【ROS机械臂入门教程】
第6讲 Moveit基础(python)

小五

日期　2023/1/14

目录

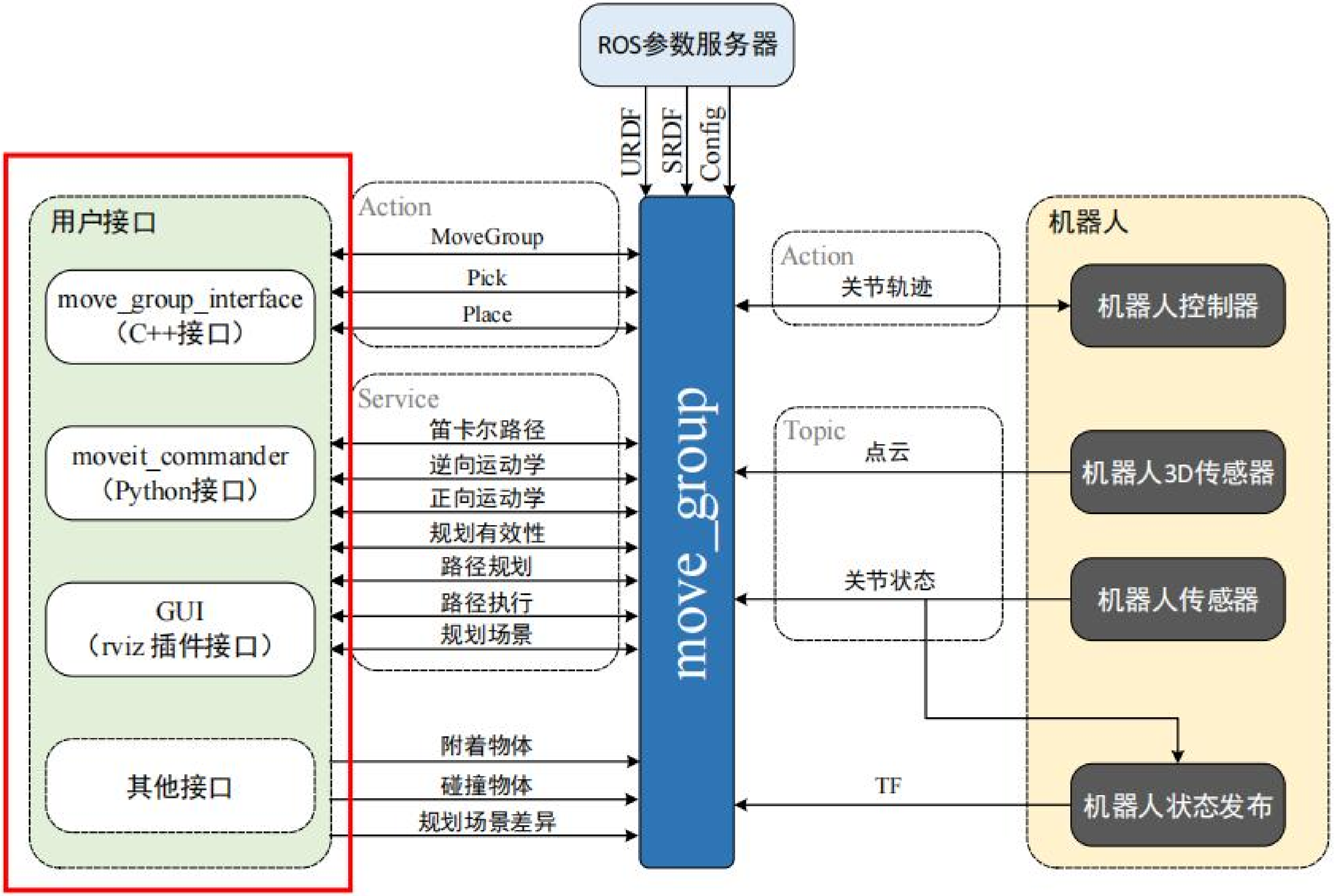
「1」 Moveit!回顾

「2」 关节空间运动

「3」 笛卡尔空间运动

「4」 与环境交互

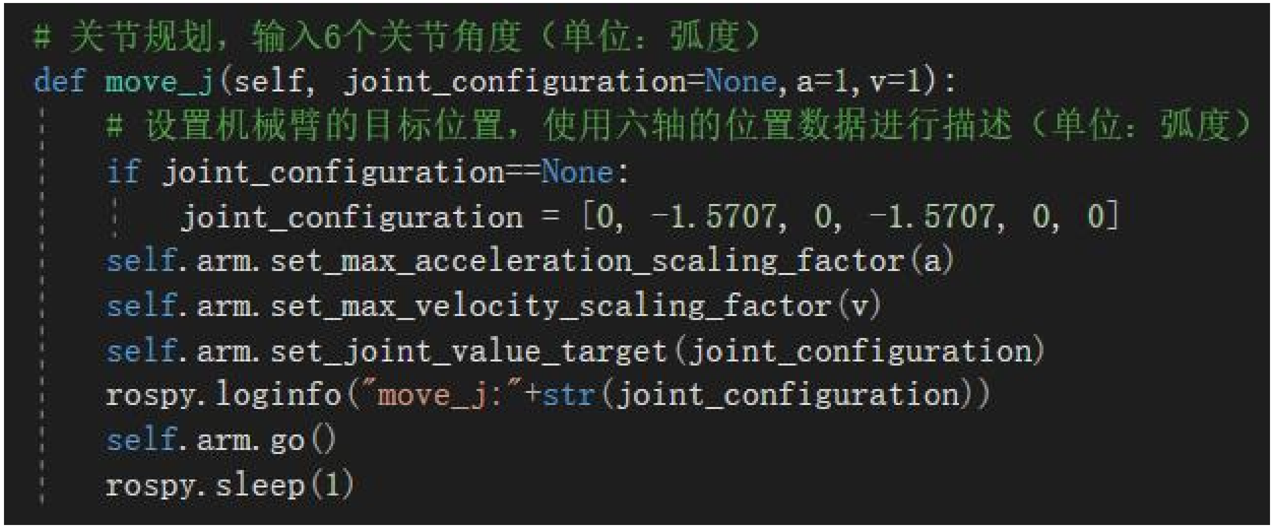
■ 用户接口



MoveIt!的核心节点——move_group

2 关节空间运动

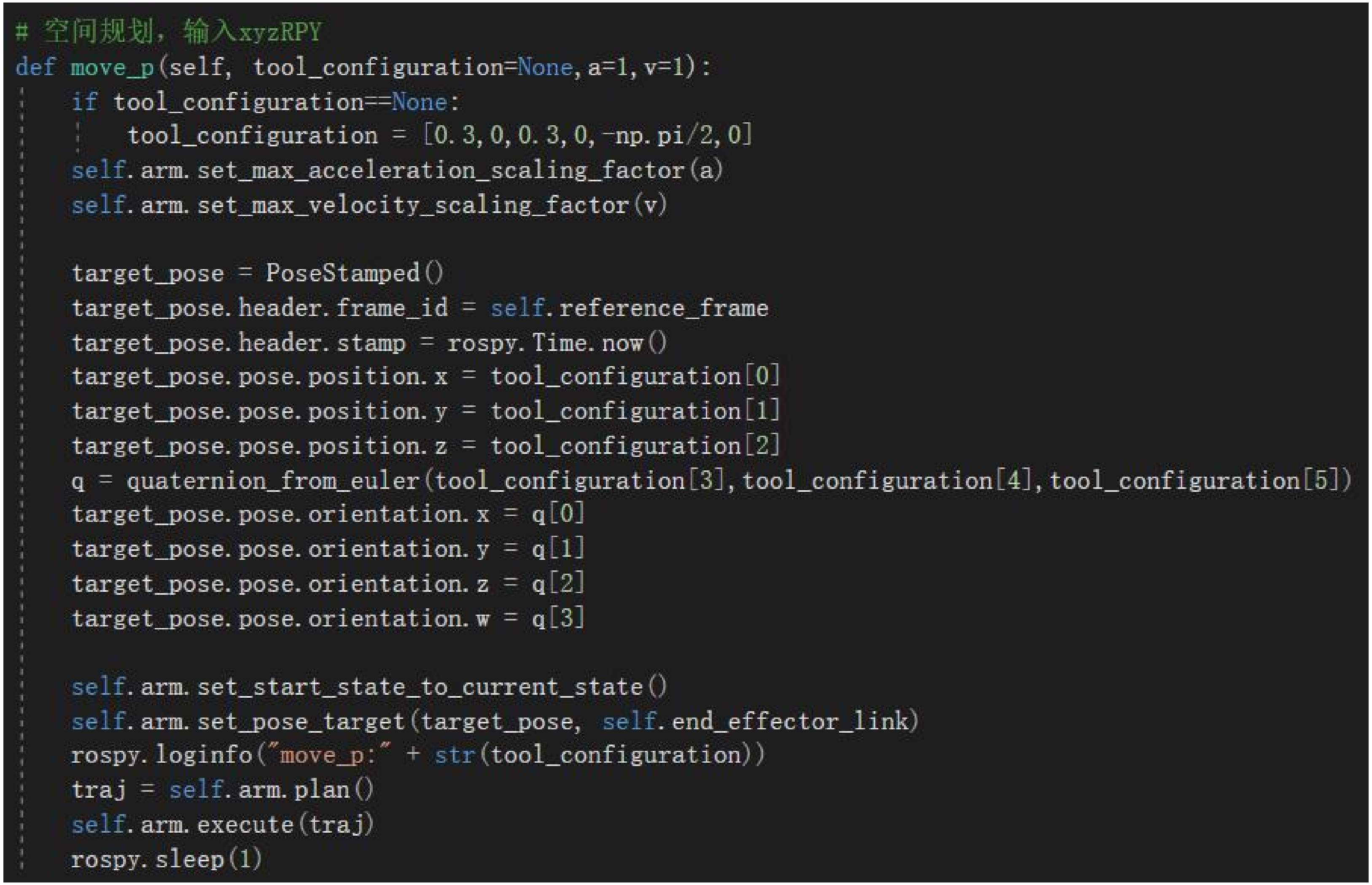
■ move_j



```
# 关节规划，输入6个关节角度（单位：弧度）
def move_j(self, joint_configuration=None,a=1,v=1):
    # 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）
    if joint_configuration==None:
        joint_configuration = [0, -1.5707, 0, -1.5707, 0, 0]
    self.arm.set_max_acceleration_scaling_factor(a)
    self.arm.set_max_velocity_scaling_factor(v)
    self.arm.set_joint_value_target(joint_configuration)
    rospy.loginfo("move_j:"+str(joint_configuration))
    self.arm.go()
    rospy.sleep(1)
```

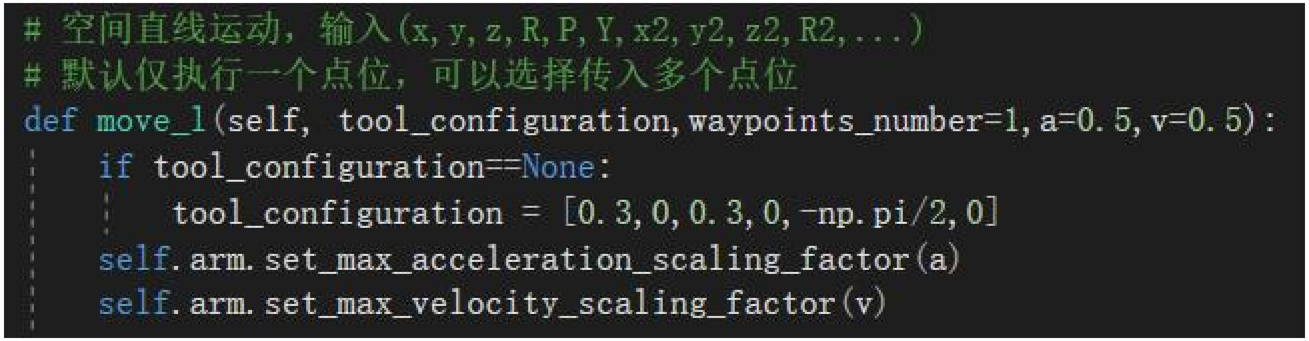
2 关节空间运动

■ move_p

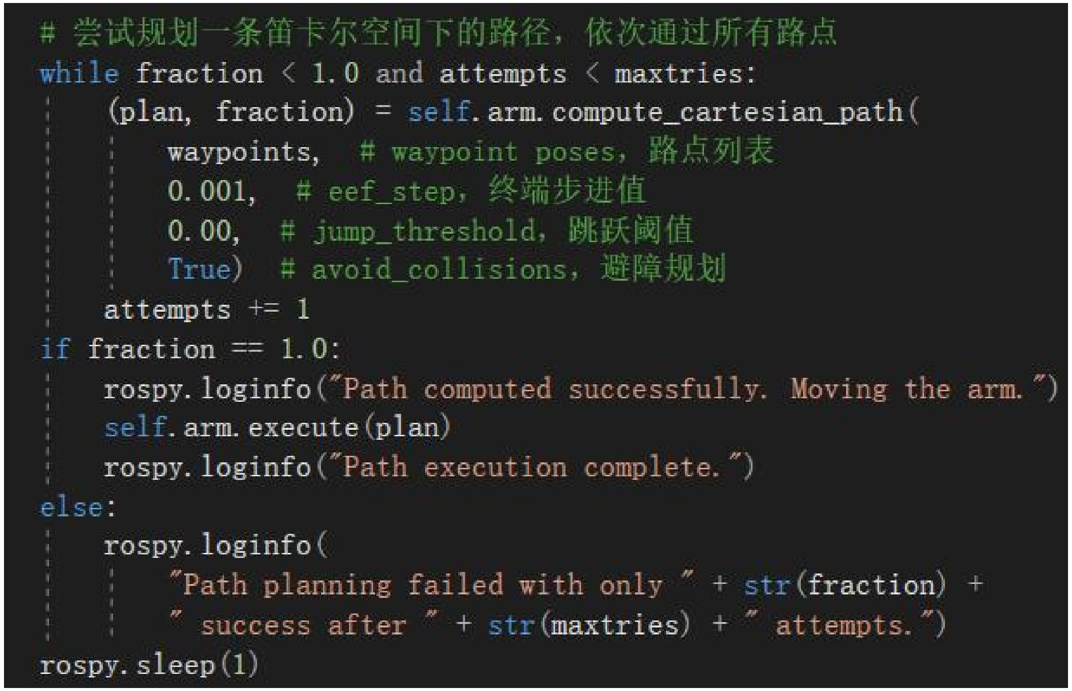


```
# 空间规划，输入xyzRPY
def move_p(self, tool_configuration=None,a=1,v=1):
    if tool_configuration==None:
        tool_configuration = [0.3,0,0.3,0,-np.pi/2,0]
    self.arm.set_max_acceleration_scaling_factor(a)
    self.arm.set_max_velocity_scaling_factor(v)

    target_pose = PoseStamped()
    target_pose.header.frame_id = self.reference_frame
    target_pose.header.stamp = rospy.Time.now()
    target_pose.pose.position.x = tool_configuration[0]
    target_pose.pose.position.y = tool_configuration[1]
    target_pose.pose.position.z = tool_configuration[2]
    q = quaternion_from_euler(tool_configuration[3],tool_configuration[4],tool_configuration[5])
    target_pose.pose.orientation.x = q[0]
    target_pose.pose.orientation.y = q[1]
    target_pose.pose.orientation.z = q[2]
    target_pose.pose.orientation.w = q[3]

    self.arm.set_start_state_to_current_state()
    self.arm.set_pose_target(target_pose, self.end_effector_link)
    rospy.loginfo("move_p:" + str(tool_configuration))
    traj = self.arm.plan()
    self.arm.execute(traj)
    rospy.sleep(1)
```

■ 直线运动

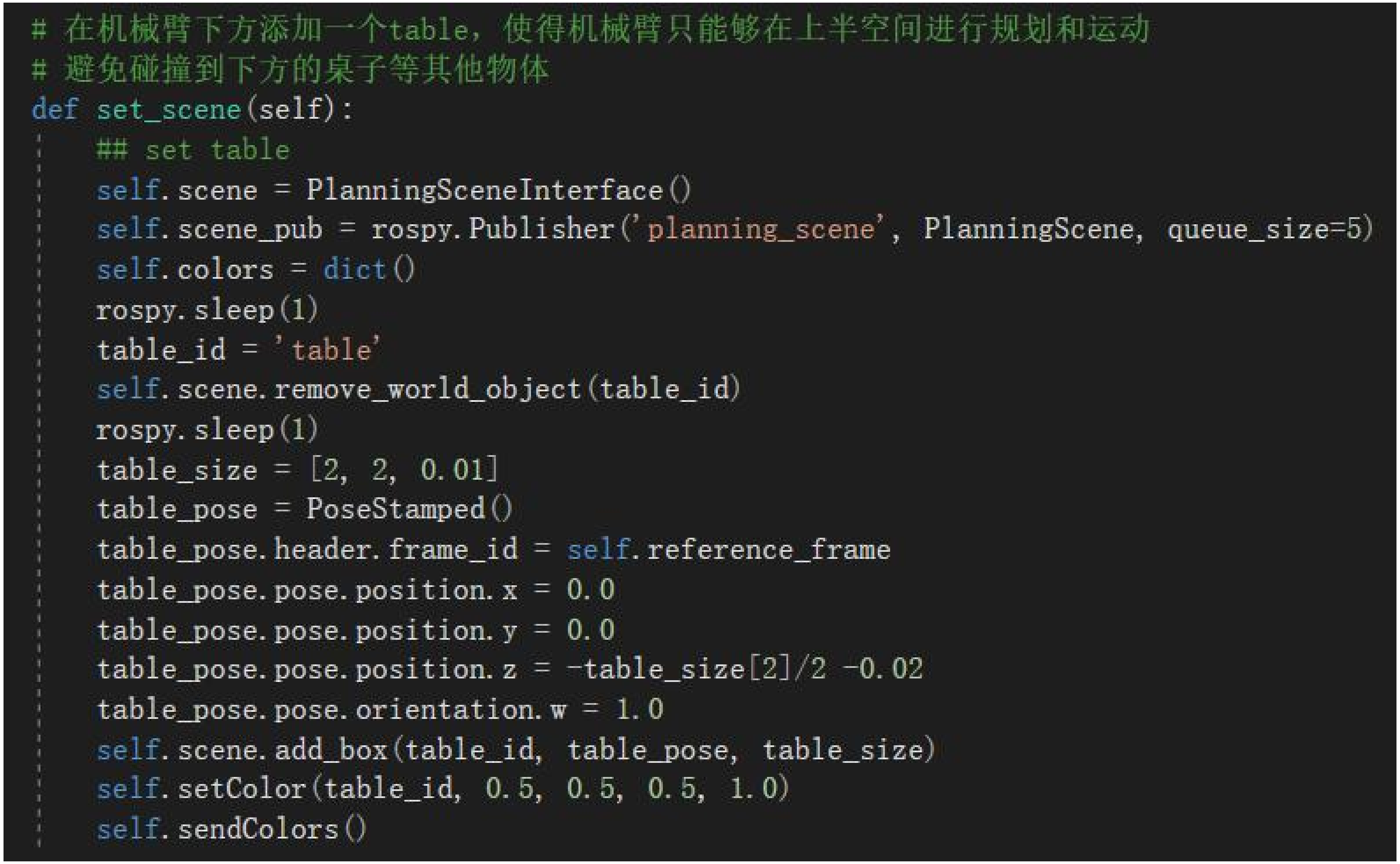


```
# 空间直线运动，输入(x,y,z,R,P,Y,x2,y2,z2,R2,...)
# 默认仅执行一个点位，可以选择传入多个点位
def move_l(self, tool_configuration,waypoints_number=1,a=0.5,v=0.5):
    if tool_configuration==None:
        tool_configuration = [0.3,0,0.3,0,-np.pi/2,0]
    self.arm.set_max_acceleration_scaling_factor(a)
    self.arm.set_max_velocity_scaling_factor(v)
```



```
# 尝试规划一条笛卡尔空间下的路径，依次通过所有路点
while fraction < 1.0 and attempts < maxtries:
    (plan, fraction) = self.arm.compute_cartesian_path(
        waypoints,  # waypoint poses，路点列表
        0.001,  # eef_step，终端步进值
        0.00,  # jump_threshold，跳跃阈值
        True)  # avoid_collisions，避障规划
    attempts += 1
if fraction == 1.0:
    rospy.loginfo("Path computed successfully. Moving the arm.")
    self.arm.execute(plan)
    rospy.loginfo("Path execution complete.")
else:
    rospy.loginfo(
        "Path planning failed with only " + str(fraction) +
        " success after " + str(maxtries) + " attempts.")
rospy.sleep(1)
```

■ 添加障碍物



```python
# 在机械臂下方添加一个table，使得机械臂只能够在上半空间进行规划和运动
# 避免碰撞到下方的桌子等其他物体
def set_scene(self):
    ## set table
    self.scene = PlanningSceneInterface()
    self.scene_pub = rospy.Publisher('planning_scene', PlanningScene, queue_size=5)
    self.colors = dict()
    rospy.sleep(1)
    table_id = 'table'
    self.scene.remove_world_object(table_id)
    rospy.sleep(1)
    table_size = [2, 2, 0.01]
    table_pose = PoseStamped()
    table_pose.header.frame_id = self.reference_frame
    table_pose.pose.position.x = 0.0
    table_pose.pose.position.y = 0.0
    table_pose.pose.position.z = -table_size[2]/2 -0.02
    table_pose.pose.orientation.w = 1.0
    self.scene.add_box(table_id, table_pose, table_size)
    self.setColor(table_id, 0.5, 0.5, 0.5, 1.0)
    self.sendColors()
```

教程视频会持续更新

敬请期待！