.2.1 拉格朗日公式

对于六自由度机械臂系统而言，将该系统的动能与势能之差定义为拉格朗日函数 L，拉格朗日公式为式 2-11[28]。

$$\frac{d}{dt}\frac{\partial L}{\partial q}-\frac{\partial L}{\partial q}=0 \tag{2-11}$$

在动力学分析中，拉格朗日函数 L 是一个重要的概念，它定义为动能 T 与势能 V 之差，即$L=T-V$。基于这个拉格朗日函数，我们可以构建出拉格朗日动力学方程，用以描述机械系统的动态行为。

具体地，拉格朗日动力学方程可以表示为：$\tau=M(\theta)\ddot{\theta}+C\left(\theta,\dot{\theta}\right)\dot{\theta}+G(\theta)$。在这个方程中，$\tau$表示关节所需的输入力，它是一个 n×1 的向量。$M(\theta)$是质量项系数，它是一个 n×n 的矩阵，并且这个矩阵的元素是电机角度θ的函数。$C\left(\theta,\dot{\theta}\right)$代表离心力和科氏力项系数，同样是一个 n×n 的矩阵，它的值取决于关节角度θ和角速度$\dot{\theta}$。而$G(\theta)$则是重力项系数，它是一个 n×1 的向量。

为了应用这个方程，我们需要求解上述各项系数。具体的求解算法将依据机械臂系统的具体结构、参数和约束条件来确定。这些系数的准确求解对于理解和控制机械系统的动态行为至关重要。

假设质量项系数矩阵$M(\theta)$、离心力和科氏力项系数矩阵$C\left(\theta,\dot{\theta}\right)$、重力项系数矩阵$G(\theta)$分别为式 2-12、式 2-13 和式 2-14。

$$M(\theta)=\left[h_{i,j}\right],\ i,j=1,2,\dots 6 \tag{2-12}$$

$$C\left(\theta,\dot{\theta}\right)=\left[c_{i,j}\right],\ i,j=1,2,\dots 6 \tag{2-13}$$

$$G(\theta)=[g_i],\ i=1,2,\dots 6 \tag{2-14}$$

其中$h_{i,j}$、$c_{i,j}$、g_i的计算公式为式 2-15 到式 2-17。

$$h_{i,j}=\sum_{k=\max(i,j)}^{6} tr\left(\frac{\partial {}^0T_k}{\partial\theta_i}J_K\frac{\partial({}^0T_k)^T}{\partial\theta_j}\right),\ i,j=1,2,\dots 6 \tag{2-15}$$

$$c_{i,j}=\sum_{k=1}^{6}\frac{1}{2}\left(\frac{\partial h_{i,j}}{\partial\theta_k}+\frac{\partial h_{i,k}}{\partial\theta_j}-\frac{\partial h_{j,k}}{\partial\theta_i}\right)\theta_k,\ i,j=1,2,\dots 6 \tag{2-16}$$

$$g_i = -\sum_{j=1}^{6} m_j g^T \frac{\partial^0 T_j}{\partial \theta_i} P_{Cj}\ ,\ i = 1,2,\dots 6 \qquad (2-17)$$

式中 T 为连杆变换齐次矩阵，J 为雅各比矩阵，m 为连杆质量，P_C为连杆质心位置。

根据上述公式可得到机械臂每一关节力矩值的解析解。

2.2.2 六自由度机械臂动力学方程

2.2.2.1 拉格朗日直接法

系统的总动能为机械臂所有连杆动能之和为式 2-18。

$$T = \sum_{i=1}^{n} T_i = \sum_{i-1}^{n} \left(\frac{1}{2} m_i v_{cm_i}^T v_{cm_i} + \frac{1}{2} w_{cm_i}^T I_i w_{cm_i} \right) \qquad (2-18)$$

系统的总势能为机械臂所有连杆势能之和为式 2-19。

$$V = \sum_{i}^{n} V_i = \sum_{i-1}^{n} m_i g^T p_0^{cm_i} \qquad (2-19)$$

将$L = T - V$带入拉格朗日方程得式 2-20。

$$\frac{d}{dt}\frac{\partial L}{\partial q} - \frac{\partial L}{\partial q} = 0 \qquad (2-20)$$

解算拉格朗日方程并重新组合得式 2-21。

$$\tau = M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right)\dot{\theta} + G(\theta) \qquad (2-21)$$

2.2.2.2 拉格朗日迭代法

随着连杆数量的增多，拉格朗日直接法的计算会变得越来越复杂，为了简化计算量，提高计算效率可以选用拉格朗日迭代法。

将系统的总动能和总势能方程分别拆分单个连杆的动能和势能方程，如式 2-22 和式 2-23。

$$T_i = \frac{1}{2} m_i v_{cm_i}^T v_{cm_i} + \frac{1}{2} w_{cm_i}^T I_0^i w_{cm_i} \qquad (2-22)$$

$$V_i = m_i g^T p_0^{cm_i} \qquad (2-23)$$

其中连杆质心位置$p_0^{cm_i}$、质心速度v_{cm_i}、质心角速度w_{cm_i}、转动惯量I_0^i变为未知量，这些量均是与机械臂的运动状态相关的量。

对这些未知量进一步拆分，将其拆分成与关节角有关的坐标转换和与关节角无关的物理量。质心位置$p_0^{cm_i} = p_0^i + R_0^i p_i^{cm_i}$，质心速度$v_{cm_i} = J_{v_i}^{cm_i}\dot{q}$，质心角速度$w_{cm_i} = J_{w_i}^{cm_i}\dot{q}$，转动惯量$I_0^i = R_0^{cm_i} N_{cm_i}\left(R_0^{cm_i}\right)^T$，$R_0^{cm_i} = R_0^i R_i^{cm_i}$。

机械臂旋转关节的雅克比求解公式为式 2-24。

$$J_{cm_i} = \begin{bmatrix} J_{v_i}^{cm_i} \\ J_{w_i}^{cm_i} \end{bmatrix} = \begin{bmatrix} z_i \times p_i^{cm_i} \\ z_i \end{bmatrix} \tag{2-24}$$

将拆分后的变量带入系统动能方程式 2-25。

$$T = \sum_{i=1}^{n} T_i = \sum_{i-1}^{n} \left(\frac{1}{2} m_i v_{cm_i}^T v_{cm_i} + \frac{1}{2} w_{cm_i}^T I_i w_{cm_i} \right) = \frac{1}{2} \dot{q} M(q) \dot{q} \tag{2-25}$$

其中$M(q)$为惯性矩阵。

此时系统动能表达式变成只和关节角度有关，通过分析可知，动能方程只在速度为 0 的时候为 0，而惯性矩阵是一个对称正定矩阵。

将动能方程由矩阵的形式转换成累加的形式再带入拉格朗日量公式中得到式 2-26 和式 2-27：

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} = \frac{d}{dt}\left(\sum_j M_{kj}\dot{q}_j \right) = \sum_j M_{kj}\ddot{q}_j + \sum_{i,j} \frac{\partial M_{ij}}{\partial q_i} \dot{q}_i \dot{q}_j \tag{2-26}$$

$$\frac{\partial L}{\partial q_k} = \frac{1}{2} \sum_{i,j} \frac{\partial M_{ij}}{\partial q_i} \dot{q}_i \dot{q}_j - \frac{\partial P}{\partial q_k} \tag{2-27}$$

将拉格朗日量带入到拉格朗日方程中得到式 2-28：

$$\tau_k = \sum_j M_{kj}\ddot{q}_j + \sum_j C_{kj}\dot{q}_j + \frac{\partial P}{\partial q_k} \tag{2-28}$$

将六自由度机械臂的相关参数带入，根据上述拉格朗日动力学公式可以得出相应的机械臂的动力学方程。关于机械臂的 D-H 参数已经详细的列在表 2-1 中，同时，表 2-3 则详细记录了机械臂各连杆的质量、长度以及每个连杆质心的具体位置等关键参数。在这个列表中，M 代表连杆的质量，L 代表连杆的长度，而 R 则代表了连杆

质心到其对应关节轴线的距离。此外，对于在机械臂末端安装的如抓手等工件，它们的质量和长度也被标注为 M_x 和 L_x。

这些参数和量值在动力学方程中的应用，可以清晰地揭示出相关因素对机械臂动力学性能的影响程度。例如，在考虑空间机械臂时，由于处于失重环境，重力项在动力学方程中的影响可以忽略不计，此时质量项成为主导因素。另外，当机械臂各关节的角速度保持在较低水平，如小于 1°/s 时，科氏力和离心力项在动力学方程中的影响也可以近似忽略。这样的分析有助于我们更准确地理解和预测机械臂在不同条件下的动力学行为。

连杆编号 i	M/Kg	L/mm	R/mm
1	1.62	152	0
2	4.38	425	212.5
3	14.46	395	183.4
4	7.67	102	51
5	1.63	102	48.8
6	1.58	100	50
7	$0.53+M_x$	L_x	0

在机械臂的实际应用场景中，由于摩擦力、突发的外部冲击力以及复杂多变的作业环境等不确定性因素的存在，建立一个精确的机械臂数学模型往往面临极大的挑战。因此，在不涉及外界环境干扰的假设条件下，本节主要探讨和描述了机械臂的力与运动之间的基础关系。这一探讨不仅为理解机械臂的基本运动特性提供了理论依据，也为后续在仿真环境中对机械臂进行精确控制的研究奠定了坚实的基础。

2.3 本章小结

本章主要完成了对六自由度机械臂的理论分析。首先，建立了机械臂的 D-H 参数表，准确描述了其几何结构和运动特性。接着，进行了正逆运动学分析，探究了关节变量与末端执行器位置和姿态的关系。最后，本章构建了机械臂的拉格朗日动力学模型，以深入理解其动态响应。这些分析将为后续控制系统设计、轨迹规划及仿真实验验证工作提供了坚实的理论基础。

3 协作六自由度机械臂可视化模型构建

本章主要完成六自由度机械臂的 SolidWorks 的建模以及从 SolidWorks 到 ROS 系统三维模型文件 URDF 的转换，并对该文件进行解析，完成在 Rviz 中显示该模型。

3.1 SolidWorks 机械臂模型建立

为了完成运动学、动力学及六自由度机械臂的控制和轨迹规划的虚拟仿真，通过 SolidWorks 建立了该机械臂的三维模型。

在 SolidWorks 软件中，我们按照 1:1 的尺寸比绘制了六自由度机械臂的外形，并基于各连杆和关节的连接关系在装配图中设置了配合关系。最终，本课题得到了如图 3-1 所示的三维模型，它准确模拟了机械臂的结构和运动状态，为后续的研究提供了重要参考。

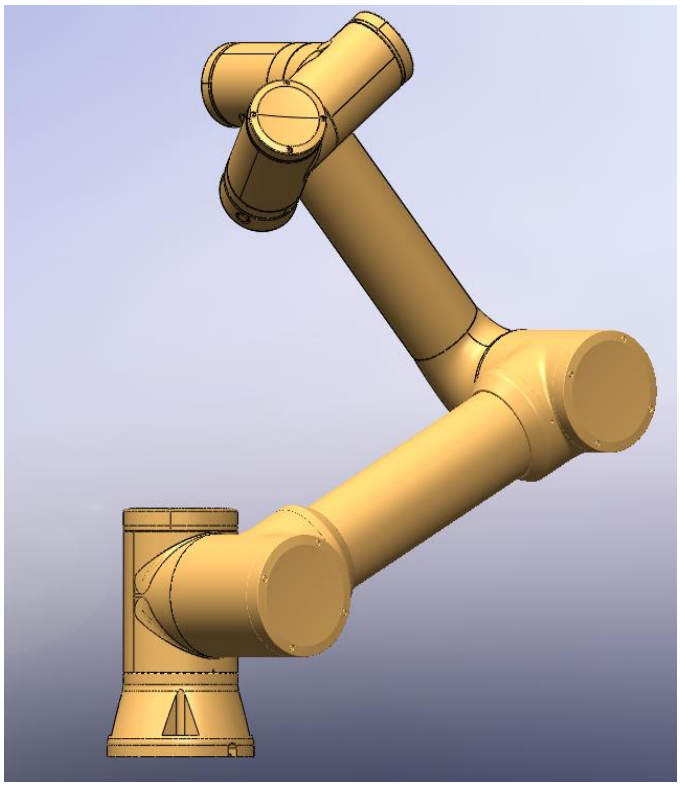


图 3-1 机械臂 SolidWorks 三维模型图

3.2 SW 到 URDF 的转换

在对复杂构造的机械臂进行模型创建时，ROS 提供了将 Solidworks 里绘制的模型导出成 URDF 文件并导入 ROS 的插件 sw2urdf。sw2urdf 是一款专为 SolidWorks 设计的插件，它的主要功能是将 SolidWorks 中的零件和装配体轻松转换为 URDF 文件。该导出器插件在转换过程中会生成一个类似于 ROS 软件包的目录结构，其中包含了网格模型、纹理贴图以及机械臂的 URDF 文件。

对于单独的 SolidWorks 零件，这款零件导出器能够智能地提取其材料属性，并在生成的 URDF 文件中创建相应的链接描述。而对于装配体，导出器则能够依据 SolidWorks 装配体的层次结构，自动构建链接关系并生成树状结构，从而完整地反映装配体的结构特性。此外，sw2urdf 插件还具备自动识别功能，它能够根据零件间的装配关系，自动确定合适的连接类型、连接变换以及转轴参数，极大地简化了从 SolidWorks 到 URDF 的转换过程，为后续的机器人仿真和控制提供了极大的便利。

首先需要在装配体中完成连杆属性的设置，建立连杆树（如图 3-2）。对于每个连杆，需要给定它一个唯一的名称，在属性设置中添加与该连杆相关联子连杆，同时需要指定与该子连杆的关节属性。需要的设置和修改的连杆属性包括各连杆的质量，各连杆相对位置坐标及各连杆的转动惯量矩阵，具体各数值通过该插件自动生成和人工修改汇总如下表 3-1 和表 3-2。

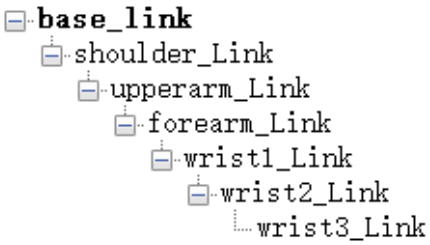


图 3-2 Solidworks 连杆配置树

表 3-1 连杆质量和坐标原点参数表

连杆名称	质量 m(kg)	坐标原点位置(m)		
		X	Y	Z
基底连杆	1.62	-0.0004	0.0005	0.0326
肩部连杆	4.38	0.0000	-0.0031	0.1453
上臂连杆	14.46	0.0000	0.3328	0.0842

前臂连杆	7.67	0.0922	0.0000	0.0842
手腕连杆 1	1.63	0.4872	0.0000	0.0842
手腕连杆 2	1.58	0.0000	0.0046	0.0993
手腕连杆 3	0.53	0.0000	0.0001	0.0791

表 3-2 连杆转动惯量矩阵参数表

连杆名称	转动惯量矩阵(Kg·m²)					
	i_{xx}	i_{xy}	i_{xz}	i_{yy}	i_{yz}	i_{zz}
基底连杆	2.457E-03	-1.544E-05	-1.351E-05	2.458E-03	1.442E-05	3.888E-03
肩部连杆	1.040E-02	-1.005E-08	1.627E-07	1.028E-02	1.586E-04	7.631E-03
上臂连杆	2.907E+00	-2.310E-04	-2.505E-05	1.265E-01	-1.479E-01	2.816E+00
前臂连杆	1.265E-01	2.310E-04	1.479E-01	2.907E+00	-2.505E-05	2.816E+00
手腕连杆 1	1.265E-01	2.310E-04	1.479E-01	2.907E+00	-2.505E-05	2.816E+00
手腕连杆 2	2.239E-03	-1.002E-07	-3.212E-08	1.391E-03	-2.222E-05	2.186E-03
手腕连杆 3	2.780E-04	-3.535E-07	-1.743E-07	2.774E-04	-7.589E-08	4.158E-04

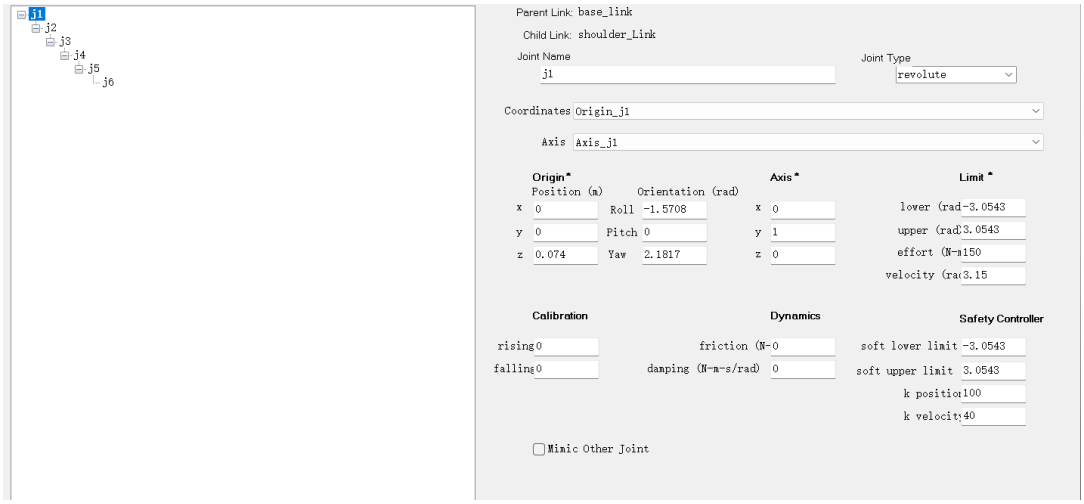


图 3-3 Solidworks 关节属性配置界面

在关节属性界面完成对机械臂关节属性的设置（如图 3-3），设置所有的关节属性为有最大最小关节角度限制的旋转关节。关节属性还需要设置其上级连杆和下级连杆，其固连坐标系的原点位置，并设置所有关节均绕其固连坐标系的 Z 轴旋转，具体数值见表 3-3。除此之外，还需要设置每个关节的最大关节角度的上下限，最大关节可提供的转矩以及每个关节的最大关节转动速度，具体数值见表 3-4。

表 3-3 关节上下级连杆和原点位置参数表

关节名称	上级连杆	下级连杆	原点位置	
			(X,Y,Z) (m)	(R,P,Y)(rad)
关节 1	基底连杆	肩部连杆	(0,0,0)	(0,0,0)
关节 2	肩部连杆	上臂连杆	(0,0,0.155)	(1.571,-1.571,0)
关节 3	上臂连杆	前臂连杆	(0,0.425,0)	(0,0,-1.571)
关节 4	前臂连杆	手腕连杆 1	(-0.395,0,0)	(0,0,1.571)
关节 5	手腕连杆 1	手腕连杆 2	(0,0,0.13)	(1.571,0,-1.571)
关节 6	手腕连杆 2	手腕连杆 3	(0,0,0.102)	(-1.571,2.269,0)

表 3-4 关节最大角度上下限和最大转矩及最大转动速度

关节名称	最大关节角度下限(rad)	最大关节角度上限(rad)	最大关节转矩(N*m)	最大关节转动速度(rad/s)
关节 1	-3.0543	3.0543	150	3.15
关节 2	-4.6251	1.4835	150	3.15
关节 3	-2.8274	2.8274	150	3.15
关节 4	-4.6251	1.4835	28	3.2
关节 5	-3.0543	3.0543	28	3.2
关节 6	-3.0543	3.0543	28	3.2

在连杆和关节的属性配置完善后，SolidWorks 软件会自动为每一个连杆和关节生成精确的固连坐标系，这些坐标系如图 3-4 所示，清晰地展示了机械臂的结构和连接关系。

紧接着，我们利用 SolidWorks 与 ROS 的集成功能，在预定的路径下快速生成一个包含机械臂所有必要信息的 ROS 功能包。这个包不仅包含了机械臂的三维网格模型，还附带了丰富的纹理贴图以及描述机械臂结构的目录文件。这些文件共同构成了 ROS 系统中机械臂的完整描述。

一旦 ROS 功能包生成完毕，用户只需简单地将其复制到 ROS 的工作空间中，即可在 ROS 环境下直接使用这个包来描述和控制机械臂。这一步骤不仅简化了机械臂在 ROS 系统中的集成流程，还提高了整体的开发效率。

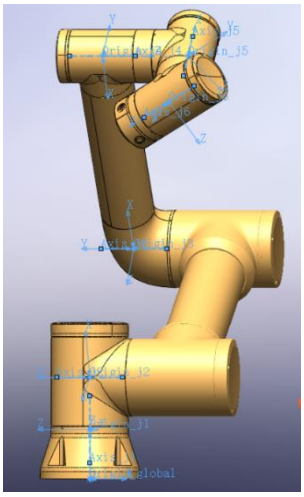


图 3-4 自动生成的固连坐标系图

3.3 URDF 文件解析

由 Solidworks 导出的机械臂描述功能包包含存放机械臂描述文件的 urdf 文件夹和存放机械臂三维模型文件的 meshes 文件夹以及相关的启动文件。

URDF(Unified Robot Description Format)统一机器人描述格式是 ROS 系统中一个非常重要的机器人模型描述格式[29]。URDF 文件按照 XML 语法规则编写，在其首行为声明语句，文件正文首行为 URDF 文件根标签<robot>用来定义机械臂的名称属性，然后声明机械臂所处环境的世界坐标系 world 连杆，再依次声明基底连杆、肩部连杆、上臂连杆、前臂连杆、手腕连杆 1、手腕连杆 2 和手腕连杆 3 七个连杆。每个连杆都需声明视觉可视化标签<visual>用来描述机械臂连杆的外观参数、材料属性标签<inertial>描述连杆的惯性参数以及碰撞标签<collision>用来描述机械臂连杆的碰撞属性。在设置六个关节时，需声明每个关节的名称、类型、位姿、父子连杆、旋转角度限制、最大速度、最大作用力以及阻尼和静摩擦系数等。每个标签的具体功能如表 3-5 所示。由于本课题的核心为仿真，部分具体数值无法测量，故假设所有关节的阻尼和静摩擦系数均为 0。

表 3-5 URDF 标签功能

标签	功能
<robot>	定义机械臂名称
<link>	定义连杆名称
<visual>	描述机械臂连杆的外观参数
<inertial>	描述连杆的惯性参数
<collison>	描述连杆的碰撞属性
<joint>	定义关节名称
<parent link>	设置父连杆
<child link>	设置子连杆
<anis xyz>	设置旋转中心
<origin rpy xyz>	设置关节坐标
<limit>	设置关节限位角度

为了能够在 Rviz 中看到 Solidworks 中所搭建模型，在 URDF 文件中的<geometry>标签中将 dae 模型文件添加到 URDF 模型文件中，使其以网格对象所需的外观在 Rviz 中进行显示。为了检查 URDF 是否设置成功，本文使用了 liburdfdom-tools 中的工具 check_urdf 检测所构建的 URDF 文件是否有 XML 语法错误，其配置成功反馈信息如图 3-5 所示[30]。为了可以直观的看到 URDF 文件所描述的机械臂结构，本文也提供了图形化的机械臂 URDF 结构，其中长方形框表示连杆，椭圆形框表示关节，在连杆和关节之间的是连杆相对于邻近关节的位姿信息。整个模型的树状结构如图 3-6 所示。

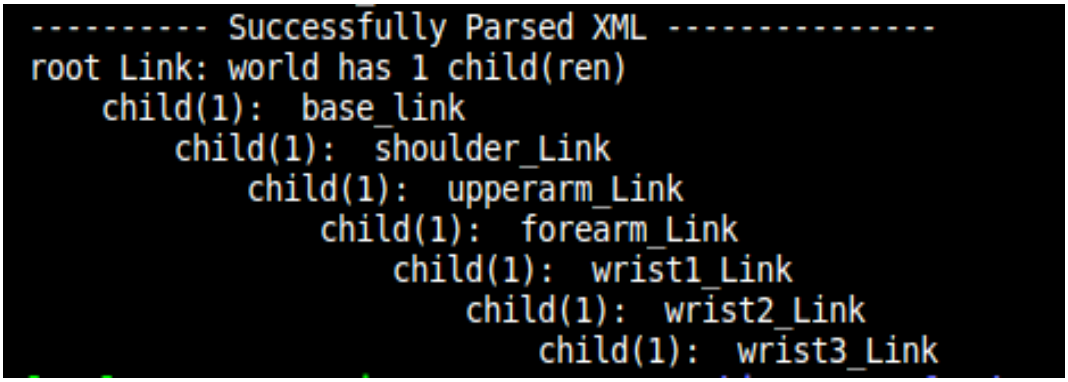


图 3-5 URDF 配置成功图

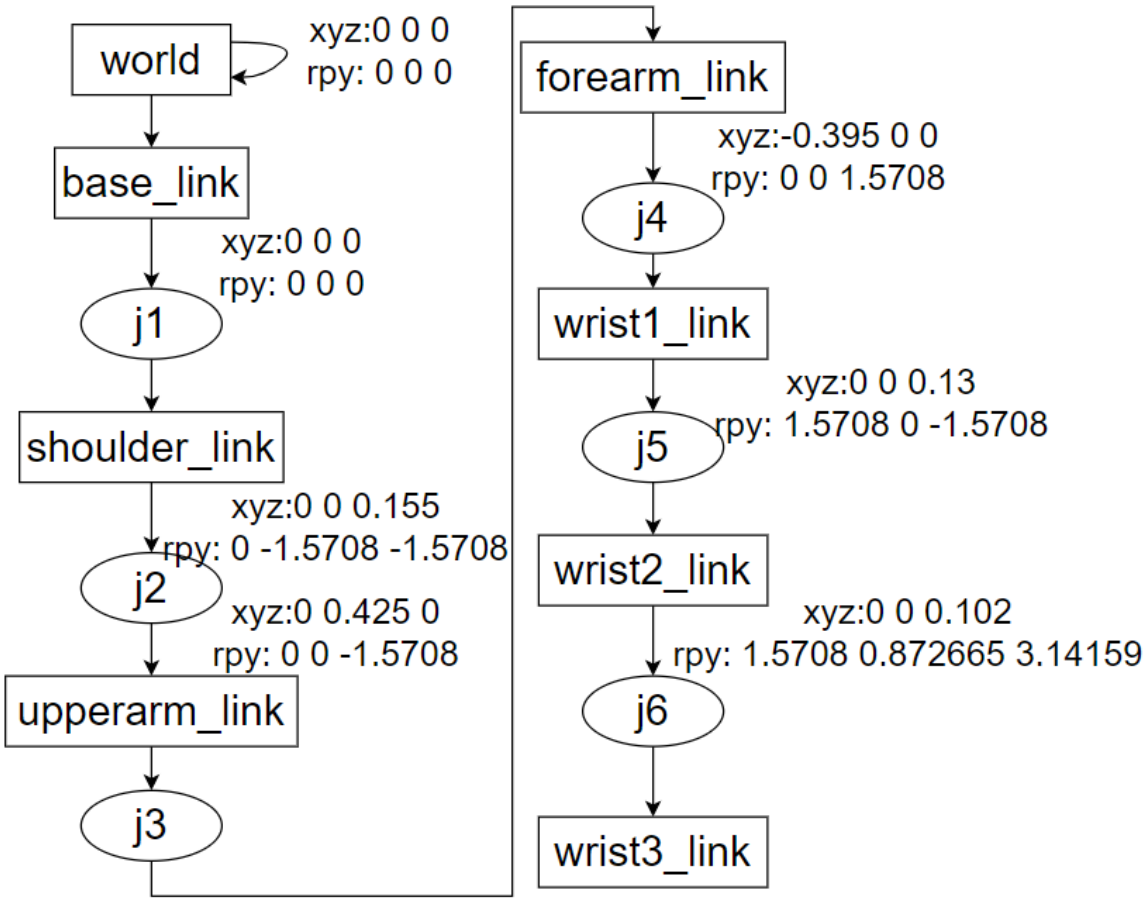


图 3-6 URDF 检测树状结构图

3.4 Rviz 可视化机械臂模型

在 Rviz 中可视化 URDF 模型能够直观地看出模型设计是否符合预期。其启动文件配置包含加载机械臂模型参数，运行关节状态发布节点发布机械臂的关节状态，运行机器人状态节点发布坐标系变换信息以及运行 Rviz 可视化界面。在 Rviz 中通过 Robotmodel 插件，我们可以看到机械臂模型，如图 3-7 所示。

TF 坐标图是 ROS 中管理坐标系变换关系的一种数据结构，它对应于机械臂不同位置固连坐标系之间的变换关系，以显示机器人的运动状态。在 Rviz 中还可以看到该机械臂位姿相应的 TF 坐标图，如图 3-8 所示。图中红轴表示 X 轴，绿轴表示 Y 轴，蓝轴表示 Z 轴。各个坐标系之间由粉色箭头表示子连杆坐标系连杆位置相对于父连杆原点的相对位置。通过与 Solidworks 中坐标系的建立相比较，此 TF 坐标图构建正确。

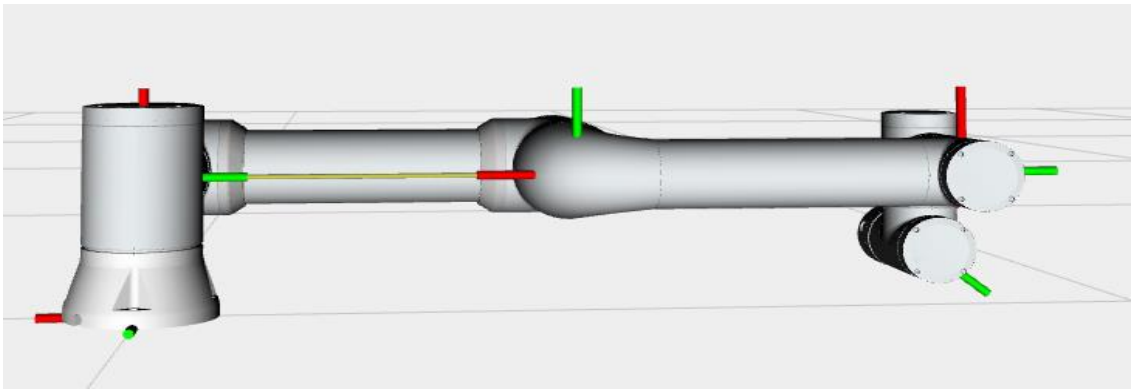


图 3-7 Rviz 中机械臂模型图

图 3-8 TF 坐标图

3.5 本章小结

本章主要完成了六自由度机械臂在 Solidworks 中的建模和装配，完成从 Solidworks 到 URDF 的文件转换，配置了七个连杆和六个有限位的旋转关节相关信息，并对 URDF 进行了解析和验证，最后在 Rviz 中完成该模型的三维可视化和 TF 坐标图的显示。

4 基于 MoveIt 策略的机械臂运动控制研究

MoveIt!集成了运动规划、操作、逆运动学、控制、3D 感知、碰撞检测算法领域的最新成果，为六自由度机械臂提供了易于使用和集成的平台[31]。MoveIt!由一系列 ROS 功能包组成，部分算法模块以插件的形式集成进来，为机械臂的仿真开发提供了便利条件。

4.1MoveIt!核心架构

MoveIt!的系统结构大致可分为四部分，分别为：用户界面、move_group 核心节点、ROS 参数服务器和外部机器人部件。MoveIt!首先需要进行运动学插件、碰撞检测插件、运动规划插件、规划请求适配器插件以及一些高层功能插件组的配置。然后 MoveIt!会接收机器人模型发布节点和关节节点发布的机器人信息，将其保存在 MoveIt!节点中。通过 MoveIt!节点对机器人的分析，会将规划出的控制信息发送给控制器管理插件组，并由其将控制信息发布到机器人执行器。具体的核心架构如图 4-1 所示。

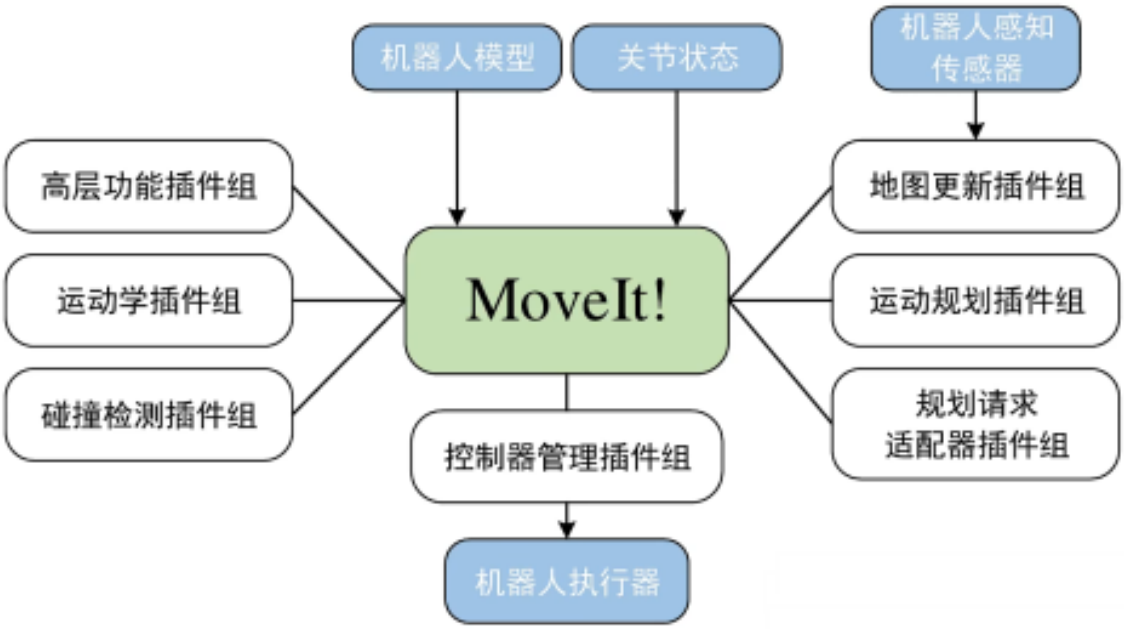


图 4-1 MoveIt!核心架构图

4.1.1move_group 核心节点

move_group 是 MoveIt!的核心节点，虽然该节点本身不具备丰富的功能，但其可以通过完成各功能包以及插件的集成，综合话题、服务、动作方式接收机械臂模型及相关指令数据为用户提供 ROS 中的机械臂轨迹规划动作指令和服务，如图 4-2 所示[31]。

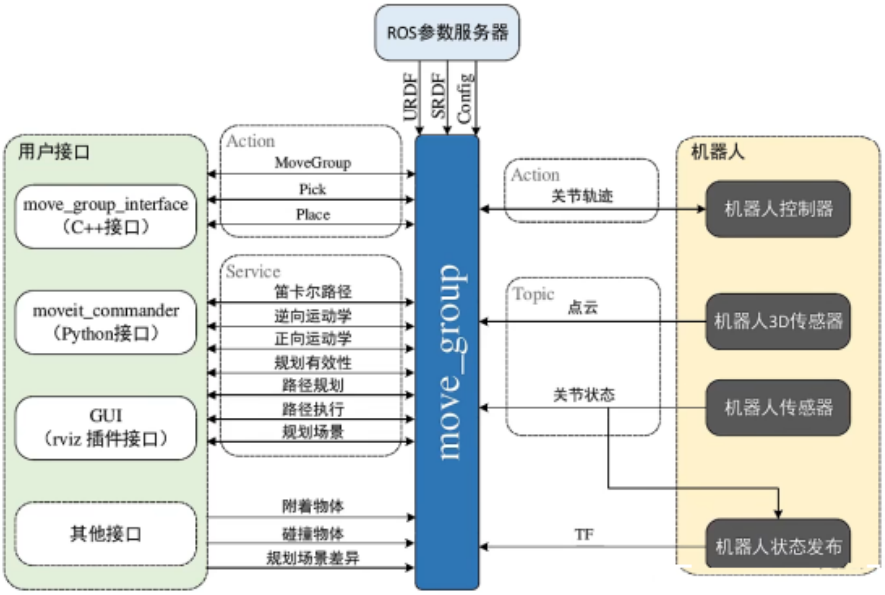


图 4-2 move_group 核心节点图

move_group 作为核心节点，其运作依赖于接收来自机械臂传感器通过消息或服务形式发布的点云数据、关节状态数据以及机器人的 TF 坐标变换信息。同时，它还需从 ROS 参数服务器中检索机械臂的模型文件和配置文件内包含的运动学参数。这些关键参数信息随后被存储在 move_group 节点中，以供后续的运动规划和控制使用。

move_group 核心节点还提供了丰富的可供调用的编程接口，例如 C++和 Python 接口用户可以调用其提供的 API 对机械臂进行编程控制。

4.1.2 运动规划器

由于本课题所研究的六自由度机械臂已知机械臂的初始姿态和目标姿态以及机械臂和周围环境的模型参数，那么可以通过 MoveIt!中内置的运动规划算法，在躲避周围障碍物及工件并防止自身发生碰撞的同时，找到一条到达目标位姿的较优路径。在 MoveIt!完成这些步骤的规划器被称为 MoveIt!的运动规划器。其工作流程如图 4-3 所示。

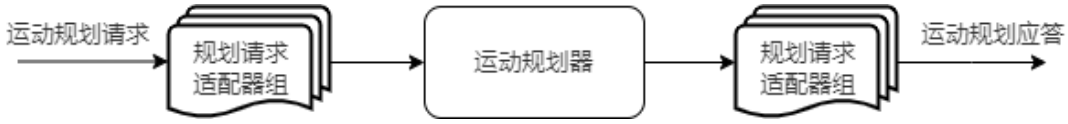


图 4-3 运动规划器结构

在 MoveIt!中，运动规划算法需要通过运动规划器完成。运动规划计算需要根据实际情况设置一些约束条件。位置约束限制连杆的运动区域。方向约束限制连杆的运动方向。可见性约束通过视觉传感器感知限制连杆上的某点在某区域内的可见性。关节约束限制关节的运动范围。用户定义约束是用户通过回调函数自定义所需的约束条件。

规划场景可以为机械臂创建一个类似化工操作台的工作环境，包括外界环境中的桌面、烧杯、搅拌器等物体，并在 Rviz 中将这一场景展现出来，如图 4-4。这一功能的核心实现依赖于 move_group 节点中的规划场景监听器。该监听器专门负责监听和接收与机械臂相关的特定话题信息。具体来说，它会实时监听由 joint_states 话题发布的机械臂关节状态信息，同时还会收集来自机械臂外部传感器的数据以及通过传感器所获取的关于外界环境的详细信息。通过这种方式，move_group 节点能够获取到完整的机械臂状态以及周围环境的信息，为后续的规划和控制提供数据支持。

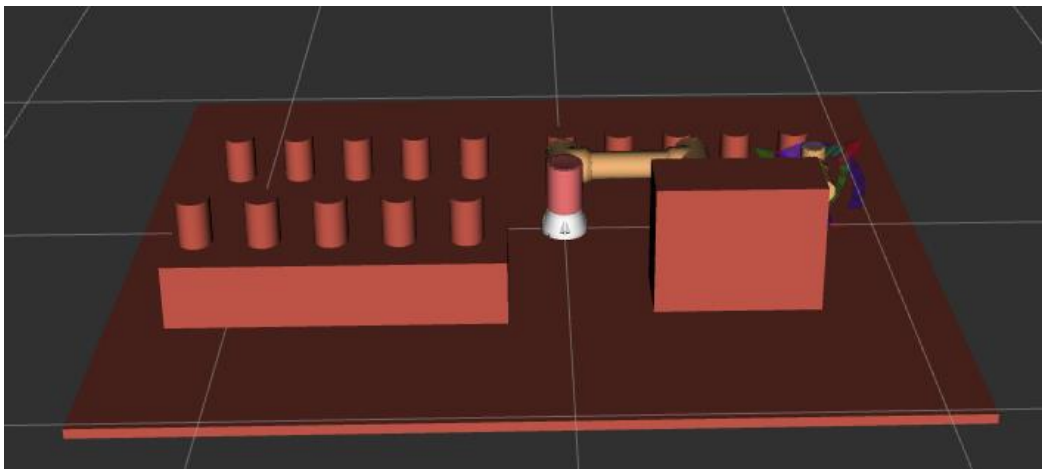


图 4-4 Rviz 场景图

4.1.3 运动学求解器

MoveIt!中的运动学插件允许开发者灵活选择多种可供使用的运动学求解器，可以在 MoveIt! Setup Assistant 工具中进行配置。MoveIt! 采用 FCL 功能包实现机械臂对周围环境的碰撞检测。在机械臂的运动规划过程中，碰撞检测常常是一项计算密集型的任务。为了优化这一过程的效率，我们可以利用 MoveIt! Setup Assistant 工具

中的免检冲突矩阵(ACM)设置来减少不必要的计算量。具体来说，当我们将两个连杆之间的 ACM 系数设定为 1 时，我们实际上是在告诉系统这两个连杆在任何情况下都不会发生碰撞。因此，在进行碰撞检测时，系统将跳过对这两个连杆的碰撞检测，从而显著提高整体的运动规划效率。

4.2 MoveIt!可视化配置

本文所采用的 ROS 系统为 Ubuntu20.04 所适配的 Noetic 版本，其中 MoveIt!可视化配置由 Moveit! Setup Assistant 辅助完成。其具体流程为加载机械臂 URDF 文件，配置免检碰撞矩阵 ACM，配置虚拟关节，创建规划组并定义机械臂姿态，配置末端执行器，添加被动关节，设置外部传感器参数，完成 ROS Control 配置，添加作者信息，最后生成配置文件。其具体流程如图 4-5 所示。

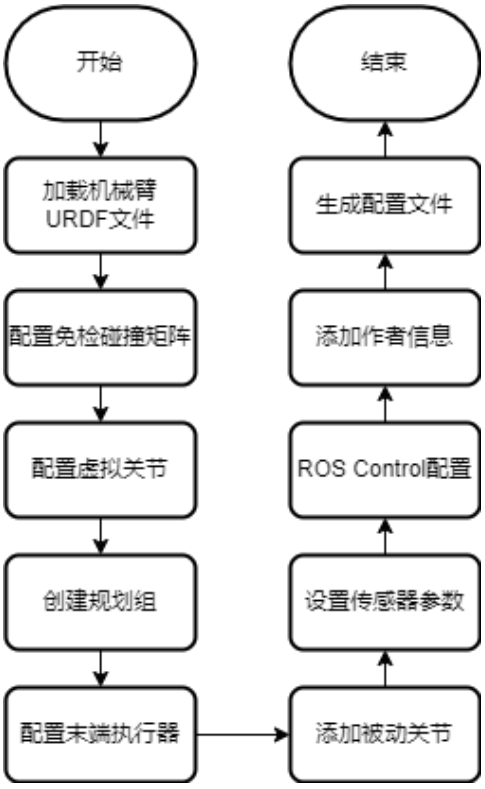


图 4-5 MoveIt!可视化配置流程图

在 Moveit! Setup Assistant 配置的模型文件采用本文第三章从 SolidWorks 转化来的 URDF 文件作为机械臂的模型文件。

在 Moveit! Setup Assistant 配置免检碰撞矩阵是为了减轻计算机检验碰撞的计算量。本文所采用的 ACM 免检冲突矩阵如表 4-1 所示。表中“√”表示这两个连杆需

要在运动规划时检测这两个连杆是否会发生碰撞，“×”则表示与之对应的两个连杆永远不会发生碰撞。

表 4-1 ACM 免检冲突矩阵

	基底连杆	肩部连杆	上臂连杆	前臂连杆	手腕连杆 1	手腕连杆 2	手腕连杆 3
基底连杆	×	√	√	×	×	×	×
肩部连杆	√	×	×	×	×	×	×
上臂连杆	√	√	×	√	×	×	×
前臂连杆	×	×	√	×	√	√	×
手腕连杆 1	×	×	×	√	×	√	√
手腕连杆 2	×	×	×	√	√	×	√
手腕连杆 3	×	×	×	×	√	√	×

在 Moveit! Setup Assistant 配置的运动规划组为六个机械臂关节，即 j1 到 j6 六个关节。

4.3 MoveIt!控制机械臂运动

在 Rviz 界面使用 moveItplanning 插件对六自由度机械臂进行碰撞检测，当机械臂检测到机械臂本体在规划的路径上执行运动时，将会与周围环境的物体发生碰撞时，将会以标红的方式在 Rviz 界面显示机械臂上与周围环境物体将会发生碰撞的连杆，如图 4-6 所示，并在状态显示栏显示发生碰撞的连杆和关节信息。除此之外，当机械臂检测到机械臂以 OMPL 规划库所规划的路径执行运动时，机械臂某连杆或关节无法完成所规划的路径点时，同样将会以标红的方式显示无法完成的连杆或关节，并在状态显示栏显示发生冲突的连杆和关节信息。

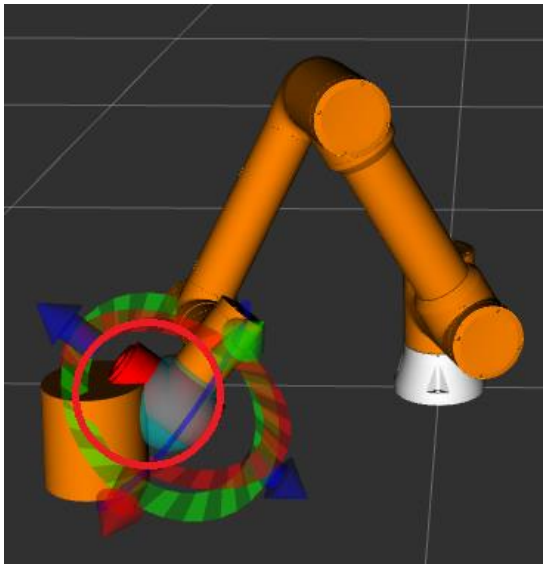


图 4-6 碰撞检测图

在 moveIt 配置时，在配置界面预设机械臂的五个位置分别为 home、pose1、pose2、pose3、pose4，其关节空间的各角度具体数值见表 4-2。

表 4-2 五种位姿关节角度配置表(单位：rad)

位姿名称	j1	j2	j3	j4	j5	j6
home	0	0	0	0	0	0
pose1	0	-1.4864	1.609	-0.1365	1.6031	0
pose2	0.6919	-1.2502	1.2965	0.1673	0.7256	0.6244
pose3	-0.4219	-1.6552	1.7027	-0.0015	2.2046	-0.1181
pose4	0.1519	-1.4189	1.4215	-0.7102	1.6368	0.3206

在 moveItplanning 界面使用 moveIt!的 python 编程接口方式，依次规划出从 home 到 pose1 的路径规划(如图 4-8a)、从 pose1 到 pose2 的路径规划(如图 4-8b)、从 pose2 到 pose3 的路径规划(如图 4-8c)、从 pose3 到 pose4 的路径规划(如图 4-8d)。其代码流程为图 4-7，首先初始化 move_group 的 API、初始化 ROS 节点、初始化需要使用的控制组、获取终端 link 的名称，设置目标位置所使用的参考坐标系、设置位置和姿态的允许误差范围、设置机械臂终端运动的目标位姿 pose1 到 pose4，然后规划运动路径、控制机械臂完成运动，最后关闭并退出 moveIt!。在 Rviz 界面实时显示出四次运动执行的可行性分析数值、最大负荷载重、饱和关节数量、六个关节的扭转力矩数值。MoveIt 可行性指数越小可执行性越强。由此可知，四次路径规划可以完成，所以所设置的机械臂五个位姿机械臂可以完成，并有能够依次完成路径执行

的关节路径。这一可行性为后续第 5 章 RRT 算法和 RRTStar 算法完成机械臂路径规划打下了可行性基础。

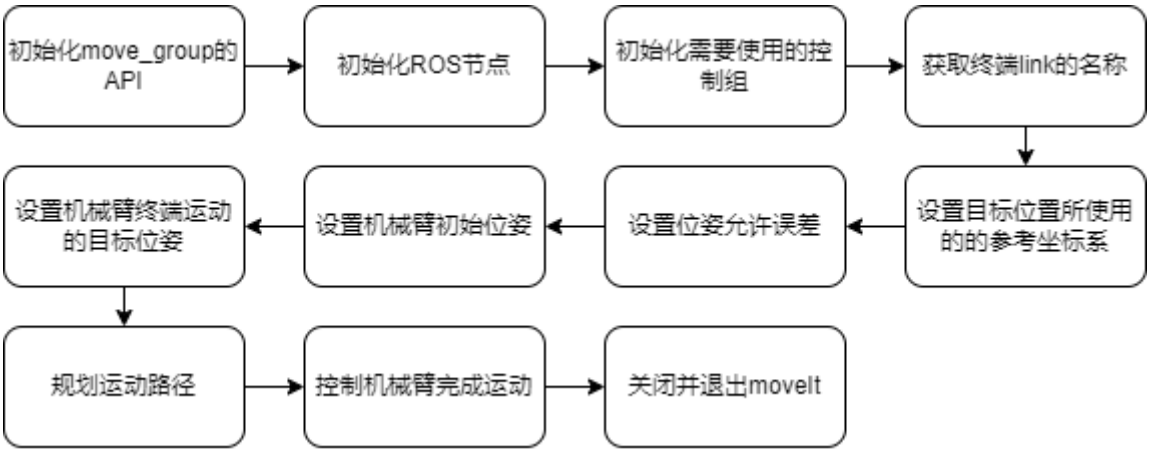
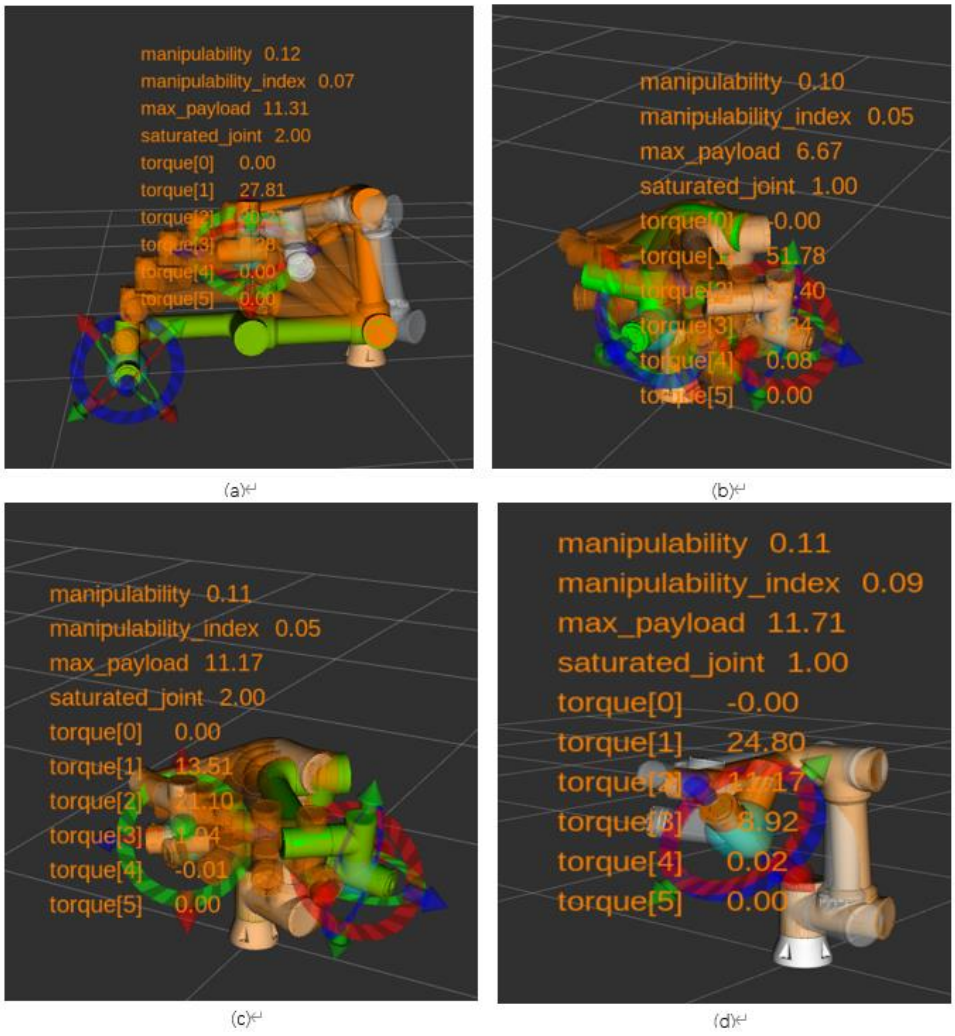


图 4-7 moveIt 编程流程图



(a) (b)

(c) (d)

图 4-8 机械臂四次规划执行图

4.4 通讯计算图

ROS 的通讯计算图是一个由 ROS 进程组成的点对点网络，包含节点、节点管理器、参数服务器、消息、话题、服务和包等，它们以不同的通讯方式为 ROS 提供数据，实现端对端的网络连接[32]。本文完成所有仿真配置后所形成的一级通讯计算图如图 4-9 所示。

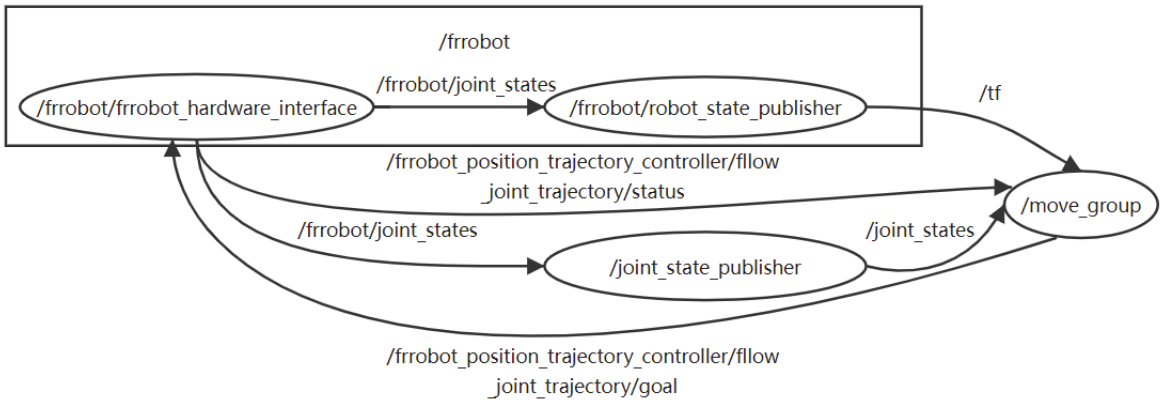


图 4-9 一级通讯计算图

在通信架构图中，椭圆形被用来标识节点，这些节点代表执行特定任务的进程或独立运行的可执行文件。而方框则用来表示话题、服务以及动作通信，它们之间流动的则是消息。节点之间通过 ROS（机器人操作系统）客户端库实现相互通信。

话题通信采用了发布/订阅的异步通信模式来传输数据。具体来说，一个节点可以在特定主题上发布消息，而另一个或多个节点可以关注并订阅该主题以接收特定类型的数据。值得注意的是，发布者和订阅者之间并不需要了解彼此的存在，它们仅仅是通过主题进行数据的传递。

与话题通信不同，服务通信则是一种同步通信机制。它基于客户端/服务器模型，客户端发送请求数据给服务器，服务器在完成相应的处理后返回应答数据给客户端。这种通信方式确保了请求和响应之间的直接对应关系。

动作通信是 ROS 消息机制中的一种交互模式，它采用问答式的通信方式，并遵循经典的客户端/服务器架构。在这种通信模式下，服务器具备连续向客户端反馈数据的能力，确保数据流的实时性和连续性。同时，客户端也拥有在任务执行过程中随时中断任务运行的权限，这种中断机制赋予了动作通信极大的灵活性和丰富的交互体验。通过动作通信，ROS 系统内的不同组件可以实现更为高效、灵活的信息交换和任务协同。

图 4-10 为本文所构建的二级节点通讯计算图。在这个节点通讯计算图中，主要展现了 move_group 核心节点在整个 ROS 仿真过程中的核心地位。机械臂的坐标变换、坐标的状态信息、机械臂正逆运动学的解算、笛卡尔路径的规划及执行、场景规划、碰撞检测以及机械臂的拾取和放置动作都会将自己的数据信息传递给 move_group 核心节点进行处理并再由 move_group 核心节点将执行信息传送给对应的执行器进行执行。

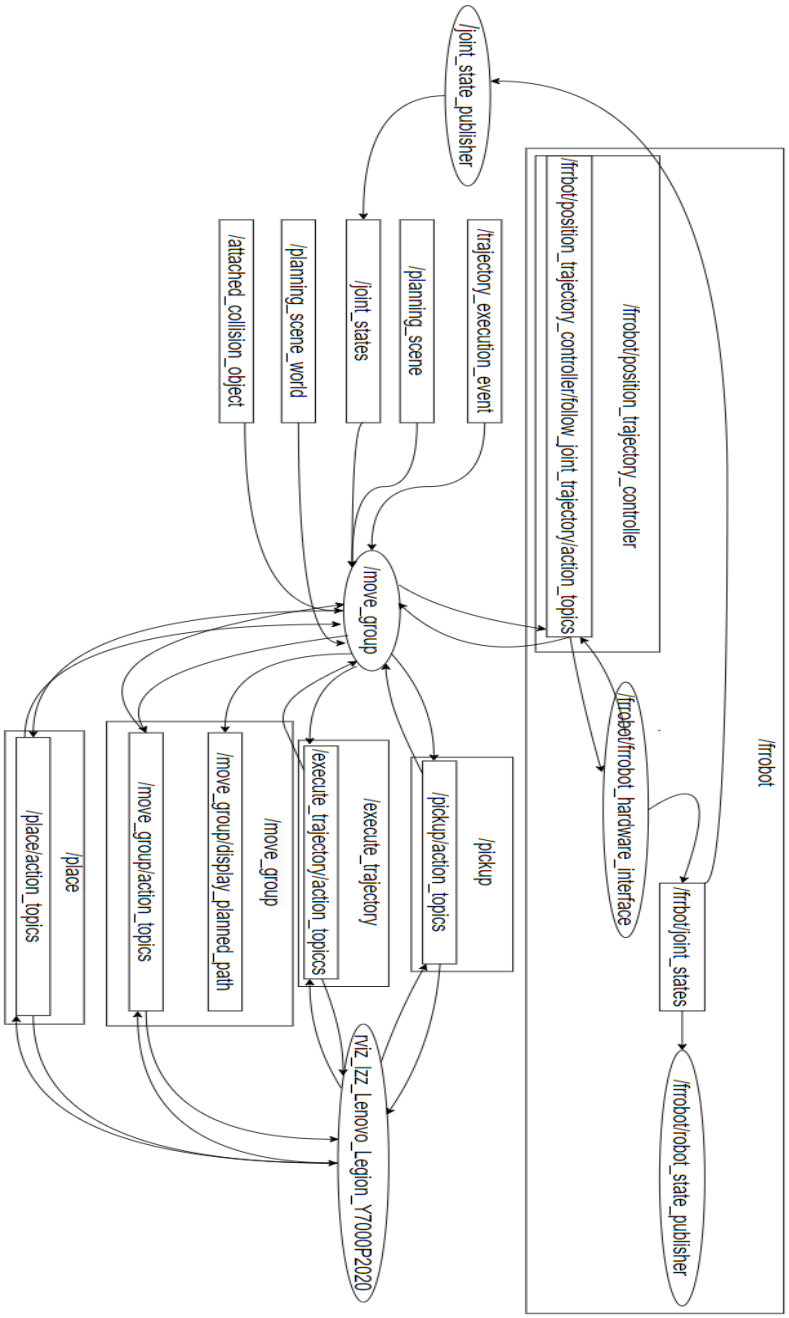


图 4-10 move_group 核心节点通讯计算图

4.5 本章小结

本章主要完成了 MoveIt!核心架构的配置工作，对其中的运动学规划器、运动学求解器、可视化配置过程作了详细阐述并展示了通讯节点的计算图。在配置过程中对机械臂做了五种位姿的运动规划，并对四次规划的结果进行了说明。最后，通过对 ROS 通讯计算图第一级和第二级的通讯进行了说明，体现了 MoveIt!的核心通讯架构。

5 基于 OMPL 的运动规划算法与实现

OMPL 是一个 MoveIt!中基于采样方法的开源机器人运动规划库，在其内部内置了一些基于 RRT 和 RPM 衍生出来的运动规划算法[33]。本章将应用 OMPL 运动规划库中的 RRT 算法和 RRTStar 算法完成对六自由度机械臂的 ROS 空间轨迹规划，并将其在 Rviz 可视化空间中展现出来。本章还将利用 rosbag 完成关键信息的记录，对其规划的轨迹做关节空间位置和速度变化进行分析。

5.1 RRT 算法理论及工具

RRT 算法，全称为快速探索随机树，是一种在路径规划和运动规划领域广泛应用的算法，特别是在自动驾驶、机器人导航等场景下[34]。该算法核心在于利用随机采样技术构建一个能够快速覆盖配置空间的随机树结构，从而找到从初始状态到目标状态的可行路径。RRT 算法示意图如图 5-1 所示。其算法伪代码如表 5-1 所示。

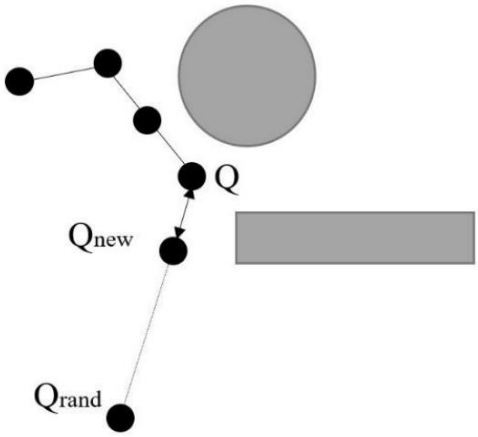


图 5-1 RRT 算法示意图

表 5-1 RRT 算法伪代码

算法 1 RRT 算法

Input:

Initial configuration q_{start},q_{goal},Map,Max number of sampling point N,Step size σ

Output:

RRT graph G=(V,E)

Vertices of tree nodes

1: **Initialize All Parameters**

2: $V=q_{start}$

3: **for** i=0 to N **do**

4: Sample() to q_{rand};

5: Nearest(V,q_{rand}) to $q_{nearest}$;

6: Steer($q_{nearest}$,q_{new}, σ) to qnew;

7: **if** CollisionFree($q_{nearest}$,q_{new}) **then**

8: V＆{q_{new}}to V

9: E＆{($q_{nearest}$,q_{new})} to E

10: **if** *InitialPathFound* **then**

11: **return** G=(V,E)

12: **end if**

13: **end if**

14: **end for**

15: **return** G=(V,E)

RRT 算法步骤如下：

(1)初始化：算法开始时，从起始点创建一个单一的节点作为随机树的根节点。

(2)随机采样：算法在每一步中都会随机选取一个状态空间中的点，这个点通常是在整个可行区域或者配置空间中随机产生的。

(3)接近性判断与树扩展：计算新采样点与树中最近节点的距离，并检查从最近节点沿直线向采样点方向延伸是否遇到障碍物。如果没有障碍阻挡，就在该方向上添加一个新节点至采样点的位置，这个新节点成为树的一部分。

(4)目标检测与终止条件：如果新添加的节点或采样点落在目标区域，或者与目标点足够接近预设的阈值，则认为找到了一条可行路径，并停止扩展。此时可以通过回溯随机树从起始节点到目标节点来获取路径。

(5)重复采样与扩展：如果上述步骤未达到终止条件，则重复随机采样、接近性判断与树扩展和目标检测与终止条件这三个步骤，直至找到路径或满足其他停止条件，如达到最大迭代次数、计算资源限制等。

RRT 算法具有概率完备性，即只要存在可行路径，随着迭代次数的增加，算法总能以概率 1 找到这条路径。也就是说，RRT 算法总能为机械臂找到一条可行的工

作路径。然而，RRT 算法生成的路径通常不是最优的，且可能包含很多不必要的转折。RRT 算法特别适用于难以用传统方法处理的复杂环境，能够高效地处理障碍物多且分布不规则的情况。

5.2 RRT 算法实现效果

六自由度机械臂 RRT 算法的实现效果由附录中的 joint_motion.py 实现。此代码初始化了 move_group 的 API、ROS 节点、arm_group 运动群，设置了机械臂运动的允许误差值为 0.001rad 以及最大速度和最大加速度分别为 0.5rad/s 和 0.5rad/s^2。在 joint_motion.py 设置机械臂依次从 home 到预设的 pose1、pose2、pose3、pose4 最后回到 home 位置。用 rosbag 完成对机械臂六个关节的位置和速度记录，使用 rqt_plot 插件完成六自由度机械臂位置和速度随时间的变化绘制。

从图 5-2 和图 5-3 可知使用 RRT 算法控制六自由度机械臂时会出现较大的瞬时位置变化和较大的刚性冲击，如从 home 到 pose1 这一时间段角度和速度均随时间发生较大振荡。使用 RRT 算法对此六自由度机械臂进行路径规划，机械臂的关节发生较大的瞬时角度和速度变化，这会极大的削弱电机的使用寿命。

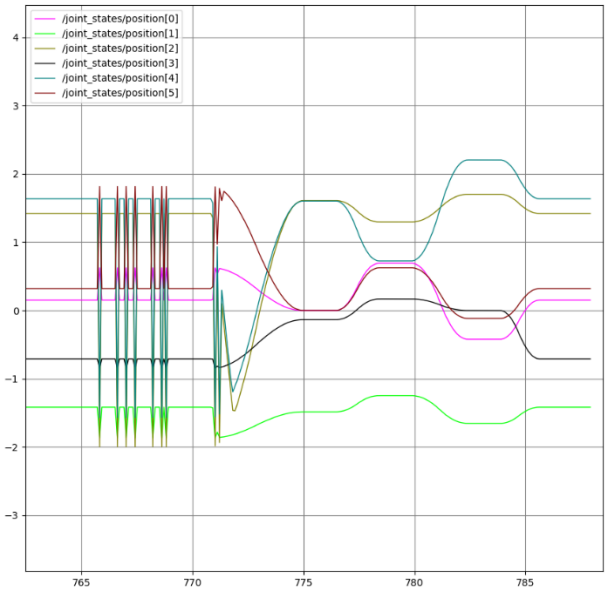


图 5-2 RRT 算法关节位置变化图

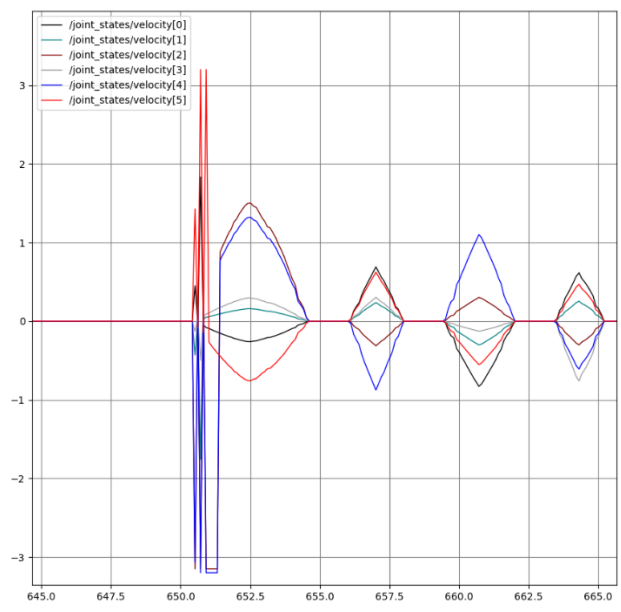


图 5-3 RRT 算法关节速度变化图

5.3 RRTStar 算法理论及工具

RRTStar 算法是 RRT 算法的一个改进版本，旨在提高路径规划的质量和效率。RRTStar 算法继承了 RRT 算法的基本框架，即通过在空间中随机采样和构建树来探索可行路径[35]。然而，与 RRT 相比，RRTStar 算法引入了一些关键的改进，使得它在路径优化和避免局部最优解方面更具优势。

RRTStar 算法在构建树的过程中，不仅关注如何快速找到一条从起点到终点的路径，还致力于优化这条路径。它采用了重连策略，即在添加新节点后，会检查是否存在更优的连接方式，如果有，则重新连接树中的节点，以形成更平滑、更短的路径。这种优化过程随采样点的增加而持续进行，使得最终得到的路径质量更高。RRTStar 算法示意图如图 5-4 所示。其算法伪代码如表 5-1 所示。

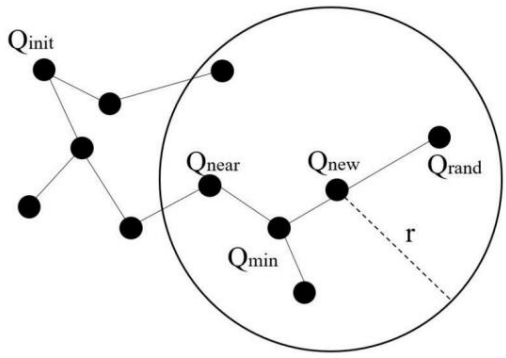


图 5-4 RRTStar 算法示意图

表 5-2 RRTStar 算法伪代码

算法 2 RRTStar 算法

Input:

Initial configuration q_{start},q_{goal},Map,R_{near},Max number of sampling point N,Step size σ

Output:

RRT graph G=(V,E)

Vertices of tree nodes

1: **Initialize All Parameters**

2: V=q_{start}

3: **for** i=0 to N **do**

4: Sample() to q_{rand};

5: Nearest(V,q_{rand}) to $q_{nearest}$;

6: Steer($q_{nearest}$,q_{new}, σ) to qnew;

7: **if** CollisionFree($q_{nearest}$,q_{new}) **then**

8: Near(V, q_{new}, R_{near}) to Q_{nears}

9: ChooseParent(Q_{nears}, q_{new}, $q_{nearest}$) to q_{parent};

10: V＆{q_{new}}to V;

11: E＆{(q_{parent},q_{new})} to E

12: **if** *NewPathIsBetter* **then**

13: GraphUpdate;

14: **end if**

15: Rewire(G,q_{new},Q_{nears}) to G;

16: **end if**

17: **end for**

18: **return** G=(V,E)

RRT 算法有时可能陷入局部最优解，即找到一条看似不错的路径但实际上并不是全局最优的。RRTStar 通过引入启发式函数和重新布线策略来避免这个问题。启发式函数用于估计当前节点与目标点之间的距离，引导树的扩展方向，使其更有可能找到全局最优解。重新布线策略则通过调整树的结构，使得路径更加平滑，并减少不必要的转折。

RRTStar 算法对不同类型的环境和问题具有较好的适应性和鲁棒性。无论是障碍物分布密集还是稀疏的环境， RRTStar 都能有效地进行路径规划。此外，由于它是基于随机采样的方法，因此不依赖于特定的地图表示或环境模型，使得它在实际应用中更加灵活和方便。

5.4 RRTStar 算法优化效果

六自由度机械臂 RRTStar 算法的实现效果同样由附录中的 joint_motion.py 实现。在 joint_motion.py 设置机械臂依次从 home 到预设的 pose1、pose2、pose3、pose4 最后回到 home 位置。五种位姿仍使用表 5-1 所示的角度参数。用 rosbag 完成对机械臂六个关节的位置和速度记录，使用 rqt_plot 插件完成六自由度机械臂位置和速度随时间的变化绘制。

由图 5-5 和图 5-6 可知，在使用 RRTStar 算法控制六自由度机械臂时，六个关节的角速度近似呈现匀加速-匀减速的变化趋势，符合机械臂电机速度曲线规划要求。四次位置运动过程整体平稳可靠，轨迹执行全程角度和速度随时间没有发生较大振荡，机械臂的关节运动过程较为平滑，符合工业运动控制的要求。由此可知 RRTStar 更为适合对此六自由度机械臂进行路径规划。

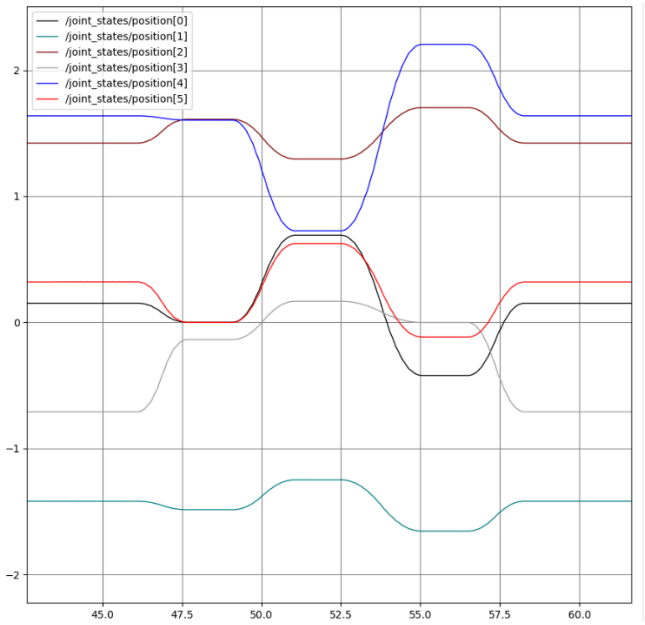


图 5-5 RRTStar 算法关节位置变化图

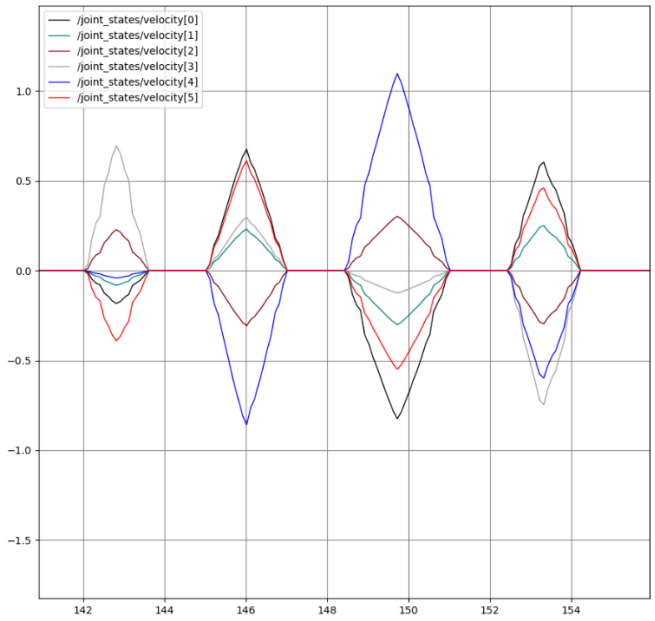


图 5-6 RRTStar **算法关节速度变化图**

虽然 RRTStar 算法在路径优化和避免局部最优解方面表现出色，但它仍然存在一定的局限性。例如，在复杂环境中，RRTStar 可能需要较长的计算时间来找到一条满意的路径。

经过 50 次实测 RRTStar 算法完成四次路径规划和执行平均耗时 21.369s，而 RRT 算法平均耗时仅为 16.928s。此外，由于算法基于随机采样，因此生成的路径可能不是严格的最优解，而是在可接受范围内的次优解。在实际应用中，需要根据具体需求和环境特点来选择合适的算法参数和策略。

5.5 RRTStar **避障轨迹规划仿真**

在对比 RRT 算法和 RRTStar 算法在六自由度轨迹规划方面的实现效果后，本文认为 RRTStar 算法对本六自由度机械臂的轨迹规划效果更好。为了进一步研究 RRTStar 算法的避障能力，本文通过对 RRTStar 所规划的轨迹进行三次插值优化，完成了在 Rviz 中六自由度机械臂避障仿真实验。

图 5-7 至图 5-9 展现了 RRTStar 轨迹规划算法避障仿真过程。仿真环境中搭建了长 2 米，宽 1.5 米，高 0.15 米的实验台。在机械臂两侧分别放置了长 1 米，宽 0.3 米，高 0.1 米的置物台，每个置物台上均匀摆放了 4 个直径为 0.1 米，高为 0.2 米的圆柱体，用来模拟被抓取的物体。在置物台的后方摆放了一个长 0.9 米，宽 0.25 米，高 0.5 米的长方体，用来模拟控制箱。

机械臂的抓取过程为从 home 位姿出发，到达第一排第一个圆柱体抓取位姿处(如图 5-7)，再从此位置出发到达第二排第四个圆柱体放置位姿处(如图 5-8)，最后返回到 home 位姿(如图 5-9)。从 home 位姿到抓取位姿执行过程耗时约 6.8 秒，从抓取位姿到放置位姿执行过程耗时约 3.9 秒，从放置位姿到 home 位姿执行过程耗时约 4.2 秒,通过对轨迹执行过程中以每秒 1 次的发布的残影分析可知，机械臂使用 RRTStar 轨迹规划算法所执行的轨迹过程中，没有与周围环境中的物体发生碰撞，成功避开所有障碍物。

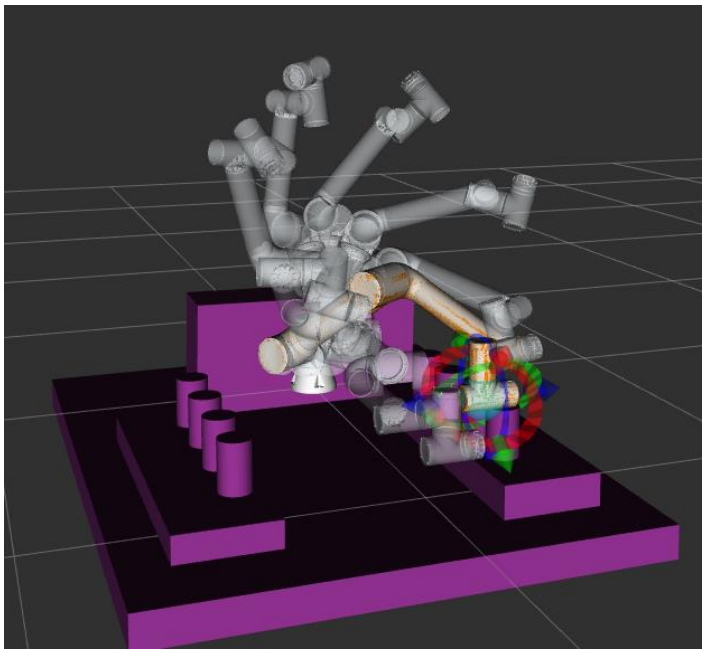


图 5-7 从 home 位姿到抓取位姿执行过程图

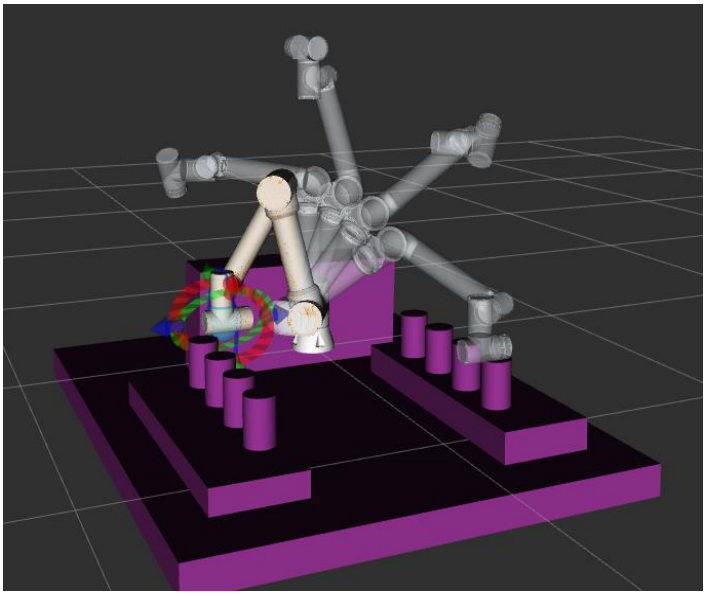


图 5-8 从抓取位姿到放置位姿执行过程图

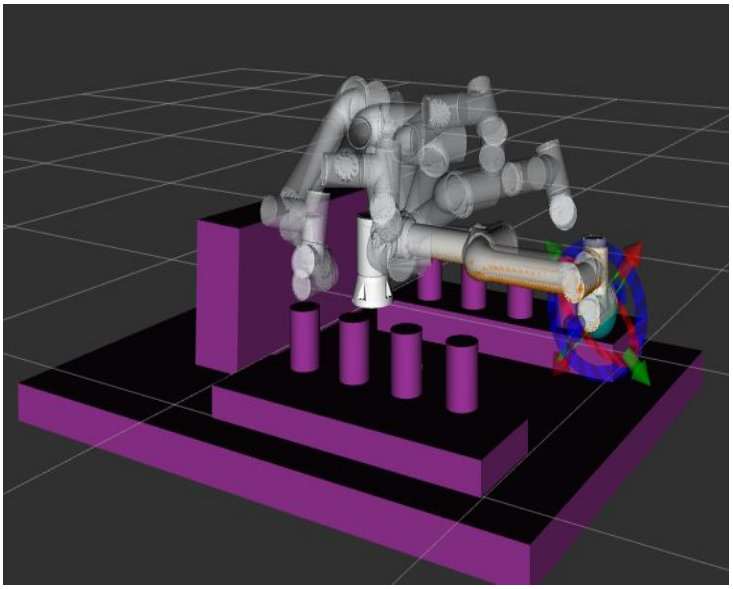


图 5-9 从放置位姿到 home 位姿执行过程图

5.6 本章小结

本章应用 OMPL 运动规划库中的 RRT 算法和 RRTStar 算法完成对六自由度机械臂的轨迹规划。通过对其规划的轨迹做关节空间角度和速度随时间的变化分析，得出使用 RRTStar 算法对此六自由度机械臂进行轨迹规划更有利于延长机械臂电机的使用寿命的结论。通过对 RRTStar 算法所规划的执行轨迹残影分析可知，RRTStar 算法适用于六自由度机械臂避障轨迹的规划。

总结

六自由度机械臂是当前机器人领域使用范围最广，使用数量最多的工业级机器人，但对其仿真研究的文献并不多。本研究借助高通量催化剂配方选型化工操作台上所使用的六自由度机械臂，完成了从理论分析到模型建立再到 ROS 系统仿真实验一系列工作。以下为本次毕业设计完成的主要工作：

(1) 查阅相关文献，了解国内外机械臂仿真研究现状

(2) 完成六自由度机械臂的理论分析，包括对该机械臂 D-H 参数表的建立，机械臂的正逆运动学分析和该机械臂的拉格朗日动力学模型建立，为本次毕业设计中的机械臂仿真打下坚实的理论基础。

(3) 在 Solidworks 中完成了六自由度机械臂的建模和装配，完成从 Solidworks 到 URDF 的文件转换，配置了七个连杆和六个有限位的旋转关节相关信息，并对 URDF 文件进行了解析，最后在 Rviz 中完成该模型的三维可视化。

(4) 在 ROS 系统中完成了 MoveIt!核心架构的配置工作，对其中的运动学规划器、运动学求解器、可视化配置过程作了详细阐述并展示了通讯节点的计算图。在配置过程中对机械臂做了五种位姿的运动规划，并对四次规划的结果进行了说明。

(5) 应用 OMPL 运动规划库中的 RRT 算法和 RRTStar 算法完成对六自由度机械臂的轨迹规划。通过对其规划的轨迹做关节空间角度和速度随时间的变化分析，分析说明使用 RRTStar 算法完成本机械臂的轨迹规划和避障更加高效合理。

参考文献

[1] 《“十四五”机器人产业发展规划》解读[J].自动化博览,2022,39(03):14-15.

[2] 本刊编辑部.十七部门联手部署 助力智能建造与新型建筑工业化协同发展 《“机器人+”应用行动实施方案》印发 强调要加快机器人应用标准研制与推广[J].工程建设标准化,2023(03):10.

[3] 杨凡. 基于 ROS 的室内移动机器人定位与导航研究与实践[D].西南大学,2024.

[4] 胡坤三. 基于 ROS 的移动机器人路径规划研究[D].重庆理工大学,2023.

[5] 李耀麟,张吕彦.虚拟实验的研究现状及其发展前景[J].陇东学院学报,2009,20(02):118-121.

[6] 马琳娜. 移动无人生物安全实验室机械臂运动规划和在线避障研究[D].浙江大学,2017.

[7] 李廷睿. 月面场地集成仿真环境搭建及漫游机器人路径规划[D].哈尔滨工业大学,2021.

[8] W. Fuan and C. Gang,Selection of Terminal Device for Virtual Simulation Experiment,2020 5th International STEM Education Conference (iSTEM-Ed), Hua Hin, Thailand, 2020, pp. 38-41.

[9] H. Lu, J. He and C. Cen,3D Virtual Sand Table Display System of Industrial Park Based on Augmented Reality,2021 IEEE International Conference on Industrial Application of Artificial Intelligence (IAAI), Harbin, China, 2021, pp. 410-415.

[10] N. Toulgaridis, E. Bougioukou and T. Antonakopoulos,Real-Time Emulation of Multiple NAND Flash Channels by Exploiting the DRAM Memory of High-end Servers,2018 21st Euromicro Conference on Digital System Design (DSD), Prague, Czech Republic, 2018, pp. 9-15.

[11] 房川琳,李俊玲,熊庆.化学虚拟仿真实验教学中心的建设与发展[J].实验室研究与探索,2021,40(11):155-159.

[12] 石莹,宋世杰.虚拟仿真实验在高校化学课堂教学中的应用研究[J].科技视界,2021(25):56-57.

[13] 周贵乐,吴晓红.NB 虚拟实验在中学化学实验教学中的应用——以“探究化学反应速率的影响因素”为例[J].中国教育技术装备,2021(19):123-125.

[14] Y. Kim, S. -y. Lee and S. Lim,Implementation of PLC controller connected Gazebo-ROS to support IEC 61131-3,2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 2020, pp. 1195-1198.

[15] J. Park, S. Jang and Y. Shin,Indoor Path Planning for an Unmanned Aerial Vehicle via Curriculum Learning,2021 21st International Conference on Control, Automation and Systems

(ICCAS), Jeju, Korea, Republic of, 2021, pp. 529-533.

[16] 王磊. 基于数字孪生的机械臂状态监测技术研究[D].齐鲁工业大学,2024.

[17] 吕继增. 基于数字图像的碑文文字提取及三维展示[D].西安电子科技大学,2013.

[18] 张勇. 基于 Simulink 的机器人虚拟现实仿真研究[D].哈尔滨工程大学,2007.

[19] 刘鹏. 基于虚拟现实技术的上肢康复机器人控制与仿真研究[D].东北大学,2014.

[20] M. Moradi, S. Dang, Z. Alsalem, J. Desai and A. Palacios,Integrating Human Hand Gestures with Vision Based Feedback Controller to Navigate a Virtual Robotic Arm,2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR), Budapest, Hungary, 2020, pp. 1-6.

[21]. D. Watanabe, M. Al-Sada, K. Fuchino and T. Nakajima,A Robot Arm-based Haptic Feedback System for Augmented Reality Applications,2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Niigata, Japan, 2023, pp. 271-272.

[22] 潘振钊. 基于改进 P-RRT*算法的机械臂运动规划和控制算法研究[D].中北大学,2024.

[23] H. Yang et al.,Obstacle Avoidance Algorithm for Robot Based on the Transformation of Master and Slave Tasks,2022 IEEE International Conference on Robotics and Biomimetics (ROBIO), Jinghong, China, 2022, pp. 815-820.

[24] 李海. 六自由度机械臂运动学及路径规划研究[D].合肥工业大学,2022.

[25] 刘本德. 基于 D-H 参数精确标定的工业机器人关节刚度辨识[D].天津大学,2017.

[26] 苏杰. 六自由度工业机器人运动规划的研究[D].杭州电子科技大学,2024.

[27] 马慧丽. 六自由度机械臂的避障路径规划研究[D].合肥工业大学,2021.

[28] 闵熔祺,张舒.多移动机械臂系统动力学建模以及稳定性分析[J].动力学与控制学报,2023,21(12):107-113.

[29] 王柄森.基于 ROS 六轴机械臂运动规划研究[J].电脑编程技巧与维护,2023(03):140-142.

[30] 程聪.六自由度机械臂半实物仿真系统研究[D].北京理工大学,2016.

[31] 王红杰. 基于 ROS 的仿人机器人控制系统研究[D].北京化工大学,2023.

[32] 靳梦凡. 复杂环境下智能小车视野增强技术的研究与实现[D].北京邮电大学,2024.

[33] 邓阔. 六自由度机械臂的避障最优运动规划研究[D].山东大学,2024.

[34] 彭君. 改进 RRT 算法在移动机器人路径规划中的应用研究[D].南京邮电大学,2023.

[35] 徐亮. 基于改进 RRT~*和 Q-learning 算法的机器人路径规划研究[D].吉林大学,2024.

附录 1 核心代码

1. 理论分析及工作空间代码

```
%六轴机械臂仿真

clear;
close all;
clc;
%六轴机械臂建模
clear L;
angle=pi/180;%角度转化

L(1)=Link('revolute','d',152,'a',0,'alpha',pi/2,'modified');
L(2)=Link('revolute','d',0,'a',-425,'alpha',0,'modified');
L(3)=Link('revolute','d',0,'a',-395,'alpha',0,'modified');
L(4)=Link('revolute','d',102,'a',0,'alpha',pi/2,'modified');
L(5)=Link('revolute','d',102,'a',0,'alpha',-pi/2,'modified');
L(6)=Link('revolute','d',100,'a',0.05,'alpha',0,'modified');

%关节限制范围
L(1).qlim=[-179*angle,179*angle];
L(2).qlim=[-269*angle,89*angle];
L(3).qlim=[-162*angle,162*angle];
L(4).qlim=[-269*angle,89*angle];
L(5).qlim=[-179*angle,179*angle];
L(6).qlim=[-179*angle,179*angle];
%显示机械臂
Six_dof_mod=SerialLink(L,'name','6-dof');
Six_dof_mod.base=transl(0,0,0.25);
f=1;%画在第一张图上
theta=[0,pi/2,0,0,pi,0];
figure(f);
Six_dof_mod.plot(theta);
title('六轴机械臂模型');
%使用 teach 指令可调整各个关节角度
Six_dof_Tmod=SerialLink(L,'name','6-dof-Tmod');
f=2;%画在第二张图上
figure(f);
Six_dof_Tmod.plot(theta);
Six_dof_Tmod.teach;
```

```
title('六轴机械臂模型可调节');
%正向运动学分析
Six_dof_mod_move=SerialLink(L,'name','6-dof-move');
q0=[0,0,0,0,0,0];
T=Six_dof_mod_move.fkine(q0);
%逆向运动学分析(数值解)
q1=ikine(Six_dof_mod_move,T);
%工作空间可视化
num=100000;
p=zeros(num,3);
for i=1:num
 q1=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q2=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q3=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q4=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q5=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q6=L(1).qlim(1)+rand*( L(1).qlim(2)-L(1).qlim(1) );
 q=[q1 q2 q3 q4 q5 q6];
 T=Six_dof_mod.fkine(q);
 p(i,:)=transl(T);
end
f=3;
figure(f);
plot3(p(:,1),p(:,2),p(:,3),'b.','markersize',1)
title('机械臂工作空间');
```

2. 四次路径规划代码

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import rospy
import sys
import moveit_commander

class MoveItFkDemo:

    def __init__(self):
        # 初始化 move_group 的 API
        moveit_commander.roscpp_initialize(sys.argv)

        # 初始化 ROS 节点
        rospy.init_node('moveit_fk_demo', anonymous=True)
```

```
15.
16.             # 初始化需要使用 move group 控制的机械臂中的 arm group
17.             self.arm = moveit_commander.MoveGroupCommander('fr5_arm')
18.
19.             # 设置机械臂运动的允许误差值
20.             self.arm.set_goal_joint_tolerance(0.001)
21.
22.             # 设置允许的最大速度和加速度
23.             self.arm.set_max_acceleration_scaling_factor(0.5)
24.             self.arm.set_max_velocity_scaling_factor(0.5)
25.
26.         def move2pose(self, pose):
27.             # 设置机械臂的目标位置，使用六轴的位置数据进行描述（单位：弧度）
28.             # joint_positions = [0.391410, -0.676384, -
0.376217, 0.0, 1.052834, 0.454125]
29.             # arm.set_joint_value_target(joint_positions)
30.
31.             # 控制机械臂运动到预设位置 pose1
32.             self.arm.set_named_target(pose)
33.             self.arm.go()
34.             rospy.sleep(1)
35.
36.
37.     if __name__ == "__main__":
38.         try:
39.             demo = MoveItFkDemo()
40.             while 1:
41.                 demo.move2pose('pose1')
42.                 demo.move2pose('pose2')
43.                 demo.move2pose('pose3')
44.                 demo.move2pose('pose4')
45.                 # 关闭并退出 moveit
46.                 moveit_commander.roscpp_shutdown()
47.                 moveit_commander.os._exit(0)
48.         except rospy.ROSInterruptException:
49.             pass
```

3.运动规划演示代码

```
1.    #!/usr/bin/python3
2.    # -*- coding: utf-8 -*-
3.    import rospy
```

```
4.  import sys
5.  import moveit_commander
6.
7.  import sys
8.  import copy
9.  import rospy
10. import moveit_commander
11. import moveit_msgs.msg
12. import geometry_msgs.msg
13. from math import pi
14. from std_msgs.msg import String
15. from moveit_commander.conversions import pose_to_list
16.
17.
18. def all_close(goal, actual, tolerance):
19.
20.     if type(goal) is list:
21.         for index in range(len(goal)):
22.             if abs(actual[index] - goal[index]) > tolerance:
23.                 return False
24.
25.     elif type(goal) is geometry_msgs.msg.PoseStamped:
26.         return all_close(goal.pose, actual.pose, tolerance)
27.
28.     elif type(goal) is geometry_msgs.msg.Pose:
29.         return all_close(pose_to_list(goal), pose_to_list(actual), toleranc)
30.     return True
31.
32.
33. class MoveGroupTest(object):
34.
35.
36.     def __init__(self):
37.         super(MoveGroupTest, self).__init__()
38.
39.         # 初始化 moveit_commander API 和 rospy 节点
40.         moveit_commander.roscpp_initialize(sys.argv)
41.         rospy.init_node('move_group_test', anonymous=True)
42.
43.         # 初始化 RobotCommander 对象，提供机器人运动学模型和机器人当前关节状态
等信息，以及机器人与外界的接口等信息。
44.         robot = moveit_commander.RobotCommander()
```

```
45.
46.             # 初始化 PlanningSceneInterface 对象以提供机器人与周围世界之间的接口。
47.             scene = moveit_commander.PlanningSceneInterface()
48.
49.             # 初始化 MoveGroupCommander 对象。该对象是机械臂规划组的接口，用于规划和
执行机械臂运动。
50.             group_name = "fr5_arm"
51.             move_group = moveit_commander.MoveGroupCommander(group_name)
52.
53.             # 创建 ROS 发布器，用于在 RViz 中显示轨迹信息
54.             display_trajectory_publisher = rospy.Publisher('/move_group/display_
planned_path',moveit_msgs.msg.DisplayTrajectory, queue_size=20)

55.
56.             #获取机器人的参考系名称
57.             planning_frame = move_group.get_planning_frame()
58.             print ("============ Planning frame: %s" % planning_frame)
59.
60.             # 获取当前规划组中机器人末端执行器联动装置的名称
61.             eef_link = move_group.get_end_effector_link()
62.             print ("============ End effector link: %s" % eef_link)
63.
64.             # 获取机器人所有规划组的名称，并以数组形式输出
65.             group_names = robot.get_group_names()
66.             print ("============ Available Planning Groups:", robot.get_group_na
mes())
67.
68.             # 获取机器人当前状态
69.             print ("============ Printing robot state")
70.             print (robot.get_current_state())
71.             print ("")
72.
73.             self.box_name = ''
74.             self.robot = robot
75.             self.scene = scene
76.             self.move_group = move_group
77.             self.display_trajectory_publisher = display_trajectory_publisher
78.             self.planning_frame = planning_frame
79.             self.eef_link = eef_link
80.             self.group_names = group_names
81.
82.         def go_to_joint_state(self):
```

```
83.
84.             #计划并执行到联合目标位置
85.             # ^^^^^^^^^^^^^^^^^^^^^^^^
86.             # 机器人的初始姿势是单一姿势，首先我们将其移动到更好的位置。
87.             # 首先从界面获取当前机器人的关节值，然后修改部分关节的值
88.             joint_goal = self.move_group.get_current_joint_values()
89.             print ("============ Printing current joint values: ", joint_goal)
90.             joint_goal[0] = 0
91.             joint_goal[1] = -pi/4
92.             joint_goal[2] = -pi/2
93.             joint_goal[3] = -pi/2
94.             joint_goal[4] = pi/3
95.             joint_goal[5] = 0
96.             print ("============ Printing joint goal: ", joint_goal)
97.
98.             #移动至关节目标位置
99.             self.move_group.go(joint_goal, wait=True)
100.            rospy.sleep(1)
101.            # 调用停止指令以确保没有残余运动
102.            self.move_group.stop()
103.
104.            # For testing:
105.            current_joints = self.move_group.get_current_joint_values()
106.            return all_close(joint_goal, current_joints, 0.01)
107.
108.        def go_to_pose_goal(self):
109.
110.            # 计划并执行最终目标姿势
111.            # ^^^^^^^^^^^^^^^^^^^^^^^^
112.            pose_goal = geometry_msgs.msg.Pose()
113.            pose_goal.orientation.w = 1.0
114.            pose_goal.position.x = 0.4
115.            pose_goal.position.y = 0.1
116.            pose_goal.position.z = 0.4
117.
118.            # 设定目标姿势
119.            self.move_group.set_pose_target(pose_goal)
120.
121.            # 计划并执行目标姿势
122.            self.move_group.go(wait=True)
123.            rospy.sleep(1)
124.            self.move_group.stop()
```

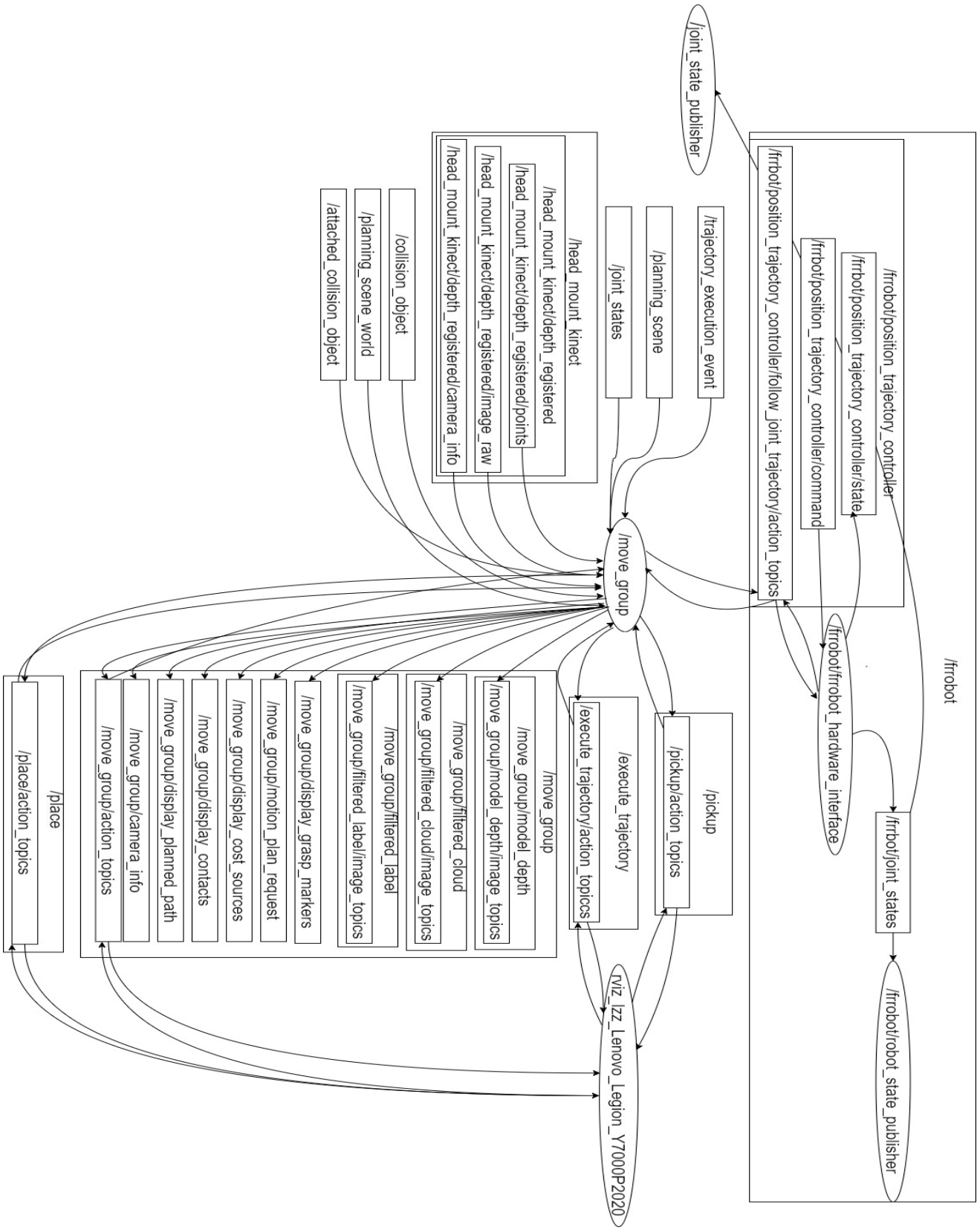
```
125.
126.         # 清除目标姿态
127.         self.move_group.clear_pose_targets()
128.
129.         # 测试用:
130.         current_pose = self.move_group.get_current_pose().pose
131.         return all_close(pose_goal, current_pose, 0.01)
132.
133.     def plan_cartesian_path(self, scale=1.0):
134.
135.         # 规划并执行笛卡尔路径
136.         # ^^^^^^^^^^^^^^^^^^^
137.         # 通过指定末端效应器要通过的路点列表，可以直接规划笛卡尔路径。
138.         waypoints = []
139.
140.         wpose = self.move_group.get_current_pose().pose
141.         wpose.position.z -= scale * 0.1  # 沿 z 轴向下移动
142.         wpose.position.y += scale * 0.2  # 在 y 轴上横向移动
143.         waypoints.append(copy.deepcopy(wpose))
144.
145.         wpose.position.x += scale * 0.1  # 沿 x 轴向前移动
146.         waypoints.append(copy.deepcopy(wpose))
147.
148.         wpose.position.y -= scale * 0.1  # 在 y 轴上横向移动
149.         waypoints.append(copy.deepcopy(wpose))
150.
151.         # 我们想要以 1cm 的分辨率对笛卡尔路径进行插值，这就是为什么我们在笛卡尔变
换中将 eef_step 指定为 0.01。
152.         # 通过将其设置为 0.0，可以禁用跳跃阈值并忽略对关节空间中不可行的跳跃的检
查。
153.         (plan, fraction) = self.move_group.compute_cartesian_path(
154.             waypoints,   # 遵循的航路点
155.             0.01,        # eef_step
156.             0.0)         # 跳跃阈值
157.
158.         # 目前只是规划路径，并没有让 move_group 执行并移动机器人
159.         return plan, fraction
160.
161.     def display_trajectory(self, plan):
162.
163.         # 显示运动轨迹
164.         # ^^^^^^^^^^
```

```
165.          # DisplayTrajectory msg 有两个主要字段，trajectory_start 和
trajectory。
166.          # 我们用当前的机器人状态填充轨迹_start，以复制任
何 AttachedCollisionObjects 并将我们的计划添加到该轨迹中。
167.          display_trajectory = moveit_msgs.msg.DisplayTrajectory()
168.          display_trajectory.trajectory_start = self.robot.get_current_state()

169.          display_trajectory.trajectory.append(plan)
170.          #发布到/move_group/display_planned_path 主题
171.          self.display_trajectory_publisher.publish(display_trajectory)
172.
173.      def execute_plan(self, plan):
174.
175.          #执行计划的笛卡尔路径
176.          #^^^^^^^^^^^^^^^^
177.          self.move_group.execute(plan, wait=True)
178.
179.          #注意：机器人当前的关节状态必须在 RobotTrajectory 中第一个路径点的容差范围
内，否则 execute()将会失败
180.
181.  def main():
182.      try:
183.          print ("-------欢迎来到基于 ROS 和 MoveIt 的六自由度机械臂控制演示-------
"    )
184.          print ("按 Ctrl+D 退出演示")
185.          print ("按 Enter 设置并初始化 moveit_commander 以启动演示 ...")
186.          input()
187.          fr5demo = MoveGroupTest()
188.
189.          print ("按“Enter”键移动到关节空间中的目标位置 ...")
190.          input()
191.          fr5demo.go_to_joint_state()
192.
193.          print ("按“Enter”键移动到笛卡尔空间目标位置 ...")
194.          input()
195.          fr5demo.go_to_pose_goal()
196.
197.          print ("按“Enter”键规划并演示笛卡尔空间中的路径...")
198.          input()
199.          cartesian_plan, fraction =  fr5demo.plan_cartesian_path()
200.
201.          print ("按“Enter”键演示笛卡尔空间中保存的路径  ...")
```

```
202.          input()
203.          fr5demo.display_trajectory(cartesian_plan)
204.
205.          print ("按‘Enter’键执行保存的笛卡尔空间路径 ...")
206.          input()
207.          fr5demo.execute_plan(cartesian_plan)
208.
209.          print ("Demo 完成! ")
210.      except rospy.ROSInterruptException:
211.          return
212.      except KeyboardInterrupt:
213.          return
214.
215.
216.  if __name__ == '__main__':
217.      main()
```

附录 2 通讯节点图


/joint_state_publisher
/frrobot
/frrobot/position_trajectory_controller
/frrobot/position_trajectory_controller/state
/frrobot/position_trajectory_controller/command
/frrobot/position_trajectory_controller/follow_joint_trajectory/action_topics
/frrobot/frrobot_hardware_interface
/frrobot/joint_states
/frrobot/robot_state_publisher
/trajectory_execution_event
/planning_scene
/joint_states
/head_mount_kinect
/head_mount_kinect/depth_registered
/head_mount_kinect/depth_registered/points
/head_mount_kinect/depth_registered/image_raw
/head_mount_kinect/depth_registered/camera_info
/collision_object
/planning_scene_world
/attached_collision_object
/move_group
/execute_trajectory
/execute_trajectory/action_topics
/pickup
/pickup/action_topics
/move_group
/move_group/model_depth
/move_group/model_depth/image_topics
/move_group/filtered_cloud
/move_group/filtered_cloud/image_topics
/move_group/filtered_label
/move_group/filtered_label/image_topics
/move_group/display_grasp_markers
/move_group/motion_plan_request
/move_group/display_cost_sources
/move_group/display_contacts
/move_group/display_planned_path
/move_group/camera_info
/move_group/action_topics
/place
/place/action_topics
/rviz_lzz_Lenovo_Legion_Y7000P2020

致谢

时光匆匆，四年的北化学生生涯即将结束。在这四年的本科生涯中，知识的积累、科研能力的提升使我的学术能力有了较大提升，老师们的帮助、同学们的友好相处更使我在情感的羽翼更加丰满。在此，向帮助我的所有的老师们和同学们致以诚挚的感谢！

首先感谢魏彬老师的悉心指导和帮助！魏老师对待科研认真严谨的态度在毕设过程中产生了很大的影响。当我毕设遇到问题时，魏老师耐心帮助我解决毕设问题，为我指明研究方向。在此衷心祝福黄老师工作顺利，您一丝不苟的科研态度和真诚的待人处事的方式，将会是我终身学习的榜样，真的谢谢您！

其次我要感谢我的各位师兄和同学们。感谢衡霖宇师兄和严腾风师兄对我毕设的帮助！即使二位师兄学习科研很忙，但总是会愿意倾听，帮我全面地分析，希望我可以成为和师兄们一样乐观善良的人！感谢机工 2001 班的同学们对我学习和生活中的帮助，以后继续一起加油啊！

感谢我的室友们：殷成威、徐会全，谢谢你们容忍我的任性，谢谢你们永远在我身边支持我，谢谢你们在我彷徨时给我的帮助！让我在疫情期间即使在家也不会感觉枯燥和无聊，并在我学习彷徨时给予我鼓励与帮助。

感谢我的父母，感谢你们始终以我为骄傲，感谢你们尊重我的一切选择。只要你们在，欢声笑语就在！