IROS 2011 Tutorial:

Motion Planning for Real Robots

Sunday, September 25, 2011

URL: http://kavrakilab.org/OMPLtutorial

Abstract

This full-day tutorial will teach both novice and experienced participants how to setup, configure and use motion planning on a real robot. Novice users can expect to learn how to set up, configure and execute the perceptual, kinematic, planning and execution components required for motion planning on an advanced multiple degree of freedom robot. Expert users will be able to explore the motion planners in more details, focusing on how they can be reconfigured for particular tasks. The tutorial will be based on a set of tools within the OMPL (Open Motion Planning Library) and ROS (Robot Operating System) software. The participants will have access to simulated environments and real robots (the Willow Garage PR2 robots) for a hands-on experience in using motion planning with real robots. The tutorial will conclude with an examination of case studies based on suggestions from the participants and organizers, highlighting how the motion planners can be configured for particular robots or motion planning scenarios.

All tutorial material will be made available online after the tutorial at http://kavrakilab.org/OMPLtutorial.

Organizers

Sachin Chitta and **E. Gil Jones**
Willow Garage, Inc.
68 Willow Road
Menlo Park, CA 94025
650-475-2700
{sachinc,gjones}@willowgarage.com

Ioan Şucan, **Mark Moll**, and **Lydia E. Kavraki**
Rice University
Department of Computer Science, MS 132
Houston, TX 77005
713-348-4834
{isucan,mmoll,kavraki}@rice.edu

1. Schedule

08:30–09:00	Overview and Introduction
09:00–09:30	Background on concepts in sampling-based motion planning
09:30–10:15	The Open Motion Planning Library (OMPL)
10:15–10:45	*Coffee break*
10:45–11:15	Introduction to ROS and connection to OMPL
11:15–12:00	Overview of the simulation environment
12:00–12:10	Live demo on PR2
12:10–13:30	*Lunch*
13:30–15:00	Hands-on programming of PR2 by participants, part I
15:00–15:30	*Coffee break*
15:30–16:00	Hands-on programming of PR2 by participants, part II
16:00–16:30	Discussion with participants and wrap-up

2. Motivation and objectives

Motion planning is easy to understand, yet state-of-the-art algorithms to solve motion planning problems in a general fashion can be hard to implement. Furthermore, integrating motion planning algorithms in a bigger software system targeted at specific robots is also challenging. The OMPL library implements many sampling-based algorithms and makes it easy to integrate with larger software systems and tailor to specific systems. ROS provides a very rich software infrastructure with perception, kinematics and execution components that can be integrated with planning to create a complete motion planning and execution pipeline. The tutorial aims to provide a high-level description of the motion planning algorithms in OMPL coupled with implementation level details on configuring motion planners on real robots. The tutorial will provide plenty of opportunity for participants to get hands-on experience with solving motion planning problems in real-world environments, both in simulation and on the robot using real sensor data.

After the tutorial participants should be able to:

- define motion planning queries and solve them with a planning algorithm,
- visualize the results,
- use the sensor data and environment models that are accessible through ROS interfaces for motion planning, and
- solve and execute motion planning queries for the PR2 hardware platform.

The skills obtained in this tutorial are easily transferrable to the rapidly growing list of other robots on which ROS can run (see http://www.ros.org/wiki/Robots).

3. Primary/secondary audience

We are primarily targeting participants who would like to learn to implement motion planners on real robots using realtime sensing. Some familiarity with ROS is desired but not essential. This will be a hands-on tutorial, so programming experience in C++/Python is desired. A secondary audience is researchers in motion planning who would like to build on the tools and components available in OMPL and ROS to create more advanced motion planners. This tutorial will also be of interest to educators wanting to use a stable, well-featured software tool for teaching motion planning.

4. Overview and Introduction

This tutorial introduces you to the motion planning and arm navigation components in the ROS and OMPL libraries. OMPL is a powerful motion planning library that offers a range of randomized planners for use with robotic systems. ROS is an open-source software system that offers implementations of several capabilities important for robot operation including perception, motion planning, control, grasping, execution, etc.

This tutorial uses a series of examples and exercises to teach you how to configure the motion planning components for your own robot and use them in actual operation for executing tasks. All the examples use the latest set of tools in the *electric* release of ROS and will use the Willow Garage PR2 robot as a canonical example. These tools have been tested (in simulation) on a wide variety of robot models and should be easily applicable to your own robot as well.

4.1 Pre-requisites

This tutorial assumes that you have basic familiarity with ROS. Extensive tutorials on ROS can be found on the ROS documentation Wiki http://www.ros.org/wiki. We encourage you to go through as many of the tutorials and examples on that page as possible before you proceed with this handout.

4.2 Computational resources

The motion planning libraries in ROS are best experienced on a decently powerful computer. In particular, ROS is known to be incompatible with certain graphics cards. A full (unofficial) list of compatible computers is available on the ROS wiki http://www.ros.org/wiki/simulator_gazebo/SystemRequirements.

4.3 ROS

ROS is best experienced (currently) on a Ubuntu system. Detailed instructions for installing ROS on different versions of Ubuntu are on the ROS installation page (http://www.ros.org/wiki/electric/Installation/Ubuntu). We recommend an Ubuntu version of *Lucid Lynx* or higher. Follow the instructions on that page to start installing ROS. Make sure to install the *pr2-desktop* variant of ROS Electric using the following command:

```
sudo apt-get install ros-electric-pr2-desktop
```

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/electric/setup.bash" >> ~/.bashrc
. ~/.bashrc
```

4.4 OMPL

The installation instructions for ROS mentioned above also pull in the *electric* version of the OMPL package. The *ompl_ros_interface* package (which is also installed by the above command) provides the necessary ROS interface to OMPL.

4.5 Getting started

To get started, login to your machine. For this tutorial, you will first need to create a directory where you can store all your work. You will need to make sure this directory is known to ROS by putting it in the ROS_PACKAGE_PATH. Define a directory named *ros* in your home directory. Update the ROS_PACKAGE_PATH environment variable to include this new location. For bash, you can do this by using the following command:

```
export ROS_PACKAGE_PATH=$HOME/ros:$ROS_PACKAGE_PATH
```

It's best if you also add this line at the bottom of the .bashrc file using the following command:

```
echo "export ROS_PACKAGE_PATH=$HOME/ros:$ROS_PACKAGE_PATH" >> ~/.bashrc
. ~/.bashrc
```

Now, define a *ROS stack* inside the *ros* directory. This is a place where you will keep all your work. For this tutorial, we will create a ROS stack named *iros_2011_tutorials*.

```
mkdir -p ~/ros
cd ~/ros
roscreate-stack iros_2011_tutorials
```

Try to *roscd* to the stack that you just created.

```
roscd iros_2011_tutorials
```

If you find yourself in the iros_2011_tutorials directory, you are almost ready to begin working. There is just one additional step.

4.6 Download a copy of the tutorials repository

We have placed a set of resources in a central repository that you can download. (Or you can copy them from the media that the organizers have). If you choose to download from the repository, first cd into the *ros* directory you created and execute the following command:

```
svn co https://kforge.ros.org/armnavigation/iros_tutorial iros_tutorial_resources
```

The resources included in this directory include mesh files for objects, pre-built databases containing planning scenes you can play with, description files for robots other than the PR2, etc.

5. Configuring arm navigation for your robot

Your first exercise in this tutorial is to use the tools in ROS to generate configuration files for the PR2 robot. The steps in this exercise are the ones you will have to follow later when you want to configure these stacks for use on your own robot.

5.1 URDF — Describing your robot

The first step in using any of the tools in ROS with your robot is to represent your robot in a format that will be understood by all the ROS components. This format in ROS is called the Universal Robot Description Format (URDF). In this tutorial, we will use the (already existing) URDF representation of the PR2 robot. The URDF format can be used to represent most robots with a tree-like structure and includes full kinematic, visual, collision and inertial information for the robot. It can also be used to represent sensors on the robot. Extensive tutorials on the URDF are available on the ROS wiki http://www.ros.org/wiki/urdf.

We will use the example URDF for the PR2 that is already available from the *planning_models* package in ROS. The URDF for the PR2 is specified in a XML format and is contained in the *test_urdf/robot.xml* file in the *planning_models* package. Have a look through the file if you are interested, but note that the URDF file for the PR2 is pretty complicated and not easily human-readable.

5.2 The Arm Navigation Wizard

The tool that you will use for this exercise is the ARM NAVIGATION WIZARD. The ARM NAVIGATION WIZARD is a GUI that makes the configuration of the arm navigation components for your robot very easy. You will use it configure the arm navigation components for the PR2 robot.

Step The first step is to launch the wizard. To launch the wizard, open up a terminal and enter the following:

```
roslaunch planning_environment planning_description_configuration_wizard.launch \
urdf_package:=planning_models urdf_path:=test_urdf/robot.xml
```

This will bring up the ROS visualizer (Rviz) in a separate window with the proper configuration, and a window containing the wizard itself. In Rviz, you should see your robot's visual URDF model and a planar grid. The wizard is designed to run alongside Rviz, and interacting with the wizard will display markers in Rviz.

5.3 Using The Wizard

The wizard is a graphical program which should guide you through the automatic generation of a set of configuration and launch files for your robot. There are two decisions you must make on the initial screen of the wizard: choose a configuration mode and a sampling density (for configuring collision checking).

5.3.1 Configuration Modes

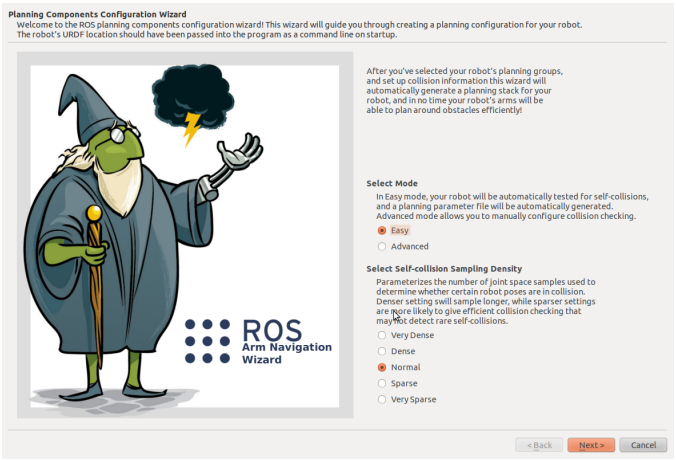
The wizard has two modes: an "Easy" mode, and an "Advanced" mode. In "Easy" mode, the wizard will automatically generate configuration files without much intervention on your part, while the "Advanced" mode allows you to manually configure a number of different parts of the configuration process.

Step For this part of the exercise, choose the "Easy" mode.

5.3.2 Levels of Sampling

To generate configuration files for your robot, the wizard will automatically sample a large number of joint states (uniformly at random), and check the robot for self collisions in these states. With more samples, the wizard can generate a more accurate picture of your robot's kinematic properties. The wizard provides five sampling densities, from "Very Dense" to "Very Sparse." To get the most accurate results you can use a denser sampling, but the program can take a very long time to run (on the order of hours). Sparser sampling methods will go very quickly, from minutes down to seconds.

Step For this part of the exercise, choose the "Normal" setting for sampling density.



5.3.3 Kinematic Chain Planning Groups (Easy Mode)

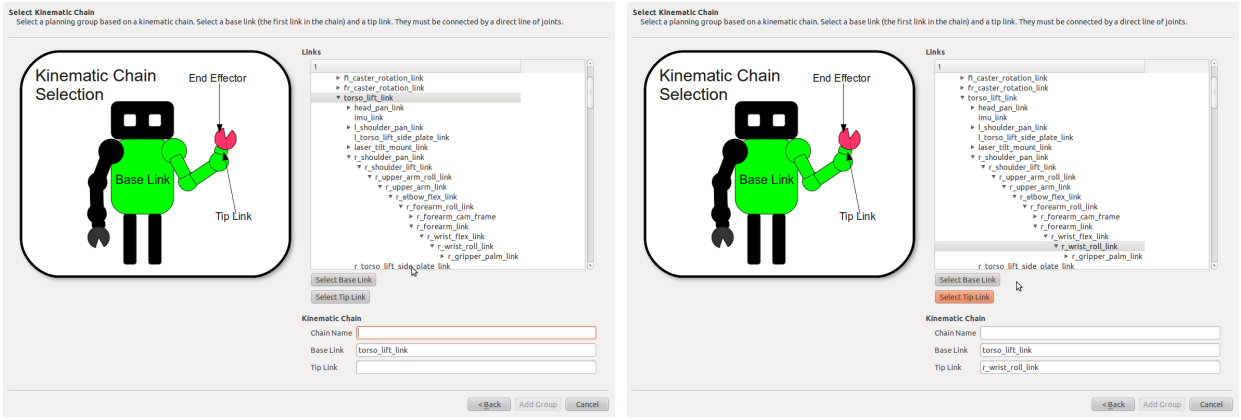
In either mode the most important part of the configuration is creating your planning groups. Planning groups are groupings of parts of your robot that you are interested in using in arm navigation applications. For instance, if you have a single arm on your robot with an end-effector on the end, that arm will need its own planning group.

Once you've selected a configuration mode and a sampling mode, you can move on to selecting planning groups by pressing "Next." The wizard provides two methods of creating planning groups: Kinematic Chains and Joint Collections. You must have at least one valid group to proceed through the wizard - if you don't have a valid group the "Done with groups" button at the bottom will be greyed out. Note that most robots will typically have kinematic chain groups - i.e. a serial chain of links and joints that represent a serial-chain manipulator. The current arm navigation components may have trouble generating plans for arbitrary joint collections. Setting up joint collection groups may be useful for your end effectors, however, as it gives a useful shorthand for the set of links associated with the end effector.

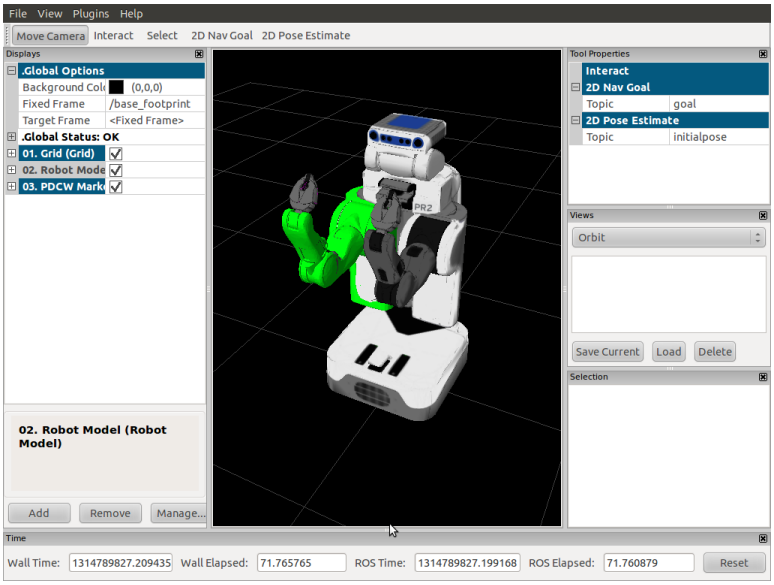
Step First choose to add a group using kinematic chains by clicking on the Add Kinematic Chain Group button.

In this exercise, you will create planning groups for the right and left arms of the PR2 robot by defining a kinematic chain for them. A kinematic chain is configured by selecting a base link and a tip link.

Step Now add the right arm as a group.Using the drop down representation on the right hand side, first click on and select the *torso_lift_link*. Now press the *Select Base Link* button to choose this link as your base link. Go down the drop-down list following the links starting with *r_* until you find the *r_wrist_roll_link*. Press the *Select Tip Link* button to choose this link as your tip link. Type in *right_arm* for the name of the group and press the *Add Group* button.



Step You should also be able to see the selected group in Rviz. Set the alpha for the robot model to 0.5 so you can see the selected group better. (Expand the "Robot Model" dialog in the left-hand menu, click on the value next to the "Alpha" entry and modify it.)



Step Use the "Add another kinematic chain" button and add the left arm as a group with name *left_arm*, base link *torso_lift_link* and tip link *l_wrist_roll_link*.

5.3.4 Joint Collection Planning Groups (Easy Mode)

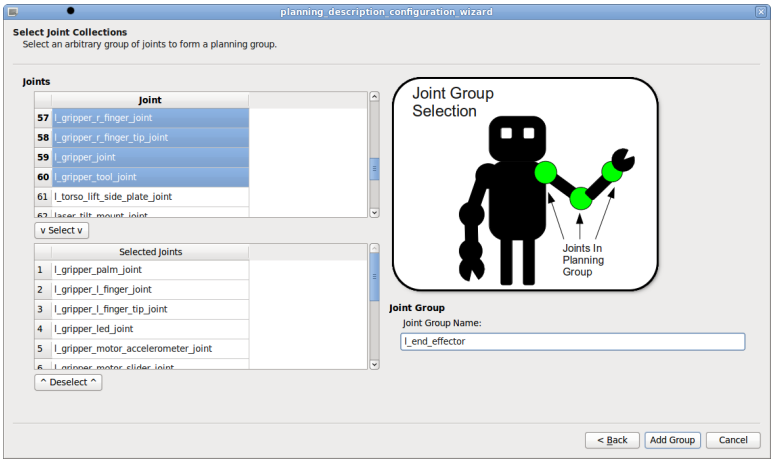
Next we will use the Joint Collection Planning Group mode to configure the groups associated with the end effector.

Step Use the "Return to Groups" button to return to the Planning Group Setup page, and then select "Add Joint Collection Group".

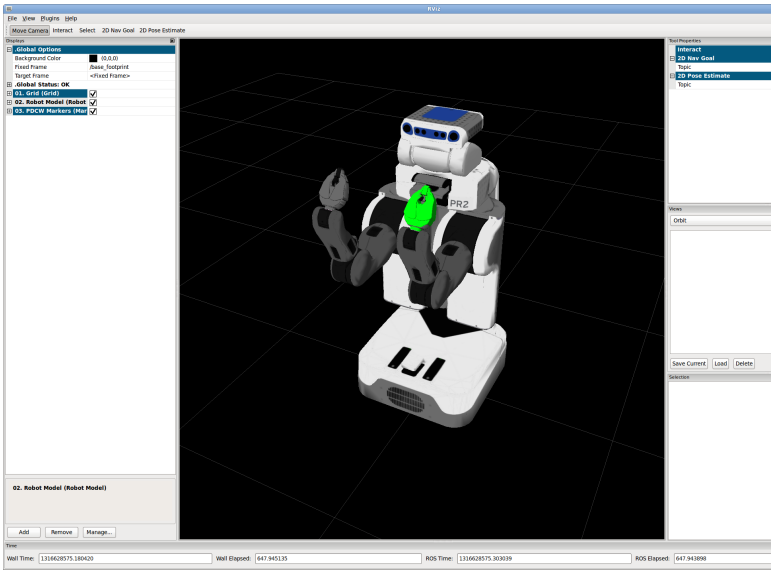
In this dialog we select a set of joints that belong to a group rather than specifying a base and tip link. We can set any arbitrary set of joints in this dialog, but the planner may not be able to generate plans for that set of joints. We are going to use this kind of group to specify the joints that are associated with the robot's end effectors - these groups will allow us a convenient shorthand for reference.

Step Find and select all joints that begin with *l_gripper_* in the upper left hand joint list. This should include 11 joints - from Joint 50 - *l_gripper_palm_joint* to Joint 60 - *l_gripper_tool_joint*. Now click the "Select" button and the list of 11 joints should be shown in the lower left window. Name the group *l_end_effector*, and click the "Add Group Button".

The wizard window should look like this:



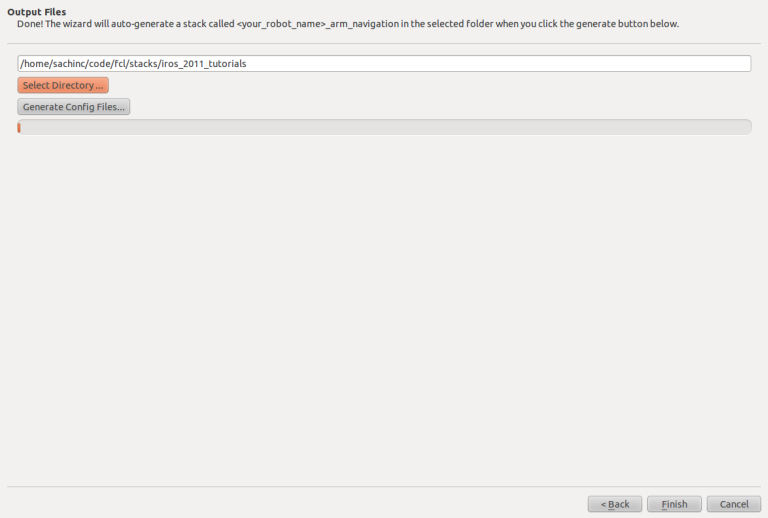
And Rviz will show the group links in green:



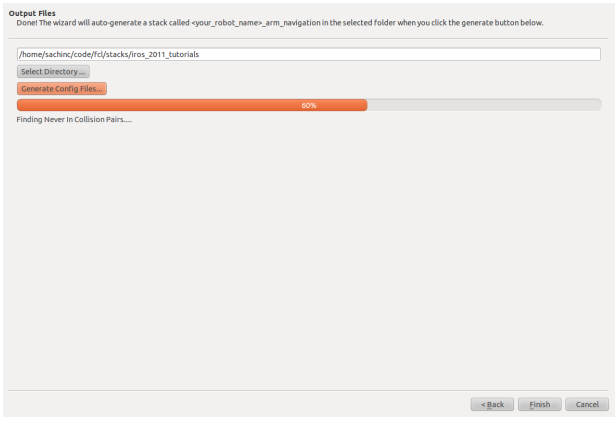
Step Click the "Add Another Joint Collection" button, use the Shift button to select all joints in the lower left window, and hit the "Deselect" button. Now select Joint 75 - *r_gripper_palm_joint* to Joint 85 - *r_gripper_tool_joint*. Click the "Select" button and the list of 11 joints should be shown in the lower left window. Name the group *r_end_effector*, and click the "Add Group Button"

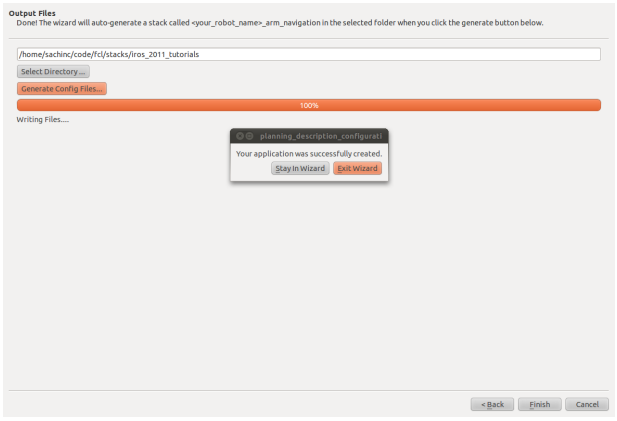
5.3.5 Application generation (Easy Mode)

Step You are now ready to generate all the configuration files for the PR2 left and right arms. Press the "Done Adding Groups" button. You will now be prompted for a directory where you can store the generated configuration files. Click the "Select Directory" button and choose the stack directory that you created at the very beginning of the tutorial.



Step Press the "Generate Config Files..." button and wait for the generation to finish. When the generation is done, click the "Exit Wizard" button to exit. Also press "Ctrl-C" in the terminal where you launched the wizard to bring down all components.





The config generation will generate a ROS package named *pr2_test_arm_navigation* in your *iros_2011_tutorials* stack. This package will contain two directories - *config* and *launch*. These two directories contain all the configuration and launch files for you to use with the ROS arm navigation system. The *config* directory includes the following three configuration files:

- *joint_limits.yaml* - contains velocity and acceleration limits for the joints of the robot. You can edit these by hand if necessary to get better performance.
- *pr2_test_planning_description.yaml* - contains definitions of multi-dof joints that might not be in the URDF. Also contains the definitions of the four groups that you just defined. Also contains the list of collision operations which are used to make collision checking function correctly and efficiently.
- *ompl_planning.yaml* - the configuration files for the OMPL planners.

The *launch* directory includes launch files for:

- *pr2_planning_environment.launch* - a node that maintains information about the environment of the robot.
- *constraint_aware_kinematics.launch* - a node that launches kinematics for the two arms.
- *ompl_planning.launch* - the planning node
- *move_right_arm and move_left_arm.launch* - nodes that launch the state machines for planning and execution
- *trajectory_filter_server.launch* - a node that filters trajectories and smoothens them.

In the next section, you will learn how to use these launch files for motion planning!

Check Make sure you have brought down the ARM NAVIGATION WIZARD before you proceed to the next section.

5.3.6 More resources

For more information on the contents of this section and to tweak some of the parameters and configuration files, have a look at the "Understanding and adjusting the auto-generated arm_navigation application" link on the ROS website (http://www.ros.org/wiki/arm_navigation/Tutorials/).

6. Motion planning for your robot

In this section, you will learn how to use the launch files generated for arm navigation with the PR2 robot. You will use a handy new tool in ROS Electric to generate motion plans for the arms of the PR2 and move in a collision-free manner to both joint and pose goals in an environment where you have placed multiple obstacles. The tool that you will use in this section is called the PLANNING COMPONENTS VISUALIZER.

6.1 Launch the visualizer

Launch the visualizer using the auto-generated file that was generated by the Arm Navigation Wizard (note that *pr2_test_arm_navigation* was the name of your auto-generated package of launch and config files for the PR2).

```
roslaunch pr2_test_arm_navigation planning_components_visualizer.launch
```

This will bring up the Planning Components Visualizer along with a special menu that will let you interact with the robot. The Planning Components Visualizer uses interactive markers for user input. To use interactive markers, press the button on the top of the Rviz window which says "Interact" (Figure 1). When you do this, a set of markers will appear on the end effectors and on top of your robot that you can interact with.

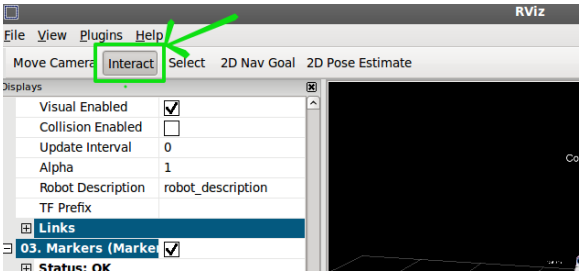


Figure 1: Interacting with the robot model in rviz.

6.2 Top level menu

The top level menu is a selectable text label floating above your robot ("Command...") right click on the top level menu to access its capabilities (Figure 2). The top level menu has the following options:

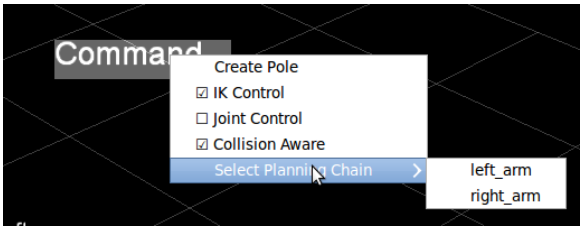


Figure 2: The top-level menu.

- Create Pole: This option will create a collision object (a pole) a few meters from the robot. This allows you to create a virtual world in which there are a bunch of obstacles that the robot needs to avoid. The pole can be moved around the environment, selected, and deselected as desired.
- IK Control: when selected, IK (Inverse Kinematics) control option creates an interactive, posable marker at the end effector of the selected planning group.

- Joint Control: when selected, the Joint Control option allows you to manually pose each degree of freedom in the selected planning group of your robot.
- Collision Aware: when selected, the Collision Aware option tells the IK solver to consider self collisions and environmental collisions before generating IK solutions.
- Select Planning Chain: This submenu allows you to select one of the planning groups in your robot.

6.2.1 Pole creation

Step Create a pole in the environment using the "Create Pole" option in the top-level menu. If you did not choose a different position for the pole, the current position will default to (0,0,0), i.e. the pole will appear underneath the robot. Select the pole and move it around using the interactive markers attached to the pole. Deselect the pole by right-clicking on any of the interactive marker controls associated with the pole.

6.2.2 End effector control

End-effector control can be accessed either through the top-level menu by right-clicking and selecting a planning chain or by clicking on the box displayed around the two end-effectors for the right and left arms (Figure 3). Selecting a planning chain or clicking on the end-effector display a set of interactive controls for the end-effector frame that can be used to move the frame around.

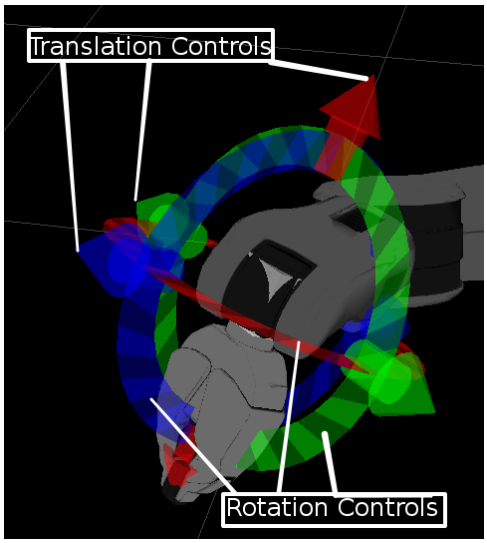


Figure 3: End effector interactive marker controls.

Step Use the end effector controls to move around the end-effector for the right arm. Notice how, when the desired position of the right gripper is too far away from the robot, the gripper separates from the robot and is displayed in red. This indicates that no inverse kinematics solutions exist for the pose you have dragged the gripper to (Figure 4).

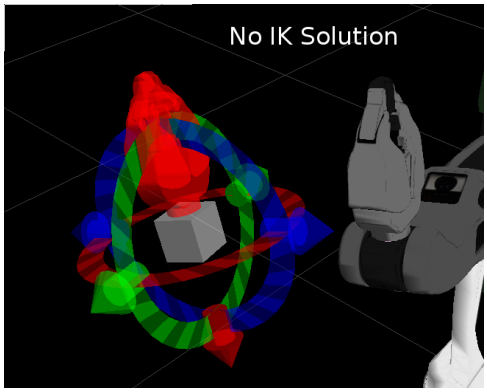


Figure 4: Display when no IK solution exists.

Step Now try to move the right gripper into the left arm of the robot. You will notice that the same thing happens, i.e. there is no inverse kinematics solution for the desired pose of the end-effector that is not in collision with the left arm (Figure 5). The inverse kinematics solver tries to find an inverse kinematics solution that is also collision free, i.e. a solution that has no self-collisions or collisions with the environment.

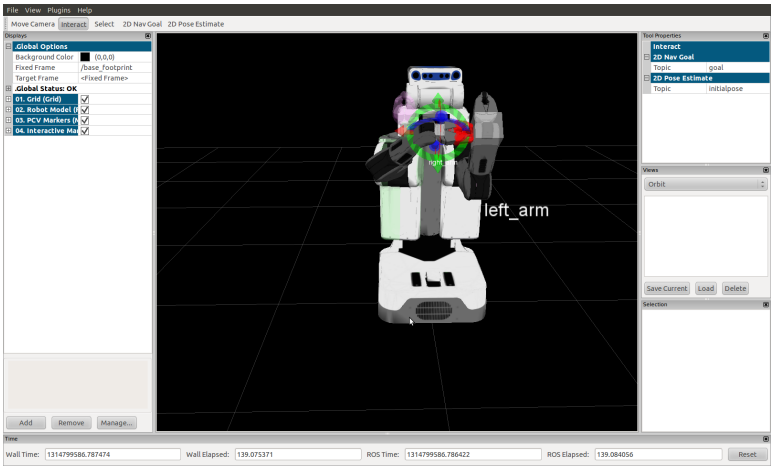


Figure 5: Robot state is in self-collision.

Step Now, go back up to the top-level menu ("Command") and uncheck the "Collision aware" flag. Try to move the right arm gripper back into collision with the left arm. You will see that the gripper is no longer red - collisions are no longer being checked for and so the solver is only looking for valid inverse kinematics solutions for the pose of the gripper.

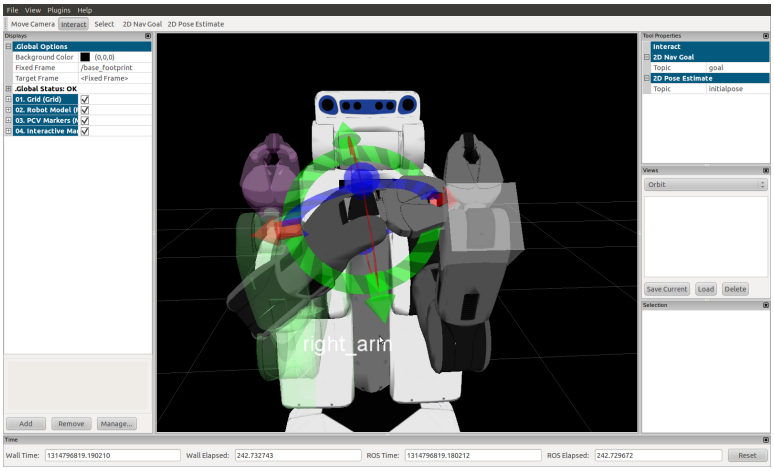


Figure 6: Collision constraints are now being ignored.

6.2.3 Motion planning

To motion plan, we will need to set the start state and the goal state.

Step Right click on the interactive controls on the end-effector and check the check-box for "Set End Position" (if its not already checked). Move the end-effector to a desired goal position.

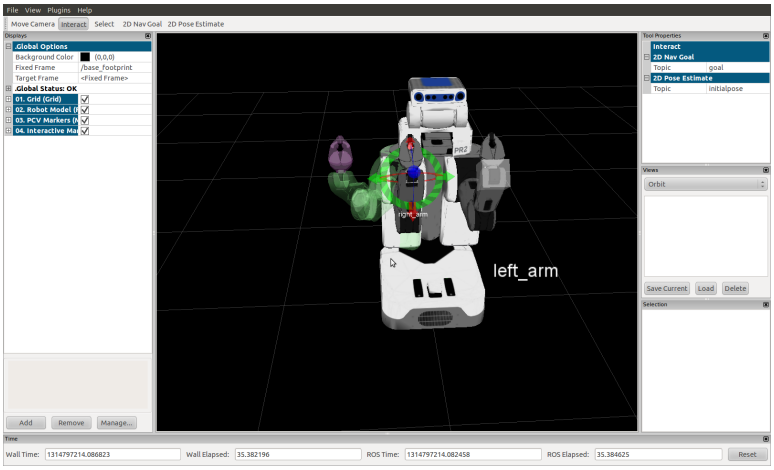


Figure 7: Setting the goal position.

Step Now, right click any of the interactive controls on the end-effector and select the "Set Start Position" box. As soon as you do this, the visualization of the arm that you were moving around will become inactive and another visualization will appear. The inactive visualization represents the goal that the planner will plan to while the new visualization will be used to define the start configuration.

Step Use the new visualization and move the arm around to a start state. Note how states that are in collision will automatically turn red. Now, right click again on the interactive controls and select "Plan". You should see a visualization of the plan being played back. The plan is generated using the OMPL motion planners integrated in ROS.

Step Introduce more poles into the environment and play around by moving the start and goal states to different configurations in the environment. Try to generate and visualize plans in different parts of the environment.

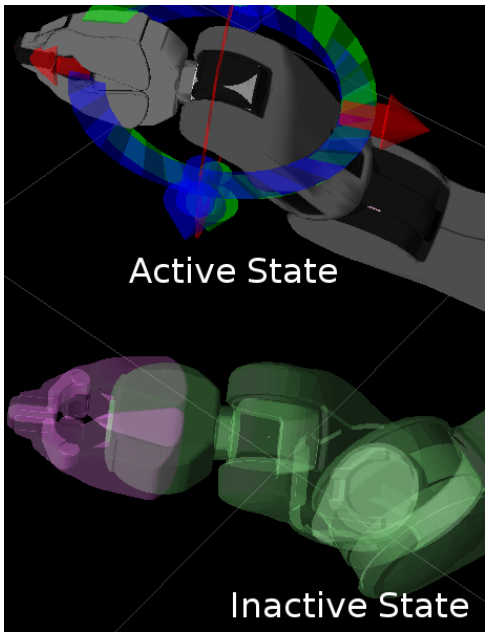


Figure 8: Setting the start position.

6.2.4 Trajectory filtering

As you experiment with the planner, you will notice that the plans coming out might be jagged and not very smooth. The trajectory filter will attempt to smooth out the plans by randomly *short-cutting* them with smooth cubic splines.

Step Select different start and goal states and plan for them as above. Now, after viewing the plan, click on the "Filter Trajectory" option in the right-click menu from the interactive controls for the end-effector. Look for differences between the plan and the new path generated by the trajectory filter. The new path should be shorted and substantially smoother.

6.2.5 Joint control

Selecting the "Joint control" option from the top-menu (Command) will also let you control the joints of the arms directly using interactive control markers on the joints. Revolute joints can be rotated while prismatic joints will translate. The motions of the joints will obey joint limits.

Step Exit the program by using Ctrl-C in the window where you opened the program.

6.2.6 Optional: Impose constraints

You can also impose orientation constraints on the end-effector. The constraints force the pitch and roll of the end-effector to stay within a small range. This is useful if you are holding something in your hands and would like to keep the object level as you move the end-effector around. Note that planning with such constraints is difficult and there is a much less likelihood of finding solutions. You can impose constraints by selecting the "Constrain in roll and pitch checkbox".

Note The treatment of constraints in the GUI is not mature right now and it might be best to skip this part. If you do decide to proceed, note that one of the hardest parts will be to define the goal and start so both obey the constraints. Otherwise, the planner will complain that either the goal or start state does not obey the constraints.

7. Advanced Motion Planning

In this section, you will learn how to use the Arm Navigation Warehouse Viewer for more advanced motion planning tasks. The Warehouse Viewer is an advanced tool that can be used to create, store and replay *scenarios* for motion planning. It provides an easy way to create worlds full of objects by using either primitive shape or mesh representations. It can also be used to store and load world representations created using the sensors on a real or simuated robot.

The Warehouse Viewer allows the user to interactively create and visualize motion plans in such worlds. It can also be used to directly address a simulated or real robot, providing a very nice interactive interface where motion plans and trajectories can be examined in fine detail.

7.1 Configuring the viewer

The Warehouse Viewer can be directly launched for the PR2 robot using a pre-existing launch file but we will not use those files right now. We will instead regenerate this launch file to illustrate the procedure.

Step Generating the required launch file for the robot you worked on in the previous section is very easy:

```
rosrun move_arm_warehouse create_launch_files.py pr2_test
```

This will create a file named planning_scene_warehouse_viewer_pr2_test.launch in the pr2_test_arm_navigation package.

7.2 Database location

By default, all database files will be stored in your home directory inside the ".ros/arm_navigation_dbs" directory. You will not need to edit these files manually. This is also where you should put any new databases that you would like to work with (inside their own directories). We will do this later in the tutorial.

7.3 Running the Warehouse Viewer

Step Start up the viewer using the launch file you just created.

```
roslaunch pr2_test_arm_navigation planning_scene_warehouse_viewer_pr2_test.launch
```

This will bring up the configuration screen for the warehouse viewer (Figure 9). The ROS visualizer (rviz) will start up as well (note that the visualizer might occupy the whole screen - you will have to minimize it to see the viewer). The configuration screen for the warehouse viewer lists a set of default topic and service name that the viewer uses to connect to a simulated or real robot. The default values are working values for the PR2 robot. However, you may have to change them for your own robot.

Accept the default values by pressing the **Accept** button. The Warehouse viewer logo will appear followed by a window that shows you stored database options to plan with (Figure 10). Ignore these options and choose the **New...** button instead. (You will get to play more with these options in the next section.) Pressing the new button should bring up the warehouse viewer.

Your rviz configuration should now look something like in Figure 11. You are now ready to interact with the warehouse. Press the **Interact** button in the top rviz menu to start.

The warehouse viewer allows you to play with multiple parts of the motion planning scenario or *planning scene*. Planning Scenes are the bread and butter of the arm navigation system in ROS. A Planning scene consists of the robot and its current state, the collision objects in the environment, motion plan requests within that planning scene, and trajectories associated with each motion plan request. The planning scene exists to tell planners and collision-aware inverse kinematics solvers what obstacles to avoid, and to provide a method of validating robot states and trajectories to ensure that they are collision-free.

When using the warehouse viewer, you will interact with a single planning scene. You can edit the motion plan requests, trajectories, and collision objects associated with that scene, and then save the scene to a MongoDB database.

Upon starting, the warehouse viewer will create an empty planning scene for you to edit. An empty planning scene consists of the robot in its default state (specified by the URDF if you are not using robot data, or the robot's current joint state if you are using robot data), and a world free of collision objects.

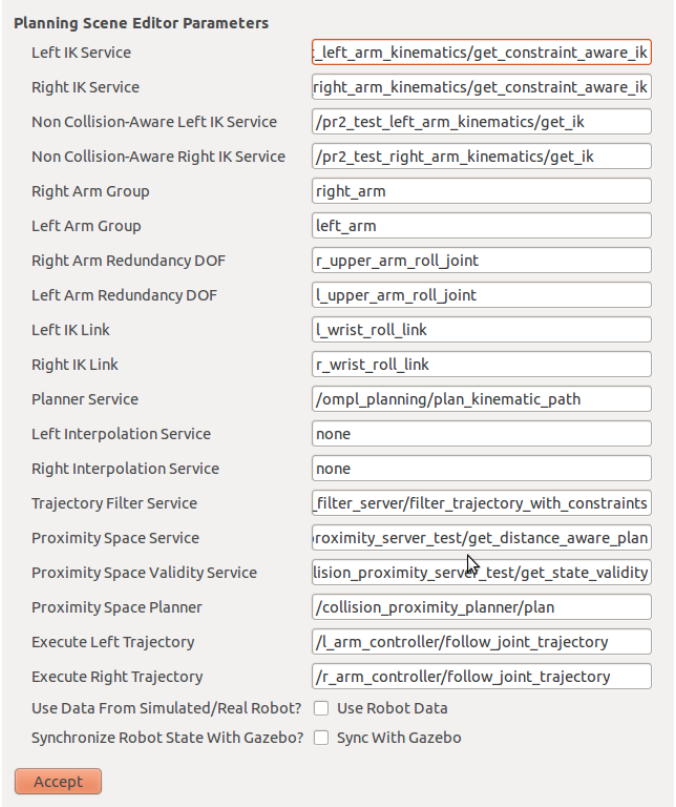


Figure 9: Initial configuration screen for the warehouse visualizer.

You can make a new planning scene at any time by using the menu command: **File->New Planning Scene**. This will prompt you to save the current planning scene before creating a new, (empty) one. The first part of the planning scene you will play with is the motion plan request itself. The request typically specifies a start and goal state that you want to plan between.

7.4 Create a new motion plan

We will first create a new motion plan using the warehouse viewer. Use the File menu in the viewer to create a new motion plan request using the command: **File->New Motion Plan Request...** (Figure 12). A new window will popup asking you to specify which group you want to motion plan for. Press **Create...** to accept the default group offered, the right_arm (Figure 12).

In the rviz window, you will see interactive controls appear for the group you have selected. Similar to the exercise you did using the Planning Components Visualizer, you can move the group around to a start and goal positions. Note that there will be two sets of controls, one for specifying the start state and the other for specifying the end (goal) state. Once you are satisfied with the start and goal state (Figure 13), you can ask for a motion plan (Figure 13) in two ways:

1. Right-click on the the interactive controls for either the start or the goal. Select the **Plan New Trajectory** option.
2. Use the **Plan New Trajectory** button at the bottom of the warehouse viewer.

You should be able to see the trajectory visualized in rviz. Once the trajectory is planned, it will also be added to the warehouse and will appear in the **Trajectories** tab of the warehouse viewer (Figure 14).

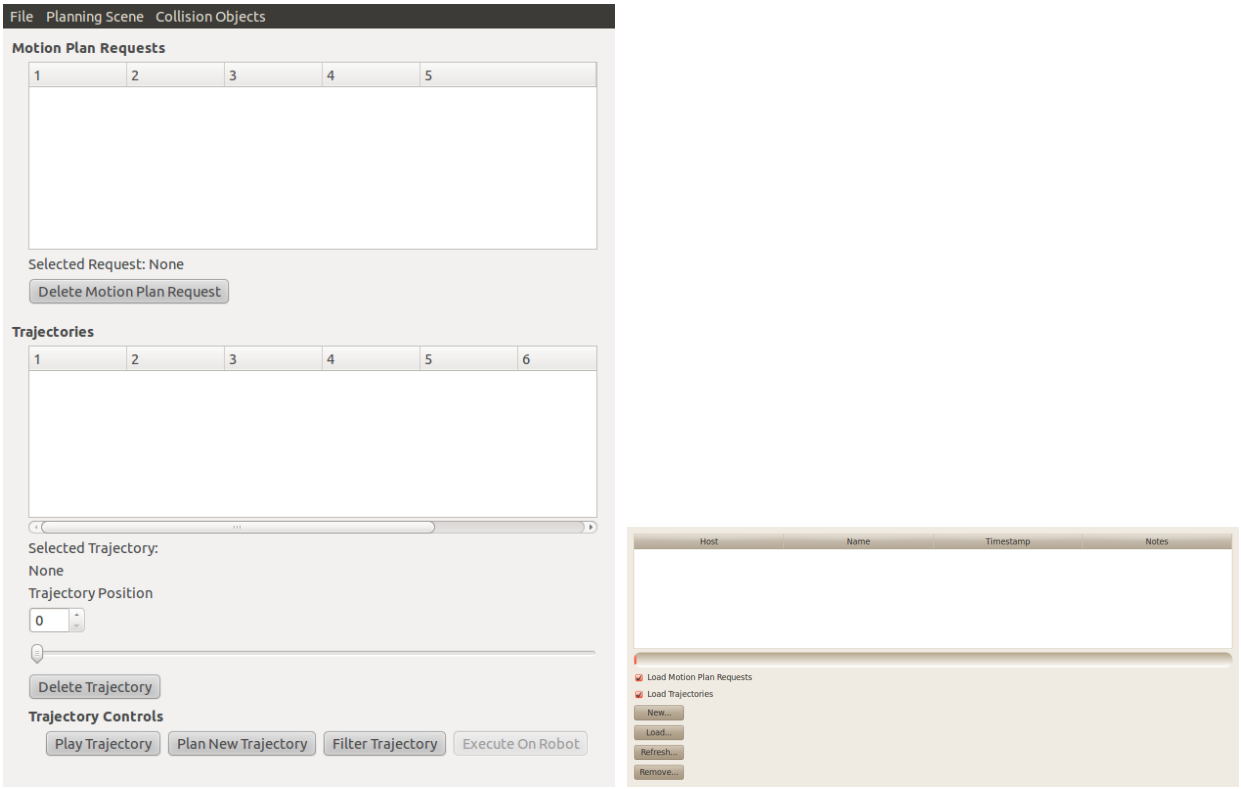


Figure 10: Left: Warehouse viewer start screen. Right: Stored planning scenes from the database,

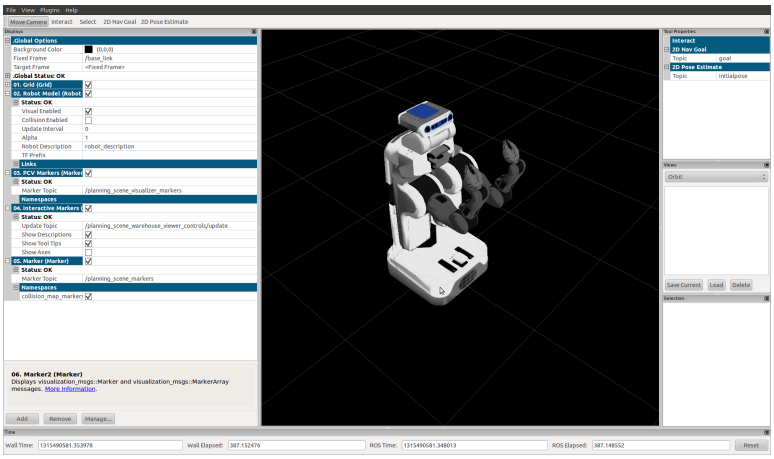


Figure 11: Interacting with the warehouse through rviz.

7.5 Visualizing planned trajectories

Expanding the trajectory tab will show you more information about the trajectory, including its source and generation time. You can also toggle the display of collisions in the trajectory (if they exist), set the type of rendering and the color of the display (Figure 15). Change the type of rendering to a **Padding Mesh** to see the mesh with extra padding that is used for collision checking by the motion planner. Viewing this mesh helps in determining when collisions are happening but it is expensive to display so switch the rendering back to **Collision Mesh** when you are done.

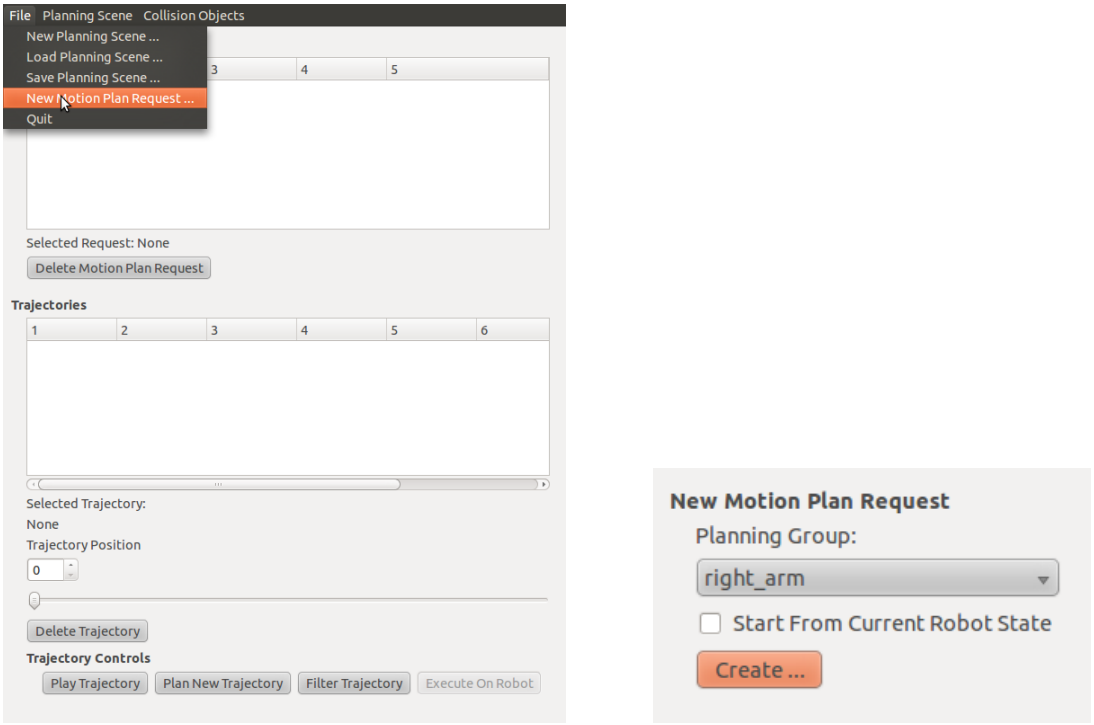


Figure 12: Creating a new motion plan.

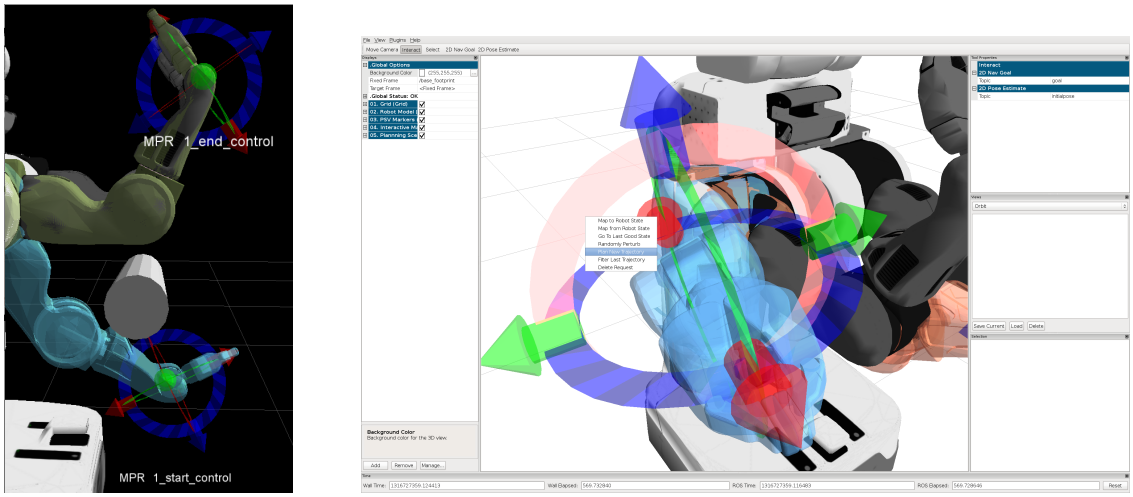


Figure 13: Left: Start and goal interactive displays, Right: Ask for a new motion plan.

You can now step through and examine the trajectory in more detail using the slider bar (Figure 16).

Note that a new display appears in Rviz showing the position of the arm as you step through the trajectory. Note that there are now four displays corresponding to the right arm in rviz - the solid display representing the actual robot, two interactive displays representing the start and goal states and a fourth display showing the progress of the trajectory. You can make the trajectory display invisible using the **Visible** checkbox in the **Trajectory display** of the warehouse viewer.

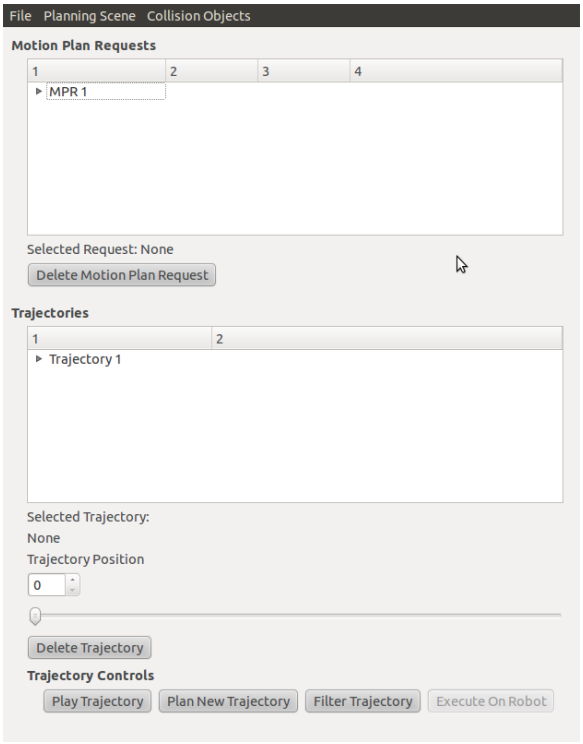


Figure 14: The trajectories tab provides information about trajectories.

7.6 Deleting motion plan requests and trajectories

Click on the name of the motion plan request (which should be "MPR 0") to select it. The selected request name will appear near the bottom of the motion plan request box. In the image above, the selected request is "MPR 0". At any time, you can delete the selected request by pressing the "Delete Motion Plan Request" button. Deleting the request will also delete the associated trajectory. You can also delete trajectories separately by choosing the trajectory and deleting it from the **Trajectory** tab.

7.7 Filtering motion plans

Many planners rely on a post-processing, or "filtering" stage to smooth the resulting planner trajectory and make it executable by an actual robot. Usually, these randomly interpolate collision-free cubic splines between random pairs of points in the trajectory, resulting in a much smoother, much slower trajectory with many more points. Trajectory filtering usually takes much longer than planning. While the OMPL planner may take a few milliseconds to produce a plan, the Trajectory filter can take up to two seconds to process the resulting planned trajectory.

To produce a filtered trajectory, either right click on the IK controller in RVIZ of the desired motion plan request and press "Filter Last Trajectory" or select a trajectory in the trajectory box of the warehouse viewer, and press the "Filter Trajectory" button. After a couple of seconds, a new trajectory will be saved to the trajectory box. If filtering failed, a relevant error code will be saved to the trajectory outcomes table defined next.

7.8 Viewing outcomes

An Outcome is an error code associated with a particular feature of the planning scene. For instance, each trajectory in the planning scene is associated with a particular error code such as Success, Planning Failed, Collision Constraints Violated, etc. In addition, each Pipeline Stage is associated with an error code. A Pipeline Stage is an arbitrary name defined by the application using the arm navigation stack. For example, the move_arm_simple_action program has several pipeline stages, such as "before ik", "after ik", "planner", "filter", "monitor", and so on.

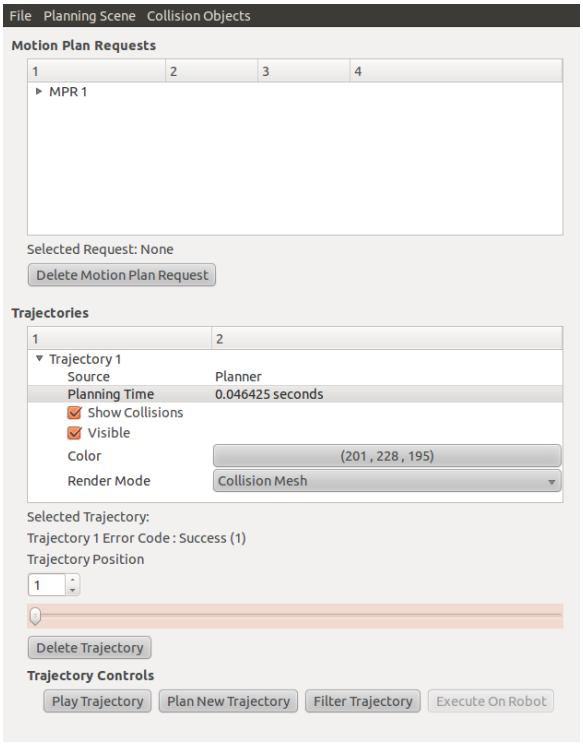


Figure 15: The trajectory tab of the warehouse viewer.

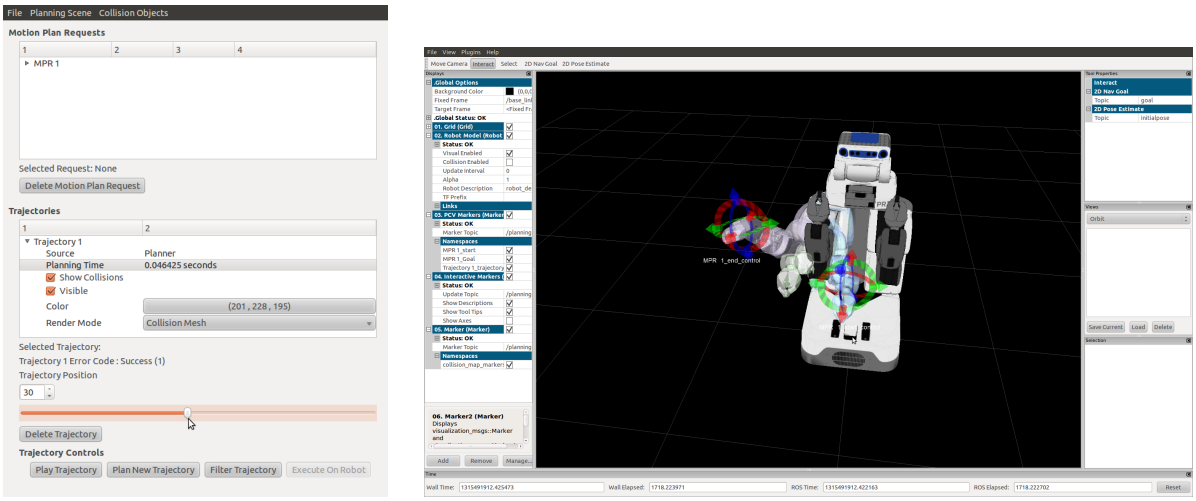


Figure 16: Step through a trajectory using the slider bar.

7.9 Invoking the Secondary Planner - Interpolated IK

The warehouse viewer is preconfigured to call OMPL as its primary planner, but it can also call the interpolated ik motion planner as a secondary planner. This planner operates very differently than OMPL, and is not capable of searching for collision-free paths. Instead it simply checks whether the goal location can be reached from the start location by interpolating the position of the end-effector in Cartesian space along a direct path and checking whether or not consistent IK solutions can be found. If it succeeds it will return trajectories of length 5 between the start and the

goal. Otherwise it will return an error and an invalid path. This planner is most useful for moving short segments where you want predictable motion - such as when moving from a pre-grasp pose to a grasp pose, picking up an object, or placing an object. When disabling collision checking between robot links and environment objects one should almost always use the interpolated planner as it will behave predictably. The paths produced by interpolated IK are directly executable on the robot and do not need to be filtered.

To use the warehouse viewer to generate interpolated IK paths, select the "Planner configuration" menu in the main viewer window. There are two check-boxes under the menu - one for the "Use primary planner" and one for "Use secondary planner". When the check-box for "Use secondary planner" is checked, any requests to plan a new trajectory will be generated by interpolated IK and not OMPL. See Figure 17 for an example of using interpolated IK to move into a grasp pose from a pre-grasp pose.

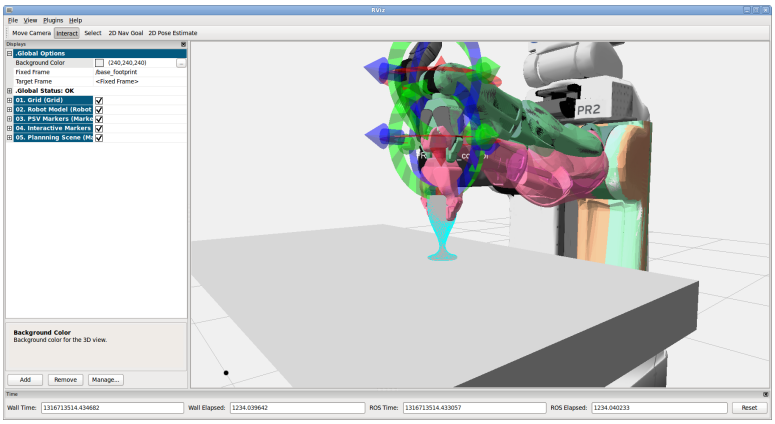


Figure 17: Interpolated IK is used to move from a pre-grasp pose to a grasp pose of an object with collision checking disabled between the left end effector and the object to be grasped.

Step Select the secondary planner, generate a motion plan request that moves an arm a short and collision-free distance, and request a plan. The plan will be so short that to see that steps will require manually stepping through the trajectory.

Caution Remember to check "Use primary planner" when moving long distances or for plans that require avoiding obstacles.

7.10 Collision Objects

We will now add a collision object to the scene using the **Collision Object->New Primitive Collision Object** menu (Figure 18).

We will add a flat box.

Select and move the box (using the interactive markers associated with the box) so it intersects the trajectory that you just planned, i.e. the planned trajectory will now be in collision. Deselect the box using the drop-down menu after a right-click on any of the interactive markers associated with the box.

7.10.1 Resizing the collision objects

You can resize the collision object by right-clicking on it and choosing the **Resize Mode->Grow** or the **Resize Mode->Shrink** options. Now the interactive markers will let you drag and grow or shrink the objects along their axes. Note, make sure to choose the **Resize Mode->Off** option once you are done.

7.10.2 Importing mesh objects

You can also import mesh collision objects. Choose **Collision Objects->New Mesh Collision Object...** from the warehouse viewer menu. First, select a new name for the collision object. Then, use the **Select Mesh file** button to

Figure 18: Adding a new collision object.

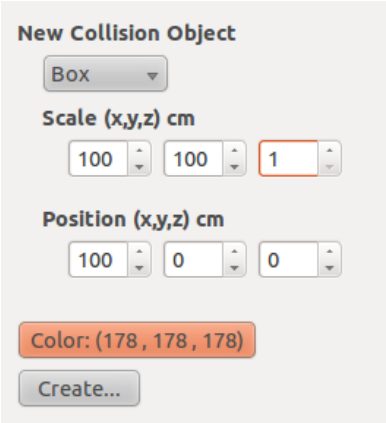


Figure 19: Adding a flat box as a collision object.

choose a mesh file from the *iros_tutorial_resources/objects* folder in the repository you downloaded earlier. Choose the bowl (by selecting "bowl.stl").

CAUTION *Make sure the Scale on x, y and z is set to 0.001 so your object is sized correctly*. You can move the object around in the environment and re-orient it using its interactive markers (just like you did for the primitive objects earlier).

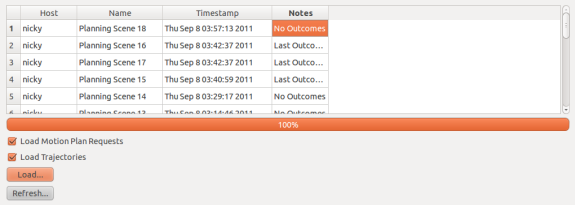
7.11 Refreshing planning scenes

During execution, other programs (such as the robot head monitor) may make changes to the planning scene stored on the arm navigation environment server. To reflect these changes in the planning scene warehouse viewer, you may refresh the planning scene by going to the menu: **Planning Scene > Refresh Planning Scene ...**.

7.12 Saving and loading planning scenes

The Warehouse Viewer allows you to save and load the planning scenes you have created. This is probably the most useful part of the viewer since it allows you to create complex scenes, store motion plan requests and trajectories for them and re-examine them later, e.g. with a new planner. Saving a planning scene is easy: **File->Save Planning Scene...**.

Loading a planning scene from the database is similarly very easy: **File->Load Planning Scene...** brings up a new window showing all the planning scenes that were stored. Planning scenes are stored with a host name (where the scene was saved from), a unique id (used only by the warehouse viewer), a timestamp and notes. You can choose to load the motion plan requests and trajectories associated with a scene or you can choose to just load the collision objects associated with the scene.



Exercise Create multiple motion plan requests for the left arm as well. Store all these requests in the warehouse by using the **File->Save Planning Scene** option.

Step After you have saved your planning scene, make sure to bring down the warehouse viewer using Ctrl-C in the window where you launched it.

8. Advanced Motion Planning — More Examples

In this section, you will explore the planning warehouse in much more detail. In particular, you will play with planning scenes we have created for the PR2. Some of these scenes involve particularly hard planning problems that the planners will take a long time to solve (or even fail within the alloted time).

Step Bring down the warehouse viewer program you were using in the previous section. You will need to make a small change in the launch file you were using to launch the viewer before we proceed.

8.1 Choose a different database

In this section, you will work with one of the pre-built databases. First, find an example file named pr2_example_db_1.tar.gz from the iros_tutorial_resources/arm_navigation_dbs directory you downloaded earlier from kforge. Move this file into the ".ros/arm_navigation_dbs" directory and extract it there using the following command:

```
tar zxf pr2_example_db_1.tgz
```

Note The **.ros** directory in your home directory is where all databases are stored.

This should extract a directory called pr2_example_db_1 with the relevant database files in it. Now, you need to put the *relative path* for this new directory (relative to the ".ros" directory) into the value field of the database_path parameter in your launch file (the file planning_scene_warehouse_viewer_pr2_test.launch in the pr2_test_arm_navigation package), i.e. find the line

```
<param name="database_path" value="arm_navigation_dbs/pr2_test"/>
```

and put your database location (the relative path) in the "value" field", i.e.

```
<param name="database_path" value="arm_navigation_dbs/pr2_example_db_1"/>
```

Now, relaunch the warehouse viewer and follow the steps from the previous section to get to the warehouse viewer and rviz windows:

```
roslaunch pr2_test_arm_navigation planning_scene_warehouse_viewer_pr2_test.launch
```

8.2 Load a planning scene

An example planning scene is shown in Figure 20. Load this planning scene using the following commands: **File->Load Planning Scene...** and then select the scene with name **Planning Scene 0** and choose **Load...**. Figure 20 shows a (rviz) visualization of the scene.

Have a look at the your warehouse viewer window. It should look like something like Figure 21. You will notice that there are multiple Motion Plan Requests now loaded in the viewer, labeled **MPR 0** onwards. Expand each motion plan request in turn (except **MPR 0**) and make sure the **Start Position** and **End Position** "visible" checkboxes are unchecked. This will make all the planning requests except **MPR 0** invisible.

You should now be able to see **MPR 0** in your rviz window. The start and goal for this motion plan request are shown in Figure 22. The right arm of the robot starts below the table and then plans to a new pose.

Right click the interactive controls for the start or goal position and choose **Plan new trajectory...** to plan and display a new trajectory for the PR2. Also, filter the trajectory and observe the difference, if any, between the planned and filtered trajectory.

8.3 Harder example

Now, for a harder example, choose **MPR 2**. Figure 23 shows the start and goal positions for this motion plan request. The robot is now trying to start from the back of a shelf. Choose **MPR 2**. Make all the other motion planning requests invisible so you can focus on this one. Now, use the "Plan new trajectory" button (or the option from the right-click menu for the interactive marker corresponding to the start or goal positions for **MPR 2**).

It's possible that the planner may (sometimes) fail. You can try to plan again in those cases. Use the stored trajectories in the warehouse viewer to step through the path (frame by frame) and make sure none of the waypoints are in collision.

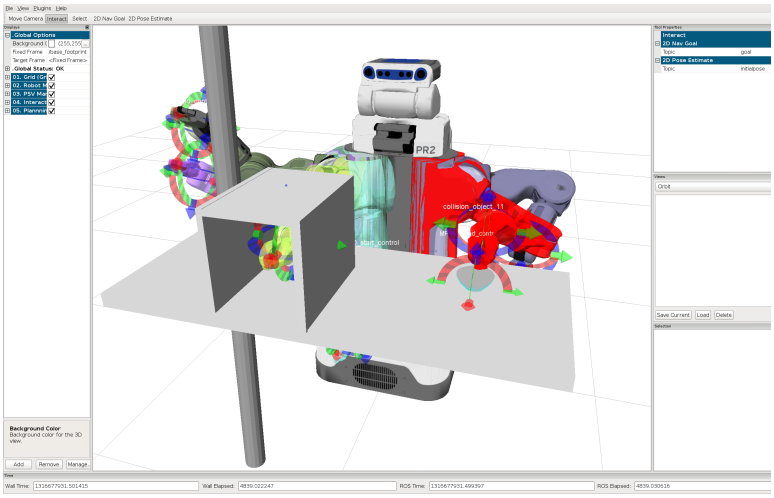


Figure 20: A visualization of Planning Scene 0.

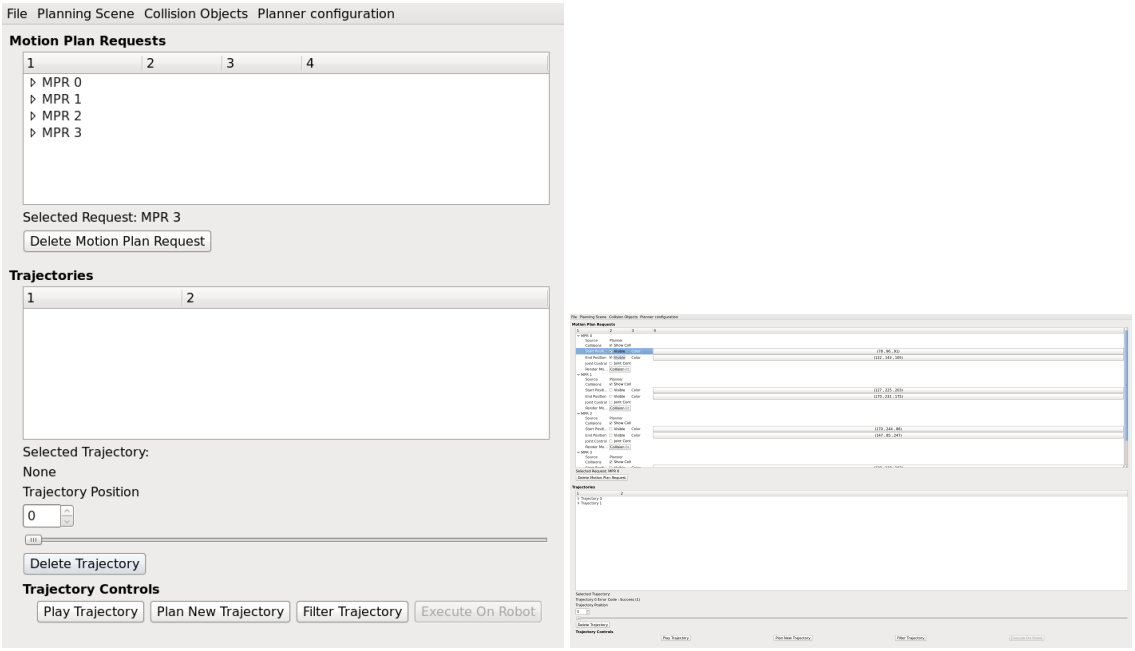


Figure 21: Left: The loaded example contains multiple motion plan requests. Right: Unchecking the "visible" checkboxes will make the corresponding markers for the motion plan requests disappear.

8.4 Collision operations

In the previous section (and the tutorial itself), you learned about "collision operations". Collision operations specify whether the planner (or any component dealing with collisions) needs to care about collisions between two specific collision bodies. A collision operation is always specified between two collision bodies, one of which is always a robot part or an object attached to the robot while the other can be a robot part, an attached object or a part of the environment.

Collision operations serve two main purposes:

1. A set of default collision operations specify which pairs of body parts should be checked for collisions against each other and which pairs should not. This checking helps substantially in optimizing collision checking since a

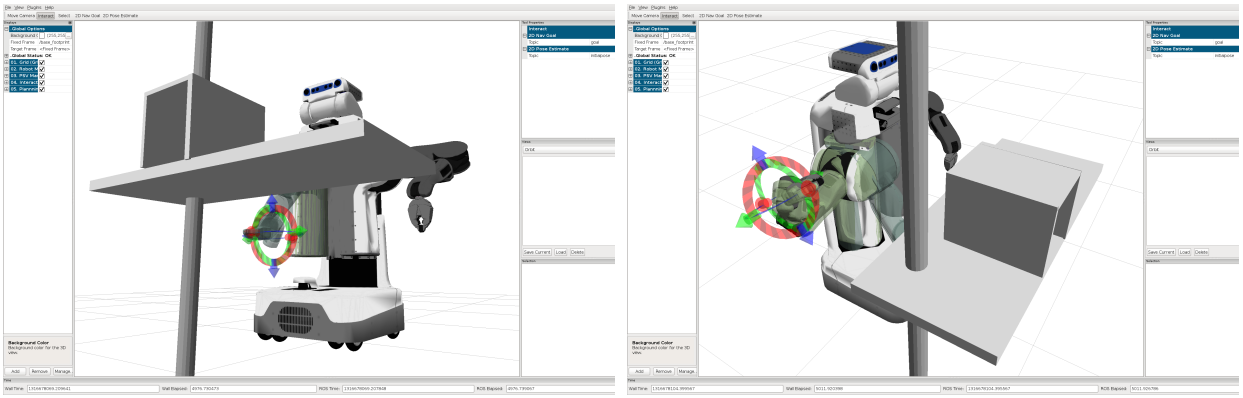


Figure 22: Left: Start position, Right: Goal position

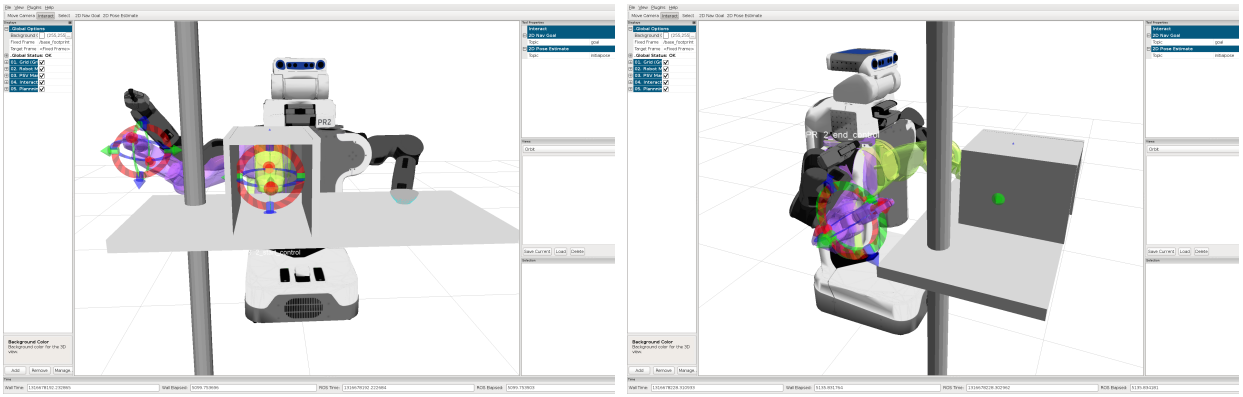


Figure 23: Left: Start position, Right: Goal position

lot of body parts are never in collision with each other.

2. Collision operations can also be applied selectively to a motion plan request, if, e.g. you want to specify that its ok for the a part of the body to be in contact with another part.

We will use **MPR 3** to illustrate the use of these operations. Note that this is just an example of how you could modify the operations. You should be very careful about disabling collisions in general. First, make sure just **MPR 3** is visible by making all other requests invisible. You should see something similar to Figure 24-Left in your rviz window. The left arm is in collision with the bowl and should appear red (try refreshing the planning scene if does not and also note the collision markers between the left gripper and the bowl).

Now, from the warehouse viewer menu, choose **Planning Scene -> Alter Allowed Collision Operations ..**. You will see two lists of links. Clicking on two links will display the status of the collision operation corresponding to that pair. The "Enable" and "Disable" buttons on the right allow you to change that status.

Choose "collision_object_11" in the first set of links. This is the bowl placed on the table. Then, choose all links in the second list whose names begin with "l_gripper", one after the other. Disable all the corresponding collision pairs, i.e. you are now telling the collision checker to disregard collisions between the *bowl* and any of the links that make up the left gripper. Now, click on the left gripper again. The color of the arm should change back to a regular (non-red) color indicating that the system no longer considers the goal location to be in collision. To change the collision operations back to the default values, use the "Reset" button. Make sure to refresh the planning scene **Planning Scene -> Refresh Planning Scene...**.

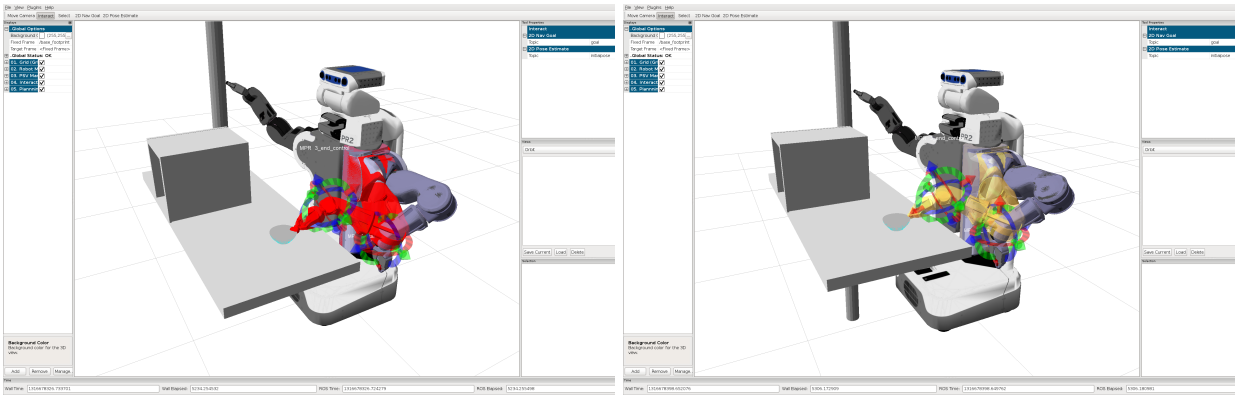


Figure 24: Left: MPR 5 (Left arm is in collision with bowl), Right: Goal is no longer considered in collision after collision operations have been applied.

8.5 More examples

Now, try yet another example by loading "Planning Scene 1". This scene has the robot in front of a narrow shelf like object as shown in Figure 25. There are several motion planning requests corresponding to this scene as well. The start and goal positions for **MPR 0** are also shown in Figure 25.

Figure 25: Top: Another planning scenario, Left: Start Pose, Right: Goal Pose.

Note how the goal pose is inside a narrow shelf. Try planning with this motion plan request. Again, you may find

that the planner takes a little time to solve the request. Some pre-computed plans are already included in the database. You can play these trajectories to see successful plans for this motion plan request.

8.6 Examining link paddings

The collision checkers we use currently for planning only give back information about whether parts of the body are in contact or not. Sampling based planners using such information tend to get close to obstacles. To account for this, we often add "padding" to each link's collision representation, i.e. we grow or enlarge the collision meshes by a certain amount. You can examine this amount for each link by clicking on the **Planning Scene -> Alter Link Padding** menu option in the warehouse viewer. Now, for **MPR 0**, set the view in rviz so you can see the goal pose of the left gripper well.

Find the entry corresponding to the *r_gripper_palm_link* in the list of link paddings. Use the *arrow buttons* next to the value for the padding to push padding up (all values are in meters). Note how once you push the padding to around 0.07, two additional red collision markers appear on the scene indicating that the gripper collision model with the addiontal padding is now in collision with the shelf (Figure 26). Choose the **Padding Mesh** option in **MPR 0 -> Render Mode** to see how the mesh has really changed in size and is now in collision with the environment.

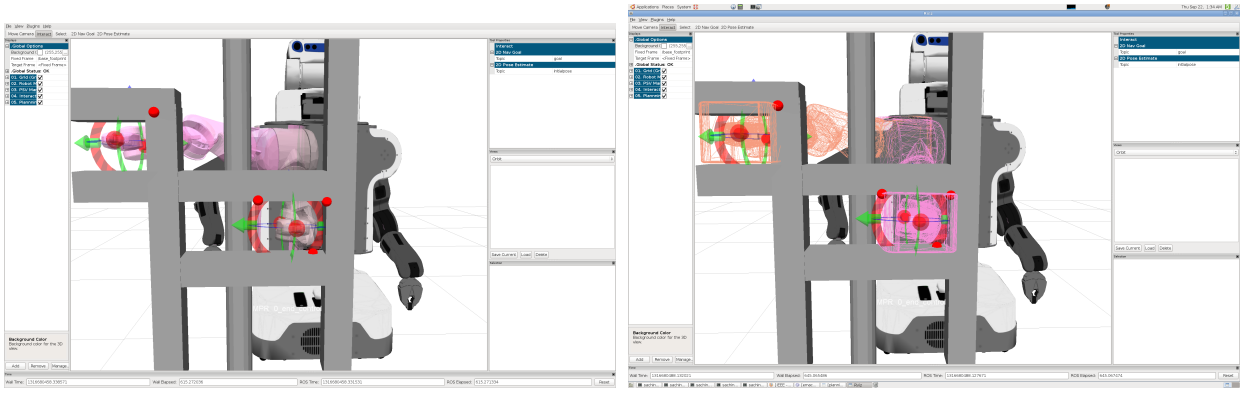


Figure 26: Left: Altered padding leads to collision with the shelf, Right: The modified collision mesh after padding was altered.

Exercise Using all the skills you have gained so far: (a) create a new world, (b) try to create a motion plan request that takes the planner a long time to solve, (c) plan and filter the resultant trajectory.

9. Advanced Motion Planning — In Simulation

In this section, you will explore operation on a simulated robot using the warehouse viewer. You will learn how to command motion on a simulated robot, add a (virtual) object in the scene and pick up the object.

Caution Before you attempt this part of the tutorial, make sure you have brought down all previous versions of the planning warehouse viewer and rviz.

9.1 Starting up in simulation

If your computer is somewhat powerful you will be able to run in simulation. Otherwise skip to Section 10 for running directly on the PR2.

Step Launch the simulator in a simple tabletop scene using the following command:

```
roslaunch pr2_warehouse_viewer pr2_tabletop_world.launch
```

A screenshot of the Gazebo simulator in the tabletop configuration is shown in Figure 27.

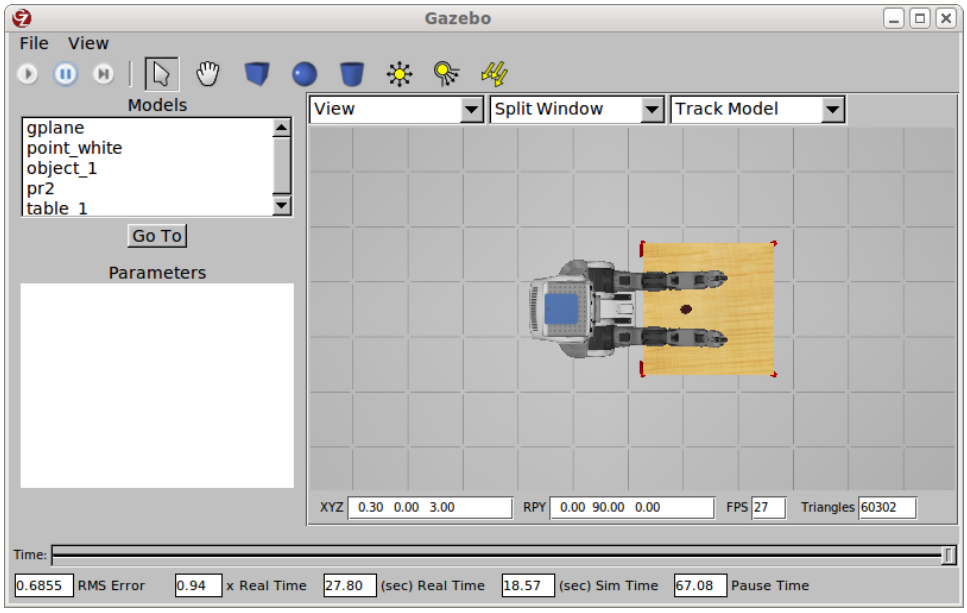


Figure 27: The Gazebo simulator running in a tabletop manipulation scenario.

9.2 Starting up the arm navigation software for the simulated robot

Step You will have to start up the motion planning pipeline using the default launch files already provided in the move_arm_warehouse package. To do this, use the following command in a separate terminal:

```
export ROBOT=sim;
roslaunch move_arm_warehouse planning_scene_warehouse_viewer_pr2_real.launch
```

9.3 Using a teleop interface

You will also be using a simple teleoperation keyboard interface for working with the simulated robot. To start this interface, use the following command on your computer in a separate terminal:

```
roslaunch pr2_teleop_general pr2_teleop_general_keyboard_noik.launch
```

You should see a screen come up with a keyboard interface message of the following form:

```
Reading from keyboard
---------------------------
Use 'h' for head commands
Use 'b' for body commands
Use 'l' for left arm commands
Use 'r' for right arm commands
Use 'a' for both arm commands
Use 'q' to quit
```

You can use the interface to control almost all the parts of the robot, even in simulation. For example, press "h" to access the commands for the head and then use "z" to turn the textured projector on. You should see a red projected light glowing on the head of the robot. Now, use the "i/j/k/l" keys to move the head up and down and side to side. Quit the head interface (using the "q" key).

Try out the keyboard interface to the arms. Choose the right arm first and then move the arm from side to side and up and down. Open and close the gripper using the "o" and "p" keys. Now repeat the same for the left arm. You are now ready to use the warehouse viewer with the robot.

9.4 Starting up the warehouse viewer

In simulation you'll start up the warehouse viewer by running the following command:

```
roslaunch move_arm_warehouse planning_scene_warehouse_viewer_pr2_only.launch
```

When the viewer starts, first **Accept** the default configuration and then choose a **New...** planning scene.

9.5 Starting up the visualizer

You will have to separately startup the ROS visualizer (rviz) as well. In a separate terminal, execute the following command:

```
rosrun rviz rviz
```

Once you have started the visualizer, set the "Fixed Frame" in rviz to *odom_combined*.

9.6 Start working on the simulated robot

You are now ready to start working on the simulated robot. The typical view you should see in the visualizer will look like the one in Figure 28 where the robot has been placed in front of a table with multiple objects on it. Figure 29 shows the view from the robot's camera. You can also see such a view by adding a "Camera" in the visualizer and pointing its topic to the relevant camera image topic from the robot.

9.7 Add a virtual object

First, using the keyboard interface introduced earlier, open the right gripper. Then, add a virtual object into the planning scene (using a collision object). Move the object using its interactive marker controls and put it on the table as in Figure 30. Now, create a new motion plan request. When working with the robot, the start state of a new request is automatically set to the current state of the robot. Your request should look similar to the motion plan request shown in Figure 30. Try to get your pre-grasp pose as close to the robot as possible without getting into contact with the object.

Now, create a plan to this location using the **Plan New Trajectory** button. Make sure to filter the trajectory using the right-click menu on the interactive markers corresponding to the goal. If (and only if) you are satisifed with the resultant trajectory and there are no errors in the trajectory outcomes, execute the trajectory on the robot.

EXECUTE WITH CAUTION Use the right-click menu from the interactive marker for the goal and choose **Execute Last Trajectory** to send the trajectory to the robot. You can alternatively use the **Execute On Robot** button in the warehouse viewer. *Only execute trajectories that have been filtered successfully on the robot. Never execute planned trajectories on the robot without filtering them first.*

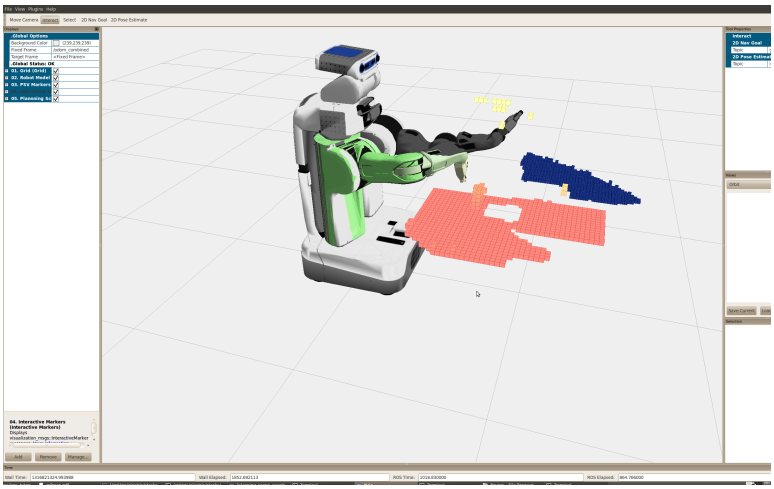


Figure 28: A visualization of the view in front of the robot.

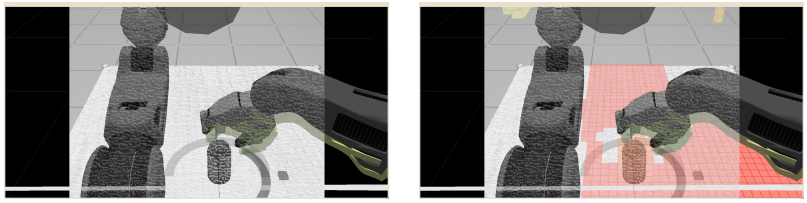


Figure 29: The robot's view.

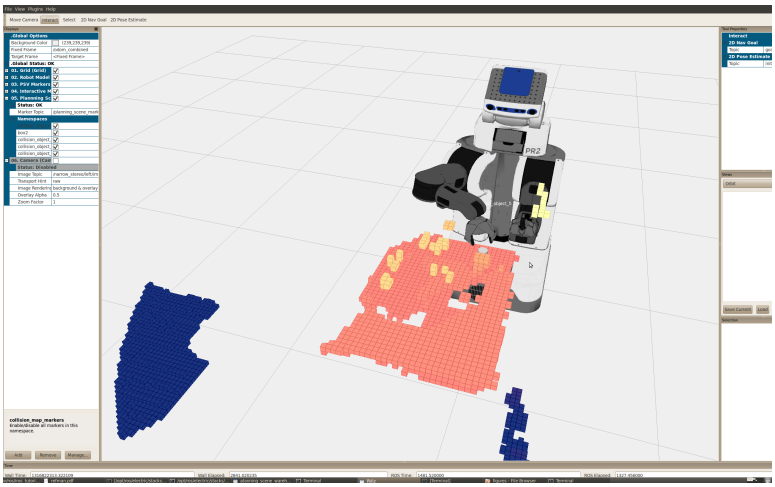


Figure 30: A pre-grasp pose.

Exercise Repeat the above step using the collision operations concepts you learnt earlier to try and get closer to the object.

9.8 Attaching an object

Once the robot's arm gets to the pre-grasp location (Figure 31), you can start thinking about how to get the object back to the side with the arm. Now, we will make a simplifying assumption here - we will assume at this point that the object is already attached to the robot gripper, i.e. we assume that the robot hand has grasped the object successfully. To convey this to the system, you need to *attach* the object to the robot gripper. This tells the system that the object will now move with the hand and the motion planner will account for this as the robot moves.

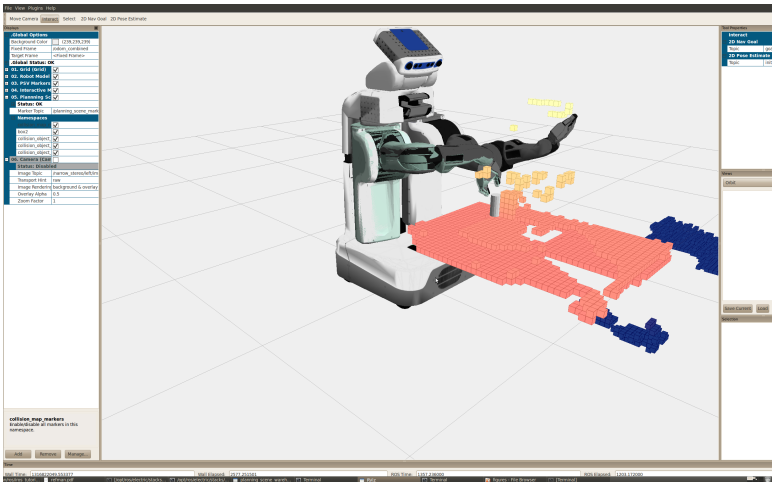


Figure 31: A pre-grasp pose.

To attach the object, right-click on its interactive marker controls and select **Attach**. In the warehouse viewer, a new dialog will come up asking for the link that the object should be attached to and the names of *touch links*. Grasping and manipulation inherently involve contact. The touch links allow you to specify which links of the robot are allowed to be in contact with the object. You can either choose a set of links, or a group. In this case, you should choose the *r_end_effector* group that you had created earlier.

Now, add a new motion plan request. You must *always* add a new motion plan request for any new motion of the robot. Use the motion plan request to move the object away and to the side of the robot. You will notice that the end goal for the motion plan request displays the object attached to the gripper, i.e. the planner now assumes that the object stays fixed relative to the gripper and moves with it (Figure 32).

Follow the same procedure you did earlier, i.e. planning followed by filtering followed by execution on the robot. Before the filtering and execution steps, check the error code on the trajectory in the warehouse viewer. If its not **SUCCESS**, do not execute the trajectory. The robot visualization (after execution) is shown in Figure 33.

CAUTION When working with the simulated robot, *always* add a new motion planning request for a new motion. Never reuse a motion planning request for a new motion.

CAUTION If anything in the environment changes, make sure to use the **Planning Scene...->Refresh Planning Scene** command from the top-menu of the warehouse viewer. This brings the latest environment representation into the planning scene. Remember that if you do not refresh the planning, it *will not* get refreshed, i.e. you will always be working with stale/old data. This could lead to collisions with the environment or the object.

Exercise Using a similar approach, try and grab an actual object off the table and put it back down. You may have to make extensive use of collision operations to make this succeed.

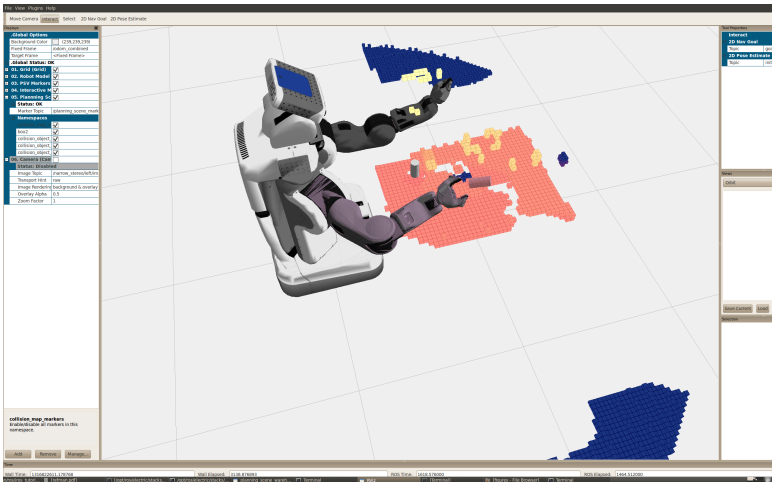


Figure 32: A new goal for the right arm while carrying the object. Note how the object is assumed to stay fixed relative to the gripper.

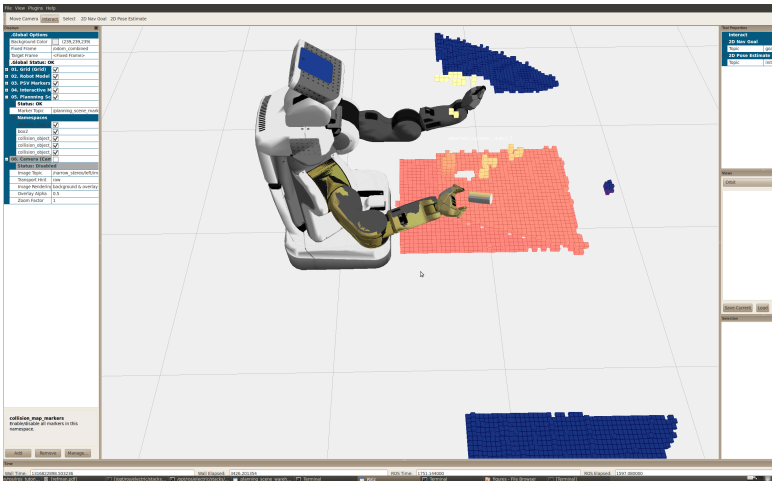


Figure 33: A visualization of the actual robot pose after execution.

10. Advanced Motion Planning — On the Robot

In this section, you will explore operation on the robot using the warehouse viewer. You will learn how to command motion on the actual robot, add a (virtual) object in the scene and pick up the object. You can then attempt to pickup a real object using the PR2.

Caution Before you attempt this part of the tutorial, make sure you have brought down all previous versions of the planning warehouse viewer and rviz.

10.1 Starting up on the robot

Caution This part of the tutorial is only for the PR2 robot or running in simulation. It should be performed only by someone who knows how to run code on the PR2 robot. For the purpose of this tutorial, you should skip this part and move directly to Section 10.2.

Step On the PR2 robot, you will have to start up the motion planning pipeline using the default launch files already provided in the move_arm_warehouse package. To do this, use the following command (on the robot):

```
roslaunch move_arm_warehouse planning_scene_warehouse_viewer_pr2_real.launch
```

10.2 Using a teleop interface

You will also be using a simple teleoperation keyboard interface for working with the robot. To start this interface, use the following command on your computer (ask your instructors what the correct value for *robot_hostname* should be):

```
export ROS_MASTER_URI=http://robot_hostname:11311
roslaunch pr2_teleop_general pr2_teleop_general_keyboard_noik.launch
```

You should see a screen come up with a keyboard interface message of the following form:

```
Reading from keyboard
---------------------------
Use 'h' for head commands
Use 'b' for body commands
Use 'l' for left arm commands
Use 'r' for right arm commands
Use 'a' for both arm commands
Use 'q' to quit
```

You can use the interface to control almost all the parts of the robot, even in simulation. For example, press "h" to access the commands for the head and then use "z" to turn the textured projector on. You should see a red projected light glowing on the head of the robot. Now, use the "i/j/k/l" keys to move the head up and down and side to side. Quit the head interface (using the "q" key).

Try out the keyboard interface to the arms. Choose the right arm first and then move the arm from side to side and up and down. Open and close the gripper using the "o" and "p" keys. Now repeat the same for the left arm. You are now ready to use the warehouse viewer with the robot.

10.3 Starting up the warehouse viewer

10.3.1 On the Physical Robot

On your desktop (or laptop), you will be starting a warehouse viewer that connects directly to the robot. For this to work, you will have to specify the address of the robot to the viewer. You can do this by editing the file "pr2_warehouse_viewer.launch" in the "pr2_warehouse_viewer" package. This package is in the iros_tutorial_resources directory that you downloaded earlier and you should be able to use "roscd" to get to it.

In a separate terminal, first set your local environment variables to connect to ROS master on the robot using the following command (you do not need to do this operating in simulation):

```
export ROS_MASTER_URI=http://robot_hostname:11311
```

Now, cd to the pr2_warehouse_viewer package.

```
roscd pr2_warehouse_viewer
```

Edit the pr2_warehouse_viewer.launch file in this directory and replace the warehouse_host param (which will be set to "localhost" or "prj1") with the hostname of your robot. Now, run this file in a terminal on your computer to bring up the warehouse viewer. When the viewer starts, first **Accept** the default configuration and then choose a **New...** planning scene.

```
roslaunch pr2_warehouse_viewer pr2_warehouse_viewer.launch
```

10.4 Starting up the visualizer

You will have to separately startup the ROS visualizer (rviz) as well.

```
export ROS_MASTER_URI=http://robot_hostname:11311
rosrun rviz rviz
```

Once you have started the visualizer, set the “Fixed Frame” in rviz to *odom_combined*.

10.5 Start working on the robot

You are now ready to start working on the robot. The typical view you should see in the visualizer will look like the one in Figure 34 where the robot has been placed in front of a table with multiple objects on it. Figure 35 shows the view from the robot’s camera. You can also see such a view by adding a “Camera" in the visualizer and pointing its topic to the relevant camera image topic from the robot.

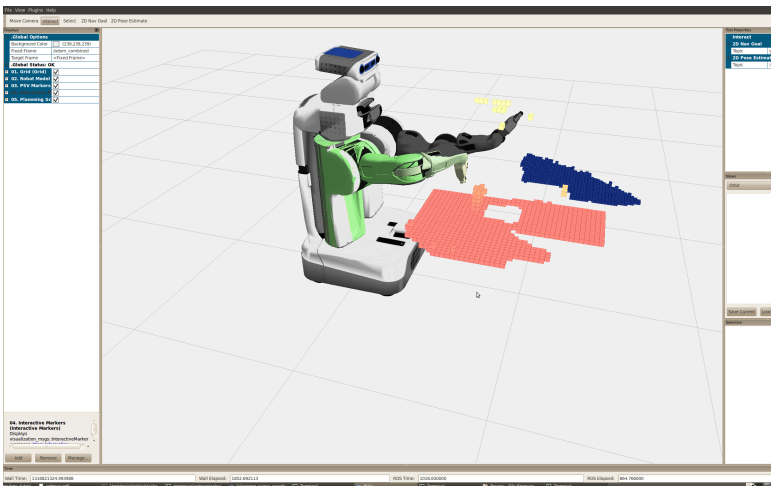


Figure 34: A visualization of the view in front of the robot.

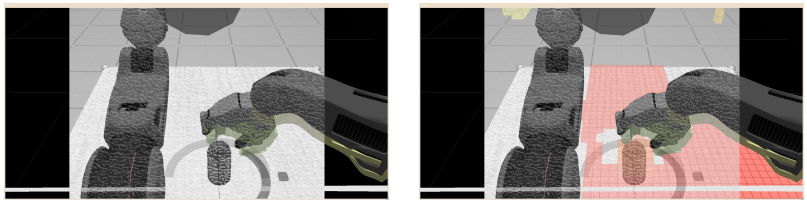


Figure 35: The robot’s view.

10.6 Add a virtual object

First, using the keyboard interface introduced earlier, open the right gripper. Then, add a virtual object into the planning scene (using a collision object). Move the object using its interactive marker controls and put it on the table as in Figure 36. Now, create a new motion plan request. When working with the robot, the start state of a new request is automatically set to the current state of the robot. Your request should look similar to the motion plan request shown in Figure 36. Try to get your pre-grasp pose as close to the robot as possible without getting into contact with the object.

Now, create a plan to this location using the **Plan New Trajectory** button. Make sure to filter the trajectory using the right-click menu on the interactive markers corresponding to the goal. If (and only if) you are satisifed with the resultant trajectory and there are no errors in the trajectory outcomes, execute the trajectory on the robot.

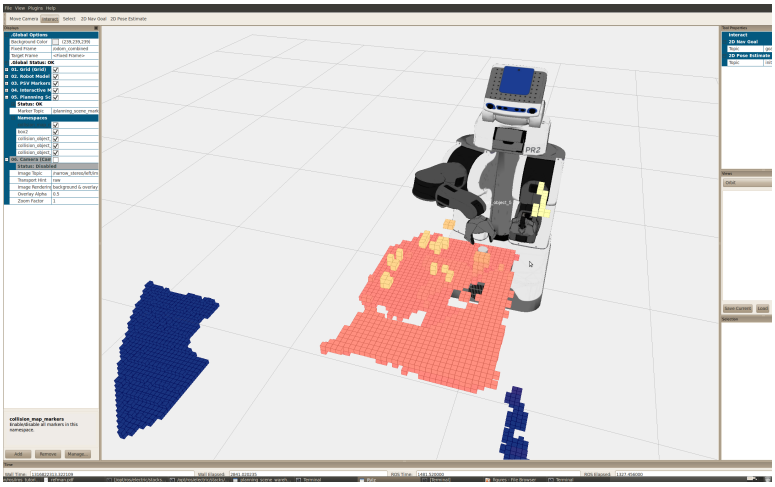


Figure 36: A pre-grasp pose.

EXECUTE WITH CAUTION Use the right-click menu from the interactive marker for the goal and choose **Execute Last Trajectory** to send the trajectory to the robot. You can alternatively use the **Execute On Robot** button in the warehouse viewer. *Only execute trajectories that have been filtered successfully on the robot. Never execute planned trajectories on the robot without filtering on the robot.*

Exercise Repeat the above step using the collision operations concepts you learnt earlier to try and get closer to the object.

10.7 Attaching an object

Once the robot's arm gets to the pre-grasp location (Figure 37), you can start thinking about how to get the object back to the side with the arm. Now, we will make a simplifying assumption here - we will assume at this point that the object is already attached to the robot gripper, i.e. we assume that the robot hand has grasped the object successfully. To convey this to the system, you need to *attach* the object to the robot gripper. This tells the system that the object will now move with the hand and the motion planner will account for this as the robot moves.

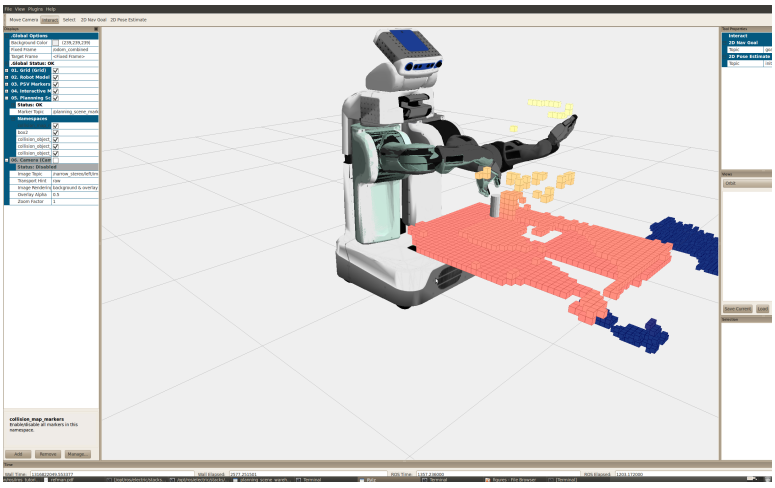


Figure 37: A pre-grasp pose.

To attach the object, right-click on its interactive marker controls and select **Attach**. In the warehouse viewer, a new

dialog will come up asking for the link that the object should be attached to and the names of *touch links*. Grasping and manipulation inherently involve contact. The touch links allow you to specify which links of the robot are allowed to be in contact with the object. You can either choose a set of links, or a group. In this case, you should choose all the links in the *r_gripper*, i.e. all links that start with *r_gripper_*.

Now, add a new motion plan request. You must *always* add a new motion plan request for any new motion of the robot. Use the motion plan request to move the object away and to the side of the robot. You will notice that the end goal for the motion plan request displays the object attached to the gripper, i.e. the planner now assumes that the object stays fixed relative to the gripper and moves with it (Figure 38).

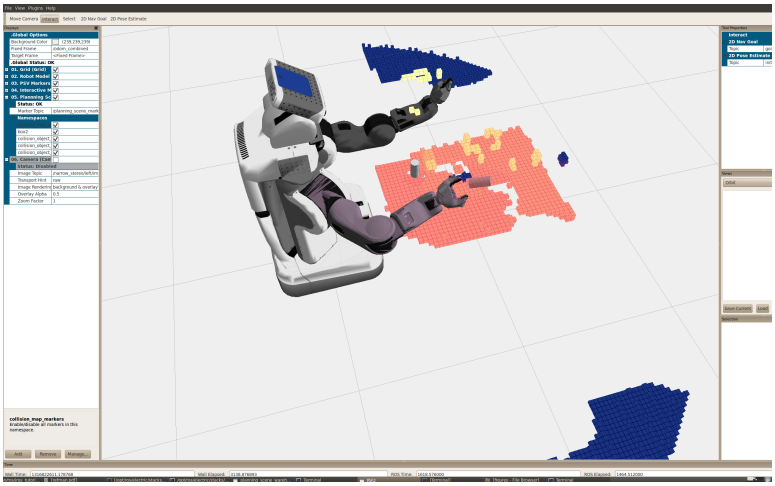


Figure 38: A new goal for the right arm while carrying the object. Note how the object is assumed to stay fixed relative to the gripper.

Follow the same procedure you did earlier, i.e. planning followed by filtering followed by execution on the robot. Before the filtering and execution steps, check the error code on the trajectory in the warehouse viewer. If its not **SUCCESS**, do not execute the trajectory. The robot visualization (after execution) is shown in Figure 39.

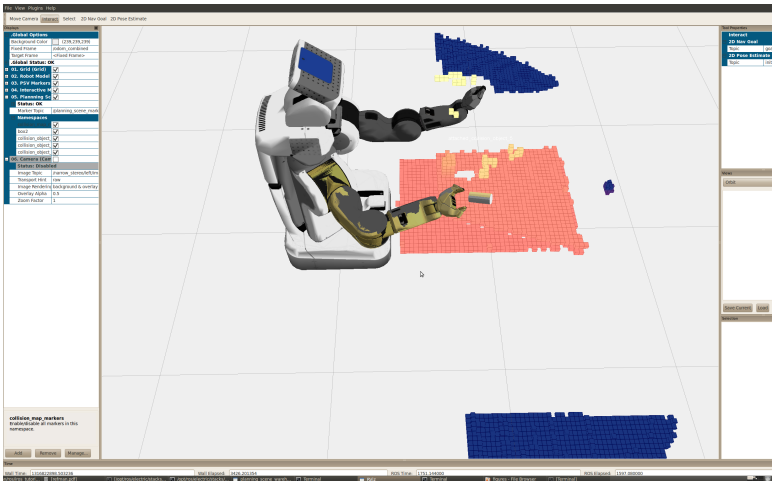


Figure 39: A visualization of the actual robot pose after execution.

CAUTION When working with the robot, *always* add a new motion planning request for a new motion. Never reuse a motion planning request for a new motion.

CAUTION If anything in the environment changes, make sure to use the **Planning Scene...->Refresh Planning Scene** command from the top-menu of the warehouse viewer. This brings the latest environment representation into the planning scene. Remember that if you do not refresh the planning, it *will not* get refreshed, i.e. you will always be working with stale/old data. This could lead to collisions with the environment or the object.

Exercise Using a similar approach, try and grab an actual object off the table and put it back down. You may have to make extensive use of collision operations to make this succeed.

Suggested Exercise Try to recreate the environment you created in your earlier exercise using the props you are provided with. Create difficult motion planning requests and try to solve them.

11. Programmatically Controlling the Robot Arm

This section will show you how to interact with the robot arm programmatically, i.e., how to write code that asks the robot to plan arm motions to different locations. The code is included in the iros_tutorial_resources repository, which you checked out earlier, and should be in your ROS_PACKAGE_PATH. The code works for either arm, but it configured for the right arm.

11.1 Setting up for executing the code

Step You will need to compile the code:

```
roscd simple_armstack_goals
rosmake
```

Step If you are running on the robot, skip this step.
If you are running in simulation, you must do:

Terminal 1:

```
export ROBOT=sim
roscd pr2_gazebo
roslaunch pr2_empty_world.launch
```

Terminal 2:

```
export ROBOT=sim
roscd pr2_3dnav
roslaunch right_arm_navigation.launch
```

Caution You will need to do the `export ROBOT=sim` for every terminal you use while in simulation.

Step If you are running in simulation, skip this step.
If you are running on the robot, you must point your ROS_MASTER_URI to the correct robot:

```
export ROS_MASTER_URI=http://robot_hostname:11311
```

Caution Make sure you talk to one of the instructors before you start working on this section. They will need to issue the following command on the robot: `roslaunch pr2_3dnav right_arm_navigation.launch`

11.2 The programs to execute

In the simple_armstack_goals package that you compiled earlier, there are three source files, under the src/ folder:

- `move_arm_joint_goal.cpp`

  This program allows you to specify the 7 joint values that you want your robot arm to achieve.

- `move_arm_pose_goal.cpp`

  This program allows you to specify the pose (position and orientation) that the end effector should be moved to.

- `move_arm_pose_goal_with_constraints.cpp`

  This program allows you to specify the pose (position and orientation) that the end effector should be moved to, but it will also impose constraints on the solution path: maintaining the orientation of the end effector.

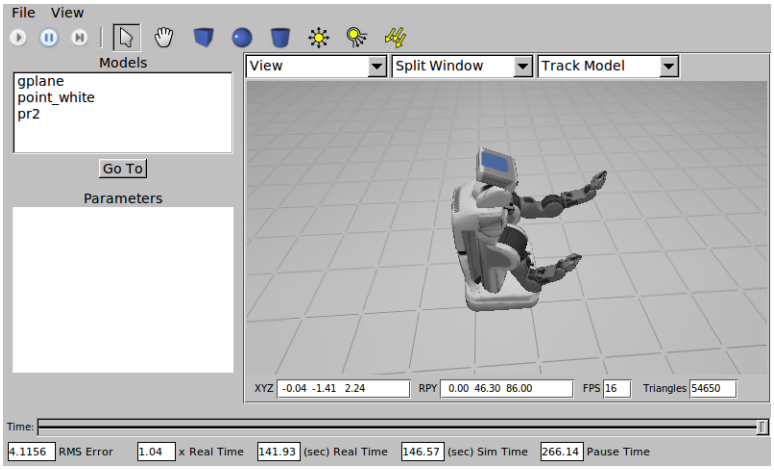


Figure 40: Result of move_arm_joint_goal.

Step To execute the programs, you will have to do:

```
// don't forget the export ROBOT=sim if you are in simulation
roscd simple_armstack_goals

bin/move_arm_joint_goal
```

This command should make the robot achieve a pose as shown in Figure 40.

```
// don't forget the export ROBOT=sim if you are in simulation
roscd simple_armstack_goals

bin/move_arm_pose_goal
```

This command should make the robot achieve a pose as shown in Figure 41.

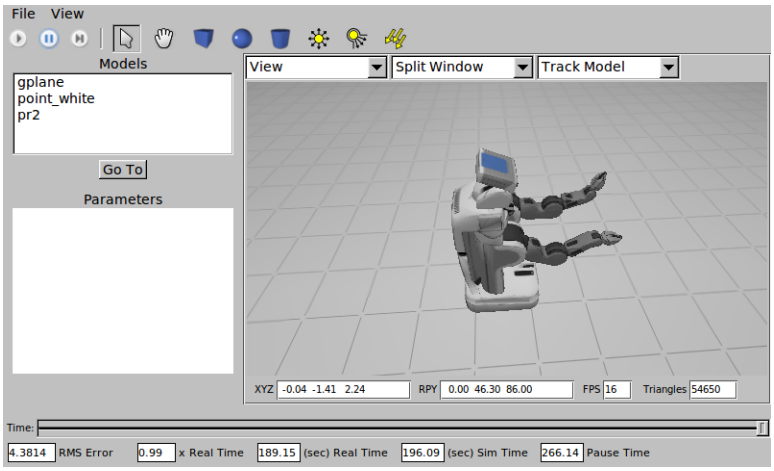


Figure 41: Result of move_arm_pose_goal.

```
// don't forget the export ROBOT=sim if you are in simulation
```

```
roscd simple_armstack_goals

bin/move_arm_pose_goal_with_constraints
```

This command should make the robot achieve a pose as shown in Figure 42.

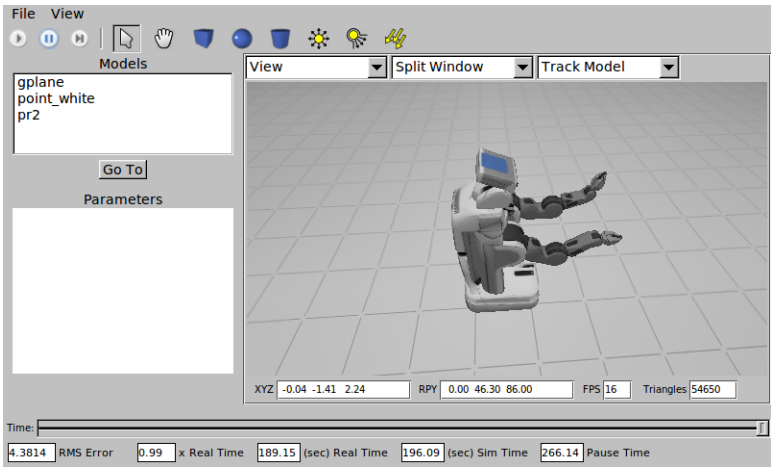


Figure 42: Result of move_arm_pose_goal_with_constraints.

Step Feel free to modify the source files supplied, and change the goal locations, or the arm you operate on.