项目二: URDF 移动机器人及 Movelt!机械臂控制

第二部分: 使用 Movelt!控制机器人

1. Introduction to Movelt!

Moveit is a sophisticated piece of software written on top of ROS for achieving inverse kinematics, motion or path planning, 3D perception of the environment, collision checking, and so on. It is the primary source of functionality for manipulation in ROS. Moveit understands a robot arm configuration (geometry and link information) through urdf and ROS message definitions and utilizes the ROS visualizing (RViz) tool to perform manipulation.

Moveit is used in more than 100 robot arms and you can find more information about those robots here: <u>https://moveit.ros.org/robots/</u>. Moveit has lots of advanced features and is used by many industrial robots as well. Covering all the Moveit! concepts is out of the scope of this course, we shall only look at it from the engineering and application point of view, and what we need to move and control our robot arm. The following is the architecture of Moveit!.



Figure 1: Movelt Architecture

Here, we have the most important component, that is, the move_group node, which is responsible for putting all other components together to provide the user with

necessary actions and service calls to use. The user could interface by using a simple scripting interface (for beginners) called the moveit_commander interface, a C++ wrapper called move_group_interface, a Python interface written on top of move_it_commander, or the GUI interface using an RViz plugin. The move_group node would need the robot information that is defined through URDF, as well as configuration files. Moveit understands the robot in a format called SRDF (semantic robot description format) that Moveit converts into URDF while setting up the robot arm. Also, the move_group node understands the robot arm's joint states and talks back via the FollowJointTrajectoryAction client interface. For more information about Movelt! please see: https://moveit.ros.org/documentation/concepts/.

2. Install and configure Movelt for our mobile robot

Installing and configuring Moveit is a multistep process. Let's begin by learning how to install it.

2.1 Installing Moveit

\$ sudo	apt inst	tall ros-	-melodic-moveit
\$ sudo	apt-get	install	ros-melodic-moveit-setup-assistant
\$ sudo	apt-get	install	ros-melodic-moveit-simple-controller-manager
\$ sudo	apt-get	install	ros-melodic-moveit-fake-controller-manager

Once they're all installed, we can begin configuring our robot using a Moveit setup assistant wizard.

2.2 Configuring the Movelt setup assistant wizard

This wizard is very useful, particularly because it helps us save time. Some of the things that we can do with this wizard are as follows:

- Define collision zones for our robot arm
- Set custom poses
- Choose the necessary kinematics library
- Define ROS controllers
- Create the necessary simulation files

We can invoke the setup assistant using the following command:

\$ roslaunch moveit_setup_assistant setup_assistant.launch

You should see the window shown here:

		Moveltl Setup Assistant		000
Start Self Collisions	Movelt! Setup	Assistant reating a Semantic Robot Desc tilizing all aspects of Movetti fo	ription Format (SRDF) file, various	syaml configuration
Virtual Joints	Create new or edit ex	isting?		
Planning Groups	configuration package. Here a new configuration package	you have the option to create or load an existing one. Note: ration package public this		In
Robot Poses	Setup Assistant are likely to b	be overwritten by this tool.	200	2
End Effectures	Create <u>New Movelt</u> Configuration Package	Edit Existing Movelt Configuration Package	A MP	
Passive Joints				
ROS Centrol				
Simulation			N<	lovelt!
3D Perception			Setup	Assistant 2.0
Author Information				
Configuration Files				

Figure 2: Movelt setup assistant wizard

Now, let's look into the configuration steps, one by one.

2.2.1 Loading the robot model

Let's configure our robot in Moveit by selecting the respective robot URDF. We do that by clicking Create New Moveit Configuration Package, loading our robot URDF, mobile_manipulator.urdf, and selecting Load Files. You should see a success message, along with our robot in the right-hand pane:

	Moveiti Setup Assistant	© © ©
Start	Movelt! Setup Assistant	
Self-Collisions	These tools will assist you in creating a Semantic Robot Description Format (SRDF) file, various yaml configuration and many roslaunch files for utilizing all aspects of Movelt! functionality.	
Virtual Joints	Create new or edit existing?	
Planning Groups	Christian News Movelt Charling Frankage Edit Existing Movelt Care Again Allow Package	P
Robot Poses	Load a URDF or COLLADA Robot Model	None I
End Effectors		
Passive Joints	[pter_3_within_thet_description/setf/matile_munips/attocurdf]browse_}	100
ROS Control		
Simulation		
3D Perception		
Author Information	Success! Use the left navigation pane to continue.	
Configuration Files	100% Load Files	

Figure 3: Movelt loading success

Now, let's set up the components on the left-hand pane.

2.2.2 Setting up self-collisions

Click on Self-Collisions on the left pane and select Generate Collision Matrix. Here, you can set the sampling density high if you wish to move the robot arm in a more confined space. This may increase the planning time for the robot to execute a trajectory and may sometimes fail execution due to a collision assumption.

2.2.3 Setting up planning groups

Let's set up planning groups by following these steps:

1. In Planning Groups, add our robot arm group by selecting Add Group.

2. Name our group arm.

3. Select Kinematic Solver as kdl_kinematics_plugin/KDLKinematicsPlugin. Set the resolution and timeout as the default values.

- 4. Select RRTStar as our Planner.
- 5. Now, add our robot arm joints and click Save.

Your final window should look as follows:

	Movelt! Setup Assistant	000
Start	Define Planning Groups	
Self-Collisions	Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of light link pairs considered for planning and collision checking. Define	
Virtual Joints	individual groups for each subset of the robot you want to plan for.Note: when adding a link to the group, its parent joint is added too and vice versa.	
Planning Groups	Current Groups	
Robot Poses	Joints arm_base_joint - Revolute	
End Effectors	elbow Joint - Revolute shoulder_joint - Revolute	
Passive Joints	top_wrist_joint-Revolute Links Chain	
ROS Control	Subgroups	
Simulation		
3D Perception		
Author Information		
Configuration Files	Expand All Collapse All Delete Selected Edit Selected Add Group	

Figure 4: Movelt planning groups

Once the arm group has been set, we can set the poses for the arm.

2.2.4 Setting up arm poses

Now, let's define the robot poses. Click Add Pose and add the following poses in the following format (Posename : arm_base_joint, shoulder_joint, bottom_wrist_joint,

elbow_joint, top_wrist_joint):

- Straight: 0.0, 0.0, 0.0, 0.0, 0.0
- Home: 1.5708, 0.7116, 1.9960, 0.0, 1.9660



Figure 5: Robot arm pose

We don't have an end effector, so we can skip this step

2.2.5 Setting up passive joints

Now, let's define the Passive Joints—those whose joint states are not expected to be published:

	,	Movelti S	etup Assistant	0.00
Start	Define Passive J	oints		
Self-Collisions	Specify the set of passive joints (published for these joints.	not actual	ted). Joint state is not expected to be	
Virtual Joints	Active Joints		Passive Joints	
Planning Groups	Joint Names		Joint Names	8
	1 arm_base_joint		1 front_left_wheel_joint	
Robot Poses	2 shoulder_joint		2 front_right_wheel_joint	Non-
End Effectors	3 bottom_wrist_joint	>	3 rear_left_wheel_joint	
LINE ETTECTORS	4 elbow_joint		4 rear_right_wheel_joint	
Passive Joints	5 top_wrist_joint		the shows the body is added for the effective interaction in the biology of	
ROS Control	6 front_left_wheel_joint			
The second of	7 front_right_wheel_joint			
Simulation	8 rear_left_wheel_joint	e		
3D Perception	9 rear_right_wheel_joint			
Author information				
Configuration Files				

Figure 6: Movelt passive joints

Now, it's time to check the ROS controllers we set up with the robot URDF.

2.2.6 Setting up ROS controllers

Now, we need to connect our robot with Moveit for manipulation through the ROS controllers we defined. Click ROS control, then click Auto Add FollowJointsTrajectory Controllers For Each Planning Group. You should see the controller being automatically ported in, as shown here:

	Movelt! Setup Assistant	
Start	Setup ROS Controllers	
Self-Collisions	Configure Movelti to work with ROS Control to control the robot's physical hardware	
Virtual Joints	Auto Add FollowJointsTrajectory Controllers For Each Planning Group	
Planning Groups	Controller Controller Type	
a manual strates	 arm_controller FollowjointTrajectory 	
Robot Poses	Joints arm base Joint shoulder laint	
End Effectors	bottom wrist_joint elbow joint	
Passive Joints	top_was_joint	
ROS Control		
Simulation		
3D Perception		
Author Information		
Configuration Files	Expand All Collapse All Add Controller	

Figure 7: Movelt setup ROS controllers

The FollowJointTrajectory plugin we had called upon in our plugin is shown in the preceding screenshot. Now, let's finalize the Moveitconfig package.

2.2.7 Finalizing the MoveitConfig package

The next step will autogenerate a URDF for simulation:

1. In case you made any changes, these changes will be highlighted in green. We can skip this step as we didn't change anything.

2. We don't need to define a 3D sensor, so skip this step as well.

3. Add any appropriate information in the Author Information tab.

4. The final step is the **Configuration Files**, where you will see a list of files that have been generated. The window is as follows:



Figure 8: Configuration files

5. Give a configuration name such as robot_description_moveit_config, click on Generate Package, and exit the setup assistant.

Now, let's control the robot arm using Moveit.

3. Controlling robot arm using Movelt!

Once Moveit has been configured, we can test our robot arm manipulation using the GUI interface (RViz plugin):

1. Launch the mobile manipulator in Gazebo:

\$ source devel/setup.bash

\$ roslaunch my_robot mobile_manipulator_gazebo_xacro.launch

2. In a new Terminal, open the move_group.launch file that was auto-generated by the Moveit setup assistant wizard:

\$ source devel/setup.bash

\$ roslaunch robot_description_moveit_config move_group.launch

Your Terminal's output would be similar to what's shown in the following screenshot:



Figure 9: move_group.launch

3. Now, let's open RViz to control the robot's motion:

\$ source devel/setup.bash

\$ roslaunch robot_description_moveit_config movit_rviz.launch config:=True

You should see the following window.

					mayell.ivit* - Ryle	
Elle Banels Help						
Conterast @ Move C	amera 🛄 Select					
Displays Gobal Options Gobal Status: Ok Gold Gold				2		
Add					Array -	
1 MillionPlanning					•	
Context Planning	Manipulation	Scene Objects	Stored Sce	mes R	red States	
Planning Library						
Planning Library OMPL	p	lanner Parameter			- Chine	
Planning Library OMPL BRTstar		lanner Parameter				
Planning Library OMDL BRTStar	*) *)	lanner Parameter				
Planning Library OMDE MRTStar Warehouse	, j	lanner Parameter				
Planning Library CMPL BRTStar Werehouse Host: 127.9.0.1		lanner Parameter	Port	11629 :	Converz	
Planning Library CMPL MRTstar Werehouse Host: 127.8.0.1 Workspace	-1	lanner Parameter) Port: (23829 ;	Connect	
Planning Library OMPL MRTstar Warehouse Host: 127.8.0.1 Workspace Center (XYZ):	* *	lanner Parameter	Port (33829 :		

Figure 10: rviz Movelt launch

4. Go to the Planning tab, select home in Goal State, and click on Plan. You should see a visual of the robot arm planning (moving) to the target position.

Congratulations! Now you know how to build a robot using URDF/XACRO, configure and control it using Movelt! That's a great achievement.