Robotic Manipulation Course

Exercise 3

February 6, 2024

1 Instruction for getting, compiling, and submitting the code.

1.1 Getting the code

- Step 1: Fork the exercise from main repository to your subgroup.
- Step 2: Clone the exercise from your subgroup to your PC with commands below:

```
cd /home/ros/src
```

```
git clone <the-link-you-found-when-you-click-clone-button>
```

1.2 Compiling the code

- Note that you need to be in the "ros" folder to be able to compile the code. Now you can compile the code with command:

```
cd /home/ros/
```

```
catkin_make
```

- Check if there are any error during the compilation. If not, feel free to proceed.
- Note that whenever you **open a new terminal, or done compiling the code**, run the command below so that ROS is "activated" in the terminals:

```
source devel/setup.bash
```

Aalto University
School of Electrical
Engineering

ELEC-E8126
Intelligent Robotics Group
aalto.fi, irobotics.aalto.fi

1.3 Submiting the code

Says if you finish your code and want to submit it, you need to push everything to your git. You can do that using the commands below:

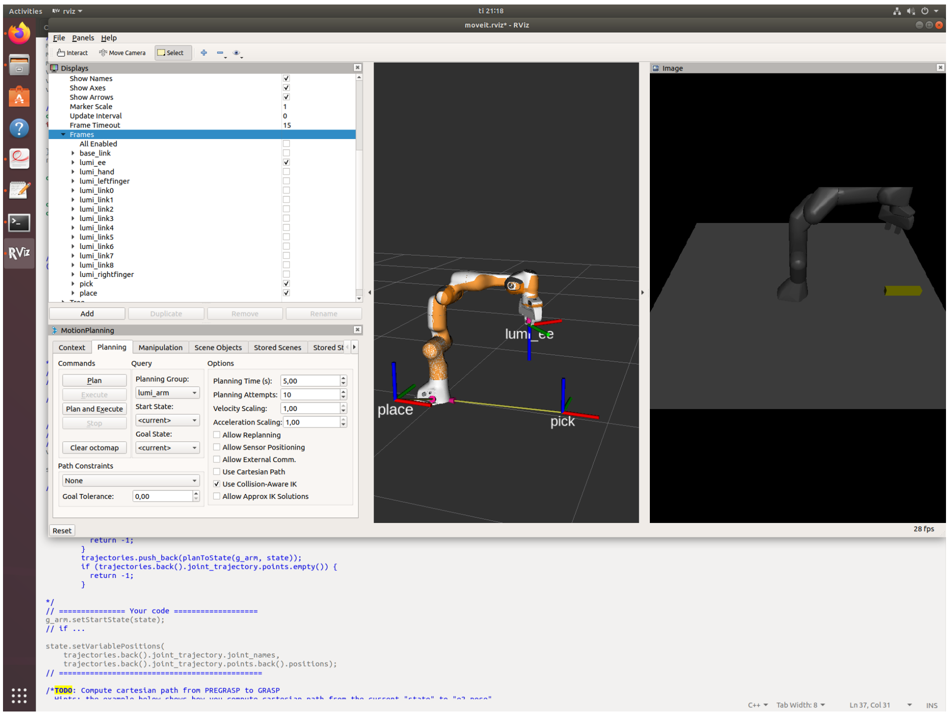
-
  ```
  git add .

  git commit −m "Your␣note␣about␣the␣changes␣in␣your␣code"
  ```

-
  ```
  git push −u origin master
  ```
- Go to your subgroup and check whether you see your lastest commit there.

2 Assignment (10 points)

The goal is to perform *pick and place* task with *lumi* robot in the simulated environment. We created a template in the course gitlab group which contains package configuration, and example of node which moves the robot. Your goal is to modify that node to perform following steps:

- from tf server, read the transformation of the frames named *pick* and *place*:
  - *pick* is the pose of the object to pick
  - *place* is the pose where to put the object



Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
All Enabled	□
base_link	□
lumi_ee	✓
lumi_hand	□
lumi_leftfinger	□
lumi_link0	□
lumi_link1	□
lumi_link2	□
lumi_link3	□
lumi_link4	□
lumi_link5	□
lumi_link6	□
lumi_link7	□
lumi_link8	□
lumi_rightfinger	□
pick	✓
place	✓

Planning Group: lumi_arm

Start State: <current>

Goal State: <current>

Path Constraints: None

Goal Tolerance: 0,00

Planning Time (s): 5,00

Planning Attempts: 10

Velocity Scaling: 1,00

Acceleration Scaling: 1,00

□ Allow Replanning

□ Allow Sensor Positioning

□ Allow External Comm.

□ Use Cartesian Path

☑ Use Collision-Aware IK

□ Allow Approx IK Solutions

```
      return -1;
    }
    trajectories.push_back(planToState(g_arm, state));
    if (trajectories.back().joint_trajectory.points.empty()) {
      return -1;
    }

*/
// ============== Your code ==================
g_arm.setStartState(state);
// if ...

state.setVariablePositions(
    trajectories.back().joint_trajectory.joint_names,
    trajectories.back().joint_trajectory.points.back().positions);
// ==========================================

/*TODO: Compute cartesian path from PREGRASP to GRASP
  Hints: the example below shows how you compute cartesian path from the current "state" to "o2 pose"
```

C++ Tab Width: 8 Ln 37, Col 31 INS

Figure 1: The frames. To show the frames: Click Add button -> By Display Type -> Frame.

- compute and visualise following poses of the gripper:
  - pre-grasp pose (10 cm above the pick pose in z-axis of pick coordinate system)
  - grasp pose
  - place pose
- compute and visualise following trajectories:
  - plan between current pose and pre-grasp pose
  - cartesian path from pre-grasp pose to grasp pose
  - cartesian path from grasp pose to pre-grasp pose
  - plan to place pose
- execute visualised trajectories with appropriate gripper opening/closing

To test your code you need to launch the simulation and frame publishers by:

```
roslaunch exercise3 sim_with_box.launch
```

```
roslaunch exercise3 publish_frames.launch
```

and than run your node with

```
rosrun exercise3 pick_and_place
```

The same commands will be used for testing by TA. Therefore, do not change the package/node names.

3 Report (10 points)

In addition to code, you are supposed to write a technical report (pdf) in which you will document the steps performed to fulfill the assignment. Your report should contains:

- your name, student number, date, exercise number and course name
- **(2 points)** the equations which were used to compute the poses (use math instead of code), you can use following math symbols:
  - $T_x(\cdot), T_y(\cdot), T_z(\cdot)$ - 4x4 translation matrices (rotation is identity)
  - $R_x(\cdot), R_y(\cdot), R_z(\cdot)$ - 4x4 rotation matrices (translation is zero)
  - e.g. $T = T_z(0.25)R_z(\pi)$ will translate frame by 0.25 m and then rotate by π rad
- answers to following questions:
  - **(2 points)** In which coordinate frame the MoveIt assumes the poses are specified and what should I do if my pose is specified with respect to another frame?
  - **(2 points)** What are the differences between cartesian path computation and planning?
  - **(2 points)** Is there any chance that the object will be moved by robot before grasping? Why yes/no?
  - **(2 points)** Can robot collide with itself during execution of computed pick-and-place path? Why/why not?
- estimates of time spent on this exercise

4 Submission

To submit your code and report, fork a repository named *robotic_manipulation_2024/exercise3* to your gitlab subgroup. Modify the code in the forked repository. The solution needs to be pushed to your repository in your own subgroup. **Be sure to push your code before the assignment deadline. Commits pushed after the deadline will be subjected to penalties according to the course's practicality**.

5 Deadline

Deadline for this assignment is 20th of February at 23:59.

6 Resources

- MoveIt `https://moveit.ros.org`
- ROS `https://www.ros.org`