CS 6341 Robotics Homework 5

Professor Yu Xiang

November 17, 2025

In this homework, write down your solutions for problems 1 and finish the coding problem 2. Upload your solutions and code to eLearning. Our TA will check your solutions and run your scripts to verify them.

Problem 1

(5 points)

Robot Control. Exercise 11.6 in Lynch and Park, Modern Robotics.

Develop a controller for a one-dof mass-spring-damper system of the form $m\ddot{x} + b\dot{x} + kx = f$, where f is the control force and $m = 4$ kg, $b = 2$ Ns/m, and $k = 0.1$ N/m.

(a) Choose a P controller $f = K_p x_e$, where $x_e = x_d - x$ is the position error and $x_d = 0$. What value of K_p yields critical damping?

(Hint) Critically damped: damping ratio $\zeta = 1$.

(b) Choose a D controller $f = K_d \dot{x}_e$, where $x_d = 0$. What value of K_d yields critical damping?

(c) Choose a PD controller that yields critical damping. What is the relationship between K_p and K_d?

(d) For the PD controller above, if $x_d = 1$ and $\dot{x}_d = \ddot{x}_d = 0$, what is the steady-state error $x_e(t)$ as t goes to infinity? What is the steady-state control force?

(e) Now insert a PID controller for f. Assume $x_d \neq 0$ and $\dot{x}_d = \ddot{x}_d = 0$. Write the error dynamics in terms of $\ddot{x}_e$, $\dot{x}_e$, x_e, and $\int x_e(t)dt$ on the left-hand side and a constant forcing term on the right-hand side.

(Hint) You can write kx as $-k(x_d - x) + kx_d$.

Problem 2

(5 points)

ROS2 programming, grasping.

In this problem, you will use grasp planning and motion planning to enable a robot to grasp a cracker box in ROS2. **You can directly use Ubuntu, or Docker or virtual machine to install ROS2 according to your own set up**. Refer to the ROS2 Jazzy page if needed:

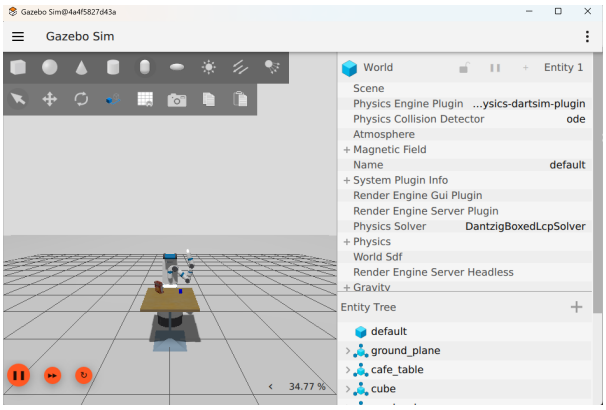
`https://docs.ros.org/en/jazzy/`.

For the following steps, start with your ROS2 workspace from the previous homework.

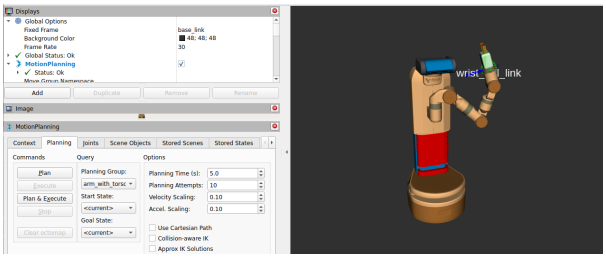
(2.1) I made some changes to the following repositories again. **Please update your code and then rebuild your workspace.** You can either git clone the new code, or do a git pull for each repo.

- Git clone the source code to the src folder of your ROS workspace (use the master branch): git clone `https://github.com/IRVLUTD/panda_gz_moveit2.git`
- Git clone the source code to the src folder of your ROS workspace (use the main branch): git clone `https://github.com/IRVLUTD/pymoveit2`

(2.2) Launch Fetch Gazebo Simulator by following the steps in Homework 2. Use terminator to start multiple windows. Make sure that you do not see any error from the terminal; sometimes some controllers may not start correctly, so you can restart it.



Gazebo sim interface



Rviz interface

Figure 1: Two windows after launching the Fetch Gazebo simulation

- Use the following command to add the model path to Gazebo:

  `export GZ_SIM_RESOURCE_PATH=$GZ_SIM_RESOURCE_PATH:`

  `$(ros2 pkg prefix fetch_moveit_config)/share:$(ros2 pkg prefix fetch_description)/share`

- Use one terminator window. Launch the Fetch Gazebo simulation by

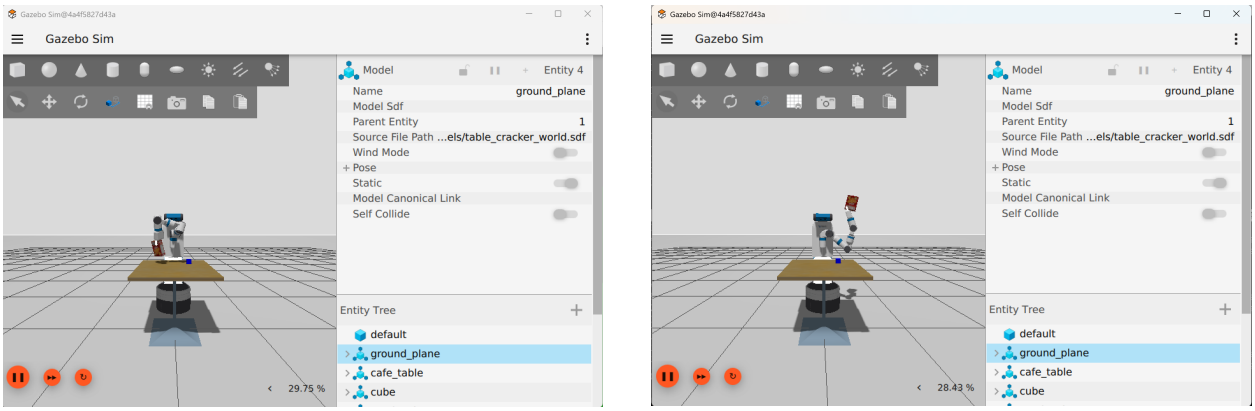
```
ros2 launch panda fetch.launch.py world:=table_cracker_world.sdf
```

- The following command can be used to change the viewpoint of the camera in Gazebo:

```
gz service -s /gui/move_to/pose --reqtype gz.msgs.GUICamera --reptype gz.msgs.Boolean
--timeout 5000 --req 'pose:{position:{x:3,y:0,z:2} orientation:{x:0,y:0,z:1.0,w:0.0}}'
```

- To verify your setup is correct, you can use the motion planning panel in Rviz to try some planning tasks.

You shall see the Gazebo environment as in Figure 1 with a table and a cracker box.



Lift the cracker box Move to the initial configuration

Figure 2: Two windows after grasping the cracker box

(2.3) **Coding assignment.** Up to now, you shall have the Gazebo and Moveit running. In this coding assignment, you need to use moveit to plan a trajectory to enable Fetch to grasp a cracker box on a table.

Read the code in grasp_cracker.py. The code loads 50 planned grasps of the cracker box from GraspIt! and then use Moveit to try which grasp can be used to grasp the object. Finally, it executes the planned trajectory for grasping.

Finish the implementation of the TODOs in the python script. You need to finish the function plan_grasp(). Then you can run the python script. Figure 2 shows a Gazebo scene of running the script.

Submission guideline: Upload your implemented grasp_cracker.py file and the screen capture of Gazebo after running the script to eLearning.