Karl Kruusamäe, karl.kruusamae@ut.ee University of Tartu

ROS: Motion Planning with MoveIt!

The objective of this lab is to learn how to use ROS MoveIt! for motion planning. Familiarize yourself with key concept and go through the process of setting up MoveIt! configuration package for an industrial robot.

Setting up your computer

Open terminal window.

Clone your git repository to your home folder:

```
$ git clone <URL for your personal repository <yourname>-rtech>
```

Use `ls` to confirm that ***<yourname>-rtech*** has been downloaded to your home folder**.**

MoveIt! tutorials with Franka Emika Panda

In this section you are going to:

1) Learn the basic concepts of MoveIt! motion planning on a Franka Emika Panda robot manipulator using RViz and its GUI elements.
2) Write example C++ code for motion planning for Panda manipulator.

First, install MoveIt!, its tutorial package, and packages for Panda robot manipulator.

```
$ sudo apt install ros-kinetic-moveit

$ git submodule add -b kinetic-devel
https://github.com/ros-planning/moveit_tutorials.git

$ git submodule add -b melodic-devel
https://github.com/ros-planning/panda_moveit_config.git

$ rosdep install -y --from-paths . --ignore-src --rosdistro kinetic
```

Build your workspace!

Now you are all set up to learn the main concepts of MoveIt! by completing the following tutorial, start from **Step 1**.

- MoveIt! Quickstart in RViz

Next, create a new ROS package `planning_tutorial` to your catkin workspace. In order to truly test the code examples of the following tutorial, create a ROS node in the `planning_tutorial` package. Make the entire code example from the following tutorial part of your planning tutorial package and play with it to understand individual code blocks better.

Please note that the source code provided in the following online tutorial contains a lot of different examples for making the robot move. The objective, however, is to understand each and every one of these options for making the manipulator robot move. For instance, deleting large portions of the redundant code is a very good way of pinpointing specific examples and understanding the complexity/simplicity of the programming.

- Move Group C++ Interface

SETTING UP MOVEIT! FOR YASKAWA SIA5D INDUSTRIAL MANIPULATOR

In this section you are going to:

1) Download support packages for Yaskawa Motoman industrial robots.
2) Use MoveIt! Setup Assistant to generate MoveIt! configuration package for Yaskawa SIA5D industrial robot (Figure 1).
3) Plan and execute pre-programmed motion of Yaskawa SIA5D.

Figure 1. *Motoman Yaskawa SIA5D.*

First, let us download the motoman repository from ROS-Industrial GitHub. Add the motoman repository to your own repository as a submodule.

```
$ git submodule add –b kinetic-devel <URL to motoman's git repository>
```

Compile your catkin workspace and if you get dependency-related error messages, try installing the following ROS packages.

```
$ sudo apt install ros-kinetic-industrial-msgs
ros-kinetic-industrial-robot-client
```

After you have successfully compiled your catkin workspace, test it by launching

```
$ roslaunch motoman_sia20d_moveit_config demo.launch
```

Next, you are going to set up a MoveIt! configuration package for the SIA5D manipulator. A robot-specific MoveIt! configuration package is always required for using motion planning with MoveIt!.

MoveIt! has a super convenient tool – Setup Assistant – that you can use for setting up the configuration packages. Launch the Setup Assistant:

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

In the opening window choose the option for creating a new package and load robot description from `motoman_sia5d_support/urdf/sia5d.xacro`

Under *Self-Collisions*, set sampling density to high and click "Generate Collision Matrix". Double-check the collision list assumptions.

Under *Virtual Joints*, add a new virtual joint.

- Name: FixedBase
- Child: base_link
- Parent: world
- Type: fixed

Add new planning group.

- Group Name: sia5
- Kinematic Solver: KDL Kinematics Plugin
- Add Kin Chain: from `base_link` to `tool0`

Create a new pose, name it *home_pose*, and leave all the joint values unchanged.

Create a new end-effector with `tool0` as parent link.

Out of the remaining sections, you only need to fill in author information.

Generate the configuration package (name it: `motoman_sia5d_moveit_config`) in your catkin workspace (NB! Not inside the motoman repository, though). Close the Setup Assistant.

In order to test your newly generated package, run

```
$ roslaunch motoman_sia5d_moveit_config demo.launch
```

Play around with the robot in RViz to make sure that motion planning is working as expected. Test motion planning with different planners from OMPL.

To change the configuration of an existing MoveIt! package, simply type

```
$ roslaunch motoman_sia5d_moveit_config setup_assistant.launch
```

Add more saved poses to your configuration package.

Write a C++ node that moves the end effector of the SIA5D along a triangular path!

>>> Show the result to your lab instructor! <<<

Clean up your workspace

NB! Before you leave the lab, make sure you have pushed all the files in your catkin workspace to your git cloud service.

In terminal, `cd` to **<yourname>-rtech**

Type

```
git config user.email "youremail@example.com"
```

Type

```
git status
```

You should now see all the new and modified files in red.

Prepare the relevant files for the commit.

```
git add file_name_in_red1 file_name_in_red2
```

When you now type

```
git status
```

you should see all the added files in green. You are now ready to commit changes. Type

```
git commit –m “Insert a brief explanation”
```

Your changes have now been committed but not yet uploaded to the cloud. To upload your files, type

```
git push
```

In your web browser, **verify that all the files** have been uploaded to the **<yourname>-rtech** repository.

Delete the **<yourname>-rtech** folder and any other files you created from the lab's computer.