HAW HAMBURG
Fakultät TI, Department Informatik
Intelligente Sensorik
Prof. Dr. Tim Tiedemann, Max De Muirier

Übungsblatt 6
zur Veranstaltung „RobE – Einführung in die Robotik"

6. Trajektorienplanung eines Manipulators

In der Übung 6 geht es um die Trajektorienplanung des Manipulators Panda der Firma Franka Emika mittels des MoveIt!-Framework in ROS.

Zur Bearbeitung der Übungsaufgaben: Bearbeiten Sie auf jeden Fall *alle* Übungsaufgaben. Ausgenommen hiervon sind lediglich die mit „optional" gekennzeichneten Textstellen. Lesen Sie sich die Aufgaben gut durch. Sollten Sie eine Aufgabe nicht lösen können, so beschreiben Sie zumindest, wie weit Sie gekommen sind und auf welche Weise Sie vorgegangen sind. Aufgaben mit der Randbemerkung *"Protokoll!"* sind im Protokoll zu beantworten. Als Lösung der Aufgaben ist (je nach Aufgabe) Programmcode abzugeben (kommentiert) oder ein kurzer Text.

6.1. Vorbereitungsaufgaben

– keine –

6.2. Praktikumsaufgaben

1. Bearbeiten Sie die Schritte **Installation and setup of MoveIt!** (Anhang A).
2. Bearbeiten Sie die Schritte **First steps with MoveIt!** (Anhang B). Welche verschiedenen Bereiche gehören zum *MotionPlanning* und sind nun in *RVIZ* zu sehen? Überlegen Sie, warum diese Aufteilung notwendig ist. *Protokoll!*
3. Wählen Sie nun eine beliebige Zielstellung (*Goal state*) und planen Sie eine Trajektorie. Wählen Sie nun den Algorithmus *RRTstarkConfigDefault* und führen Sie eine weitere Planung aus. Wiederholen Sie das gleich mit *BFMtkConfigDefault*. Welcher Einfluss auf den Planungsprozess ist zu beobachten? *Protokoll!*
4. Wählen Sie nun die Option *Use Carthesian Path* aus und planen Sie eine neue Trajektorie Wie hat diese verändert? Visualisieren Sie die geplante Trajektorie in einer Schrittgröße (*step size*) von 10 und planen Sie die Trajektorie mit ein- und ausgeschalteter Option. Was fällt nun auf? *Protokoll!*
5. Bearbeiten Sie die Schritte **Planning trajectories the MoveIt!-commander** (Anhang C). Bewegen Sie nun den Endeffektor 20 cm nach oben. Was zeichnet die Bewegung bezüglich der inversen Kinematik aus? *Protokoll!*
6. Positionieren Sie den Endeffektor zwischen zwei Boxen in Bodennähe und setzen Sie das nächste Ziel zwischen die nächsten Boxen. Planen Sie nun die Trajektorie. Welches Verhalten wird beobachtet? *Protokoll!*
7. Bearbeiten Sie die Schritte **Pick-and-place motion planning** (Anhang D). Beschreiben Sie die verschiedenen Schritten des *Pick-and-Place*-Prozesses. *Protokoll!*

8. Öffnen Sie den Quellcode zu der *Pick-and-Place*-Demonstration. Beschreiben Sie Hauptfunktionen des Prozesses und ordnen Sie diesen den zuvor beschriebenen Schritten zu. *Protokoll!*

9. Ändern Sie nun die x-Koordinate des zu platzierenden Objekts in der *place*-Funktion zu 0.2 und bauen Sie den *catkin workspace* erneut. Was ist bei erneuter Ausführung der Demonstration zu beobachten? *Protokoll!*

A. Installation and setup of MoveIt!

The detailed installation process can be found here:

`https://ros-planning.github.io/moveit_tutorials/doc/getting_started/getting_started.html`

Please download the zipped package *moveit_ws.zip* to your *home* directory and unpack. It can be found in **Teams**. This will be your workspace folder.

```
1 echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
2 echo "export PATH=/usr/lib/ccache:$PATH" >> ~/.bashrc
3 source ~/.bashrc
```

Finally, build the *source* folder with:

```
4 sudo apt-get update
5 sudo apt-get install ros-noetic-moveit
  ros-noetic-panda-moveit-config ros-noetic-moveit-visual-tools
6 mkdir -p moveit_ws/src
7 cd moveit_ws/src
8 git clone https://github.com/ros-planning/moveit_tutorials
9 git clone https://github.com/ros-planning/
10 rosdep update
11 rosdep install --from-paths . --ignore-src --rosdistro noetic -y
12 catkin build
13 echo "source ~/moveit_ws/devel/setup.bash" >> ~/.bashrc
```

B. First steps with MoveIt!

Let's start with taking a look into a simple demonstration in *RVIZ*.

Launch the simple demo:

```
14 source  /.bashrc
15 roslaunch panda_moveit_config demo.launch rviz_tutorial:=true
```

To visualize the motion planning, add the *MotionPlanning*-view via the button *Add*. On the bottom left you will have a new window. Choose the tab *Planning* and select *panda_arm* as *planning group* in order to manually move the end-effectors target position. Press *Plan* to start the motion planning. After that, select *Plan and Execute*.

There are plenty of path planning algorithms. You can find them under the tab *Context*.

You can visualize every step of the planned path by going to *Planned path* in *Motion-Planning* in *Displays* and select the option *Show Trail*. By modifying *Trail Step Size* you can skip a specified number of steps.

C. Planning trajectories the MoveIt!-commander

MoveIt! comes with an C++ and Python Interface that provides a set of functions for motion control. A Python-based *command line tool* allows to access these function in a very simple way.

To begin, the MoveIt! Panda demonstration has to be launch:

```
16 roslaunch panda_moveit_config demo.launch
```

The *MoveIt!-Commander* can be launch with:

```
17 rosrun moveit_commander moveit_commander_cmdline.py
```

To view all the functionalities, type *help*.

The *commander* has to be connected to the right *move group node*, in order to send motion commands. This can be done by typing:

```
18 use panda_arm
```

You can now move the end-effector by sending commands.Try:

```
19 go <up, down, left, right> <distance in meters>
```

The current state can be check with

```
20 current
```

and saved to a variable

```
21 rec <state name>
```

A trajectory can then be planned and executed to that saved state in one step with

```
22 go <state name>
```

or two steps

```
23 plan <state name>
24 execute
```

Furthermore, a numbered list of commands can be saved to a file and then loaded within the *commander*.

Let's make a command list. Go to:

```
25 cd ~/moveit_ws/src
26 gedit command_list
```

and save this example:

```
use panda_arm
go up 0.3
rec state1
go rotate 0 0 1
go up 0.4
rotate 1 0 0
go state1
exit
```

Save this file as *command_list* in your workspace source folder. To execute the command list, open the *commander* and type:

```
27 load command_list
```

We can load collision objects into the planning scene:

```
28 rosrun moveit_tutorials collision_scene_example.py cluttered
```

These objects are taken into account when planning a path to avoid collision.

D. Pick-and-place motion planning

Now we are going to play through the *Pick-and-Place*-demonstration and take a look into its process.

Begin as usual:

```
29 source ~/moveit_ws/devel/setup.bash
30 roslaunch panda_moveit_config demo.launch
```

Start the *Pick-and-Place*-demonstration:

```
30 rosrun moveit_tutorials pick_place_tutorial
```

The demonstration will automatically start.

The source code to this demonstration can be found under:

```
31 ~/moveit_ws/src/moveit_tutorials/doc/pick_place/src
32 gedit pick_place_tutorial.cpp
```

For example, the target position and pose of the placed object can be modified within the *place*-function. Also, collision objects can be added or modified in the *addCollisionObjects*-function. For that, the number of objects has to be changed and the object's name, dimension, pose and primitive have to be added. To grasp the

newly added object, the function *pick* and *place* has to be adapted to it as wells.

After changes have been made, the *catkin workspace* has to be rebuild, using:

```
33 cd ~/moveit_ws
34 catkin build
```