6. MoveIt collision detection

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).

6.1. 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
```

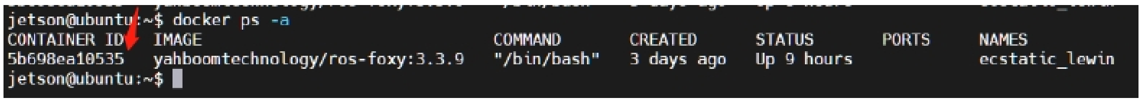
Start MoveIT

```
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:

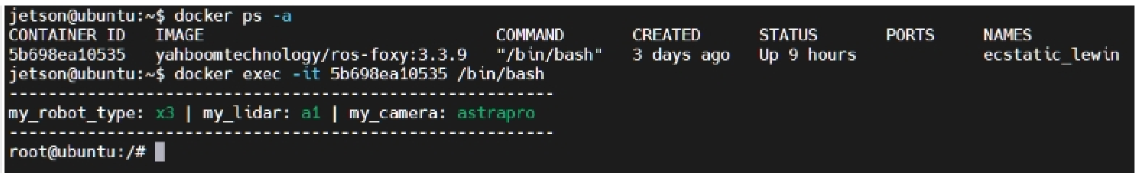
```
docker ps -a
```



```
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:/#
```

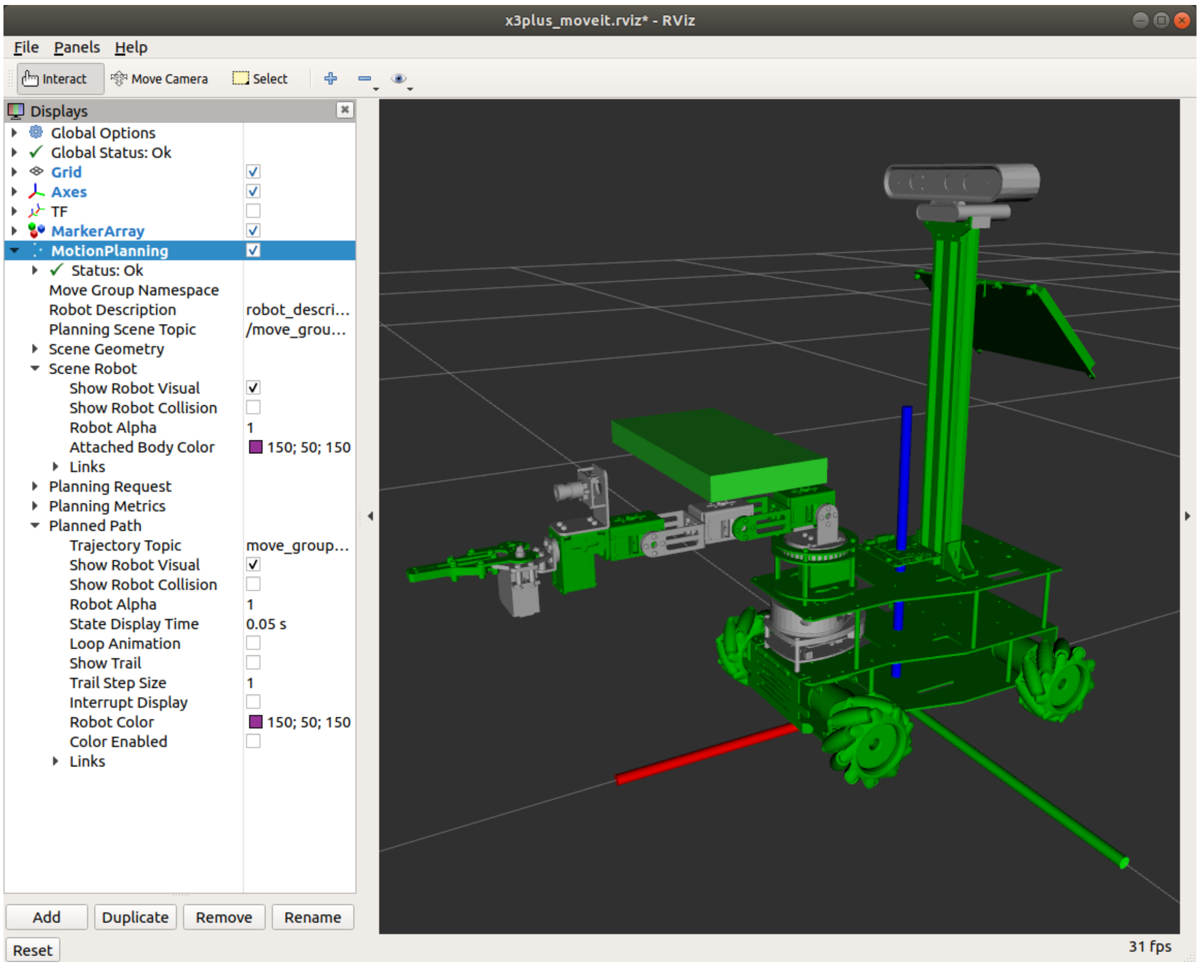
After successfully entering the container, you can open countless terminals to enter the container.

Start collision detection node

```
rosrun arm_moveit_demo 05_attached_object # C++
rosrun arm_moveit_demo 05_attached_object.py # python
```

Set up random movements for about 10 times. Since the target position is random, it may also be randomly placed on the obstacle. When the target position comes into contact with the obstacle, the planning will fail.

The renderings are as follows



Global Options	
Global Status: Ok	
Grid	✓
Axes	✓
TF	☐
MarkerArray	✓
MotionPlanning	✓
Status: Ok	
Move Group Namespace	
Robot Description	robot_descri…
Planning Scene Topic	/move_grou…
Scene Geometry	
Scene Robot	
Show Robot Visual	✓
Show Robot Collision	☐
Robot Alpha	1
Attached Body Color	🟪 150; 50; 150
Links	
Planning Request	
Planning Metrics	
Planned Path	
Trajectory Topic	move_group…
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	

6.2, source code

6.2.1, py file

Set the size and position of obstacles

```
    #Set the three-dimensional dimensions of the obstacle [length, width and
height]
    table_size = [0.1, 0.2, 0.02]
    table_pose = PoseStamped()
    table_pose.header.frame_id = 'base_link'
    table_pose.pose.position.x = 0.2
    table_pose.pose.position.y = 0
    table_pose.pose.position.z = 0.2 + table_size[2] / 2.0
    table_pose.pose.orientation.w = 1.0
```

Add obstacles to the scene

```
    scene.add_box('obj', table_pose, table_size)
```

Set random movement

```
    index = 0
    while index < 10:
        #Set random target point
        yahboomcar.set_random_target()
        # Start exercising
        yahboomcar.go()
        sleep(0.5)
        index += 1
        print ("{}th plan!!!".format(index))
```

6.2.2, C++ files

Set the size and position of obstacles

```
    moveit::planning_interface::PlanningSceneInterface scene;
    //////////////////////////Add obstacles////////////////////// /
    vector<string> object_ids;
    scene.removeCollisionObjects(object_ids);
    vector<moveit_msgs::CollisionObject> objects;
    moveit_msgs::CollisionObject obj;
    //Set the ID of the obstacle
    obj.id = "obj";
    object_ids.push_back(obj.id);
    //The status of the obstacle
    obj.operation = obj.ADD;
    obj.header.frame_id = frame;
    shape_msgs::SolidPrimitive primitive;
    //Set obstacle type
    primitive.type = primitive.BOX;
    //Set obstacle dimensions
    primitive.dimensions.resize(3);
    //Set the length, width and height of the obstacle
    primitive.dimensions[0] = 0.2;
    primitive.dimensions[1] = 0.1;
    primitive.dimensions[2] = 0.02;
    obj.primitives.push_back(primitive);
    geometry_msgs::Pose pose;
    //Set the position information of the obstacle [x,y,z]
    pose.position.x = 0.2;
    pose.position.y = 0;
    pose.position.z = 0.26;
    tf::Quaternion quaternion;
    // The units of R, P and Y are angles
    double Roll = 0.0;
    double pitch = 0.0;
    double Yaw = 90.0;
    quaternion.setRPY(Roll * M_PI / 180, Pitch * M_PI / 180, Yaw * M_PI / 180);
    pose.orientation.x = quaternion.x();
    pose.orientation.y = quaternion.y();
    pose.orientation.z = quaternion.z();
    pose.orientation.w = quaternion.w();
    //Set the pose information of the obstacle
    obj.primitive_poses.push_back(pose);
    objects.push_back(obj);
```

Set the color of the obstacle

```
    ///////////////////////////Set the color of obstacles/////////////////////
///
    // Create a color container for the detection object
    std::vector<moveit_msgs::ObjectColor> colors;
    moveit_msgs::ObjectColor color;
    //Add the id whose color needs to be set
    color.id = "obj";
    //Set RGBA value, range [0~1]
    color.color.r = 1.0;
    color.color.g = 0;
    color.color.b = 0;
    color.color.a = 0.5;
    colors.push_back(color);
    //Add the set information to the scene
    scene.applyCollisionObjects(objects, colors);
```

Set random movement

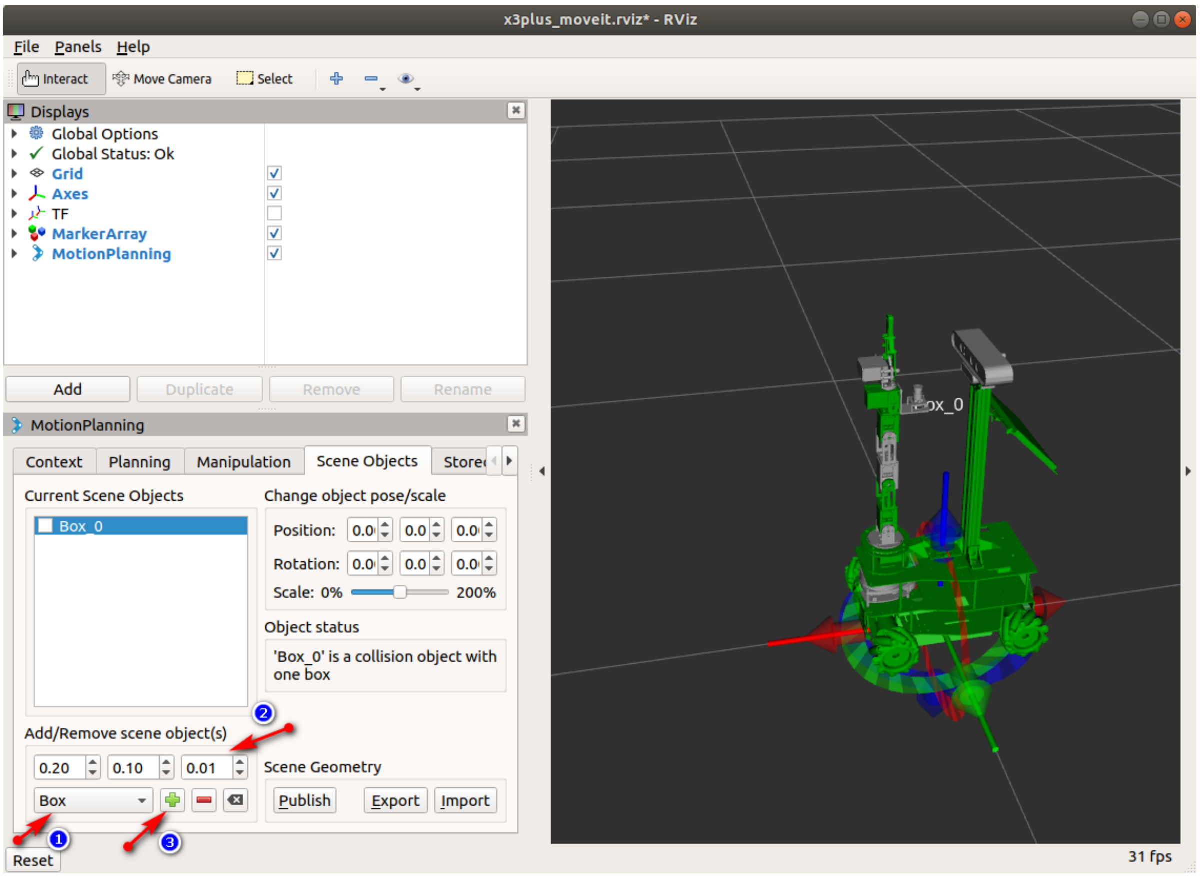
```
    int index = 0;
    while (index < 10) {
        yahboomcar.setRandomTarget();
        yahboomcar.move();
        sleep(0.5);
        index++;
        cout << "The " << index << " plan!!!" << endl;
    }
```

6.3, Scene Object

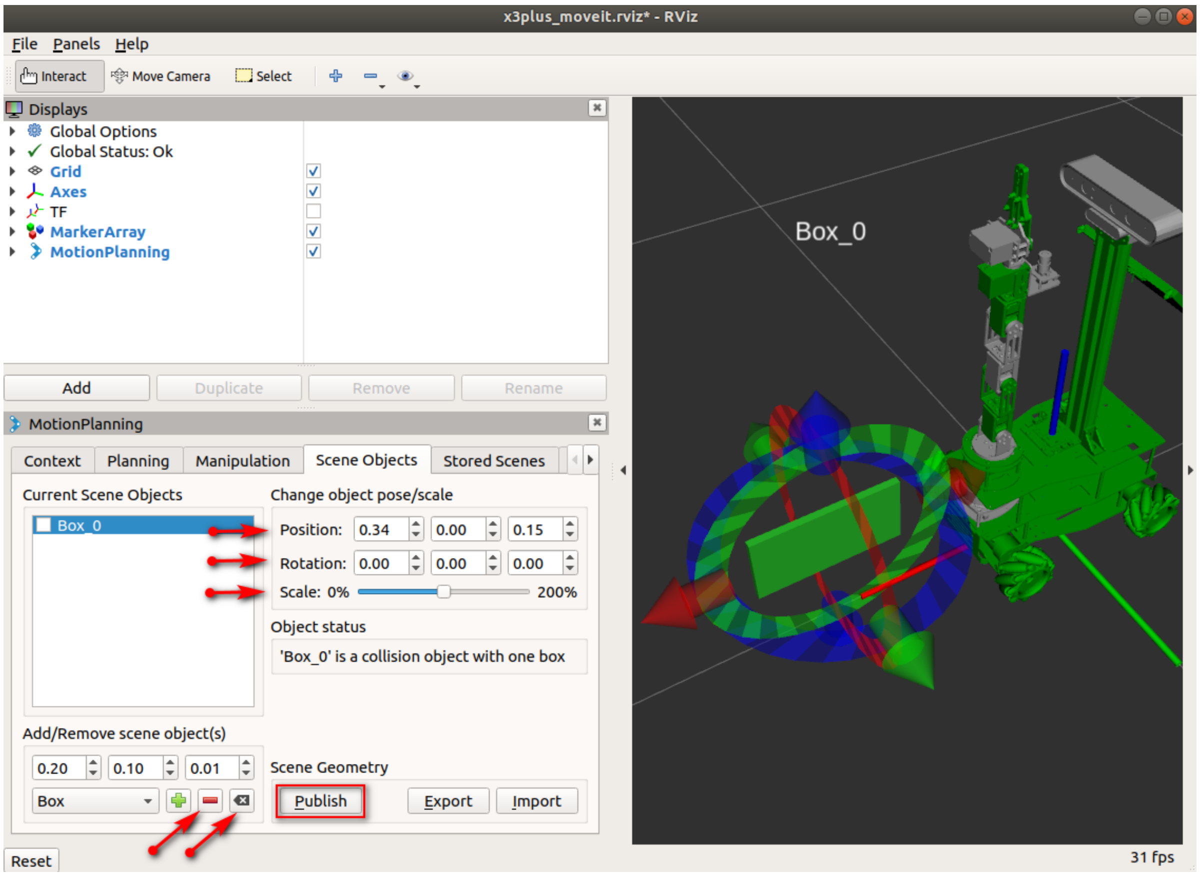
In addition to using code to add or delete obstacles, you can also use the [MotionPlanning] tool in rviz to add or delete.

6.3.1. Add obstacles

The first step is to select the shape of the obstacle [for example: Box], then set the size of the obstacle, and then click the [+] sign. At this time, the position of the obstacle is at the center of the origin. As shown below



In the second step, drag the arrow on the right to drag out the obstacle. You can also directly set the [Position] position information, [Rotation] posture information, [Scale] scaling ratio, etc. As shown below



Step 3: After completing the above setting steps, be sure to click [Publish] for it to take effect; otherwise, the robot may pass through obstacles.

6.3.2. Delete obstacles

As shown above,

Delete: Select the obstacle first, then click the [-] sign to delete the current obstacle.

Clear: Click directly on the [x] sign to clear the obstacles.