
# 基于ROS和MOVEIT的Marm机械臂绘制任务仿真


- 学号:2013365
- 姓名: 颜铭
- 专业:智能科学技术
- 指导老师：周璐

# 目录

# 1 实验条件

# 1 实验条件 EXPERIMENTAL CONDITION

## 实验主题

本次实验研究的是涵盖知识体系较为全面的关节与笛卡尔坐标系运动规划问题，通过位姿、速度和加速度的基础上推演到多项式限位控制并进行基本的软件仿真加深对关节连杆到空间物理量概念的熟悉

仿真基于ROS操作系统和开源社区的marm机械臂模型，通过moveit组件提供的Transfomer和 Pose Control等一系列API，结合节点代码和启动文件编程。实现了定点控制，绘制8字的功能

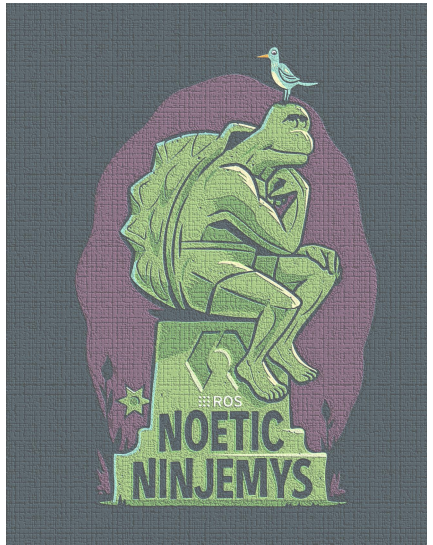
# 1 实验条件 EXPERIMENTAL CONDITION

实验仿真基于ubuntu 20.04 和ros noetic,使用的ros组件包括rviz和moveit

## ROS介绍

ROS（Robot Operating System）是一个开源的机器人操作系统框架，旨在简化机器人软件开发和复用。以下是对ROS的简要介绍：

ROS是一个灵活、分布式的操作系统，提供了一系列工具、库和约定，用于协调和管理机器人软件的各个组件。它提供了一个通用的通信框架，使不同模块和节点能够相互通信和协作。




MoveIt

MoveIt!是一个功能强大的机器人操作系统（Robot Operating System，ROS）软件包，专门用于机器人运动规划和控制

实验主要基于以下核心组件提供的功能:

·运动规划器（Motion Planner）：运动规划器负责计算机器人的运动轨迹，以在给定的规划场景中实现所需的任务。MoveIt!提供了多个运动规划器，包括基于采样的规划器（例如随机采样一致性算法RRT、RRT*）和基于优化的规划器（例如CHOMP、OMPL）等。

· 规划场景（Planning Scene）：规划场景定义了机器人和周围环境的几何形状和约束条件。它包括机器人模型、障碍物、目标位置等信息。规划场景提供了机器人规划所需的环境上下文，以便有效地进行路径规划和避障。

# 1 实验条件 EXPERIMENTAL CONDITION

## 实验环境搭建

1.安装ubuntu 20.04双系统与ros noetic（使用apt install）

2. 下载源码，地址如下
https://github.com/huchunxu/ros_exploring/tree/master/robot_marm
使用命令进行安装

```
git clone https://github.com/huchunxu/ros_exploring.git
```

3.使用命令安装机器人关节控制器，以及若干插件。

```
sudo apt-get install ros-noetic-ros-control ros-noetic-ros-controllers
```

继续安装moveit以及可视化插件

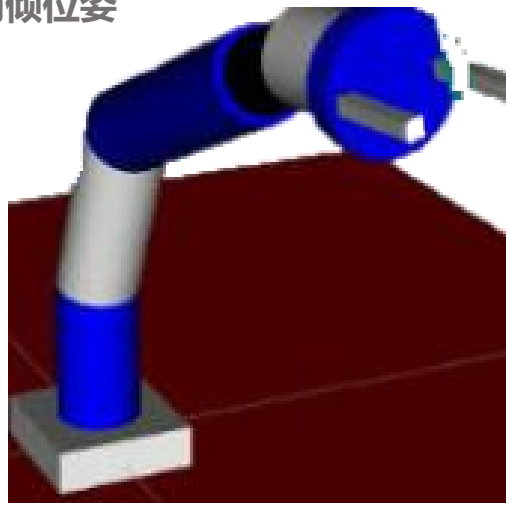
```
1. sudo apt-get install ros-noetic-moveit
2. sudo apt-get install ros-noetic-moveit
```

## Marm机器人

Marm机械臂是古月居社区开源的六关节五自由度机器人，其具体几何构型,如右图

**这是默认的’ Home’ 的直立位姿**

**这是用户定义的” forward”前倾位姿**



# 2 仿真设计

## 机器人规划原理

### 1 关节路径规划

机械臂关节路径规划算法旨在计算机控制下，确定机械臂关节角度的平滑运动轨迹。其中一种常用算法是样条插值方法，通过生成一系列中间关节角度点，使机械臂关节运动平滑过渡。公式表示为：$\theta(t) = a_0 + a_1 * t + a_2 * t^2 + a_3 * t^3$，其中θ为关节角度，t为路径参数，$a_{\{0\}}$，$a_{\{1\}}$，$a_{\{2\}}$，$a_{\{3\}}$为样条插值的系数。通过求解系数，可以满足起始关节角度、目标关节角度和插值点的约束条件，实现平滑的关节运动轨迹。该算法常用于机器人轨迹规划、运动控制和运动学求解等应用。

### 2 笛卡尔路径规划

笛卡尔路径规划是机器人运动控制中的一种方法，其目标是通过规划机器人末端执行器的路径来实现在笛卡尔坐标系中的精确位置和姿态。

基本原理是根据起始位置和目标位置之间的连续路径点，通过插值方法计算出机器人末端执行器的轨迹。假设起始位置为P1，目标位置为P2，路径上的中间点为Pi，路径长度为L。

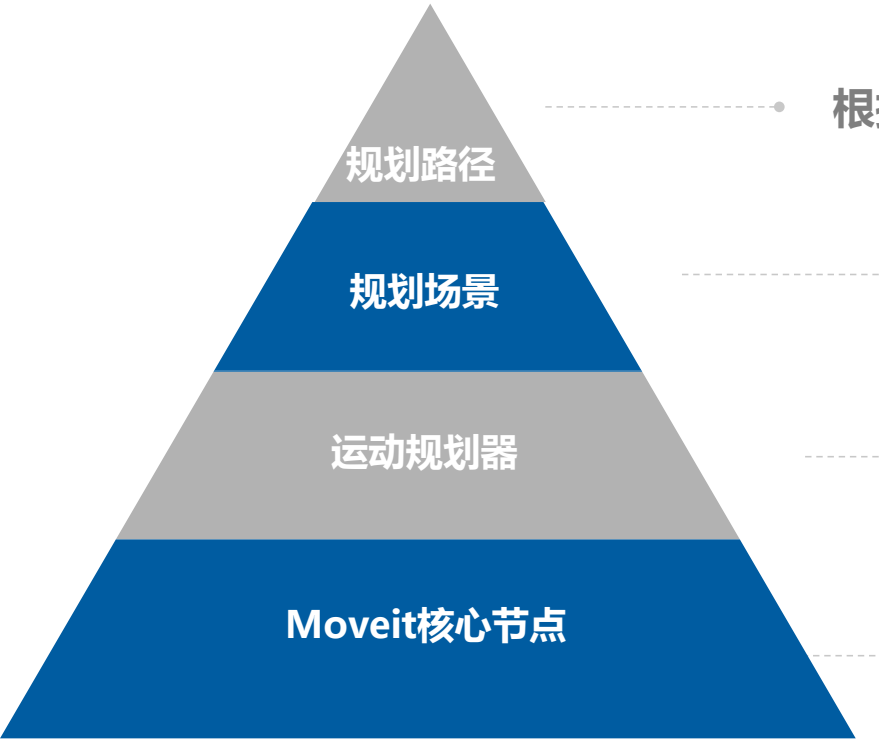
首先，可以使用插值方法（如直线插值、样条插值等）计算出路径上的连续点Pi，其中i从1到N。然后，使用逆运动学或优化算法计算机器人的关节角度或位置，以实现末端执行器的所需位置和姿态。

数学公式可以表示为：
$P(t) = (1 - t) * P_1 + t * P_2$，其中t为路径上的参数，范围从0到1。

通过调整插值步长和路径点密度，可以控制路径的平滑程度和精度。同时，还需要考虑机器人的运动学约束和碰撞检测，以确保生成的路径是可行和安全的。

笛卡尔路径规划在机器人控制、自动化生产和导航等领域有广泛应用，能够实现机器人在复杂环境中的精确运动控制。

## 机器人轨迹规划控制过程



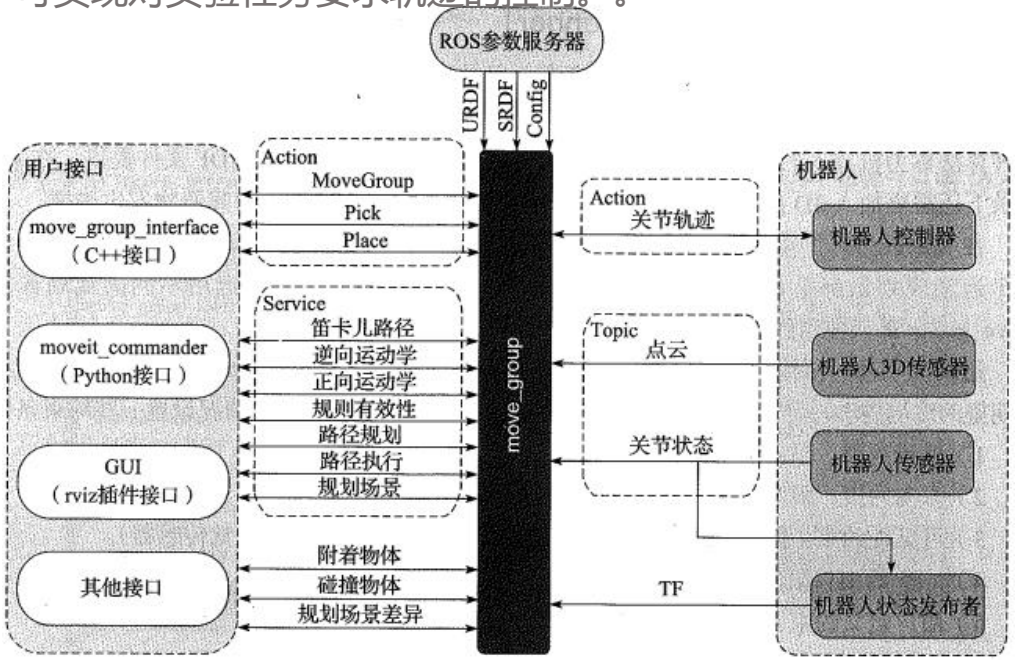

根据路径几何信息对应路点

考虑工作空间和坐标系选取

考虑避障和约束的物理引擎规划

ROS通讯，启动控制仿真

仿真实验初始化名为moveit_cartesiam_demo的节点，并且通过自定义的规划组和机器人关节位姿调用基于ROS动作服务的运动组指令单元。然后通过路点载入通讯需要的PoseStamped数据结构进行调包对接，路点指的是一个空间位姿列表，相邻两个路点之间使用的是直线轨迹运动，因此路径叠加后可以模拟曲线效果。因此，我们只需要对画8字任务执行两次连接圆圈的路径规划即可实现对实验任务要求轨迹的控制。。



# 3 仿真流程

# 3 仿真流程

SIMULATION PROCEDURE

**仿真代码**

①在设计中我们知到要使用urdf并且在rviz文件中显示，初步了解并验证机械臂模型的完备性

```
<!-- 加载机器人模型参数 -->
<param name="robot_description" command="$(find xacro)/xacro --inorder $(find marm_description)/urdf/

<!-- 设置GUI参数，显示关节控制插件 -->
<param name="use_gui" value="true"/>

<!-- 运行joint_state_publisher节点，发布机器人的关节状态 -->
<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />

<!-- 运行robot_state_publisher节点，发布tf -->
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />

<!-- 运行rviz可视化界面 -->
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find marm_description)/urdf.rviz" required="true"
```

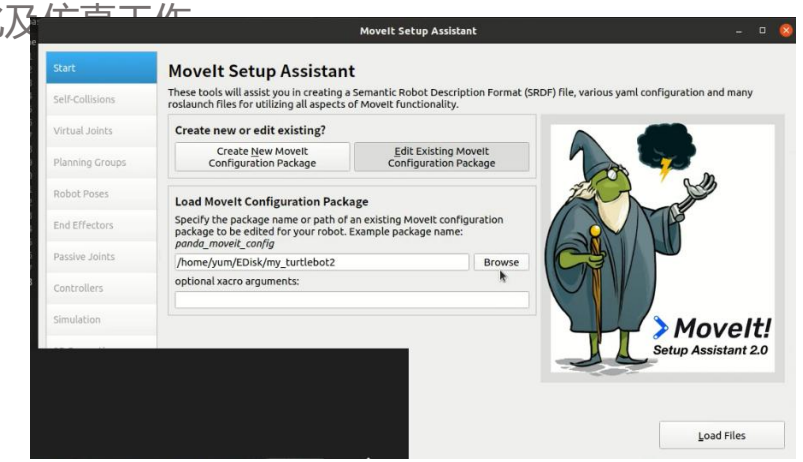
使用命令

1.roslaunch marm_description view_arm.launch

进行可视化仿真，在可视化界面通过拖动joint控制插件的滑动条，可以看到对应机器人的关节开始转动

②使用Setup Assistant配置机械臂

为了使用自己创建的机器人urdf模型，需要同前文所述进行一些配置工作，构建助手会根据用户导入的机器人URDF模型生成SRDF文件，从而创建一个MoveIt的配置功能包。完成机器人的配置与可视化及仿真工作




利用配置可视化界面自定义机器人位姿，位姿需要依据场景进行合理的自定义，点击添加姿态按钮，再选择对应的规划组如arm。可以将marm机器人直立的默认状态设置为机器人初始位姿‘ home’ 此外，我们可以自定义定点控制中的位姿‘ forward’ ，一样步骤保持拖动关节控制的滑动条，将右侧显示的机器人控制到指定的期望位姿。因此可以多定义几个目标点，再通过端到端的轨迹连线即可实现定点控制

# 3 仿真流程

SIMULATION PROCEDURE

仿真代码

③使用rospy的笛卡尔运动规划

一般来说，我们在工作空间的意义下使用需要对末端执行器进行轨迹约束，目标位置给定后，可以通过运动学反解获得关节空间下的各种弧度，接下来的规划和运动不同于关节空间完成的规划，我们要给出特定的机械臂起始和终止位姿，对运动过程的位姿也有要求。因此对末端执行器执行直线到圆弧轨迹求解需要使用笛卡尔运动规划。

使用命令

```
roslaunch marm_planning arm_planning_with_trail.launch
```

启动可视化笛卡尔绘制路径的例程

---

然后启动原始的轨迹设计规划器节点:

```
rosrun marm_planning moveit_cartesian_demo.py _catersian:=True
```

整个代码的核心部分是，使用了笛卡尔路径规划的API，compute_cartesian_path()，它需要四个参数，第一个参数之前创建过的路点列表，第二个参数是终端步进值，第三个参数是跳跃阈值，第四个参数是用于设置运动过程中是否考虑避障

函数执行后返回两个值，plan为规划出来的运动轨迹，fraction为用于描述规划成功的轨迹在给定的路点列表中的覆盖率。从0到1，如果fraction小于1，说明给定路点列表没办法完整规划，这种情况下可以重新进行规划，这时就需要人为进行示教给出规划次数的限制。

```
while fraction < 1.0 and attempts < maxtries:
    (plan, fraction) = self.arm.compute_cartesian_path (
                                waypoints,   # waypoint poses, 路点列表
                                0.01,        # eef_step, 终端步进值
                                0.0,         # jump_threshold, 跳跃阈值
                                True)        # avoid_collisions, 避障规划

    # 尝试次数累加
    attempts += 1

    # 打印运动规划进程
    if attempts % 10 == 0:
        rospy.loginfo("Still trying after " + str(attempts) + " attempts...")

# 如果路径规划成功（覆盖率100%），则开始控制机械臂运动
if fraction == 1.0:
    rospy.loginfo("First Circle Path computed successfully. Moving the arm.")
    self.arm.execute(plan)
    rospy.loginfo("First Circle Path execution complete.")
# 如果路径规划失败，则打印失败信息
else:
    rospy.loginfo("First Circle Path planning failed with only " + str(fraction) + " success
```

# 3 仿真流程




④轨迹设计代码:

综上，我们可以保留定点运动到forward位姿，且此时正视面为yz面。因此我们可以设计yz面的连续两次圆周运动

```
radius = 0.04  #半径
centerA = target_pose.pose.position.y
centerB = target_pose.pose.position.z-radius #注意yz平面为正视面，第一次要设置z的最高点

rospy.loginfo("the first pos center z is %f",centerB)

centerAt = centerA-2*radius  # 第二次在y方向的平移
centerBt = centerB

rate_a =1.0
rate_b =1.0
cnt =0
for th in np.arange(np.pi/2,3*np.pi, 0.02):
    target_pose.pose.position.y= centerA + rate_a*radius * np.cos(th)
    target_pose.pose.position.z = centerB + rate_b*radius * np.sin(th)
    wpose = deepcopy(target_pose.pose)
    if self.cartesian:
        waypoints.append(deepcopy(wpose))
    else:
        self.arm.set_pose_target(wpose)
        self.arm.go()
```

本质上就是注意为了提高规划成功率，一定不要把初始点设置为圆心或是在目标点和圆心连线间设置曲线缓冲点。接下来使用极坐标利用cos和sin函数即可实现两个圆圈的绘制，连起来即时一个八字，沿y轴方向绘制。

其中需要注意的是一般姿态控制尤其是state订阅器交换的消息是欧拉角，而PoseStamped进行规划的位姿信息使用的是四元数，因此需要eul2quat等工具进行转换。

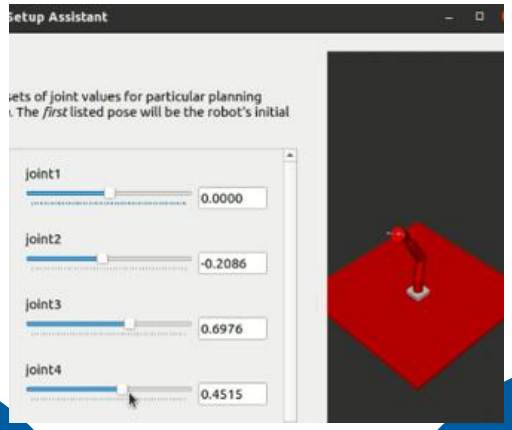
```
# 设置微调的缓冲路点数据，并加入路点列表
target_pose = deepcopy(start_pose)
target_pose.pose.position.x -= 0.02
target_pose.pose.position.y-=0.02
target_pose.pose.position.z+=0.04
self.arm.set_pose_target(target_pose)
self.arm.go()
rospy.sleep(1)
# target_pose.pose.orientation.x = -0.482974
# target_pose.pose.orientation.y = 0.517043
# target_pose.pose.orientation.z = -0.504953
# target_pose.pose.orientation.w = -0.494393
```

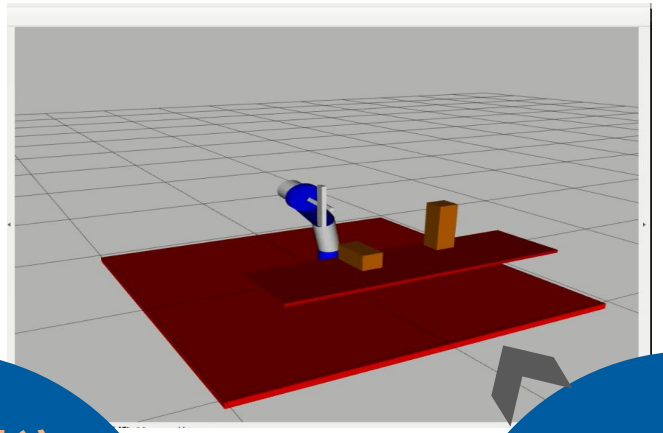
# 4 仿真结果

# 4 仿真结果

SIMUIATION AND RESULT

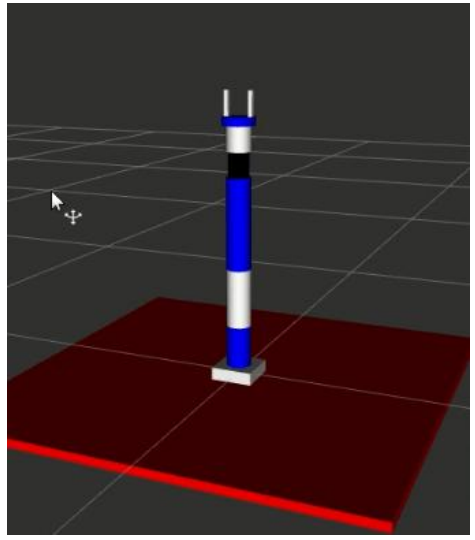
机械臂可视化



拓展:机械臂抓取和放置

01

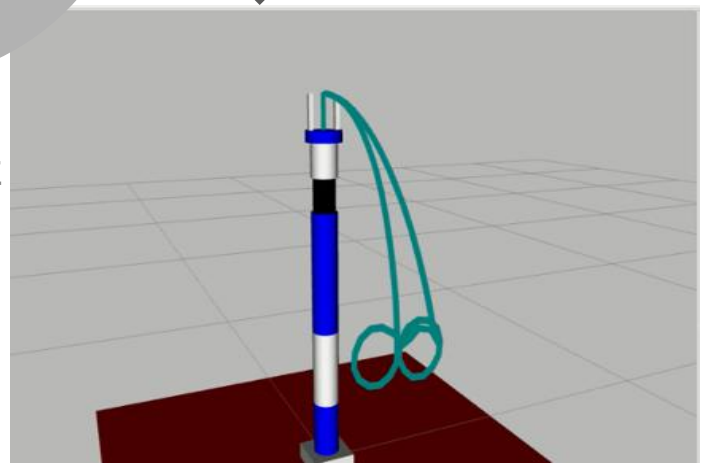


02

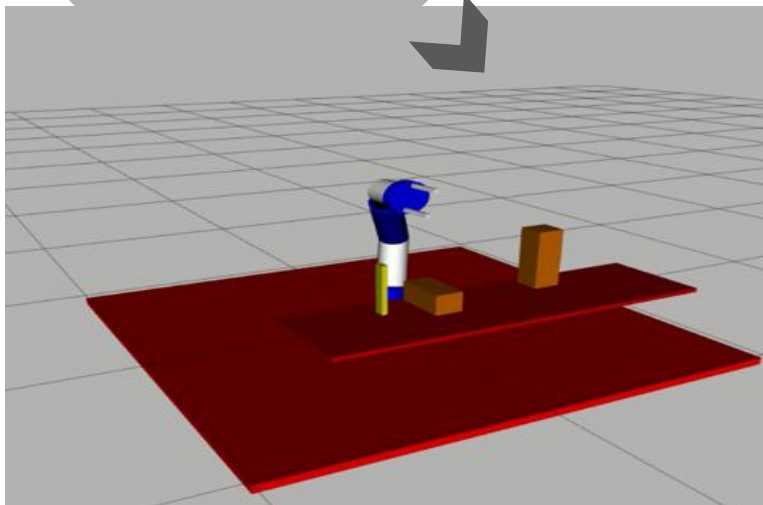
MoveIt可视化配置

绘制八字形轨迹

03



04



# 5 总结分析

# 5 总结分析

SUMARY AND ANALYSIS

## 结果分析

本实验将机器人学的轨迹规划以及正逆运动学等理论联系实践，利用最流行的机器人仿真与控制开发生态ROS实现了合理的求解。在整个实验过程中，接触到了四元数，欧拉角，旋转矩阵以及平移矩阵和传动等概念，更巩固了笛卡尔坐标系关节运动的有关知识。在这一过程中，我们理解了位姿规划建模到配置再到编程的基本流程，也明确了通过动力学设置抓取目标的基本原理。整体较为完整和稳定地完成了8字形规划。值得注意的是，在多次尝试中如果将8字形改为沿着z轴会多次规划失败，而且在第一个圆轨迹绘制中不确定度较高，也就尝试了更多的规划次数，可能是由于连接目标点的精度问题，可以采用关节坐标系规划进行平滑并有待进一步实验结果验证。

## 实验总结与展望

本次实验由于拓展不足，没有足够探索外开源项目的内容。预期的是在已经定义的末端抓取执行器的运动组中执行受力分析，模拟带抓握动力学的加速度规划，并通过不同受力的变色实现更好的可视化效果。

本实验进行过程存在许多不足之处，但是通过这次实践过程，我较为具体地熟悉了ROS编程控制机械臂轨迹以及基于理论到设计的实践，对未来进一步的学习与工作研究有所帮助。


# 感谢聆听