Workshop 6 Using the ROS Moveit!

Aims and Objectives

Addressing the motion planning problem.

Things to learn

Moving a robot by directly controlling its joints manually might be a difficult task, especially if we want to add position or velocity constraints to the robot motion. For this reason, we will solve these problems using the ROS MoveIt!

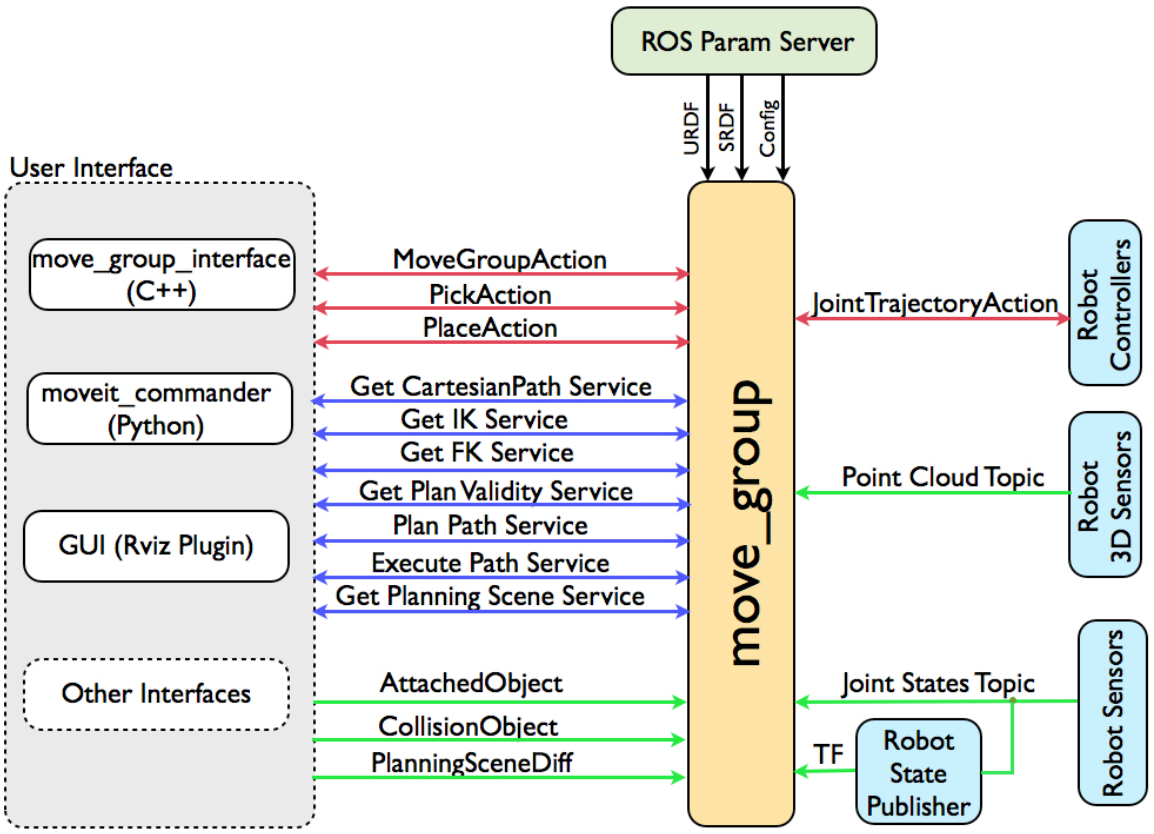
MoveIt! contains state-of-the-art software for motion planning, manipulation, 3D perception, kinematics, collision checking, control, and navigation. Apart from the command line interface, MoveIt! has some good GUI to interface a new robot to MoveIt!.
Also, there is a RViz plugin, which enables motion planning from RViz itself. We will also see how to motion plan our robot using MoveIt! C++ APIs.

Exercise

1. **Installing MoveIt!**

```
$ sudo apt-get install ros-melodic-moveit ros-melodic-moveit-plugins ros-melodic-moveit-planners
```

2. MoveIt! architecture



- **The move_group node**
  We can say that move_group is the heart of MoveIt!, as this node acts as an integrator of the various components of the robot and delivers actions/services according to the user's needs.

Move_group connects all the functionalities as plugins. There are plugins for kinematics solvers, motion planning, and so on.
After motion planning, the generated trajectory talks to the controllers in the robot using the FollowJointTrajectoryAction interface. This is an action interface in which an action server is run on the robot, and move_node initiates an action client which talks to this server

- **Motion planning using MoveIt!**
  Assume that we know the starting pose of the robot, a desired goal pose of the robot, the geometrical description of the robot, and geometrical description of the world, then motion planning is the technique to find an optimum path that moves the robot gradually from the start pose to the goal pose, while never touching any obstacles in the world and without colliding with the robot links. and executes the trajectory on the real robot/Gazebo simulator.

  MoveIt! can talk to the motion planners through the plugin interface. We can use any motion planner by simply changing the plugin. This method is highly extensible so we can try our own custom motion planners using this interface. The move_group node talks to the motion planner plugin via the ROS action/services. The default planner for the move_group node is OMPL (http://ompl.kavrakilab.org/). With additional kinematic constraints for motion planner (inbuilt constraints in MoveIt! are position constraints, orientation constraints, visibility constraints, joint constraints and user specified constraints), the move_group node will generate the suitable trajectory from the motion planner which obeys all the constraints. This can be
  sent to robot joint trajectory controllers.

3. **Generating MoveIt! configuration package using the Setup Assistant tool**

Step 1 – Launching the Setup Assistant tool

To start the MoveIt! Setup Assistant tool, we can use the following command:

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

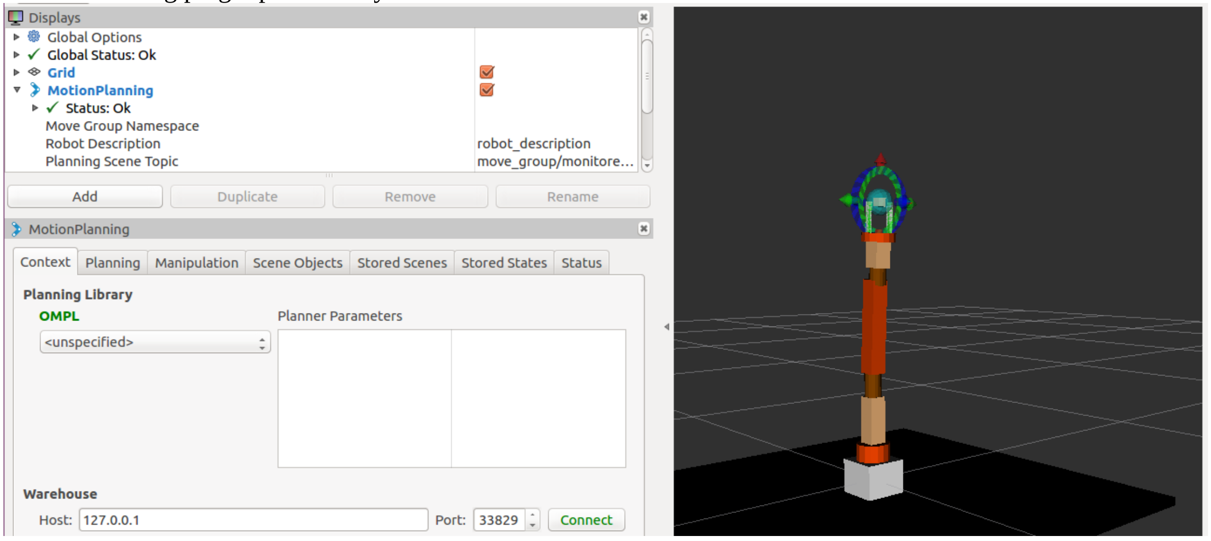
This will bring up a window with two choices: **Create New MoveIt! Configuration Package** or **Edit Existing MoveIt! Configuration Package**. Here we are creating a new package, so we need that option.

Step 2 – Generating the Self-Collision matrix
Step 3 – Adding virtual joints
Step 4 – Adding planning groups
Step 5 – Adding the robot poses
Step 6 – Setting up the robot end effector
Step 7 – Adding passive joints
Step 8 – Author information
Step 9 – Generating configuration files

After the final stop, you shall generate a configuration folder which saves all the configuration files. You can also download the config folder ***seven_dof_arm_config*** I uploaded to the week 07 channel's file folder. Please replace the string **mystudentid** to your own **"initial+id"** just like previous workshop in the following three files: *package.xml* and *CMaketLists.txt* in this folder and *planning_context.launch* file in the launch subfolder.

4. **Motion planning of robot in RViz using MoveIt! configuration package**
   MoveIt! provides a plugin for RViz, which allows it to create new planning scenes where **robot works**, **generate motion plans**, and **add new objects**, visualize the planning output and can directly interact with the visualized robot.
   The MoveIt! configuration package consists of configuration files and launch files to start motion planning in RViz. There is a demo launch file in the package to explore all the functionalities of this package.
   The following is the command to invoke the demo launch file:

$ roslaunch seven_dof_arm_config demo.launch
If everything works fine, we will get the following screen of RViz being loaded with the MotionPlanning plugin provided by MoveIt!:



Displays
- Global Options
- Global Status: Ok
- Grid ✓
- MotionPlanning ✓
  - Status: Ok
  - Move Group Namespace
  - Robot Description: robot_description
  - Planning Scene Topic: move_group/monitore...

Add | Duplicate | Remove | Rename

MotionPlanning

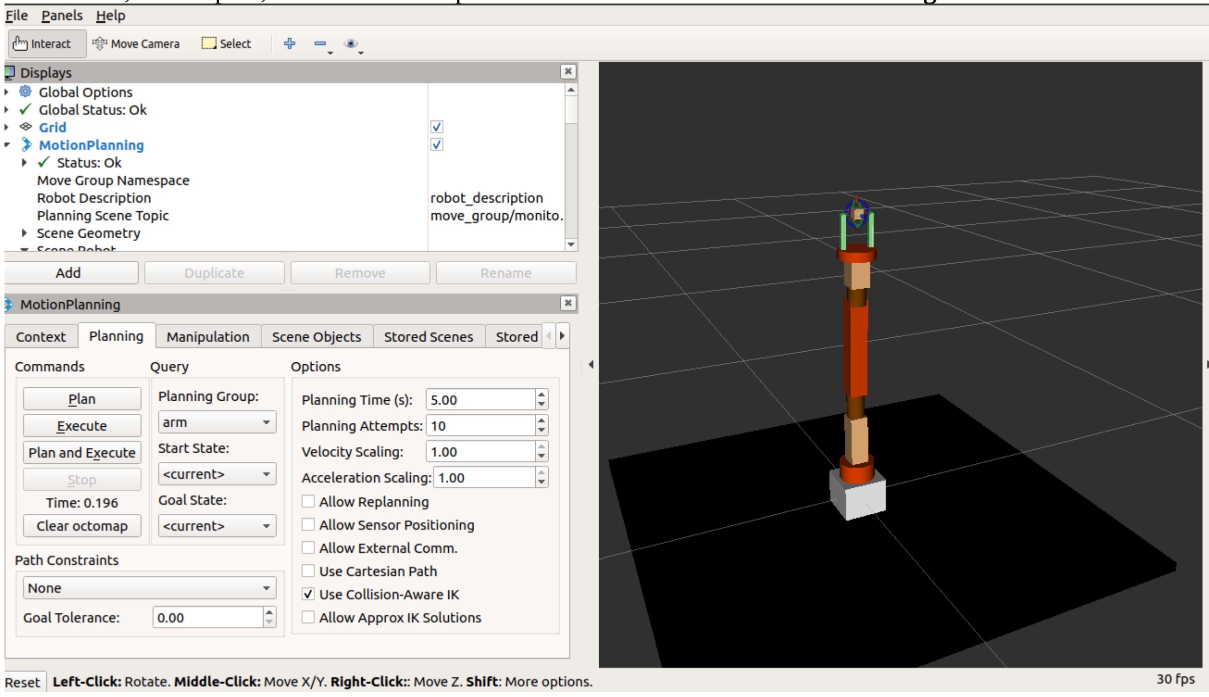
Context | Planning | Manipulation | Scene Objects | Stored Scenes | Stored States | Status

Planning Library
OMPL
<unspecified>
Planner Parameters

Warehouse
Host: 127.0.0.1 Port: 33829 Connect

a) **Using the RViz Motion Planning plugin**
From the preceding figure , we can see that the RViz-Motion Planning plugin is loaded on the left side of the screen. There are several tabs on the **Motion Planning** window, such as **Context**, **Planning**, and so on. The default tab is the **Context** tab and we can see the default **Planning Library** as OMPL, which is shown in green. It indicates that MoveIt! successfully loaded the motion planning library. If it is not loaded, we can't perform motion planning.

b) Next is the **Planning** tab. This is one of the frequently used tabs used to assign the **Start State**, **Goal State**, **Plan** a path, and **Execute** the path. Shown next is the GUI of the **Planning** tab:



File Panels Help

Interact | Move Camera | Select

Displays
- Global Options
- Global Status: Ok
- Grid ✓
- MotionPlanning ✓
  - Status: Ok
  - Move Group Namespace
  - Robot Description: robot_description
  - Planning Scene Topic: move_group/monito...
  - Scene Geometry

Add | Duplicate | Remove | Rename

MotionPlanning

Context | Planning | Manipulation | Scene Objects | Stored Scenes | Stored

Commands: Plan, Execute, Plan and Execute, Stop, Time: 0.196, Clear octomap

Query: Planning Group: arm; Start State: <current>; Goal State: <current>

Options:
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
☐ Allow Replanning
☐ Allow Sensor Positioning
☐ Allow External Comm.
☐ Use Cartesian Path
☑ Use Collision-Aware IK
☐ Allow Approx IK Solutions

Path Constraints: None
Goal Tolerance: 0.00

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click::** Move Z. **Shift:** More options. 30 fps

c) **Interfacing the MoveIt! configuration package to Gazebo**
We have already worked with the Gazebo simulation of this arm and attached controllers to it. For interfacing the arm in MoveIt! to Gazebo, we need a trajectory controller which has the FollowJointTrajectoryAction interface, as we mentioned in the MoveIt! architecture.
The following is the procedure to interface MoveIt! to Gazebo.

i. **Writing the controller configuration file for MoveIt!**

See the controllers.yaml has to be created inside the config folder of the seven_dof_arm_config package.

ii. **Creating the controller launch files**
You can find seven_dof_arm_moveit_controller_manager.launch file in launch folder of the seven_dof_arm_config package.

iii. **Creating the controller configuration file for Gazebo**
trajectory_control.yaml in config folder, which contains the list of the Gazebo ROS controllers that need to be loaded along with Gazebo.
You will get this file from the the seven_dof_arm_gazebo package created in last workshop

iv. **Creating the launch file for Gazebo trajectory controllers**
After creating the configuration file, we can load the controllers along with Gazebo. We have to create a launch file which launches Gazebo, the trajectory controllers, and theMoveIt! interface in a single command.
The launch file *seven_dof_arm_bringup_moveit.launch* contains the definition to launch all these commands

After all these steps, we can start motion planning inside RViz and execute in Gazebo using the following single command:

```
$ roslaunch seven_dof_arm_gazebo seven_dof_arm_bringup_moveit.launch
```

Note that, before properly launching the planning scene, we should use the following command to install some packages needed by MoveIt! to use ROS controllers:

```
$ sudo apt-get install ros-melodic-joint-state-controller ros-melodic-position-controllers ros-melodic-joint-trajectory-controller
```

You can run: $ ***rostopic list*** to see what happens.