# Scenario 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 slow 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.

## 1. Functional Description

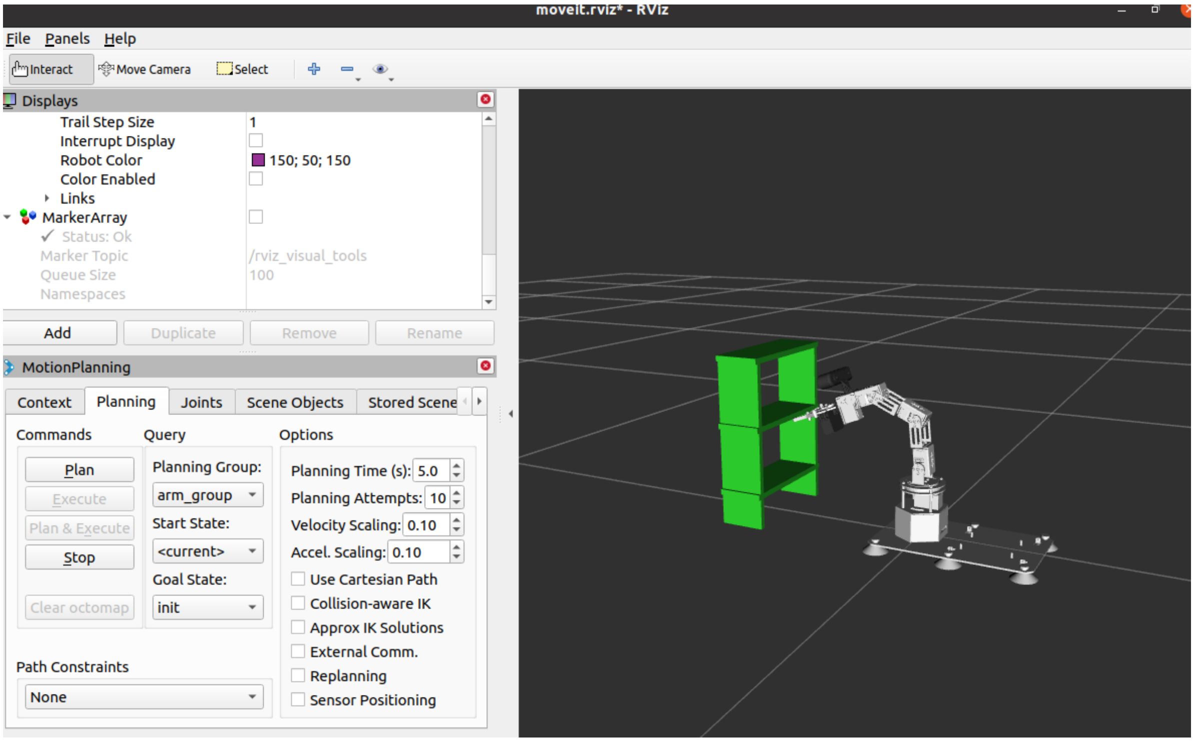
After the program is started, a scene will be added to rviz: the robot arm grabs a block and moves back and forth on a three-layer cabinet.

## 2. Start

First, start MoveIt. Enter the following command in the terminal to start it.

```
roslaunch dofbot_pro_config demo.launch
```

After MoveIt is successfully started, you need to add the scene. Click [Scene Objects]-[Import] and select the saved scene file. The file is located in /home/jetson/dofbot_pro_ws/src/arm_moveit_demo/scene/shape.scene. After importing, it is shown as follows.



Trail Step Size 1
Interrupt Display □
Robot Color 150; 50; 150
Color Enabled □
Links
MarkerArray □
✓ Status: Ok
Marker Topic /rviz_visual_tools
Queue Size 100
Namespaces

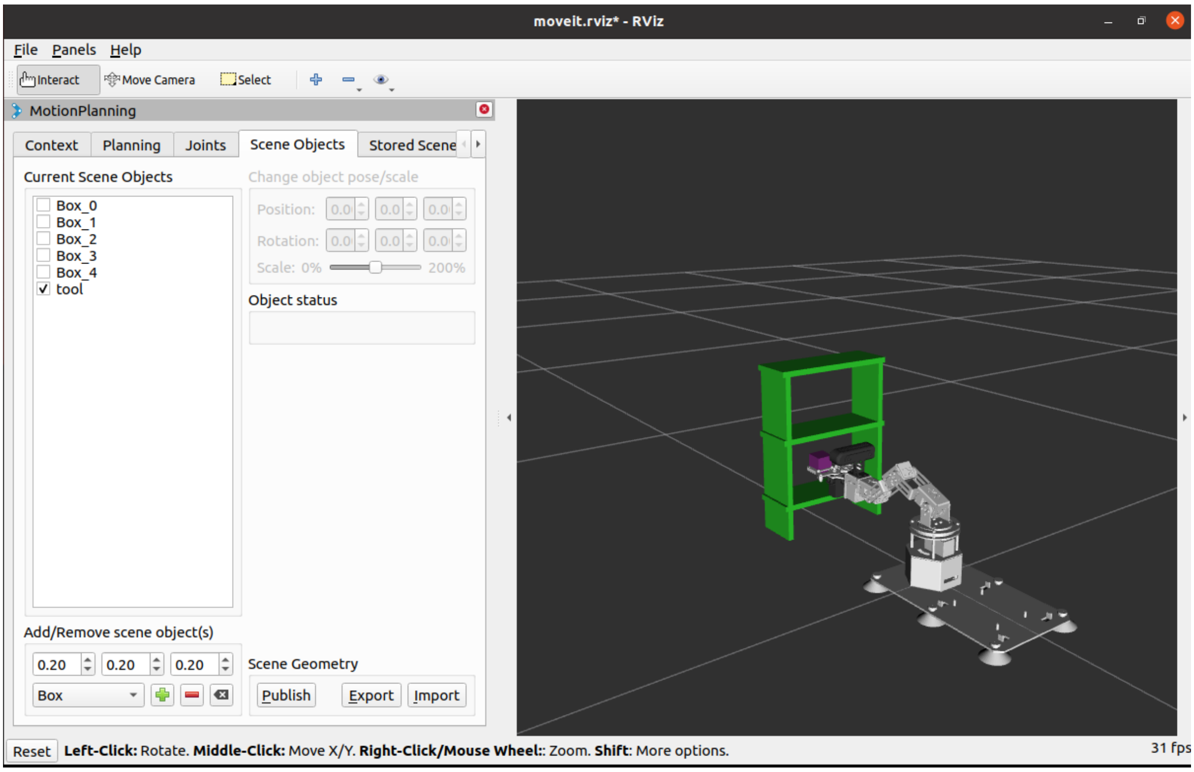
Planning Group: arm_group
Start State: <current>
Goal State: init
Planning Time (s): 5.0
Planning Attempts: 10
Velocity Scaling: 0.10
Accel. Scaling: 0.10
□ Use Cartesian Path
□ Collision-aware IK
□ Approx IK Solutions
□ External Comm.
□ Replanning
□ Sensor Positioning
Path Constraints
None

Click [Publish] to publish the scene information. In the subsequent clamping block placement program, the robot arm will consider the scene information when planning the path to avoid obstacles.

The gripper's posture needs to be set to "open" to avoid the overlap of the gripper and the block, which will cause the program to not run normally. Select [Planning]->[Planning Group], set it to gripper_group, and then select "open" in [Goal State]; click [Plan&Execute] in [Commands] to plan and execute the program.

Enter the following command to start the scene design program,

```
rosrun arm_moveit_demo 06_set_Scene.py
```



□ Box_0
□ Box_1
□ Box_2
□ Box_3
□ Box_4
☑ tool

0.20 0.20 0.20

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

Close the case: [ctrl+c] to close, if it cannot be closed, execute [ctrl+z].

## 3. Core code analysis

Code path:

`/home/jetson/dofbot_pro_ws/src/arm_moveit_demo/scripts/06_set_Scene.py`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from math import pi

import rospy, sys
from time import sleep
import moveit_commander
from geometry_msgs.msg import PoseStamped
from moveit_commander import MoveGroupCommander, PlanningSceneInterface

if __name__ == "__main__":
    moveit_commander.roscpp_initialize(sys.argv)
    rospy.init_node('Set_Scene')
    # Initialize the scene object
    # 初始化场景对象
    scene = PlanningSceneInterface()
    # Initialize the robotic arm motion planning group
    # 初始化机械臂运动规划组
    dofbot = MoveGroupCommander("arm_group")
    # Allow replanning when motion planning fails
    # 当运动规划失败后，允许重新规划
    dofbot.allow_replanning(True)
    dofbot.set_planning_time(5)
    # number of attempts to plan
    # 尝试规划的次数
    dofbot.set_num_planning_attempts(10)
    # Set allowable target position error
    # 设置允许目标位置误差
    dofbot.set_goal_position_tolerance(0.01)
    # Set the allowable target attitude error
    # 设置允许目标姿态误差
    dofbot.set_goal_orientation_tolerance(0.01)
    # Set allowable target error
    # 设置允许目标误差
    dofbot.set_goal_tolerance(0.01)
    # set maximum speed
    # 设置最大速度
    dofbot.set_max_velocity_scaling_factor(1.0)
    # set maximum acceleration
    # 设置最大加速度
    dofbot.set_max_acceleration_scaling_factor(1.0)
    # Set "up" as the target point
    # 设置"up"为目标点
    dofbot.set_named_target("up")
    dofbot.go()
    sleep(0.5)
    target_joints1 = [0, -1.18, -1.17, 0.77, 0.03]
    target_joints2 = [0, -1.21, 0.52, -0.89, 0.08]
    tool_size = [0.03, 0.03, 0.03]
    # Get the name of the terminal link
    # 获取终端link的名称
    end_effector_link = dofbot.get_end_effector_link()
    # Set the pose of the tool
    # 设置tool的位姿
    p = PoseStamped()
    p.header.frame_id = end_effector_link
    p.pose.position.x = 0
    p.pose.position.y = 0.09
    p.pose.position.z = 0.0
    p.pose.orientation.x = 0
    p.pose.orientation.y = 0
    p.pose.orientation.z = 0
    p.pose.orientation.w = 1
    # Attach the tool to the gripper of the robotic arm
    # 将tool附着到机械臂的夹爪上
    scene.attach_box(end_effector_link, 'tool', p, tool_size)
    while 1:
        dofbot.set_joint_value_target(target_joints1)
        dofbot.go()
        sleep(0.5)
        dofbot.set_joint_value_target(target_joints2)
        dofbot.go()
        sleep(0.5)
```