8. MoveIt trajectory planning

8. MoveIt trajectory planning

- 8.1. Start
- 8.2. Source code analysis

This lesson takes MoveIT simulation as an example. If you need to set up the real machine and simulation to be synchronized, please see the lesson [02, MoveIt Precautions and Controlling the Real Machine]. ! ! ! be safe! ! !

The effect demonstration is a virtual machine and other main control running conditions (related to the main control performance, depending on the actual situation).

8.1. Start

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

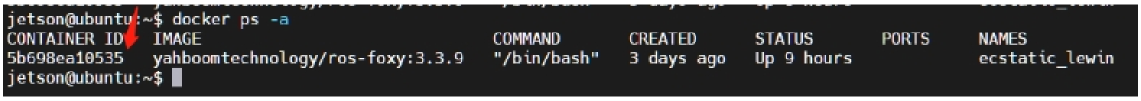
Start MoveIT

```
roslaunch arm_moveit_demo x3plus_moveit_demo.launch sim:=true
```

<PI5 needs to open another terminal to enter the same docker container

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

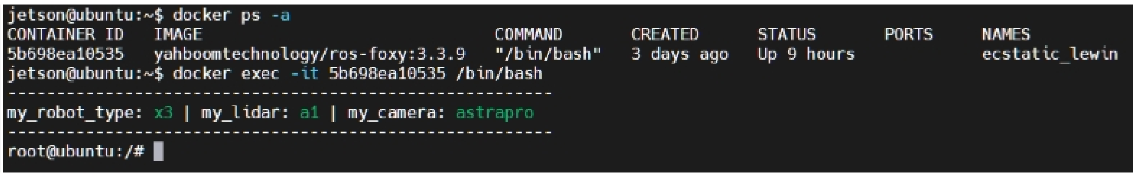
```
docker ps -a
```



```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                 COMMAND       CREATED       STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9      "/bin/bash"   3 days ago    Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```



```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                 COMMAND       CREATED       STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9      "/bin/bash"   3 days ago    Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
-------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
-------------------------------------------------------
root@ubuntu:/#
```

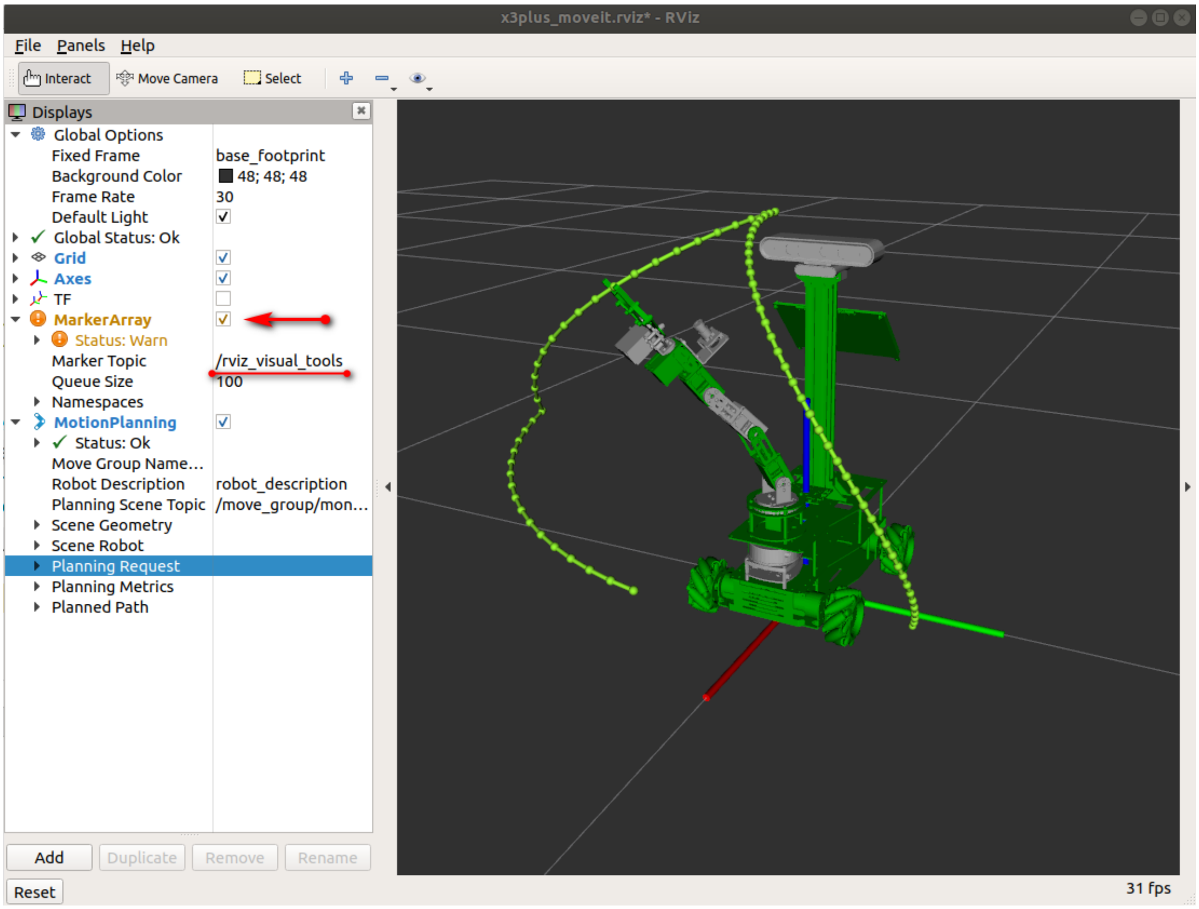
After successfully entering the container, you can open countless terminals to enter the container.

Start trajectory planning node

```
rosrun arm_moveit_demo 06_multi_track_motion
```

renderings

To view the trajectory, you need to add the [MarkerArray] plug-in and select the [/rviz_visual_tools] topic.



Displays
- Global Options
  - Fixed Frame: base_footprint
  - Background Color: 48; 48; 48
  - Frame Rate: 30
  - Default Light: ☑
- Global Status: Ok
- Grid ☑
- Axes ☑
- TF □
- MarkerArray ☑
  - Status: Warn
  - Marker Topic: /rviz_visual_tools
  - Queue Size: 100
  - Namespaces
- MotionPlanning ☑
  - Status: Ok
  - Move Group Name...
  - Robot Description: robot_description
  - Planning Scene Topic: /move_group/mon...
  - Scene Geometry
  - Scene Robot
  - Planning Request
  - Planning Metrics
  - Planned Path

Given three reachable target points of the robotic arm, MoveIT will plan three feasible trajectories based on the target points, and then merge the three trajectories into one continuous trajectory.

8.2. Source code analysis

Set three reachable target points (you can have several target points, they must be reachable)

```
    vector<vector<double>> poses{
            {1.34, -1.0, -0.61, 0.2, 0},
            {0, 0, 0, 0, 0},
            {-1.16, -0.97, -0.81, -0.79, 3.14}
    };
    for (int i = 0; i < poses.size(); ++i) {
        multi_trajectory(yahboomcar, poses.at(i), trajectory);
    }
```

Plan each trajectory

```
void multi_trajectory(
        moveit::planning_interface::MoveGroupInterface &yahboomcar,
        const vector<double> &pose, moveit_msgs::RobotTrajectory &trajectory) {
    moveit::planning_interface::MoveGroupInterface::Plan plan;
    const robot_state::JointModelGroup *joint_model_group;
    // Get the starting position of the robot
    moveit::core::RobotStatePtr start_state(yahboomcar.getCurrentState());
    joint_model_group = start_state->getJointModelGroup(yahboomcar.getName());
    yahboomcar.setJointValueTarget(pose);
    yahboomcar.plan(plan);
    start_state->setJointGroupPositions(joint_model_group, pose);
    yahboomcar.setStartState(*start_state);
    trajectory.joint_trajectory.joint_names =
plan.trajectory_.joint_trajectory.joint_names;
    for (size_t j = 0; j < plan.trajectory_.joint_trajectory.points.size(); j++)
{

trajectory.joint_trajectory.points.push_back(plan.trajectory_.joint_trajectory.po
ints[j]);
    }
}
```

Trajectory merge

```
    moveit::planning_interface::MoveGroupInterface::Plan joinedPlan;
    robot_trajectory::RobotTrajectory rt(yahboomcar.getCurrentState()-
>getRobotModel(), "arm_group");
    rt.setRobotTrajectoryMsg(*yahboomcar.getCurrentState(), trajectory);
    trajectory_processing::IterativeParabolicTimeParameterization iptp;
    iptp.computeTimeStamps(rt, 1, 1);
    rt.getRobotTrajectoryMsg(trajectory);
    joinedPlan.trajectory_ = trajectory;
```

Track display

```
    moveit_visual_tools::MoveItVisualTools tool(yahboomcar.getPlanningFrame());
    tool.deleteAllMarkers();
/*
...
*/
// display track
    tool.publishTrajectoryLine(joinedPlan.trajectory_,
yahboomcar.getCurrentState()->getJointModelGroup("arm_group"));
    tool.trigger();
```

Execute trajectory planning

```
    if (!yahboomcar.execute(joinedPlan)) {
        ROS_ERROR("Failed to execute plan");
        return false;
    }
```