Planning Around Objects

moveit.picknik.ai/humble/doc/tutorials/planning_around_objects/planning_around_objects.html

This tutorial will introduce you to inserting objects into the planning scene and planning around them.

Prerequisites

If you haven't already done so, make sure you've completed the steps in Visualizing in RViz. This project assumes you are starting with the `hello_moveit` project, where the previous tutorial left off. If you just want to run the tutorial, you can follow the Docker Guide to start a container with the completed tutorial.

Steps

1 Add include for Planning Scene Interface_

At the top of your source file, add this to the list of includes:

```
#include <moveit/planning_scene_interface/planning_scene_interface.h>
```

2 Change the Target Pose_

First, update the target pose with the following change to make the robot plan to a different location:

```
// Set a target Pose
auto const target_pose = [] {
  geometry_msgs::msg::Pose msg;
  msg.orientation.w = 1.0;
  msg.position.x = 0.28;
  msg.position.y = 0.4;  // <---- This value was changed
  msg.position.z = 0.5;
  return msg;
}();
move_group_interface.setPoseTarget(target_pose);
```

3 Create a Collision Object_

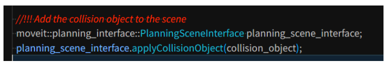
In the next block of code, we create a collision object. The first thing to notice is that it is being placed in the frame of the robot. If we had a perception system that reported the location of obstacles in our scene, then this is the sort of message it would build. Because this is just an example, we are creating it manually. One thing to notice at the end of this block of code is that we set the operation on this message to `ADD`. This results in the object getting added to the collision scene. Place this code block between setting the target pose from the previous step and creating a plan.

在接下来的代码块中，我们创建一个碰撞对象。首先要注意的是，它被放置在机器人的框架内。如果我们有一个感知系统来报告我们场景中的障碍物的位置，那么这就是它要建立的那种信息。因为这只是一个例子，我们正在手动创建它。在这段代码的最后，有一点需要注意的是，我们把这个消息的操作设置为ADD。这将导致物体被添加到碰撞场景中。把这个代码块放在上一步设置目标姿势和创建计划之间。

```
// Create collision object for the robot to avoid
auto const collision_object = [frame_id =
                                 move_group_interface.getPlanningFrame()] {
  moveit_msgs::msg::CollisionObject collision_object;
  collision_object.header.frame_id = frame_id;
  collision_object.id = "box1";
  shape_msgs::msg::SolidPrimitive primitive;

  // Define the size of the box in meters
  primitive.type = primitive.BOX;
  primitive.dimensions.resize(3);
  primitive.dimensions[primitive.BOX_X] = 0.5;
  primitive.dimensions[primitive.BOX_Y] = 0.1;
  primitive.dimensions[primitive.BOX_Z] = 0.5;

  // Define the pose of the box (relative to the frame_id)
  geometry_msgs::msg::Pose box_pose;
  box_pose.orientation.w = 1.0;
  box_pose.position.x = 0.2;
  box_pose.position.y = 0.2;
  box_pose.position.z = 0.25;

  collision_object.primitives.push_back(primitive);
  collision_object.primitive_poses.push_back(box_pose);
  collision_object.operation = collision_object.ADD;

  return collision_object;
}();
```

4 Add the Object to the Planning Scene

Finally, we need to add this object to the collision scene. To do this, we use an object called the `PlanningSceneInterface` that uses ROS interfaces to communicate changes to the planning scene to `MoveGroup`. This code block should directly follow the code block that creates the collision object.



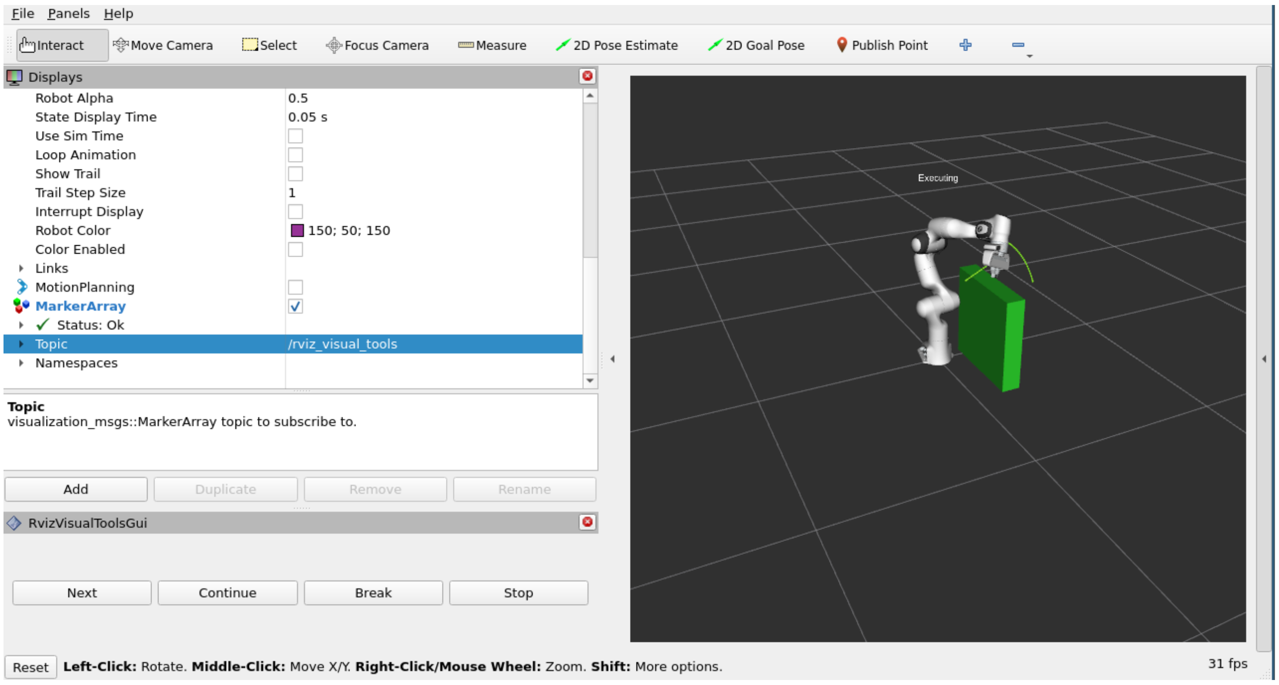
```
//!!! Add the collision object to the scene
moveit::planning_interface::PlanningSceneInterface planning_scene_interface;
planning_scene_interface.applyCollisionObject(collision_object);
```

```
// Add the collision object to the scene
moveit::planning_interface::PlanningSceneInterface planning_scene_interface;
planning_scene_interface.applyCollisionObject(collision_object);
```

5 Run the Program and Observe the Change

Just as we did in the last tutorial, start RViz using the `demo.launch.py` script and run our program. If you're using one of the Docker tutorial containers, you can specify a different RViz configuration that already has the RvizVisualToolsGui panel added using:

```
ros2 launch moveit2_tutorials demo.launch.py rviz_config:=panda_hello_moveit.rviz
```



Robot Alpha	0.5
State Display Time	0.05 s
Use Sim Time	☐
Loop Animation	☐
Show Trail	☐
Trail Step Size	1
Interrupt Display	☐
Robot Color	150; 50; 150
Color Enabled	☐
Links	
MotionPlanning	☐
MarkerArray	✓
✓ Status: Ok	
Topic	/rviz_visual_tools
Namespaces	

Topic
visualization_msgs::MarkerArray topic to subscribe to.

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

31 fps

Summary

- You extended the program you wrote with MoveIt to plan around an object in the scene.
- Here is a copy of the full hello_moveit.cpp source.

Further Reading

- Examples of using the Planning Scene for collision and constraint checking.
- Examples of using the Planning Scene ROS API.
- Example of visualizing collision objects.
- Example of subframes used for planning with objects.

Next Step

In the next tutorial Pick and Place with MoveIt Task Constructor, you will be introduced to a higher layer tool designed to solve harder motion plans. In this next tutorial, you will create a program to pick and place an object.