MoveIt positive kinematics design

Preface

We have built MoveIt environments on both the Jetson_Nano motherboard and the Orin series motherboard. Due to the onboard performance of the Jetson_Nano, running the MoveIt program on the motherboard will be more laggy and slow to load, and it will take about 3 minutes to complete the loading. Therefore, we recommend that users of the Jetson motherboard run the MoveIt program on the configured virtual machine we provide. The Orin motherboard can run MoveIt smoothly on the motherboard without running it on a virtual machine. Whether running on a virtual machine or on the motherboard, the startup instructions are the same. The following tutorials will take running on the Orin motherboard as an example to explain

1. Functional Description

After the program runs, the simulated robotic arm in Rviz will move to the target posture set by the program.

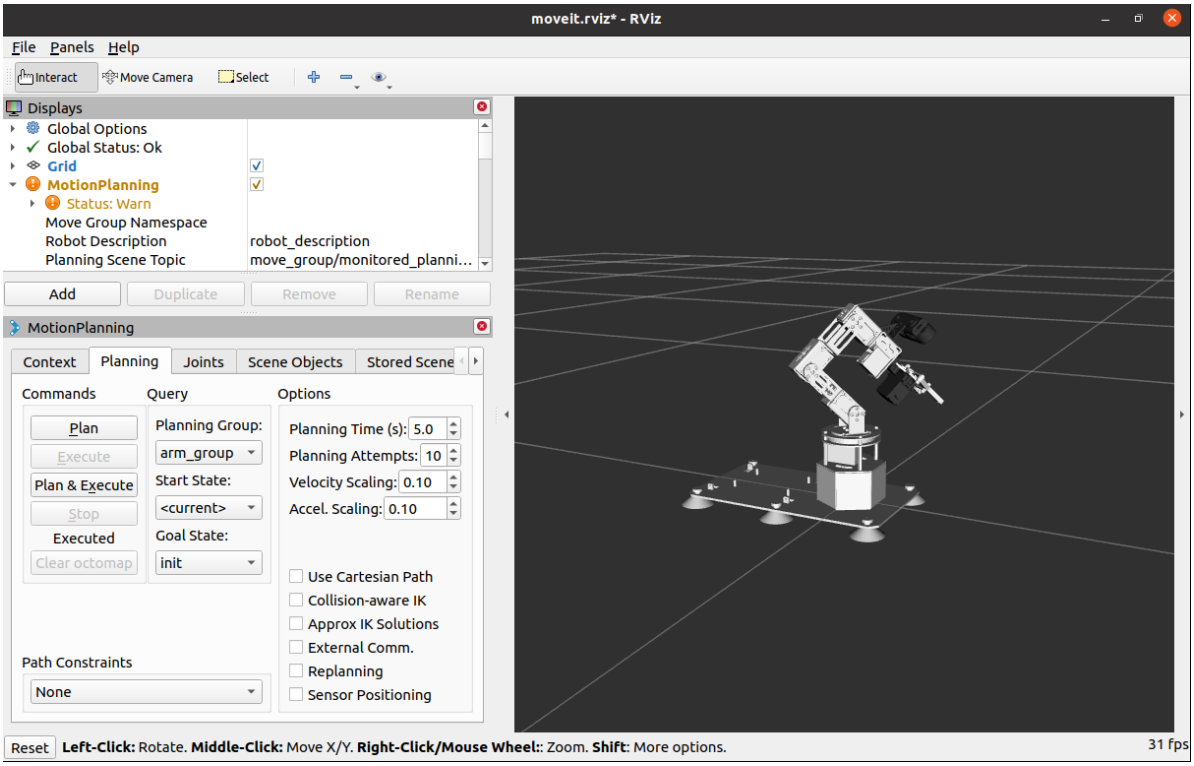
2. Start

Enter the following command in the terminal to start the program,

```
#Start MoveIt
roslaunch dofbot_pro_config demo.launch
```

After MoveIt is successfully started, enter in the terminal,

```
#Start the positive kinematics node program (choose one of the following commands to start)
#Python
rosrun arm_moveit_demo 03_set_joint_plan.py
#C++
rosrun arm_moveit_demo 03_set_joint_plan
```



3. Core code analysis

Take Python code as an example, the code path:

`/home/jetson/dofbot_pro_ws/src/arm_moveit_demo/scripts/03_set_joint_plan.py`

```
#!/usr/bin/env python
# coding: utf-8
import rospy
from math import pi
from time import sleep
from moveit_commander.move_group import MoveGroupCommander

# Convert angle to radians
DE2RA = pi / 180

if __name__ == '__main__':
    # Initialize the node
    rospy.init_node("set_joint_py", anonymous=True)
    # Initialize the robot arm
    dofbot_pro = MoveGroupCommander("arm_group")
    # Allow replanning after motion planning fails
    dofbot_pro.allow_replanning(True)
    dofbot_pro.set_planning_time(5)
    # Number of planning attempts
    dofbot_pro.set_num_planning_attempts(10)
    # Set the target angle tolerance
    dofbot_pro.set_goal_joint_tolerance(0.001)
    # Set the maximum allowed velocity and acceleration
    dofbot_pro.set_max_velocity_scaling_factor(1.0)
    dofbot_pro.set_max_acceleration_scaling_factor(1.0)
    # Set "down" as the target point
    dofbot_pro.set_named_target("down")
    dofbot_pro.go()
    sleep(0.5)
    # Set target point radians
```

```
    joints = [0.79, 0.79, -1.57, -1.57, 0]
    dofbot_pro.set_joint_value_target(joints)
    # Execute multiple times to improve the success rate
    for i in range(5):
        # Motion Planning
        plan = dofbot_pro.plan()
        #print("plan: ",plan)
        if len(plan.joint_trajectory.points) != 0:
            print ("plan success")
            # Run after successful planning
            dofbot_pro.execute(plan)
            break
        else:
            print ("plan error")
```