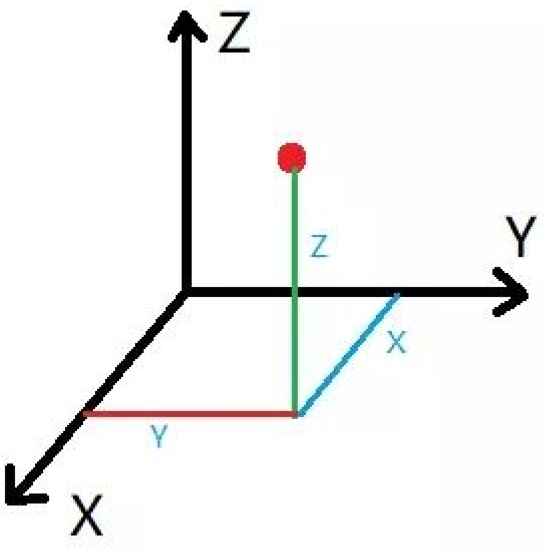
5. MoveIt Cartesian Path

This lesson takes the MoveIT simulation as an example. If you need to set the synchronization between the real machine and the simulation, please refer to the lesson [02, MoveIt Precautions and Controlling the Real Machine]. ! ! ! be careful! ! !

The effect demonstration is a virtual machine, and other masters are running (related to the performance of the master, 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

Start the MoveIT

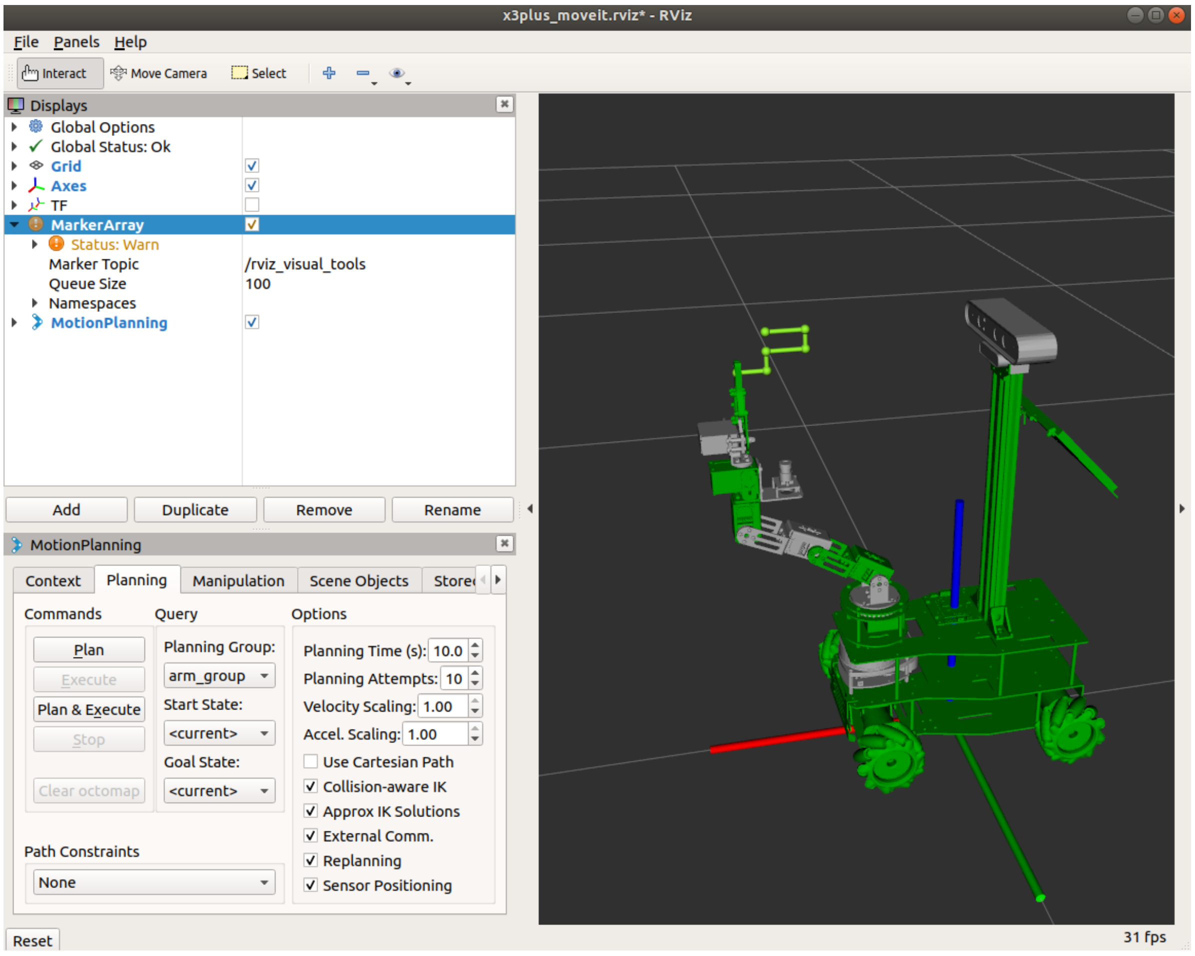
```
roslaunch arm_moveit_demo x3plus_moveit_demo.launch sim:=true
```

Start the Cartesian path node

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

- C++ code example

To view the track, you need to add the [MarkerArray] plugin and select the [/rviz_visual_tools] topic.



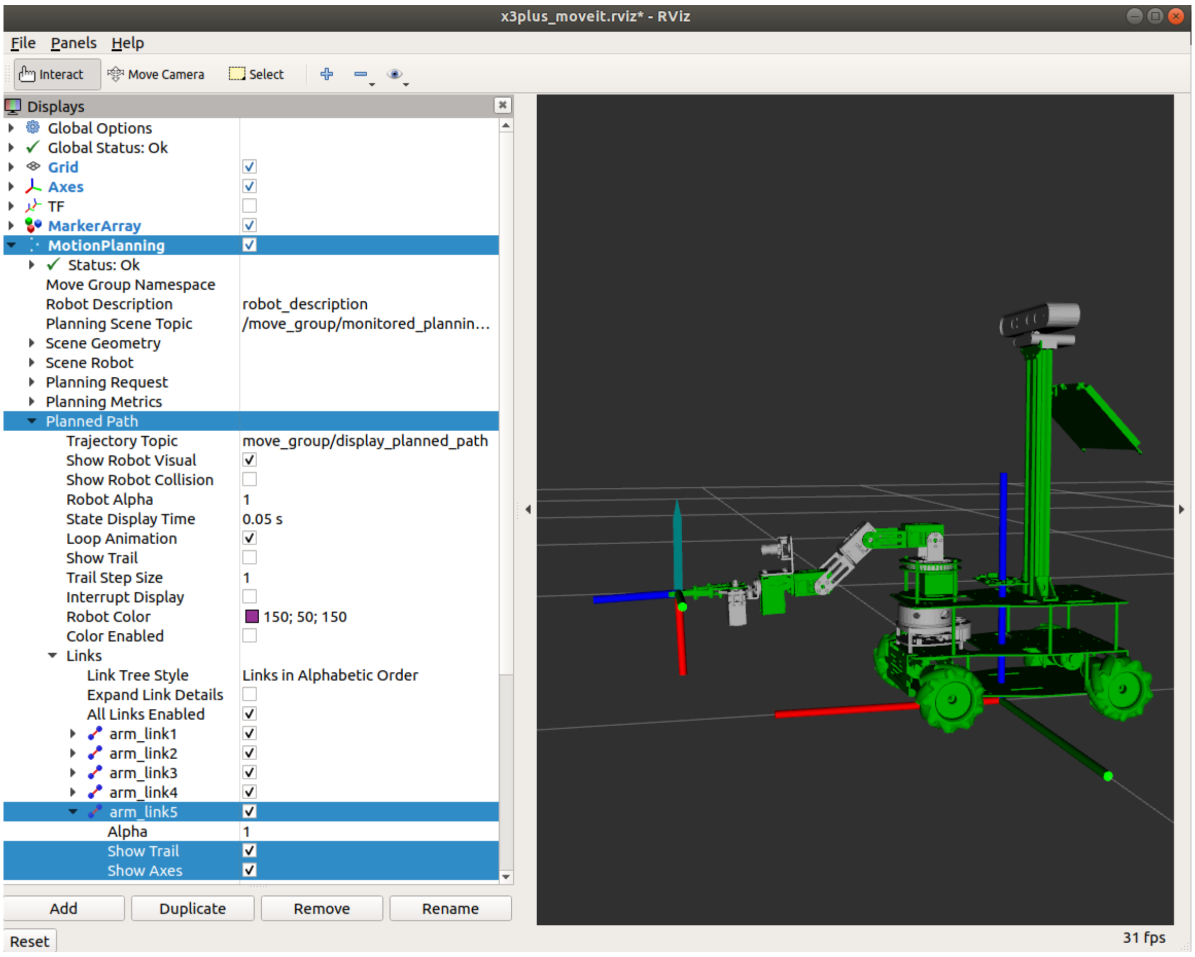
Global Options
Global Status: Ok
Grid ✓
Axes ✓
TF
MarkerArray ✓
Status: Warn
Marker Topic /rviz_visual_tools
Queue Size 100
Namespaces
MotionPlanning ✓

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 example

The python code does not have a similar trajectory to C++, but you can view the end description and open it as shown 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 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 waypoint list
    waypoints = []
    # If True, add the initial pose to waypoint list
    waypoints.append(start_pose)
    for i in range(3):
        # Set the 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:waypoints:waypoint list
        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 means jumping is not allowed
        plan: path, fraction:  path planning coverage
        '''
        (plan, fraction) = yahboomcar.compute_cartesian_path(waypoints, 0.1,
0.0, True)
```

5.3.1.C++ file

Set 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 waypoint 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);
```