机器人学导论实验

实验报告

智能科学与技术 张恒硕 2212266

二〇二五年五月

目录

引言

随着机器人技术的快速发展，机械臂的运动规划与控制已成为工业自动化和智能服务机器人领域的核心研究方向。本实验以 KUKA iiwa7 机械臂为研究对象，基于 Ubuntu 20.04 操作系统与 ROS1 Noetic 集成环境，结合 MoveIt 运动规划框架、RViz 可视化工具与 Gazebo 仿真环境，探索机械臂的点位控制与轨迹控制方法，实现了机械臂的无碰撞路径规划点位控制、避障控制以及复杂轨迹控制的功能。

实验核心目标：

- 平台搭建与配置：完成 Ubuntu 20.04、ROS1 Noetic、MoveIt、RViz 及 Gazebo 的环境配置，确保软件生态的稳定性和兼容性；
- 机械臂建模与仿真：导入 iiwa7 机械臂的 urdf 模型，通过 MoveIt Setup Assistant 生成运动规划配置文件，并在 Gazebo 中验证其动力学特性；
- 控制算法实现：基于 MoveIt 框架，开发点位控制与轨迹控制模块，分别实现五角星点位控制、避障路径规划以及任意字符（如字母、数字）的轨迹控制；
- 仿真结果分析：通过 RViz 与 Gazebo 的联合仿真，验证控制算法的有效性，并分析不同场景（无障碍与障碍物环境）下的规划性能。

当前实验内容可参考本人 github 开源仓库 https://github.com/xunlan11/moveit_ws。

第一章 实验平台与环境配置

1.1 Ubuntu20.04

Ubuntu 是基于 Linux 发行版的开源操作系统，由英国公司 Canonical Ltd.开发和维护。其首发于 2004 年 10 月，之后每半年发布一个新版本，每两年发布一个长期支持版本，按年月命名。

所用的 Ubuntu20.04 的支持期限为 2020 年 4 月 - 2025 年 4 月，虽然已经超出范围，但是其是目前最为稳定的最新版本。

Ubuntu 系统主要由以下核心组件构成：

- Linux 内核：系统核心，管理硬件资源。
- 桌面环境：主要使用 GNOME，也支持其他 DE。
- 包管理器：APT (Advanced Package Tool)。
- 软件仓库：官方和社区维护的软件源。

安装方式略，请于 csdn 寻找教程，建议丰富浏览，以确保安装前后的注意事项，减少重装次数。

在安装后，更换软件源，并更新软件包索引：

```
sudo apt update
sudo apt dist-upgrade
```

以上操作可以在每填第一次下载包前都进行一次。

1.2 Ros1-noetic

1.2.1 Ros1 介绍

ROS（Robot Operating System）是一个用于机器人软件开发的灵活框架，提供了包括硬件抽象、底层设备控制、常用功能实现、进程间消息传递以及包管理等服务。其作为一个开源免费的丰富生态框架，支持 C++和 Python 多语言，支持分布式多机器人协作，并凭借松耦合的节点通信支持模块化设计。

其核心概念为：

- 节点（Node）：基本计算单元，负责特定功能，相互通信。
- 话题（Topic）：节点间异步通信的总线，为发布者-订阅者模式，支持一对多、多对一通信。

ROS Noetic 是 ROS1 的最后一个发行版，也是当前最成熟稳定的版本，其后转向 ROS2。其适配的目标平台为 Ubuntu 20.04，支持期限为 2020 年 5 月 – 2025 年 5 月，即将到期。相比于先前版本，它支持 Python 3，拥有更好的内存管理和计算效率，并在稳定性和向前兼容性上表现出色。

1.2.2 Ros1-noetic 安装

推荐一个便捷的一键安装方法——鱼香 ROS，其是国人汇总的常用软件源，并省略了繁冗的安装步骤，适合新手，按照指引输入数字即可安装包括 ROS 在内的常用软件，安装过程中的选项也提供了适合新手的默认值。

```
wget http://fishros.com/install -O fishros && . fishros
```

1.2.3 Rosdep

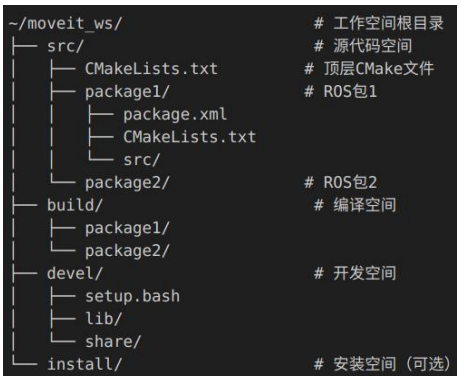
Rosdep 是 ROS 的依赖管理工具，可以自动解析和安装 ROS 包的系统依赖，管理不同平台的包依赖关系，并确保编译和运行时的依赖完整性。

使用以下命令安装 Rosdep 并初始化、更新：

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
sudo rosdep init
rosdep update
```

1.2.4 Ros 工作空间

ROS 工作空间是一个目录结构，用于组织和管理 ROS 项目的源代码、编译文件和配置信息，是 ROS 开发的基础环境，其按照下图格式排布。



先安装构建工具：

```
sudo apt install ros-noetic-catkin python3-catkin-tools
```

其后创建上述 moveit_ws 工作空间：

```
mkdir -p ~/moveit_ws/src
cd ~/moveit_ws/src
catkin_init_workspace
cd ~/moveit_ws
catkin_make
```

这里使用“catkin_make”进行编译，其是并行的编译方法，也可采用串行的“catkin build”进行编译，二者有很大区别。以下编译一般采用快捷的前者，但是在指定使用后者时，一定要采用后者。

在编译后，可以选择将当前工作空间的环境变量放入“.bashrc”文件中，避免每次打开窗口都要更新环境变量“source ~/moveit_ws/devel/setup.bash”。

```
echo "source ~/moveit_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

1.3 Movelt

Movelt 是 ROS 中重要的机器人运动规划和控制框架，为机械臂和移动机器人提供了一套完整的运动规划、操作和控制解决方案。

核心功能：

- 运动规划：智能路径规划和轨迹生成。
- 碰撞检测：实时环境感知和避障。
- 运动学求解：正向和逆向运动学计算。
- 可视化：RViz 集成的直观操作界面。
- 模块化设计：灵活的插件架构。

使用以下命令安装 MoveIt 及其依赖：

```
sudo apt install ros-noetic-moveit
sudo apt install ros-noetic-moveit-ros-visualization ros-noetic-moveit-planners
ros-noetic-moveit-ros-move-group ros-noetic-moveit-ros-perception
```

1.4 RViz

RViz（ROS Visualization）是 ROS 的官方 3D 可视化工具，是机器人开发中最重要的调试和展示工具之一，提供了直观的 3D 环境来可视化机器人状态、传感器数据和规划结果。

核心特点：

- 3D 可视化：实时显示机器人模型和环境。
- 多数据类型支持：激光雷达、相机、点云、轨迹等。
- 交互式操作：通过图形界面控制机器人。
- 插件化架构：丰富的显示插件和扩展。
- 配置保存：可保存和共享可视化配置。

使用以下命令安装 RViz：

```
sudo apt install ros-noetic-rviz
```

1.5 Gazebo

Gazebo 是一个开源 3D 机器人仿真器，提供了高保真的物理仿真环境，是机器人开发中最重要的仿真工具之一，能够在虚拟环境中测试机器人算法和控制系统。

核心特点：

- 物理引擎：基于 ODE、Bullet、DART 等。
- 传感器仿真：激光雷达、相机、IMU、GPS 等。
- 动力学仿真：精确的刚体动力学和碰撞检测。
- 可视化渲染：基于 OGRE 图形引擎的高质量渲染。
- ROS 集成：与 ROS 深度集成，无缝数据交互。

使用以下命令安装 Gazebo：

```
sudo apt install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-ros-control
sudo apt install ros-noetic-gazebo-plugins
sudo apt install ros-noetic-rviz-visual-tools
```

第二章 iiwa7 机械臂

2.1 iiwa7 机械臂介绍

KUKA iiwa（intelligent industrial work assistant）是德国 KUKA 公司开发的智能工业协作机械臂系列，代表了现代工业机器人的先进技术水平。其具有协作性、高精度、智能感知、灵活编程的特点。

参数：

- 自由度：7-DOF
  - 关节 1（A1）：基座旋转±170°
  - 关节 2（A2）：肩部俯仰±120°
  - 关节 3（A3）：肘部旋转±170°
  - 关节 4（A4）：肘部俯仰±120°
  - 关节 5（A5）：腕部旋转±170°
  - 关节 6（A6）：腕部俯仰±120°
  - 关节 7（A7）：法兰旋转±175°
- 负载能力：7kg。
- 工作半径：800mm。
- 重复精度：±0.1mm。
- 重量：约 23.9kg。

2.2 iiwa7 机械臂包导入

首先下载 urdf 文件：

```
cd ~/moveit_ws/src
git clone https://github.com/facebookresearch/differentiable-robot-model.git
sudo apt install ros-noetic-franka-description
sudo ubuntu-drivers autoinstall
```

然后为 differentiable-robot-model 添加 package.xml 和 CMakeLists.txt，配置为 ros 包，详细代码见仓库。下载时也可以只下载 urdf 文件和其关联项，但要注意正确的索引。

moveit_ws/src/differentiable-robot-model/diff_robot_data/kuka_iiwa/urdf/iiwa7.urdf 文件需要进行修改：

- 索引文件改为包的形式。
- 固定基坐标系。
- 修改过时语法。

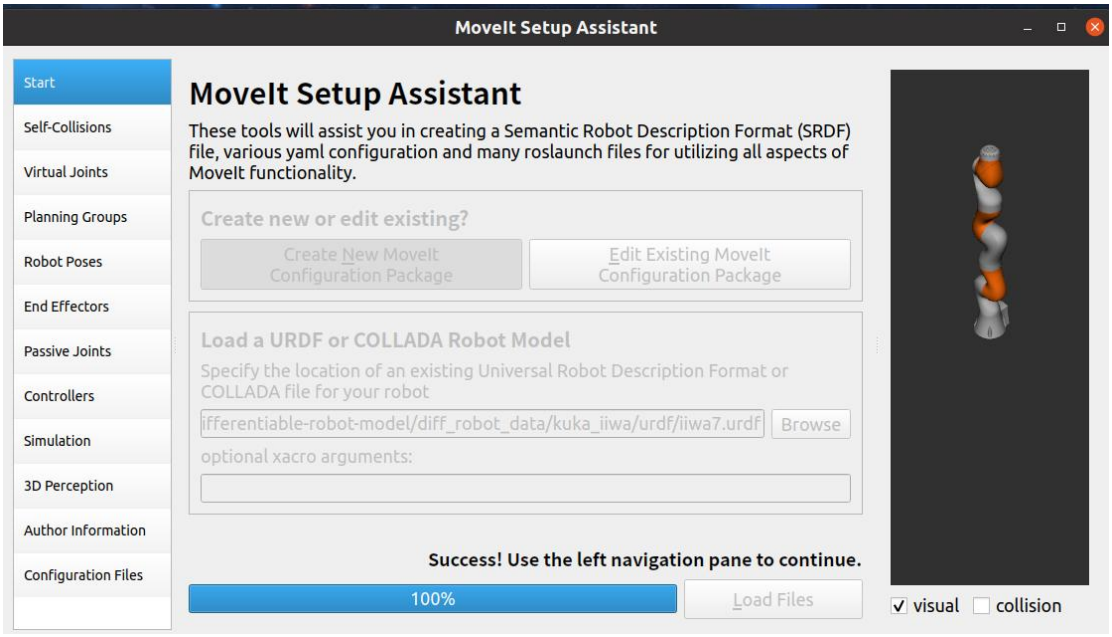
建议直接从库中复制修改后的文件，避免遗漏。

之后启动 moveit_setup_assistant，按照下述操作获取机械臂配置文件：

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

- 放大页面到完全显示。
- 导入文件：Create New MoveIt Configuration Package -> 选择上述 urdf 文件 ->

Browse（正确导入会显示如下图片）。



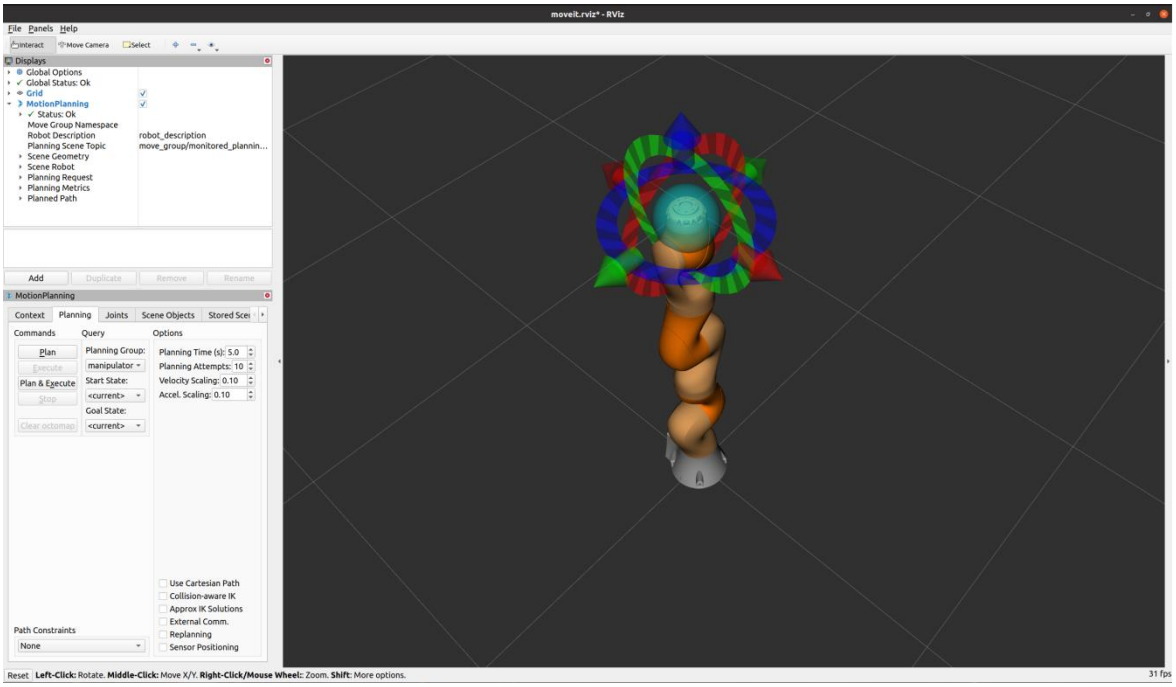
- 自碰撞矩阵：Self-Collisions -> Generate Collision Matrix。
- 虚拟基座关节：Virtual Joints -> Add Virtual Joint -> Virtual Joint Name:virtual_joint -> Parent Frame Name:world。
- 规划组：Planning Groups -> Add Group -> Group Name:manipulator -> Kinematic Solver:kdl_kinematics_plugin/KDLKinematicsPlugin -> Add Joints:iiwa_joint_1---iiwa_joint_7, 左端>右端。
- 原姿态：Robot Poses -> Add Pose -> Pose Name:home -> Planning Group:manipulator -> 默认初始零值。
- 末端执行器：End Effectors -> Add End Effector -> End Effector Name:end_effector -> End Effector Group:manipulator -> Parent Link:iiwa_link_7。
- 作者信息填写：Author Information -> 略。
- 生成配置文件：Configuration Files -> 路径/名称 -> Generate Package。

2.3 iiwa7 机械臂测试

重新编译后使用 launch 文件启动 rviz 进行测试：

```
cd ~/moveit_ws
catkin_make
roslaunch iiwa7_moveit_config demo.launch
```

由于即将到期，会弹出提示框，需要按确认后再进行后续操作。以下展示了导入成功的结果。



第三章 原理与实现

3.1 点位控制

3.1.1 点位控制原理

通过与机器人运动规划软件建立通信，初始化机器人模型和场景信息。允许在虚拟环境中添加或移除障碍物，以模拟真实作业场景。当需要机器人运动时，用户可以指定一系列目标点，这些目标点可以是机器人各关节的目标角度，也可以是末端执行器在三维空间中的目标位置和姿态。控制系统接收到这些目标后，会综合考虑机器人自身的运动学约束以及环境中障碍物的位置，自动规划出一条从当前状态到达目标状态的无碰撞、平滑的运动轨迹。规划完成后，系统将该轨迹发送给机器人硬件执行实际的物理移动。此外，还支持将规划的轨迹和场景中的物体在可视化界面中显示，方便用户监控和调试整个运动过程。

3.1.2 点位控制实现——五角星&避障

```
类 PointControl:
    方法 __init__():
        1. 初始化 MoveIt 和 ROS。
        2. 设置机器人、规划场景、运动组和 Rviz 发布器。
        3. 获取并记录机器人基本信息（参考系、末端执行器等）。
```

方法 add_obstacle_box(名称, 位姿, 大小):

1. 在规划场景中添加一个盒子障碍物。
2. 等待并确认障碍物已添加。

方法 remove_obstacle(名称):

1. 从规划场景中移除指定的障碍物。
2. 等待并确认障碍物已移除。

方法 move_J(目标关节角度):

1. 设置运动组的关节角度目标。
2. 规划并执行运动。

方法 plan_cartesian_path(路径点列表):

1. 计算笛卡尔路径。
2. 返回规划结果和成功比例。

方法 execute_plan(规划结果):执行已规划的轨迹。

方法 clear_markers():发送消息清除 Rviz 中的所有标记。

方法 show_path(路径点姿态列表, 颜色):

1. 清除旧标记。
2. 发送消息在 Rviz 中显示新路径。

方法 run():

// --- 无障碍物演示 ---

1. 清除场景障碍物和 Rviz 标记。
2. 机器人移动到初始关节姿态 (调用 move_J)。
3. 计算五角星形状的路径点。
4. 在 Rviz 中显示五角星路径 (调用 show_path)。
5. 规划无障碍物的笛卡尔路径 (调用 plan_cartesian_path)。
6. 在 Rviz 中显示规划的轨迹。
7. 执行无障碍物路径 (调用 execute_plan)。

// --- 有障碍物演示 ---

8. 机器人移回初始关节姿态。
9. 在五角星路径附近添加一个盒子障碍物 (调用 add_obstacle_box)。
10. 规划有障碍物的笛卡尔路径。
11. 在 Rviz 中显示规划的轨迹。
12. 执行有障碍物路径（可能部分执行）。
13. 移除障碍物 (调用 remove_obstacle)。

主程序:

1. 创建 PointControl 对象。
2. 调用对象的 run() 方法开始演示。

3.2 轨迹控制

3.2.1 轨迹获取与控制原理

首先解析字体文件获取指定字符（任意字符，推荐简单的字母或数字）的轮廓（闭环的内外轮廓，部分字符由于内外轮廓无法连接而无法完全执行轨迹）数据，其由直

线段和不同阶次的贝塞尔曲线段组成。接着，将连续几何线段离散化为二维坐标点，并通过缩放和平移变换，映射到机器人的三维工作空间中，形成一系列目标位姿点，其中 X 轴保持固定，而 Y 和 Z 轴则根据字符形状进行变化，同时保持末端执行器的姿态不变。最后，机器人运动控制系统接收这些三维路径点，利用运动规划算法计算出一条平滑的笛卡尔空间轨迹，并驱动机器人末端执行器精确地沿该轨迹运动，从而在预定平面上完成字符的绘制。整个流程还包含了人机交互、路径可视化以及机器人姿态的初始化与重置等辅助控制环节。

3.2.2 轨迹获取实现——字符（字母/数字）内外轮廓

```
类 OutlineDecomposer:
    方法 __init__(每个曲线段的采样点数):
        1. 初始化一个空列表 `points_2d_font_units` 用于存储字体单位的 (x,y) 点。
        2. 初始化 `_current_pen_pos` (当前画笔位置) 为 None。
        3. 设置 `_num_samples` (每个曲线段的采样点数)，确保至少为 2。
    方法 _add_point(点元组):如果点列表为空，或者新点与上一个点的 x 或 y 坐标差大于一个小阈值(epsilon)，则将点元组添加到 `points_2d_font_units` 列表。
    方法 move_to(目标点 a): // FreeType 回调：移动画笔到新位置
        1. 更新 `_current_pen_pos` 为 (a.x, a.y)。
        2. 调用 `_add_point` 添加当前画笔位置。
    方法 line_to(目标点 a): // FreeType 回调：从当前位置画直线到目标点
        1. 如果 `_current_pen_pos` 为 None (之前没有 move_to)，记录警告，将当前位置设为 (a.x, a.y)。
        2. 更新 `_current_pen_pos` 为 (a.x, a.y)。
        3. 调用 `_add_point` 添加当前画笔位置。
    方法 conic_to(控制点, 终点): // FreeType 回调：画二次贝塞尔曲线
        1. 如果 `_current_pen_pos` 为 None，记录警告并返回。
        2. 获取起点 p0 (当前画笔位置)，控制点 p1_ctrl，终点 p2_end。
        3. 循环 `_num_samples` - 1 次 (从 1 到 `_num_samples`-1):
            a. 计算参数 t (从 0 到 1)。
            b. 使用二次贝塞尔公式计算曲线上的点 (x, y)。
            c. 调用 `_add_point` 添加计算出的点。
        4. 更新 `_current_pen_pos` 为终点 p2_end。
    方法 cubic_to(控制点 1, 控制点 2, 终点): // FreeType 回调：画三次贝塞尔曲线
        1. 如果 `_current_pen_pos` 为 None，记录警告并返回。
        2. 获取起点 p0 (当前画笔位置)，控制点 1 p1_ctrl，控制点 2 p2_ctrl，终点 p3_end。
        3. 循环 `_num_samples` - 1 次 (从 1 到 `_num_samples`-1):
            a. 计算参数 t (从 0 到 1)。
            b. 使用三次贝塞尔公式计算曲线上的点 (x, y)。
            c. 调用 `_add_point` 添加计算出的点。
```

4. 更新 `_current_pen_pos` 为终点 p3_end。

函数 generate_letter_waypoints_from_font(字符, 字体路径, 期望高度, 每曲线采样数, 中心位姿, rospy 实例):

1. 初始化日志记录器和空路径点列表 `waypoints`。
2. 验证 `center_pose` 是否提供。
3. 从 `center_pose` 提取机器人 X 轴基准位置 `base_x`，Y 轴偏移 `offset_y_world`，Z 轴偏移 `offset_z_world`，以及固定姿态 `fixed_orientation`。
4. **加载字体**:
    a. 使用 FreeType 加载指定 `font_path` 的字体文件。
    b. 如果加载失败，记录错误并返回空列表。
5. **设置字体大小**:设置字体像素大小（例如，64 像素高）用于轮廓提取。
6. **加载字符字形**:
    a. 加载指定 `char_to_draw` 的字形轮廓，不进行缩放或位图渲染。
    b. 如果加载失败，记录错误并返回空列表。
7. **分解轮廓**:
    a. 获取字形的轮廓数据。
    b. 创建 `OutlineDecomposer` 类的实例，传入 `num_samples_per_curve`。
    c. 调用轮廓的 `decompose` 方法，将 `OutlineDecomposer` 实例的 `move_to`, `line_to`, `conic_to`, `cubic_to` 方法作为回调函数。这将填充分解器的 `points_2d_font_units` 列表。
    d. 获取分解后的 2D 点列表 (字体单位)。
    e. 如果列表为空，记录警告并返回空列表。
8. **计算缩放和居中参数**:
    a. 计算字体单位下，所有点的最小/最大 X 和 Y 值，得到字符的边界框。
    b. 计算字符在字体单位下的宽度 `font_width_fu` 和高度 `font_height_fu`。
    c. 根据 `desired_letter_height_m` 和 `font_height_fu` (或 `font_width_fu` 如果高度为零) 计算缩放因子 `scale_factor`。处理高度或宽度为零的情况。
    d. 计算字符在字体单位下的中心点 (`center_font_x`, `center_font_y`)。
9. **转换点为机器人路径点 (Pose)**:遍历 `font_units_points` 中的每个字体单位点 (fx, fy):
    a. 将字体点相对于其中心进行平移: `(fx - center_font_x)`, `(fy - center_font_y)`。
    b. 应用 `scale_factor` 进行缩放，得到局部坐标 `local_y` 和 `local_z`。
        (注意坐标映射：字体 X -> 机器人 Y，字体 Y -> 机器人 Z)
    c. 创建一个新的 `Pose` 对象。
    d. 设置 `pose.position.x` 为 `base_x` (固定)。
    e. 设置 `pose.position.y` 为 `offset_y_world + local_y`。
    f. 设置 `pose.position.z` 为 `offset_z_world + local_z`。
    g. 设置 `pose.orientation` 为 `fixed_orientation`。
    h. 将此 `Pose` 对象添加到 `waypoints` 列表。

10. 记录生成路径点的数量。

11. 返回 `waypoints` 列表。

3.2.3 轨迹控制实现——字符（字母/数字）内外轮廓

类 TrajectoryControl:

方法 __init__():

1. 初始化 MoveIt 和 ROS。
2. 设置机器人、运动组（设置规划时间）和 Rviz 标记发布器。
3. 获取并记录机器人基本信息（参考系、末端执行器、初始位姿等）。

方法 move_J(目标关节角度):

1. 设置运动组的关节角度目标。
2. 规划并执行运动。

方法 plan_cartesian_path(路径点列表):

1. 设置运动组的最大速度和加速度缩放因子。
2. 计算笛卡尔路径。
3. 返回规划结果和成功比例。

方法 execute_plan(规划结果):执行已规划的轨迹。

方法 clear_markers(命名空间):发送消息清除 Rviz 中指定命名空间下的所有标记。

方法 show_path(路径点姿态列表, 命名空间, 颜色):

1. 清除指定命名空间下的旧标记。
2. 发送消息在 Rviz 中显示新路径。

方法 run():

1. 机器人移动到预定义的初始关节姿态（调用 move_J)。
2. 进入循环，允许用户多次绘制字符:
    a. 获取当前末端执行器位姿。
    b. 清除上一个字符的 Rviz 标记（调用 clear_markers)。
    c. 提示用户输入要绘制的单个字符（字母/数字）。
        i. 如果用户直接回车，则退出循环。
        ii. 处理输入错误，如多字符输入或读取错误。
    d. 如果成功获取字符：
        i. 调用外部 `letter.generate_letter_waypoints_from_font` 函数：
            - 传入要绘制的字符、字体路径、期望高度、采样点数、当前末端位姿（作为绘制中心）。
            - 生成字符的路径点列表。
        ii. 如果路径点生成成功：
            - 在 Rviz 中显示生成的字符路径（调用 show_path)。
            - 规划笛卡尔路径（调用 plan_cartesian_path)。

- 如果规划成功比例足够高，则执行路径（调用 execute_plan）。
- 否则，记录规划覆盖率低且不执行。

iii. 如果路径点生成失败，记录错误。

e. 机器人返回到初始关节姿态（调用 move_J）。

f. 询问用户是否继续绘制下一个字符。如果否，则退出循环。

主程序：

1. 创建 TrajectoryControl 对象。
2. 调用对象的 run() 方法开始字符绘制循环。
3. 程序结束时，关闭 MoveIt。

3.3 控制包实现

3.3.1 创建控制包

创建控制包，注意导入正确的依赖。因为要使用 Python 编写代码，创建“scripts”文件夹。

```
cd ~/moveit_ws/src
catkin_create_pkg iiwa7_control moveit_ros_planning_interface roscpp rospy std_msgs geometry_msgs
cd iiwa7_control
mkdir -p scripts
```

3.3.2 编写控制代码

创建代码文件，并为需要运行的程序赋予权限，详细代码见仓库。

```
touch scripts/point_control.py
chmod +x scripts/point_control.py
touch scripts/letter.py
touch scripts/trajectory_control.py
chmod +x scripts/trajectory_control.py
```

3.3.3 Gazebo 文件

为启动 Gazebo 进行观察，需要编写以下两个文件。

一是控制器配置文件 iiwa7_moveit_config/config/controllers.yaml，详细代码见仓库。其定义了两套 PID 参数：

- iwa_arm_controller/gains：作用于上层控制器。影响跟踪输入的目标轨迹、平滑过渡和跟踪误差响应
- gazebo_ros_control/pid_gains：作用于 Gazebo 插件。直接模拟硬件底层伺服控制器行为，决定仿真环境中的物理表现（刚度、阻尼、扰动抵抗能力，响应速度和

稳定性）

二是 Gazebo 启动文件 iiwa7_moveit_config/launch/iiwa7_gazebo.launch，详细代码见仓库。

3.3.4 重新编译

在正式进行测试前，建议清理重新编译：

```
cd ~/moveit_ws
rm -rf build/ devel/
catkin build
```

第四章 仿真与结果

4.1 点位控制

4.1.1 Movelt+RViz 仿真

```
roslaunch iiwa7_moveit_config demo.launch
rosrun iiwa7_control point_control.py
```

在进行以上两个操作中间，为方便观察，进行以下两个操作：

- 去除原始机械臂：左上 Displays -> MotionPlanning -> Planning Request -> Query Goal State 取消。
- 可视化轨迹：左中 Add -> Maker -> OK -> 左中 Maker -> Maker Topic 下拉选取话题。

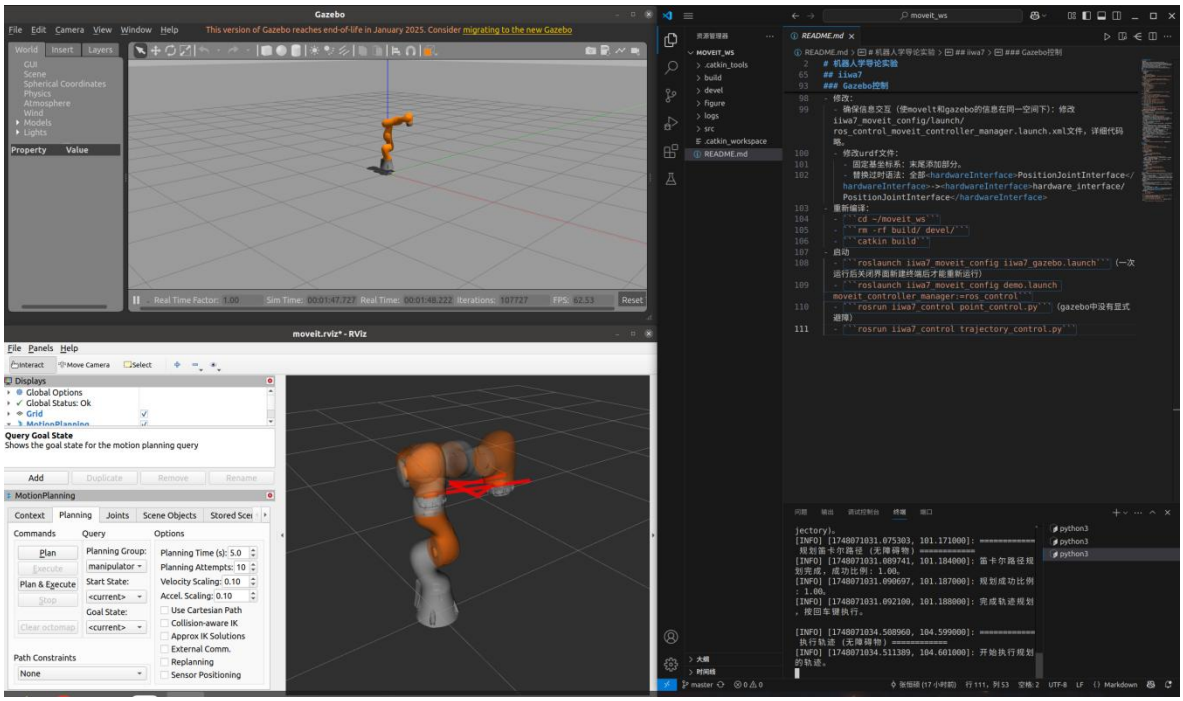
4.1.2 Gazebo 仿真

```
roslaunch iiwa7_moveit_config iiwa7_gazebo.launch
roslaunch iiwa7_moveit_config demo.launch moveit_controller_manager:=ros_control
rosrun iiwa7_control point_control.py
```

建议每次运行 Gazebo 后关闭界面新建终端后重新运行。

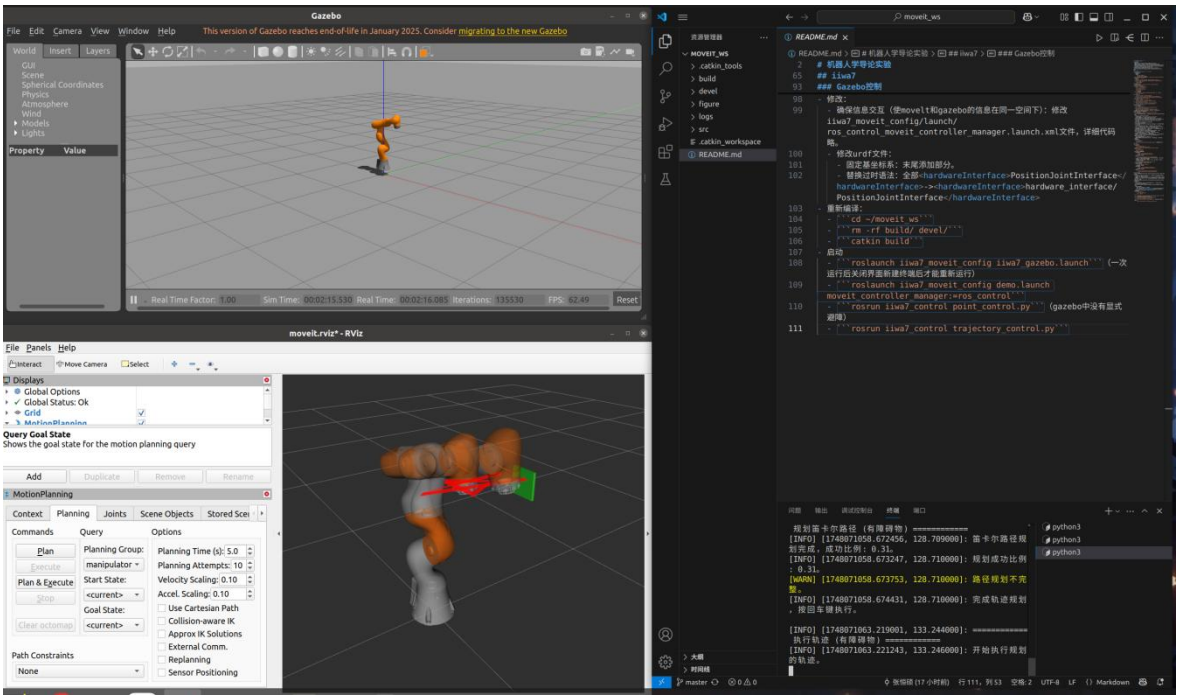
4.1.3 仿真结果

五角星：

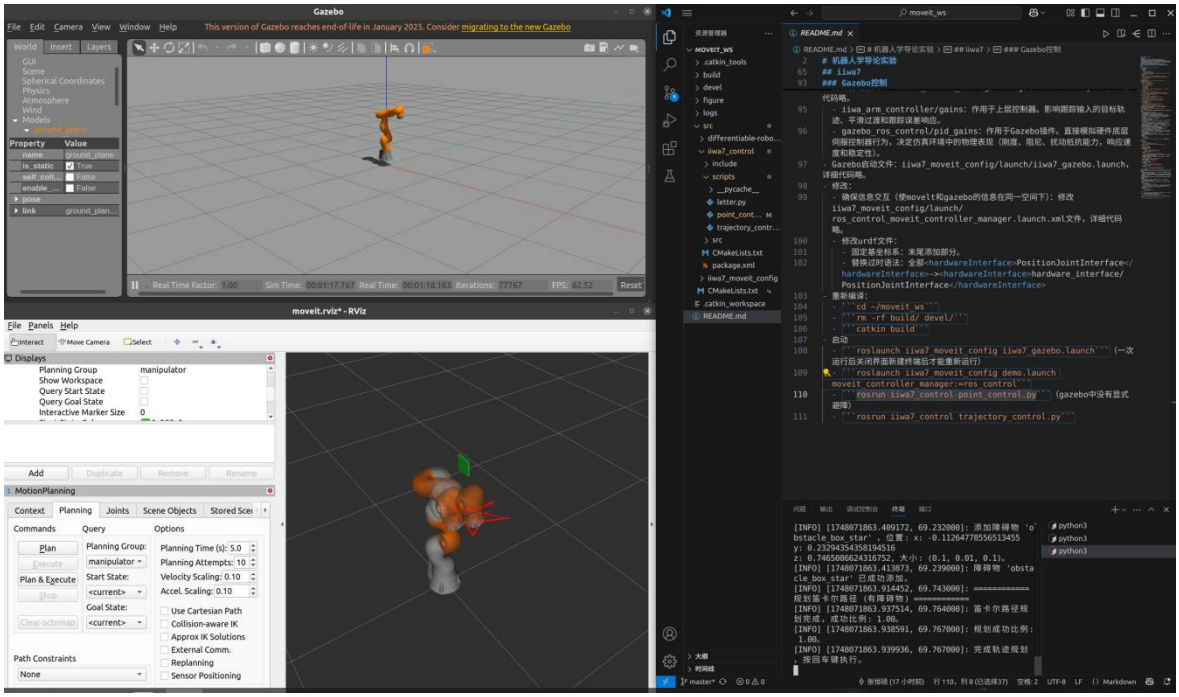


设置五个定点，机械臂能按顺序（非相邻顺序，五角星绘制顺序）完成五角星轨迹的移动。

避障：



在障碍物完全阻挡轨迹时，无法完成轨迹规划，只执行已完成规划的部分轨迹。



在障碍物没有完全阻挡轨迹，只是限制部分关节活动空间时，仍可以完成轨迹规划，执行全部轨迹。

4.2 轨迹控制

4.2.1 Movelt+RViz 仿真

```
roslaunch iiwa7_moveit_config demo.launch
rosrun iiwa7_control trajectory_control.py
```

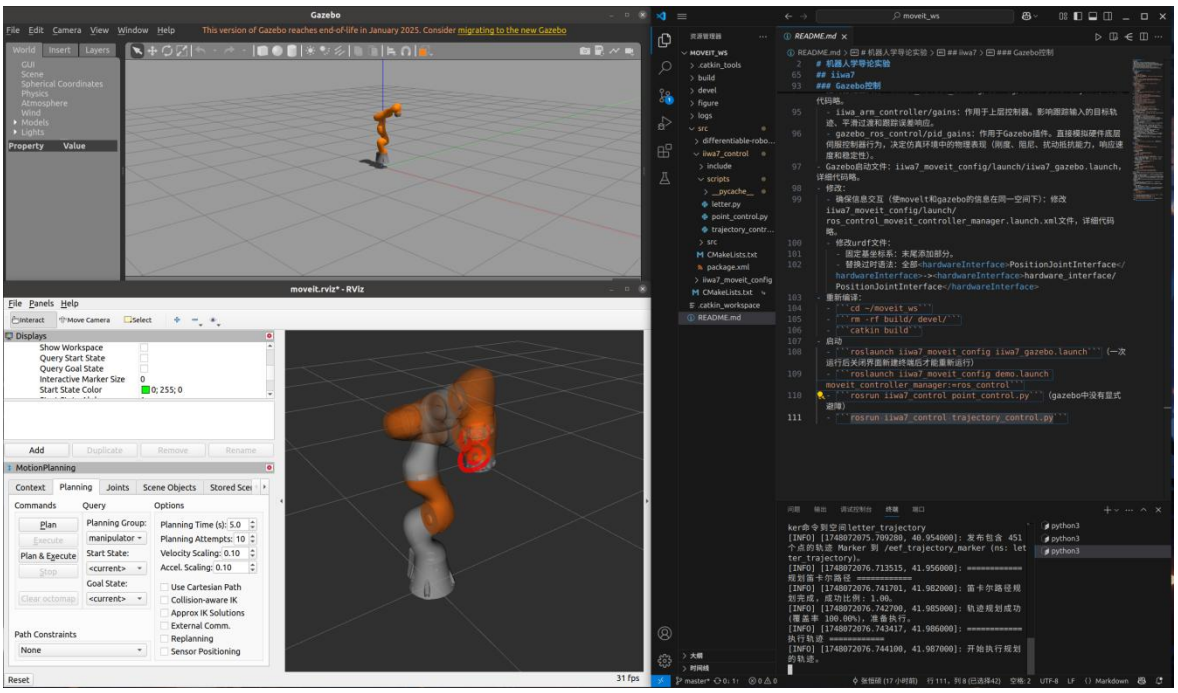
同上进行可视化操作。

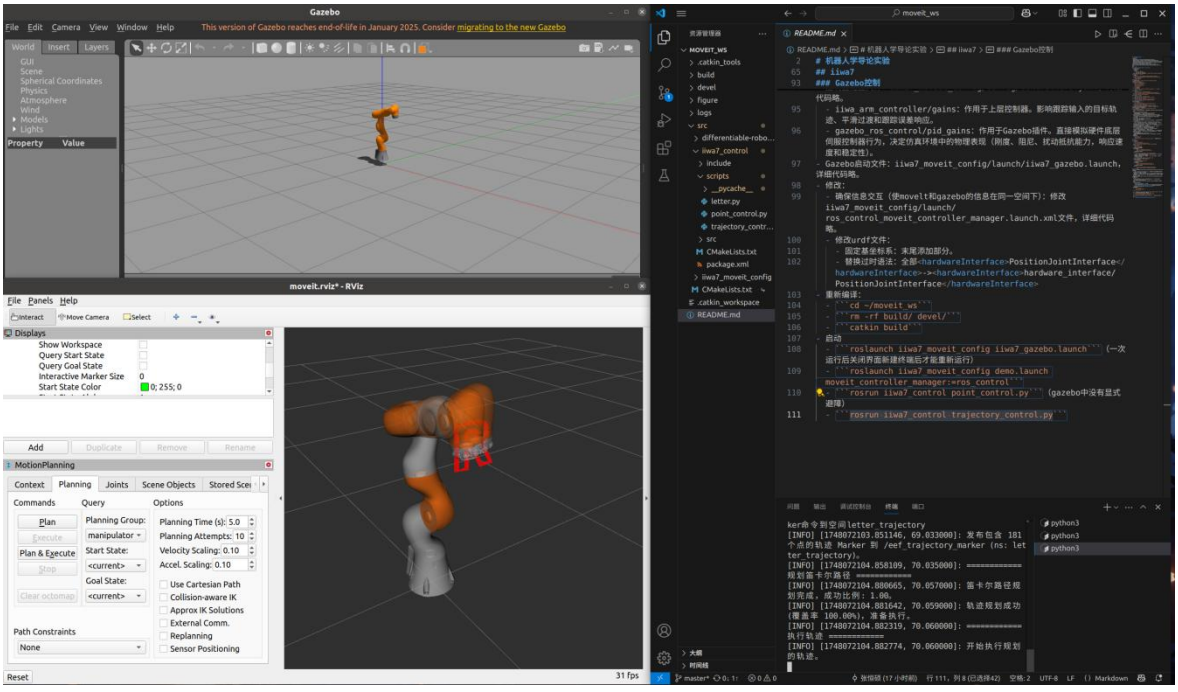
4.2.2 Gazebo 仿真

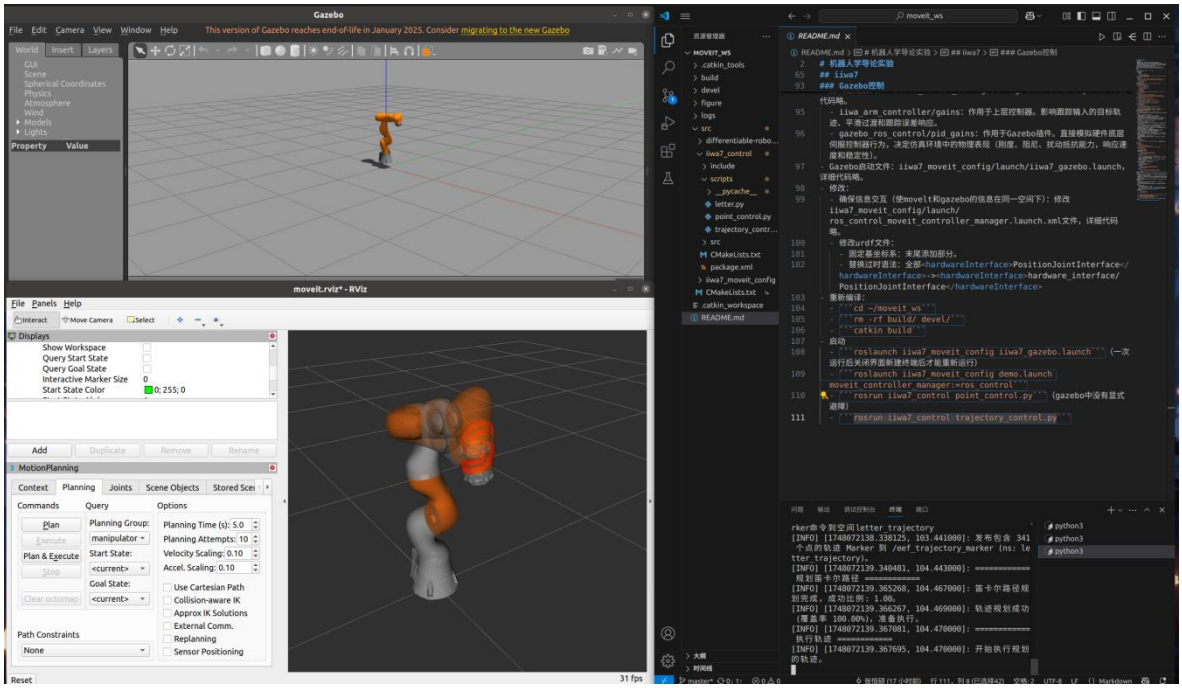
```
roslaunch iiwa7_moveit_config iiwa7_gazebo.launch
roslaunch iiwa7_moveit_config demo.launch moveit_controller_manager:=ros_control
rosrun iiwa7_control trajectory_control.py
```

4.2.3 仿真结果

以下依次展示了 8、R、S 的轨迹控制。


Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider migrating to the new Gazebo
moveit.rviz* - RViz
File Panels Help
Displays
MotionPlanning
Context Planning Joints Scene Objects Stored Scenes
Commands Query Options
Plan
Execute
Plan & Execute
Stop
Clear octomap
Planning Group: manipulator
Start State: <current>
Goal State: <current>
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 0.10
Accel. Scaling: 0.10
Use Cartesian Path
Collision-aware IK
Approx IK Solutions
External Comm.
Replanning
Sensor Positioning
Path Constraints
None
Reset
31 fps
README.md
moveit_ws
MOVEIT_WS
catkin_tools
build
devel
figure
logs
src
iiwa7_control
include
scripts
letter.py
point_control.py
trajectory_contr...
CMakeLists.txt
package.xml
iiwa7_moveit_config
.catkin_workspace
README.md
机器人学导论实验
iiwa7
Gazebo控制
python3
python3
python3


Gazebo
File Edit Camera View Window Help
This version of Gazebo reaches end-of-life in January 2025. Consider migrating to the new Gazebo
moveit.rviz* - RViz
File Panels Help
Displays
MotionPlanning
Context Planning Joints Scene Objects Stored Scenes
Commands Query Options
Plan
Execute
Plan & Execute
Stop
Clear octomap
Planning Group: manipulator
Start State: <current>
Goal State: <current>
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 0.10
Accel. Scaling: 0.10
Use Cartesian Path
Collision-aware IK
Approx IK Solutions
External Comm.
Replanning
Sensor Positioning
Path Constraints
None
Reset
31 fps
README.md
moveit_ws
MOVEIT_WS
catkin_tools
build
devel
figure
logs
src
iiwa7_control
include
scripts
letter.py
point_control.py
trajectory_contr...
CMakeLists.txt
package.xml
iiwa7_moveit_config
.catkin_workspace
README.md
机器人学导论实验
iiwa7
Gazebo控制
python3
python3
python3



第五章 总结

本实验基于 ROS1 Noetic 与 MoveIt 框架，成功实现了 KUKA iiwa7 机械臂的运动规划与控制功能，完成了点位控制与轨迹控制的核心目标。

主要成果：

- 环境搭建：完成了 Ubuntu 20.04、ROS1 Noetic、MoveIt、RViz 及 Gazebo 的集成配置，构建了稳定的机器人仿真与开发平台。
- 机械臂建模：通过 MoveIt Setup Assistant 生成 iiwa7 机械臂的运动规划配置文件，支持 RViz 与 Gazebo 的联合仿真，验证了模型的准确性。
- 点位控制：实现了五角星轨迹的规划与执行，并在障碍物环境下验证了避障能力，证明了 MoveIt 的实时路径重规划功能。
- 轨迹控制：基于字体轮廓提取与贝塞尔曲线离散化方法，实现了任意字符的三维轨迹规划与跟踪，展示了机械臂的高精度末端控制能力。
- 仿真验证：通过 RViz 的可视化与 Gazebo 的物理仿真，直观展示了机械臂的运动过程，验证了控制算法的可行性与鲁棒性。

问题与改进方向：

- 环境兼容性：Ubuntu 20.04 与 ROS1 Noetic 的支持期限即将到期，未来需迁移到 ROS2 与 Ubuntu 22.04。
- 算法优化：当前轨迹规划依赖 MoveIt 的默认算法，未来可引入更高效的路径搜索算法（如 RRT*、PRM）或深度强化学习方法，提升复杂场景下的规划效率。

参考文献

张涛同学的库：https://github.com/Suixin04/ws_moveit.git（主要参考内容）
张禹豪同学的库：https://github.com/Ruochen0513/forABB_IRB_1410
ROS 官方安装教程：https://wiki.ros.org/noetic/Installation/Ubuntu
MoveIt 官方配置教程：
https://moveit.github.io/moveit_tutorials/doc/getting_started/getting_started.html
MoveIt Setup Assistant 教程：
https://moveit.github.io/moveit_tutorials/doc/setup_assistant/setup_assistant_tutorial.html