Technical Report

IRI-TR-13-03

The MoveIt motion planning framework Configuration for the IRI_WAM robot

Adrià González
Guillem Alenyà

September, 2013

Institut de Robòtica i Informàtica Industrial

Abstract

Perform the setup and prepare WAM robots to use the the motion planning and obstacle avoidance facilities present at the MoveIt! framework. In particular, determine the configurations and API function calls to set the target of the robot arm with joint state and add collision objects programatically, without using the graphical interface. Supporting page with videos: `http://www.iri.upc.edu/groups/perception/MoveItForIRIWam`

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain

Tel (fax): +34 93 401 5750 (5751)
`http://www.iri.upc.edu`

Corresponding author:

A González
tel: +34 93 401 0775
`agonzalez@iri.upc.edu`
`http://www.iri.upc.edu/staff/agonzalez`

Copyright IRI, 2013

1 External dependencies

In order to execute the following nodes you need to install ROS groovy[1], openni camera for groovy[2] and MoveIt![3]

2 Internal IRI dependencies

If not already available, create a catkin workspace[4] in order to execute the collision_object and test_move_group nodes[5] and the description of the iri_wam robot from moveit![6]

3 Run WAM server

- Turn ON the WAM robot arm and connect to the computer:
  - `ssh robot@(bawse/bawseclon)`
- Check that the configuration of the end effector is correct and run the wam_server in the terminal:
  - `vim /etc/barrett/default.conf`
  - `cd ~/code/wam/trunk/bin/`
  - `./wam_server`

4 Run WAM client

Assuming that you have installed the iri_wam package[7], do:

- Run the iri_wam_bringup ROS launch in a different terminal with the IP of the server and the argument of the kinect as True to launch the Kinect and his tf for the estirabot robot:
  - `roslaunch iri_wam_bringup iri_wam_bringup.launch IP:=192.168.101.1 KINECT:=True`
- Press the intro key in the client's terminal, follow the server's terminal instructions and press the intro key again in the client's terminal.

5 Run MoveIt!

- Run the moveit_planning_execution ROS launch in a new terminal for launching all the MoveIt! nodes necessaries for the WAM robot:
  - `roslaunch iri_wam_moveit_config moveit_planning_execution.launch`

[1] http://wiki.ros.org/groovy/Installation/Ubuntu
[2] http://wiki.ros.org/openni_camera
[3] http://moveit.ros.org/wiki/Installation
[4] http://wiki.ros.org/catkin/Tutorials/create_a_workspace
[5] https://devel.iri.upc.edu/labrobotica/ros/iri-ros-pkg/stacks/iri_wam/branches/
[6] wstool set iri_wam_moveit_config –git https://github.com/ros-planning/moveit_robots
[7] http://wiki.iri.upc.edu/index.php/Download_IRI_ROS_software

6 Run Move Group node

- If you want to know the joint state, run the /joint_state node:

  - `rosrun test_move_group joint_state`

  This node subscribes to the topic who publishes the current joint state and prints it in the screen to use the following node easier.

- Run the /move_group_interface_demo node[8] followed by the goal joint values:

  - `rosrun test_move_group move_group_interface_demo`

  This node defines the joint values typed as the target state and runs the move_group action step by step. First plans the trajectory and shows it in the rviz and if we are agree with the trajectory proposed by MoveIt! we can finish the action and execute the trajectory.

  Here you have a little description of the best planners[9]:

  - KPIECE: planner used by default. This planner first discretizes the workspace in feasible blocks (similar to octomap), distributes nodes in this zone and expands a tree from the start node. It's a planner that usually works well.
  - EST: planner that expands a tree from the start node verifying the feasibility of the nodes in every expansion. So the difference with the KPIECE planner is that in KPIECE the verification of the feasibility it's made before the expansion of the whole tree. It's good for scenes where there are narrow paths to achieve the goal node.
  - SBL: planner similar to EST but expands two trees from the start and goal nodes. For this reason, the solutions found by the planner follow two different trajectories. The firstone escapes from the start and the other approachs to the goal.
  - RRT: planner totally random. For this reason, if there is one solution or it isn't easy to find a solution.

  If you want to change the planner you have to use the next action:

  - `move_group_interface::MoveGroup::setPlannerId("planner")`

  There are also some interesting functions in `move_group_interface::MoveGroup`:

  - `setPathConstraints` lets define some constraints to the trajectory. For example, impose that the end effector has constantly one orientation, important if we use the pick and place service or impose that one joint has always the same joint value to let find less solution by the planner.
  - `setPlanningTime` lets plan until a time different to the 5 seconds given by default.
  - `setJointValueTarget/setPositionTarget` to define the goal state without using the interface of rviz.
  - `allowReplanning` to let the MoveIt! replan if there is a dynamic scene.
  - `attachObject` lets define an object as part of the robot useful for pick and place service.

[8] http://moveit.ros.org/doxygen/classmoveit_1_1planning__interface_1_1MoveGroup.html
[9] http://ompl.kavrakilab.org/planners.html

7 Run Collision Object node

- Run the /collision_object node:
  - `rosrun collision_object collision_object`

  This node lets you create, erase, move, resize and rotate a collision object; following the instructions given by the terminal. If you change the environment using the interface of MoveIt! you have to press the "Publish Current Scene" button in the "Context" tab because the node doesn't know if you change the environment until you publish it.

8 Shutdown

- Finish the collision object node
- Press `Ctrl + c` in the client's terminal and follow the instructions in the server's terminal
- Press `Ctrl + c` in the other terminals

Using the MoveIt! setup assistant the necessary files to allow planning in your platform are created. They need some modifications and adjustments to enable their use in IRI_WAM platform.

A Set Kinect

- Add[10] these lines to the iri_wam_moveit_sensor_manager ROS launch from iri_wam_moveit_config because by default the robot doesn't have sensing with the kinect and we have to introduce it as an octomap:

  ```
  <param name="octomap_frame" type="string" value="iri_wam_link_footprint" />
  <param name="octomap_resolution" type="double" value="0.02" />
  <param name="max_range" type="double" value="5.0" />
  <rosparam command="load" file="$(find iri_wam_moveit_config)/config/sensors_rgbd.yaml"
  />
  ```

- Create sensor_rgbd.yaml file in the config directory:

  ```
  sensors:
  - sensor_plugin: occupancy_map_monitor/PointCloudOctomapUpdater
  point_cloud_topic: /camera/depth_registered/points
  max_range: 5.0
  frame_subsample: 1
  point_subsample: 1
  shape_padding: 0.02
  shape_scale: 1.0
  filtered_cloud_topic: filtered_cloud
  ```

  This converts the depth images from the kinect to an octomap for MoveIt! The defined parameters are for the self filter of the arm robot in the collision map. If you take a glance to the launch above, you can see that it loads this .yaml file.

B Set MoveIt! capability move_group

- Add the GetPlanningScene service capability in move_group ROS launch because by default MoveIt! doesn't let use the GetPlanningScene service and without this is impossible to know the changes of the environment if we use the interface of MoveIt!:

  ```
  move_group/MoveGroupGetPlanningSceneService
  ```

[10] http://moveit.ros.org/wiki/PR2/Gazebo/Quick_Start#Configure_MoveIt.21_For_Sensing_With_The_PR2

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.