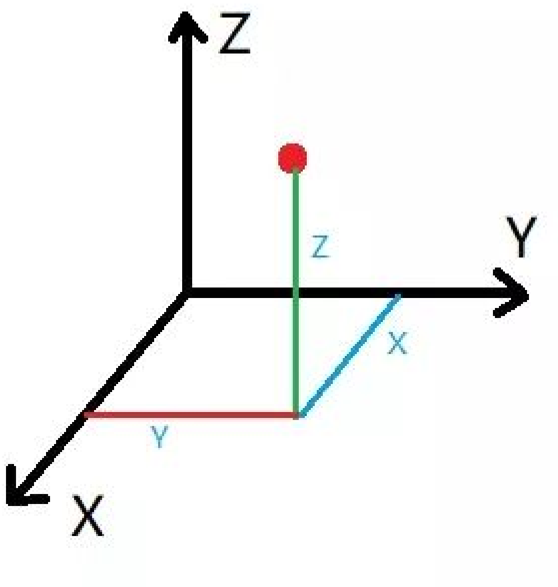
5. MoveIt Cartesian path

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).

5.1, Introduction

The Cartesian coordinate system is the collective name for the Cartesian coordinate system and the oblique coordinate system. A Cartesian path is actually a line connecting any two points in space.



5.2, 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
```

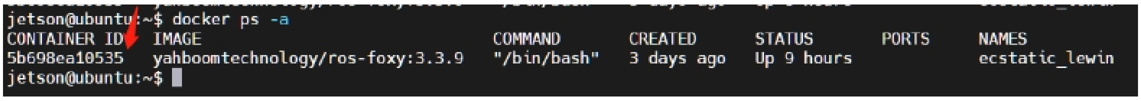
Start MoveIT

```
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:

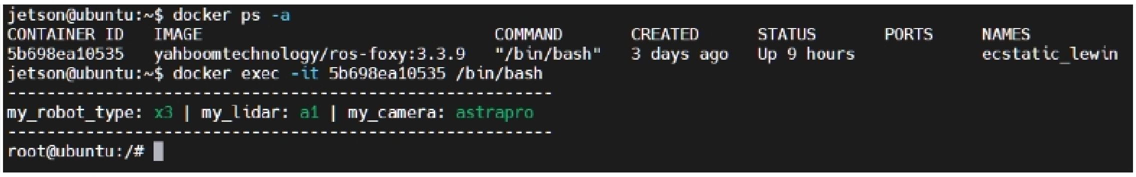
```
docker ps -a
```



```
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:/# 
```

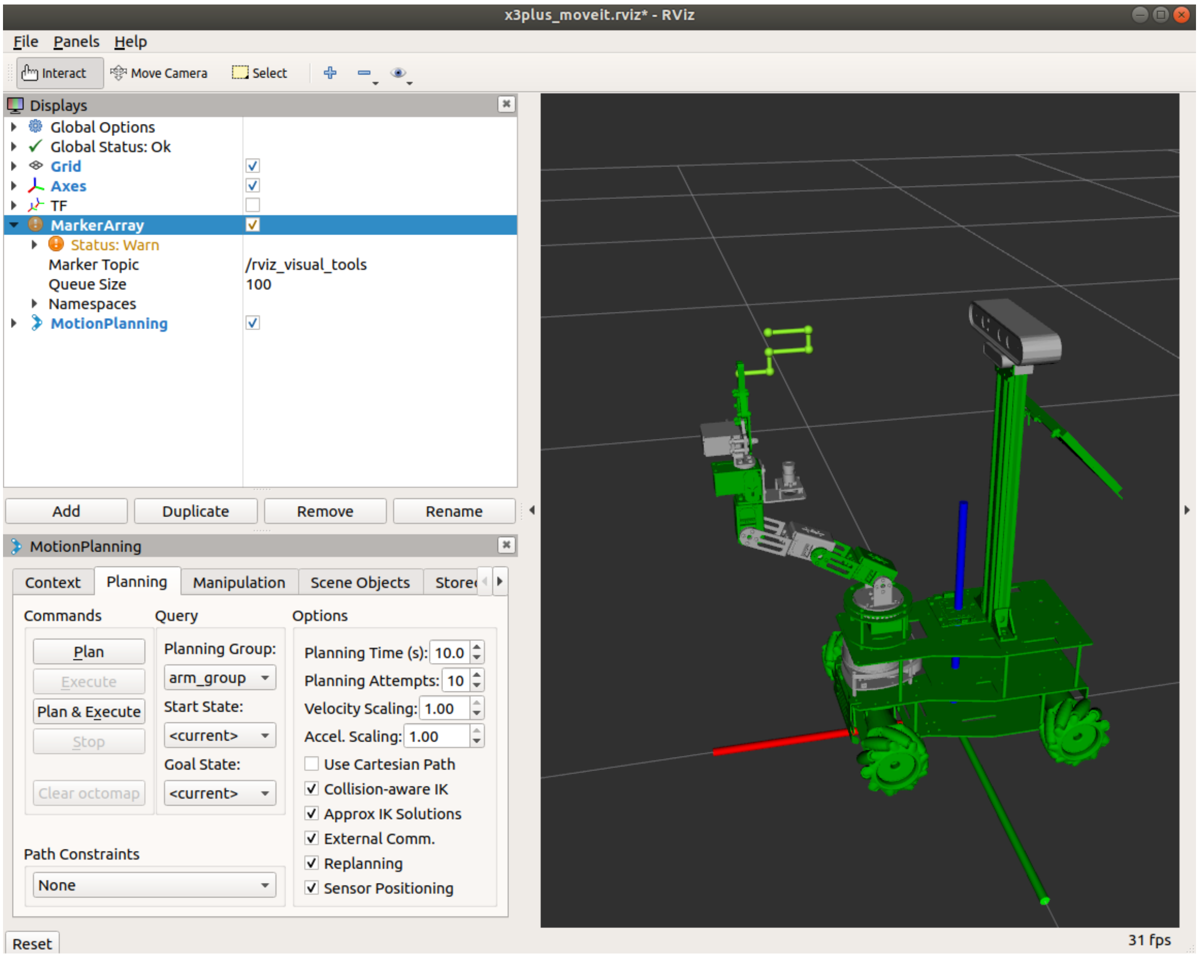
After successfully entering the container, you can open countless terminals to enter the container.

Start Cartesian Path Node

```
rosrun arm_moveit_demo 04_cartesian # C++
rosrun arm_moveit_demo 04_cartesian.py # python
```

- C++ code examples

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



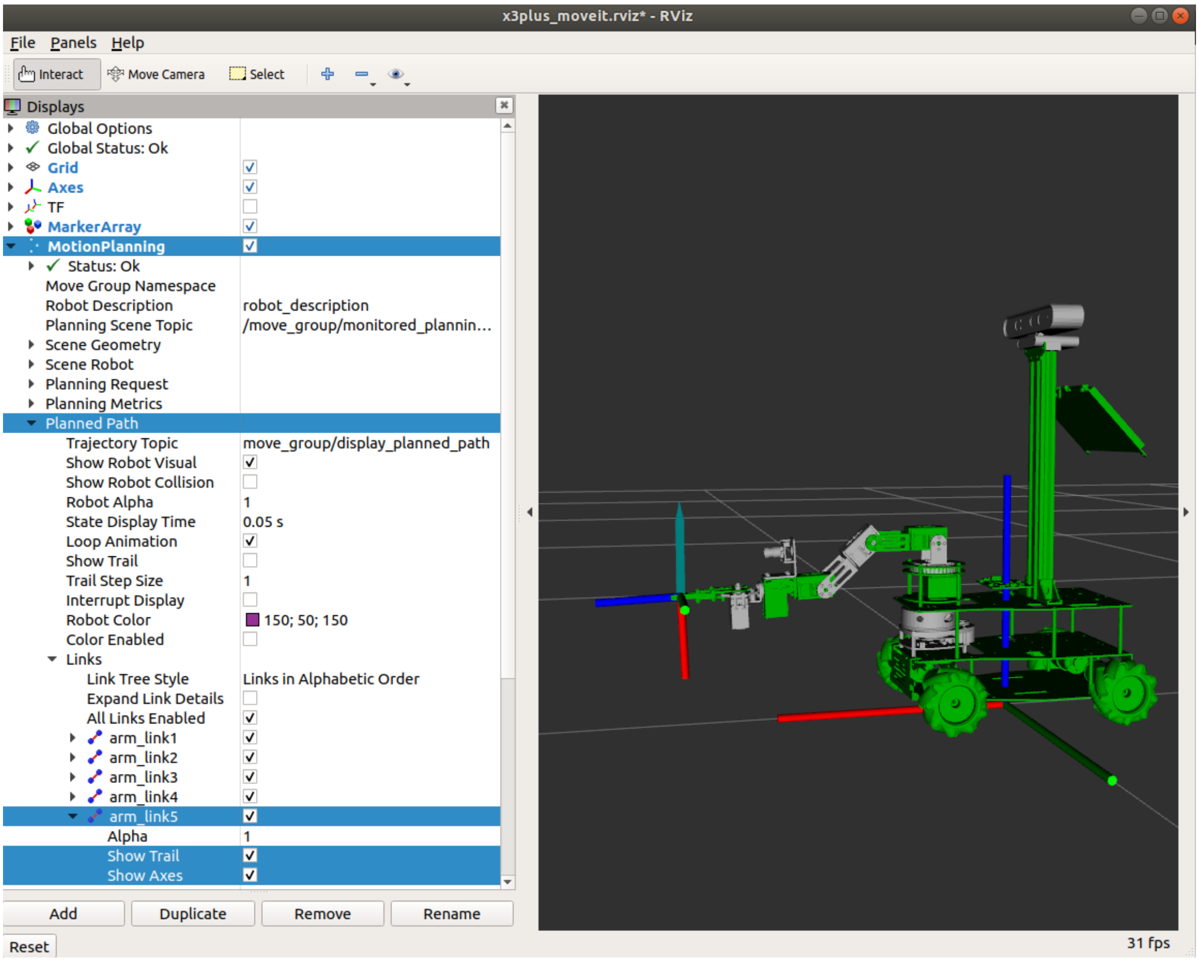
☑ Grid
☑ Axes
□ TF
☑ MarkerArray
Marker Topic /rviz_visual_tools
Queue Size 100
☑ MotionPlanning

Planning Group: arm_group
Start State: <current>
Goal State: <current>
Path Constraints: None

Planning Time (s): 10.0
Planning Attempts: 10
Velocity Scaling: 1.00
Accel. Scaling: 1.00
□ Use Cartesian Path
☑ Collision-aware IK
☑ Approx IK Solutions
☑ External Comm.
☑ Replanning
☑ Sensor Positioning

- python code examples

The python code does not have a similar trajectory to C++, but you can view the end description and open it as shown in the figure below.



Global Options
Global Status: Ok
Grid ☑
Axes ☑
TF □
MarkerArray ☑
MotionPlanning ☑
Status: Ok
Move Group Namespace
Robot Description robot_description
Planning Scene Topic /move_group/monitored_plannin...
Scene Geometry
Scene Robot
Planning Request
Planning Metrics
Planned Path
Trajectory Topic move_group/display_planned_path
Show Robot Visual ☑
Show Robot Collision □
Robot Alpha 1
State Display Time 0.05 s
Loop Animation ☑
Show Trail □
Trail Step Size 1
Interrupt Display □
Robot Color 150; 50; 150
Color Enabled □
Links
Link Tree Style Links in Alphabetic Order
Expand Link Details □
All Links Enabled ☑
arm_link1 ☑
arm_link2 ☑
arm_link3 ☑
arm_link4 ☑
arm_link5 ☑
Alpha 1
Show Trail ☑
Show Axes ☑

5.3, source code

5.3.1, py file

Set a specific location

```
    rospy.loginfo("Set Init Pose")
    joints = [0, -1.57, -0.74, 0.71, 0]
    yahboomcar.set_joint_value_target(joints)
    yahboomcar.execute(yahboomcar.plan())
```

Add waypoint

```
    #Initialize the waypoint list
    waypoints = []
    # If True, add the initial pose to the waypoint list
    waypoints.append(start_pose)
    for i in range(3):
        #Set waypoint data and add it to the waypoint list
        wpose = deepcopy(start_pose)
        wpose.position.z += 0.13
        waypoints.append(deepcopy(wpose))
        wpose.position.z -= 0.13
        waypoints.append(deepcopy(wpose))
```

waypoint planning

```
        '''
        waypoints: list of waypoints
        eef_step: terminal step value, calculate the inverse solution every 0.1m
to determine whether it is reachable
        jump_threshold: Jump threshold, set to 0 to disallow jumping
        plan: path, fraction: path planning coverage
        '''
        (plan, fraction) = yahboomcar.compute_cartesian_path(waypoints, 0.1,
0.0, True)
```

5.3.2, C++ files

Set a specific location

```
    ROS_INFO("Set Init Pose.");
    //Set specific location
    vector<double> pose{0, -0.69, -0.17, 0.86, 0};
    yahboomcar.setJointValueTarget(pose);
```

Add waypoint

```
    //Initialize path point vector
    std::vector<geometry_msgs::Pose> waypoints;
    //Add the initial pose to the waypoint list
    waypoints.push_back(start_pose);
    start_pose.position.x -= 0.04;
    waypoints.push_back(start_pose);
    start_pose.position.z -= 0.02;
    waypoints.push_back(start_pose);
    start_pose.position.x += 0.04;
    waypoints.push_back(start_pose);
    start_pose.position.z -= 0.02;
    waypoints.push_back(start_pose);
    start_pose.position.x += 0.03;
    waypoints.push_back(start_pose);
```

waypoint planning

```
fraction = yahboomcar.computeCartesianPath(waypoints, eef_step, jump_threshold,
trajectory);
```